CHAPTER 7

OPERATIONS SCHEDULING

Scheduling is an important aspect of operations control in both manufacturing and service industries. With increased emphasis on time to market and time to volume as well as improved customer satisfaction, efficient scheduling will gain increasing emphasis in the operations function in the coming years.

In some sense, much of what has been discussed so far in this text can be considered a subset of production scheduling. Aggregate planning, treated in Chapter 3, is aimed at macroscheduling of workforce levels and overall production levels for the firm. Detailed inventory control, discussed in Chapters 4 and 5, concerns methods of scheduling production at the individual item level, and MRP, discussed in Chapter 6, provides production schedules for end items and subassemblies in the product structure.

There are many different types of scheduling problems faced by the firm. A partial list includes

1. *Job shop scheduling.* Job shop scheduling, known more commonly in practice as shop floor control, is the set of activities in the shop that transforms inputs (a set of requirements) to outputs (products to meet those requirements). Much of this chapter will be concerned with sequencing issues on the shop floor, and more will be said about this problem in the next section.
2. *Personnel scheduling.* Scheduling personnel is an important problem for both manufacturing and service industries. Although shift scheduling on the factory floor may be considered one of the functions of shop floor control, personnel scheduling is a much larger problem. Scheduling health professionals in hospitals and other health facilities is one example. Determining whether to meet peak demand with overtime shifts, night shifts, or subcontracting is another example of a personnel scheduling problem.
3. *Facilities scheduling.* This problem is particularly important when facilities become a bottleneck resource. Scheduling operating rooms at hospitals is one example. As the need for health care increases, some hospitals and HMOs

(health maintenance organizations) find that facilities are strained. A similar problem occurs in colleges and universities in which enrollments have increased without commensurate increases in the size of the physical plant.

4. *Vehicle scheduling.* Manufacturing firms must distribute their products in a cost-efficient and timely manner. Some service operations, such as dial-a-ride systems, involve pick-ups and deliveries of goods and/or people. Vehicle routing is a problem that arises in many contexts. Problems such as scheduling snow removal equipment, postal and bank deliveries, and shipments to customers with varying requirements at different locations are some examples. Vehicle scheduling is discussed in Section 10 of this chapter.
5. *Vendor scheduling.* For firms with just-in-time (JIT) systems, scheduling deliveries from vendors is an important logistics issue. Purchasing must be coordinated with the entire product delivery system to ensure that JIT production systems function efficiently. Vollman et al. (1992, p. 191) discuss the application of vendor scheduling to the JIT system at Steelcase. (JIT is discussed in Chapters 1 and 6 of this book.)
6. *Project scheduling.* A project may be broken down into a set of interrelated tasks. Although some tasks can be done concurrently, many tasks cannot be started until others are completed. Complex projects may involve thousands of individual tasks that must be coordinated for the project to be completed on time and within budget. Project scheduling is an important component of the planning function, which we treat in detail in the next chapter.
7. *Dynamic versus static scheduling.* Most scheduling theory that we review in this chapter views the scheduling problem as a static one. Numerous jobs arrive simultaneously to be processed on a set of machines. In practice, many scheduling problems are dynamic in the sense that jobs arrive continuously over time. One example is the problem faced by an air traffic controller who must schedule runways for arriving planes. The problem is a dynamic one in that planes arrive randomly and runways are freed up and committed randomly over time. Dynamic scheduling problems, treated in Section 9 of this chapter, are analyzed using the tools of queuing theory (discussed in detail in Supplement 2 at the end of this chapter).

Scheduling is a complex but extremely important operations function. The purpose of this chapter is to give the reader a flavor for the kinds of results one can obtain using analytical models, and how these models can be used to solve certain classes of scheduling problems. Our focus is primarily on job shop scheduling, but we consider several other scheduling problems as well.

7.1 Production Scheduling and the Hierarchy of Production Decisions

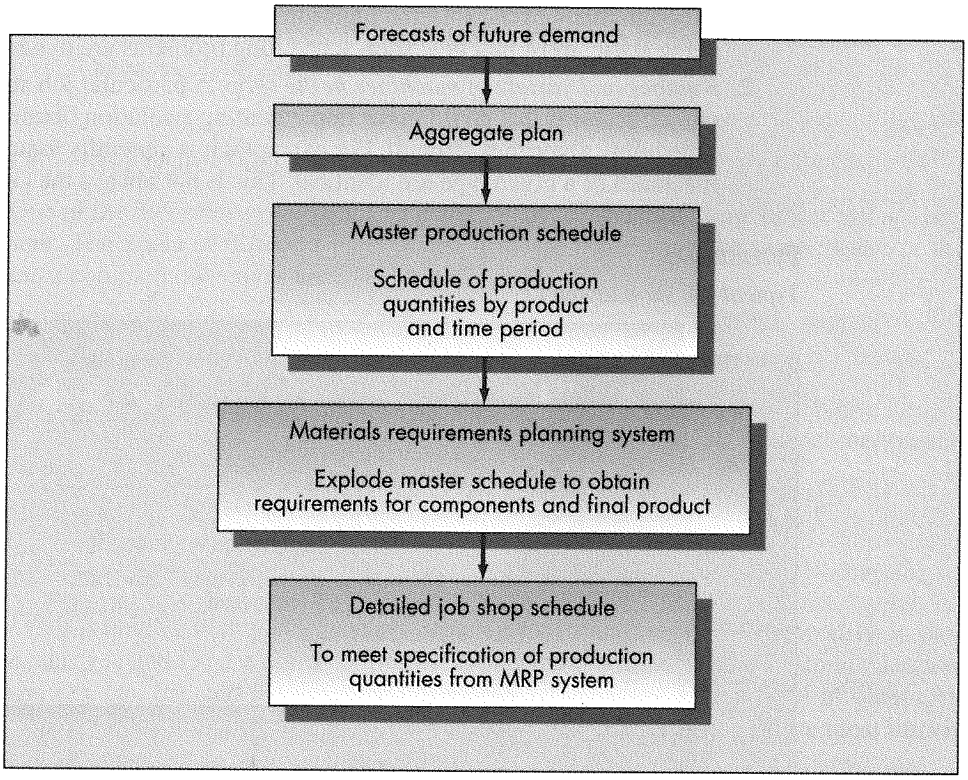
Crucial to controlling production operations is the detailed scheduling of various aspects of the production function. We may view the production function in a company as a hierarchical process. First, the firm must forecast demand for aggregate sales over some predetermined planning horizon. These forecasts provide the input for determining the aggregate production and workforce levels for the planning horizon. Aggregate planning techniques are discussed in Chapter 3. The aggregate production plan must then be translated into the master production schedule (MPS). The MPS results in specific production goals by product and time period.

Materials requirements planning (MRP), treated in detail in Chapter 6, is one method for meeting specific production goals of finished-goods inventory generated by the MPS. The MRP system "explodes" the production levels one obtains from the MPS analysis back in time to obtain production targets at each level of assembly by time period. The result of the MRP analysis is specific planned order releases for final products, subassemblies, and components.

Finally, the planned order releases must be translated into a set of tasks and the due dates associated with those tasks. This level of detailed planning results in the shop floor schedule. Because the MRP or other lot scheduling system usually recommends revisions in the planned order releases, shop floor schedules change frequently. The hierarchy of production planning decisions is shown schematically in Figure 7–1.

Shop floor control means scheduling personnel and equipment in a work center to meet the due dates for a collection of jobs. Often, jobs must be processed through the

FIGURE 7–1
Hierarchy of production decisions


Forecasts of future demand
Aggregate plan
Master production schedule
Schedule of production quantities by product and time period
Materials requirements planning system
Explode master schedule to obtain requirements for components and final product
Detailed job shop schedule
To meet specification of production quantities from MRP system

machines in the work center in a unique order or sequence. Figure 7–2 shows the layout of a typical job shop.

Both jobs and machines are treated as indivisible. Jobs must wait, or queue up, for processing when machines are busy. This is referred to as discrete processing. Production scheduling in continuous-process industries, such as sugar or oil refining, has a very different character.

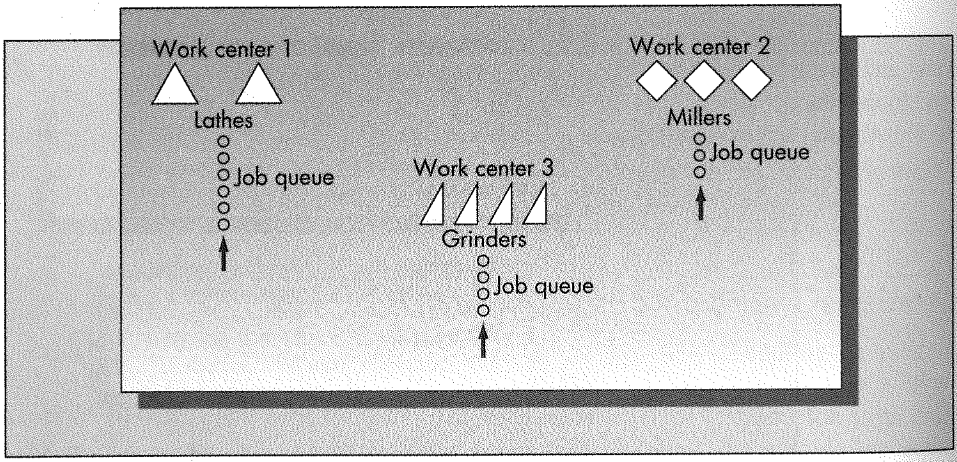
Although there are many problems associated with operations scheduling, our concern in this chapter will be job sequencing. Given a collection of jobs remaining to be processed on a collection of machines, the problem is how to sequence these jobs to optimize some specified criterion. Properly choosing the sequencing rule can effect dramatic improvements in the throughput rate of the job shop.

7.2 Important Characteristics of Job Shop Scheduling Problems

Significant issues for determining optimal or approximately optimal scheduling rules are the following:

1. *The job arrival pattern.* We often view the job shop problem as a static one in which we take a "snapshot" of the system at a point in time and proceed to solve the problem based on the value of the current state. However, the number of jobs waiting to be processed is constantly changing. Hence, although many of the solution algorithms we consider view the problem as being static, most practical shop scheduling problems are dynamic in nature.
2. *Number and variety of machines in the shop.* A particular job shop may have unique features that could make implementing a solution obtained from a scheduling algorithm difficult. For example, it is generally assumed that all machines of a given type are identical. This is not always the case, however.

FIGURE 7–2
Typical job shop layout



The throughput rate of a particular machine could depend upon a variety of factors, such as the condition of the machine or the skill of the operator. Depending on the layout of the shop and the nature of the jobs, constraints might exist that would make solutions obtained from an "all purpose" procedure infeasible.

3. *Number of workers in the shop.* Both the number of workers in the shop and the number and variety of machines in the shop determine the shop's capacity. Capacity planning is an important aspect of production planning. Many control systems, such as traditional MRP discussed in Chapter 6, do not explicitly incorporate capacity considerations. Furthermore, capacity is dynamic. A breakdown of a single critical machine or the loss of a critical employee could result in a bottleneck and a reduction in the shop's capacity.
4. *Particular flow patterns.* The solutions obtained from the scheduling algorithms to be presented in this chapter require that jobs be completed in a fixed order. However, each sequence of jobs through machines results in a pattern of flow of materials through the system. Because materials-handling issues are often treated separately from scheduling issues, infeasible flow patterns may result.
5. *Evaluation of alternative rules.* The choice of objective will determine the suitability and effectiveness of a sequencing rule. It is common for more than one objective to be important, so that it may be impossible to determine a unique optimal rule. For example, one may wish to minimize the time required to complete all jobs, but also may wish to limit the maximum lateness of any single job.

Objectives of Job Shop Management

One of the difficulties of scheduling is that many, often conflicting, objectives are present. The goals of different parts of the firm are not always the same. Some of the most common objectives are

1. Meet due dates.
2. Minimize work-in-process (WIP) inventory.
3. Minimize the average flow time through the system.
4. Provide for high machine/worker time utilization. (Minimize machine/worker idle time.)
5. Provide for accurate job status information.
6. Reduce setup times.
7. Minimize production and worker costs.

It is obviously impossible to optimize all seven objectives simultaneously. In particular, (1) and (3) are aimed primarily at providing a high level of customer service, and (2), (4), (6), and (7) are aimed primarily at providing a high level of plant efficiency. Determining the trade-off between cost and quality is one of the most important strategic issues facing a firm today.

Some of these objectives conflict. If the primary objective is to reduce work-in-process inventory (as, for example, with just-in-time inventory control systems, discussed in Chapter 6), it is likely that worker idle time would increase. As the system tightens up by reducing the inventory within and between manufacturing operations, differences in the throughput rate from one part of the system to another may force the faster operations to wait. Although not recommended by those espousing the just-in-time philosophy, buffer inventories between operations can significantly reduce idle time.

As an example, consider the simple system composed of two operations in series, pictured in Figure 7–3. If work-in-process inventory is zero, then the throughput of the system at any point in time is governed by the smaller of the throughputs of the two operations. If operation 1 is temporarily halted by a machine failure, then operation 2 must also remain idle. However, if there is a buffer inventory placed between the operations, then 2 can continue to operate while 1 is undergoing repair or recalibration.

Finding the proper mix between WIP inventory and worker idle time is equivalent to choosing a point on the trade-off curve of these conflicting objectives. (Trade-off, or exchange, curves were discussed in Chapter 5 in the context of multi-item inventory control.) Such a curve is pictured in Figure 7–4*a*. A movement from one point to another along such a curve does not necessarily imply that the system has improved, but rather that different weights are being applied to the two objectives. A true improvement in the overall system would mean that the entire trade-off curve undergoes a downward shift, such as that pictured in Figure 7–4*b*.

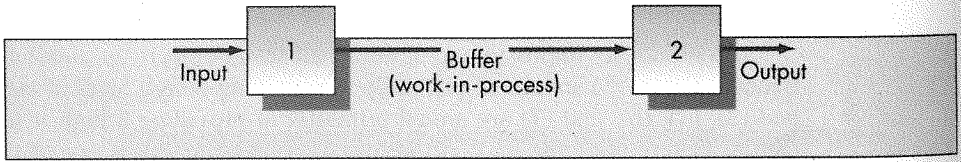
7.3 Job Shop Scheduling Terminology

In general, a job shop scheduling problem is one in which n jobs must be processed through m machines. The complexity of the problem depends upon a variety of factors, such as what job sequences are permissible and what optimization criteria are chosen. In this section we define some of the terms that will be used throughout this chapter.

1. *Flow shop.* In a flow shop each of the n jobs must be processed through the m machines in the same order, and each job is processed exactly once on each machine. This is what we typically think of as an assembly line.

FIGURE 7–3

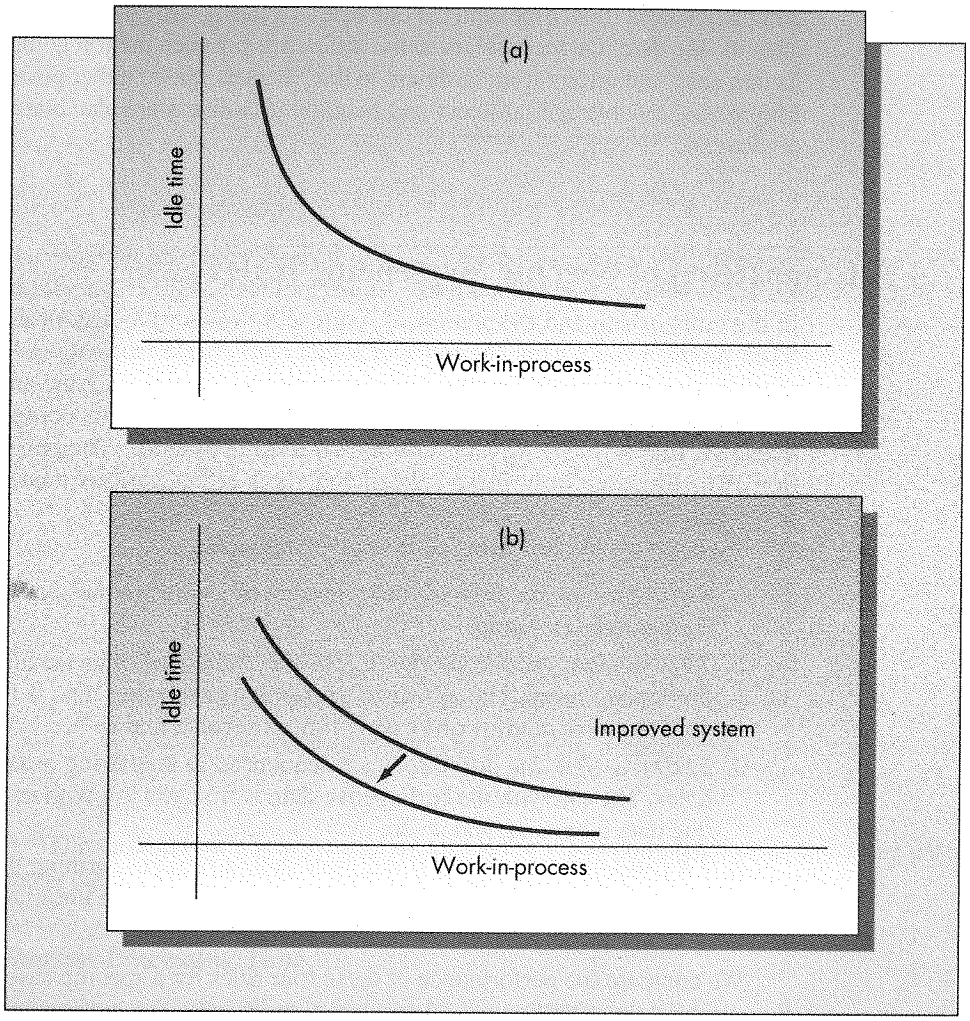
A process composed of two operations in series



2. *Job shop.* A general job shop differs from a flow shop in that not all jobs are assumed to require exactly m operations, and some jobs may require multiple operations on a single machine. Furthermore, in a job shop each job may have a different required sequencing of operations. General job shop problems are extremely complex. All-purpose solution algorithms for solving general job shop problems do not exist.

3. *Parallel processing versus sequential processing.* Most of the problems that we will consider involve sequential processing. This means that the m machines are distinguishable, and different operations are performed by different machines. In parallel

FIGURE 7–4
Conflicting objectives in job shop management



processing we assume that the machines are identical, and any job can be processed on any machine. An example of parallel processing occurs in a phone switching center, in which calls are processed through the next available server. Parallel processing is discussed in the context of stochastic scheduling in Section 8.

4. *Flow time.* The flow time of job i is the time that elapses from the initiation of the first job on the first machine to the completion of job i. Equivalently, it is the amount of time that job i spends in the system. The *mean flow time,* which is a common measure of system performance, is the arithmetic average of the flow times for all n jobs.

5. *Makespan.* The makespan is the flow time of the job that is completed last. It is also the time required to complete all n jobs. Minimizing the makespan is a common objective in multiple-machine sequencing problems.

6. *Tardiness and lateness.* Tardiness is the positive difference between the completion time (flow time) and the due date of a job. A tardy job is one that is completed after its due date. Lateness refers to the difference between the job completion time and its due date, and differs from tardiness in that lateness can be either positive or negative. Minimizing the average tardiness and maximum tardiness are also common scheduling objectives.

7.4 A Comparison of Specific Sequencing Rules

In the comparison and evaluation of sequencing rules we consider the job shop at a fixed point in time. This section will focus on a single machine only. Assume that there is a collection of jobs that must be processed on the machine and that each job has associated with it a processing time and a due date. We compare the performance of four sequencing rules commonly used in practice. The purpose of this section is to illustrate how these sequencing rules affect various measures of system performance.

We compare the following four sequencing rules:

1. *FCFS (first-come, first-served).* Jobs are processed in the sequence in which they entered the shop.
2. *SPT (shortest processing time).* Jobs are sequenced in increasing order of their processing times. The job with the shortest processing time is first, the job with the next shortest processing time is second, and so on.
3. *EDD (earliest due date).* Jobs are sequenced in increasing order of their due dates. The job with the earliest due date is first, the job with the next earliest due date is second, and so on.
4. *CR (critical ratio).* Critical ratio scheduling requires forming the ratio of the processing time of a job divided by the remaining time until the due date, and scheduling the job with the largest ratio next.

We compare the performance of these four rules for a specific case based on mean flow time, average tardiness, and number of tardy jobs. The purpose of the next example is to help the reader develop an intuition for the mechanics of scheduling before presenting formal results.

Example 7.1

A machining center in a job shop for a local fabrication company has five unprocessed jobs remaining at a particular point in time. The jobs are labeled 1, 2, 3, 4, and 5 in the order that they entered the shop. The respective processing times and due dates are given in the table below.

Job Number	*Processing Time*	*Due Date*
1	11	61
2	29	45
3	31	31
4	1	33
5	2	32

■

First-Come, First-Served

Because the jobs are assumed to have entered the shop in the sequence that they are numbered, FCFS scheduling means that the jobs are scheduled in the order 1, 2, 3, 4, 5. This results in

Sequence	*Completion Time*	*Due Date*	*Tardiness*
1	11	61	0
2	40	45	0
3	71	31	40
4	72	33	39
5	74	32	42
Totals	268		121

$$\text{Mean flow time} = 268/5 = 53.6,$$

$$\text{Average tardiness} = 121/5 = 24.2,$$

$$\text{Number of tardy jobs} = 3.$$

The tardiness of a job is equal to zero if the job is completed prior to its due date and is equal to the number of days late if the job is completed after its due date.

Shortest Processing Time

Here jobs are sequenced in order of increasing processing time.

Job	*Processing Time*	*Completion Time*	*Due Date*	*Tardiness*
4	1	1	33	0
5	2	3	32	0
1	11	14	61	0
2	29	43	45	0
3	31	74	31	43
Totals		135		43

$$\text{Mean flow time} = 135/5 = 27.0,$$
$$\text{Average tardiness} = 43/5 = 8.6,$$
$$\text{Number of tardy jobs} = 1.$$

Earliest Due Date

Here jobs are completed in the order of their due dates.

Job	*Processing Time*	*Completion Time*	*Due Date*	*Tardiness*
3	31	31	31	0
5	2	33	32	1
4	1	34	33	1
2	29	63	45	18
1	11	74	61	13
Totals		235		33

$$\text{Mean flow time} = 235/5 = 47.0,$$
$$\text{Average tardiness} = 33/5 = 6.6,$$
$$\text{Number of tardy jobs} = 4.$$

Critical Ratio Scheduling

After each job has been processed, we compute

$$\frac{\text{Due date} - \text{Current time}}{\text{Processing time}}$$

known as the critical ratio and schedule the next job in order to minimize the value of the critical ratio. The idea behind critical ratio scheduling is to provide a balance between SPT, which only considers processing time, and EDD, which only considers due dates. The ratio will grow smaller as the current time approaches the due date, and more priority will be given to those jobs with longer processing times. One disadvantage of the method is that the critical ratios need to be recalculated each time a job is scheduled.

It is possible that the numerator will be negative for some or all of the remaining jobs. When that occurs it means that the job is late, and we will assume that late jobs are automatically scheduled next. If there is more than one late job, then the late jobs are scheduled in SPT sequence.

First we compute the critical ratios starting at time $t = 0$.

Current time: $t = 0$

Job	*Processing Time*	*Due Date*	*Critical Ratio*
1	11	61	61/11 (5.545)
2	29	45	45/29 (1.552)
3	31	31	31/31 (1.000)
4	1	33	33/1 (33.00)
5	2	32	32/2 (16.00)

The minimum value corresponds to job 3, so job 3 is performed first. As job 3 requires 31 units of time to process, we must update all of the critical ratios in order to determine the next job to process. We move the clock to time $t = 31$ and recompute the critical ratios.

Current time: $t = 31$

Job	*Processing Time*	*Due Date − Current Time*	*Critical Ratio*
1	11	30	30/11 (2.727)
2	29	14	14/29 (0.483)
4	1	2	2/1 (2.000)
5	2	1	1/2 (0.500)

The minimum is 0.483, which corresponds to job 2. Hence, job 2 is scheduled next. Since job 2 has processing time of 29, we update the clock to time $t = 31 + 29 = 60$.

Current Time: $t = 60$

Job	*Processing Time*	*Due Date − Current Time*	*Critical Ratio*
1	11	1	1/11 (.0909)
4	1	−27	$-27/1 < 0$
5	2	−28	$-28/2 < 0$

Jobs 4 and 5 are now late so they are given priority and scheduled next. Since they are scheduled in SPT order, they are done in the sequence job 4, then job 5. Finally, job 1 is scheduled last.

Summary of the Results for Critical Ratio Scheduling

Job	*Processing Time*	*Completion Time*	*Tardiness*
3	31	31	0
2	29	60	15
4	1	61	28
5	2	63	31
1	11	74	13
Totals		289	87

$$\text{Mean flow time} = 289/5 = 57.8.$$

$$\text{Average tardiness} = 87/5 = 17.4.$$

$$\text{Number of tardy jobs} = 4.$$

We summarize the results of this section for all four scheduling rules:

Summary of the Results of Four Scheduling Rules

Rule	*Mean Flow Time*	*Average Tardiness*	*Number of Tardy Jobs*
FCFS	53.6	24.2	3
SPT	27.0	8.6	1
EDD	47.0	6.6	4
CR	57.8	17.4	4

7.5 Objectives in Job Shop Management: An Example

Example 7.2

An air traffic controller is faced with the problem of scheduling the landing of five aircraft. Based on the position and runway requirements of each plane, he estimates the following landing times:

Plane:	1	2	3	4	5
Time (in minutes):	26	11	19	16	23

Only one plane may land at a time. The problem is essentially the same as that of scheduling five jobs on a single machine, with the planes corresponding to the jobs, the landing times to the processing times, and the runway to the machine.

1. Two reasonable objectives given the information above would be to minimize the total time required to land all planes (i.e., the makespan), or the average time required to land the planes (the mean flow time). The makespan for any sequence is clearly 95 minutes, the sum of the landing times. However, as we saw in the previous

example, the mean flow time is not sequence independent. As with the previous example, the shortest-processing-time rule minimizes the mean flow time. We will show in the next section that SPT is the optimal sequencing rule for minimizing mean flow time for a single machine in general.

2. An alternative objective might be to land as many people as quickly as possible. In this case we would also need to know the number of passengers on each plane. Suppose that these numbers are as follows:

Plane	1	2	3	4	5
Landing time	26	11	19	16	23
Number of passengers	180	12	45	75	252

The appropriate objective in this case might be to minimize the *weighted* makespan or the weighted sum of the completion times, where the weights would correspond to the number of passengers in each plane. Notice that the objective function would now be in units of passenger-minutes.

3. An issue that we have not yet addressed is the time that each plane is scheduled to arrive. Assume the following data:

Plane	1	2	3	4	5
Landing time	26	11	19	16	23
Scheduled arrival time	5:30	5:45	5:15	6:00	5:40

Sequencing rules that ignore due dates could give very poor results in terms of meeting the arrival times. Some possible objectives related to due dates include minimizing the average tardiness or minimizing the maximum tardiness.

4. Thus far we have ignored special conditions that favor some planes over others. Suppose that plane number 4 has a critically low fuel level. This would probably result in plane 4 taking precedence. Priority constraints could arise in other ways as well: planes that are scheduled for continuing flights or planes carrying precious or perishable cargo might also be given priority. ■

The purpose of this section was to demonstrate the difficulties of choosing an objective function for job sequencing problems. The optimal sequence is highly sensitive to the choice of the objective, and the appropriate objective is not always obvious.

Problems for Sections 1–5

1. Discuss each of the objectives listed below and the relationship each has with job shop performance.
   a. Reduce WIP inventory.
   b. Provide a high level of customer service.

c. Reduce worker idle time.
d. Improve factory efficiency.

2. In Problem 1, why are (*a*) and (*c*) conflicting objectives, and why are (*b*) and (*d*) conflicting objectives?
3. Define the following terms:
   a. Flow shop.
   b. Job shop.
   c. Sequential versus parallel processing.
   d. Makespan.
   e. Tardiness.
4. Four trucks, 1, 2, 3, and 4, are waiting on a loading dock at XYZ Company that has only a single service bay. The trucks are labeled in the order that they arrived at the dock. Assume the current time is 1:00 P.M. The times required to unload each truck and the times that the goods they contain are due in the plant are given in the table below.

Truck	*Unloading Time*	*Time Material Is Due*
1	20 minutes	1:25 P.M.
2	14 minutes	1:45 P.M.
3	35 minutes	1:50 P.M.
4	10 minutes	1:30 P.M.

   Determine the schedules that result for each of the rules FCFS, SPT, EDD, and CR. In each case compute the mean flow time, average tardiness, and number of tardy jobs.
5. Five jobs must be scheduled for batch processing on a mainframe computer system. The processing times and the promised times for each of the jobs are listed below.

Job	1	2	3	4	5
Processing time	40 min.	2.5 hr.	20 min.	4 hr.	1.5 hr.
Promised time	11:00 A.M.	2:00 P.M.	2:00 P.M.	1:00 P.M.	4:00 P.M.

   Assume that the current time is 10:00 A.M.
   a. If the jobs are scheduled according to SPT, find the tardiness of each job and the mean tardiness of all jobs.
   b. Repeat the calculation in part (*a*) for EDD scheduling.

7.6 An Introduction to Sequencing Theory for a Single Machine

Assume that n jobs are to be processed through one machine. For each job i, define the following quantities:

t_i = Processing time for job i,
d_i = Due date for job i,
W_i = Waiting time for job i,
F_i = Flow time for job i,
L_i = Lateness of job i,
T_i = Tardiness of job i,
E_i = Earliness of job i.

The processing time and the due date are constants that are attached to the description of each job. The waiting time for a job is the amount of time that the job must wait before its processing can begin. For the cases that we consider, it is also the sum of the processing times for all of the preceding jobs. The flow time is simply the waiting time plus the job processing time ($F_i = W_i + t_i$). The flow time of job i and the completion time of job i are the same. We will define the lateness of job i as $L_i = F_i - d_i$, and assume that lateness can be either a positive or negative quantity. Tardiness is the positive part of lateness ($T_i = \max[L_i, 0]$), and earliness is the negative part of lateness ($E_i = \max[-L_i, 0]$).

Other related quantities are maximum tardiness $T_{\max}$, given by the formula

$$T_{\max} = \max\{T_1, T_2, \ldots, T_n\},$$

and the mean flow time F', given by the formula

$$F' = \frac{1}{n}\sum_{i=1}^{n} F_i.$$

As we are considering only a single machine, every schedule can be represented by a permutation (that is, ordering) of the integers 1, 2, . . . , n. There are exactly $n!$ different permutation schedules ($n! = n(n-1)\ldots(2)(1)$).

Shortest-Processing-Time Scheduling

We have the following result:

Theorem 7.1

The scheduling rule that minimizes the mean flow time F' is SPT.

Theorem 7.1 is easy to prove. Let [1], [2], . . . , [n] be any permutation of the integers 1, 2, 3, . . . , n. The flow time of the job that is scheduled in position k is given by

$$F_{[k]} = \sum_{i=1}^{k} t_{[i]}.$$

It follows that the mean flow time is given by

$$F' = \frac{1}{n}\sum_{k=1}^{n} F_{[k]} = \frac{1}{n}\sum_{k=1}^{n}\sum_{i=1}^{k} t_{[i]}.$$

The double summation term may be written in a different form. Expanding the double summation, we obtain

$$\begin{aligned} k &= 1{:}\ t_{[1]} \\ k &= 2{:}\ t_{[1]} + t_{[2]} \\ &\ \vdots \\ k &= n{:}\ t_{[1]} + t_{[2]} + \ldots + t_{[n]}. \end{aligned}$$

By summing down the column rather than across the row, we may rewrite F' in the form

$$nt_{[1]} + (n-1)t_{[2]} + \ldots + t_{[n]},$$

which is clearly minimized by setting

$$t_{[1]} \le t_{[2]} \le \ldots \le t_{[n]},$$

which is exactly the SPT sequencing rule.

We have the following corollary to Theorem 7.1.

Corollary 7.1

The following measures are equivalent:

1. **Mean flow time.**
2. **Mean waiting time.**
3. **Mean lateness.**

Taken together, Corollary 7.1 and Theorem 7.1 establish that SPT minimizes mean flow time, mean waiting time, and mean lateness for single-machine sequencing.

Earliest-Due-Date Scheduling

If the objective is to minimize the maximum lateness, then the jobs should be sequenced according to their due dates. That is, $d_{[1]} \le d_{[2]} \le \cdots \le d_{[n]}$. We will not present a proof of this result. The idea behind the proof is to choose some schedule that does not sequence the jobs in order of their due dates; that implies that there is some value of k such that $d_{[k]} > d_{[k+1]}$. One shows that by interchanging the positions of jobs k and $k+1$, the maximum lateness is reduced.

Minimizing the Number of Tardy Jobs

There are many instances in which the penalty for a late (tardy) job remains the same no matter how late it is. For example, any delay in the completion of all tasks required for preparation of a space launch would cause the launch to be aborted, independent of the length of the delay.

We will describe an algorithm from Moore (1968) that minimizes the number of tardy jobs for the single machine problem.

Step 1. Sequence the jobs according to the earliest due date to obtain the initial solution. That is $d_{[1]} \le d_{[2]} \le \ldots \le d_{[n]}$.

Step 2. Find the first tardy job in the current sequence, say job $[i]$. If none exists go to step 4.

Step 3. Consider jobs [1], [2], . . . , [i]. Reject the job with the largest processing time. Return to step 2.

Step 4. Form an optimal sequence by taking the current sequence and appending to it the rejected jobs. The jobs appended to the current sequence may be scheduled in any order because they constitute the tardy jobs.

Example 7.3

A machine shop processes custom orders from a variety of clients. One of the machines, a grinder, has six jobs remaining to be processed. The processing times and promised due dates (both in hours) for the six jobs are given below.

Job	1	2	3	4	5	6
Due date	15	6	9	23	20	30
Processing time	10	3	4	8	10	6

The first step is to sequence the jobs according to the EDD rule.

Job	2	3	1	5	4	6
Due date	6	9	15	20	23	30
Processing time	3	4	10	10	8	6
Completion time	3	7	17	27	35	41

We see that the first tardy job is job 1, and there are a total of four tardy jobs. We now consider jobs 2, 3, and 1 and reject the job with the longest processing time. This is clearly job 1. At this point, the new current sequence is

Job	2	3	5	4	6
Due date	6	9	20	23	30
Processing time	3	4	10	8	6
Completion time	3	7	17	25	31

The first tardy job in the current sequence is now job 4. We consider the sequence 2, 3, 5, 4, and reject the job with the longest processing time, which is job 5. The current sequence is now

Job	2	3	4	6
Due date	6	9	23	30
Processing time	3	4	8	6
Completion time	3	7	15	21

Clearly there are no tardy jobs at this stage. The optimal sequence is 2, 3, 4, 6, 5, 1 or 2, 3, 4, 6, 1, 5. In either case the number of tardy jobs is exactly 2. ■

Precedence Constraints: Lawler's Algorithm

Lawler's algorithm (Lawler, 1973) is a powerful technique for solving a variety of constrained scheduling problems. The objective function is assumed to be of the form

$$\min \max_{1 \le i \le n} g_i(F_i)$$

where g_i is any nondecreasing function of the flow time F_i. Furthermore, the algorithm handles *any* precedence constraints. Precedence constraints occur when certain jobs must be completed before other jobs can begin; they are quite common in scheduling problems. Some examples of functions g_i that one might consider are $g_i(F_i) = F_i - d_i = L_i$, which corresponds to minimizing maximum lateness, or $g_i(F_i) = \max(F_i - d_i, 0)$, which corresponds to minimizing maximum tardiness.

The algorithm Lawler's algorithm first schedules the job to be completed last, then the job to be completed next to last, and so on. At each stage one determines the set of jobs not required to precede any other. Call this set V. Among the set V, choose the job k that satisfies

$$g_k(\tau) = \min_{i \in V}(g_i(\tau)),$$

where $\tau = \sum_{i=1}^{n} t_i$ and corresponds to the processing time of the current sequence.

Job k is now scheduled last. Consider the remaining jobs and again determine the set of jobs that are not required to precede any other remaining job. After scheduling job k, this set may have changed. The value of τ is reduced by t_k and the job scheduled next to last is now determined. The process is continued until all jobs are scheduled. Note that as jobs are scheduled, some of the precedence constraints may be relaxed, so the set V is likely to change at each iteration.

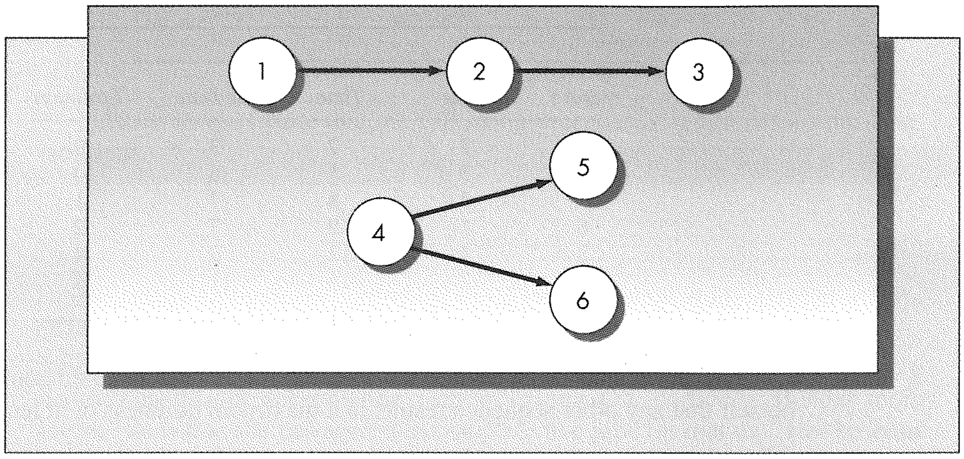
Example 7.4

Tony D'Amato runs a local body shop that does automotive painting and repairs. On a particular Monday morning he has six cars waiting for repair. Three (1, 2, and 3) are from a car rental company and he has agreed to finish these cars in the order of the dates that they were promised. Cars 4, 5, and 6 are from a retail dealer who has requested that car 4 be completed first because a customer is waiting for it. The resulting precedence constraints can be represented as two disconnected networks, as pictured in Figure 7–5.

The times required to repair each of the cars (in days) and the associated promised completion dates are

Job	*1*	*2*	*3*	*4*	*5*	*6*
Processing time	2	3	4	3	2	1
Due date	3	6	9	7	11	7

FIGURE 7–5
Precedence constraints for Example 7.4



Determine how the repair of the cars should be scheduled through the shop in order to minimize the maximum tardiness.

Solution

1. First we find the job scheduled last (sixth). Among the candidates for the last position are those jobs that are not predecessors of other jobs. These are 3, 5, and 6. The total processing time of all jobs is 2 + 3 + 4 + 3 + 2 + 1 = 15. (This is the current value of τ.) As the objective is to minimize the maximum lateness, we compare the lateness of these three jobs and pick the one with the smallest value. We obtain min{15 − 9, 15 − 11, 15 − 7} = min{6, 4, 8} = 4, corresponding to job 5. Hence job 5 is scheduled last (position 6).

2. Next we find the job scheduled fifth. The candidates are jobs 3 and 6 only. At this point the value of τ is 15 − 2 = 13. Hence, we find min{13 − 9, 13 − 7} = min {4, 6} = 4, which corresponds to job 3. Hence, job 3 is scheduled in the fifth position.

3. Find the job scheduled fourth. Because job 3 is no longer on the list, job 2 now becomes a candidate. The current value of τ = 13 − 4 = 9. Hence, we compare min {9 − 6, 9 − 7} = min{3, 2} = 2, which corresponds to job 6. Schedule job 6 in the fourth position.

4. Find the job scheduled third. Job 6 has been scheduled, so job 4 now becomes a candidate along with job 2, and τ = 9 − 1 = 8. Hence, we look for min{8 − 6, 8 − 7} = min{2, 1} = 1, which occurs at job 4.

5. At this point we would find the job scheduled second. However, we are left with only jobs 1 and 2, which, because of the precedence constraints, must be scheduled in the order 1–2.

Summarizing the results above, the optimal sequence to repair the cars is 1–2–4–6–3–5.

In order to determine the value of the objective function, the maximum tardiness, we compute the flow time for each job and compare it to the due date. We have

Job	*Processing Time*	*Flow Time*	*Due Date*	*Tardiness*
1	2	2	3	0
2	3	5	6	0
4	3	8	7	1
6	1	9	7	2
3	4	13	9	4
5	2	15	11	4

X ↑

Hence, the maximum tardiness is four days. The reader should convince him- or herself that any other sequence results in a maximum tardiness of at least four days. ■

Problems for Section 6

6. Consider the information given in Problem 4. Determine the sequence that the trucks should be unloaded in order to minimize
   a. Mean flow time.
   b. Maximum lateness.
   c. Number of tardy jobs.
7. On May 1, a lazy MBA student suddenly realizes that he has done nothing on seven different homework assignments and projects that are due in various courses. He estimates the time required to complete each project (in days) and also notes their due dates:

Project	*1*	*2*	*3*	*4*	*5*	*6*	*7*
Time (days)	4	8	10	4	3	7	14
Due date	4/20	5/17	5/28	5/28	5/12	5/7	5/15

Because projects 1, 3, and 5 are from the same class, he decides to do those in the sequence that they are due. Furthermore, project 7 requires results from projects 2 and 3, so 7 must be done after 2 and 3 are completed. Determine the sequence in which he should do the projects in order to minimize the maximum lateness.

8. Eight jobs are to be processed through a single machine. The processing times and due dates are given below.

Job	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
Processing time	2	3	2	1	4	3	2	2
Due date	5	4	13	6	12	10	15	19

Furthermore, assume that the following precedence relationships must be satisfied:

$$2 \rightarrow 6 \rightarrow 3$$

$$1 \rightarrow 4 \rightarrow 7 \rightarrow 8.$$

Determine the sequence in which these jobs should be done in order to minimize the maximum lateness subject to the precedence restrictions.

9. Jane Reed bakes breads and cakes in her home for parties and other affairs on a contract basis. Jane has only one oven for baking. One particular Monday morning she finds that she has agreed to complete five jobs for that day. Her husband John will make the deliveries, which require about 15 minutes each. Suppose that she begins baking at 8:00 A.M.

Job	*Time Required*	*Promised Time*
1	1.2 hr.	11:30 A.M.
2	40 min.	10:00 A.M.
3	2.2 hr.	11:00 A.M.
4	30 min.	1:00 P.M.
5	3.1 hr.	12:00 noon
6	25 min.	2:00 P.M.

Determine the sequence in which she should perform the jobs in order to minimize

a. Mean flow time.
b. Number of tardy jobs.
c. Maximum lateness.

10. Seven jobs are to be processed through a single machine. The processing times and due dates are given below.

Job	*1*	*2*	*3*	*4*	*5*	*6*	*7*
Processing time	3	6	8	4	2	1	7
Due date	4	8	12	15	11	25	21

Determine the sequence of the jobs in order to minimize

a. Mean flow time.
b. Number of tardy jobs.
c. Maximum lateness.
d. What is the makespan for any sequence?

7.7 Sequencing Algorithms for Multiple Machines

We now extend the analysis of the previous section to the case in which several jobs must be processed on more than one machine. Assume that n jobs are to be processed through m machines. The number of possible schedules is staggering, even for moderate values of both n and m. For each machine, there are $n!$ different orderings of the jobs. If the jobs may be processed on the machines in any order, it follows that there are a total of $(n!)^m$ possible schedules. For example, for $n = 5$, $m = 5$, there are $2{,}4883 \times 10^{10}$, or about 25 billion, possible schedules. Even with the availability of inexpensive computing today, enumerating all feasible schedules for even moderate-sized problems is impossible or, at best, impractical.

In this section we will present some known results for scheduling jobs on more than one machine. A convenient way to represent a schedule is via a Gantt chart. As an example, suppose that two jobs, I and J, are to be scheduled on two machines, 1 and 2. The processing times are

	Machine 1	*Machine 2*
Job I	4	1
Job J	1	4

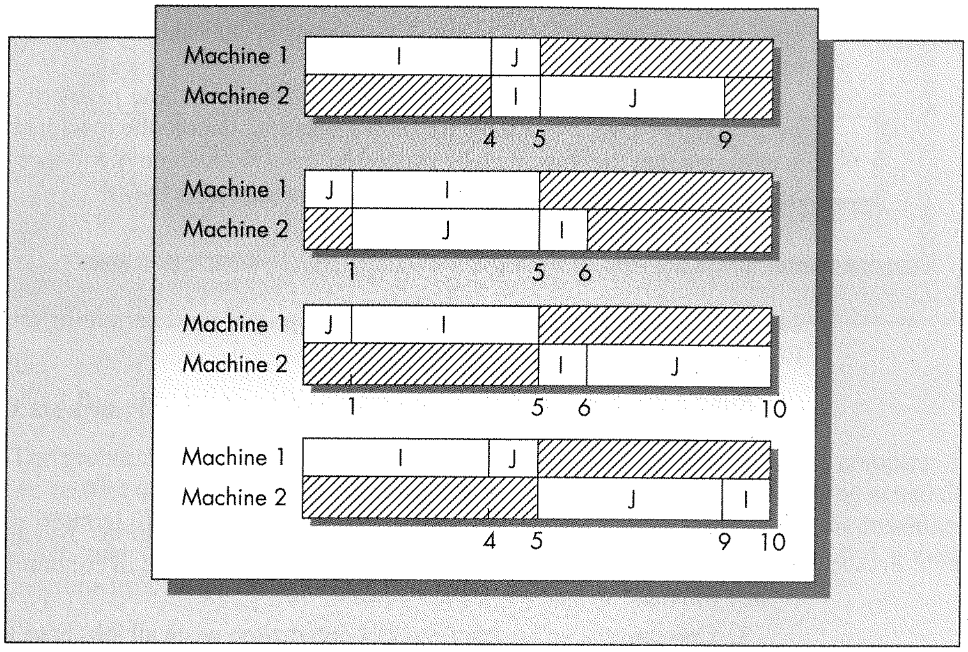
Assume that both jobs must be processed first on machine 1 and then on machine 2. The possible schedules appear in four Gantt charts in Figure 7–6. The first two schedules are known as permutation schedules. That means that the jobs are processed in the same sequence on both machines. Clearly, for this example, the permutation schedules provide better system performance in terms of both total and average flow time.

Recall that the total flow time (or makespan) is the total elapsed time from the initiation of the first job on the first machine until the completion of the last job on the last machine. For the schedules above, the makespans (total flow time) are respectively 9, 6, 10, and 10.

The mean flow time is also used as a measure of system performance. For the first schedule in the example above, the mean flow time is $(5 + 9)/2 = 7$. For the second schedule, it is $(5 + 6)/2 = 5.5$, and so on.

A third possible objective is minimization of the mean idle time in the system. The mean idle time is the arithmetic average of the idle times for each machine. In schedule 1, we see that machine 1 is idle for 4 units of time (between times 5 and 9) and machine 2 is idle for 4 units of time as well (between times 0 and 4). Hence the mean idle time for schedule 1 is 4. In schedule 2, both machines 1 and 2 are idle for 1 unit of time, giving a mean idle time of 1. The mean idle times for schedules 3 and 4 are 5 units of time.

FIGURE 7–6
All possible schedules for two jobs on two machines



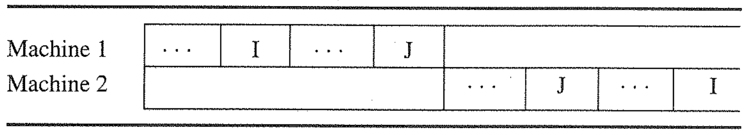
Scheduling n Jobs on Two Machines

Assume that *n* jobs must be processed through two machines and that each job must be processed in the order machine 1 then machine 2. Furthermore, assume that the optimization criterion is to minimize the makespan. (As long as we restrict attention to only two machines, the schedule that minimizes the makespan will also minimize average flow time.) The problem of scheduling on two machines turns out to have a relatively simple solution.

Theorem 7.2

The optimal solution for scheduling *n* jobs on two machines is always a permutation schedule.

This theorem means that one can restrict attention to schedules in which the sequence of jobs is the same on both machines. This result can be demonstrated as follows. Consider a schedule for *n* jobs on two machines in which the sequencing of the jobs on the two machines is different. That is, the schedule looks as follows:



By reversing the position of these jobs on either machine, the flow time decreases. By scheduling the jobs in the order I–J on machine 2 the pair (I, J) on machine 2 may begin after I is completed on machine 1, rather than having to wait until J is completed on machine 1.

Because the total number of permutation schedules is exactly $n!$, determining optimal schedules for two machines is roughly of the same level of difficulty as determining optimal schedules for one machine.

A very efficient algorithm for solving the two-machine problem was discovered by Johnson (1954). Following Johnson's notation, denote the machines by A and B. It is assumed that the jobs must be processed first on machine A and then on machine B. Suppose that the jobs are labeled i, for $1 \leq i \leq n$, and define

A_i = Processing time of job i on machine A.
B_i = Processing time of job i on machine B.

Johnson's result is that the following rule is optimal for determining an order in which to process the jobs on the two machines.

Rule: Job i precedes job $i + 1$ if $\min(A_i, B_{i-1}) < \min(A_{i+1}, B_i)$.

An easy way to implement this rule is as follows:

1. List the values of A_i and B_i in two columns.
2. Find the smallest remaining element in the two columns. If it appears in Column A, then schedule that job next. If it appears in column B, then schedule that job last.
3. Cross off the jobs as they are scheduled. Stop when all jobs have been scheduled.

Example 7.5

Five jobs are to be scheduled on two machines. The processing times are

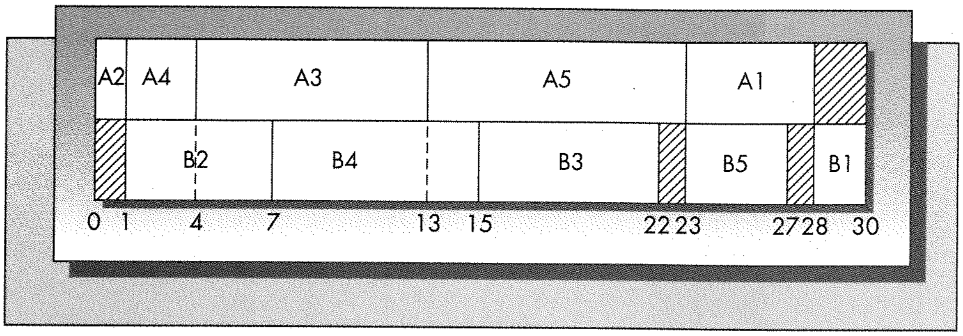
Job	*Machine A*	*Machine B*
1	5	2
2	1	6
3	9	7
4	3	8
5	10	4

The first step is to identify the minimum job time. It is 1, for job 2 on machine A. Because it appears in column A, job 2 is scheduled first and row 2 is crossed out. The next smallest element is 2, for job 1 on machine B. This appears in the B column, so job 1 is scheduled last. The next smallest element is 3, corresponding to job 4 in column A, so that job 4 is scheduled next. Continuing in this fashion, we obtain the optimal sequence

2–4–3–5–1.

The Gantt chart for the optimal schedule is pictured in Figure 7–7. Note that there is no idle time between jobs on machine A. This is a feature of all optimal schedules. ■

FIGURE 7–7
Gantt chart for the optimal schedule for Example 7.5



Extension to Three Machines

The problem of scheduling jobs on three machines is considerably more complex. If we restrict attention to total flow time only, it is still true that a permutation schedule is optimal (this is not necessarily the case for average flow time). Label the machines A, B, and C. The three-machine problem can be reduced to (essentially) a two-machine problem if the following condition is satisfied:

$$\min A_i \geq \max B_i \quad \text{or} \quad \min C_i \geq \max B_i$$

It is only necessary that *either one* of these conditions be satisfied. If that is the case, then the problem is reduced to a two-machine problem in the following way.

Define $A'_i = A_i + B_i$, and define $B'_i = B_i + C_i$. Now solve the problem using the rules described above for two machines, treating A'_i and B'_i as the processing times. The resulting permutation schedule will be optimal for the three-machine problem.

Example 7.6

Consider the following job times for a three-machine problem. Assume that the jobs are processed in the sequence A–B–C.

	Machine		
Job	*A*	*B*	*C*
1	4	5	8
2	9	6	10
3	8	2	6
4	6	3	7
5	5	4	11

Checking the conditions, we find

$\min A_i = 4,$

$\max B_i = 6,$
$\min C_i = 6,$

so that the required condition is satisfied. We now form the two columns A′ and B′.

	Machine	
Job	*A′*	*B′*
1	9	13
2	15	16
3	10	8
4	9	10
5	9	15

The problem is now solved using the two-machine algorithm. The optimal solution is

1–4–5–2–3.

Note that because of ties in column A, the optimal solution is not unique. ■

If the conditions for reducing a three-machine problem to a two-machine problem are not satisfied, this method will usually give reasonable, but possibly suboptimal, results. As long as the objective is to minimize the makespan or total flow time, a permutation schedule is optimal for scheduling on three machines. (It is not necessarily true, however, that a permutation schedule is optimal for three machines when using an average flow time criterion.)

Note that we assume that the machines are different and the processing proceeds sequentially: all jobs are assumed to be processed first on machine 1, then on machine 2. For example, machine 1 might be a drill press and machine 2 a lathe. A related problem that we discuss in the context of stochastic scheduling is that of parallel processing on identical machines. In this case the machines are assumed to perform the same function, and any job may be assigned to any machine. For example, a collection of 10 jobs might require processing on either one of two drill presses. The results for parallel processing suggest that SPT is an effective rule for minimizing mean flow time, but LPT (longest processing time first) is often more effective for minimizing total flow time or makespan. We will discuss parallel processing in the context of random job times in a later section.

The Two-Job Flow Shop Problem

Assume that two jobs are to be processed through m machines. Each job must be processed by the machines in a particular order, but the sequences for the two jobs need not be the same. We present a graphical procedure for solving this problem developed by Akers (1956).

1. Draw a cartesian coordinate system with the processing times corresponding to the first job on the horizontal axis and the processing

times corresponding to the second job on the vertical axis. On each axis, mark off the operation times in the order in which the operations must be performed for that job.

2. Block out areas corresponding to each machine at the intersection of the intervals marked for that machine on the two axes.
3. Determine a path from the origin to the end of the final block that does not intersect any of the blocks and that minimizes the vertical movement. Movement is allowed only in three directions: horizontal, vertical, and 45-degree diagonal. The path with minimum vertical distance will indicate the optimal solution. Note that this will be the same as the path with minimum horizontal distance.

This procedure is best illustrated by example.

Example 7.7

A regional manufacturing firm produces a variety of household products. One is a wooden desk lamp. Prior to packing, the lamps must be sanded, lacquered, and polished. Each operation requires a different machine. There are currently shipments of two models awaiting processing. The times required for the three operations for each of the two shipments are

Job 1		**Job 2**	
Operation	*Time*	*Operation*	*Time*
Sanding (A)	3	A	2
Lacquering (B)	4	B	5
Polishing (C)	5	C	3

The first step is to block out the job times on each of the axes. Refer to Figure 7–8 for this step. Every feasible schedule is represented by a line connecting the origin to the tip of block C, with the condition that the line not go through a block. Only three types of movement are allowed: horizontal, vertical, and 45-degree diagonal. Horizontal movement implies that only job 1 is being processed, vertical movement implies that only job 2 is being processed, and diagonal movement implies that both jobs are being processed. Minimizing the flow time is the same as maximizing the time that both jobs are being processed. This is equivalent to finding the path from the origin to the end of block C that maximizes the diagonal movement and therefore minimizes either the horizontal or the vertical movement.

Two feasible schedules for this problem are represented in Figure 7–8. The total time required by any feasible schedule can be obtained in two ways: it is either the total time represented on the horizontal axis (12 in this case) plus the total vertical movement (4 and 3 respectively), or the total time on the vertical axis (10 in this case) plus the total horizontal movement (6 and 5 respectively). Schedule 1 has total time 16, and schedule 2 has total time 15. Schedule 2 turns out to be optimal for this problem.

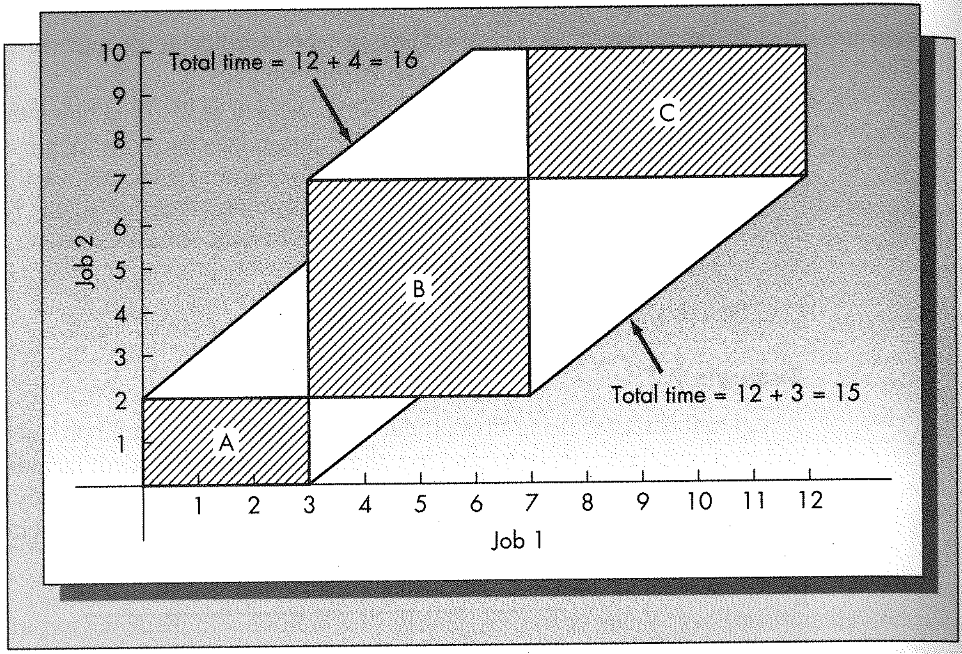


FIGURE 7–8
Graphical solution of Example 7.7

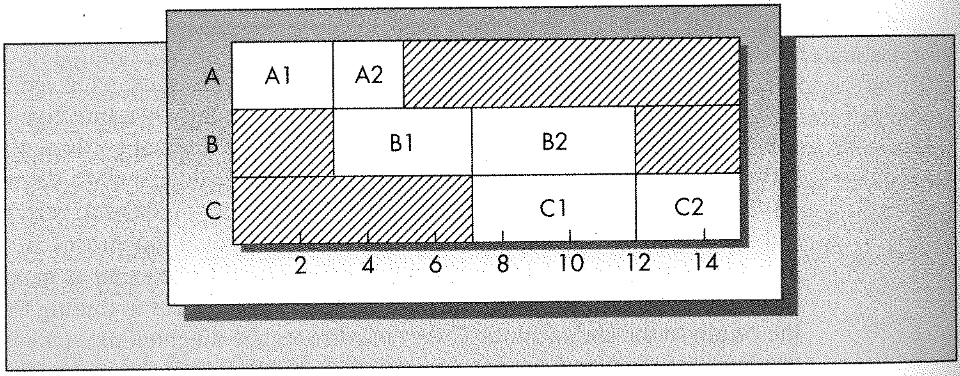


FIGURE 7–9
Gantt chart for optimal solution to Example 7.7

The Gantt chart for the optimal schedule appears in Figure 7–9. ■

We should note that this method does *not* require the two jobs to be processed in the same sequence on the machines. We present another example to illustrate the case in which the sequence of the jobs is different.