Chapter 1

Classification of Scheduling Problems

The theory of scheduling is characterized by a virtually unlimited number of problem types (see, e.g. Baker [12], Blazewicz et al. [27], Coffman [69], Conway et al. [72], French [93], Lenstra [151] , Pinedo [180], Rinnooy Kan [181], Tanaev et al. [193], Tanaev et al. [194]). In this chapter, a basic classification for the scheduling problems covered in the first part of this book will be given. This classification is based on a classification scheme widely used in the literature (see, e.g. Lawler et al. [145]). In later chapters we will extend this classification scheme.

1.1 Scheduling Problems

Suppose that m **machines** $M_j (j = 1, \ldots, m)$ have to process n **jobs** $J_i (i = 1, \ldots, n)$. A **schedule** is for each job an allocation of one or more time intervals to one or more machines. Schedules may be represented by **Gantt charts** as shown in Figure 1.1. Gantt charts may be machine-oriented (Figure 1.1(a)) or job-oriented (Figure 1.1(b)). The corresponding scheduling problem is to find a schedule satisfying certain restrictions.

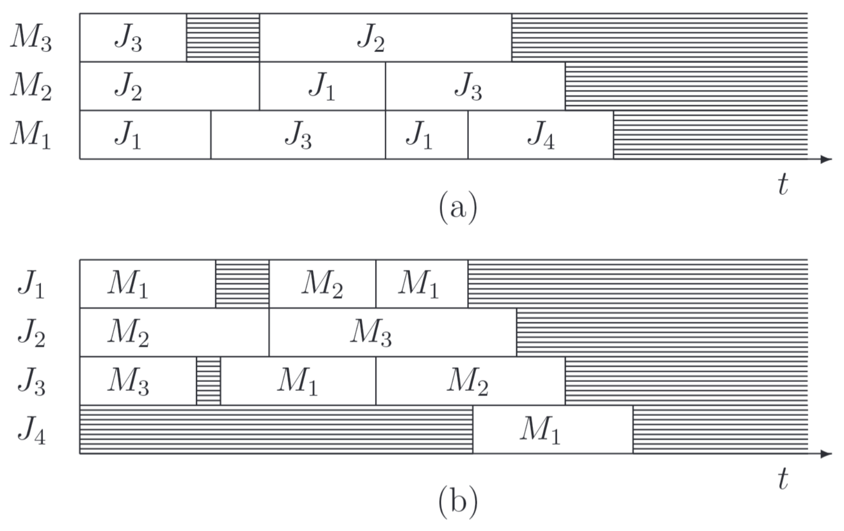

Figure 1.1: Machine- and job-oriented Gantt charts.

1.2 Job Data

A job J_i consists of a **number** n_i **of operations** $O_{i1}, \ldots, O_{i,n_i}$. Associated with operation O_{ij} is a **processing requirement** p_{ij}. If job J_i consists of only one operation ($n_i = 1$), we then identify J_i with O_{i1} and denote the processing requirement by p_i. Furthermore, a **release date** r_i, on which the first operation of J_i becomes available for processing may be specified. Associated with each operation O_{ij} is a **set of machines** $\mu_{ij} \subseteq \{M_1, \ldots, M_m\}$. O_{ij} may be processed on any of the machines in μ_{ij}. Usually, all μ_{ij} are one element sets or all μ_{ij} are equal to the set of all machines. In the first case we have **dedicated machines**. In the second case the machines are called **parallel**. The general case is introduced here to cover problems in flexible manufacturing where machines are equipped with different tools. This means that an operation can be processed on any machine equipped with the appropriate tool. We call scheduling problems of this type problems with **multi-purpose machines** (MPM).

It is also possible that all machines in the set μ_{ij} are used simultaneously by O_{ij} during the whole processing period. Scheduling problems of this type are called **multiprocessor task scheduling problems**. Scheduling problems with multi-purpose machines and multiprocessor task scheduling problems will be classified in more detail in Chapters 10 and 11.

Finally, there is a **cost function** $f_i(t)$ which measures the cost of completing J_i at time t. A **due date** d_i and a **weight** w_i may be used in defining f_i.

In general, all data $p_i, p_{ij}, r_i, d_i, w_i$ are assumed to be integer. A schedule is **feasible** if no two time intervals overlap on the same machine, if no two time intervals allocated to the same job overlap, and if, in addition, it meets a number of problem-specific characteristics. A schedule is **optimal** if it minimizes a given optimality criterion.

Sometimes it is convenient to identify a job J_i by its index i. We will use this brief notation in later chapters.

We will discuss a large variety of classes of scheduling problems which differ in their complexity. Also the algorithms we will develop are quite different for different classes of scheduling problems. Classes of scheduling problems are specified in terms of a three-field classification $\alpha|\beta|\gamma$ where α specifies the **machine environment** , β specifies the **job characteristics** , and γ denotes the **optimality criterion**. Such a classification scheme was introduced by Graham et al. [108].

1.3 Job Characteristics

The job characteristics are specified by a set β containing at the most six elements $\beta_1, \beta_2, \beta_3, \beta_4, \beta_5$, and β_6.

β_1 indicates whether **preemption** (or job splitting) is allowed. Preemption of a job or operation means that processing may be interrupted and resumed at a later time, even on another machine. A job or operation may be interrupted several times. If preemption is allowed, we set $\beta_1 = pmtn$. Otherwise β_1 does not appear in β.

β_2 describes **precedence relations** between jobs. These precedence relations may be represented by an acyclic directed graph $G = (V, A)$ where $V = \{1, \ldots, n\}$ corresponds with the jobs, and $(i, k) \in A$ iff J_i must be completed before J_k starts. In this case we write $J_i \rightarrow J_k$. If G is an arbitrary acyclic directed graph we set $\beta_2 =$ **prec**. Sometimes we will consider scheduling problems with restricted precedences given by chains, an intree, an outtree, a tree or a series-parallel directed graph. In these cases we set β_2 equal to chains, intree, outtree, and *sp*-graph.

If $\beta_2 =$ **intree** (**outtree**), then G is a rooted tree with an outdegree (indegree) for each vertex of at the most one. Thus, in an intree (outtree) all arcs are directed towards (away from) a root. If $\beta_2 =$ **tree**, then G is either an intree or an outtree. A set of **chains** is a tree in which the outdegree and indegree for each vertex is at the most one. If $\beta_2 =$ chains, then G is a set of chains.

Series-parallel graphs are closely related to trees. A graph is called **series-parallel** if it can be built by means of the following rules:

Base graph. Any graph consisting of a single vertex is series-parallel. Let $G_i = (V_i, A_i)$ be series-parallel $(i = 1, 2)$.

Parallel composition. The graph $G = (V_1 \cup V_2, A_1 \cup A_2)$ formed from G_1 and G_2 by joining the vertex sets and arc sets is series parallel.

Series composition. The graph $G = (V_1 \cup V_2, A_1 \cup A_2 \cup T_1 \times S_2)$ formed from G_1 and G_2 by joining the vertex sets and arc sets and adding all arcs (t, s) where t belongs to the set T_1 of sinks of G_1 (i.e. the set of vertices without successors) and s belongs to the set S_2 of sources of G_2 (i.e. the set of vertices without predecessors) is series parallel.

We set $\beta_2 =$ **sp-graph** if G is series parallel. If there are no precedence constraints, then β_2 does not appear in β.

If $\beta_3 = r_i$, then release dates may be specified for each job. If $r_i = 0$ for all jobs, then β_3 does not appear in β.

β_4 specifies restrictions on the processing times or on the number of operations. If β_4 is equal to $p_i = 1 (p_{ij} = 1)$, then each job (operation) has a **unit processing requirement**. Similarly, we may write $p_i = p(p_{ij} = p)$. Occasionally, the β_4 field contains additional characteristics with an obvious interpretation such as $p_i \in \{1, 2\}$ or $d_i = d$.

If $\beta_5 = d_i$, then a deadline d_i is specified for each job J_i, i.e. job J_i must finish not later than time d_i.

In some scheduling applications, sets of jobs must be grouped into batches. A batch is a set of jobs which must be processed jointly on a machine. The finishing time of all jobs in a batch is defined as equal to the finishing time of the batch. A batch may consist of a single job up to n jobs. There is a set-up time s for each batch. We assume that this set-up time is the same for all batches and sequence independent. A **batching problem** is to group the jobs into batches and to schedule these batches. There are two types of batching problems, denoted by **p-batching problems**, and **s-batching problems**. For p-batching problems (s batching-problems) the length of a batch is equal to the maximum (sum) of processing times of all jobs in the batch. $\beta_6 = p$-batch or $\beta_6 = s$-batch indicates a batching problem. Otherwise β_6 does not appear in β.

1.4 Machine Environment

The machine environment is characterized by a string $\alpha = \alpha_1\alpha_2$ of two parameters. Possible values of α_1 are $\circ, P, Q, R, PMPM, QMPM, G, X, O, J, F$. If $\alpha_1 \in \{\circ, P, Q, R, PMPM, QMPM\}$, where $\circ$ denotes the empty symbol (thus, $\alpha = \alpha_2$ if $\alpha_1 = \circ$), then each J_i consists of a single operation.

If $\alpha_1 = \circ$, each job must be processed on a specified **dedicated** machine.

If $\alpha_1 \in \{P, Q, R\}$, then we have parallel machines, i.e. each job can be processed on each of the machines $M_1, \ldots, M_m$. If $\alpha_1 = P$, then there are **identical parallel machines**. Thus, for the processing time p_{ij} of job J_i on M_j we have $p_{ij} = p_i$ for all machines M_j. If $\alpha_1 = Q$, then there are **uniform parallel machines**, i.e. $p_{ij} = p_i/s_j$ where s_j is the speed of machine M_j. Finally, if $\alpha_1 = R$, then there are **unrelated parallel machines**, i.e. $p_{ij} = p_i/s_{ij}$ for job-dependent speeds s_{ij} of M_j.

If $\alpha_1 = PMPM$ and $\alpha_1 = QMPM$, then we have multi-purpose machines with identical and uniform speeds, respectively.

If $\alpha_1 \in \{G, X, O, J, F\}$, we have a multi-operation model, i.e. associated with each job J_i there is a set of operations $O_{i1}, \ldots, O_{i,n_i}$. The machines are dedicated, i.e. all μ_{ij} are one element sets. Furthermore, there are precedence relations between arbitrary operations. This general model is called a **general shop**. We indicate the general shop by setting $\alpha_1 = G$. Job shops, flow shops, open shops, and mixed shops are special cases of the general shop. In a **job shop** , indicated by $\alpha_1 = J$, we have special precedence relations of the form

$$O_{i1} \rightarrow O_{i2} \rightarrow O_{i3} \rightarrow \ldots \rightarrow O_{i,n_i} \quad \text{for } i = 1, \ldots, n.$$

Furthermore, we generally assume that $\mu_{ij} \neq \mu_{i,j+1}$ for $j = 1, \ldots, n_i - 1$. We call a job shop in which $\mu_{ij} = \mu_{i,j+1}$ is possible a **job shop with machine repetition**.

The **flow shop**, indicated by $\alpha_1 = F$, is a special case of the job-shop in which $n_i = m$ for $i = 1, \ldots, n$ and $\mu_{ij} = \{M_j\}$ for each $i = 1, \ldots, n$ and $j = 1, \ldots, m$. The **open shop** , denoted by $\alpha_1 = O$, is defined as the flow shop, with the exception that there are no precedence relations between

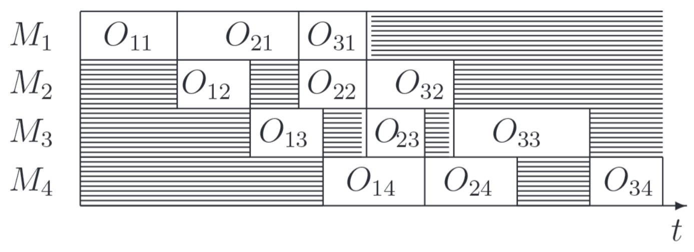

Figure 1.2: Feasible schedule for a permutation flow shop problem.

the operations. A **mixed shop**, indicated by $\alpha_1 = X$, is a combination of a job shop and an open shop.

A **permutation flow shop** is a flow shop in which jobs are processed in the same order on each machine. Figure 1.2 shows a feasible schedule for a permutation flow shop. If we have a job shop problem, we may set β_4 equal to $n_i \leq 2$. In this case all jobs have at the most two operations.

If α_2 is equal to a positive integer $1, 2, \ldots$, then α_2 denotes the number of machines. If $\alpha_2 = k$, then k is an arbitrary, but fixed number of machines. If the number of machines is arbitrary, we set $\alpha_2 = \circ$.

1.5 Optimality Criteria

We denote the **finishing time** of job J_i by C_i, and the associated **cost** by $f_i(C_i)$. There are essentially two types of total cost functions

$$f_{\max}(C) := \max\{f_i(C_i) | i = 1, \ldots, n\}$$

and

$$\sum f_i(C) := \sum_{i=1}^{n} f_i(C_i)$$

called **bottleneck objectives** and **sum objectives**, respectively. The **scheduling problem** is to find a feasible schedule which minimizes the total cost function.

If the functions f_i are not specified, we set $\gamma = f_{\max}$ or $\gamma = \sum f_i$. However, in most cases we consider special functions f_i.

The most common objective functions are the **makespan** $\max\{C_i | i = 1, \ldots, n\}$, **total flow time** $\sum\limits_{i=1}^{n} C_i$, and **weighted (total) flow time**

$\sum_{i=1}^{n} w_i C_i$. In these cases we write $\gamma = C_{\max}, \gamma = \sum C_i$, and $\gamma = \sum w_i C_i$, respectively.

Other objective functions depend on due dates d_i which are associated with jobs J_i. We define for each job J_i:

$$
\begin{array}{lll}
L_i := & C_i - d_i & \textbf{lateness} \\[2ex]
E_i := & \max\{0, d_i - C_i\} & \textbf{earliness} \\[2ex]
T_i := & \max\{0, C_i - d_i\} & \textbf{tardiness} \\[2ex]
D_i := & |C_i - d_i| & \textbf{absolute deviation} \\[2ex]
S_i := & (C_i - d_i)^2 & \textbf{squared deviation} \\[2ex]
U_i := & \begin{cases} 0 & \text{if } C_i \leq d_i \\ 1 & \text{otherwise} \end{cases} & \textbf{unit penalty}.
\end{array}
$$

With each of these functions G_i we get four possible objectives $\gamma = \max G_i, \max w_i G_i, \sum G_i, \sum w_i G_i$. The most important bottleneck objective besides $C_{\max}$ is **maximum lateness** $L_{\max} := \max_{i=1}^{n} L_i$. Other objective functions which are widely used are $\sum T_i$, $\sum w_i T_i$, $\sum U_i$, $\sum w_i U_i$, $\sum D_i$, $\sum w_i D_i$, $\sum S_i$, $\sum w_i S_i$, $\sum E_i$, $\sum w_i E_i$. Linear combinations of these objective functions are also considered.

An objective function which is nondecreasing with respect to all variables C_i is called **regular**. Functions involving E_i, D_i, S_i are not regular. The other functions defined so far are regular.

A schedule is called **active** if it is not possible to schedule jobs (operations) earlier without violating some constraint. A schedule is called **semiactive** if no job (operation) can be processed earlier without changing the processing order or violating the constraints.

1.6 Examples

To illustrate the three-field notation $\alpha|\beta|\gamma$ we present some examples. In each case we will describe the problem. Furthermore, we will specify an instance and present a feasible schedule for the instance in the form of a Gantt chart.

Example 1.1 $P|prec; p_i = 1|C_{\max}$ is the problem of scheduling jobs with unit processing times and arbitrary precedence constraints on m identical machines such that the makespan is minimized.

An instance is given by a directed graph with n vertices and the number of machines.

Figure 1.3 shows an instance of this problem and a corresponding feasible schedule.

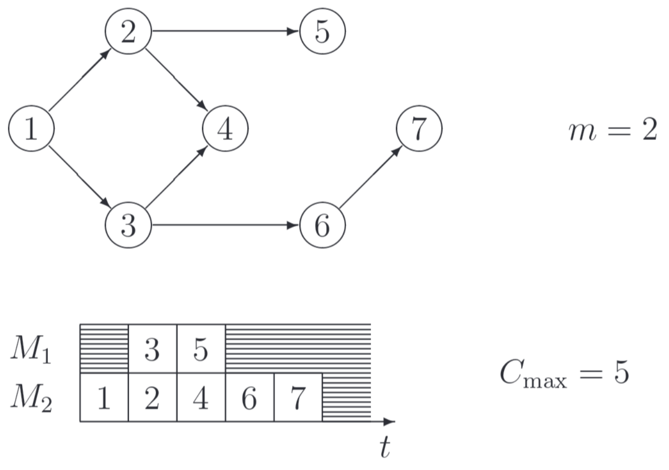

Figure 1.3: Instance for $P \mid prec; p_i = 1 \mid C_{\max}$.

Example 1.2 $1|s\text{-batch}|\sum w_iC_i$ is the problem of splitting a set of jobs into batches and scheduling these batches on one machine such that the weighted flow time is minimized. The processing time of a batch is the sum of processing times of the jobs in the batch.

Figure 1.4 shows a schedule with three batches for the following instance of this problem:

i	1	2	3	4	5	6
p_i	3	2	2	3	1	1
w_i	1	2	1	1	4	4

$s = 1$

The objective value for the schedule is

$$\sum w_iC_i = 2 \cdot 3 + (1 + 1 + 4) \cdot 10 + (1 + 4)15.$$

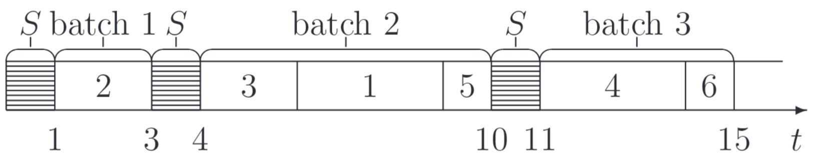

Figure 1.4: Schedule with three batches.

Example 1.3 $1|r_i; pmtn|L_{\max}$ is the problem of finding a preemptive schedule on one machine for a set of jobs with given release times $r_i \neq 0$ such that the maximum lateness is minimized. An instance is presented in Figure 1.5

i	1	2	3	4
p_i	2	1	2	2
r_i	1	2	2	7
d_i	2	3	4	8

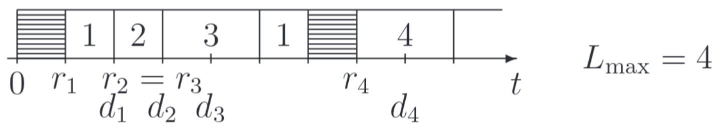

Figure 1.5: Instance for $1 \mid r_i; pmtn \mid L_{\max}$.

Example 1.4 $J3|p_{ij} = 1|C_{\max}$ is the problem of minimizing maximum completion time in a three-machine job shop with unit processing times. An instance is presented in Figure 1.6. The table contains the machines μ_{ij} associated with the operations O_{ij}.

i/j	1	2	3	4
1	M_1	M_3	M_2	M_1
2	M_2	M_3	–	–
3	M_3	M_1	–	–
4	M_1	M_3	M_1	–
5	M_3	M_1	M_2	M_3

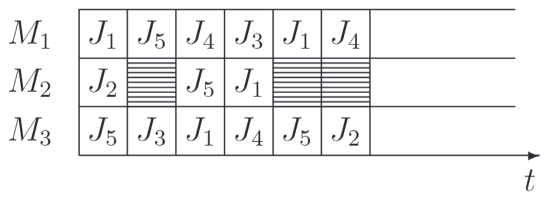
M_1 J_1 J_5 J_4 J_3 J_1 J_4
M_2 J_2 J_5 J_1
M_3 J_5 J_3 J_1 J_4 J_5 J_2
t

Figure 1.6: Instance for $J_3 \mid p_{ij} = 1 \mid C_{\max}$.