A bank of machines in parallel is a setting that is important from both a theoretical and a practical point of view. From a theoretical point of view it is a generalization of the single machine, and a special case of the flexible flow shop. From a practical point of view, it is important because the occurrence of resources in parallel is common in the real world. Also, techniques for machines in parallel are often used in decomposition procedures for multi-stage systems.

In this chapter several objectives are considered. The three principal objectives are the minimization of the makespan, the total completion time, and the maximum lateness. With a single machine the makespan objective is usually only of interest when there are sequence dependent setup times; otherwise the makespan is equal to the sum of the processing times and is independent of the sequence. When dealing with machines in parallel the makespan becomes an objective of considerable interest. In practice, one often has to deal with the problem of balancing the load on machines in parallel; by minimizing the makespan the scheduler ensures a good balance.

One may actually consider the scheduling of parallel machines as a two step process. First, one has to determine which jobs have to be allocated to which machines; second, one has to determine the sequence of the jobs allocated to each machine. With the makespan objective only the allocation process is important.

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>The Makespan without Preemptions</td>
<td>112</td>
</tr>
<tr>
<td>5.2</td>
<td>The Makespan with Preemptions</td>
<td>122</td>
</tr>
<tr>
<td>5.3</td>
<td>The Total Completion Time without Preemptions</td>
<td>129</td>
</tr>
<tr>
<td>5.4</td>
<td>The Total Completion Time with Preemptions</td>
<td>133</td>
</tr>
<tr>
<td>5.5</td>
<td>Due Date Related Objectives</td>
<td>136</td>
</tr>
<tr>
<td>5.6</td>
<td>Online Scheduling</td>
<td>137</td>
</tr>
<tr>
<td>5.7</td>
<td>Discussion</td>
<td>142</td>
</tr>
</tbody>
</table>
With parallel machines, preemptions play a more important role than with a single machine. With a single machine preemptions usually only play a role when jobs are released at different points in time. In contrast, with machines in parallel, preemptions are important even when all jobs are released at the same time.

For most models considered in this chapter there are optimal schedules that are non-delay. However, if there are unrelated machines in parallel and the total completion time must be minimized without preemptions, then the optimal schedule may not be non-delay.

Most models considered in this chapter fall in the category of the so-called offline scheduling problems. In an offline scheduling problem all data (e.g., processing times, release dates, due dates) are known in advance and can be taken into account in the optimization process. In contrast, in an online scheduling problem, the problem data are not known a priori. The processing time of a job only becomes known the moment it is completed and a release date only becomes known the moment a job is released. Clearly, the algorithms for online scheduling problems tend to be quite different from the algorithms for offline scheduling problems. The last section in this chapter focuses on online scheduling of parallel machines.

The processing characteristics and constraints considered in this chapter include precedence constraints as well as the set functions $M_j$. Throughout this chapter it is assumed that $p_1 \geq \cdots \geq p_n$.

5.1 The Makespan without Preemptions

First, the problem $Pm || C_{\text{max}}$ is considered. This problem is of interest because minimizing the makespan has the effect of balancing the load over the various machines, which is an important objective in practice.

It is easy to see that $P2 || C_{\text{max}}$ is NP-hard in the ordinary sense as it is equivalent to PARTITION (see Appendix D). During the last couple of decades many heuristics have been developed for $Pm || C_{\text{max}}$. One such heuristic is described below.

The Longest Processing Time first (LPT) rule assigns at $t = 0$ the $m$ longest jobs to the $m$ machines. After that, whenever a machine is freed the longest job among those not yet processed is put on the machine. This heuristic tries to place the shorter jobs more towards the end of the schedule, where they can be used for balancing the loads.

In the next theorem an upper bound is presented for

$$\frac{C_{\text{max}}(\text{LPT})}{C_{\text{max}}(\text{OPT})},$$

where $C_{\text{max}}(\text{LPT})$ denotes the makespan of the LPT schedule and $C_{\text{max}}(\text{OPT})$ denotes the makespan of the (possibly unknown) optimal schedule. This type