Chapter 3
Overview of Scheduling Models

3.1 Introduction

As it has been discussed in a previous chapter (see Sect. 1.5), scheduling refers to a decision-making process in which, in order to solve a real-world problem, a formal *model* is obtained. This part of the book is devoted to (formal) scheduling models and address the elements composing a scheduling model (jobs/machines/operations, constraints, criteria) as well as the main issues related to building scheduling models. In particular, this chapter overviews the basics of modelling scheduling decision problems. As we will point out, the modelling process in scheduling goes first through a rather detailed classification of scheduling models.

More specifically, in this chapter, we

- give an outline of the nature of modeling for problem solving (Sect. 3.1.1), together with the main characteristics of models, types of models, common errors and shortcomings (Sect. 3.1.2)
- introduce the basic definitions for building manufacturing scheduling models (Sect. 3.2.1), with special emphasis in the resources (Sect. 3.2.2) and processing layouts (Sect. 3.2.3), discuss the main processing constraints (Sect. 3.2.4) and outline the types of scheduling criteria (Sect. 3.2.5)
- present a classification for scheduling models (Sect. 3.3),
- discuss the identification of jobs as the processing unit in scheduling models (Sect. 3.4).

3.1.1 Modelling as an Approach to Scheduling Problem Solving

Modelling lies within the very core of the scientific approach to problem solving. Reality is utterly complex and varied. Models give some rough classification of problems and settings in an attempt to guide the researcher in a fruitful direction.

J. M. Framinan et al., *Manufacturing Scheduling Systems*,
DOI: 10.1007/978-1-4471-6272-8_3, © Springer-Verlag London 2014

In a nutshell, a model is a stripped down, simplified and abstract representation or view of an otherwise complex, detailed and broad reality. Scientific modelling is therefore the process of generating the abstract and conceptual models that will be of use to researchers and practitioners. Modelling is, in itself, a science that stems from the philosophy of science, systems theory, data and knowledge visualisation, statistics and many other branches of science.

Models represent objects or entities that have an empirical existence. In manufacturing scheduling, we will model all relevant productive resources such as machines, workforce, tooling and storage areas. How a model is constructed is closely tied to the final intended use of the model itself. A map, for example, is a geographical model. A very simplified map is useful to understand, in a few seconds, a weather forecast. However, we need a more detailed topological map in order to construct a bridge, for example.

One preliminary question at this stage is 'Why modelling?'. Models have a countless series of characteristics and benefits that can be summarised as follows:

- Models focus on certain aspects of reality. Human thought can then be applied to a more focused part and later amplified.
- A model is always related to the reality or 'target' that it modelises. As mentioned previously, the model nature and the model building process will vary as a function of the intended use (Gilbert 1991). Furthermore, the model should be as close as possible to the target so as to achieve its intended goal. A too broad or general model might fail to serve its purpose, so might be a very detailed model.
- A model is different from reality. It has to be a simplification. If a model matches exactly the target, it is no longer a model but a copy.
- As a result of the two previous items, a model is always a compromise between verboseness and simplification.
- Models, or better, the results given by the experimentation of the model, can be often easily measured. This is specially important when measurements over the modelled reality are impossible or impractical to obtain (Mayer 1992).
- Models can be used to study altered realities. Once a model is thought to be valid for the present modelled reality, altered models can be used to study 'What if?' scenarios on different realities, with some degree of confidence, that would be otherwise impossible to study.
- Models, once validated, allow to explain past events, to predict future observations and/or to refute or approve hypotheses over the modelled reality.
- Models have to be easy to work with. The reality is modelled since it is too complex. A model that is hard to handle and to interpret is not a valid model.
- Models are full of limitations. The interpretation of the results of a model are only valid for the model and extrapolation to reality has to be done in a sound and scientific way. The model has to be validated and the results are closely examined.

Once the definition of a model has been introduced, we discuss the different types of models, along with common mistakes in modelling in the next section.

3.1.2 Types of Models and Common Errors in Modelling

Models range from simple pictograms (e.g. traffic signs) to complex distributed and evolutionary agent systems. The following list is just a small summary of the types of models and methodologies that result in models and that are of use in scheduling problem solving:

- Mathematical models: mathematical programming, linear programming, non-linear programming, deterministic, probabilistic, etc.,
- Graphical model: pictograms, drawings, Venn diagrams, flow charts, etc.,
- Statistical models,
- Simulation models, and
- Algorithmic models.

Mathematical and algorithmic modelling are the bulk of the types of models used in deterministic scheduling. This will be later explained in Chap. 6.

Models are not the holy grail of the scientific method. Actually, they are a necessary evil. This is so since no matter how perfect the analysis of the outcome of a model might be, the conclusions of the whole research could be completely flawed due to a countless series of problems that abound in model building. In the following, we provide just a summarised list:

1. The model could be incomplete to the point of rendering the results impractical for the studied reality.
2. The model might be unrelated with the studied reality. This is often referred to as the 'engineering problem' which is, 'perfectly solving the wrong problem.'
3. The model might be fed with inaccurate or incomplete data. Hence, the conclusions from its study might be off from the modelled reality.
4. The model works only under severe assumptions and limitations that hinder its generality and inferential power.
5. The model might be too complex to solve realistically sized instance problems. Again, the generalisations might be hard to achieve under such circumstances.
6. Improper procedure when testing the results of the model. Lack of sound statistical testing of the sampled results.
7. Related with the above, small and unrepresentative samples used in the tests.
8. Perceptual, interested bias or even fraud in the interpretation of the results of the model.
9. Difficulty in reproducing the results. The lack of the reproducibility is behind many scientific faux pas. Many of the above items result in irreproducibility.
10. Reasoning errors. Confusing causality with casuality. Mistaking correlation for causation, etc.

All the previous problems also occur in building manufacturing scheduling models. As we will closely study in the following chapters, specially in Chap. 6, the most important problems in scheduling model construction are the ones listed in the previous list as 1, 2, 3, 4 and 5. More specifically, if a scheduling model is

overly simple, the resulting production scheduling will be orientative at best for the modelled production scheduling problem. In many extreme cases, the scheduling model studied is such an overly simplification that the results could be completely disconnected with practice. Furthermore, many scheduling models work under so many severe assumptions that result also in disconnections with reality. Similarly, complex models are many times hard to solve with current technologies and solutions to all but the smallest instance problems are unattainable. As we will mention in later chapters, the correct formulation of sufficiently precise manufacturing scheduling models is one of the main issues in the scheduling field.

3.2 Formal Definitions

A rough sketch of some basic definitions related to manufacturing scheduling (such as, e.g. jobs, machines, routing process) has been already given in Sect. 1. In this section, we delve into all these items by providing a formal definition, along with a mathematical notation that we will use to model manufacturing scheduling decision problems.

3.2.1 Scheduling Model

A *scheduling model* is a formal abstraction of a (manufacturing) scheduling decision-making problem which is described by considering the system of tasks/operations, the processing constraints and the criteria.

Note that the above definition is more or less equivalent to that of *scheduling problem* in most (classical-oriented) texts. However, we want here to emphasize that we solve (real-world) scheduling problems first by modeling them (hence the need of scheduling models), second by finding methods that provide solutions for these models—and not for the real-world problem—, and third by transferring these solutions to the real world (see Chap. 1).

A scheduling model can be further concretised by selecting specific (i.e. numerical) values for the resources, tasks and rest of elements that make the model. This specific selection is termed an *instance* of the scheduling model.

In the following sections, we will provide formal definitions for the different elements constituting the scheduling model: The system of taks/operations is discussed in Sects. 3.2.2 and 3.2.3, while the processing constraints are presented both in the previous sections as well as in Sect. 3.2.4. Finally, a hint on scheduling criteria is given in Sect. 3.2.5.

3.2.2 Jobs and Machines

For the time being, we will assume that a scheduling model is determined by a known, finite and deterministic number of jobs that have to be processed on an equally known number of machines. Although we have already discussed in Chap. 1 that all these assumptions are not easily and readily acceptable in all situations, we will start with these simplest cases in order to gain the knowledge required for addressing more complex settings.

More specifically, we will assume that there is a set N of jobs that are consecutively indexed $N = \{1, 2, \ldots, n\}$. Therefore, there is a total of n jobs to be processed. Subindices j and k will be used to refer to jobs in the set N. Similarly, there is a set M of m machines or productive resources, that are indexed as $M = \{1, 2, \ldots, m\}$. We will be using mainly the subindex i to refer to any machine in the set M. If not stated otherwise, machines and jobs are independent and all the following data is deterministic, and known a priori:

- Task or Operation (O_{ij}). Each job $j \in N$ has a predefined number of operations or tasks. These tasks are at least carried out in 1 machine each. Therefore, O_{ij} denotes the task on machine i of job j. It is also common to refer to tasks simply as (i, j).[1] Furthermore, each operation has to be scheduled, this is, each operation will end up with a start and finish (end) time as a result of the production scheduling. We will refer to those times as SO_{ij} and EO_{ij}, respectively.
- Processing route. Each job has a predefined route throughout the production floor. In our terms, we will denote as R_j to the ordered list of machine visits. For example, this list could be (2, 4, 5, 1, 3) for a given job, meaning that the job has five different operations and that it has to visit first machine 2, then machine 4 and so on until the last machine 3. A discussion on the reasons for the assumption of a processing route is given in Chap. 1).
- Processing time (p_{ij}). This is the known span of time that is needed at machine i to process job j. Machine i will be busy during this period processing job j. This processing time is associated with the operation or task O_{ij}. Usually, it has to be satisfied that $p_{ij} \leq EO_{ij} - SO_{ij}$. Notice that the previous expression does not need to be necessarily equal as the processing of a job could be interrupted several times as per example, long jobs spanning through many days being stopped during the night shifts.
- Release or ready dates (r_j). Those are instants in time from which jobs might start. They also denote the first point in time where the job is available. They are helpful when modelling earliest starting times due to availability of raw materials or order placement time, just to name two possible examples. Jobs enter the production shop not earlier than r_j. This implies that $\forall j \in N, i \in M, SO_{ij} \geq r_j$. In other words, no

[1] Note that, in order to keep the complexity at minimum, this notation excludes the situation where a job has to visit the same machine more than once.

task of job j might start before r_j. In some scenarios, each task might have a release date r_{ij}, in such cases then it has to be satisfied that $\forall j \in N, i \in M, SO_{ij} \geq r_{ij}$.

- Follow-up time (fu_j). While release dates refer to the beginning of processing of a job (or even a single operation), a job or an operation may also have, after being finished on the respective machine, a *follow-up time* where it does not require any additional resources or machines but is not able to be processed further. (Such settings occur, e.g. in cooling or aging processes). We will see later that these times are many times modeled as waiting times or lags.
- Delivery or due dates (d_j). These are instants in time at which jobs have to be ideally completed. They model the delivery dates agreed with customers and constitute a compromise. Therefore, the last operation of each job should be completed before d_j. Usually, due dates are not fully mandatory and can be violated, subject to a penalty or cost. Exceeding the due date, i.e. finishing tardy, usually is connected to some penalty, e.g. by reduction of earnings.
- Mandatory due dates or deadlines ($\overline{d}_j$). Contrary to due dates, deadlines cannot be violated and it is mandatory, due to many possible reasons, to finish the job before $\overline{d}_j$. Therefore, $\forall j \in N, i \in M, EO_{ij} \leq \overline{d}_j$. As we will see in later chapters, the presence of deadlines may cause infeasibilities in scheduling problems as it might be physically impossible to satisfy all deadlines of a set of jobs in a given production environment.
- Due windows. An extension of the concept of due dates are the so-called *due windows*, i.e. a time interval where a job should be delivered. This time interval might be different per job or all or some jobs have a common due window. These windows can be modelled as $[d_j^-, d_j^+]$ where d_j^- and d_j^+ denote the start and finish of the delivery window, respectively.
- Costs, priorities, importance or weights (w_j). This figure, measured in any desired scale, models the importance or priority (as well as many other possible indicators) of each job. With the weights it is easy to establish a relative hierarchy among the set of n independent jobs. This weight could be established as a function of the total production cost of the job, importance of the client that placed the order, magnitude of penalty for finishing the job beyond its due date or many others.
- Release dates for machines (rm_i). Similar to the job release dates, machines might not be available before a point in time. This implies that $\forall i \in M, j \in N, SO_{ij} \geq rm_i$ Usually, this is much more complex as machines are seldom considered to be continuously available from rm_i. As a matter of fact, machines might have unavailability windows due to, for example, preventive maintenance or even unexpected breakdowns.

As we can easily see, the previous list is just a meagre extract of the sheer amount of data that could be associated with jobs and/or machines (for a more exhaustive list, see e.g. Pinedo 2012). More information would be added in subsequent sections and chapters whenever needed.

3.2.3 Processing Layouts

With the previous data, and among all possible classifications, by far, the scheduling literature is sectorised according to how the different machines in the set M are laid out and also according on how the different job processing routes R_j are specified. The most typical layouts are discussed next, together with some examples of schedules. Unless stated otherwise, we assume in the examples that the jobs are readily available for processing and that preemption is not allowed, for reasons of simplicity.

3.2.3.1 Single Machine Layout

As the same implies, the plant layout is formed by a single machine. Therefore, each job has to be processed exactly once on that machine. So there are really no processing routes.

Single machine models are relatively simple and are often viewed as too theoretical. However, they can be seen as special cases of other more complex layouts and therefore their study is of significant interest. In addition, we have already discussed in Chap. 1 that a single machine can adequately represent, under certain circumstances, larger real-world entities within the shop floor, or even the whole shop floor.

The flow layout schematics of a single machine model is clear, jobs enter the machine, each one of them keeping it busy for at least p_j processing time units (note that the processing times p_{ij} are simplified to just p_j in this case) and leave the machine once finished. The schematics are shown in Fig. 3.1.

Picture an example with six jobs, whose processing times are given in Table 3.1.

In a single machine model, each possible semi-active schedule (see Sect. 1.5.2) can be represented by simply specifying the sequence in which the jobs are to be processed in the machine. For instance, if we assume the following sequence $\pi = (1, 2, 3, 4, 5, 6)$, the resulting Gantt chart (see Sect. 1.5.1) of this single machine example is shown in Fig. 3.2.

Since in semi-active schedules, all jobs are processed without interruptions on the single machine, $SO_1 = 0$ and $EO_1 = SO_1 + p_1 = 0 + 8 = 8$ for this sequence. Similarly, $SO_2 = EO_1 = 8$ and $EO_2 = SO_2 + 4 = 12$. After all, it is easy to see that the machine is going to be busy during $\sum_{j=1}^{6} p_j = 42$ units of time. As a matter of fact, and as will be discussed later, the order of the jobs in the single machine setting is not going to affect the total busy time of the machine.

3.2.3.2 Parallel Machine Layout

After single machines, the straightforward extension to increase production capacity, is to replicate the single machine into several parallel machines. The processing routes for the jobs are equally simple: each job has to be processed in one out of the m available parallel machines. Usually, all m parallel machines are eligible for the jobs.

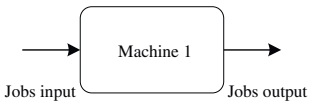


Fig. 3.1 Flow layout schematics of a single machine

Table 3.1 Processing times (p_j) of the six jobs for the single machine example

Machine (i)	Job (j)					
	1	2	3	4	5	6
1	8	4	10	5	3	12

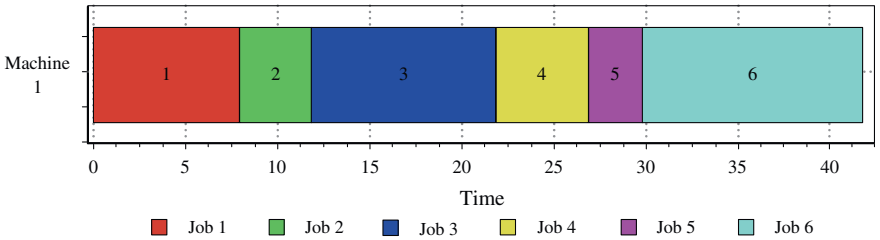


Fig. 3.2 Gantt chart with the result of the sequence $\pi = (1, 2, 3, 4, 5, 6)$ for a single machine scheduling example with the processing times of Table 3.1

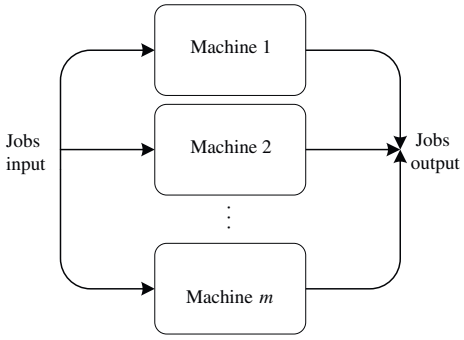


Fig. 3.3 Flow layout schematics of a parallel machine layout

Parallel machine models are interesting since a second dimension arises in the scheduling model. Again, if we assume semi-active schedules, a schedule can be represented by the assignment of each job to each machine plus the specific sequence of processing the jobs for each machine.

The flow layout schematics of a parallel machine is also simple. Jobs enter the shop, are assigned to one out of the m machines, processed and then leave the shop. The schematics are shown in Fig. 3.3.

Table 3.2 Processing times (p_{ij}) of the eight jobs and three unrelated parallel machines example

Machine (i)	Job (j)							
	1	2	3	4	5	6	7	8
1	37	25	25	11	48	68	28	28
2	44	42	20	27	56	59	12	39
3	44	66	58	87	53	41	47	76

Parallel machine models are further divided into three categories:

1. Identical parallel machines. Each machine is identical as regards the processing of the jobs. This means that the processing time of each job (p_j) does not depend on the machine to which it is assigned to. These models are easier since, much of the time, the assignment problem can be reduced to assigning each job to the first available machine.
2. Uniform parallel machines. This setting is also called machines with different speeds. Under this scenario, some machines are slower or faster by a constant ratio for all the jobs. Therefore, a fast machine is actually faster for all the jobs. In order to calculate the processing times, each machine is assigned a speed ratio v_i. Then, the processing time of job j for that machine is calculated as $p_{ij} = p_j/v_i$. Therefore, v_i values greater than one represent faster machines whereas v_i values lower than one are given to slower machines. This uniform parallel machine model can be reduced to the identical parallel machine case if $\forall i \in M$, $v_i = c$, where c is any constant value.
3. Unrelated parallel machines. This is the most general case of the three. Machines are assumed to be different from one another. This might be helpful when modelling production shops where machines that serve the same purpose have been purchased and installed at different dates and therefore, different machine versions coexist in the shop. It is expected that each machine might perform differently in relation to the job. Under this scenario, the processing time of a given job depends on the machine to which is assigned to, and, as a result, the input data is a processing time matrix p_{ij}.

Let us exemplify one unrelated parallel machine model. We have eight jobs to be processed on three unrelated parallel machines. The processing time matrix is given in Table 3.2.

Now let us picture a processing sequence. Jobs 4, 2 and 8 have been assigned, in this sequence, to machine 1. Similarly, jobs 7, 3 and 1 have been assigned to machine 2 and jobs 6 and 5 to machine 3. The resulting Gantt chart can be observed in Fig. 3.4.

3.2.3.3 Flow Shop Layout

Flow shops (often, this term is spelled together as flowshop, we will use both forms interchangeably) are the first of a series of the so-called 'shop' layouts, which include

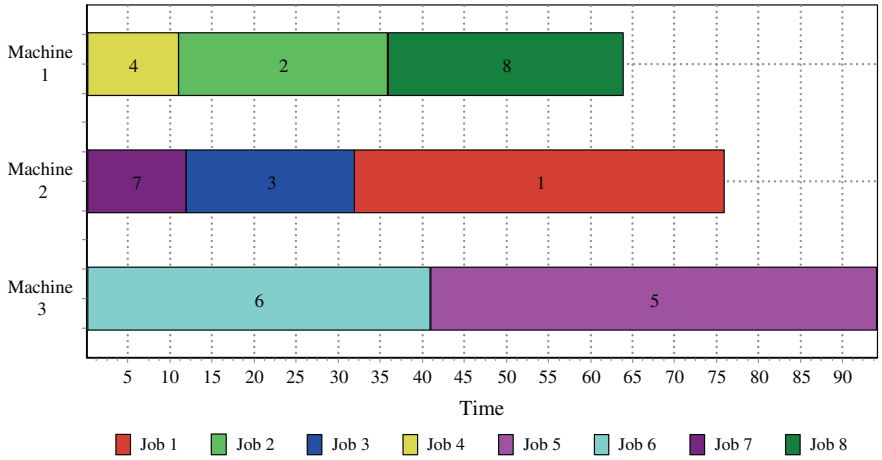


Fig. 3.4 Gantt chart with the result of the assignment and sequencing of eight jobs on three parallel machines

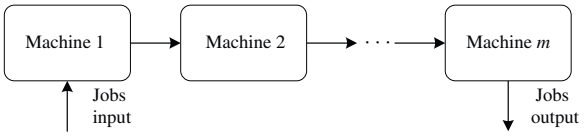


Fig. 3.5 Flow layout schematics of a flow shop

flow shops, job shops and open shops (these last two will be object of study in the following sections).

In a flow shop, there are m machines, but instead of being arranged in parallel, they are organised in series. It is assumed that each machine serves a different purpose. Every job has to visit all machines, in the same specified order. This order, without loss of generality, can be assumed to be $1, 2, \ldots, m$. As the order is the same for all jobs, we have that $\forall j \in N, R_j = (1, 2, \ldots, m)$. Furthermore, p_{ij} denotes the time job j needs at machine i. The term 'flow' refers to the fact that all jobs flow from one machine to the other in the same order, this is clearly depicted in the flow layout schematics of Fig. 3.5.

Note that the order of processing of each job on each machine is, in general, different. Therefore, each semi-active schedule can be represented by giving, for each machine, the sequence to process the jobs. In many cases, and mostly for reasons of simplicity in scheduling and shop floor control, it is assumed that the processing sequence is the same for all machines. This special case is referred to as the permutation flow shop model. Clearly, in a permutation flow shop model, each semi-active schedule can be represented by a single sequence of the jobs, which means that the maximum number of possible (semi-active) schedules in a permutation flow shop is reduced to $n!$ as compared to the $(n!)^m$ in the (general) flow shop model.

Table 3.3 Processing times (p_{ij}) of the five jobs and four machines permutation flow shop model example

Machine (i)	Job (j)				
	1	2	3	4	5
1	31	19	23	13	33
2	41	55	42	22	5
3	25	3	27	14	57
4	30	34	6	13	19

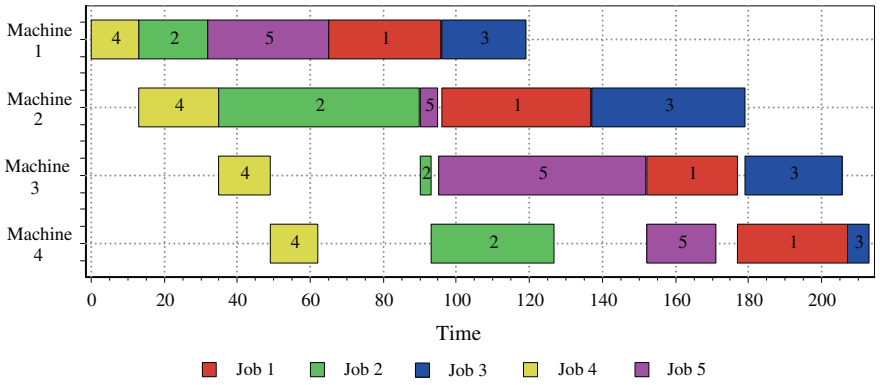


Fig. 3.6 Gantt chart with the result of the sequence $\pi = \{4, 2, 5, 1, 3\}$ for a permutation flow shop model with the processing times of Table 3.3

Due to its higher simplicity, let us start with an example of a permutation flow shop with four machines disposed in series and five jobs to schedule. The processing times are given in Table 3.3. Assuming the sequence $\pi = (4, 2, 5, 1, 3)$, the resulting Gantt chart is provided in Fig. 3.6.

Notice how some idle times start to appear on machines. This is different from the two previous single and parallel machine problems that have been depicted in Figs. 3.2 and 3.4 where machines were continuously busy. In Fig. 3.6, we see that the first machine is never idle from start of its first operation until the finishing time of its last operation, as all the jobs are being 'launched' to the shop one after another. However, and at least while the first job in the sequence is being processed on the first machine, all other machines are left waiting. Other notable idle times occur at machines three and four after processing the first job (4) of the sequence. Those large idle times obey to the fact that job (2), the second in the sequence, has a very large processing time on the second machine (55) and machine 3 has to wait until this job is completed in the previous machine.

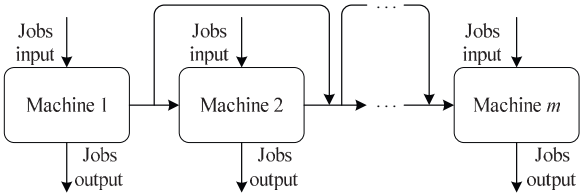


Fig. 3.7 Flow layout schematics of flexible flow shop

In the example, we have done some assumptions:

- At any time, each machine can process at most one job and each job can be processed only on one machine.
- An infinite in-process storage buffer is assumed. If a given job needs an unavailable machine, then it joins a queue of unlimited size waiting for that machine.
- Transportation time of jobs between machines is negligible and assumed to be zero.

These assumptions (together with the availability of jobs and no preemption already mentioned in the beginning of this section) are rather common in the scheduling literature, also for other shop-like layouts (see Baker and Trietsch 2009, among many others). These aspects as well as other constraints will be discussed in Chap. 4. The resulting model, although more realistic in the sense that different production facilities disposed in series are modelised, is rather theoretical. As a matter of fact, it has received a number of criticisms, most notably the ones of Dudek et al. (1992) or MacCarthy and Liu (1993). Nevertheless, the permutation flow shop model has been thoroughly studied in the past and it is still a very active and fruitful field of study nowadays.

One interesting variant of the permutation flow shop model is the so-called *flow line* in which all jobs follow the same relative route visiting the machines but some machines might be skipped by some jobs. The skipped machines can be seen as optional processing of treatments that are not necessary for all jobs. A schematic layout is given in Fig. 3.7.

As mentioned before, the (general) flow shop does not preclude the possibility of changing the processing sequence from machine to machine. Figures 3.8 and 3.9 show a simple example of a three machine, four job Gantt chart example of the same schedule on permutation and a (general or non-permutation) flow shop, respectively.

Notice how in the non-permutation version of Fig. 3.9, the sequence of the first machine $\pi_1 = (2, 3, 1, 4)$ changes in the second and third machines to $\pi_2 = \pi_3 = (2, 1, 3, 4)$. In this example, we would say that job 1 has passed job 3 on machines 2 and 3. As already mentioned, the (non-permutation) flow shop problem is more complex than the permutation version. Clearly, all permutation flow shop solutions are included in the set of solutions of the general flowshop. Therefore, the optimal solution of the general flow shop cannot be worse than the one from the permutation

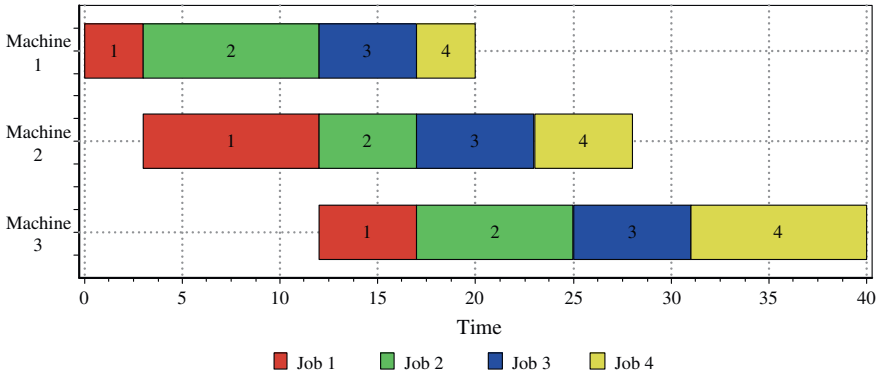


Fig. 3.8 Gantt chart of a 3 machine, four job permutation flow shop

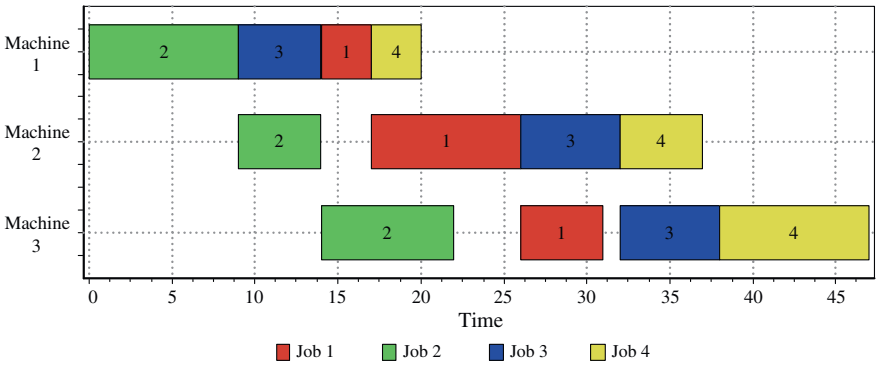


Fig. 3.9 Gantt chart of a 3 machine, four job (non-permutation) flow shop

problem. However, it might very well be that some of the non-permutation solutions are worse than some of the permutation solutions.

3.2.3.4 Job Shop Layout

The job shop is the second type of the so-called shop environments. Similar to flow shops, in a job shop there are m machines disposed in series. However, there is a big difference as compared to the flow shop: Every job has a potentially different route. For example, given 3 machines and four jobs, we could have $R_1 = (1, 2, 3)$, $R_2 = (2, 1, 3)$, $R_3 = (3, 2, 1)$ and $R_4 = (2, 3, 1)$. Job shops are very frequent in practice when modelling shops where each order is different from the others and requires a different machine routing. In most theoretical job shop research, the jobs visit all machines exactly once, with no missing machines and no re-visitation of

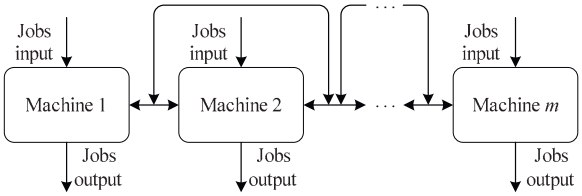


Fig. 3.10 Flow layout schematics of a job shop with machine skipping and recirculation

Table 3.4 Processing routes (R_j) for the five jobs in the job shop example

Job (j)	R_j			
1	1	2	3	4
2	4	2	1	3
3	3	1	2	4
4	2	4	3	1
5	4	1	3	2

machines, although in Fig. 3.10 we give a (more realistic) flow layout schematics including these aspects.

Let us picture also one example of a job shop. We have five jobs and three machines. We take the processing times of the previous Table 3.3 and consider the processing routes of the five jobs as indicated in Table 3.4. With these data, we can construct one possible schedule as given in the Gantt chart of Fig. 3.11.

Job shops are much more complex than flow shops as these last problems are special cases when all R_j are identical. Notice how the possibilities for improving a sequence are endless. For example, in the last Fig. 3.11, there is a large idle time on machine 3 between jobs 1 and 2. It would be better to start task O_{35} around time 110, just when task O_{15} finishes, delaying a bit O_{32} but in turn also advancing O_{25}.

3.2.3.5 Open Shop Layout

The open shop is the more general and most complex of the shop layouts. Again, as in the flow and job shops, there are m machines disposed in series and n jobs that have to visit all m machines. However, the big difference is that the processing route for each job is not fixed as is to be determined in the scheduling process. From another perspective, it can be assumed that all R_j for the jobs are arbitrary and that operations of a job can be processed in any order on the machines. Figure 3.12 shows a layout schematics of an open shop.

It is easy to see that flow shops and job shops are special cases of the open shop. One way to see this is to consider the enormous number of possible solutions that could be obtained following the example of Fig. 3.11 knowing that the only thing that has to be satisfied is that operations of the same job cannot be overlapped and that

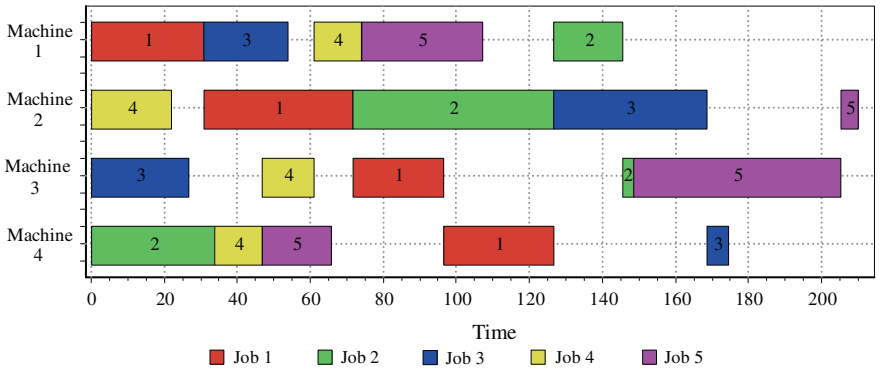


Fig. 3.11 Gantt chart with a possible sequence in a job shop with the processing times of Table 3.3 and the processing routes of Table 3.4

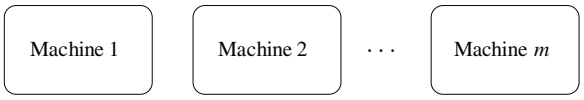


Fig. 3.12 Flow layout schematics of an open shop

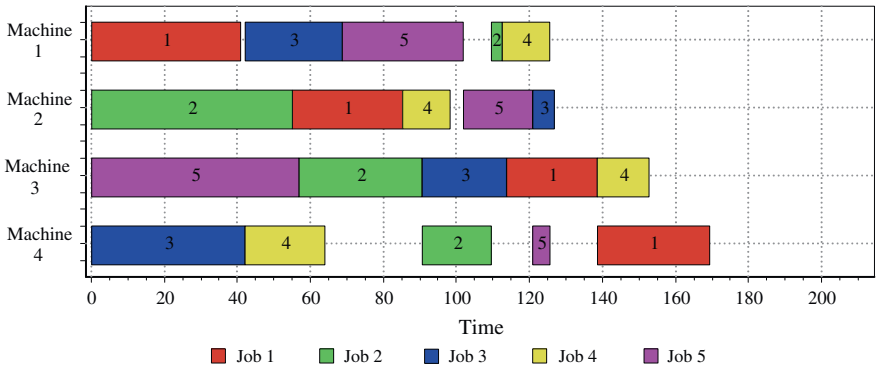


Fig. 3.13 Gantt chart with a possible sequence for an open shop with the processing times of Table 3.3

machines cannot process more than one job at the same time. One possible sequence, following the processing time data of Table 3.3 is given in Fig. 3.13.

As can be seen, the open shop is very flexible. The schedule has been 'squeezed' and there are less idle times in the machines. This is because of the total routing flexibility.

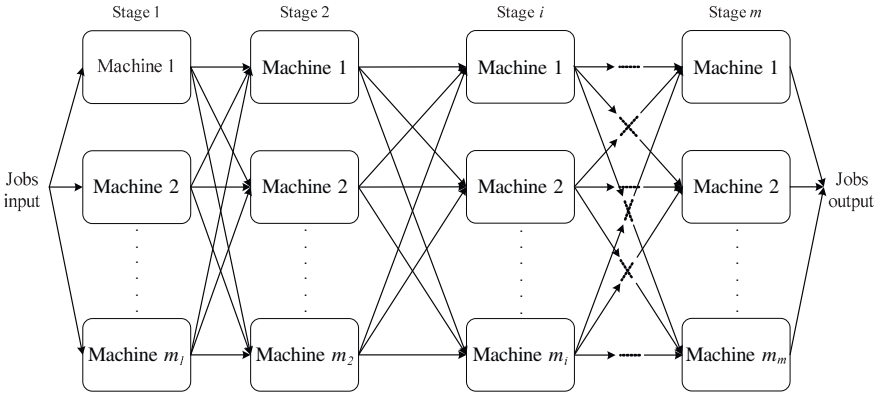


Fig. 3.14 Flow layout schematics of a hybrid flow shop

3.2.3.6 Hybrid Layouts

Normally, at production plants, important machinery is replicated. This was already commented in the parallel machine layout. However, in parallel machines, there is only one task per job. More general layouts are those that join the nature of the parallel machines and shop environments. Instead of machines, what we have in these hybrid layouts are production stages, each one consisting of a number of machines disposed in parallel. More specifically, there are m stages, each one with m_i parallel machines, where these machines can be identical, uniform or unrelated. The number of parallel machines can vary from stage to stage and all possible shop routings are possible. Therefore, we can have, for example, a hybrid job shop with three stages where at the first stage there are two identical parallel machines, four unrelated parallel machines at the second stage and a single machine at the last stage. In Fig. 3.14, we depict the layout schematics of a *hybrid flow shop*, which is a consecutive series of parallel machine layouts.

Hybrid layouts add the machine assignment dimension into the scheduling model as for each job and stage, an assignment decision to machines should be made. Let us give an example. We have a hybrid flow shop problem with three stages and 3, 4 and 3 machines respectively. There are five jobs to be processed. The processing times are given in Table 3.5.

As can be seen, the parallel machines at each stage are unrelated, as it does not take the same processing time to process job one on machines one, two or three of the first stage, for example. Now we need job assignments to machines at each stage and also the sequences for the jobs assigned to each machine at each stage. These are given in Table 3.6. The resulting Gantt chart can be seen in Fig. 3.15.

Several interesting conclusions can be drawn from this example. One may think that, since the parallel machines at each stage are unrelated, good schedules will consist on assigning each job to the fastest machine at that stage. However, this

Table 3.5 Processing times (p_{ij}) of the five jobs, three stages and 10 machines in a hybrid flow shop

Stage (i)	Machine (l)	Jobs (j)				
		1	2	3	4	5
1	1	27	45	56	40	30
	2	80	85	25	12	75
	3	44	17	39	29	25
2	4	25	80	12	16	25
	5	39	45	97	24	88
	6	44	71	28	25	96
	7	38	26	10	27	44
3	8	93	30	67	29	10
	9	45	63	66	88	89
	10	20	80	35	15	25

Table 3.6 Job sequence for each machine in the five jobs, three stages and 10 machines hybrid flow shop scheduling example

Stage (i)	Machine (l)	Job Sequence	
1	1	1	
	2	4	3
	3	2	5
2	4	1	5
	5	4	
	6		
	7	2	3
3	8	2	5
	9	1	
	10	4	3

greedy approach will not, in general, result in the best solution. Consider that the objective is to finish the set of jobs as soon as possible. The fastest machine for job 1 at the third stage is machine 10 as $p_{10,1} = 20$. However, job 1 has been assigned to machine 9, which has more than double the processing time. This is because jobs 3 and 4 have been already assigned to machine 10 (the fastest machine for them) and machine 10 is already overloaded. Therefore, if we want to finish job 1 as soon as possible, a slower machine has to be employed at the third stage. A second conclusion is also that machine 6 in the second stage is not used for this schedule. This is because the second stage has more machines than are actually needed.

3.2.3.7 Other Layouts

The previous layouts are not, by a long shot, the only possible ones. They should be seen as a broad classification inside which, with some degree of abstraction, some

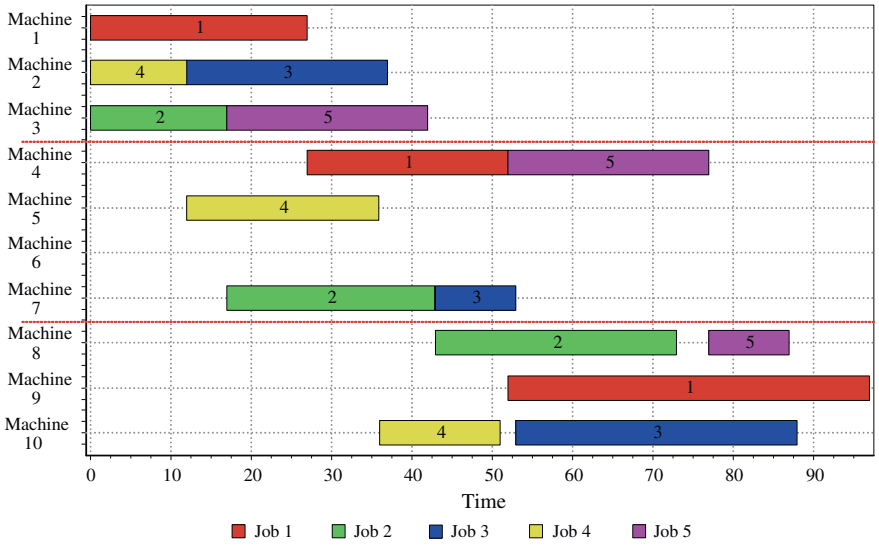


Fig. 3.15 Gantt chart with the a possible sequence for a hybrid flow shop problem with the processing times of Table 3.5 and the job sequences of Table 3.6

production environments could fit. Some other prototypical environments can be briefly summarised as follows:

- Manufacturing cells. They are a mixture between the high flexibility of the job shop and the mass production oriented low cost flow shop. Often referred to as cellular manufacturing, results in sets of machines grouped into cells. Each cell produces one type of product or family of products. Most of the time, machines at each cell are numerically controlled and are highly automated and it is not unusual to have robotic arms or automatic handling systems to move products from machine to machine inside the cell. Properly implemented cells have shown to be more flexible and responsive than more traditional mass production lines, like flow shops. Some basic references about cellular manufacturing are Hyer and Wemmerlöv (2002), Irani (1999).
- Assembly shops. These layouts model the bill of materials structure of several products whose intermediate products have to be manufactured first. Therefore, tasks of some jobs have to be finished in order to a subsequence assembly task to begin, which uses the previous tasks as raw materials. We will later see that these assembly operations can be modelled by precedence relationships.
- Production lines. A production line is very well understood when picturing a car manufacturing line. Production lines are often called assembly lines. Here, the products move in a linear way along conveyor belts or along any other possible means of transportation. Operations are performed over the products as they move as there is no buffer in between operations. Often, the places at the production line where operations are performed are referred to as stations. In general, an

assembly line is a medium- to high-volume, low product variety version of the assembly shop. An assembly line need not to be linear. It can have several feeder lines at certain points. Assembly lines are used, e.g. for manufacturing aircrafts, automobiles, appliances and farm equipment (see Morton and Pentico 1993).

- Batch shop. Batch processing exists when a specific number of jobs can be produced simultaneously on one machine at the same time. The completion time of a batch will be determined by the longest processing time of a job in its batch. A batch is characterised by homogenous properties (of its sub-jobs) and is processed as a whole through the shop, sometimes with additional no wait or other constraints. Batch shops occur e.g. in refinery, chemical, semiconductor or other process industries. Batch operations can be found at ovens where several jobs are collected to guarantee a high level of utilisation of the oven.
- Flexible manufacturing systems. Under this category, we find a very large number of possible layouts. We have included them here as in all other mentioned layouts, machines are more or less fixed and no flexibility is allowed in the short or medium term. Flexible manufacturing systems or FMS in short, are engineered with flexibility in mind so that new product types, or rapid changes in specifications of the products can be performed easily. FMS systems are also characterized by routing flexibility or by allowing different ways of manufacturing for the same products. FMS systems are commonly employed in the production of small lots of highly personalized products.

3.2.4 Additional Processing Constraints

Layouts are subject to countless situations, constraints, characteristics and events of all types. In the following, we provide just a short list of the number of processing constraints that real problems might have:

- Jobs might not be independent among each other as there might be a Bill of Materials (BOM) structure as commented in the assembly shops.
- Machines are actually never continuously available. Aside from economic/demand reasons, programmed maintenance activities, random breakdowns or other failures might prevent machines to operate at any time.
- In many situations, the processing of jobs can be interrupted and resumed at a later time. Pre-emption is allowed in most production settings.
- Setup times occur frequently in practice as well as many other operations on machines that are not just processing times of jobs on machines. Mounting or unmounting jigs or fixtures, cleanings, start-up or stop-down delays, etc.
- Buffers and storage areas for holding in-process semi-finished jobs are not infinite. For bulky and/or large products, storage areas might be even more limiting than machine availabilities.
- Jobs might not be allowed to wait indefinitely in between machines. Cooling down of products that have to be treated while hot (like steel rods) or products that might

dry up or evaporate are clear examples. In some extremes, products might not be allowed to wait at all in between stages. Frozen food is a good example of such no waiting allowed scenarios.

- Similarly, a minimum waiting time in between operations might be necessary since products might be too hot to handle after a furnace or kiln operation, to name an example.
- Assuming that all data is known in advance and that this data is deterministic is likely to be unacceptable. In controlled environments and tested manufacturing systems, the time needed for a given operation of a product might be known and pretty much fixed. However, the nature of some production processes is stochastic and processing times (as well as any other data) might be hard to estimate or might come in the form of a statistic distribution with known parameters.
- Production floors are highly dynamic. This means that jobs arrive over time, release dates change, due dates might be re-arranged with clients, weights or importances might vary. Basically, every piece of information, even if originally known in advance, is subject to change.
- In-process inventory, or semi-finished products do not instantly move from one machine to the other. Transportation times have to be taken into account. Sometimes, transportation is carried out by simple conveyor belts, other times, complex robotic or automated guided vehicles are used to move the products inside production plants. As a matter of fact, transporting the products can be seen as a routing problem in complex plant layouts and large factories.
- Machines are operated by personnel. This has huge implications at so many levels. From work shifts or timetables, to skill levels, learning effects and others. Machines might operate at different speeds depending on the number of persons attending to them. Similarly, more than one machine might be supervised by a single person depending on the shop layout.
- Fabrication is not perfect. Frequently, reworkings, fixings and reintroduction of jobs in the lines occur.
- In today's globalised economy, factories are no longer isolated elements. Distributed manufacturing and complex supply chains bring implications that reach into the scheduling decisions at manufacturing plants. The number of possible situations in such cases is very large.
- Jobs might have utterly complex processing routes, even alternative processing routes that might even change the bill of materials in dependence of the chosen manufacturing route.
- Processing times, even if known, can be worked on. Assigning more specialised personnel or additional tooling or any other resource could shorten, for an additional cost, the processing times.

As a matter of fact, the previous list is just a glimpse of situations that may arise in manufacturing scheduling. These will be treated in larger detail in Chap. 4.

3.2.5 Scheduling Criteria

As already discussed in Sect. 3.2.1, another element required to define a scheduling model is to establish the criterion (or criteria) employed in such decision problem. Usual criteria are based on some broad categories that can be summarised as follows:

- Utilisation-based criteria. These objectives are concerned with making the most out of the available productive resources: maximising machine utilisation, minimising idle times, reducing changeover or setup times, maximising the number of completed jobs per unit of time, etc.
- Customer's satisfaction. These include performance measures related with the fulfilment of the due dates, or the number of jobs that are completed before their due dates d_j or the equivalent of minimising the number of late jobs. Some other related criteria are those in which it is desired to minimise the number of 'late units' or the amount of time jobs are completed beyond their due dates.
- Storage oriented. When storage is a serious concern, minimising the work-in-progress, minimising the queue lengths in between machines or other similar procedures is beneficial. In order not to store finished products for too long, it might be interesting not to complete jobs way before their corresponding due dates.
- Cost oriented. Large batches are usually more economical due to economies of scale. Many other policies can be enforced and decisions can be made in order to reduce production costs. Usually, cost objectives are related with the previous items, specially with throughput and storage.
- Just in time, lean manufacturing and others. Just-in-time production requires jobs to be completed exactly at their due dates, not before and not after. In lean manufacturing, special emphasis is put in reducing waste and non-value adding operations throughout production. Other criteria could come in the form of maximising the number of accepted jobs for production subject that those jobs can be completed before their due dates (i.e. deadlines), etc.

Most of the scheduling literature is concerned about a single criterion or objective, and most of the time is of the minimisation type. As it happens with the processing constraints, scheduling criteria are as varied and numerous as possible real-life manufacturing scheduling scenarios one might encounter. The previous itemized list is just a brief attempt at structuring the various measures. As stated, these performance measures, along as many others, will be discussed in more detail in Chap. 5. Later, the consideration of more than one criteria under multiobjective approaches will be introduced in Chap. 10.

3.3 Classification of Scheduling Models

As already noted in several parts in this book, there is a sheer variety of production layouts, processing constraints and criteria in scheduling models. The traditional way to handle the diversity of scheduling models has been to classify them using taxonomies and notations that allow establishing the similarities and differences among the different models. As we will discuss in Chap. 7, this classification is extremely useful to build solution procedures for scheduling models based on these of similar models. In the following sections, we present the two most popular classification schemes.

3.3.1 Early Classification Schemes

Taxonomies or classifications for scheduling models started to appear in the 1960s with the book of Conway et al. (1967). These first taxonomies try to classify production scheduling models according to their main characteristics already mentioned, namely the floor layout, the processing constraints and the performance measure studied. Conway et al. (1967) proposed a quadruplet $A/B/C/D$ where each capital letter has a more or less precise meaning:

- A indicates the number of jobs that have to be scheduled in a given problem. In other words, A indicates the number of jobs n.
- B describes m the number of machines in the shop floor.
- C describes the processing layout in a simplistic way:
  - If $C = 1$ the layout is formed by a single machine. In some other related works, the field A is omitted in the case of one single machine.
  - If $C = P$ then we have a parallel machines layout.
  - If $C = F$ the problem is a flow shop.
  - $C = J$ indicates a job shop.
  - Finally, if $C = O$ an open shop is dealt with.
- D serves for indicating the type of optimisation criterion for the scheduling model.

For example, a problem denoted as $10/1/1/D$ or simply $10/1/D$ represents a single machine problem where 10 jobs have to be scheduled under the unspecified criterion D (performance measures will be precisely detailed in Chap. 5). Another example could be $n/m/F/D$ which represents the typical flow shop problem with a general number of jobs n and m machines disposed in series and an unspecified criterion D.

It is easy to see that the previous classification is sorely incomplete. For example, there are no distinctions between the three types of parallel machines. Also, processing constraints are neglected as the notation does not allow for their inclusion. Hybrid layouts and/or more complex layouts do not have a corresponding letter or expression way. For all these reasons, in the early 1970s, other much more comprehensive

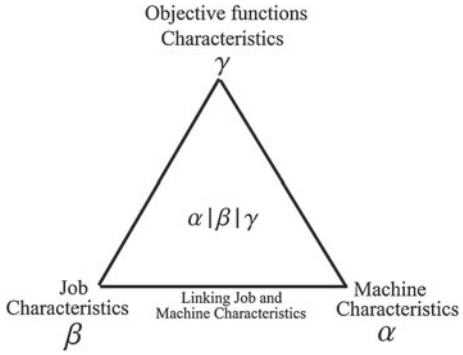


Fig. 3.16 $\alpha|\beta|\gamma$-classification of scheduling models

taxonomies or scheduling problem classification schemes were proposed. These are discussed in next section.

3.3.2 Current Classification Schemes

From the shortcomings of the previous classification of Conway et al. (1967), other authors proposed extensions, like RinnooyKan (1976) which proposed a notation based on the triplet $\alpha/\beta/\gamma$. This scheme was further extended and refined in the works of Graham et al (1979), Błazewicz et al. (1983), Lawler et al. (1993).

In the following, we highlight the most important aspects of this notation, which is pictured in Fig. 3.16.

- The first field in the triplet α defines the processing layouts. It is further split into two sub-fields $\alpha_1\alpha_2$ so that:

  $\alpha_1 = \varnothing$: single machine layout, sometimes also denoted as $\alpha_1 = 1$.
  $= P$: identical parallel machines.
  $= Q$: uniform parallel machines.
  $= R$: unrelated parallel machines.
  $= F$: flow shop.
  $= J$: job shop.
  $= O$: open shop.
  $\alpha_2 = 1, 2 \ldots, m$: fixed number of machines in the layout.
  $= \varnothing$: the number of machines is not fixed to a precise number (i.e. the problem studied is not limited to a case with a specific machine count). Often this is also referred to as simply m. Note that in single machine problems ($\alpha_1 = 1$), α_2 has no meaning.

  Therefore, with the α field, the machine layout can be easily defined. For example, $\alpha = P3$ means the considered layout is composed by three identical parallel machines. $\alpha = Fm$ or $\alpha = F$ represents a flow shop layout with any number of machines.

Note again that the previous α field suffers from some shortcomings as there is not a predefined way of representing hybrid layouts. The review paper of Vignier et al. (1999) introduced a significant extension. The authors split the α field into four sub-fields $\alpha_1\alpha_2\alpha_3\alpha_4$ as follows:

$\alpha_1 = HF$: hybrid flow shop.
$= HJ$: hybrid job shop.
$= HO$: hybrid open shop.
$\alpha_2 = 1, 2 \ldots, m$: the number of stages. Again, it could be a fixed number or a general number of stages m.
$\alpha_3 = P$: identical parallel machines at stage i.
$= Q$: uniform parallel machines at stage i.
$= R$: unrelated parallel machines at stage i.
$\alpha_4 = 1, 2 \ldots, m_i$: number of parallel machines at stage i.

Note that the two last sub-fields α_3 and α_4 are repeated for every stage i specified in the sub-field α_2. For example, a hybrid layout with $\alpha = HJ3, 1, P2, R3$ denotes a hybrid job shop with three stages where the first stage has one single machine, the second stage has two identical parallel machines and the third stage has three unrelated parallel machines. Another example could be $\alpha = HFm, \left((Pm_i)_{i=1}^{m}\right)$ which stands for a general hybrid flow shop with m stages where each stage i contains m_i identical parallel machines.

- The second field in the triplet notation, β indicates the processing constraints. This field is split into as many sub-fields as processing constraints there might be in the studied problem, separated by commas. As we exposed in previous sections, processing constraints are varied and numerous and are object of later study in Chap. 4, where we will define many possible β fields.
- Lastly, the third field in the triplet notation, γ, gives information about the single objective considered in the studied problem. As with the processing constraints, we have devoted a specific chapter to scheduling objectives which is Chap. 5. Furthermore, in Chap. 10 we will significantly extend the γ field in order to allow the notation of multiobjective problems.

We have to bear in mind that even with the hybrid layout extensions given by Vignier et al. (1999), it is simply not possible to capture every possible production scheduling layout. Not only this is not possible, but also it would be unrealistic to expect so. Notation schemes have to be succinct enough so to facilitate a quick and unambiguous identification of a layout. Complex layouts are simply approximated to the closest possible and still easily representable layout. Additional characteristics of complex layouts are given in plain text afterwards. As a result, other researchers and practitioners might grasp the most important aspects of the represented layout as if there are complex job routes, stages and so on.

3.4 Production Planning, Lot Sizing and Lot Streaming

In Sect. 1.5 we discussed already that the concept of a job in scheduling manufacturing, albeit probably clear in the scientific literature, is far from being a precise concept in real manufacturing. As a matter of fact, probably, the first barrier when having a scheduling practitioner at a company communicating with a scheduling researcher is to agree on what a 'job' is. Given the extreme importance of identifying this unit of information in order to build scheduling models, we devote this section to discuss first how jobs (as they are interpreted in the scheduling models presented in this chapter) result as the output of a production planning process (as it is described in Sect. 2.4) and, secondly, some alternatives (lot sizing and lot streaming) to aggregate/disaggregate job entities.

Although Sect. 2.4 has already provided additional insight on the planning process, production management is a large and broad field and it is outside the scope of this book and therefore and we will simply stress that, at least with respect to the implications over manufacturing scheduling, production planning is the process of obtaining a list of products to be manufactured, along with their quantities.

Some traditional production planning techniques, and more particularly, Materials Requirements Planning (MRP) suffer from several shortcomings that have been long documented in the literature and identified in practice. Some of these problems are:

- Data integrity. Unless high integrity of the data is assured, the result is unsatisfactory. Bill of Materials, inventories, orders and all related data has to be carefully checked in order to avoid what has been sometimes called a GIGO (Garbage In, Garbage Out) situation.
- MRP systems usually assume constant lead times and are inflexible as regards as other data like fixed reorder quantities, points, safety stocks, etc.
- The major problem is that the planning carried out by MRP systems does not take production capacity into account. Results might be impossible to implement.
- MRP typically does not issue early warnings for stock outs, demand spikes, shortages, etc.

Furthermore, some authors have criticised the MRP approach for its limited structure that basically breaks down the production demands into product families and later into individual products without actually considering the interactions between them and the underlying model structure. This myopic decomposition leads to important productivity and flexibility losses. Although more advanced techniques such as Manufacturing Resources Planning (MRPII) or Capacity Requirements Planning (CRP) overcome some of these problems, alternatives involving the use of linear and integer programming to model and to solve complex production planning problems in which high quality solutions (usually optimal) are of interest. One of the basic models is the well known Single-Item, Single Level, Uncapacitated Lot-Sizing

problem, denoted usually as *SLULSP*, which can be formulated as a simple MILP model considering holding, fixed and unit production costs for the product.[2]

Although simplistic, *SLULSP* model takes into account variable demands, variable holding, fixed and unitary production costs over the periods and therefore, even in its simplest form, surpasses the capacities of the outdated MRP methodology. Furthermore, it can be easily extended in several ways. Maximum storage capacities can be added. Similarly, maximum production capacities or maximum number of production runs over the periods are also easy to add. Other more complex production planning optimisation models include the multiple-item, single level, uncapacitated lot-sizing problem, or the more general multi-level variants.

From a manufacturing scheduling perspective, the output of such models provides the value of the variables indicating the quantity to be produced for each periods in the planning period for each type of product, thus identifying the 'jobs' (the amount of each type of product to be produced) to be scheduled by aggregating the items to be produced at the same time (lot sizing), together with their corresponding due dates (the times in which the amount has to be completed).

In the view discussed in the paragraph above, jobs would be actually made up of lots of several discrete and identical items, like for example lots of microprocessors in a semiconductor industry or lots of metal sheets in a metallurgy firm. Consequently, the whole lot will be considered a single unit of processing, as thus will have to be finished in the upstream machine before starting in the subsequent downstream machine. *Lot streaming* breaks this assumption by allowing the job to be broken down into 'sublots'. The sublots, once completed on the previous machine, move on to the next machine, effectively resulting in different sublots of the same product or job being simultaneously produced on different machines. Obviously, this 'overlap' may be beneficial in many ways. Lot streaming within the scheduling context is the short-term optimisation approach to the previous production planning models. Recall that a production planning period t is usually not shorter than a week and there is a lot of production time in a week available for scheduling. Lot streaming has many different variants and characteristics, like for example a limit on the number of sublots, equal or variable size sublots, continuous or discrete size sublots and many others. On the other hand, lot streaming may not be interesting due to economic/technical reasons if the subsequent process has an operating cost which is not depending on the lot size (here the classical example is that of a furnace). The effect of lot streaming in a flow shop layout, considering the previous example given in Fig. 3.6 is shown in Fig. 3.17.

As we can see in Fig. 3.17, machines are better utilised as when compared with Fig. 3.6. The effect is particularly interesting in job 4 which is in the first position of the sequence. Lot streaming allows for units of job 4 to be available as soon as after 25 units of time have elapsed. This is much shorter when compared to the first units that could be available at time 50 in the flow shop pictured in Fig. 3.6.

[2] Note that, in some references, processing costs are time-independent and consequently, are removed from the objective function since processing costs are decision-irrelevant.

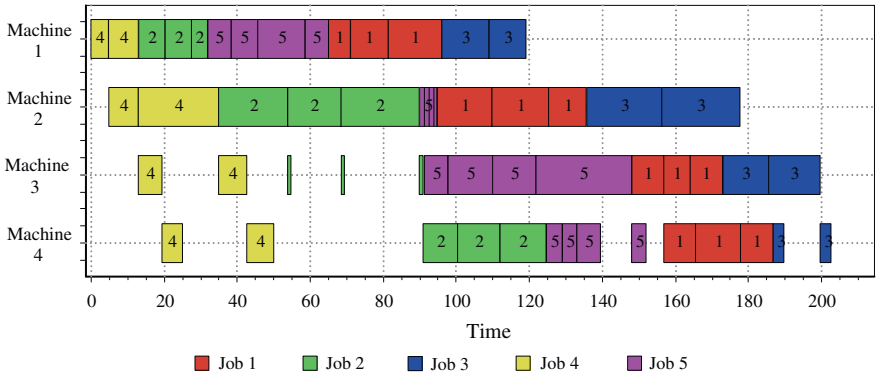


Fig. 3.17 Gantt chart for a streaming flow shop problem based in the example of Fig. 3.6

A final remark has to be done to discard the idea that, in principle, sublots could be simply modeled as different jobs. Although theoretically interesting, this approach quickly results in problems with a very large number of jobs (literally thousands) and it is therefore quite unpractical.

3.5 Conclusions and Further Readings

Manufacturing scheduling is a complex problem arising in many industries. As with any complex problem, a scientific approach is often needed in order to obtain satisfactory solutions. In this chapter, we have introduced the basic concepts of modelling as a necessity for scheduling problem solving, together with the main characteristics of models, types of models, common errors and shortcomings.

Next, some basic definitions for scheduling models have been provided, and special emphasis has been given to the processing layouts as defined in the rich scheduling literature. We have defined the layouts of single machine problems, parallel layouts, shop problems (flow, job and open shops) and the hybrid layouts often found in practice. A short introduction to other less frequent shop arrangements has also been given.

The main processing constraints and an introduction to scheduling criteria has been given in this chapter, although both will be significantly extended in the next two chapters, namely Chaps. 4 and 5. Of special interest are the links to real production scheduling problems not often highlighted elsewhere. Early as well as current scheduling model classification schemes, along with some extensions, have been introduced. Finally, we have devoted one section to discuss first how jobs (as they are interpreted in the manufacturing scheduling context) result as the output of a production planning process and, second, some alternatives (lot sizing and lot streaming) to aggregate/disaggregate job entities.

We have already noted that the definition of a scheduling model would more or less match what it is termed in most (classical-oriented) textbooks as a *scheduling problem*. Accordingly, in these sources the taxonomy presented in Sect. 3.3 is referred to a taxonomy for scheduling problems, and then the scheduling methods provide solutions to scheduling problems. However the need of the modeling and transference processes described in Sect. 3.2.1 is often overlooked in some sources, and the reader is given the impression that a scheduling model arises naturally from a (real-world) problem and that this problem is solved once a solution for the model has been obtained. Therefore, we have opted for somewhat abandoning the mainstream scheduling literature, but in our defense we can always state that our definition of scheduling model (and instance) is taken from a source as early as Coffman (1976).

Regarding further readings, the literature is full of comprehensive works on scientific modelling. Some examples are Gilbert (1991) or Mayer (1992) and also the classical book of Churchman (1984) or more modern volumes like DaCosta and French (2003). The formal definitions and notation given in Sects. 3.2 and 3.3 follow the mainstream scheduling literature, such as for example, Conway et al. (1967), Baker (1974), French (1982), Błazewicz et al. (2002), Brucker (2007) or more modern texts like Pinedo (2012), Baker and Trietsch (2009), Pinedo (2009). Production planning is a very rich field where lots of books and papers have been published. Some of these texts are Orlicky (1975), Plossl (1994), Wight (1995), Higgins et al. (1996), Toomey (1996) Artiba and Elmaghraby (1997), Drexl and Kimms (1998), Onwubolu (2002), Sheikh (2003), Voss and Woodruff (2003), Stadtler and Kilger (2005), Proud (2007). Specially interesting is the book of Pochet and Wolsey (2006), where the mathematical and optimisation aspects of production planning are studied in great detail. A good review paper for the Single-Item, Single Level, Uncapacitated Lot-Sizing problem is given by Brahimi et al. (2006). Regarding lot streaming, some basic references are Potts and VanWassenhove (1992), Trietsch and Baker (1993) or more recently, Chang and Chiu (2005). Additionally, in the case of the flow shop layout, there is a book published that solely deals with the lot streaming variant (Sarin and Jaiprakash 2007).

References

Artiba, A. and Elmaghraby, S. E., editors (1997). *The planning and scheduling of production systems: methodologies and applications.* Chapman & Hall, London.

Baker, K. R. (1974). *Introduction to Sequencing and Scheduling.* John Wiley & Sons, New York.

Baker, K. R. and Trietsch, D. (2009). *Principles of Sequencing and Scheduling.* Wiley, New York.

Błazewicz, J., Ecker, K. H., Pesch, E., Schmidt, G., and Węglarz, J. (2002). *Scheduling Computer and Manufacturing Processes.* Springer-Verlag, Berlin, second edition.

Błazewicz, J., Lenstra, J. K., and RinnooyKan, A. H. G. (1983). Scheduling Subject to Constraints: Classification and Complexity. *Discrete Applied Mathematics,* 5:11–24.

Brahimi, N., Dauzère-Pérès, S., Najid, N. M., and Nordli, A. (2006). Single item lot sizing problems. *European Journal of Operational Research,* 168(1):1–16.

Brucker, P. (2007). *Scheduling Algorithms.* Springer, New York, fifth edition.

Chang, J. H. and Chiu, H. N. (2005). Comprehensive review of lot streaming. *International Journal of Production Research,* 43(8):1515–1536.

Churchman, C. W. (1984). *The Systems Approach.* Dell Publishing Company, New York. Revised and upgraded edition from the 1968 original.

Coffman, E. G. (1976). *Computer & Job/shop Scheduling Theory.* John Wiley & Sons.

Conway, R. W., Maxwell, W. L., and Miller, L. W. (1967). *Theory of Scheduling.* Dover Publications, New York. Unabridged publication from the 1967 original edition published by Addison-Wesley.

DaCosta, N. and French, S. (2003). *Science and Partial Truth: A Unitary Approach to Models and Scientific Reasoning.* Oxford University Press, Oxford.

Drexl, A. and Kimms, A., editors (1998). *Beyond Manufacturing Resource Planning (MRP II): advanced models and methods for production planning.* Springer, New York.

Dudek, R. A., Panwalkar, S. S., and Smith, M. L. (1992). The lessons of flowshop scheduling research. *Operations Research,* 40(1):7–13.

French, S. (1982). *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop.* Ellis Horwood Limited, Chichester.

Gilbert, S. W. (1991). Model-building and a definition of science. *Journal of Research in Science Teaching,* 28(1):73–79.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Rinnooy Kan, A. H. G. (1979). Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. *Annals of Discrete Mathematics,* 5:287–326.

Higgins, P., Le Roy, P., and Tierney, L., editors (1996). *Manufacturing planning and control: beyond MRP II.* Springer, New York.

Hyer, N. and Wemmerlöv, U. (2002). *Reorganizing the Factory: Competing Through Cellular Manufacturing.* Productivity Press, Portland.

Irani, S. A., editor (1999). *Handbook of Cellular Manufacturing Systems.* Manufacturing & Automation Engineering. John Wiley & Sons, New York.

Lawler, E. L., Lenstra, J. K., and RinnooyKan, A. H. G. (1993). Sequencing and Scheduling: Algorithms and Complexity. In Graves, S. C., RinnooyKan, A. H. G., and Zipkin, P. H., editors, *Logistics of Production and Inventory,* volume 4 of *Handbooks in Operations Research and Management Science,* Amsterdam. Elsevier Science Publishers, B. V.

MacCarthy, B. L. and Liu, J. (1993). Addressing the gap in scheduling research: A review of optimization and heuristic methods in production scheduling. *International Journal of Production Research,* 31(1):59–79.

Mayer, R. E. (1992). Knowledge and thought: Mental models that support scientific reasoning. In Duschl, R. A. and Hamilton, R. J., editors, *Philosophy of science, cognitive psychology, and educational theory and practice,* pages 226–243, Albany. New York State University.

Morton, T. E. and Pentico, D. W. (1993). *Heuristic Scheduling Sysmtems With Applications to Production Systems and Project Management.* Wiley Series in Engineering & Technology Management. John Wiley & Sons, Hoboken.

Onwubolu, G. C. (2002). *Emerging optimization techniques in production planning and control.* Imperial College Press, London.

Orlicky, J. (1975). *Material Requirements Planning.* McGraw-Hill, New York.

Pinedo, M. (2009). Planning and *Scheduling in Manufacturing and Services.* Springer, New York, second edition.

Pinedo, M. L. (2012). *Scheduling: Theory, Algorithms, and Systems.* Springer, New York, fourth edition.

Plossl, G. W. (1994). *Orlicky's Material Requirements Planning.* McGraw-Hill, New York, second edition.

Pochet, Y. and Wolsey, L. A. (2006). *Production planning by mixed integer programming.* Springer, New York.

Potts, C. N. and VanWassenhove, L. N. (1992). Integrating scheduling with batching and lot-sizing: A review of algorithms and complexity. *The Journal of the Operational Research Society,* 43(5):395–406.

Proud, J. F. (2007). Master Scheduling: *A Practical Guide to Competitive Manufacturing.* Wiley, New York, third edition.

RinnooyKan, A. H. G. (1976). *Machine Scheduling Problems: Classification, Complexity and Computations.* Martinus Nijhoff, The Hague.

Sarin, S. C. and Jaiprakash, P. (2007). *Flow Shop Lot Streaming.* Springer, New York.

Sheikh, K. (2003). *Manufacturing resource planning (MRP II): with introduction to ERP, SCM and CRM.* McGraw-Hill Professional, New York.

Stadtler, H. and Kilger, C., editors (2005). *Supply chain management and advanced planning: concepts, models, software and case studies.* Springer, New York, third edition.

Toomey, J. W. (1996). *MRP II: planning for manufacturing excellence.* Springer, New York.

Trietsch, D. and Baker, K. R. (1993). Basic techniques for lot streaming. *Operations Research,* 41(6):1065–1076.

Vignier, A., Billaut, J.-C., and Proust, C. (1999). Les problèmes d'ordonnancement de type flow-shop hybride: État de l'art. *RAIRO Recherche opérationnelle,* 33(2):117–183. In French.

Voss, S. and Woodruff, D. L. (2003). *Introduction to computational optimization models for production planning in a supply chain.* Springer, New York.

Wight, O. W. (1995). *Manufacturing Resource Planning: MRP II: Unlocking America's Productivity Potential.* John Wiley & Sons, New York, second edition.

Chapter 4
Scheduling Constraints

4.1 Introduction

This chapter belongs to the part of the book devoted to scheduling models, one of the three elements (recall from Chap. 1 that the other two are methods and tools) that constitute an scheduling system. More specifically, this chapter discusses the constraints describing scheduling models. Already in Chap. 2 (Sect. 2.5), it was stated that manufacturing scheduling is subject to a large variety of constraints. These were refined in Sect. 3.2.4, where some constraints arising in real-life settings were introduced. This chapter gives further insight into this issue by bringing a characterisation and description of scheduling constraints. Since—as already mentioned in Sect. 3.3.2—the scheduling constraints are gathered in current scheduling classification schemes in the field β of the notation triplet $\alpha/\beta/\gamma$ the scheduling constraints, we will comprehensively work with the notation and with this field β in this chapter as well.

In order to provide a (hopefully) coherent view of scheduling constrains, we have necessarily simplified many of them, and broadly classified them into process, operations, transportation and storage constraints. Some other situations will be also addressed. Despite this classification, note that scheduling constraints are rarely independent from each other and that, even inside the same constraint, the treatment and the low-level details inevitably vary from one production floor to another. Last, it is to note that some constraints only make sense in some production layouts and/or under some specific criteria.

More specifically, in this chapter we

- describe constraints affecting the flow of operations in the shop floor (Sect. 4.2),
- introduce scheduling constraints that are more closely related to the operations or tasks of the jobs (Sect. 4.3),
- discuss restrictions attached to the movement of material in the shop floor (Sect. 4.4),

J. M. Framinan et al., *Manufacturing Scheduling Systems*,
DOI: 10.1007/978-1-4471-6272-8_4, © Springer-Verlag London 2014

- present constraints arising when considering that storage capacity is not unlimited (Sect. 4.5) and
- give some hints on further relevant constraints (Sect. 4.6).

4.2 Process Constraints

Under process constraints we list all situations that affect the otherwise 'standard' flow of operations in single, parallel or shop environments.

4.2.1 Job or Task Precedence Constraints

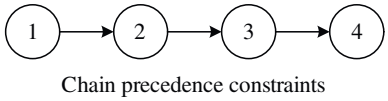
Precedence constraints started to be studied very early in the scheduling literature with some pioneering work in the early 1970s of Lawler (1973), Sidney (1975) and the already mentioned one of Lenstra and Rinnooy Kan (1978). Nowadays, it is still a very hot topic of research within the scientific community where literally hundreds of papers are being published with new results for many different production layouts. This type of scheduling constraint breaks the common assumption that all jobs to be scheduled and to be processed on machines are independent. It is very common for jobs to model actual products that have a Bill of Materials (BOM) structure. While all the items in the BOM might be just raw materials that are purchased from third companies, it is easy to find real cases where these products are intermediate items that also have to be processed.

Precedence constraints, in their simplest form, dictate that the first task or operation of a job cannot be started until the last job or operation of all of its predecessors are completed. The usual notation is the following: $\beta = prec$: Precedence relations exist among jobs. In the scientific literature, precedence constraints are divided into four different categories:

1. Chain precedence relations ($\beta = chains$), where each job might have, at most, one predecessor and at most one successor.
2. Intree precedences ($\beta = intree$), where each job might have many predecessors but, at most, one successor.
3. Outtree precedences ($\beta = outtree$), where each job might have, at most, one predecessor and many successors.
4. Tree precedences ($\beta = tree$), where there is no limit in the number of predecessors and successors.

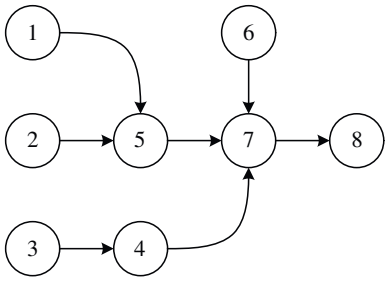
These categories are depicted in Figs. 4.1 through 4.4.

Precedence constraints are not very intuitive. At a first glance, one could think that having them in a problem simplifies things. In some sense, this is true. For example, a five job, single machine problem will have $5! = 120$ possible different job processing sequences. This number drastically decreases as precedence constraints



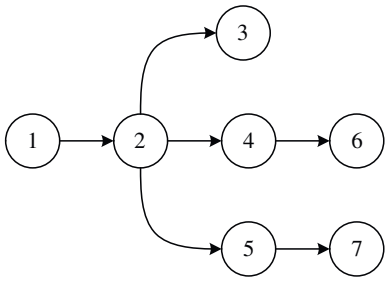
Chain precedence constraints

Fig. 4.1 Job precedence graph example. Chain precedence constraints



Intree precedence constraints

Fig. 4.2 Job precedence graph example. Intree precedence constraints



Outtree precedence constraints

Fig. 4.3 Job precedence graph example. Outtree precedence constraints

are added. The most extreme case would be when every job has a predecessor except the first, for example, $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$. In this situation, only the sequence $\pi = (1, 2, 3, 4, 5)$ would be feasible. However, precedence constraints complicate matters entirely. Picture for example a parallel machines layout. This layout was presented in Sect. 3.2.3.2. There an example problem was introduced in Table 3.2 and pictured in the Gantt chart of Fig. 3.4. Now, let us assume that $7 \rightarrow 4$, $7 \rightarrow 3$ and $2 \rightarrow 1$. The resulting Gantt chart is shown in Fig. 4.5.

As we can see, idle times appear on machines. This might be seen as not a big deal, but in most parallel machine problems with no constraints, machines are not needed to lay idle, if we only consider semi-active schedules. As a matter of fact, the regular parallel machine problem under some performance criteria, is basically reduced to an assignment problem. Although more theoretically oriented, authors like Lenstra and Rinnooy Kan (1978) pointed out already long ago, that adding precedence constraints makes relatively simple problems much more difficult.

Precedence constraints also allow to capture more complex relationships. For example, the precedences might be among tasks of different jobs (as in a flow shop) and not only between the last task of a job and the first task of the succeeding job.

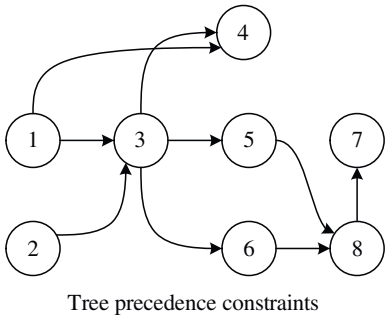


Fig. 4.4 Job precedence graph example. Tree precedence constraints

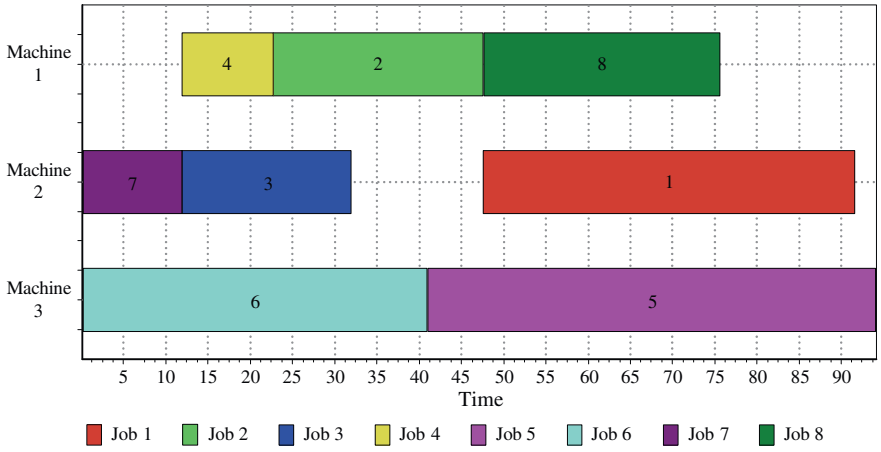


Fig. 4.5 Gantt chart with the result of the assignment and sequencing of eight jobs on three *parallel* machines with some precedence constraints

Precedences do not necessarily have to be what is called 'end-start' as some more complex situations could be tasks that have to start at the same time ('start-start') or finish at the same time ('finish-finish') and many others.

4.2.2 Changeovers or Setup Times

Generally speaking, operations carried out at machines that are not directly related with the processing of the jobs are commonly referred to as 'changeovers'. Changeovers include, but are not limited to, adjustments, changes, cleaning, testings, re-tooling, fixing or removing jobs, etc. Changeovers are also commonly known as setup times since they model the time that is needed to setup a machine prior and/or after processing a job.

Evidently, as changeovers model many different situations, a complete classification and taxonomy of the different setup times is clearly out of the scope of this book. However, and broadly speaking, there are some possible distinctions. First, setup times can be independent of the job sequence or dependent of the job sequence. Setup times are job sequence dependent if the amount of setup time depends both on the job that was just processed on the machine and on the next job to be processed in the sequence. After processing a job, the machine ends up in a 'configured state' suitable for the just finished job. The amount of needed reconfiguration or setup might be dependent of the type of job to be processed next. For example, if the next job to be processed is essentially similar in type to the previous one, only minor setups might be needed. However, if a completely different job is to be processed, large reconfigurations might be necessary.

Second, setup times might be anticipatory (also referred to as separable or detached setup times) or non-anticipatory (inseparable or attached). Anticipatory changeovers or setups are those that can be performed at the machine as soon as this machine finishes the previous job in the sequence. Note that due to the nature of the expected processing sequence, there might be an idle time in between processing of consecutive jobs. Therefore, an anticipatory setup can be carried out at any time during this idle period. A machine cleaning, for example, is a clear case of an anticipatory setup time. Conversely, non-anticipatory setup times require the next job in the sequence to be already present or to have arrived at the machine in order to carry out the setup. For example, imagine the process of adjusting an unpolished wood board in a polishing machine. This adjustment might require the wood board (job) to be already present in order to firmly fix it (setup) to the machine. If the wood board is not present, the setup cannot be carried out.

Let us ellaborate a simple example. Figure 4.6 shows a small two job, two machine flow shop problem where there is a single anticipatory setup between the jobs on the second machine. As can be seen, the setup time has no impact on the processing sequence as there is enough idle time at machine two between the two jobs to do the setup, which is depicted in solid black. Notice also that the setup time has been 'right justified' to the start of the second job but it could have been also left justified to be performed right after job one is finished or it could have been performed at any time during the idleness period of machine two.

A significantly different scenario arises in the case of a non-anticipatory setup, shown in Fig. 4.7. The setup can only be started as soon as job two is completed on the first machine and once it arrives to the second machine.

From the two previous classifications we have the following four combinations:

- Non-anticipatory and sequence independent setup times.
- Non-anticipatory and sequence dependent setup times.
- Anticipatory and sequence independent setup times.
- Anticipatory and sequence dependent setup times.

Notice that the first case, when the setup cannot be detached and when it is independent from the sequence, can be modeled by simply adding its duration to the

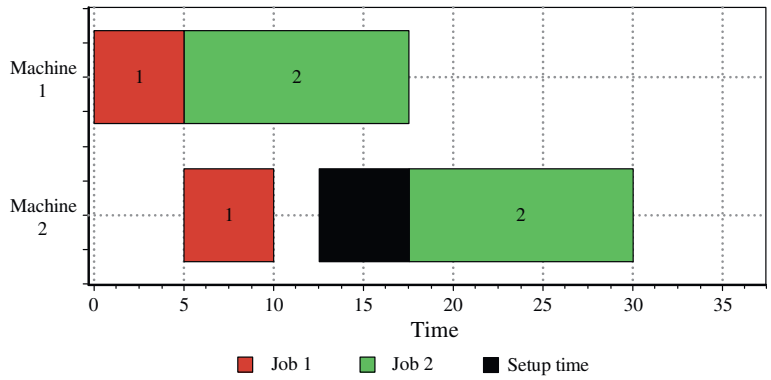


Fig. 4.6 Two job, two machine flow shop with an anticipatory setup on the second machine

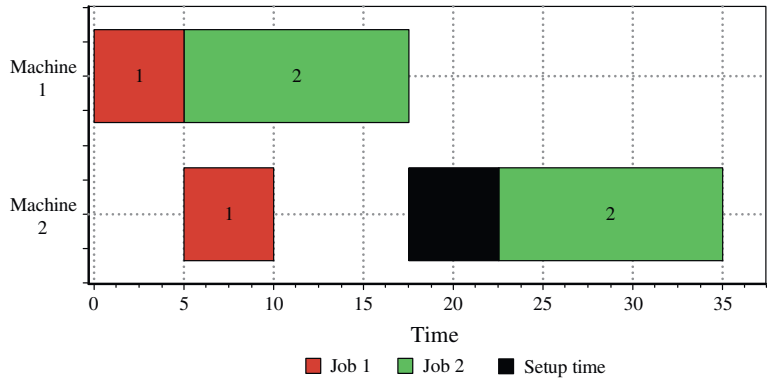


Fig. 4.7 Two job, two machine flow shop with a non-anticipatory setup on the second machine

processing time of the job in the machine. After all, the machine is going to be busy anyway. All other cases require special attention.

It is important to remark that in flexible manufacturing systems, as well as in many production environments, setup times can amount to a significant portion of the productive time if they are not handled with care. The time needed to clean, to name an example, a chemical reactor might be superior to the processing time needed to produce a batch of a given product.

The previous classifications are not the only possible ones. Other classifications further break down setups into what is referred to as family, class or batch setup times. In these cases, jobs are grouped into families so that setup times occur only when finishing production of one family and before starting production of the next one. Sometimes, there are minor setups in between jobs of the same family and major setups between families. Some other classifications include also removal times, which are seemingly identical to setup times, but that are performed after jobs are completed and not prior to the starting of jobs.

It is safe to say that most production systems include, in one way or another, setup times. In some cases, setup times might be sufficiently small to be considered negligible, but in many other cases, setups have to be considered explicitly. Real cases of setup times abound. A clear example comes from the paper cutting industry where the paper cutting machines (guillotines) need to be adjusted when changing from one cutting batch to another. Another example can be obtained from the ceramic tile manufacturing sector. Ceramic tiles are produced in identical processing lines composed of several processes; moulding press, dryer, glazing line, kiln, quality control and finally, packing and delivery. If a production line is set up for a given product format and colour, for example a 33×33 cm. black glazed ceramic tile, changing to a 45×45 cm. white glazed ceramic tile will need significant setup times to change the moulds in the moulding press and to clean and prepare the glazing line for the white glaze. However, changing to a 33×33 cm. yellow glazed tile will only require cleaning in the glazing line, as the moulding press is already set up. A more detailed case study taken from the ceramic tile sector is given in Chap. 15.

As far as the field β in the notation triplet $\alpha/\beta/\gamma$ is concerned, we denote setup and removal times as follows:

β = S_{nsd} : There exists sequence independent setup times. S_{ij} denotes the known and deterministic amount of setup time to be performed on machine i before processing job j.
= S_{sd} : There exists sequence dependent setup times. In this case, for each machine, we have the so-called 'setup time matrix', where S_{ijk} denotes the setup time needed on machine i for processing job k after having processed job j.
= R_{nsd} : Sequence independent removal times.
= R_{sd} : Sequence dependent removal times.

For families or batches, one can use, for example, SB_{nsd}, SB_{sd}, RB_{nsd} and RB_{sd}.

No specific notation is given to differentiate the anticipatory from the non-anticipatory setups cases. However, anticipatory setups are largely assumed in the scientific literature.

4.2.3 Machine Eligibility

In parallel machine problems, or in hybrid environments where stages have parallel machines, it is normal not to expect that all machines are able to process all jobs. In Chap. 3, at Sect. 3.2.3.2, it was already mentioned that in real life, the unrelated parallel machines model, where each machine is able to process each job at a specific speed, is the most probable layout. Factories evolve over time. Old machinery is replaced by new equipment and newer machines do not have the same speed and/or characteristics as old ones. However, usually constraints go further than just varying processing times. In some settings, some machines might be able to process just a subset of jobs and/or some jobs might be assigned to just a subset of machines. This is called machine eligibility.

When machine eligibility is present in a parallel machine layout, each job $j \in N$ has an associated set M_j so that $M_j \subseteq M$ which indicates the subset of machines where job j can be processed. For multistage hybrid problems, this set can be further indexed by the stage number, i.e. M_{ij}. The notation is as follows:

$$\beta = M_j : \text{Machine eligibility constraints.}$$

Once again, restricting eligible machines seems to simplify problems. However, this is not often the case (see, e.g. Urlings et al. 2010).

4.2.4 Permutation Sequences

In the flow shop layout, as commented in Sect. 3.2.3.3 is usually simplified to consider only permutation sequences. This is notated as follows:

$$\beta = prmu : \text{Permutation flow shop problem.}$$

Basically, most of the scheduling research is carried out in the permutation version of the flow shop. Also, in some hybrid environments, the term 'permutation schedules' indicates that there is a single permutation of jobs as a simplified solution for the problems. At each stage, jobs are launched to machines in the permutation order and assignment to machines at each stage is solved separately.

4.2.5 Machine Availability and Breakdowns

Almost throughout the whole book, we have always assumed that machines are continuously available throughout the time horizon. While this simplifying assumption is very convenient, it is really a long shot away from reality. Machines are subject to preventive maintenance, corrective maintenance (breakdowns) or just timetabling or working shift constraints, among other things. The result is the same: machines are not continuously available.

The scientific literature about machine availability and breakdowns is abundant. However, two main distinctions can be stated. The first is when the intervals of machine availability (or unavailability) are known in advance. In this case, for every machine $i \in M$ and for a given production scheduling horizon T, there is a set of h machine availability intervals $AV_i = \{\{BAV_1^i, EAV_1^i\}, \{BAV_2^i, EAV_2^i\}, \ldots, \{BAV_h^i, EAV_h^i\}\}$ where BAV_l^i and EAV_l^i are the starting and ending time of the machine availability period, respectively, and $BAV_l^i < EAV_l^i$ for $1 \leq l \leq h$ and for $i \in M$. A possible notation for this is as follows:

β = *brkdwn*: Machines are subject to breakdowns (general notation).
= $NC_{zz}, NC_{inc}, NC_{dec}, NC_{inczz}, NC_{deczz}, NC_{sc}, NC_{win}$ model different patterns of of machine availabilities. These are zig-zag, increasing, decreasing, increasing in zig-zag, decreasing in zig-zag, staircase and arbitrary, respectively. These patterns are more or less explanatory and can be well understood in parallel machine settings. In a zig-zag pattern, machines with high availability are alternated with machines with low availability and increasing patterns denote settings in which each subsequent machine has less availability.

The second distinction in the machine availability literature is when some information of the machine availability intervals is not known a priori or when the machines simply breakdown with some probabilities or following some probability distributions.

4.2.6 Re-circulation, Re-processing and Skipping

Some important process constraints or special situations include the cases where jobs might visit one specific machine or stage more than once. A similar situation is when some jobs skip one or more machines or stages in the processing sequence. Some of the processing layouts commented in Chap. 3, like the flow shop or the job shop already consider what is called re-circulation. Re-circulation models, for example, double firing processes in ceramic tile manufacturing, the repeated polishing operations in furniture manufacturing and any other environment where a job has to visit a given machine or stage more than once in the processing sequence. Some examples of studies in the literature where re-circulation is allowed include, but are not limited, in the very least, to, Bertel and Billaut (2004) or Choi et al. (2005).

As far as re-circulation goes, the notation is as follows:

$\beta = recrc$: Re-circulation exists as a process constraint. Indicates that at least one job visits at least one machine more than once.

Some related scenarios arise when the outcome of the processing of a job is not perfect and there is a possibility of needing a re-processing. Note that this is in several ways different to re-circulation. First, there is no certainty in that a re-processing will be needed and therefore, re-processing is stochastic. Second, usually a re-process directly involves the same job leaving and entering again the same machine, whereas in re-circulation several machines could be visited in between two visits to the same machine. There is no accepted notation for re-processing but $\beta = reproc$ looks like a valid proposal.

Last, stage skipping could be a factor to consider in some problems. In some scenarios and layouts, like for example flow shops or hybrid flow shops, some jobs might not need processing in some machines or might not need to visit some stages. This is referred to as stage skipping and can be denoted by $\beta = skip$. Note that skipping (or synonymously passing of stations or missing operations) is often (e.g. in flowshops)

modelled/interpreted by adding respective operations of jobs with processing time 0. However, this approach does not allow jobs simply to skip a station. Instead, the respective jobs have to queue up in front of the machine. And when they are to be processed, they immediately pass the machine because of their processing time 0. While this makes the consideration of the respective scheduling problem much simpler from an optimization method point of view, in fact this approach ignores the real-world potential for optimization where the respective situation is interpreted by 'real' skipping of a machine.

4.2.7 No-Idle Machines

An interesting situation arises when no-idle time is allowed at machines. This constraint models an important practical situation that arises when expensive machinery is employed. Idling on expensive equipment is often not desired or even not possible due to technological constraints. Some examples are the steppers used in the production of integrated circuits by means of photolithography. Other examples come from sectors where less expensive machinery is used but where machines cannot be easily stopped and restarted or is completely uneconomical to do so. Ceramic roller kilns, for example, consume a large quantity of natural gas when in operation. Idling is not an option because it takes several days to stop and to restart the kiln due to a very large thermal inertia. In all such cases, idling must be avoided. Notation goes as follows:

β = *no-idle*: No machine can stay idle after starting the process of operations.

Generally speaking, the processing sequence of jobs where a no-idle processing constraint is present in machines must ensure that once a machine starts processing, it is not stopped until the last job in the sequence is finished. This is graphically depicted in Fig. 4.8 for a five job, four machine flow shop example.

As one can see, the start of each job is delayed so to end up with complete blocks of consecutive jobs and no-idle times on machines.

The no-idle processing constraint has been seldom studied in the literature. Additional work thus remain to be done in this setting as much more complex processing layouts might benefit from the consideration of such processing constraints.

4.2.8 Batching Machines

High throughput machines usually accept more than one job at the same time. Actually, arriving jobs might be grouped into batches and then the machine simultaneously completes all the batch at the same time. This is very common in the semiconductor industry (Lee et al. 1992) or in tyre manufacturing (Bellanger and Oulamara 2009).

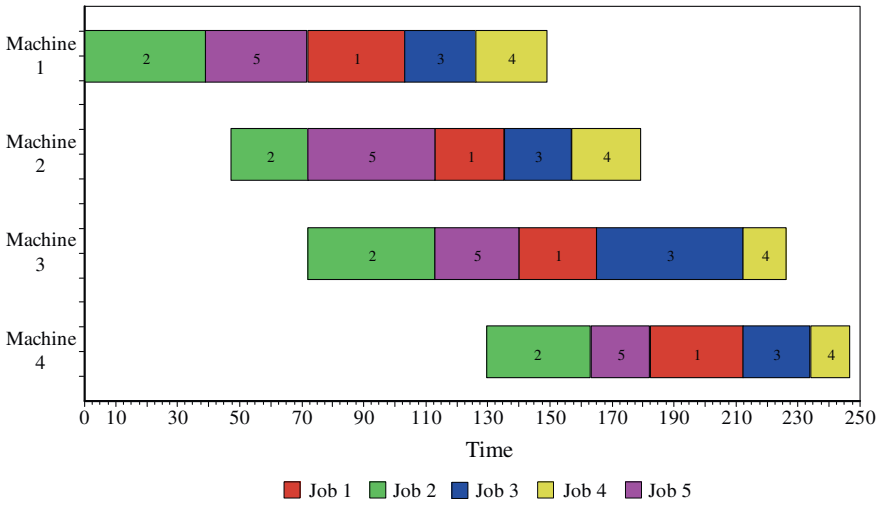


Fig. 4.8 Gantt chart example of a no-idle flow shop

Batching might be economical if the arriving jobs require the same setup and can be processed in batches. Usually, two types of batches are considered. $p-batch$ (parallel batching) is when the processing time of the whole batch in the machine is considered to be equal to that of the largest processing time of any job in the batch. $s-batch$ (serial batching) occurs when the processing time of the batch is equal to the sum of the processing times of all the jobs in the batch. Additionally, b denotes the *batch size*, or the maximum number of jobs that can enter a batch. If this number changes from machine to machine, then it is used b_i instead. As a result of all of the above, the notation is therefore:

$\beta = p-batch(b)$ (parallel batching) : All machines consider batches of maximum size b (or b_i) where the processing time of the batch is equal to that of the largest processing time among the jobs in the batch.

$= s-batch(b)$ (serial batching) : All machines consider batches of maximum size b (or b_i) where the processing time of the batch is equal to that of the sum of the processing times of the jobs in the batch.

The literature on batching scheduling is extremely large, as there are many possible sub-cases, generalisations and variations in the batching decisions.

4.3 Operations Constraints

This section deals with all the scheduling constraints that are more closely related to the operations or tasks of the jobs. Note that the distinction with the previous process constraints is, in some cases, not entirely clear cut and in many cases debatable. However, in some way or another, these constraints can be linked to jobs more than to machines.

4.3.1 Interruption, Preemption and Splitting

The concept of preemption was discussed in this book as early as in Sect. 1.5. In contrast, if it is assumed that, once a job or task starts in a machine, it cannot be interrupted and the job has to be processed to completion, then preemption is not allowed. There are many scenarios where preemption might be needed or even interesting:

- Arrival of new urgent job orders might require to stop jobs already being processed.
- It might be economical or beneficial to preempt a job and/or to continue it in another machine and/or at a later time.
- Sudden cancellations of clients' orders might require to stop processing a batch of products.
- Breakdowns on machines or any other unexpected event might result in preemption.

There are many possible types of preemptions and varied classifications. If, once interrupted, the preempted operation is lost and when resumed, processing has to restart from the beginning, the preemption is denoted as non-resumable. This is the case for example in a pre-heating phase in metallurgy processes. If one stops the pre-heating operation, it will be required to start pre-heating from the beginning again at a later time since the processed product might have cooled off by then. Note that setup time operations can also be preempted and most of the time, if a setup operation is aborted, a different setup is started (the next product in the sequence is changed) and therefore, the initial setup will have to be restarted from the beginning. Setup times are then denoted as non-resumable. Conversely, a preemption can be resumable if the interrupted job is just continued from the point where it was interrupted to completion. Sometimes, an intermediate situation exists where the operation can be resumed, but some penalty (time, costs, etc.) exists. This case is referred as semi-resumable.

Jobs might be preempted just once, a limited number of times or an unlimited number of times. Jobs that break down into operations, as in the shop layouts, might complicate things, since preempting a non-resumable operation might result in the whole job being repeated again (all operations). A preempted operation, for example in a parallel machine scenario, might have to return to the same machine or might be able to return to any other machine. As one can see, the possibilities are endless. However, not all scenarios are so complex. Sometimes, preemptions are as simple as stopping a machine because the working shift is over and the processing of the same job, on the same machine, continues the next working shift or after the weekend with no introduction of a different job in between.

As far as the notation goes, the basic preemption is referred to as follows:

$\beta = pmtn$: Preemption allowed (general). In other works it is also referred to as *prmp*.
$= pmtn - non - resumable$: Non-resumable preemption.
$= pmtn - semi - resumable$: Semi-resumable preemption.
$= pmtn - resumable$: Resumable preemption.

Another aspect that might affect the different tasks of jobs is splitting. In specific manufacturing environments, jobs/operations might be split into several sub-jobs/sub-operations. This problem is partially related to preemption of jobs/operations. However, the latter is often induced by processing requirements of other (more urgent) operations while job splitting is usually caused by lot sizing or related considerations.

4.3.2 Release Dates, Due Dates, Deadlines, Processing Windows

Release dates, as well as due dates and deadlines where already defined in Chap. 3 as input data for a scheduling problem. However, they also indicate constraints that affect operations. As commented, the first task of a job cannot start before the release date and the last task of a job should be ideally completed before the due date and forcibly completed before the deadline.

When such data exist in a production setting, it is denoted as follows:

$\beta = r_j$: There are release dates for the jobs. Omitting this field is the same as assuming that all release dates are zero, i.e. $r_j = 0, \forall j \in N$.
$\beta = d_j$: There are distinct due dates for each job.
$= d_j = d$: There is a single identical due date for all the jobs. This is referred to as the common due date case.
$= \overline{d}_j$: The due dates are compulsory and are referred to as deadlines. Some early research work denotes deadlines as $\triangle_j$.

Note that many times the due dates d_j are not explicitly stated in the β field of the notation since, as we will see in Chap. 5, many optimisation objectives implicitly consider the due dates.

If one finds in the same production setting release dates r_j and deadlines $\overline{d}_j$ the result is the so-called 'processing window' as the entire processing of a job must be completed within the $\overline{d}_j - r_j$ time interval. This is often referred to as 'time window'. Deadlines in general and processing windows in particular raise the issue of schedule feasibility, i.e. not all schedules or possible orderings of the jobs and tasks on the machines might be feasible. It might even be impossible to complete all jobs within their respective processing windows.

4.3.3 No Wait, Minimum and Maximum Time Lags and Overlapping

Normally, in shop or hybrid shop layouts, the next task of a given job can start as soon as the previous task is fully finished. However, if the next machine or stage is busy, then the task, in the form of semi-finished product or in-process inventory,

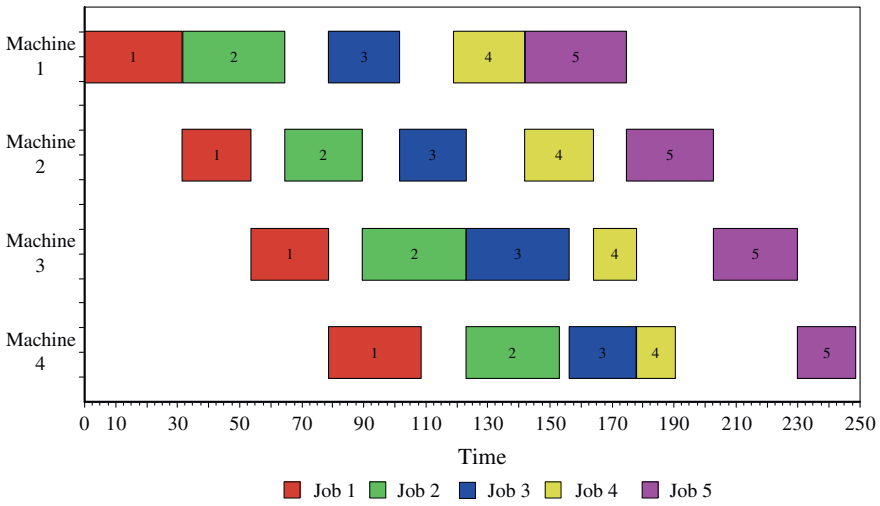


Fig. 4.9 Gantt chart example of a no-wait flow shop

waits at some place until the machine is free. In other words, tasks are allowed to wait indefinitely in between machines or stages.

There are many scenarios where waiting is not an option and the processing of some or all jobs needs to be carried out without interruptions between machines. This situation is commonly known as 'no-wait'. A typical example comes from steel production where the steel ingots are not allowed to wait between production stages or otherwise they would cool off. Most prepared food production lines also have, in some way or another, no-wait restrictions. A clear example is a frozen food line where no-wait must be enforced or otherwise the cool chain might break. Other examples can be found in chemical and pharmaceutical industries, among many others.

Constructing a no-wait schedule is not trivial. If the next machine needed by a job is busy, it is required to go back to the previous tasks of the same job and to delay their start, so that tasks never wait. Figure 4.9 shows a five job, four machine no-wait flow shop example. From the figure, we see that, for job four, the three initial tasks have to be delayed, even including an idle time on the first machine, to make sure that the job four does not find the last machine busy.

Note that in the flow shop layout, the no-wait constraint necessarily implies a permutation sequence. It is of paramount importance to consider that much of the constraints usually interact and a seemingly easy way to deal with a constraint, could end up being hard to solve when in conjunction with another special situation. As regards notation, the no-wait situation is easily described:

$\beta = nwt$: Jobs cannot wait in between stages. Mathematically, this implies that $SO_{ij} = EO_{i-1,j}, \forall j \in N, i \in N \backslash 1$. Note that this is sometimes referred to as *no-wait*.

In some works, like in the one of Hall and Sriskandarajah (1996), the authors extend this notation to clearly specify between which machines or stages waiting is not allowed. For example $no - wait(3, 4)$ indicates that no waiting is allowed for any job between stages or machines three and four.

The no-wait setting is in obvious contrast with the regular setting in which indefinitely waiting could be possible. It should be pointed out that the Gantt chart profiles of every single no-wait job is fixed, independent of the sequence of jobs. This information can be used when developing optimization approaches to no-wait problems.

Intermediate scenarios between assuming indefinite waiting times and the no-wait case are possible, as there are situations where jobs or tasks might wait some time, but not too much. For example, in a frozen food cold chain, 10 min waiting might be acceptable knowing that the temperature will not raise beyond a microbial contamination threshold in those 10 min. Therefore, a maximum waiting time or 'maximum time lag' is allowed. Jobs or tasks can wait, for no more than the maximum time lag. In general, we refer to lag_{ij}^{+} to the maximum time lag for job $j \in N$ after being processed on machine $i \in M$ and before proceeding to machine $i + 1$. In order to satisfy this maximum time lag, the following inequality must hold for all tasks: $EO_{ij} \leq SO_{i+1,j} \leq EO_{ij} + lag_{ij}^{+}$.

Similarly, there might be 'minimum time lags'. This implies that subsequent tasks of a job cannot start until a given minimum time has elapsed since the last operation. There are many situations where minimum time lags could appear. Picture for example painting operations that require a minimum drying time before subsequent operations can be performed. Products that require some type of furnacing or kiln firing might need some cooling off time before being handled. We denote by lag_{ij}^{-} to the minimum time lag for job $j \in N$ after being processed on machine $i \in M$ and before proceeding to machine $i + 1$. Minimum time lags effective need jobs to satisfy the following: $SO_{i+1,j} \geq EO_{ij} + lag_{ij}^{-}$.

A straightforward extension is when both maximum and minimum time lags exist. This imposes a 'time window' interval inside which the next task of any job should be started. Some other authors, even consider negative time lags. Negative time lags allow for a subsequent task to start actually before the preceding task is finished. This can be used to model, for example, large batches of products that can effectively overlap in between stages, much like the lot streaming and lot sizing discussion of Sect. 3.4 of Chap. 3.

4.3.4 Special Processing Times

Nothing has been mentioned about the nature of the processing times so far apart from mentioning in Chap. 3 that processing times are supposed to be known and fixed in advance. However, there are many situations that might affect this otherwise strong assumption.

First of all, there is a sizeable part of the theoretical research in scheduling that deals with very specific processing times. These are referred to as the unit processing times and identical processing times cases, denoted as follows:

$\beta = p_{ij} = 1$: All processing times are equal to 1. This is referred to as unit processing times.
$= p_{ij} = \{0, 1\}$: Zero or unit processing times.
$= p_{ij} = p$: All processing times are equal to a value p.

Note that these special processing times cases might be erroneously classified as too theoretical. While it is true that the bulk of research in these cases is theoretical, there are some situations in practice that benefit from these special cases. For instance, in Lee et al. (1992), batching machines for the burn-in operations in semiconductor manufacturing are studied. In these machines, all the jobs that simultaneously enter the machine are modelled as having the processing time equal to the largest job that enters the batching machine. As we can see, such studies are highly relevant in practice.

Some other specific situations regarding processing times are also often studied in the literature. For instance, *proportional processing times* model the situation in which the processing times of the tasks of a given job are proportional to each other. For example, the first machine might be faster than the second machine, the third faster than the fourth machine and so on. These problems are usually noted as $\beta = p_{ij} = p_j/s_i$, where s_i is a speed factor of the machine i. There is a body of the literature about the proportionate flow shop, with some key works like the ones of Ow (1985) and Hou and Hoogeveen (2003).

So far, it has also been assumed that processing times, once determined, do not depend on anything else. In reality, processing times can be acted upon. Picture a painting operation of a large wood or metal piece. Painting might be carried out by personnel operating spray paint guns. The more personnel assigned to that painting operation, the shorter the processing time and the more expensive the operation will be. In other words, sometimes, processing times depend on other factors and can be somehow controlled. This situation is referred to in the scientific literature as controllable processing times and there is also a large body of research. The variety and diversity of controllable processing times scheduling is overwhelming as there are literally hundreds of possible variants. Note that controllable processing times are often also referred to as resource dependent processing times or even compressible processing times. The interested reader is advised to read through the previously cited reviews for more information.

No matter how the processing times are estimated, they are no more than that: an estimate. In some production systems, production or processing times cannot just be estimated by any amount. In these cases, if a fixed and deterministic processing time is not acceptable, some statistical distribution for each processing time has to be assumed. Under this situation, we enter the stochastic processing times case. The body of research under stochastic processing times is huge. Note that when one refers to stochastic scheduling, most of the time the meaning is stochastic processing times.

However, it has to be noted that analytical results are only possible for simplified production settings. For complex production scenarios, simulation is believed to be the only viable choice.

The above special cases for processing times are not the only possible ones. Time-dependent processing times occur when the actual processing time depends on the instant in which operations are started. A special case of the previous one is the so-called deteriorating processing times where the processing time of a given task increases as time passes by.

Last, it has been demonstrated that processing times decrease over time in repetitive production lines. This is the so-called learning effect. While it is debatable that processing times with learning considerations should be addressed in short-term scheduling, this has not deterred authors from proposing several models in which the processing times are subject to the most varied learning effects.

4.4 Transportation Constraints

Another situation that is very common in practice is that a task of a job cannot start just as soon as the previous task of the same job is finished. Doing so implies that the job, whatever product it models, arrives instantaneously from the output of the previous machine to the input of the next machine. Normally, the job will need to be carried over to the next machine. There are numerous ways of transporting products from machine to machine: conveyor belts, robotic cells, robotic arms, wagons, product boxes, automated guided vehicles, overhead cranes, as well as many others. As a matter of fact, the variety in product handling inside a factory is probably as high as the variety in manufacturing systems themselves. Alongside with this variety, there are many different approaches to transportation constraints in the scientific literature.

Basically, two types of transportation constraints can be considered. First, there is the transportation time, so that jobs do not instantaneously move from machine to machine. Should there be no more constraints, transportation times could be modeled simply as a minimum time lag with the time it takes to transport the job from one machine to the next. However, sometimes lot streaming (see Sect. 3.4) is employed so jobs are able to be splitted for transport from one machine to the next which might allow overlapping processing of jobs (sub-jobs or sub-lots of a job) on consecutive machines.

There is a second group of transportation constraints: the number of transporters is limited. This second situation results in a hard problem as the handling of products must be solved together with the scheduling. Obviously, if all transporters are busy transporting products, then the job has to wait somewhere until one is free. The immediate result of this is that the transportation time is not a fixed and constant term, but a time that depends on how the scheduling is being done and on the traffic in the transportation.

In the most complex scenario, the scheduling problem becomes a combined scheduling and routing problem in which countless routes for different transporters

Fig. 4.10 An automated guided vehicle (AGV) transporting a box full of ceramic tiles

have to be derived to transport all the goods inside the factory. Lately, some special emphasis is being observed on the routing of automated guided vehicles (AGVs) like those shown in Fig. 4.10.

The previous aspects of transportation constraints are not the only ones. Sometimes, transportation constraints go as far as the final delivery of products to clients. This is referred to as *product delivery coordination* and has been widely studied. Other scenarios include the necessary coordination of scheduling and transportation in complex supply chains.

4.5 Storage Constraints

In all previous sections, we have shown some alternatives for the many assumptions of the theoretical scheduling problems. However, there is still an important one remaining: storage. In-process inventory has to be stored inside the plant and obviously, space is not unlimited. When the storage areas are full, upstream machines have to stop while downstream machines work over the in-process inventory to free up some space in the storage areas.

Buffer capacity requirements occur per job between two consecutive operations of this job, i.e. after the previous operation has been finished and before the next operation starts. Often, unlimited buffers are supposed in manufacturing scheduling. However, sometimes physically buffer limitations are present explicitly or the goal

of minimising working capital implicitly demands for limited in-process inventory. Buffer capacity requirement can be simply measured by 1 as buffer unit per job, i.e. by the number of jobs in a buffer (e.g. if jobs are buffered in standardised containers) or by some other measures (size, temperature category, etc.). This requirement or demand for buffer capacity has to be balanced with the available buffer capacity which, (a) might be limited by an upper bound with respect to the indicators determining jobs'/operations' buffer capacity type, and (b) can be provided centrally for all respective buffer requirements of the system, decentrally for a pair of consecutive or nearby located machines/stages or something in between, e.g. a subset of machines/stages. Additionally, e.g. in job shop environments, one might distinguish between buffers which are located physically and/or logically in front of a machine and those behind a machine.

Blocking is a special situation that results from the limited storage. It mainly affects the flow shop layout. If there is a limited storage capacity in between machines, it might happen that when there is no more capacity, the previous machine cannot finish and release a job since there is no place where to put it. As a result, the machine is 'blocked' with the job. As regards the notation, buffers and blocking are as follows:

β = buffer $= b$: There are $m - 1$ buffers of capacity b in between the m machines.
= buffer $= b_i$: There are $m - 1$ buffers of different capacities b_i which depend on the machines.
= *block* : There is blocking. Implies a limited storage or buffer in between machines.

4.6 Other Constraints

At the beginning of the present chapter it was stated that the reality of the production systems is extremely complex and that there are countless possible constraints. In previous sections we have just scratched the surface of this issue. Now we proceed with a brief enumeration of many other possible constraints not mentioned before.

It has been mentioned that in supply chains, transportation operations have to be considered. However, there is one basic assumption that has been considered up to this point. There is one single production floor or factory, or, if there is more than one, scheduling is carried out separately at each one. Large companies with many factories have additional decisions on where to produce the products that form the production plan, i.e. there is a previous decision of deciding in which factory each product should be manufactured. Then, on each factory, a scheduling problem has to be solved. These situations have been recently referred to as 'distributed scheduling problems' and are recently being studied. One example is the study of the distributed permutation flow shop layout by Naderi and Ruiz (2010). Figure 4.11 clearly shows one full flow shop problem with 10 jobs and four machines that has to be distributed between two factories, each one also with four machines. Obviously, the job factory assignment decisions and the scheduling problem at each factory cannot be separated if a good quality result is desired.

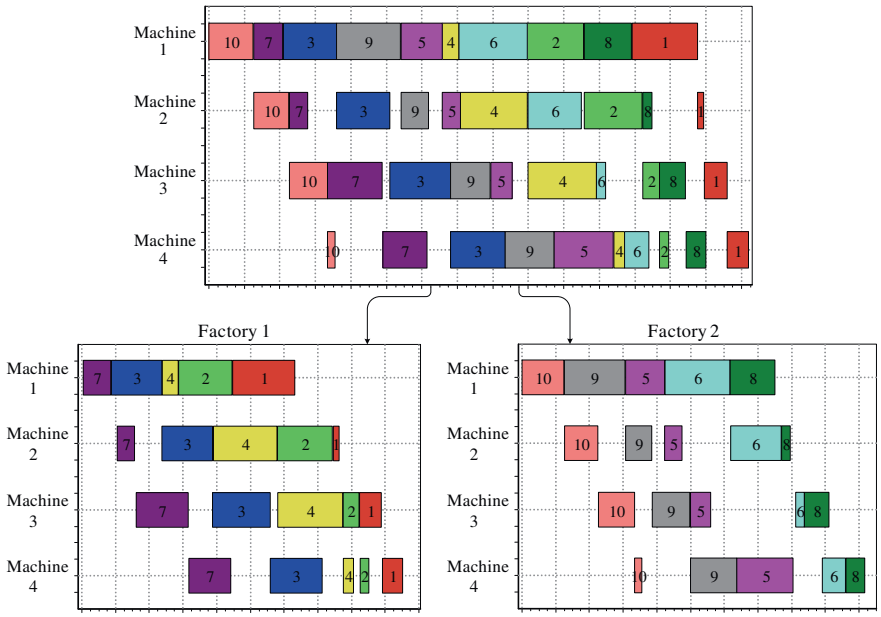


Fig. 4.11 An example of a distributed permutation flow shop problem

There are many other constraints. Setup times are a never-ending source of special situations. Some machines, for example, cannot be setup unless an specific technician is working and therefore, setups during the weekend, for example, have to be avoided. Furthermore, setups are often cleaning and adjustments of machines that are carried out by plant personnel. In this situation, setup times are also controllable.

4.7 Conclusions and Further Readings

During this chapter we have introduced some of the situations that can be found in production settings, grouped by process constraints, operations constraints, transportation and storage constraints. Note that this is just one possible grouping as the sheer variety of real situations easily overcomes any exhaustive classification attempt. This variety of scheduling settings and constraints results in a vast literature where most constraints are still studied in an isolated way most of the time. For example, the literature on no-wait scheduling is plenty, as it is the literature on setup times. However, in the intersection one finds very few research papers, i.e. no-wait scheduling with setup times. Moreover, it is safe to say that the intersection among three or more sets of constraints for any shop layout would return an empty set of references most of the time. Furthermore, the more constraints considered, the less complex the studied scheduling layout usually is. This is, there are some studies with

many constraints considered simultaneously for the single machine problem, but just a handful of papers have been published where several constraints are simultaneously considered for complex hybrid job shop problems, to name an example.

The previous issue is actually a symptom that shows that more work needs to be done in the area of scheduling in order to cope with more realistic problems where several simultaneous constraints are considered. Any production scheduler working at a company has surely identified several of the previously commented constraints as important in his/her production setting. In the next chapters we will make an attempt at providing some general techniques and algorithms, as well as a case study, to help in bridging this gap between academic scheduling and real scheduling at production shops.

Further readings on the topic discussed in this chapter stem from the already commented main textbooks in scheduling. When describing the notation, we have sought to balance the simplicity against the thoroughness, also taking into account its acceptance. This is not possible in many cases: For instance, the notation on machine availability is not widely accepted (see as, for example, Lee et al. 1997 for a different notation).

Regarding specific works, apart from the references that have been cited in-line when describing specific sets of constraints in the corresponding sections, state-of-the-art and comprehensive reviews of the scheduling literature and setup times are available from Yang and Liao (1999), Allahverdi et al. (1999), Cheng et al. (2000) and Allahverdi et al. (2008). A good state-of-the-art review of scheduling problems with machine availabilities is available from Schmidt (2000). Reviews on flow shop are Park et al. (1984), Turner and Booth (1987), Framinan et al. (2004), Ruiz and Maroto (2005), Hejazi and Saghafian (2005) or Gupta and Stafford (2006), while for hybrid flow shops it is worth citing Linn and Zhang (1999), Vignier et al. (1999), Wang (2005), Ruiz and Maroto (2006), Quadt and Kuhn (2007), Ribas et al. (2010) and Ruiz and Vázquez-Rodríguez (2010). Stage skipping has been addressed by a number of authors, see, e.g. Leisten and Kolbe (1998) and Urlings et al. (2010) for the flow shop/hybrid flow shop settings, respectively. No-idle processing constraints have been seldom studied in the literature. To the best of our knowledge, the first papers are those of Adiri and Pohoryles (1982) and Vachajitpan (1982), and recent references on the topic are Pan and Wang (2008) and Ruiz et al. (2009). A comprehensive review of scheduling with batching decisions is due to Potts and Kovalyov (2000), although it only covers research that is already more than a decade old. In the last decade, more than double that number of papers have appeared for batching scheduling. Some recent examples of work in preemption are Agnetis et al. (2009) and Hendel et al. (2009).

Due dates are intensively studied in the research literature. Some examples are Sen and Gupta (1984), Baker and Scudder (1990) and more recently, Vallada et al. (2008). Some examples of processing windows research appear in Maffioli and Sciomachen (1997) or Yeung et al. (2009). Minimum, maximum and both types of time lags have been dealt with, among many others, by Riezebos and Gaalman (1998), Brucker et al. (1999) and Fondrevelle et al. (2006). Negative time lags and overlaps are discussed in network planning textbooks, being a recent reference Urlings et al. (2010).

Unit processing times are studied in Liua et al. (1999) and Averbakh et al. (2005) just to name a few. Zero and unit processing times are examined in Lushchakova and Kravchenko (1998), among others. Results for identical processing times have been published in Crama and Spieksma (1996) and Lee et al. (1992). A recent survey about scheduling with controllable processing times is that of Shabtay and Steiner (2007).

In the book of Pinedo (2012) there is a whole part devoted to stochastic scheduling. Regarding time depending processing times, a main source is Gawiejnowicz (2008). An interesting study for the time-dependent processing times single machine layout is given by Chen (1996), and a example of deteriorating processing times where the processing time of a given task increases as time passes by is Kubiak and van de Velde (1998). Some examples of processing times with learning effect are the papers of Lee et al. (2004) and Eren and Guner (2009) just to name a few.

Some recent references on lot streaming are Defersha and Chen (2012) and Feldmann and Biskup (2008), being Sarin and Jaiprakash (2007) a book on the topic. A good review of scheduling with transportation times is due to Lee and Chen (2001). Product delivery coordination can be found in Chang and Lee (2004). Excellent review papers on the coordination of scheduling and transportation are available from Potts and Hall (2003), Li et al. (2005) and Chen and Vairaktarakis (2005).

The main results of scheduling with blocking are due to Hall and Sriskandarajah (1996). Other relevant papers are those of Papadimitriou and Kanellakis (1980), Leisten (1990), Brucker et al. (2003) and Mascis and Pacciarelli (2005). Finally, an interesting application where setup times are considered controllable is Ruiz and Andres-Romano (2011).

References

Adiri, I. and Pohoryles, D. (1982). Flowshop no-idle or no-wait scheduling to minimize the sum of completion times. *Naval Research Logistics*, 29(3):495–504.

Agnetis, A., Alfieri, A., and Nicosia, G. (2009). Single-machine scheduling problems with generalized preemption. *INFORMS Journal on Computing*, 21(1):1–12.

Allahverdi, A., Gupta, J. N. D., and Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *OMEGA, The International Journal of Management Science*, 27(2):219–239.

Allahverdi, A., Ng, C. T., Cheng, T. C. E., and Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187(3):985–1032.

Averbakh, I., Berman, O., and Chernykh, I. (2005). The *m*-machine flowshop problem with unit-time operations and intree precedence constraints. *Operations Research Letters*, 33(3):263–266.

Baker, K. R. and Scudder, G. D. (1990). Sequencing with earliness and tardiness penalties: a review. *Mathematics of Operations Research*, 15:483–495.

Bellanger, A. and Oulamara, A. (2009). Scheduling hybrid flowshop with parallel batching machines and compatibilities. *Computers & Operations Research*, 36(6):1982–1992.

Bertel, S. and Billaut, J.-C. (2004). A genetic algorithm for an industrial multiprocessor flow shop scheduling problem with recirculation. *European Journal of Operational Research*, 159(3):651–662.

Brucker, P., Heitmann, S., and Hurink, J. (2003). Flow-shop problems with intermediate buffers. *OR Spectrum*, 25(4):549–574.

Brucker, P., Hilbig, T., and Hurink, J. (1999). A branch and bound algorithm for a single-machine scheduling problem with positive and negative time-lags. *Discrete Applied Mathematics*, 94 (1–3):77–99.

Chang, Y. C. and Lee, C.-Y. (2004). Machine scheduling with job delivery coordination. *European Journal of Operational Research*, 158(2):470–487.

Chen, Z.-L. (1996). Parallel machine scheduling with time dependent processing times. *Discrete Applied Mathematics*, 70(1):81–93.

Chen, Z. L. and Vairaktarakis, G. L. (2005). Integrated scheduling of production and distribution operation. *Management Science*, 51(4):614–628.

Cheng, T. C. E., Gupta, J. N. D., and Wang, G. Q. (2000). A review of flowshopn scheduling research with setup times. *Production and Operations Management*, 9(3):262–282.

Choi, S. W., Kim, Y. D., and Lee, G. C. (2005). Minimizing total tardiness of orders with reentrant lots in a hybrid flowshop. *International Journal of Production Research*, 43(11):2149–2167.

Crama, Y. and Spieksma, F. C. R. (1996). Scheduling jobs of equal length: Complexity, facets and computational results. *Mathematical Programming*, 72(3):207–227.

Defersha, F. and Chen, M. (2012). Jobshop lot streaming with routing flexibility, sequence-dependent setups, machine release dates and lag time. *International Journal of Production Research*, 50(8):2331–2352.

Eren, T. and Guner, E. (2009). A bicriteria parallel machine scheduling with a learning effect. *International Journal of Advanced Manufacturing Technology*, 40(11–12):1202–1205.

Feldmann, M. and Biskup, D. (2008). Lot streaming in a multiple product permutation flow shop with intermingling. *International Journal of Production Research*, 46(1):197–216.

Fondrevelle, J., Oulamara, A., and Portmann, M.-C. (2006). Permutation flowshop scheduling problems with maximal and minimal time lags. *Computers & Operations Research*, 33(6):1540–1556.

Framinan, J. M., Gupta, J. N. D., and Leisten, R. (2004). A review and classification of heuristics for permutation flow-shop scheduling with makespan objective. *Journal of the Operational Research Society*, 55(12):1243–1255.

Gawiejnowicz, S. (2008). *Time-Dependent Scheduling*. Springer.

Gupta, J. N. D. and Stafford, Jr, E. F. (2006). Flowshop scheduling research after five decades. *European Journal of Operational Research*, 169(3):699–711.

Hall, N. G. and Sriskandarajah, C. (1996). A survey of machine scheduling problems with blocking and no-wait in process. *Operations Research*, 44(3):510–525.

Hejazi, S. R. and Saghafian, S. (2005). Flowshop-scheduling problems with makespan criterion: a review. *International Journal of Production Research*, 43(14):2895–2929.

Hendel, Y., Runge, N., and Sourd, F. (2009). The one-machine just-in-time scheduling problem with preemption. *Discrete Optimization*, 6(1):10–22.

Hou, S. and Hoogeveen, H. (2003). The three-machine proportionate flow shop problem with unequal machine speeds. *Operations Research Letters*, 31(3):225–231.

Kubiak, W. and van de Velde, S. (1998). Scheduling deteriorating jobs to minimize makespan. *Naval Research Logistics*, 45(5):511–523.

Lawler, E. L. (1973). Optimal Sequencing of a Single Machine Subject to Precedence Constraints. *Management Science*, 19(5):544–546.

Lee, C.-Y. and Chen, Z.-L. (2001). Machine scheduling with transportation considerations. *Journal of Scheduling*, 4(1):3–24.

Lee, C.-Y., Lei, L., and Pinedo, M. (1997). Current trends in deterministic scheduling. *Annals of Operations Research*, 70:1–41.

Lee, C.-Y., Uzsoy, R., and Martin-Vega, L. A. (1992). Efficient algorithms for scheduling semiconductor burn-in operations. *Operations Research*, 40(4):764–775.

Lee, W. C., Wu, C. C., and Sung, H. J. (2004). A bi-criterion single-machine scheduling problem with learning considerations. *Acta Informatica*, 40(4):303–315.

Leisten, R. (1990). Flowshop sequencing problems with limited buffer storage. *International Journal of Production Research*, 28(11):2085.

Leisten, R. and Kolbe, M. (1998). A note on scheduling jobs with missing operations in permutation flow shops. *International Journal of Production Research*, 36(9):2627–2630.

Lenstra, J. K. and Rinnooy Kan, A. H. G. (1978). Complexity of Scheduling under Precedence Constraints. *Operations Research*, 26(1):22–35.

Li, C. L., Vairaktarakis, G. L., and Lee, C.-Y. (2005). Machine scheduling with deliveries to multiple customer locations. *European Journal of Operational Research*, 164(1):39–51.

Linn, R. and Zhang, W. (1999). Hybrid flow shop scheduling: A survey. *Computers & Industrial Engineering*, 37(1–2):57–61.

Liua, Z., Yua, W., and Cheng, T. C. E. (1999). Scheduling groups of unit length jobs on two identical parallel machines. *Information Processing Letters*, 69(6):275–281.

Lushchakova, I. N. and Kravchenko, S. A. (1998). Two-machine shop scheduling with zero and unit processing times. *European Journal of Operational Research*, 107(2):378–388.

Maffioli, F. and Sciomachen, A. (1997). A mixed-integer model for solving ordering problems with side constraints. *Annals of Operations Research*, 69:277–297.

Mascis, A. and Pacciarelli, D. (2005). Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research*, 143(3):498–517.

Naderi, B. and Ruiz, R. (2010). The distributed permutation flowshop scheduling problem. *Computers & Operations Research*, 37(4):754–768.

Ow, P. S. (1985). Focused scheduling in proportionate flowshops. *Management Science*, 31(7):852–869.

Pan, Q.-K. and Wang, L. (2008). A novel differential evolution algorithm for no-idle permutation flow-shop scheduling problems. *European Journal of Industrial Engineering*, 2(3):279–297.

Papadimitriou, C. H. and Kanellakis, P. C. (1980). Flowshop scheduling with limited temporary-storage. *Journal of the ACM*, 27(3):533–549.

Park, Y. B., Pegden, C., and Enscore, E. (1984). A survey and evaluation of static flowshop scheduling heuristics. *International Journal of Production Research*, 22(1):127–141.

Pinedo, M. L. (2012). *Scheduling: Theory, Algorithms, and Systems*. Springer, New York, fourth edition.

Potts, C. N. and Hall, N. G. (2003). Supply chain scheduling: Batching and delivery. *Operations Research*, 51(4):566–584.

Potts, C. N. and Kovalyov, M. Y. (2000). Scheduling with batching: A review. *European Journal of Operational Research*, 120(2):222–249.

Quadt, D. and Kuhn, D. (2007). A taxonomy of flexible flow line scheduling procedures. *European Journal of Operational Research*, 178(3):686–698.

Ribas, I., Leisten, R., and Framinan, J. M. (2010). Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Computers & Operations Research*, 37(8):1439–1454.

Riezebos, J. and Gaalman, G. J. C. (1998). Time lag size in multiple operations flow shop scheduling heuristics. *European Journal of Operational Research*, 105(1):72–90.

Ruiz, R. and Andres-Romano, C. (2011). Scheduling unrelated parallel machines with resource-assignable sequence-dependent setup, times. 57(5–8):777–794.

Ruiz, R. and Maroto, C. (2005). A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research*, 165(2):479–494.

Ruiz, R. and Maroto, C. (2006). A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research*, 169(3):781–800.

Ruiz, R., Vallada, E., and Fernández-Martínez, C. (2009). Scheduling in flowshopns with no-idle machines. In Chakraborty, U. K., editor, *Computational Intelligence in Flow Shop and Job Shop Scheduling, volume 230 of Studies in Computational Intelligence*, pages 21–51, Berlin. Springer-Verlag.

Ruiz, R. and Vázquez-Rodríguez, J. A. (2010). The hybrid flowshop scheduling problem. *European Journal of Operational Research*, 205(1):1–18.

Sarin, S. C. and Jaiprakash, P. (2007). *Flow Shop Lot Streaming*. Springer, New York.

Schmidt, G. (2000). Scheduling with limited machine availability. *European Journal of Operational Research*, 121(1):1–15.

Sen, T. and Gupta, S. K. (1984). A state-of-art survey of static scheduling research involving due dates. *OMEGA, The International Journal of Management Science*, 12(1):63–76.

Shabtay, D. and Steiner, G. (2007). A survey of scheduling with controllable processing times. *Discrete Applied Mathematics*, 155(13):1643–1666.

Sidney, J. B. (1975). Decomposition Algorithms for Single-Machine Sequencing with Precedence Relations and Deferral Costs. *Operations Research*, 23(2):283–298.

Turner, S. and Booth, D. (1987). Comparison of Heuristics for Flow Shop Sequencing. *OMEGA, The International Journal of Management Science*, 15(1):75–78.

Urlings, T., Ruiz, R., and Sivrikaya-Şerifoğlu, F. (2010). Genetic algorithms with different representation schemes for complex hybrid flexible flow line problems. *International Journal of Metaheuristics*, 1(1):30–54.

Vachajitpan, P. (1982). Job sequencing with continuous machine operation. *Computers & Industrial Engineering*, 6(3):255–259.

Vallada, E., Ruiz, R., and Minella, G. (2008). Minimising total tardiness in the *m*-machine flowshop problem: A review and evaluation of heuristics and metaheuristics. *Computers & Operations Research*, 35(4):1350–1373.

Vignier, A., Billaut, J.-C., and Proust, C. (1999). Les problèmes d'ordonnancement de type flow-shop hybride: État de l'art. *RAIRO Recherche opérationnelle*, 33(2):117–183. In French.

Wang, H. (2005). Flexible flow shop scheduling: optimum, heuristics and artificial intelligence solutions. *Expert Systems*, 22(2):78–85.

Yang, W.-H. and Liao, C.-J. (1999). Survey of scheduling research involving setup times. *International Journal of Systems Science*, 30(2):143–155.

Yeung, W.-K., Oğuz, C., and Cheng, T.-C. E. (2009). Two-machine flow shop scheduling with common due window to minimize weighted number of early and tardy jobs. *Naval Research Logistics*, 56(7):593–599.

Chapter 5
Objectives

5.1 Introduction

The framework for manufacturing scheduling envisioned in Chap. 1 presented three main blocks—models, methods and tools—that, together with the humans involved in the process, compose a scheduling system. This part of the book deals with scheduling models, and in Chaps. 3 and 4 we have reviewed the main elements of these models concerning systems layouts and processing constraints, respectively. Now a final element of the scheduling models—the objectives sought in the decision problem—are presented in this chapter before the building of scheduling models is properly discussed in Chap. 6.

Interestingly, most scheduling literature naturally assumes the existence of some objectives *intrinsic* to scheduling. However, we have already discussed that manufacturing scheduling decisions are part of a broader set of decisions collectively known as production management (see Chap. 2), and as a consequence, scheduling objectives should be aligned/integrated with those regarding other decisions in production management. Therefore, before introducing the most employed scheduling objectives, we provide a discussion of their choice in terms of the (broader) objectives considered in production management. Next, we classify most of the objectives employed in the scheduling literature depending on the existence or not of due dates, and will study the most important in detail. As we will discuss, reality can hardly be modeled with a single objective or criterion, therefore we will comment on multi-criteria or multi-objective problems as a pointer to a Chap. 10 where these topics will be treated in much more detail.

More specifically, in this chapter, we

- present a rationale for scheduling objectives within the broader context of production management (Sect. 5.2),
- introduce a notation and classification for scheduling performance measures (Sect. 5.3),
- discuss the main scheduling objectives (Sect. 5.4),

J. M. Framinan et al., *Manufacturing Scheduling Systems*,
DOI: 10.1007/978-1-4471-6272-8_5, © Springer-Verlag London 2014

- introduce ways to modulate the aforementioned objectives by using weights and/or rankings (Sect. 5.5),
- present an illustrative example (Sect. 5.6),
- point out the main issues regarding the evaluation of more than one criteria (Sect. 5.7).

5.2 A Rationale for Scheduling Objectives

As is well-known from general decision theory, quantifiable objective functions intend that the decision-maker wishes to determine extreme solutions (maximum, minimum), or solutions which achieve fixed and pre-determined objective function values, or intervals of objective function values which might be bounded from both sides or only from one side. The latter give minimum or maximum levels to be reached. In production management (and consequently in manufacturing scheduling), often the objectives are classified into the categories of costs (sometimes profit), time, quality and flexibility.

Since scheduling decisions are located in the operational, very short-term horizon level of the multi-level manufacturing planning and control framework (see Sect. 2.4.1), cost components are often not considered as objective functions in scheduling models, at least not explicitly. Instead, it is either supposed that on the scheduling level as well as on the shop floor level costs can only be influenced via time objectives (and therefore time-related objective functions are explicitly considered) or costs are proportional to time parameters and consequently, weighted time parameters are considered as a representation of costs. These time parameters may refer to machine and/or resource-related parameters, or to job-related parameters. The latter may be further classified into intrasystem objectives, e.g. maximising the utilisation of resources, and extrasystem objectives, such as minimising the delays with respect to the committed due dates. The category 'quality' in models of manufacturing scheduling is also usually exclusively regarded within the time framework, i.e. by service level indicators, being the reason why this category is not discussed further here as well. Referring to flexibility, as a well-accepted category of manufacturing goals, it has not been explicitly considered in manufacturing scheduling models for many years. However, during the last years several models and approaches have been published which explicitly consider variances and/or variabilities (usually measured by the coefficient of variation), e.g. of completion times, as objective functions, though usually in a multi-criteria setting. These indicators are intended to yield stable and robust schedules and are obviously directly related to flexibility aspects. Although flexibility is usually not primarily considered in static-deterministic models as described here, recognition of these objective functions reflects the fact that the models discussed here usually represent simplifications of dynamic-stochastic real-world settings, which require consideration of flexibility aspects at least indirectly.

Figure 5.1 summarises the above basic discussion of types of objective functions in manufacturing scheduling and gives examples for the different categories.

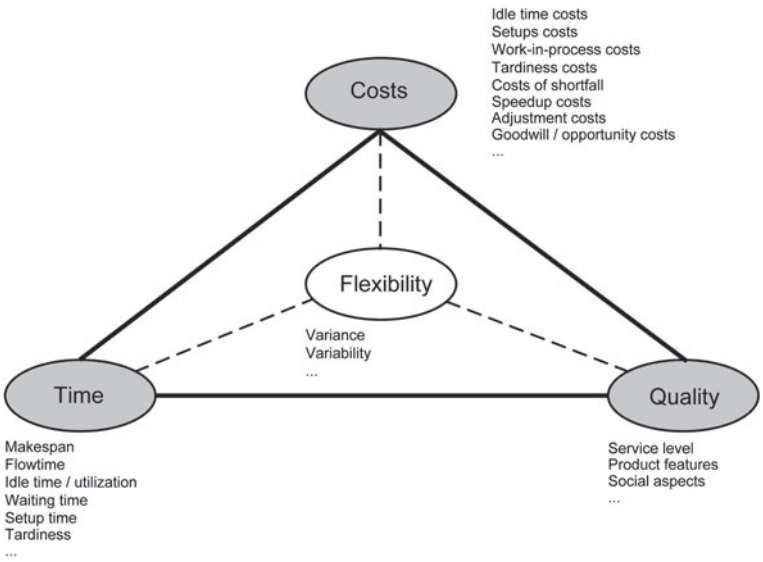


Fig. 5.1 (Extended) triangle of objective categories in manufacturing scheduling including flexibility

Note that the objective categories mentioned here are highly interdependent. Sometimes they act in parallel, such as in the case of idle time costs and utilisation. Unfortunately, most of them are often conflicting: The case of maximising machine utilisation is strongly against reducing work-in-progress or inventory levels. High machine utilisation is also conflicting with service level or with meeting due dates. If machine utilisation is low, there is a certain guarantee that machines will be free when needed for finishing a job that is about to be late. On the other hand, a very high inventory level practically guarantees that lead times will approach zero as all products will be readily available.

In addition, there is the issue of establishing a good level of performance in the shop floor for any of the previously cited criteria. A reference point, like for example, 'A cost threshold' or a 'customer's satisfaction level' has to be set. However, this is in fact a circular problem since usually measuring these criteria is, as mentioned, not easy. As a result, the performance of the schedules is often measured against past performance. Given that manufacturing systems are highly dynamic, comparing against past performance inevitably introduces a large degree of bias in the process. Another important issue is that, when measuring the performance of a schedule, the schedule has to be broken down into specific time horizons (e.g. weeks) and each time horizon is scheduled independently of others. The result might be that a very good optimisation for one period affects negatively the following periods. Other problems arise when breaking levels of management. For example, the schedule manager at a higher decision level and the plant manager at an operational level. While the scheduler might be interested in maximising customer's satisfaction, the

plant manager will probably be centered around maximising machine utilisation or throughput.

In case of narrow manufacturing capacities, utilisation-oriented objective functions seem to be appropriate. This might be one reason why these objective functions became popular in the early days of manufacturing scheduling, i.e. in the mid of the twentieth century when after World War II every product which was produced could be sold and the only bottleneck in many industries was exclusively manufacturing capacity. However, looking at many industries during the last decades, surplus capacity was characteristic and the persistence of utilisation-oriented objective functions in many papers appears to be somewhat nostalgic. Instead, guaranteeing short and/or reliable delivery dates has become much more relevant for successful manufacturing. Thus, cycle-time-oriented and/or due-date oriented objective functions seem to be more relevant during the last years and in the foreseeable future. Only if capacity represents the true bottleneck for short-term business success, utilisation-oriented objective functions should be considered.

5.3 Performance Measures

Objectives employed in manufacturing scheduling models are related to the optimisation of some indicator of the performance of one schedule in the shop floor. We denote these indicators as *performance measures* and some of the most important will be defined next. Note that the notation corresponding to the data required below has been given in Sect. 3.2.2:

- C_j: Completion time of job j, i.e. time at which job j finishes its processing in the shop.
- F_j: Flowtime of job j. This models the time that the job stays in the shop while in production or waiting for processing. Clearly, $F_j = C_j - r_j$. Most importantly, in the lack of release dates, flowtime and completion time are equal, i.e. $F_j = C_j$.
- L_j: Lateness or slack of job j with respect to its due date d_j, or deadline $\overline{d}_j$. As such, it is calculated as $L_j = C_j - d_j$ or $L_j = C_j - \overline{d}_j$. The lateness function is shown in Fig. 5.2.
- T_j: Tardiness of job j. This measures only jobs that finish beyond their due dates and it is calculated as $T_j = \max\{L_j, 0\}$. Tardiness ignores negative lateness values, i.e. all jobs that are finished before the due date are not tardy and therefore not a reason to worry about. The tardiness function is shown in Fig. 5.3.
- E_j: Earliness of job j. Similarly to tardiness, it is only calculated for early jobs, i.e. $E_j = \max\{-L_j, 0\}$. Note that the earliness function is non-increasing with respect to C_j as shown in Fig. 5.4.
- U_j Tardy job or late job. This measure yields 1 if job j is late (i.e. $T_j > 0$ or equivalently $L_j > 0$, or equivalently $C_j > d_j$), and zero if not. Figure 5.5 depicts this function.

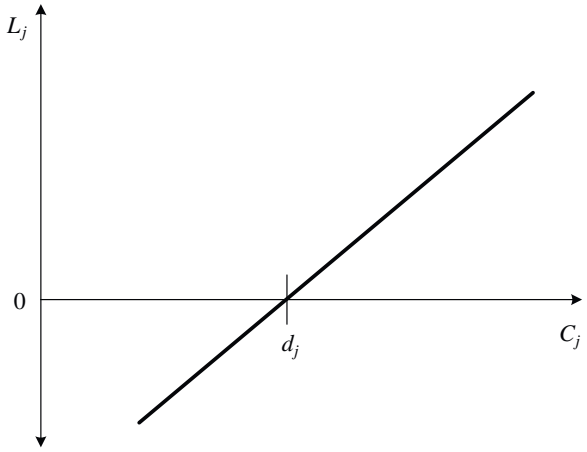


Fig. 5.2 Lateness function

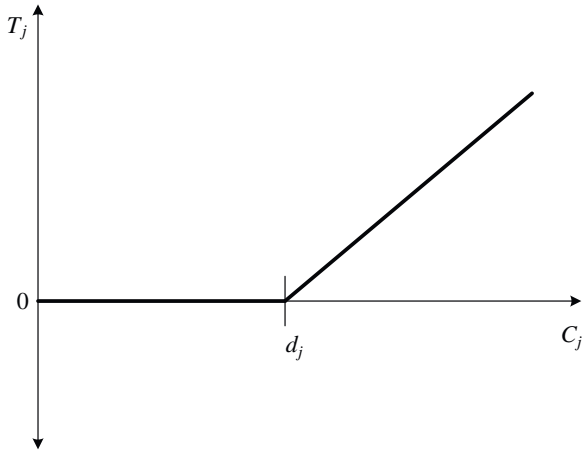


Fig. 5.3 Tardiness function

- V_j Early job. This measure yields 1 if job j is early (i.e. $L_j < 0$, or equivalently $C_j < d_j$), and zero if not. Figure 5.6 depicts this function.
- ET_j Earliness-Tardiness of job j. This measure is defined as the sum of the tardiness and the earliness of job j, i.e. $ET_j = E_j + T_j$. The function of ET_j is given in Fig. 5.7.
- JIT_j Just-In-Time job. This measure yields 1 if job j is neither early nor late (i.e. $L_j = 0$, or equivalently $C_j = d_j$), and zero if not. This is often referred to as 'just-in-time' or JIT. The function of JIT_j is given in Fig. 5.8.

Aside from the data in the model (most notably r_j and d_j), note that all above measures depend on the completion time of the job. The completion time of the job

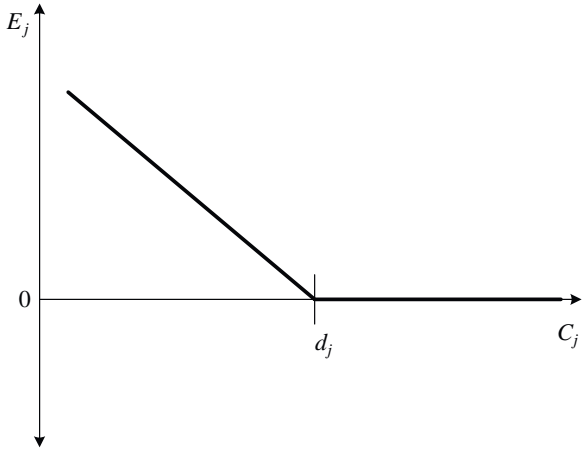


Fig. 5.4 Earliness function

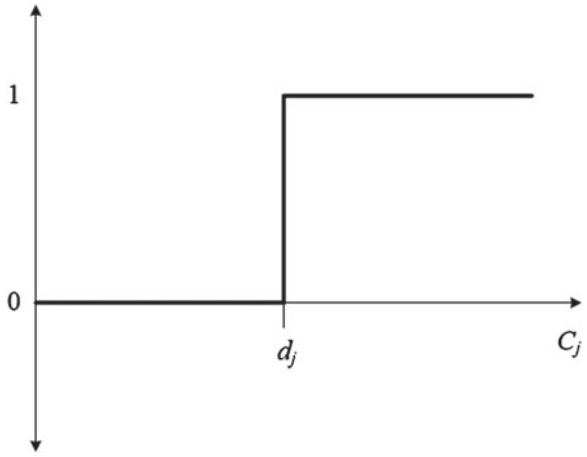


Fig. 5.5 Tardy jobs function

can be obtained for an particular schedule, i.e. the collection SO_{ij} of the starting times of operation for each job j on each machine i, according to:

$$C_j = \max_i \{SO_{ij} + p_{ij}\}.$$

When assuming some form of stochastic behaviour in the processing times, then the completion time of the job is a random variable that cannot be calculated according to a deterministic procedure. In these cases, statistical indicators of such random variable (such as its mean or variance) could be either obtained by a closed formula (unfortunately, only for specific cases), or at least estimated via samples.

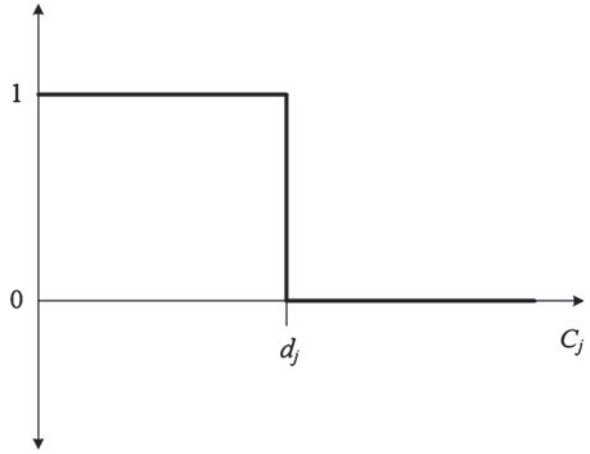


Fig. 5.6 Early job function

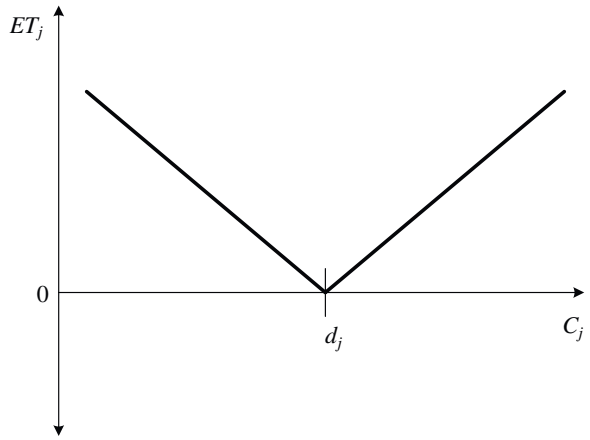


Fig. 5.7 Earliness-tardiness function

In addition, we will introduce two measures that are associated to rescheduling. Recall that, in rescheduling, the goal is to minimise disruptions from existing plans whenever new events occur. Therefore, we assume that there are two schedules Π_A and Π_B corresponding to points in time before and after the arrival of the new event, respectively. Two employed measures are:

- $D_j(\Pi_A, \Pi_B)$: Sequence disruption of job j. It indicates the difference (in absolute value) between the position of job j in schedule Π_A, and its position in schedule Π_B, i.e. $D_j(\Pi_A, \Pi_B) = |\Pi_A(j) - \Pi_B(j)|$.

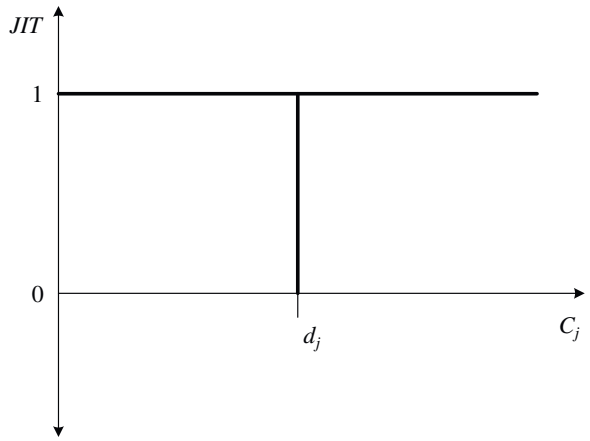


Fig. 5.8 JIT job

Table 5.1 Summary of performance measures

Measure	Notation	Formulae
Flowtime	F_j	$C_j - r_j$
Lateness	L_j	$C_j - d_j$
Tardiness	T_j	$\max\{0, C_j - d_j\} = \max\{0, L_j\}$
Earliness	E_j	$\max\{0, d_j - C_j\} = \max\{0, -L_j\}$
Earliness and tardiness	ET_j	$E_j + T_j = \lvert C_j - d_j\rvert = \lvert L_j\rvert$
Tardy job	U_j	$\begin{cases}1 & C_j > d_j \\ 0 & \text{otherwise}\end{cases} = \begin{cases}1 & L_j > 0 \\ 0 & \text{otherwise}\end{cases}$
Early job	V_j	$\begin{cases}1 & C_j < d_j \\ 0 & \text{otherwise}\end{cases} = \begin{cases}1 & L_j < 0 \\ 0 & \text{otherwise}\end{cases}$
Just-in-time job	JIT_j	$\begin{cases}1 & C_j < d_j \text{or } C_j > d_j \\ 0 & \text{otherwise, i.e. if } C_j = d_j\end{cases} = \begin{cases}1 & L_j \neq 0 \\ 0 & L_j = 0\end{cases}$
Sequence disruption	$D_j(\Pi_A, \Pi_B)$	$\lvert\Pi_A(j) - \Pi_B(j)\rvert$
Time disruption	$\Delta_j(\Pi_A, \Pi_B)$	$\lvert C_j(\Pi_A) - C_j(\Pi_B)\rvert$

- $\Delta_j(\Pi_A, \Pi_B)$: Time disruption of job j. It indicates the difference (in absolute value) between the completion time of job j in schedule Π_A, and its completion time in schedule Π_B, i.e. $\Delta_j(\Pi_A, \Pi_B) = |C_j(\Pi_A) - C_j(\Pi_B)|$.

The above measures are summarised in Table 5.1.

5.4 Scheduling Objectives

From the previous notation, we are ready to define the most common objectives. A summary is provided in Fig. 5.2. According to the classification of scheduling models introduced in Sect. 3.3.2, the γ field of this notation will be used to represent the criteria under consideration. For practical purposes, we will classify them in five general groups:

- Feasibility. In this case, the objective is to find a feasible solution for the model.
- Non due-date related. There is no information (or it is not required) regarding due dates or deadlines.
- Due-date related. The objective involves the usage of information about due dates or deadlines.
- Rescheduling related. The objective involves information from two different schedules (usually associated to two different decision intervals).
- Additional objectives. Other objectives that can be considered.

Feasibility models indicate the absence of an objective (except that of finding a feasible schedule) and will not be discussed further. In such models, the γ field is indicated by a slash, i.e.:

$\gamma = -$: Feasibility problem.

Therefore, the model $1|prec, prmpt|-$ indicates the formal decision problem of finding (at least) one feasible schedule for the single machine layout with precedence constraints where pre-emption is allowed.

For the rest of the groups, a distinction can be done depending on the type of function g applied to the completion times, i.e. function f is of the max-form if $f = \max_{1 \leq j \leq n} g(C_j)$. In contrast, is of the sum-form if $f = \sum_{j=1}^{n} g(C_j)$. Furthermore, recall from Sect. 1.5 that g is said to be regular if it is a non-decreasing function of $(C_1, C_2, \ldots, C_n)$. Unless stated otherwise, the objectives presented in the following sections are regular.

A final remark can be done with respect to an stochastic settings, as there is an stochastic counterpart of all scheduling objectives considered here. In such cases, the otherwise deterministic objectives are now random variables and therefore the objective can be minimising some of their characteristics, most notably their mean or variance. These objectives can also be integrated in the model. For instance, the objective of minimising the expected value of objective O is notated as follows:

$\gamma = E[O]$: expected value of objective O.

5.4.1 Non Due Date Related Objectives

In this section, we discuss all objectives that do not consider due dates in their calculation. These are mainly concerned with the completion times and with the total time each job spends in the shop.

5.4.1.1 Max-Form Objectives

Makespan

The maximum completion time or makespan is defined as $C_{\max} = \max\{C_1, C_2, \ldots, C_n\}$. Regarding notation, makespan minimisation is denoted as:

$\gamma = C_{\max}$: Makespan minimisation.

Makespan can be seen as the time needed to finish the whole production plan since it measures from the time the first job starts processing (normally this is assumed to be zero, unless release times or other constraints exist), to the time the last job in the processing sequence is finished on the last machine it visits. It is thus connected with the idle time. However, no standard definition of idle time exists, as it depends if the 'heads' (i.e. the time that each machine is idle before starting processing its first job) and 'tails' (i.e. the time that each machine is idle after processing its last job) are both included, excluded, or just one of them is considered. For the definition of idle time including both 'heads' and 'tails', it can be proved that both objectives are equivalent and thus minimizing makespan and minimizing this definition of idle time are equivalent. Additionally, makespan is also closely connected to the production rate which can be calculated as $\mathrm{PR} = \frac{n}{C_{\max}}$. The utilisation of machine i is calculated as $\frac{\mathrm{TC}_i}{\mathrm{PR}}$ where TC_i is the total capacity of machine i.

Maximising machine utilisation is interesting as far as cost accounting is concerned. Capital equipment and expensive machinery have to be paid for. Therefore, using machines to produce products that increase the revenue of companies yield higher returns of the investments and, in the end, a higher profitability. On the contrary, idle machines are not generating revenue and are seen as a lost profit. However, it has to be noted that utterly increasing machine utilisation can result in undesired effects. For example, keeping machines busy churning out products that nobody demands will only reduce the level of service, increase the level of inventory and in the end, reduce benefits. In a make-to-stock environment, increasing machine utilisation makes all the sense. In addition, it should be mentioned that maximisation of utilization makes sense for bottleneck machines or bottleneck systems albeit high utilization might in turn result in larger waiting times as is well known from queuing theory.

Maximum flowtime

The maximum flowtime objective is defined as $\max_{1\leq j\leq n} F_j$ and it is denoted as:

$\gamma = \max F_j$: maximum flowtime minimisation.

Maximum flowtime is a niche objective that has been seldom studied in the scientific literature. The reason is that with no release times, i.e. $r_j = 0, \forall j \in N$, we have that $C_{\max} = \max F_j$. Therefore, and specially for long production horizons, the effect of the release date r_j over the $F_{\max}$ decreases as $C_{\max}$ increases unless very large release dates are present. A weighted version of this objective can be also formulated (see Table 5.2).

5.4.1.2 Sum-Form Objectives

Total/average completion time or flowtime

Total flowtime and total completion time add up the flow time and completion times of all the jobs, respectively. Total flowtime and total completion times are seen as surrogates for cycle time, and it was already mentioned that WIP and cycle time (CT) are connected.

Minimisation of total/average flowtime and total completion times can be denoted as follows:

$\gamma = \sum C$: Total completion time, calculated as $\sum C = \sum_{j=1}^{n} C_j$.
$= \overline{C}$: Average completion time, calculated as $\overline{C} = \frac{1}{n} \cdot \sum_{j=1}^{n} C_j$.
$= \sum F$: Total flowtime, calculated as $\sum F = \sum_{j=1}^{n} F_j$.
$= \overline{F}$: Average flowtime time, calculated as $\overline{F} = \frac{1}{n} \cdot \sum_{j=1}^{n} F_j$.

Note that the minimisation of total flowtime/completion time is equivalent to the minimisation of the average flowtime/completion time.

5.4.2 Due Date Related Objectives

Meeting due dates is probably the highest regarded indicator of customer's satisfaction influenced by short-term operations management. As a result of this, many different objectives exist that in one way or another capture this importance. All these criteria are mainly based in the previous definitions of lateness (L_j), Tardiness (T_j) and Earliness (E_j).

At this stage, it is important to make some punctualisations. In a make-to-order production environment, every job usually corresponds to a client order (or to several orders from different clients that have been batched). Therefore, the due date set by or agreed with the client, is considered. Make-to-stock production systems are different as client orders are ideally served from the stock immediately as they are placed.

Table 5.2 Summary of objectives

Type	Objective	Notation	Formulae
Feasibility problem			
	Finding any feasible schedule	–	
Non due date related			
Max-form	Makespan	$\max C_j$ or $C_{\max}$	$\max_{1\le j\le n} C_j$
	Weighted makespan	$\max w_j C_j$	$\max_{1\le j\le n} w_j C_j$
	Maximum flowtime	$\max F_j$	$\max_{1\le j\le n} F_j$
	Maximum weighted flowtime	$\max w_j F_j$	$\max_{1\le j\le n} w_j F_j$
Sum-form	Total completion time	$\sum C_j$	$\sum_{j=1}^{n} C_j$
	Total weighted completion time	$\sum w_j C_j$	$\sum_{j=1}^{n} w_j C_j$
	Total flowtime	$\sum F_j$	$\sum_{j=1}^{n} F_j$
	Total weighted flowtime	$\sum w_j F_j$	$\sum_{j=1}^{n} w_j F_j$
Due date related			
Max-form	Maximum lateness	$\max L_j$	$\max_{1\le j\le n} L_j$
	Maximum weighted lateness	$\max w_j L_j$	$\max_{1\le j\le n} w_j L_j$
	Maximum tardiness	$\max T_j$	$\max_{1\le j\le n} T_j$
	Maximum weighted tardiness	$\max w_j T_j$	$\max_{1\le j\le n} w_j T_j$
	Maximum earliness	$\max E_j$	$\max_{1\le j\le n} E_j$
	Maximum weighted earliness	$\max w_j E_j$	$\max_{1\le j\le n} w_j E_j$
Sum-form	Total lateness	$\sum L_j$	$\sum_{j=1}^{n} L_j$
	Total weighted lateness	$\sum w_j L_j$	$\sum_{j=1}^{n} w_j L_j$
	Total tardiness	$\sum T_j$	$\sum_{j=1}^{n} T_j$
	Total weighted tardiness	$\sum w_j T_j$	$\sum_{j=1}^{n} w_j T_j$
	Number of tardy jobs	$\sum U_j$	$\sum_{j=1}^{n} U_j$
	Number of weighted tardy jobs	$\sum w_j U_j$	$\sum_{j=1}^{n} w_j U_j$
	Total earliness	$\sum E_j$	$\sum_{j=1}^{n} E_j$
	Total weighted earliness	$\sum w_j E_j$	$\sum_{j=1}^{n} w_j E_j$
	Number of early jobs	$\sum V_j$	$\sum_{j=1}^{n} V_j$
	Number of weighted early jobs	$\sum w_j V_j$	$\sum_{j=1}^{n} w_j V_j$
	Total earliness and tardiness	$\sum ET_j$	$\sum_{j=1}^{n} E_j + T_j$
	Total weighted earliness and tardiness	$\sum w_j ET_j$	$\sum_{j=1}^{n} w_j (E_j + T_j)$
	Total weighted earliness and tardiness with different weights	$\sum w_j E_j + w'_j T_j$	$\sum_{j=1}^{n} w_j E_j + w'_j T_j$
	Number of just-in-time jobs	$\sum JIT_j$	$\sum_{j=1}^{n} JIT_j$
	Number of weighted just-in-time jobs	$\sum w_j JIT_j$	$\sum_{j=1}^{n} w_j JIT_j$
Rescheduling related			
Max-form	Maximum sequence disruption	$\max D(\Pi_A, \Pi_B)$	$\max_{1\le j\le n} \max D_j(\Pi_A, \Pi_B)$
	Maximum time disruption	$\max \Delta(\Pi_A, \Pi_B)$	$\max_{1\le j\le n} \max \Delta_j(\Pi_A, \Pi_B)$
Sum-form	Total sequence disruption	$\sum D(\Pi_A, \Pi_B)$	$\sum_{j=1}^{n} D_j(\Pi_A, \Pi_B)$
	Total time disruption	$\sum \Delta(\Pi_A, \Pi_B)$	$\sum_{j=1}^{n} \Delta_j(\Pi_A, \Pi_B)$

As a result, there are no due dates. However, stock levels decline as orders are served and reach a point where replenishment is needed. An 'internal' order is placed so to refill the stock with products. These internal orders are, as regards scheduling, identical to make-to-order client orders. The reason is that if the internal orders are produced late, client orders might go unserved due to stockouts.

In make-to-order systems, it is interesting to measure the service level which is just the fraction of client orders served by or before the due date. Alternatively, in make-to-stock production environments, the fill rate is the fraction of orders that are served from the inventory. Other measures include the amount of time in backorder or number of stockouts. All the following scheduling objectives are related to these real-life goals.

5.4.2.1 Max-Form Objectives

Maximum lateness

The objective of maximum lateness minimisation can be defined as follows:

$\gamma = L_{\max}$: Maximum lateness minimisation, calculated as $\max L = \max_{1 \leq j \leq n} L_j\}$.

This objective tries to minimise the maximum deviation from the due date or deadline in the schedule. The idea is to limit as much as possible the delay with respect to the committed date. Note that $\max L_j$ might be a negative number if all jobs in a given schedule are completed before their corresponding due dates. Indeed, if all jobs can be finished before their due dates (i.e. there are schedules for which $L_j < 0 \forall j$), then maximum lateness minimisation leads to finishing the jobs as early as possible).

Maximum tardiness

Maximum tardiness minimisation can be defined as follows:

$\gamma = T_{\max}$: Maximum tardiness minimisation, calculated as $\max T_j = \max_{1 \leq j \leq n} T_j$.

Minimising maximum tardiness has a direct meaning. For example, if $\max T_j = 15$ then the most tardy job in the schedule is completed in 15 units of time (whatever these units might be) after its due date. However, $\sum T_j$ can quickly grow to large numbers if n is large in a given layout, which might be difficult to interpret.

Maximum earliness

In some scenarios, tardy jobs might be of no concern in comparison with having jobs finished before the due date. Finishing a product early has several added costs. First of all, the product has to be stocked if the due date is still far ahead in time, which results in inventory handling and costs, specially so if the product is cumbersome or voluminous. Furthermore, if the product is very expensive and/or if the manufacturing process is costly, finishing it early precludes the company to invoice the client, which

can add a considerable financial burden. Some extreme cases could be perishable products, that have to be used shortly after production. Dairy foods or some chemical products are clear examples. In these cases, minimising earliness might be not just preferable, but mandatory.

Maximum earliness minimisation can be defined as follows:

$\gamma = \max E_j$: Maximum earliness minimisation, calculated as $\max E_j = \max_{1 \leq j \leq n} E_j$.

Earliness minimisation opens a very important aspect to consider when building schedules: idle time insertion (i.e. non semi-active schedules). It is usually desired to leave machines idle even when there are jobs to process. This is expected since doing otherwise, would result in jobs finishing before the due date. However, inserting idle time is not a straightforward issue at all since the amount of inserted time can be considered a continuous variable. It has to be decided before which tasks idle time is inserted. Furthermore, inserting idle time might in turn result in tardy jobs.

5.4.2.2 Sum-Form Objectives

Total/average lateness

Minimisation of total/average lateness can be denoted as follows:

$\gamma = \sum L_j$: Total lateness minimisation, calculated as $\sum L_j = \sum_{j=1}^{n} L_j$.
$= \overline{L}$: Average lateness minimisation, calculated as $\overline{L} = \frac{1}{n} \cdot \sum_{j=1}^{n} L_j$.

Total lateness and average lateness are not very appropriate objectives. The reason is that negative lateness values will compensate the positive ones and the final total or average lateness value might have no real meaning or interpretation. For example, a small average lateness $\overline{L}$ could be the result of all jobs meeting due dates, something that is desired, or it could be the outcome of a bad schedule where half of the jobs are very early (large negative lateness values) and the other half of the products are finished very late (large positive lateness values).

Total/average tardiness

As we have just discussed, total/average lateness minimisation does not seem to make sense in many settings. Instance, total/average tardiness minimisation can be used:

$= \sum T_j$: Total tardiness minimisation, calculated as $sumT_j = \sum_{j=1}^{n} T_j$.
$= \overline{T}$: Average tardiness minimisation, calculated as $\overline{T} = \frac{1}{n} \cdot \sum_{j=1}^{n} T_j$.

Note that as it was the case with the total and average completion time or flowtime, there is no real difference from the optimisation point of view in minimising total

or average lateness or tardiness. As expected, the main drawback of the tardiness measure is that some jobs might be finished very early, which in many situations is not a desired result.

Total/average earliness

As with lateness minimisation, two variants are commonly studied:

$= \sum E_j$: Total earliness minimisation, calculated as $\sum E_j = \sum_{j=1}^{n} E_j$.
$= \overline{E}$: Average earliness minimisation, calculated as $\overline{E} = \frac{1}{n} \cdot \sum_{j=1}^{n} E_j$.

Number of tardy/early jobs

Sometimes, when a job is tardy or early, the damage is already done and it is not so important how much tardy or early the job is. In these cases, the interest lies in 'counting' how many jobs are early or late. Therefore, the following objectives can be defined:

$\gamma = \sum U_j$: Number of tardy jobs minimisation, calculated from : $\sum U_j = \sum_{j=1}^{n} U_j$.
$= \sum V_j$: Number of early jobs minimisation, calculated as $\sum V_j = \sum_{j=1}^{n} V_j$.

Note that $\sum U_j$ is a regular performance measure whereas $\sum V_j$ is non-regular.

An interesting feature of $\sum U_j$ and $\sum V_j$ is that they are very easy to interpret. As a matter of fact, they can be easily transformed to measures like the percentage of early or tardy jobs. From that perspective, it is very easy to assess the quality of a given schedule. Note however, that it might be impossible to finish all products before or after their due dates. As expected, the main drawback of these measures is that they neglect extremely early or tardy jobs. Normally, in real life, it is desired to have a small number of tardy (early) jobs and for the jobs that are tardy (early) a small tardiness (earliness) is desired.

Total earliness and tardiness

A straightforward extension of the tardiness and earliness minimisation is to add them both in a single expression. For all aforementioned reasons, it might be of interest to complete a job as close as possible to its due date, i.e. not early, not late. Earliness-tardiness minimisation is defined as follows:

$\gamma = \sum_{j=1}^{n} E_j + T_j$: Total Earliness and tardiness minimisation. Sometimes this is simply expressed as ET_j.

It is interesting to note that $ET_j = E_j + T_j = \max\{d_j - C_j, 0\} + \max\{C_j - d_j, 0\} = |C_j - d_j|$. This expression is equivalent to what is referred in some texts as the minimisation of the absolute deviation from the due date (Brucker 2007).

Number of JIT jobs

Similarly with $\sum U_j$ or $\sum V_j$, we can maximise the number of just-in-time jobs *JIT* as follows:

$\gamma = \sum JIT_j$: Maximisation of the number of just in time jobs, calculated as $\sum JIT_j = \sum_{j=1}^{n} JIT_j$.

Note that, in contrast to the rest of objectives, a maximisation criterion is the one that makes sense.

5.4.3 Rescheduling-Related Objectives

In Sect. 5.3, some rescheduling-related performance measures have been introduced, i.e. $D_j(\Pi_A, \Pi_B)$ the sequence disruption of job j, and $\Delta_j(\Pi_A, \Pi_B)$ the time disruption of job j. From these two performance measures, the following minimisation objectives can be formulated:

$\gamma = \max D_j(\Pi_A, \Pi_B)$: Minimisation of the maximum sequence disruption, calculated as $\max D_j(\Pi_A, \Pi_B) = \max 1 \leq j \leq nD_j(\Pi_A, \Pi_B)$.
$= \max \Delta_j(\Pi_A, \Pi_B)$: Minimisation of the maximum time disruption, calculated as $\max \Delta_j(\Pi_A, \Pi_B) = \max 1 \leq j \leq n\Delta_j(\Pi_A, \Pi_B)$.
$= \sum D_j(\Pi_A, \Pi_B)$: Minimisation of the total sequence disruption, calculated as $\sum D_j(\Pi_A, \Pi_B) = \sum_{j=1}^{n} D_j(\Pi_A, \Pi_B)$.
$= \sum \Delta_j(\Pi_A, \Pi_B)$: Minimisation of the total time disruption, calculated as $\sum \Delta_j(\Pi_A, \Pi_B) = \sum_{j=1}^{n} \Delta_j(\Pi_A, \Pi_B)$.

Note that these objectives correspond to performance measures related to the disruption induced by a new schedule. Nevertheless, more than one disruption can be considered, and additional objectives may arise.

5.4.4 Additional Objectives

In this section, we cover other potentially interesting albeit not as thoroughly studied objectives. One common characteristic of some previous objectives is that they are linear on C_j. Each unit of time that a job is completed after its due date, adds one unit to the tardiness value (or w_j units if we are talking about weighted tardiness). However, very large deviations from the due date are more important than small deviations. A simple way of penalising more large deviations from the due date is by simply squaring the deviations. A breed of new objectives and criteria arise:

$\gamma = \max D_j$: Maximum squared deviation from the due date. This is calculated as $\max\{(C_1 - d_1)^2, (C_2 - d_2)^2, \ldots, (C_n - d_n)^2\}$.
$= \sum D$: Total squared deviation from the due date. Calculated as $\sum_{j=1}^{n}(C_j - d_j)^2$.
$= \overline{D}$: Average squared deviation from the due date, calculated as $\overline{D} = \frac{1}{n} \cdot \sum_{j=1}^{n}(C_j - d_j)^2$.

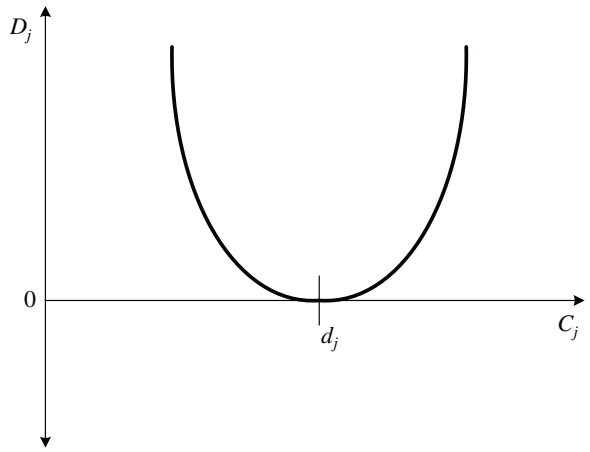


Fig. 5.9 Squared deviation from the due date, $(C_j - d_j)^2$

A typical D_j function is given in Fig. 5.9.

Note that the squared deviation is equal to the sum of the squared earliness and tardiness. However, if we are only interested in the earliness or in the tardiness all that we have to do is to add the square which results in objectives such as $\sum E^2$, $\sum T^2$ and all other alternatives. Also, it is possible to have weighted variants of these objectives, like $\sum w_j D_j^2$ or similar. The main drawback from the squared objectives is that they have little to no interpretation. It might be interesting to minimise the average squared earliness but the value of the objective itself will have no meaning. This is easily solved by considering other objectives, such as for example the percentage of early jobs, total earliness, etc.

In all tardiness measures, we have assumed that the due date d_j is a very specific point in time. Anything before or after this point in time is early or tardy, respectively. As expected, the defined objective of number of just-in-time jobs is a complicated one to maximise as it is hardly expected that all jobs finish exactly by their due date. As a matter of fact, for single machine or flow shop layouts, only a few jobs will exactly finish by their due dates, unless the due dates are very well spaced in time.

In practice, being a few hours early or late (even one or more days) is usually of no concern. As a result, it seems much more interesting to set the due date window or simply a due window, as seen in Sect. 3.2.2. Recall that d_j^- is the minimum due date, or 'earliest due date' before which the job will be considered early and d_j^+ is the maximum due date, or 'maximum due date' after which the job will be late. Under this scenario, the total earliness-tardiness is denoted and calculated as follows:

$\gamma = \mathrm{ET}_j^{ddw} = \sum_{j=1}^{n} E_j^{ddw} + T_j^{ddw}$: Total Earliness and tardiness minimisation, using a due date window, where $E_j^{ddw} = \max\{d_j^- - C_j, 0\}$ and $T_j^{ddw} = \max\{C_j - d_j^+, 0\}$.

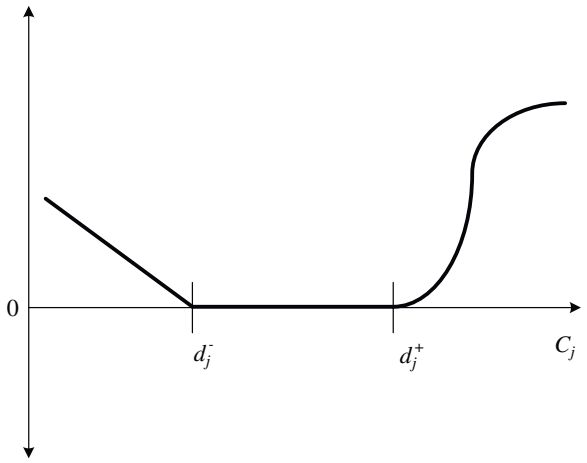


Fig. 5.10 A more realistic non-linear earliness and tardiness function with a due date window

Again, we could have all other possibilities as regards weights, different weights for earliness and tardiness or even some squared and weighted schemes. Picture for example the more realistic function of Fig. 5.10.

When some specific constraints are present in the system, lots of additional objectives can be derived. Setup times are a clear example. When setups require organisation or when they involve lots of time, money or both, it might be interesting to just minimise the number of setups, or to minimise the total setup time of the sequence. Instead of measuring time, one can measure costs. This is interesting when doing setups for a non-critical machine is cheap, for example, and carrying over setups for a critical or expensive machine might be more expensive. Furthermore, different setup operations might require more personnel or special tooling. Under these situations minimising the total setup cost is an interesting alternative.

Bringing up the cost minimisation objective raises a very complex and daunting issue. When doing scheduling, one is tempted to just state 'minimise the production costs'. However, this is easy to say, but very difficult to measure. While overall production costs are more or less accountable, specific production costs that actually might depend on how the products are scheduled is much more difficult. Furthermore, the costs of not meeting due dates are not easily measured. Some clients might be lost and measuring the lost revenue in these situations is all but easy. However, in some specific production companies, there might be an accepted and established production cost accounting. Under these settings, minimising production cost might be an interesting alternative to all other aforementioned performance measures.

5.5 Adding Weights, Priorities or Importance

We defined in Sect. 3.2.2 of Chap. 3 the weights of the jobs w_j. These weights are useful for expressing the importance, the priority, cost or whatever index of relative importance we might need. Given different weights for the jobs, we can modify all previous objectives to consider weights. Some examples are $\sum w_j C_j$ denoting $\sum_{j=1}^{n} w_j C_j$, or $\max w_j T_j$ denoting $\max_{1 \leq j \leq n} w_j C_j$.

One big drawback of adding weights to the objectives is that, in some cases, the value of the objective cannot interpreted in an easy way. However, weighting allows for more richer and realistic objectives. Picture for example the earliness-tardiness minimisation. Not all jobs are equally important and it is not the same not being just in time for a big order of an important client that being just in time for a make-to-stock or stock-replenishment order. For this last case of earliness-tardiness minimisation, we have the following notation:

γ = $\sum w_j ET_j$: Weighted sum of earliness and tardiness minimisation, calculated as $\sum w_j ET_j = \sum_{j=1}^{n} w_j ET_j = \sum_{j=1}^{n} w_j E_j + w_j T_j$.
= $\sum w_j E_j + w'_j T_j$: Weighted sum of earliness and tardiness minimisation with different weights for earliness and tardiness, calculated as $\sum w_j E_j + w'_j T_j = \sum_{j=1}^{n} w_j E_j + w'_j T_j$

Notice how the last function is even more interesting. By giving different weights to each job and also different weights for earliness and tardiness, we can closely match up the importance of being early and being late. Normally, the consequences of being early are not as dire as those of being late. We could have a more realistic function given in Fig. 5.11.

5.6 An Illustrative Example

In this section, we introduce an example to illustrate some of the objectives discussed before. The example is given for a permutation flow shop with the five jobs and four machines with due dates, so we will compute the values for some objectives in the problem $F4|prmu|\gamma$ or $F4|prmu, d_j|\gamma$, depending on whether we refer to non-due date related or due-date related objectives. Table 5.3 provides the data of one instance, i.e.: p_{ij} the processing times of job j on machine i, weights (w_j), and due dates (d_j) for each job.

We will compute some objectives for a sequence, for example $(1, 2, 3, 4, 5)$. Figure 5.12 shows the Gantt chart for our example. Table 5.4 shows the completion times of each job j on each machine m_i for the given sequence $(1, 2, 3, 4, 5)$, the completion times C_j (note that they are the same than C_{4j} values), the differences between $C_j - d_j$, i.e the lateness of each job L_j, and the weighted lateness $w_j L_j = w_j(C_j - d_j)$.

In general, the following formulae describe a way to compute the completion time of a job j in the position [j] of a sequence π in a problem $Fm|prmu|\gamma$:

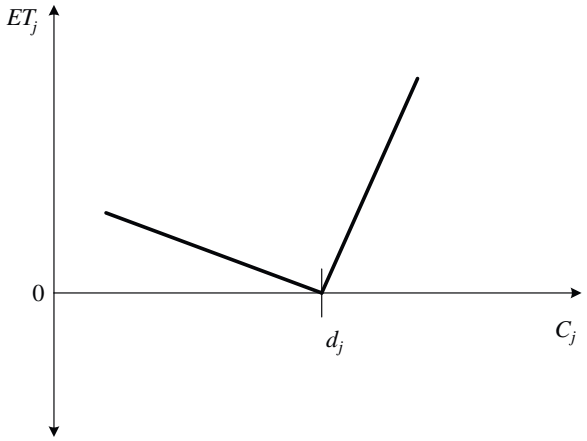


Fig. 5.11 Weighted earliness-tardiness function with different weights for earliness and tardiness

Table 5.3 Data for the illustrative example

Machine (m_i)	Job (j)				
	1	2	3	4	5
m_1	89	19	35	4	22
m_2	37	73	93	19	9
m_3	15	47	18	91	4
m_4	47	3	6	94	98
d_j	156	506	522	238	445
w_j	5	8	2	10	3

Table 5.4 Completion times of each job j on each machine m_i for the sequence (1, 2, 3, 4, 5)

c_{ij}	1	2	3	4	5
m_1	89	108	143	147	169
m_2	126	199	292	311	320
m_3	141	246	310	402	406
m_4	188	249	316	496	594

$$C_{i[j]} = \max\{C_{i-1[j]}, C_{i[j-1]} + p_{i[j]}\}$$

In Table 5.5, the first column includes all performance measures in Table 5.1. They are computed for each job. The last two columns show the maximum and the sum of the values, respectively, obtaining a great number of the objectives presented in the previous sections for the given sequence in this example. Note the completion times C_j for each job (note that they are the same than C_{4j} values of the previous table).

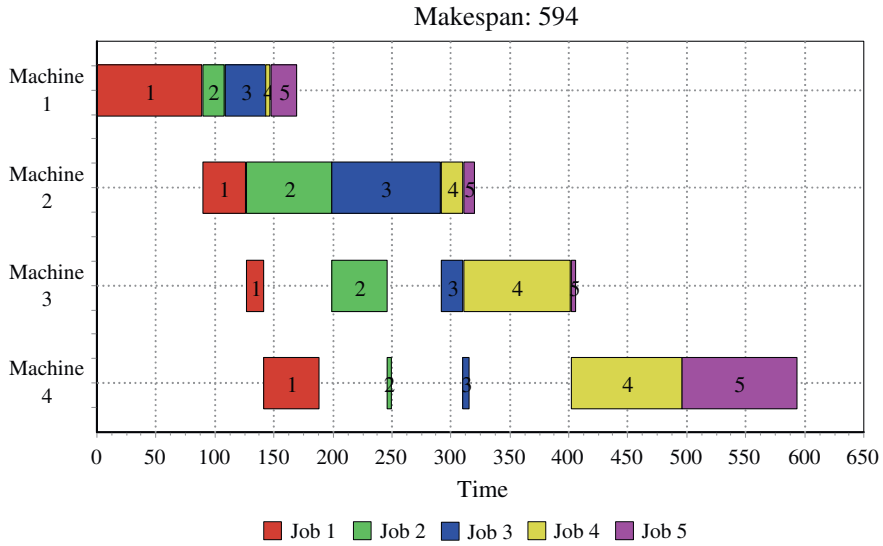


Fig. 5.12 Gantt chart for the permutation flow shop example with the sequence (1, 2, 3, 4, 5)

Table 5.5 Performance measures for each job j and objective values for the sequence (1, 2, 3, 4, 5)

	1	2	3	4	5	Max-form		Sum-form	
C_j	188	249	316	496	594	$C_{\max}$	594	$\sum C_j$	1843
w_jC_j	940	1992	632	4960	1782	$\max w_jC_j$	4960	$\sum w_jC_j$	10306
L_j	32	−257	−206	258	149	$\max L_j$	258	$\sum L_j$	−24
w_jL_j	160	−2056	−412	2580	447	$\max w_jL_j$	2580	$\sum w_jL_j$	719
T_j	32	0	0	258	149	$\max T_j$	258	$\sum T_j$	439
w_jT_j	160	0	0	2580	447	$\max w_jT_j$	2580	$\sum w_jT_j$	3187
E_j	0	257	206	0	0	$\max E_j$	257	$\sum E_j$	463
w_jE_j	0	2056	412	0	0	$\max w_jE_j$	2056	$\sum w_jE_j$	2468
U_j	1	0	0	1	1			$\sum U_j$	3
w_jU_j	5	0	0	10	3			$\sum w_jU_j$	18
V_j	0	1	1	0	0			$\sum V_j$	2
w_jV_j	0	8	2	0	0			$\sum w_jV_j$	10
ET_j	32	257	206	258	149	$\max ET_j$	258	$\sum ET_j$	902
w_jET_j	160	2056	412	2580	447	$\max w_jET_j$	2580	$\sum w_jET_j$	5655

Note that max-form objectives for U_j and V_j do not make sense, and they have not been included in the table.

5.7 Dealing with Conflicting Criteria: Multiobjective Scheduling

At the beginning of this chapter, we mentioned that most of the presented criteria may be conflicting. High machine utilisation usually results in high flowtime. Many other examples can be drawn. As a result of all this, the optimisation problem cannot be dealt with by optimising a single criterion, basically because by doing so, many other criteria will be severely affected.

The ideal situation would be to transfer everything into a 'profitability' measure and maximising this profitability would be the solution. The problem, as has been mentioned, is that this is an impossible task as it is not possible to calculate the costs and benefits of every possible schedule and production decision.

In any case, one has to consider the decision-making process as a more abstract entity. T'Kindt and Billaut (2006) state several items that make the decision-making process extremely complicated. Here we instantiate all these items for production scheduling:

- Usually it is not easy to define, without ambiguity, what is to be done or to be decided. In production scheduling problems, the very best one is to expect is an orientation or a vague indication of what is wanted.
- Put five top management individuals from the same company in a room and ask them about the production scheduling goal. Most probably, five different answers will be obtained and each one of them will be correct in its own right.
- Objectives, if rarely agreed upon, will evolve over time, i.e. they are not fixed.
- The best possible schedule can only be defined under some specific assumptions and normally for a simplified, approximated or theoretical scheduling problem. Therefore, it matters relatively not much if this solution is the best possible or just a very good one.
- There is a real possibility that a company, after using a scheduling system, changes objectives in view of 'what can be done', i.e. even if objectives are agreed upon, are measurable and there is no ambiguity, they might change.

Given all of the above, the only viable approach is to deal with several objectives *simultaneously*. By simultaneously we mean that several different performance measures have to be measured for every possible schedule and that a compromise decision has to be made. For example, let us assume that from five plan managers, three agree in that maximising machine utilisation is a key concern due to the very expensive equipment employed. The other two agree but insist that service level must not be sacrificed. One possible solution to this problem is to minimise the makespan $C_{\max}$ subject to a minimum service level, which could be defined as a maximum number of tardy jobs $\sum U_j$. Another possibility could be to minimise makespan and once the best makespan solution has been obtained, we could minimise the number of tardy jobs subject to no or moderate makespan deterioration. A third alternative would be to obtain several trade-off solutions between the best possible makespan

value and the number of tardy jobs, and the lowest possible number of tardy jobs and the makespan value.

There are many different techniques aimed at multi-objective optimisation. Reality is indeed multi-objective, therefore it is needed to approach real problems from a multi-objective perspective. However, before delving into this, we need to first introduce single objective scheduling methods. Therefore, multi-objective scheduling will be later discussed in Chap. 10.

5.8 Conclusions and Further Readings

In the last two chapters, the different scheduling models were presented and many possible scheduling constraints were detailed. It was therefore expected that scheduling criteria were equally as rich and varied. We have made two main classifications of objectives: those based on completion times of the jobs and those based on the due dates. It is safe to say that a sizeable part of the scheduling literature is concerned with makespan minimisation. However, the practical relevance of this objective is debatable, specially with such a high prevalence.

In the long run, the best objective is the one that is trusted. This trust comes after the realisation of several schedules for which objectives where known and for which production and delivery went as expected. Advanced scheduling at production plants usually entail ad-hoc and evolved objectives that satisfy several criteria simultaneously.

Regarding further readings, a discussion about the relation between scheduling objectives and corporate objectives are given as back as in Gupta and Dudek (1971) or more recently, in more detail in Gary et al. (1995). This issue is also commented in the paper of Stoop and Wiers (1996). Some authors even apply quality control charts to measure the schedule performance (MacCarthy and Wilson 2001). The notation employed in Sect. 5.3 is rather standard and can be found e.g. in French (1982) or Pinedo (2012), among many others. Proofs that the maximisation of utilisation is equivalent to minimisation of makespan and equivalent to the minimisation of idletime, including heads and tails are commented in most textbooks, such as Conway et al. (1967), Baker (1974), French (1982), Błazewicz et al. (2002), Brucker (2007), Pinedo (2009, 2012), Baker and Trietsch (2009). The aforementioned textbooks usually commented and detailed the objectives presented here.

Regarding specific contributions, makespan minimisation is reviewed in many different papers, as the review papers are mostly linked with the processing layout or specific constraints. For the flow shop layout, some interesting reviews centered around makespan are those of Framinan et al. (2004), Ruiz and Maroto (2005), Hejazi and Saghafian (2005). Parallel machines have received attention in the reviews of Cheng and Sin (1990) and Mokotoff (2001) although these last review papers are not solely centered around makespan. The same is applicable to the hybrid shop layouts (mainly hybrid flow shops), which are reviewed in Linn and Zhang (1999), Vignier et al. (1999), Wang (2005), Ruiz and Maroto (2006), Quadt and Kuhn (2007),

Ribas et al. (2010), Ruiz and Vázquez-Rodríguez (2010). There is a whole body of scientific research dealing with inserted idle time in scheduling, being Kanet and Sridharan (2000) a good review on the topic. Good reviews of due-date related research are available from Sen and Gupta (1984), Baker and Scudder (1990) and more recently, Vallada et al. (2008), Jozefowska (2007) is a book solely devoted to Just-In-Time scheduling.

References

Baker, K. R. (1974). *Introduction to Sequencing and Scheduling*. John Wiley & Sons, New York.

Baker, K. R. and Scudder, G. D. (1990). Sequencing with earliness and tardiness penalties: a review. *Mathematics of Operations Research*, 15:483–495.

Baker, K. R. and Trietsch, D. (2009). *Principles of Sequencing and Scheduling*.Wiley, New York.

Błazewicz, J., Ecker, K. H., Pesch, E., Schmidt, G., and Węglarz, J. (2002). *Scheduling Computer and Manufacturing Processes*. Springer-Verlag, Berlin, second edition.

Brucker, P. (2007). *Scheduling Algorithms*. Springer, New York, fifth edition.

Cheng, T. C. E. and Sin, C. C. S. (1990). A state-of-the-art review of parallel-machine scheduling research. *European Journal of Operational Research*, 47(3):271–292.

Conway, R. W., Maxwell, W. L., and Miller, L. W. (1967). *Theory of Scheduling*. Dover Publications, New York. Unabridged publication from the 1967 original edition published by Addison-Wesley.

Framinan, J. M., Gupta, J. N. D., and Leisten, R. (2004). A review and classification of heuristics for permutation flow-shop scheduling with makespan objective. *Journal of the Operational Research Society*, 55(12):1243–1255.

French, S. (1982). *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*. Ellis Horwood Limited, Chichester.

Gary, K., Uzsoy, R., Smith, S. P., and Kempf, K. (1995). Measuring the quality of manufacturing schedules. In Brown, D. E. and Scherer, W. T., editors, *Intelligent Scheduling Systems*, volume 4 of *Operations Research/Computer Science Interfaces*, pages 129–154, Dordrecht. Kluwer Academic Publishers.

Gupta, J. N. D. and Dudek, R. A. (1971). Optimality Criteria for Flowshop Schedules. *IIE Transactions*, 3(3):199–205.

Hejazi, S. R. and Saghafian, S. (2005). Flowshop-scheduling problems with makespan criterion: a review.*International Journal of Production Research*, 43(14):2895–2929.

Jozefowska, J. (2007). *Just-in-Time Scheduling*. Springer, Berlin.

Kanet, J. J. and Sridharan, V. (2000). Scheduling with inserted idle time: Problem taxonomy and literature review. *Operations Research*, 48(1):99–110.

Linn, R. and Zhang, W. (1999). Hybrid flow shop scheduling: A survey. *Computers & Industrial Engineering*, 37(1–2):57–61.

MacCarthy, B. L. and Wilson, J. R., editors (2001). *Human performance in Planning and Scheduling*. Taylor & Francis.

Mokotoff, E. (2001). Parallel machine scheduling problems: A survey. *Asia-Pacific Journal of Operational Research*, 18(2):193–242.

Pinedo, M. (2009). *Planning and Scheduling in Manufacturing and Services*. Springer, New York, second edition.

Pinedo, M. L. (2012). *Scheduling: Theory, Algorithms, and Systems*. Springer, New York, fourth edition.

Quadt, D. and Kuhn, D. (2007). A taxonomy of flexible flow line scheduling procedures. *European Journal of Operational Research*, 178(3):686–698.

Ribas, I., Leisten, R., and Framinan, J. M. (2010). Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Computers & Operations Research*, 37(8):1439–1454.

Ruiz, R. and Maroto, C. (2005). A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research*, 165(2):479–494.

Ruiz, R. and Maroto, C. (2006). A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research*, 169(3):781–800.

Ruiz, R. and Vázquez-Rodríguez, J. A. (2010). The hybrid flowshop scheduling problem. *European Journal of Operational Research*, 205(1):1–18.

Sen, T. and Gupta, S. K. (1984). A state-of-art survey of static scheduling research involving due dates. *OMEGA, The International Journal of Management Science*, 12(1):63–76.

Stoop, P. and Wiers, V. (1996). The complexity of scheduling in practice. *International Journal of Operations and Production Management*, 16(10):37–53.

T'Kindt, V. and Billaut, J.-C. (2006). *Multicriteria Scheduling: Theory, Models and Algorithms*. Springer, New York, second edition.

Vallada, E., Ruiz, R., and Minella, G. (2008). Minimising total tardiness in the *m*-machine flowshop problem: A review and evaluation of heuristics and metaheuristics. *Computers & Operations Research*, 35(4):1350–1373.

Vignier, A., Billaut, J.-C., and Proust, C. (1999). Les problèmes d'ordonnancement de type flow-shop hybride: État de l'art. *RAIRO Recherche opérationnelle*, 33(2):117–183. In French.

Wang, H. (2005). Flexible flow shop scheduling: optimum, heuristics and artificial intelligence solutions. *Expert Systems*, 22(2):78–85.

Chapter 6
Construction of Scheduling Models

6.1 Introduction

After the definition of the main elements constituting scheduling models, this chapter is devoted to the process of actually building scheduling models. Manufacturing scheduling models are intended to simplify and to map the relevant real-world problem settings in a (usually formal) model which will be used to yield—via scheduling methods—a solution to this formal model. This solution would be then transferred to and implemented in the real-world setting.

Mathematical (i.e. formal) models include a combination of logical dependencies, mathematical relationships such as equations and inequalities, data structures and criteria. The quality of the answers to the real-world problem's questions which are produced by a model obviously depends on the accuracy of the structure and of the data of the model. Here, we describe and discuss the most prominent approaches to construct scheduling models.

More specifically, in this chapter we

- present basic approaches to construction of scheduling models (Sect. 6.2),
- describe decomposition and reintegration as a basic tool for handling model complexity (Sect. 6.3),
- address aggregation and disaggregation in manufacturing scheduling to reduce a model's complexity (Sect. 6.4),
- sketch out what validation and verification means in the context of (scheduling) models (Sect. 6.5).

6.2 Basic Approaches to Construction of Scheduling Models

Recall from Fig. 1.2 in Chap. 1 that we mapped the flow of a decision process where we started from a real-world problem for which a decision is required. Then a formal model is derived by means of simplification and formalisation. By some formal

J. M. Framinan et al., *Manufacturing Scheduling Systems*,
DOI: 10.1007/978-1-4471-6272-8_6, © Springer-Verlag London 2014

procedure, this formal model is solved and this formal solution is transferred to a real-world solution which is implemented. In this section, we mainly address the 'upper right corner' of Fig. 1.2, i.e. we will focus on simplification and formalisation, especially the transfer of real-world relations into a formal, mathematical decision model and the techniques to set up a respective formal model.

Formulation of a decision model for manufacturing scheduling requires adequate real-world assumptions about constraints, objectives, tasks and logical dependencies and their transfer to the formal sphere. Both, decision variables and parameters/coefficients of the model have to be defined precisely and on an adequate level of detail (see also Chap. 2). 'Adequate' here means that both, the real-world aspects of the problem are sufficiently represented in the model *and* the model itself is expected to be tractable. Complexity aspects, therefore, are of eminent importance in model construction.

In Sect. 2.3.2.2 a classification of complexity has been given. According to this classification, note that model-oriented complexity in manufacturing scheduling might result both from mass aspects (multiplicity and variance) and chaos aspects (ambiguity and changeability) such as:

- number and heterogeneity of operations/jobs (including (non-) pre-emptability, etc.),
- number and heterogeneity of resources,
- number and heterogeneity of objectives,
- number and heterogeneity of possible decisions/decision variables (including, e.g. in flow shops the modelling decision whether only permutation flow shop solutions are considered or whether the permutation might vary from machine to machine),
- type, number and heterogeneity of interactions between operations, resources, objectives and/or decisions (including both, real-world and formal aspects such as (non-) linearity, etc.),
- ...

General approaches to manage these complexity drivers include abstraction (including deletion of respective aspects from the model) and coarsening (including decomposition, aggregation/disaggregation, e.g. with respect to time, operations, resources, objectives, ...).

While the above aspects of model building refer to both, the real-world and the formal sphere, simultaneously the formal type of model for the scheduling problem under consideration has to be determined. Several approaches based on different techniques are available. In Sect. 6.2.2 some of these approaches will be addressed, i.e. linear programming, mixed-integer programming, agent-based approaches, stochastic approaches and fuzzy logic and Petri nets.

Before going more into detail, it has to be mentioned that several standard modelling approaches are available for certain types of problems. These types of models are often somewhat like a reference point if a new (or modified) problem is addressed. Figure 6.1 shows some indications for the choice of the model type according to some classification criteria.

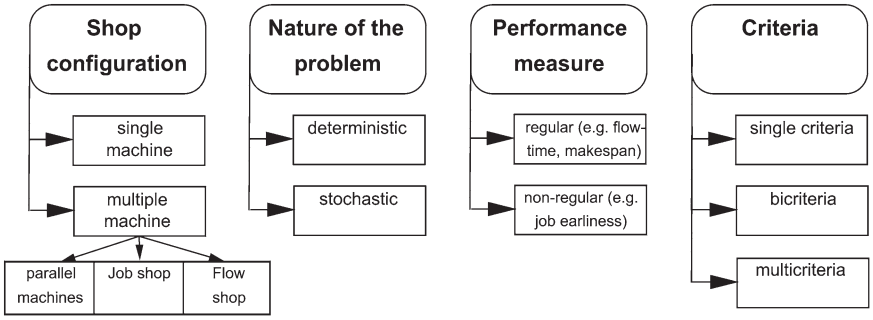


Fig. 6.1 Framework for selection of scheduling model types (Nagar et al. 1995)

6.2.1 Variables, Constraints and Objectives

The goal of a manufacturing scheduling model is to generate a valid schedule, i.e. a feasible and good, efficient or even optimal assignment of operations to the time scale. Schedules can be characterised via their activities/jobs/operations, machines/resources, hard and soft constraints, objectives and possibly other problem-specific aspects.

As already mentioned in Chap. 2 and later developed in Chap. 3, the basic element (atom) of every scheduling problem is the single operation. In its simplest version, this operation is assigned to a pair of exactly one job j and one machine i with a pre-specified and deterministic processing time. In consequence, the operation is fixed with respect to its machine index i, its job index j and its duration p_{ij}.

However, in more complex settings, an operation might also be split (or, on the contrary, be clustered) to batches of one or several jobs' overall processing requirements on the machine (i.e. more than one operation per job exists on one machine or several operations of several jobs on one machine are clustered and processed as one 'macro' operation) and/or to several machines on a certain manufacturing stage (e.g. in hybrid structures with multiple, possibly different machines on one stage of the manufacturing system). Or 'macro' operations, with respect to their processing time and/or resource requirements, might be fixed in advance or are variable themselves. Therefore, on one hand, depending on the specific setting, an operation might be broken down to several sub-operations on several machines. In turn, these sub-operations must add up to the overall operation under consideration. Every sub-operation must be dimensioned according to the resources/machine capacity it needs, i.e. according to its resource's/machine's production function. On the other hand, several operations might be processed together, in parallel (e.g. in an oven) or in sequence (because of their same setup status) and may or must be separated after this joint processing for the subsequent processing of the respective jobs. Formal approaches to settings like these are only partly standardised (batch processing, batch scheduling, simultaneous lot sizing and scheduling) and are often rather specific for the respective

problem under consideration. This is the reason why we do not further specify these aspects here.

With respect to the availability of resources, every operation (or sub-operation) on one hand has to regard the availability of the machine(s) it is assigned to, i.e. constraints have to guarantee that only a limited number of operations (usually at maximum one operation) are executed simultaneously on the respective machine. On the other hand, limitations on possibly required additional resources (manpower, etc.) might have to be considered as well if this is relevant for the problem under consideration.

With respect to the availability of jobs (or operations), constraints relative to predecessors and successors of the operation under consideration have to be included. Usually, all predecessor operations have to be finished before the current operation can start. This is what is called end–start relation with minimum time distance (usually, in scheduling problems, this minimum time distance is 0) in network planning models. Supposing SO_{ij} being the starting time of the operation of job j on machine i and C_{ij} the respective finishing time, this constraint can be expressed as $SO_{ij} \geq C_{i'j'} + \text{MinTimeDist}\big((i, j), (i', j')\big)$, for all operations (i', j') being (immediate) predecessors of operation (i, j). Both, SO_{ij} and C_{ij}, are continuous-type variables of the scheduling model.

However, recall from Sect. 4.2.1 that other type of time distance, i.e. apart from the above described end–start relation, also can occur. Usually, not both, predecessor and successor relations have to be mapped in a model since they result from each other. (Jobs' or operations' release times and deadlines can be easily included in the SO_{ij}, C_{ij} notation.)

Basically, there are two different types of predecessors for every operation, i.e. fixed and changeable ones. Fixed predecessors result from physical constraints of operations or jobs, e.g. from pre-specified job/product routings. (MinTimeDist in the above formula might represent, for fixed predecessors, transportation times between two consecutive operations of one job.) Changeable predecessors result from the different decisions/schedules which represent the different solutions of the scheduling problem. A best choice of these changeable predecessors represent the optimal solution to the manufacturing scheduling problem under consideration. (MinTimeDist in the above formula might represent, for changeable predecessors, sequence-dependent setup times.)

Standard scheduling objective functions obviously can also easily be described by using the above SO_{ij}, C_{ij} formulas (see also Sect. 5.3). For a given solution (i.e. a complete assignment of operations to the time scale, ending times of jobs' last operations immediately define makespan, flowtime, utilisation, etc.).

Many other aspects might be included in a manufacturing scheduling model, such as:

- predictable or unpredictable non-availability periods of machines/resources,
- non-availability of materials, fixtures or tools (being also relevant with a short-term horizon, e.g. in just in time setting),

- pre-emption of jobs (with restarting of jobs/operations later on from scratch or from the status when pre-emption occurred or something in between),
- logical dependences of variables/jobs/... (e.g. joint production settings, cyclic production calendars, joint machine (non-) availability times),
- ...

All of these problem extensions require specific model expansions. The detailed description of these expansions is beyond the scope of this chapter.

The decision problem itself, however, is not covered completely by the above descriptions as is already indicated by the above discussion of fixed and changeable predecessors. In general, for all combinations of changeable predecessors (i_1, j_1) and (i_2, j_2), a decision has to be taken whether an operation (i_1, j_1) will precede (not necessarily immediately) an operation (i_2, j_2) or vice versa. In scheduling models, this is either included implicitly, e.g. if a specific permutation defines this precedence configuration in a permutation flow shop problem. Or the decision is explicitly designed, usually by defining binary variables, e.g. as

$$x_{(i_1,j_1),(i_2,j_2)} = \begin{cases} 1 & \text{if } (i_1, j_1) \text{ precedes } (i_2, j_2) \\ 0 & \text{if } (i_2, j_2) \text{ precedes } (i_1, j_1) \end{cases}$$

for all pairs of operations (i_1, j_1) and (i_2, j_2) for which both relative sequences are relevant, i.e. which might return an infeasibility if they are at least partly processed in parallel. (This is obviously the case if both operations take place on the same machine, which is opposed to be able to process no more than one operation at a time, but might happen sometimes also otherwise.)

Obviously, a necessary constraint for feasibility of a schedule then is

$$x_{(i_1,j_1),(i_2,j_2)} + x_{(i_2,j_2),(i_1,j_1)} = 1.$$

Additionally, a transitivity relation must hold, i.e.

$$x_{(i_1,j_1),(i_2,j_2)} = 1 \ \wedge \ x_{(i_2,j_2),(i_3,j_3)} = 1 \ \Rightarrow \ x_{(i_1,j_1),(i_3,j_3)} = 1$$

for all relevant triples, to avoid circles of precedence.

If such an explicit representation of precedence alternatives is used, the number of binary variables will usually be very large, and sophisticated approaches to reduce this number are very welcome as one approach to complexity reduction. Another way to reduce complexity, which might be applied in combination with the first one, is the design and the application of heuristic solution procedures which are addressed later in this book.

Time variables SO (and at least implicitly also C) and precedence variables x can be linked as follows. Suppose that only operations on a single machine i (maybe in a multi-machine setting) are considered. Then

$$M \cdot (1 - x_{(i,j_1),(i,j_2)}) + SO_{ij_2} \geq SO_{ij_1} + p_{ij_1}$$
$$M \cdot x_{(i,j_1),(i,j_2)} + SO_{ij_1} \geq SO_{ij_2} + p_{ij_2}$$

for all i, j_1 and j_2 with $j_1 \neq j_2$ and a large number M represents both alternatives of precedence for the operations (i, j_1) and (i, j_2), see Williams (2013). This representation of precedence alternatives on one machine requires only half of the binary variables as compared with the full table since (i, j_1) preceding (i, j_2) means automatically (i, j_2) not preceding (i, j_1) and vice versa.

Using (and possibly expanding or modifying) the approaches described in this section, generates a decision model for manufacturing scheduling problems, defining variables, constraints as well as objective function(s) in one model which is the starting point for the optimisation or search process to generate a good, efficient or even optimal solution.

So far, in this section we exclusively addressed the design of scheduling models with constraints that were strictly to be kept, i.e. hard constraints. However, in real-world settings sometimes constraints are not as strict, may it be because of the existence of processing alternatives not mapped in the model, for (time) buffers included only implicitly in the model or for other reasons. In such cases, a relaxation might be included in the respective constraints. E.g., in the above constraints, a time relaxation of size u (as a variable) might be included, e.g. as

$$M \cdot (1 - x_{(i,j_1),(i,j_2)}) + SO_{ij_2} \geq SO_{ij_1} + p_{ij_1} - u$$
$$M \cdot x_{(i,j_1),(i,j_2)} + SO_{ij_1} \geq SO_{ij_2} + p_{ij_2} - u$$

meaning here that the strict precedence requirement between the operations is replaced by some allowance of parallelity, e.g. in real-world terms by using a time buffer. However, to avoid unlimited violation of the precedence requirement, u has to be bounded either primarily by an additional constraint (addressing the size of u) or by dually penalising a positive value of u in the objective function. An alternative way of handling such soft constraints is by using fuzzy set approaches where the degree of membership corresponds to the amount by which the constraint is violated.

Another setting demanding for similar approaches are conflicting constraints in the original model, resulting in the non-existence of a feasible solution. Relaxing these constraints primarily or dually or using goal programming approaches might be adequate as well because of implicit buffers and/or technological or process flexibility not being included in the model.

Many other aspects of real-world scheduling settings can be mapped in formal scheduling models. Lead times, release or processing patterns, late modifications of jobs (with respect to time, size, design, ...), quality aspects and many other might increase the complexity of the real-world scheduling task under consideration. Also variability aspects, e.g. with respect to processing times might be a severe issue in many situations, may it be influenced by a human component, by planning deficiencies, by technological variability, etc. The enormous diversity of these aspects

does not allow an comprehensive description of the respective modelling approaches which are by far not standardised.

Concluding this section, we point out:

1. In addition to the above-mentioned modelling aspects, the modelling process itself has to be executed regarding also the following. 1) The model must be (easily) understandable for everybody who is basically able to understand quantitative modelling approaches and gets into contact with the model. 2) The model must be easy/adequately to solve, to maintain and to update with respect to possible different future real-world settings (see also Williams 2013). 3) And the model—sometimes in contrast to solution efforts with respect to quantitative models—additional insight in the real-world setting gained from the (often iterative) modelling process itself is frequently even more valuable than solving the model to optimum.
2. The quality of the solution of a scheduling problem depends both on model formulation *and* on the solution method. Both these levels of dealing with manufacturing scheduling problems are highly interdependent and therefore have to be considered jointly, may it be simultaneously or in an iterative loop.

6.2.2 Some Basic Modelling Techniques Used in Manufacturing Scheduling

A real-world manufacturing scheduling problem can often be (quantitatively) modelled in more than one way. This results in different model and solution complexity as well as different solution quality. We therefore sketch out several typical modelling approaches in the sequel.

6.2.2.1 Binary/Integer Programming Modelling

As already indicated by the formulas in the previous section, mixed-binary formulations of scheduling problems are very popular. Because of the wide dissemination of both, mixed-integer model formulation as well as respective solution techniques, these approaches contribute a lot to understanding and solution of manufacturing scheduling problems. However, because of the enormous effort required to solve MIP models to optimality, only small problem instances might be solved accordingly.

Early basic mixed-binary formulations of scheduling problems are presented, among others by Bowman (1959), Wagner (1959), Dantzig (1960), Manne (1960), Greenberg (1968). Some of these models will be addressed in Chap. 8. Here, we will only present the model of Greenberg (1968) as a straightforward formulation of the standard n job, m machine job shop scheduling problem with makespan objective.

Table 6.1 (Modified) Greenberg model: notation

Data	
n	Number of jobs, $j = 1, \ldots, n$
m	Number of machines, $i = 1, \ldots, m$
R_j	Vector that specifies the machine sequence of the operations of job j (supposed to be a permutation of $1, \ldots, m$ which means that every job visits every machine)
p_{ij}	Processing time of job's j operation on its i-th machine (i.e. on machine $R_j(i)$)
M	Large number
Variables	
C_{max}	Makespan of the schedule
SO_{ij}	Starting time of processing job j on machine i (continuous variable)
x_{ijk}	Binary variable, being 1 if, on machine i, job j starts before job k, 0 otherwise

(Modified) Greenberg model (Greenberg 1968)

Objective function:

$$\min \; C \tag{6.1}$$

Constraints:

$$SO_{R_j(i-1),j} + p_{i-1,j} \leq SO_{R_j(i),j} \qquad \forall\, j = 1, \ldots, n,\; i = 2, \ldots, m \tag{6.2}$$

$$SO_{ij} + x_{ijk} \cdot p_{ij} \leq SO_{ik} + x_{ikj} \cdot M \qquad \forall\, i, j, k \text{ with } j \neq k \tag{6.3}$$

$$x_{ijk} + x_{ikj} = 1 \qquad \forall\, i, j, k \text{ with } j \neq k \tag{6.4}$$

$$SO_{R_j(m),j} + p_{ij} \leq C \qquad \forall\, j \tag{6.5}$$

$$x_{ijk} \in \{0, 1\} \qquad \forall\, i, j, k \tag{6.6}$$

Combining constraints (6.5) and objective function (6.1), the minimum value for C is the minimum makespan since the left-hand side of (6.5) defines the finishing time of every job's last operation. Constraint set (6.2) guarantees the machine sequence per job, saying that job's $j\,R_j(i-1)$-th operation must be finished before its $R_j(i)$-th operation can start. Constraint set (6.3) guarantees that no two jobs are simultaneously processed on the same machine. Finally, constraint set (6.4) guarantees that, on machine i, either job j is processed before job k or vice versa.

Clearly, this model can be easily adjusted to some well-known other settings, such as, e.g. flowtime as objective function or including due dates (and related objective functions) or deadlines.

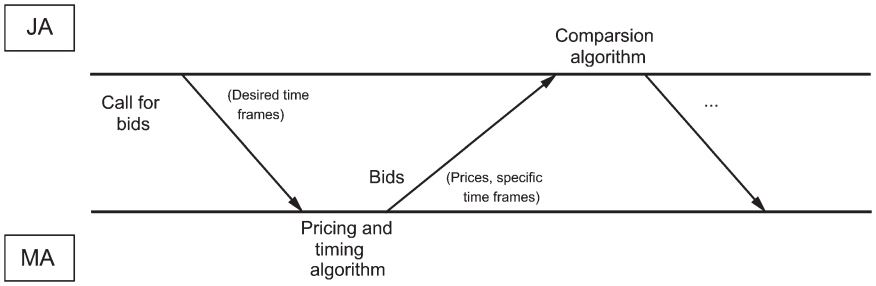


Fig. 6.2 Overview of bidding and pricing process (Pinedo 2012)

6.2.2.2 Agent-Based Approaches

In contrast to 'true' optimisation approaches to scheduling as used, e.g. in mixed-integer linear programming, Artificial Intelligence (AI) aims on satisfying aspiration levels, i.e. it is sufficient to generate solutions which are accepted by the decision-maker(s). (Therefore, 'true' optimisation approaches might be interpreted somewhat like an extreme case in the context of AI.)

Agent-based approaches (ABA) represent a special kind of AI methods. They first decompose the overall model into sub-models (agents) with specific structure, specific individual tasks and specific procedures for their 'local', agent-specific decision-making. And, second, these agents iteratively communicate with each other according to pre-specified procedures intending to give improved information to the counterpart for solving its local decision problem and, finally, to generate an accepted solution to the overall decision problem. The iterative process will be stopped when some pre-specified stopping criterion is fulfiled (e.g. with respect to solution quality or computation time).

Referring to manufacturing scheduling problems, in a most simple version of an ABA, agents might be defined as job agents (JA) or machine agents (MA). JAs internally assign a job's operations to the time scale based on the machine resources (time frames) received from the MAs, and compare the solution with former or alternative ones (comparison algorithm). On the level of communication, JAs demand machine capacity from the MAs intending to improve their current local performance. MAs locally, i.e. for the specific machine, bring jobs' operations into sequence and, in turn as feedback in communication, offer time windows for a job's operation on this machine to the JAs. Information exchange between the agents might be primal (e.g. explicit time information) and/or dual (i.e. (opportunity) cost and pricing information). This process is mapped in Fig. 6.2.

MAs will locally often use simple or more sophisticated dispatching rules or one-machine optimisation procedures, keeping in mind the requirements communicated to them by the JAs. Bottleneck machines will play a dominant role while non-bottleneck machines will often only be regarded as subordinated in the communication process. JAs will, e.g. locally determine job's earliest availability time

for a certain operation on a specific machine (according to the machine sequence of operations of the specific job), and might determine a priority indicator of the specific job (e.g., according to the tightness of job's due date or the importance of the respective customer) and will give this information to the MA(s). Exclusive priority (rule) scheduling as a job-oriented and exclusive bottleneck scheduling as a resource-oriented approach represent the two extremes for rule-based ABA. Every plausible combination of these extreme cases might be considered of course as well (see Schmidt 1992, or Shen and Norrie 1998 for an overview). Obviously, ABA is rather flexible and can easily be applied to open, dynamic scheduling environments which makes them attractive for respective real-world settings while many (most) 'true' optimisation-oriented scheduling approaches focus on more or less 'closed' static settings. However, on one hand, the higher flexibility of ABA has to be traded-off with the lower solution quality (with respect to (sub-) optimality) as compared with optimisation-oriented approaches. On the other hand, it has to be regarded that an intrinsic problem of ABAs, not only in scheduling, is the myopic perspective of the agents. Generating an overall scheme for an ABA which, as far as possible includes the overall perspective of the problem under consideration by an adequate communication scheme is the challenge of modelling ABAs.

The basic principles of ABAs in manufacturing scheduling have been described above. More specific information can often only be given with respect to the specific problem setting under consideration. We will therefore not go into detail further here. However, in addition, we point out that the concept of ABAs to manufacturing scheduling presented so far supposes that the required information is

- available wherever it is required (or even all information is available everywhere in the system—which is not a true decentralised approach to the problem) and
- transferred truly between the agents, without any opportunistic behaviour.

We only mention that it might be necessary to check these requirements in real-world settings of manufacturing scheduling and, in consequence, adjust the communication procedures accordingly.

6.2.2.3 Modelling Approaches Including Uncertainty

For several reasons non-deterministic settings might occur in manufacturing scheduling problems. Processing times might be known only by their probability distribution and/or by some range, machine/resource availability might be stochastic with respect to time and/or size, availability (release dates) and priority of jobs might be non-deterministic, etc. Sometimes, not even probability data or ranges are known but only partly quantifiable relative information is given, such as high, medium and low with respect to, e.g. processing times.

There are several ways to deal with uncertainty in manufacturing scheduling problems. The simplest way is just to ignore uncertainty and to set up and solve the model with 'some kind of' expected values for the uncertain parameters. From a formal point of view, this might be an important drawback of respective approaches.

However, keeping in mind the often rather short-term horizon perspective of real-world scheduling problems, ignoring uncertainty might be not as critical from an application point of view. Nevertheless, the extent of uncertainty's influence is often not known in advance. Therefore, this approach seems to be appropriate in situations where uncertainty will have only little influence on the problem and its solution. Only slightly more elaborated are models which include buffers for uncertainty, e.g. by expanding processing times or by reducing capacities. The 'correct' determination of the size of these buffers is a most challenging task. Both approaches might run into feasibility and/or optimality problems. In addition to the application of deterministic approaches to non-deterministic problems, sensitivity analyses can be executed, may it be by using some kind of simulation or by applying explicit formal approaches (if available).

However, if uncertainty is expected to be an important issue with respect to the problem under consideration, is has to be considered explicitly—if possible. Two basic approaches are intensively discussed in the literature, i.e. stochastic scheduling approaches and fuzzy logic approaches.

Restricting to modelling issues in this chapter, *stochastic models* (in manufacturing scheduling) might be relatively easy formulated by just defining certain parameters of an originally deterministic model to be stochastic. Problems then result 'only' from model solution and not from model formulation. However, this interpretation is profoundly naive. Modelling and solution aspects usually influence each other and should be regarded jointly. Nevertheless, we will not deal with solution methods for stochastic scheduling problems here. Instead we point the reader to respective references, e.g. the book of Pinedo (2012), where stochastic scheduling problems and solution methods are discussed. Without going into detail and well-known for many stochastic problem settings also outside of manufacturing scheduling, stochastic aspects significantly increase the complexity of a model and the solution process. A treatable deterministic model will often become intractable if stochastic model components occur. Therefore, on one hand, valid results for stochastic scheduling problems will usually only be deducible for fairly simple problem settings, e.g. single-stage problems with one or more machines or simple flow shop or job shop problems. On the other hand, results and interpretations for stochastic scheduling problems will often differ from those for deterministic problems: (Stochastic) dominance results, deriving hints for scheduling policies or shop capacities (machines, buffers, . . .) instead of detailed solutions might be the focal aspect instead of deriving single good or optimal solutions.

If uncertain problem data are available in some simple kind of distribution (e.g. triangular or trapezoidal) or only in some qualitative or linguistic categories (e.g. as low, medium or high, which is supposed to be representable in some trapezoidal distribution), fuzzy set or fuzzy logic approaches might be applied to manufacturing scheduling problems. For example, a due date for a job might be not as strict as it is often supposed in basic models. Traditionally, due dates are included in scheduling problems by respective objective functions such as lateness and tardiness. Missing the due date is then penalised. However, if due dates are not as strict, they might be represented by a trapezoidal distribution, representing the categories early, normal

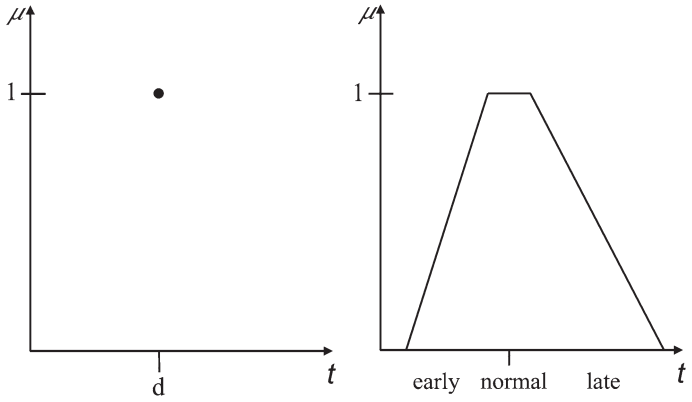


Fig. 6.3 Fixed due date and fuzzy due date

and late as is shown in Fig. 6.3, where μ indicates the so-called membership function with values between 0 and 1, and 1 indicating that the relationship under consideration is completely fulfilled while other values of μ can be interpreted accordingly.

Apart from single parameter data, also the objective function(s) and or the information on constraint satisfaction might be available only in linguistic terms. For example, makespan might be classified into short, normal and long. The same holds for the classification of trade-offs between multiple objective functions which, depending on the specific problem might be classified only linguistically. Flexible resources can be addressed accordingly (see, e.g. Slowinski and Hapke 2000).

As compared with stochastic programming approaches to manufacturing scheduling, fuzzy approaches on one hand are obviously somewhat simpler, also because of the rather simplified distributions supposed in fuzzy approaches. On the other hand, solving a fuzzy scheduling model usually also yields fuzzy result information, giving the decision-maker some impression of the expected effect of fuzziness, also with respect to the robustness of a solution subject to the fuzzy components of the model.

6.2.2.4 Other Modelling Approaches

Apart from those described above, there are several other approaches to model manufacturing scheduling problems, e.g. Petri nets and object-oriented approaches. We will just sketch out these approaches here.

From an application point of view, the main features of *Petri net-based approaches* for scheduling are twofold. First, they are easily able to handle multiple lots for complex relations that may exist among jobs, routes, machines and material handling devices, i.e. Petri nets provide an efficient method for representing concurrent activities, shared resources, precedence constraints and multiple lot sizes. Second, the generated schedule is event-driven (this facilitates real-time implementation),

deadlock-free (since the Petri net model of the system can be a detailed representation of all the operations and resource-sharing cases, a generated schedule is the one from the system's initial condition to the finally desired one; it thus avoids any deadlock) and optimal or near-optimal with respect to makespan.

Within the Petri net approach, a bottom-up method is used to synthesise the Petri net model of a system for scheduling. First, a system is partitioned into sub-systems according to the job types, then sub-models are constructed for each sub-system, and a complete net model for the entire system is obtained by merging Petri nets of the sub-systems via the shared resources.

Although the underlying problems of scheduling tasks are highly complex a simplified model can sometimes be obtained using *object-oriented techniques*. As described by Błażewicz et al (2007), Schmidt (1996), object-oriented modelling attempts to overcome the disadvantage of modelling data, functions, and interactions between both, separately. The different phases of the modelling process in object-oriented approaches are analysis, design and programming. Analysis serves as the main representation formalism to characterise the requirements from the viewpoint of the application; design uses the results of analysis to obtain an implementation-oriented representation, and programming means translating this representation using some programming language into code. Comparing object-oriented modelling with traditional techniques, its advantages lie in data abstraction, reusability and extensibility of the models, better software maintenance and direct compatibility of the models of different phases of the software development process. A model built by object-oriented analysis consists of a set of objects communicating via messages which represent dynamic relations of pairs of them. The main static relations between pairs of objects are generalisation/specialisation and aggregation.

6.3 Complexity Reduction in Manufacturing Scheduling Using Decomposition and Re-integration

Complexity of manufacturing scheduling models has already shortly been addressed in Chap. 2. It can result from many different aspects and may be classified into real-world related aspects (including organisational ones) and formal aspects. Complexity reduction with respect to the solution procedure is discussed in Chap. 7. Here, we will focus on complexity reduction of the scheduling model itself.

There are several brute force approaches to reduce a scheduling model's complexity:

- The problem as a whole is simplified with respect to the number of possible solutions. For example, a flow shop problem is considered as a permutation flow shop problem.
- Only 'relevant' jobs and/or machines are considered, i.e. only a subset of all jobs and/or machines is included in the model. Important jobs/machines remain in the

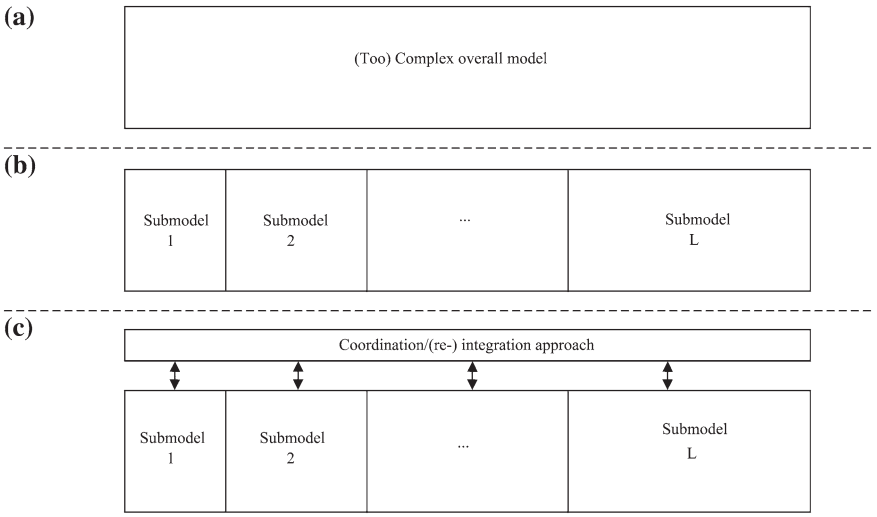


Fig. 6.4 Coordination approaches in complexity-reduced manufacturing scheduling problems: **a** Too complex overall model, **b** Model decomposition, **c** Decomposed model and coordination/integration

model, less important ones are added subsequently after a solution for the important items has been generated.

- Some constraints are relaxed or even ignored, e.g. preemption is excluded from consideration, sequence-dependent setup times are interpreted to be sequence-independent (and are therefore included in the processing times), transmission to a next machine is only possible for the whole job (although some sublot could also be transferred earlier to the next machine allowing overlapping processing, . . .).
- The time horizon is cut off earlier which usually results in less operations/jobs to be considered.
- . . .

All these and also more sophisticated approaches intend to reduce the number of scheduling objects (operations, jobs), resources (machines), variables and/or constraints to receive a model complexity which can be handled.

For a given manufacturing scheduling problem, suppose that we have either implicitly an impression of a too complex model or an explicit model which cannot be handled (solved) as is. To reduce this model's complexity, decomposition and subsequent coordination of the decomposed components of the model is required (see Fig. 6.4). The basic idea of decomposition is to separate the overall model into sub-models which are

1. tractable individually per sub-model (e.g. one machine problems or low number of jobs), and
2. connected with each other by advantageously few and clearly defined relations between the sub-models (to allow the subsequent integration of the partial solution to a solution of the overall problem).

(Decomposition might take place in several steps, on several levels and even iteratively. However, here we only describe aspects of a one-step decomposition approach.)

In most *resource-based* decomposition approaches, the overall scheduling problem is split into several sub-problems where these sub-problems contain each a subset of all machines. The subsets of machines are usually disjunct and in many cases consist only of a single machine or of the machines of a single production stage. Each sub-problem involves developing a schedule for the particular machine subset. By clustering resources/machines, from a formal point of view, resource-based decomposition approaches will primarily partition constraints referring to the respective resources/machines.

In *operation-based* decomposition approaches, an orthogonal view to the resource-oriented one is adopted. A complete shop schedule is interpreted as the integration of several job schedules. If n jobs are to be scheduled, e.g. (up to) n sub-problems are created where each involves the development of a schedule for a particular job (job group). Afterwards, these single job (job group) schedules are integrated to an overall schedule. By clustering operations in this type of decomposition approach, both constraints and variables are partitioned into the respective sub-problems.

In *event-based* decomposition approaches, each sub-problem involves making a single scheduling decision, e.g. which job/operation to schedule on a particular machine, or deciding to send a job to which machine. The schedule is developed, e.g. by an event-based simulation of the shop, as used in dispatching approaches to scheduling. In event-based decomposition, information is chronologically grouped. All information available at the time an event takes place is provided in a sub-problem.

These three basic decomposition strategies may be used isolated or in combination. The degree of sub-problem interdependencies also depends on the scheduling objective and the tightness of constraints in the specific problem instance. Therefore, bottlenecks should be anticipated before a decision for a specific decomposition approach is taken. If particular machines or jobs or time-periods/events are known to be crucial or expected to be crucial for the problem under consideration and its solution, then a focus should be laid on the respective aspects when decomposing the problem. From both, empirical evidence in more general cases as well as formal analysis for smaller problem sizes, it is well-known that the bottleneck type (machine, operations, events) indicates the direction of decomposition while in addition, first the bottlenecks themselves should not be clumped together with non-bottlenecks and second the respective other dimensions of decomposition/clustering approaches could be applied as well. For example, a problem with a clear bottleneck machine could separate this bottleneck machine from the other machines. The respective one machine problem for this bottleneck machine should be solved in detail while the

other machines might be grouped together (and possibly aggregated) in the decomposition approach as well as all or some jobs on these non-bottleneck machines.

When constructing the sub-models, these models might include some anticipating information from the other sub-models as well—if this is possible, e.g. as used in the well-known shifting bottleneck procedure, machine by machine sub-models might include job availability data (i.e. machine-specific release dates) resulting from information provided by the other sub-models. Also the classical coordination mechanisms as (primal) budgeting (e.g. of time) and/or (dual) price mechanisms might be included in the coordination schemes. The whole tool set of centralised or decentralised coordination might be applied to manufacturing scheduling models in one way or another. It should also be mentioned that the structure of the sub-models might be induced by the organisational structure of the manufacturing setting under consideration.

After the overall model has been decomposed into sub-models according to the above description, every single sub-model has to be solved. We suppose that these sub-models are solvable in one way or another without any severe problems—that is the main reason why they have been built and how they should be constructed. Finally, the solutions of the sub-models have to be (re-) integrated using an integration or coordination approach. This step cannot be described comprehensively. Basically, the local information from the sub-models' solutions will usually consist of some (local) schedule. Therefore, the individual schedules have to be integrated on the coordination level according to the requirements of the overall (original) model. If the resulting 'total' solution does not meet the decision-maker's requirements, the process of decomposition, local solution and (re-) integration of local solutions might be repeated subject to modified information delivered from the coordination process to the sub-models. For example, this information could consist of modified release times of jobs, different assignment of jointly used resources or different prices for these resources, etc.

6.4 Aggregation and Disaggregation

Aggregation and disaggregation is a well-known tool for reducing complexity in planning approaches and planning models. It can also be used for manufacturing scheduling models according to the two-level approach presented in Sect. 6.2. Aggregation in manufacturing scheduling is intended to reduce the number of objects to be considered in a respective model.

As mentioned earlier, the basic object of consideration in scheduling models is the single operation of a job on a machine. Therefore, aggregation of operations is a nearby approach to reduce the number of objects to be scheduled. Similar operations (e.g. requiring the same setup status) can be fixed as consecutive operations and therefore be added up to one 'macro' operation. Similar machines (and the respective operations of jobs on these machines), may they be in parallel on one production stage or in sequence, can be interpreted as one machine. (This approach is usually only

adequate for non-bottleneck machines.) Of course, as in every aggregation approach, the aggregates coarsen the model and induce some loss of information which has to be taken explicitly into account when the solutions for this complexity-reduced models are re-transferred to a solution of the original problem.

On one hand, the basic guideline for designing aggregation approaches is the more or less trivial declaration that no 'important' or 'relevant' information should be aggregated in a way that it will significantly influence the quality of the solution derived. However, on the other hand, there is (almost) no situation where there is no loss of information by using aggregation approaches. Therefore, a (negative) influence on the solution quality will always exist when using aggregation approaches. This influence may not be ignored but might be taken into account when designing an aggregation approach, e.g. increasing granularity of the aggregate model will usually increase the solution's quality but has to be traded-off with the increasing complexity of the resulting model.

Vancza et al (2004), although focusing on the integration of production planning on one hand and manufacturing scheduling on the other, give some more detailed hints which should be regarded when anticipating detailed scheduling models in some aggregate production planning counterpart which can also be adopted to complexity reduction in a pure scheduling context when using aggregation approaches:

1. Aggregate models must respect the main temporal constraints of jobs (e.g. due dates) as well as resource capacity constraints.
2. The aggregate model should also handle precedence relations that result from complex product structures (e.g. assemblies) and technological routings. However, resource assignment problems with finite capacities and precedence constraints are in general extremely hard to solve.
3. Albeit aggregation removes details—due to the differences between the planning and scheduling model—it may introduce new constraints that distort the original disaggregate problem. The effect of such constraints should be kept as small as possible.
4. Planning and scheduling models should be built by using common product and production data (e.g. bills of materials (BOMs), routings and resource calendars). Open orders should be addressed at both levels.

Concluding this paragraph, we list some standard subjects of aggregation and disaggregation in manufacturing scheduling:

1. An operation usually belongs to a job and a job might belong to a larger customer order or consists itself of several customer orders. Therefore, the aggregation of jobs to orders or vice versa and the respective influence on the respective operations' structure includes an aggregation/disaggregation approach. This aggregation/disaggregation is sometimes not seen as part of the scheduling model itself but is seen as external input to the scheduling model. However, clustering or decomposing jobs and/or orders accordingly might have a more or less severe influence on the quality of a schedule.

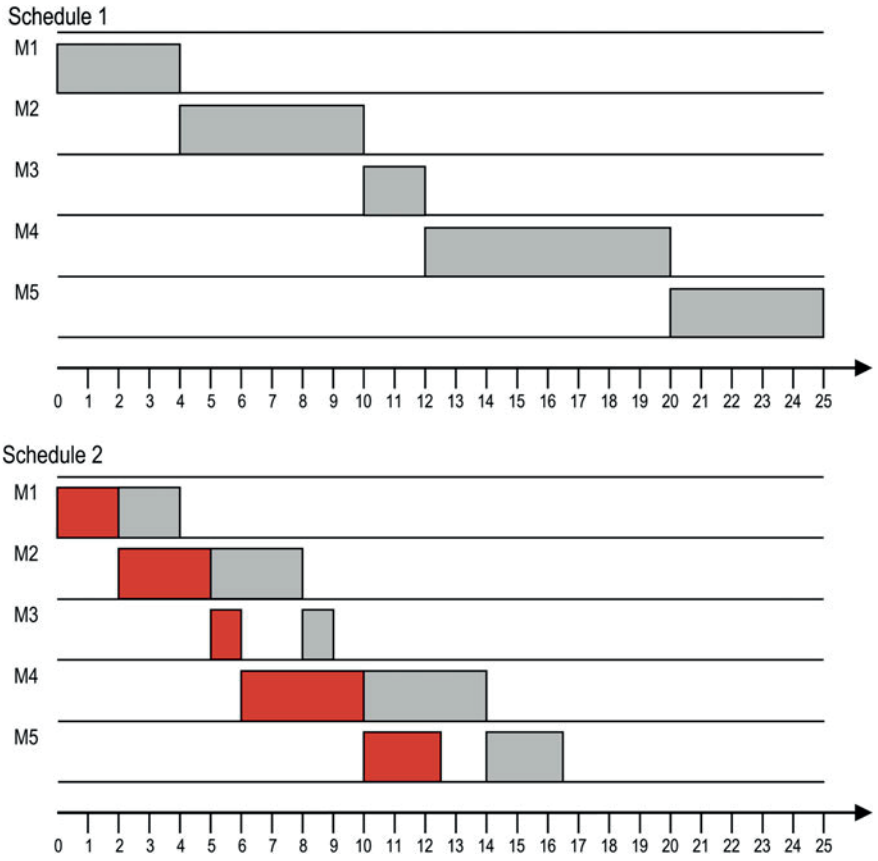


Fig. 6.5 Effect of lot streaming on makespan (cycle time)

2. Parallel and/or sequential batching (pooling) of operations (or whole jobs) on one machine might be decided outside of the scheduling problem (on an 'aggregate' decision level) or a subject of the scheduling model itself. Such questions occur, e.g. in process industries, if setups are relevant (with respect to time and/or costs, for product family scheduling problems), if ovens should not run below a certain utilisation rate (parallel batching), . . .
3. Opposite to batching, disaggregation of operations (lot streaming or lot splitting) might become an issue, e.g. if due dates are tight and customers are satisfied with partial shipments or if a next operation of a job is intended to start before the previous operation is finished for the complete job. Lot streaming was introduced in Chap. 3, Sect. 3.2.2 and further detailed in Sect. 3.4. Figure 6.5 shows the effect of lot streaming for a small example.

4. Transportation times from one stage to another might be added to the processing times of the respective operations if a separate consideration of transport operations is not required.
5. Considering every single machine of the shop structure in detail might be rather complex on one hand. On the other, organisational issues might imply an aggregate view on machines in manufacturing scheduling, e.g. if group technologies are applied. There the (macro) scheduling problem consists of assigning (macro) operations/tasks to pre-determined groups of machines while the detailed scheduling task within each group is decentralised and executed by the group itself.
6. Scheduling settings with hybrid structures on specific stages might be simplified by aggregating the capacities of a hybrid stage into one macro-machine and by adjusting the operations' processing times accordingly. Of course, disaggregating a solution from the aggregate macro-model needs to regard the hybrid structure explicitly.
7. Additional resources (apart from the machines, e.g. flexible workers) which are limited and have to be assigned to different machines competing for these resources might be considered in a detailed or atomic manner on one hand. On the other, if these resources can be pooled and can be partitioned more flexible, the solution space will be enlarged and better schedules might be generated, e.g. by improvements of balancing of production rates, of idle time reduction etc. This type of resource aggregation also includes the determination of the number of resource groups and the assignment of resource groups to machine (groups) as well as an adequate expression for the aggregate capacity itself (Burdett and Kozan 2003; Le Pape 1994).
8. Aggregation of time periods, i.e. reducing the granularity of the time grid might reduce the complexity of a scheduling model. Although ceteris paribus the number of operations remains unchanged by time aggregation, dealing with weeks or days instead of hours and minutes usually reduces the differences between the objective function values for different schedules and probably increases the number of good schedules since the differences of the objective function are reduced by this aggregation process. However, this must be in line with the requirements of the real-world scheduling problem, e.g. a schedule with a time horizon of 1 month usually will not require an hour time grid while a day's schedule will.
9. Connected with 8, the length of the time horizon itself might be an issue for complexity reduction although it is not a 'true' aggregation approach.

The above-listed approaches may be used isolated or in combination. Anyway, in most models they have to be designed individually and, as mentioned before, because of their coarsening of information, they will generate some feasibility and/or optimality lag which has to be evaluated with respect to the detailed model or, better, with respect to the real-world problem under consideration.

6.5 Validation and Verification of Modelling Approaches/Models

Model verification and validation are essential components of every modelling process. They are crucial for the acceptance of a model. Roughly spoken and according to modelling approaches to manufacturing scheduling in this chapter, the difference between verification and validation can be described as follows: While model verification tries to proof whether a model formulation (and, in addition, its solution procedure) fulfils the requirements of the model within the model's sphere, model validation is intended to check the model's adequacy with respect to the underlying real-world problem.

Model verification is intended to confirm (as far as possible) *ex ante* that the model and jointly its solution procedure performs as required. This includes that the (formal) model is constructed and mapped correctly and the algorithms are implemented properly. However, we can almost never guarantee a 100 % error-free (formal) mapping and implementation of any scheduling model. Only statistical statements on model's reliability are possible. Consequently, a properly designed model testing approach is a necessity for assessing model adequacy.

Complementarily to model verification, model validation intends to ensure that the model meets its intended requirements in terms of the methods employed and the results obtained relative to the underlying real-world scheduling setting. The goal of model validation is to make the model useful in the sense that the model addresses the right problem and to provide accurate information about the (real-world) system being modelled. Model validation includes the following components: requirements validation, face validity, calibration, process validation, data validation, theory validation, veridical (= the degree to which an experience, perception or interpretation accurately presents reality) and validation of emergent structures and processes.

It is often too costly and time consuming to validate a model comprehensively over its complete domain of applicability. Instead, tests and evaluations are conducted until sufficient confidence is obtained that a model can be considered valid for its intended application. The trade-off between confidence requirements and validation costs has to be considered case specific.

With respect to the complexity reduction approaches described in earlier sections of this chapter, a more general perception of model validation should be shortly addressed: Model validation can be interpreted also as validating a model's adequacy with respect to another model. This other model must not necessarily represent the real-world setting. It might also be the basic (too) complex model. Then, the complexity-reduced formal model is validated relative to the underlying complex model (which is supposed to be a formal model too). That is, in terms of validation, complexity reduction apart from real-world adequacy validation includes a second validation aspect, namely between the two different formal models, the complex one and the one with reduced complexity.

More operational, model validation and model verification contain one or more of the following aspects:

- The restrictions (constraints) of the model must be feasible, redundancy-free and in line with the real-world problem.
- The objective function(s) must be in line with the objectives for the real-world problem.
- The decision space in the model must be in line with the real-world decisions.
- The formulated model must be error-free.
- The subjective influence of the decision-maker(s) must be minimised (according to a purely rational point of view) or in line with the real-world structural conditions, e.g. when giving weights to job, operations and/or objectives.

Summarising, model verification and model validation are two steps being applicable to all modelling approaches where a real-world problem is mapped into one or more formal models or model levels. With respect to manufacturing scheduling, both approaches have to be specified according to the specific setting under consideration.

6.6 Conclusions and Further Readings

In this chapter, we described and discussed several typical approaches to construct scheduling models. We also addressed complexity reduction techniques in manufacturing scheduling, also including model decomposition and, inseparably connected, re-integration of information from decomposed sub-models. Aggregation and disaggregation as a prominent method for complexity reduction was discussed as well. Finally, we sketched out basic aspects of model verification and validation.

It is beyond the scope of this chapter to address the aspects mentioned before and/or to formalise the approaches in more detail. The interested reader has been referred to more specialised references in each section. The main bottom lines of this chapter are:

- Manufacturing scheduling models can be formalised in different ways with different tools. Decision-makers have to choose the adequate approach(es) carefully and according to the real-world requirements.
- Complexity reduction is a main and permanent issue in modelling scheduling problems.
- There is a voluminous tool box to address complexity reduction issues, including decomposition and re-integration as well as aggregation and disaggregation.
- Issues of complexity reduction should undergo a valid analysis instead of being treated on a gut level.
- Specific validation and verification for manufacturing scheduling models are missing to a large extent. Nevertheless, both inner-model verification and real-world oriented validation should be addressed adequately and explicitly for every scheduling model.

Regarding further readings, a key text on mathematical modelling is Williams (2013). Basic information on modelling approaches including uncertainty can be found in Zadeh (1975, 1999), Dubois et al (1996, 2003), Dubois and Prade (1988), Fayad and Petrovic (2005), Kuroda and Wang (1996), Vlach (2000). The first five references are specifically devoted to fuzzy approaches. An overview of previous work on using Petri nets for scheduling problems is given by Zhou and Venkatesh (2000), while different methods for generating object-oriented models can be found in Schmidt (1996). Finally, for further information on model validation and verification, the interested reader is referred to Macal (2005), Gibson (2001), Sargent (2007).

References

Błażewicz, J., Ecker, K. H., Pesch, E., Schmidt, G., and Weglarz, J. (2007). *Handbook on scheduling: from theory to applications*. Springer, Berlin/Heidelberg/New York.

Bowman, E. H. (1959). The schedule-sequencing problem. *Operations Research*, 7(5):621–624.

Burdett, R. L. and Kozan, E. (2003). Resource aggregation issues and effects in mixed model assembly. In Kozan, E., Beard, R., and Chattopadhyay, G., editors, *Proceedings 5th Operations Research Conference of the Australian Society for Operations Research Queensland Branch on Operations research into the 21st century*, pages 35–53, Brisbane, Queensland - Australia. Queensland University of Technology.

Dantzig, G. B. (1960). A machine-job scheduling model. *Managemant Science*, 6(2):191–196.

Dubois, D., Fargier, H., and Fortemps, P. (2003). Fuzzy scheduling: Modelling flexible constraints vs. coping with incomplete knowledge. *European Journal of Operational Research*, 147(2):231–252.

Dubois, D., Fargier, H., and Prade, H. (1996). Possibility theory in constraint satisfaction problems: Handling priority, preference and uncertainty. *Applied Intelligence*, 6(4):287–310.

Dubois, D. and Prade, H. (1988). *Possibility theory*. Plenum Press, New York.

Fayad, C. and Petrovic, S. (2005). A fuzzy genetic algorithm for real-world job shop scheduling. *Innovations in Applied Artificial Intelligence: 18th Int. Conf. on Industrial and Engineering Application of Artificial Intelligence and Expert Systems, IEA/AIE 2005. Bari, Italy, 22–24 June 2005*, 3533:524–533.

Gibson, J. P. (2001). Formal requirements models: simulation, validation and verification. *Technical, Report NUIM-CS-2001-TR-02*.

Greenberg, H. H. (1968). A branch-bound solution to the general scheduling problem. *Operations Research*, 16(2):353–361.

Kuroda, M. and Wang, Z. (1996). Fuzzy job shop scheduling. *International Journal of Production Economics*, 44(1–2):45–51.

Le Pape, C. (1994). Implementation of resource constraints in ILOG schedule: A library for the development of constraint-based scheduling systems. *Intelligent Systems Engineering*, 3(2):55–66.

Macal, C. M. (2005). Model verification and validation. *Workshop on "Threat Anticipation: Social Science Methods and Models"*.

Manne, A. S. (1960). On the job-shop scheduling problem. *Operations Research*, 8(2):219–223.

Nagar, A., Heragu, S. S., and Haddock, J. (1995). A branch-and-bound approach for a two-machine flowshop scheduling problem. *Journal of the Operational Research Society*, 46(6):721–734.

Pinedo, M. L. (2012). *Scheduling: Theory, Algorithms, and Systems*. Springer, New York, fourth edition.

Sargent, R. G. (2007). Verification and validation of simulation models. In Henderson, S. G., Biller, B., Hsieh, M.-H., Shortle, J., Tew, J. D., and Barton, R. R., editors, *Proceedings of the 2007 Winter Simulation Conference*, pages 124–137, Piscataway, NJ. IEEE Operations Center.

Schmidt, G. (1992). A decision support system for production scheduling. *Revue des Systemes de decision*, 1(2–3):243–260.

Schmidt, G. (1996). Modelling production scheduling systems. *International Journal of Production Economics*, 46–47:109–118.

Shen, W. and Norrie, D. (1998). An agent-based approach for dynamic manufacturing scheduling. *Working Notes of the Agent-Based Manufacturing Workshop, Minneapolis, MN*.

Slowinski, R. and Hapke, M. (2000). Foreword. In Slowinski, R. and Hapke, M., editors, *Scheduling under Fuzziness*, Heidelberg, New York. Physica-Verlag.

Vancza, J., Kis, T., and Kovacs, A. (2004). Aggregation: the key to integrating production planning and scheduling. *CIRP Annals of Manufacturing Technology*, 3(1):377–380.

Vlach, M. (2000). Single machine scheduling under fuzziness. In Slowinski, R. and Hapke, M., editors, *Scheduling under Fuzziness*, pages 223–245, Heidelberg, New York. Physica-Verlag.

Wagner, H. M. (1959). An integer linear-programming model for machine scheduling. *Naval Research Logistics Quarterly*, 6(2):131–140.

Williams, H. P. (2013). *Model building in mathematical programming*. Wiley, Chichester, 5th ed.

Zadeh, L. A. (1975). Calculus of fuzzy restrictions. In Zadeh, L. A., Fu, K. S., Tanaka, K., and Shimura, M., editors, *Fuzzy sets and their applications cognitive and decision processes*, pages 1–39, New York. Academic Press.

Zadeh, L. A. (1999). Fuzzy sets as a basis for a theory of possibility. *Fuzzy sets and systems*, 100:9–34.

Zhou, M. C. and Venkatesh, K. (2000). *Modelling, simulation and control of flexible manufacturing systems: A petri net approach*. World Scientific, Singapore a.o.