

Chapter 1

Scheduling Models

The so-called resource-constrained project scheduling problem (RCPSP) is one of the basic complex scheduling problems. In Section 1.1 we introduce this problem and some of its generalizations. Machine scheduling problems, which may be considered as special cases of the RCPSP, are described in Section 1.2. Several applications of the RCPSP are discussed in Section 1.3. Finally, some reference notes can be found in Section 1.4.

1.1 The RCPSP and some Generalizations

The resource-constrained project scheduling problem (RCPSP) is a very general scheduling problem which may be used to model many applications in practice (e.g. a production process, a software project, a school timetable, the construction of a house or the renovation of an airport). The objective is to schedule some activities over time such that scarce resource capacities are respected and a certain objective function is optimized. Examples for resources may be machines, people, rooms, money or energy, which are only available with limited capacities. As objective functions, for example, the project duration, the deviation from deadlines, or costs concerning resources may be minimized.

The **resource-constrained project scheduling problem (RCPSP)** may be formulated as follows. Given are a time horizon $[0, T]$, n **activities** (jobs) $i = 1, \dots, n$ and r **renewable resources** $k = 1, \dots, r$. A constant amount of R_k units of resource k is available at any time $t = 0, \dots, T$. Activity i must be processed for p_i time units. During this time period a constant amount of r_{ik} units of resource k is occupied. All data are assumed to be integers.

Furthermore, **precedence constraints** are defined between some activities. They are given by a set A of relations $i \rightarrow j$, where $i \rightarrow j$ means that activity j cannot start before activity i is completed.

Usually, we assume that activities are not preempted, i.e. if activity i starts at time S_i , it completes at time $C_i = S_i + p_i$. We may relax this condition by allowing **preemption** (activity splitting). In this case the processing of any

activity may be interrupted and resumed later. It will be stated explicitly if we consider models with preemption.

The objective is to determine starting times $S_i \in \{0, 1, \dots, T\}$ for the activities $i = 1, \dots, n$ in such a way that

- at each time the total resource demand is less than or equal to the resource availability R_k of each resource $k = 1, \dots, r$,
- the given precedence constraints are fulfilled, i.e. $S_i + p_i \leq S_j$ if $i \rightarrow j$, and
- the **makespan** $C_{\max} = \max_{i=1}^n \{C_i\}$ is minimized, where $C_i := S_i + p_i$ is the completion time of activity i .

The vector $S = (S_i)_{i=1}^n$ defines a **schedule** of the project. S is called **feasible** if all resource and precedence constraints are fulfilled. If the time horizon T is large enough, the precedence constraints are acyclic, and $r_{ik} \leq R_k$ for all $i = 1, \dots, n$ and $k = 1, \dots, r$ holds, always a feasible schedule exists: the activities may be processed in a topological ordering (i.e. an ordering which is compatible with the precedence constraints) consecutively one after the other, which leads to a schedule with $C_{\max} = \sum_{i=1}^n p_i$.

It is often useful to add a unique **dummy starting activity** 0 and a unique **dummy termination activity** $n + 1$, which indicate the start and the end of the project, respectively. The dummy activities need no resources and have processing time zero. In order to impose $0 \rightarrow i \rightarrow n + 1$ for all activities $i = 1, \dots, n$ we set $0 \rightarrow i$ for all activities i without any predecessor and $i \rightarrow n + 1$ for all activities i without any successor. Then S_0 is the starting time of the project and S_{n+1} may be interpreted as the makespan of the project. Usually we set $S_0 := 0$.

We may represent the structure of a project by a so-called **activity-on-node network** $G = (V, A)$, where the vertex set $V := \{0, 1, \dots, n, n + 1\}$ contains all activities and the set of arcs $A = \{(i, j) \mid i, j \in V; i \rightarrow j\}$ represents the precedence constraints. Each vertex $i \in V$ is weighted with the corresponding processing time p_i .

For each activity i we define

$$Pred(i) := \{j \mid (j, i) \in A\} \text{ and } Succ(i) := \{j \mid (i, j) \in A\}$$

as the sets of **predecessors** and **successors** of activity i , respectively.

Another representation of projects is based on so-called **activity-on-arc networks** where each activity is modeled by an arc. Since this representation has some disadvantages, in most cases activity-on-node networks are preferred.

Example 1.1: Consider a project with $n = 4$ activities, $r = 2$ resources with capacities $R_1 = 5, R_2 = 7$, a precedence relation $2 \rightarrow 3$ and the following data:

i	1	2	3	4
p_i	4	3	5	8
r_{i1}	2	1	2	2
r_{i2}	3	5	2	4

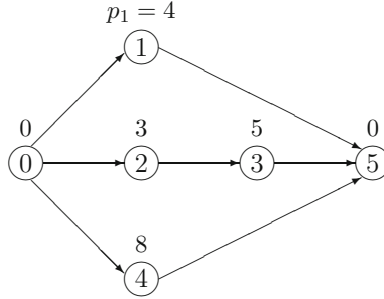


Figure 1.1: The activity-on-node network for Example 1.1

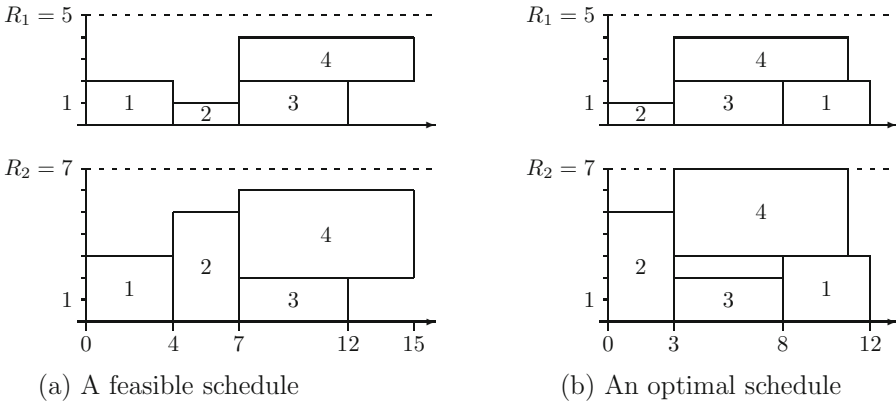


Figure 1.2: Two feasible schedules for Example 1.1

Figure 1.1 illustrates the corresponding activity-on-node network, where the dummy activities 0 and 5 have been added and the vertices are weighted with the processing times. In Figure 1.2(a) a so-called **Gantt chart** of a feasible schedule with $C_{\max} = 15$ is drawn. This schedule does not minimize the makespan, since by moving activity 1 to the right, a shorter schedule is obtained. An optimal schedule with makespan $C_{\max} = 12$ is shown in (b). \square

In the following we will discuss different generalizations of the RCPSP.

Generalized precedence constraints

A precedence relation $i \rightarrow j$ with the meaning $S_i + p_i \leq S_j$ may be generalized by a **start-start relation** of the form

$$S_i + d_{ij} \leq S_j \quad (1.1)$$

with an arbitrary integer number $d_{ij} \in \mathbb{Z}$. The interpretation of relation (1.1) depends on the sign of d_{ij} :

- If $d_{ij} \geq 0$, we must have $S_j \geq S_i + d_{ij}$, i.e. activity j cannot start before d_{ij} time units after the start of activity i . This means that activity j does not start before activity i and d_{ij} is a minimal distance (time-lag) between both starting times (cf. Figure 1.3(a)).
- If $d_{ij} < 0$, we must have $S_i \leq S_j + |d_{ij}|$, i.e. activity i cannot start more than $|d_{ij}|$ time units later than activity j . On the other hand, this implies that the earliest start of activity j is $|d_{ij}|$ time units before the start of activity i . If $S_j \leq S_i$, this means that $|d_{ij}|$ is a maximal distance between both starting times (cf. Figure 1.3(b)).



Figure 1.3: Positive and negative time-lags

If $d_{ij} > 0$ holds, the value is also called a **positive time-lag**; if $d_{ij} < 0$, it is called a **negative time-lag**. Time-lags d_{ij} may be incorporated into the activity-on-node network G by adding all arcs $i \rightarrow j$ to G and weighting them with the distances d_{ij} .

Relations (1.1) are very general timing relations between activities. For example, (1.1) with $d_{ij} = p_i$ is equivalent to the precedence relation $i \rightarrow j$. More generally, besides start-start relations also finish-start, finish-finish or start-finish relations may be considered. But if no preemption is allowed, any type of these relations can be transformed to any other type. For example, finish-start relations $C_i + l_{ij} \leq S_j$ with finish-start time-lags l_{ij} can be transformed into start-start relations $S_i + p_i + l_{ij} \leq S_j$ by setting $d_{ij} := p_i + l_{ij}$.

Generalized precedence relations may, for example, be used in order to model certain timing restrictions for a chemical process. If $S_i + p_i + l_{ij} \leq S_j$ and $S_j - u_{ij} - p_i \leq S_i$ with $0 \leq l_{ij} \leq u_{ij}$ is required, then the time between the completion time of activity i and the starting time of activity j must be at least l_{ij} but no more than u_{ij} . This may be modeled by the minimal time-lag

$d_{ij} = p_i + l_{ij}$ and the maximal time-lag $d_{ji} = -(p_i + u_{ij})$. If, for example, an activity j must start at least 4 time units and at most 7 time units after the completion of activity i , we have to fulfill $S_i + p_i + 4 \leq S_j$ and $S_j \leq S_i + p_i + 7$, which can be modeled by the minimal time-lag $d_{ij} = p_i + 4$ and the maximal time-lag $d_{ji} = -(p_i + 7)$.

In the special case $0 \leq l_{ij} = u_{ij}$ activity j must start exactly l_{ij} time units after the completion of activity i . If $l_{ij} = u_{ij} = 0$, then activity j must start immediately after activity i finishes (**no-wait constraint**).

Also release times r_i and deadlines d_i of activities i can be modeled by relations of the form (1.1). While a **release time** r_i is an earliest starting time for activity i , a **deadline** d_i is a latest completion time for i . To model release times we add the restrictions $S_0 + r_i \leq S_i$, i.e. we have minimal time-lags $d_{0i} = r_i$. To model deadlines we add the restrictions $S_i - (d_i - p_i) \leq S_0$, i.e. we have maximal time-lags $d_{i0} = -(d_i - p_i)$. In both cases we assume that $S_0 = 0$. If $r_i \leq d_i$ for a release time r_i and a deadline d_i , then the interval $[r_i, d_i]$ is called a **time window** for activity i . Activity i must be processed completely within its time window.

Sometimes, also so-called **feeding precedence constraints** are given which mean that some overlapping in the execution of two activities is allowed. For example, if activity j may already be started after 30% of activity i have been completed, the inequality $S_j \geq S_i + f_{ij} \cdot p_i$ with $f_{ij} = 0.3$ has to be satisfied.

Time-dependent resource profiles

A time period t is defined as the time interval $[t-1, t[$ for $t = 1, 2, \dots, T$, where T denotes a given time limit for the project. Until now we assumed that R_k units of resource k are available in each time period and that r_{ik} units of resource k are occupied in each time period t in which activity i is processed. This assumption may be generalized using the concept of **time-dependent resource profiles**. In this situation the availability $R_k(t)$ of resource k is a function depending on the time periods t . Especially, $R_k(t) = 0$ states that resource k is not available in time period t . The resource availability may be represented by pairs (t_k^μ, R_k^μ) for $\mu = 1, \dots, m_k$, where $0 = t_k^1 < t_k^2 < \dots < t_k^{m_k} = T$ are the jump points of the function and R_k^μ denotes the resource capacity in the time interval $[t_k^\mu, t_k^{\mu+1}[$ for $\mu = 1, \dots, m_k - 1$.

If time-dependent resource profiles are given, it is much more difficult to find a feasible schedule in the interval $[0, T]$ respecting all resource constraints.

Example 1.2: Consider an instance of a project with $n = 6$ activities, $r = 2$ resources, precedence constraints $1 \rightarrow 4 \rightarrow 5$, $2 \rightarrow 3$ and the following data:

i	1	2	3	4	5	6
p_i	2	2	3	2	2	4
r_{i1}	1	0	1	0	1	0
r_{i2}	1	2	3	2	4	2

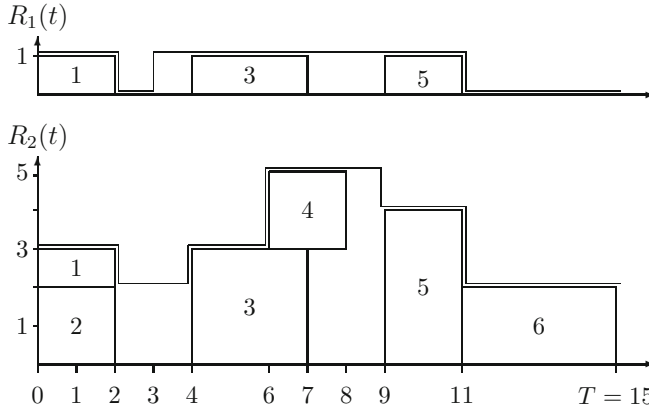


Figure 1.4: A feasible schedule for time-dependent resource profiles

Let the time-dependent resource profile of resource 1 be defined by the pairs $(0, 1)$, $(2, 0)$, $(3, 1)$, $(11, 0)$, $(15, 0)$, and the resource profile of resource 2 by $(0, 3)$, $(2, 2)$, $(4, 3)$, $(6, 5)$, $(9, 4)$, $(11, 2)$, $(15, 0)$.

In [Figure 1.4](#) a feasible schedule is represented by the corresponding Gantt chart. This schedule does not minimize the makespan because by moving activity 3 two units to the right and scheduling activity 6 between activity 1 and activity 3 a feasible schedule with smaller makespan is obtained. \square

A resource is called **disjunctive** if $R_k(t) \in \{0, 1\}$ for $t = 1, \dots, T$ holds, otherwise it is called **cumulative**. If resource k is disjunctive, two activities i, j with $r_{ik} = r_{jk} = 1$ can never be processed simultaneously.

Time-dependent resource profiles may, for example, be used in order to model a situation in which the numbers of available workers vary over time (e.g., at weekends less people are working). Sometimes restricted preemptions may be allowed in connection with time-dependent resource profiles. If due to an unavailability of a resource an activity cannot finish its processing, it may be suspended at the beginning of the unavailability period and continued when the resource is available again.

Multiple modes

In the **multi-mode** situation a set \mathcal{M}_i of so-called **modes** (processing alternatives) is associated with each activity i . The processing time of activity i in mode m is given by p_{im} and the per period usage of a renewable resource k is given by r_{ikm} . One has to assign a mode to each activity and to schedule the activities in the assigned modes.

Multiple modes may, for example, be used in order to model a situation in which an activity can be quickly processed by many workers or more slowly with less people.

Sometimes, it is required that a subset of activities must be processed in the same mode (so-called **mode identity** constraints).

A special case of the multi-mode RCPSP is the **discrete time/resource trade-off problem** (DTRTP) where only one single renewable resource (e.g., modeling available staff) with capacity R_1 is given. For each activity i a work content W_i is specified (e.g., the amount of person-days required to process i). From this work content W_i a finite set of feasible modes \mathcal{M}_i for activity i is derived where each mode $m \in \mathcal{M}_i$ is characterized by a processing time $p_{im} \in \mathbb{N}$ (e.g., indicating the number of days needed) and a resource requirement $r_{im} \in \mathbb{N}$ with $r_{im} \leq R_1$ (e.g., required persons per day). Feasible modes m have to satisfy $r_{im} \cdot p_{im} \geq W_i$ in order to fulfill the required work content. Furthermore, a mode $m \in \mathcal{M}_i$ is called efficient if no other feasible mode $m' \in \mathcal{M}_i$ exists with $r_{im'} < r_{im}$ and $p_{im'} \leq p_{im}$ or $r_{im'} = r_{im}$ and $p_{im'} < p_{im}$. It is easy to see that it is sufficient to consider only efficient modes.

Let us consider an example with $R_1 = 10$. Then, for an activity i with work content $W_i = 12$ the following modes (where for each processing time the smallest feasible resource requirements are chosen) are feasible:

m	1	2	3	4	5	6	7	8	9	10	11
p_{im}	2	3	4	5	6	7	8	9	10	11	12
r_{im}	6	4	3	3	2	2	2	2	2	2	1
eff.	×	×	×	—	×	—	—	—	—	—	×

Since mode 4 has a larger processing time than mode 3 (and the same resource requirement), it is not efficient. Similarly, modes 6 to 10 are worse than the efficient mode 5. Thus, the set \mathcal{M}_i of efficient modes contains only the five modes $\mathcal{M}_i = \{1, 2, 3, 5, 12\}$.

Non-renewable resources

Besides renewable resources like machines or people we may also have so-called **non-renewable resources** like money or energy. While renewable resources are available with a constant amount in each time period again, the availability of non-renewable resources is limited for the whole time horizon of the project. This means that non-renewable resources are consumed, i.e. when an activity i is processed, the available amount R_k of a non-renewable resource k is decreased by r_{ik} .

Non-renewable resources are only important in connection with multi-mode problems, because in the single-mode case the resource requirements of non-renewable resources are schedule independent (the available non-renewable resources may be sufficient or not). On the other hand, in the multi-mode case

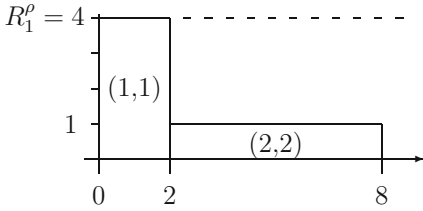
the resource requirements of non-renewable resources depend on the choice of modes (i.e. feasible and infeasible mode assignments may exist).

Besides upper bound values R_k for the total usage of a non-renewable resource in some applications also lower bound values L_k may have to be respected. For example, a worker may not only have a maximal but also a minimal working time per month. In such a situation only mode assignments satisfying both bounds are feasible.

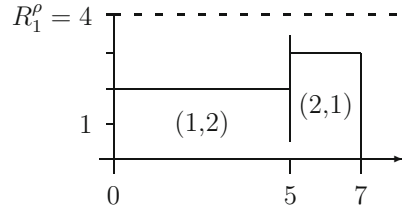
To distinguish between renewable and non-renewable resources, if both are needed, we write \mathcal{K}^ρ for the set of renewable resources, \mathcal{K}^ν for the set of non-renewable resources, r_{ikm}^ρ , R_k^ρ for the renewable resource data and r_{ikm}^ν , R_k^ν for the non-renewable resource data.

Example 1.3: Consider an instance of a multi-mode project with $n = 2$ activities, where each activity i can be processed in two different modes $m = 1, 2$. We have one renewable resource 1 with capacity $R_1^\rho = 4$ and one non-renewable resource 2 with capacity $R_2^\nu = 10$. Furthermore, the processing times p_{im} of the activities $i = 1, 2$ in modes $m = 1, 2$ are given by $p_{11} = 2$, $p_{12} = 5$, $p_{21} = 2$, $p_{22} = 6$ and the resource requirements r_{ikm} are

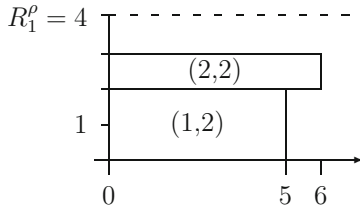
$$\begin{aligned} r_{111}^\rho = 4, \quad r_{112}^\rho = 2, \quad r_{121}^\nu = 6, \quad r_{122}^\nu = 2 & \quad \text{for activity 1, and} \\ r_{211}^\rho = 3, \quad r_{212}^\rho = 1, \quad r_{221}^\nu = 5, \quad r_{222}^\nu = 3 & \quad \text{for activity 2.} \end{aligned}$$



(a) Schedule with $C_{\max} = 8$



(b) Schedule with $C_{\max} = 7$



(c) Optimal schedule with $C_{\max} = 6$

Figure 1.5: Three feasible schedules for Example 1.3

Due to $r_{121}^\nu + r_{221}^\nu = 6 + 5 = 11 > 10 = R_2^\nu$ the non-renewable resource 2 does not allow that activities 1 and 2 are both processed in mode one. Furthermore, we

have $r_{121}^\nu + r_{222}^\nu = 6 + 3 = 9$, $r_{122}^\nu + r_{221}^\nu = 2 + 5 = 7$, and $r_{122}^\nu + r_{222}^\nu = 2 + 3 = 5$. Thus, the remaining three activity-mode combinations for the two activities are feasible: (1, 1) with (2, 2), (1, 2) with (2, 1) and (1, 2) with (2, 2). If we compare the corresponding schedules shown in Figure 1.5, we see that the schedule in (c) has the smallest makespan $C_{\max} = 6$. In this schedule 2+3=5 units of the non-renewable resource 2 are needed. \square

Doubly-constrained resources

A combination of renewable and non-renewable resources are so-called **doubly-constrained resources** which are constrained in each time period and for the whole project. With this type of resources, for example, a situation can be modeled in which the working time of a worker per day and the working time for the whole project is limited. Since a doubly-constrained resource can be treated by introducing a renewable and a non-renewable resource for it, this type of resources does not have to be considered separately.

Partially renewable resources

A generalization of renewable and non-renewable resources are so-called **partially renewable resources** \mathcal{K}^π for which the availability is restricted over subsets of time periods from the set $\{1, \dots, T\}$ with a given time horizon T . Associated with each partially renewable resource $k \in \mathcal{K}^\pi$ are (not necessarily disjoint) subsets of time periods $P_k(1), \dots, P_k(u_k) \subseteq \{1, \dots, T\}$, where for the time periods in the subset $P_k(\tau)$ ($\tau = 1, \dots, u_k$) in total $R_k^\pi(\tau)$ units of resource k are available. In all other time periods which are not contained in the subsets the resource capacity is assumed to be unlimited.

Furthermore, for each activity i its per-period requirement r_{ik}^π of resource $k \in \mathcal{K}^\pi$ is given. If activity i is (partially) processed in $P_k(\tau)$, it consumes r_{ik}^π units of resource k in each time period of $P_k(\tau)$ in which it is processed.

Example 1.4: Consider an instance with $n = 3$ activities, $r = 2$ partially renewable resources and time horizon $T = 10$. Associated with the resource $k = 1$ are the subsets $P_1(1) = \{1, 2, 3, 4, 5\}$ with $R_1^\pi(1) = 6$ and $P_1(2) = \{6, 7, 8, 9, 10\}$ with $R_1^\pi(2) = 8$. Associated with the resource $k = 2$ is the subset $P_2(1) = \{3, 4, 8, 9\}$ with $R_2^\pi(1) = 1$, in the remaining time periods resource 2 is not constrained. Furthermore, the activities have the processing times $p_1 = 3, p_2 = 3, p_3 = 1$ and the per-period resource requirements r_{ik}^π are given by

$$r_{11}^\pi = 1, r_{21}^\pi = 2, r_{31}^\pi = 3, \quad r_{12}^\pi = r_{22}^\pi = r_{32}^\pi = 1.$$

A feasible schedule for this instance is shown in Figure 1.6. In this schedule activity 1 needs 3 units of resource 1 in the subset $P_1(1)$ and 1 unit of resource 2 in $P_2(1)$. Activity 2 needs 2 units of resource 1 in $P_1(1)$, $2 \cdot 2 = 4$ units of

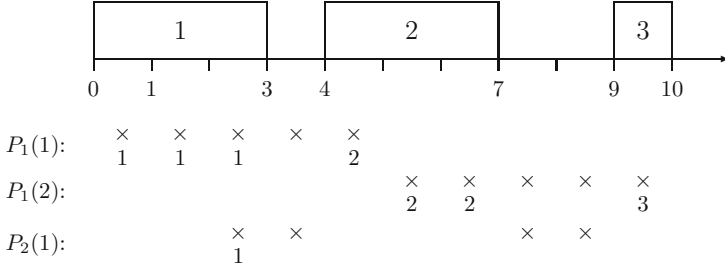


Figure 1.6: Feasible schedule for Example 1.4

resource 1 in $P_1(2)$, and activity 3 needs 3 units of resource 1 in $P_1(2)$. Activity 2 cannot start in time period $4 \in P_2(1)$ since resource 2 is already completely consumed in time period $3 \in P_2(1)$ by activity 1. Furthermore, activity 3 cannot be scheduled in a time period of the subset $P_1(1)$ since after scheduling activities 1 and 2 the remaining capacity 1 in $P_1(1)$ is less than $r_{31}^\pi = 3$. Note that it would also be feasible to process activity 3 in the time periods 6 or 7 in $P_1(2)$. \square

Often, the subsets $P_k(\tau)$ are intervals (like the sets $P_1(1), P_1(2)$ above). As an example consider a worker with a working time of 40 hours per week. This may be modeled by a partially renewable resource which is available for 40 hours in every week. On the other hand, we may also model a situation in which a worker is allowed to work on at most 5 weekend days per month. In this case a partially renewable resource with capacity 5 is introduced which is constrained over the subset of all weekend time periods.

Renewable resources \mathcal{K}^ρ may be considered as a special case of partially renewable resources by introducing for each resource $k \in \mathcal{K}^\rho$ and for $t = 1, \dots, T$ subsets $P_k(t) := \{t\}$ consisting of single time periods. Furthermore, the capacity is set to $R_k^\pi(t) := R_k^\rho$ (or $R_k(t)$ in the case of time-dependent resource profiles) and the per-period requirement of activity i is defined by $r_{ik}^\pi := r_{ik}^\rho$.

Also non-renewable resources \mathcal{K}^ν may be considered as a special case of partially renewable resources by introducing for each resource $k \in \mathcal{K}^\nu$ one subset $P_k(1) := \{1, \dots, T\}$ covering the complete time horizon. Furthermore, the capacity is set to $R_k^\pi(1) := R_k^\nu$ and the per-period requirement of activity i is defined by $r_{ik}^\pi := r_{ik}^\nu / p_i$ (recall that for non-renewable resources the resource requirements are given for the whole time horizon and not per period).

Storage resources

The availability of renewable resources like machines or people is independent of their previous utilization. For so-called **storage** resources (sometimes also called “cumulative” resources, not to be mixed up with the definition of disjunctive/cumulative resources on page 6) the resource availability at a certain time depends on the resource requirements of activities scheduled up to this time.

Such a resource can model the inventory level in some storage facility which is depleted and replenished over time.

For a storage resource k each activity i has resource demands $r_{ik}^+, r_{ik}^- \in \mathbb{N}$ where r_{ik}^+ is the amount of resource k that is replenished by activity i , and r_{ik}^- denotes the amount of resource k that is depleted by i . It is assumed that storage resources are depleted at start times and replenished at completion times of the activities. The value r_{0k}^+ of the dummy start activity 0 corresponds to the initial capacity of resource k . Associated with each storage resource k are a lower capacity (safety stock) $\underline{R}_k \geq 0$ and an upper capacity $\overline{R}_k \geq 0$. In a feasible schedule for all time periods the accumulated inventory level of every resource k must be contained in the interval $[\underline{R}_k, \overline{R}_k]$, i.e.

$$\underline{R}_k \leq \sum_{\{i|S_i+p_i \leq t\}} r_{ik}^+ - \sum_{\{i|S_i \leq t\}} r_{ik}^- \leq \overline{R}_k \quad \text{for all } t = 1, \dots, T.$$

Renewable resources \mathcal{K}^ρ may be considered as a special case of storage resources with $\underline{R}_k = 0, \overline{R}_k = R_k^\rho$ and $r_{0k}^+ = R_k^\rho$. While activity i depletes $r_{ik}^- = r_{ik}^\rho$ units of resource k at its start, it replenishes $r_{ik}^+ = r_{ik}^\rho$ at its completion. Also non-renewable resources \mathcal{K}^ν may be considered as a special case of storage resources with $\underline{R}_k = 0, \overline{R}_k = R_k^\nu$ and $r_{0k}^+ = R_k^\nu$. All activities i are depleting activities with $r_{ik}^- = r_{ik}^\nu$.

With the concept of storage resources we may model storage facilities (like tanks, containers or buffers), inventories of intermediate products within a production process, or investment capital.

Setup or transfer times

In a scheduling model with **setup times** (changeover times, transfer times) between the processing of activities resources are not available for a certain period since they have to be changed or transported. Setup times may, for example, be used to model changeover times of a machine which occur when the machine is changed for the production of a different product (e.g., if a painting machine has to be prepared for a different color). Another application are transfer times where resources have to be transported from one location to another before they can be used again.

Usually, setup times are assumed to be sequence-dependent, i.e. the time needed for changing the resource depends on the previous and the next activity. If s_{ij}^k denotes the setup time between activities i, j for resource k , we must satisfy $C_i + s_{ij}^k \leq S_j$ when activity j is directly processed after activity i on resource k .

Often one assumes that the setup times satisfy the strong triangle inequality

$$s_{ih}^k + s_{hj}^k \geq s_{ij}^k \quad \text{for all } i, j, h \in V, k = 1, \dots, r \quad (1.2)$$

or at least the weak triangle inequality

$$s_{ih}^k + p_h + s_{hj}^k \geq s_{ij}^k \quad \text{for all } i, j, h \in V, k = 1, \dots, r. \quad (1.3)$$

Other objective functions

Besides the objective of minimizing the makespan $C_{\max} := \max_{i=1}^n \{C_i\}$ one may consider other objective functions $f(C_1, \dots, C_n)$ depending on the completion times of the activities. Examples are the **total flow time** $\sum_{i=1}^n C_i$ or more generally the **weighted (total) flow time** $\sum_{i=1}^n w_i C_i$ with non-negative weights $w_i \geq 0$.

Other objective functions depend on **due dates** d_i which are associated with the activities. With the **lateness** $L_i := C_i - d_i$, the **tardiness** $T_i := \max\{0, C_i - d_i\}$, and the **unit penalty** $U_i := \begin{cases} 0, & \text{if } C_i \leq d_i \\ 1, & \text{otherwise} \end{cases}$ the following objective functions are common:

the maximum lateness	$L_{\max} := \max_{i=1}^n L_i$
the total tardiness	$\sum_{i=1}^n T_i$
the total weighted tardiness	$\sum_{i=1}^n w_i T_i$
the number of late activities	$\sum_{i=1}^n U_i$
the weighted number of late activities	$\sum_{i=1}^n w_i U_i.$

All these objective functions f are monotone non-decreasing in the completion times C_i , i.e. they satisfy $f(C_1, \dots, C_n) \leq f(C'_1, \dots, C'_n)$ for completion time vectors C, C' with $C_i \leq C'_i$ for all $i = 1, \dots, n$. They are also called **regular**. On the other hand, monotone non-increasing functions are called **antiregular**.

The **maximum earliness** $\max_{i=1}^n E_i$ with $E_i := \max\{0, d_i - C_i\}$ is an example for an antiregular objective function, the **weighted earliness-tardiness** $\sum_{i=1}^n w_i^E E_i + \sum_{i=1}^n w_i^T T_i$ with earliness weights $w_i^E \geq 0$ and tardiness weights $w_i^T \geq 0$ is neither regular nor antiregular. Also the objective function $\sum w_i C_i$ with arbitrary weights $w_i \in \mathbb{R}$ is nonregular. If $w_i > 0$, activity i should be completed as early as possible, if $w_i < 0$, activity i should be completed as late as possible.

Another nonregular objective function related to the last one deals with the so-called **net present value**. Associated with each activity i is a so-called cash-flow $c_i^F \in \mathbb{R}$ which is supposed to occur at the completion time C_i of i . It may be positive (i.e. a payment is received) or negative (i.e. a payment has to be given). The objective is to maximize the so-called net present value (NPV) $\sum_{i=1}^n c_i^F e^{-\alpha C_i}$, where $\alpha \geq 0$ is a given discount rate.

Besides these time-oriented objective functions also resource-based objective functions may be considered. They occur for example in the area of resource

investment and resource levelling problems. In the **resource investment problem** (RIP) the resource capacities R_k are not given, but have to be determined as additional decision variables. Providing one unit of resource k costs $c_k \geq 0$. The objective is to find a schedule with $C_{\max} \leq T$ for a given deadline T where the resource investment costs $\sum_{k=1}^r c_k R_k$ are minimized.

In **resource levelling problems** (RLP) the variation of the resource usage over time is measured. Let $c_k \geq 0$ be the cost for resource k and denote by $r_k^S(t)$ the resource usage of resource k in period $t \in \{1, \dots, T\}$ for a given schedule S , where $r_k^S(0) := 0$ is assumed. Besides the resource capacity R_k a target value $Y_k \geq 0$ for resource k is given.

In **resource deviation problems** the deviations (overloads) of the resource usages from a given resource profile are minimized. This can be done by minimizing

$$\begin{aligned} \text{the deviation} & \quad \sum_{k=1}^r c_k \sum_{t=1}^T |r_k^S(t) - Y_k|, \\ \text{the overload} & \quad \sum_{k=1}^r c_k \sum_{t=1}^T \max\{0, r_k^S(t) - Y_k\}, \text{ or} \\ \text{the squared deviation} & \quad \sum_{k=1}^r c_k \sum_{t=1}^T (r_k^S(t) - Y_k)^2. \end{aligned}$$

The value Y_k may also be replaced by the average resource usage

$$\bar{r}_k := \sum_{i=1}^n r_{ik} p_i / T.$$

On the other hand, in so-called **resource variation problems**, the resource usages should not vary much over time. This can be achieved by minimizing

$$\begin{aligned} & \sum_{k=1}^r c_k \sum_{t=1}^T |r_k^S(t) - r_k^S(t-1)|, \\ & \sum_{k=1}^r c_k \sum_{t=1}^T \max\{0, r_k^S(t) - r_k^S(t-1)\}, \text{ or} \\ & \sum_{k=1}^r c_k \sum_{t=1}^T (r_k^S(t) - r_k^S(t-1))^2. \end{aligned}$$

Finally, we note that an RCPSP with maximum lateness objective function can be reduced to an RCPSP with makespan objective function and negative time-lags. This follows from the fact that $L_{\max} = \max_{i=1}^n \{C_i - d_i\} = \max_{i=1}^n \{S_i + p_i - d_i\} \leq L$ for a threshold value L if and only if $S_i + p_i - d_i \leq L$ for all activities $i = 1, \dots, n$. By setting time-lags $d_{i,n+1} := -(d_i - p_i)$ for the dummy terminating activity $n+1$, the relations $S_i + p_i - d_i = S_i + d_{i,n+1} \leq S_{n+1}$ must be satisfied. Thus, by minimizing the makespan (which is equivalent to minimizing S_{n+1}) we minimize the maximum lateness L_{\max} .

1.2 Machine Scheduling

Important special cases of resource-constrained project scheduling problems are machine scheduling problems (where the resources correspond to machines). They will be discussed in more detail in this section.

1.2.1 Single-machine scheduling

In the simplest machine scheduling model we are given n jobs $j = 1, \dots, n$ with processing times p_j which have to be processed on a single machine. Additionally, precedence constraints may be given. Such a problem can be modeled by an RCPSP with one renewable disjunctive resource $r = 1$, capacity $R_1 = 1$ and resource requirements $r_{j1} = 1$ for all jobs $j = 1, \dots, n$.

Example 1.5: Consider an instance with $n = 5$ jobs, processing times $p_1 = 3$, $p_2 = 2$, $p_3 = 4$, $p_4 = 2$, $p_5 = 3$ and precedence constraints $1 \rightarrow 3$, $2 \rightarrow 4$, $4 \rightarrow 5$. A feasible schedule for this instance with makespan $C_{\max} = 14$ is shown in [Figure 1.7](#).

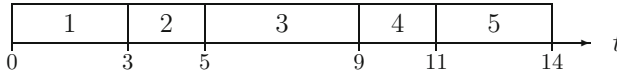


Figure 1.7: Single machine schedule

□

1.2.2 Parallel machine scheduling

Instead of a single machine we may have m machines M_1, \dots, M_m on which the jobs have to be processed. If the machines are **identical**, the processing time p_j of job j does not depend on the machine on which j is processed. This problem corresponds to an RCPSP with one cumulative resource where $R_1 = m$ and $r_{j1} = 1$ for $j = 1, \dots, n$.

Example 1.6: Consider an instance with $n = 8$ jobs, $m = 3$ machines, processing times $p_1 = 1$, $p_2 = 3$, $p_3 = 4$, $p_4 = 2$, $p_5 = 2$, $p_6 = 3$, $p_7 = 1$, $p_8 = 5$ and precedence constraints as shown in the left part of [Figure 1.8](#). A feasible schedule with makespan $C_{\max} = 9$ is shown in the right part of the figure. □

Contrary to identical machines, for so-called **unrelated** machines the processing time p_{jk} of job j depends on the machine M_k ($k = 1, \dots, m$) on which j is processed. The machines are called **uniform** if $p_{jk} = p_j/s_k$ where s_k may be interpreted as the speed of machine M_k . Problems with unrelated machines can be modeled as a multi-mode RCPSP with m renewable resources and $R_k = 1$ for $k = 1, \dots, m$. Each job j has m modes corresponding to the machines on which

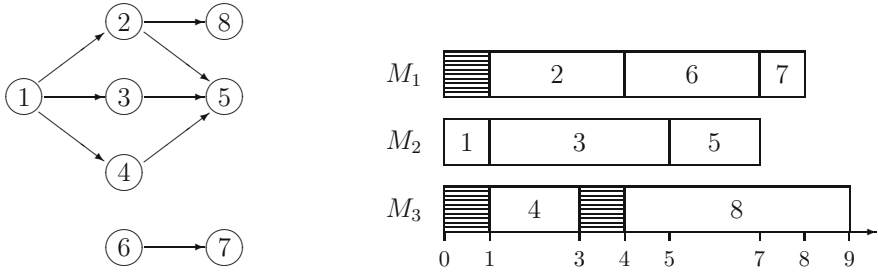


Figure 1.8: Schedule for identical parallel machines

j may be processed. If job j is processed in mode k , then p_{jk} is its processing time and j uses only one unit of resource k (machine M_k).

A further generalization are scheduling problems with **multi-purpose machines** (flexible machines). In this situation we associate with each job j a subset of machines $\mu_j \subseteq \{M_1, \dots, M_m\}$ indicating that j can be executed by any machine of this set. If job j is processed on machine M_k , then its processing time is equal to p_{jk} (or simply to p_j if the processing time does not depend on the assigned machine). This problem can be formulated as above as a multi-mode RCPSP with m renewable resources where each job j has $|\mu_j|$ modes corresponding to the machines on which j may be processed.

1.2.3 Shop scheduling

In so-called shop scheduling problems the jobs consist of several operations which have to be processed on different machines. In **general-shop scheduling problems** we have jobs $j = 1, \dots, n$ and m machines M_1, \dots, M_m . Job j consists of n_j operations $O_{1j}, \dots, O_{n_j, j}$. Two operations of the same job cannot be processed at the same time and a machine can process at most one operation at any time. Operation O_{ij} must be processed for p_{ij} time units on a dedicated machine $\mu_{ij} \in \{M_1, \dots, M_m\}$. Furthermore, precedence constraints may be given between arbitrary operations.

Such a general-shop scheduling problem can be modeled as an RCPSP with $r = m + n$ renewable resources and $R_k = 1$ for $k = 1, \dots, m + n$. While the resources $k = 1, \dots, m$ correspond to the machines, the resources $m + j$ ($j = 1, \dots, n$) are needed to model the fact that different operations of job j cannot be processed at the same time. Furthermore, we have $\sum_{j=1}^n n_j$ activities O_{ij} , where operation O_{ij} needs one unit of “machine resource” μ_{ij} and one unit of “job resource” $m + j$.

Important special cases of the general-shop scheduling problem are job-shop, flow-shop, and open-shop problems, which will be discussed next.

Job-shop problems

A **job-shop problem** is a general-shop scheduling problem with chain precedences of the form

$$O_{1j} \rightarrow O_{2j} \rightarrow \dots \rightarrow O_{n_j,j}$$

for $j = 1, \dots, n$ (i.e. there are no precedences between operations of different jobs and the precedences between operations of the same job build a chain). Note that for a job-shop problem no “job resource” is needed, since all operations of the same job are linked by a precedence relation (and thus cannot be processed simultaneously).

Flow-shop problems

A **flow-shop problem** is a special job-shop problem with $n_j = m$ operations for $j = 1, \dots, n$ and $\mu_{ij} = M_i$ for $i = 1, \dots, m$, $j = 1, \dots, n$, i.e. operation O_{ij} must be processed on M_i . In a so-called **permutation flow-shop problem** the jobs have to be processed in the same order on all machines.

Example 1.7: In Figure 1.9 a feasible schedule for a permutation flow-shop problem with $n = 4$ jobs and $m = 3$ machines is shown.

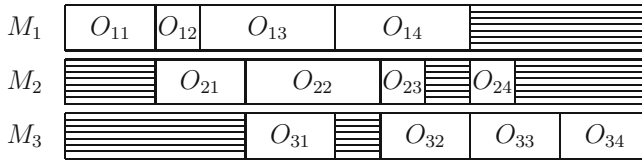


Figure 1.9: Permutation schedule for a flow-shop problem

□

Open-shop problems

An **open-shop problem** is like a flow-shop problem but without any precedence relations between the operations. Thus, it also has to be decided in which order the operations of a job are processed.

Example 1.8: In Figure 1.10 a feasible schedule for an open-shop problem with $n = 2$ jobs and $m = 3$ machines is shown. In this schedule the operations of job 1 are processed in the order (O_{11}, O_{31}, O_{21}) , the operations of job 2 are processed in the order (O_{32}, O_{22}, O_{12}) .

□

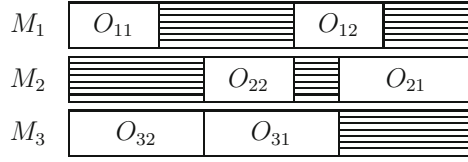


Figure 1.10: Feasible schedule for an open-shop problem

1.2.4 Multi-processor task scheduling

In a **multi-processor task scheduling problem** we have n jobs $j = 1, \dots, n$ and m machines M_1, \dots, M_m . Associated with each job j is a processing time p_j and a subset of machines $\mu_j \subseteq \{M_1, \dots, M_m\}$. During its processing job j occupies all machines in μ_j simultaneously. Furthermore, precedence constraints may be given between certain jobs.

This problem can be formulated as an RCPSP with $r = m$ renewable resources and $R_k = 1$ for $k = 1, \dots, r$. Furthermore,

$$r_{jk} = \begin{cases} 1, & \text{if } M_k \in \mu_j \\ 0, & \text{otherwise.} \end{cases}$$

Example 1.9: Consider the following instance with $n = 5$ jobs and $m = 3$ machines:

j	1	2	3	4	5
μ_j	$\{M_1, M_2\}$	$\{M_2, M_3\}$	$\{M_1, M_2\}$	$\{M_3\}$	$\{M_1, M_2, M_3\}$
p_j	1	2	2	3	1

In Figure 1.11 a feasible schedule with makespan $C_{\max} = 7$ for this instance is shown. It does not minimize the makespan since by processing job 1 together with job 4 we can get a schedule with $C_{\max} = 6$.

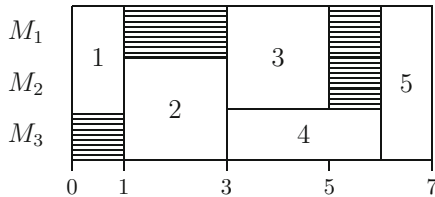


Figure 1.11: Feasible schedule for a multi-processor task problem

□

Multi-mode multi-processor task scheduling problems are a combination of problems with multi-processor tasks and multi-purpose machines. This means that with each job a set of different machine subsets (processing alternatives) is associated and a job needs all machines from a subset simultaneously.

1.3 Applications of the RCPSP

In this section we present several applications of the RCPSP from different areas.

Application 1.1: A cutting-stock problem

Materials such as paper, textiles, cellophane, etc. are manufactured in standard rolls of a large width W which is the same for all rolls. These rolls have to be cut into smaller rolls $i = 1, \dots, n$ with widths w_i such that the number of sliced rolls is minimized. In Figure 1.12 a solution of a cutting-stock problem with $n = 15$ smaller rolls is illustrated. In this solution 7 rolls have to be cut.

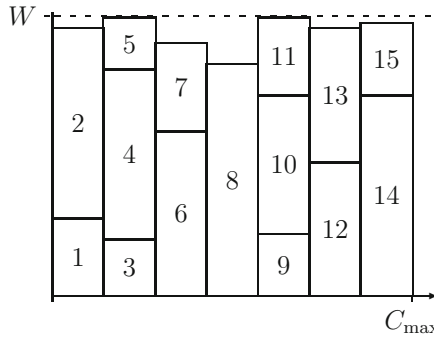


Figure 1.12: Solution of a cutting-stock problem

This problem can be formulated as an RCPSP with only one renewable resource where $R_1 = W$ units of the resource are available. The activities $i = 1, \dots, n$ correspond to the rolls to be cut. Activity i has processing time $p_i = 1$ and uses $r_{i1} = w_i$ units of this resource. The makespan corresponds to the number of standard rolls to be cut, i.e. a schedule with a minimal makespan corresponds to a solution with a minimal number of sliced standard rolls. \square

Application 1.2: Aircraft maintenance

After a certain flight duration airplanes or helicopters have to be inspected. Such an inspection consists of 300-400 tasks, each requiring 1-4 technicians for a certain duration. Furthermore, with each task a working area in the airplane is associated. Since in working areas like the cockpit or the cargo bay the physical space is very restricted, only a limited number of technicians can work in such an area simultaneously. Between certain tasks precedences may exist. The objective is to find a schedule such that

- all maintenance tasks are executed for their given durations,
- for all tasks sufficient technicians are available,
- for all tasks the physical space of all areas is respected,

- the precedences are respected, and
- the total duration is minimized.

This problem can be formulated as an RCPSP minimizing the makespan. The inspection tasks correspond to activities, and the technicians are modeled by a renewable resource whose capacity is equal to the number of available technicians. Each activity i has a given duration p_i and needs a specified number a_i of technicians. In order to model the space restrictions, for each working area an additional renewable resource is introduced whose capacity equals the maximal number of technicians who can work simultaneously in the area. Then each activity i additionally requires a_i units from its corresponding working area resource. The given precedences between certain tasks are modeled by corresponding precedence relations. \square

Application 1.3: Scheduling of ground handling operations

At an airport several activities have to be performed when an aircraft stays on the ground. Besides technical services (like fuelling, wheel and tire checks, ground power supply, cooling and heating, cleaning of cockpit windows, de-icing) passenger-related activities have to be executed (unloading and loading of baggage, disembarkment and embarkment of passengers, catering and cabin cleaning). Between some of these activities precedence relations exist (some of them also with timing restrictions in form of minimal or maximal time-lags). Furthermore, resources (like technical staff, ramp equipment, baggage cars) with limited capacities are needed. The objective is to find a feasible schedule such that the ground time of the aircrafts is as small as possible. This problem may be modeled as an RCPSP with generalized precedence constraints. \square

Application 1.4: Gate scheduling

Another optimization problem arising at an airport is the assignment of flights to terminal positions (gates). Given is a flight schedule containing for every flight its arrival time, the assigned aircraft and the next flight for this aircraft with a corresponding departure time. With each flight a subset of gates is associated to which the aircraft can be assigned (due to length restrictions not every assignment is feasible). An aircraft goes through two or three stages at the airport: arrival, optional intermediate parking, and departure. At any stage a gate must be assigned to the aircraft, different gates are possible, then the aircraft has to travel between them (so-called towing). The objective is to assign each flight to one or more feasible gates and to determine starting and completion times for these assignments such that no conflict between different aircrafts arise, the total flight-gate preference value (e.g., taking into account passenger walking distances) is maximized, and the number of required towing operations is as small as possible.

This problem may be formulated as multi-mode RCPSP where the gates correspond to resources and modes. For every flight three activities are introduced:

one arrival, one parking, and one departure activity. These three activities are linked by precedence relations ensuring that they are processed in this order (cf. Figure 1.13).

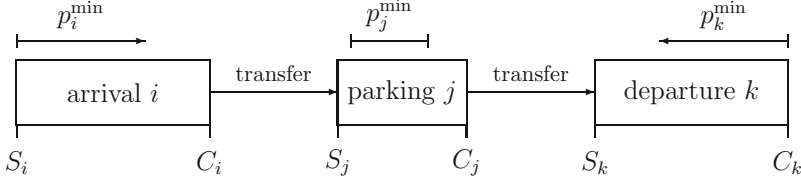


Figure 1.13: Three activities for every flight

- For an arrival activity i its starting time S_i is fixed to the arrival time of the corresponding flight. Contrary to the standard RCPSP no fixed processing time, but only a minimal processing time p_i^{\min} is given (modeling the minimum time required for passenger disembarkment, baggage unloading, etc.). The completion C_i is a decision variable and must satisfy $C_i \geq S_i + p_i^{\min}$.
- Symmetrically, for a departure activity k the completion time C_k is fixed to the departure time of the corresponding flight and a minimum processing time p_k^{\min} is given. The starting time S_k has to be determined such that $S_k \leq C_k - p_k^{\min}$ holds.
- For every flight between the arrival and departure activities a parking activity j is introduced. For this activity, both the starting time S_j and the completion time C_j have to be determined. Since during the stay at the airport an aircraft must always be assigned to some position, the starting time S_j has to be equal to C_i plus the transfer time needed for the towing operation between arrival and parking gate. Similarly, the completion time C_j has to be equal to S_k minus the transfer time needed for the towing operation between parking and departure gate. These two equality constraints can be modeled by minimal and maximal time-lags (with the same value). Finally, again a minimal processing time p_j^{\min} for parking is given, i.e. we must satisfy $C_j - S_j \geq p_j^{\min}$.

Every gate is modeled as a disjunctive resource to which only one aircraft can be assigned at any time. Additionally, between two successive assignments sequence dependent setup (transfer) times may have to be taken into account. Furthermore, for every activity a set of modes is introduced corresponding to all gates to which the aircraft can be assigned. The objective is to find modes as well as starting and completion times for all activities such that the timing and resource constraints are respected and the objective function is optimized.

□

Application 1.5: Batch scheduling in process industries

In process industries (e.g., chemical, pharmaceutical, or food industry) final products are built by several successive transformations of intermediate products. In this context operations correspond to chemical reactions in processing units like reactors, heaters or filters. An operation must be carried out on one out of several alternative processing units. Each operation needs several input products and produces different output products. Intermediate products may be perishable (i.e. have only a limited shelf-life and must be consumed after this period) or may be stocked in dedicated storage facilities like tanks or silos. Furthermore, cleaning times may have to be taken into account on processing units between different operations.

In a batch processing mode the intermediate and final products are partitioned into different batches which specified sizes (specifying the quantities of needed input products and produced output products). All input products have to be available at the start of an operation, the output is available after the completion of an operation.

After the sizes of the batches have been determined, in the scheduling problem

- the operations have to be allocated to processing units and storage facilities, and
- the operations have to be scheduled such that the capacities of all processing units and storage facilities are not exceeded and the total processing time is minimized.

This batch scheduling problem can be formulated as an RCPSP with renewable and storage resources, setup times, and makespan objective. Each processing unit type is modeled as a renewable resource whose capacity is equal to the number of available processing units belonging to this type. Each operation corresponds to an activity which needs a specified processing unit for a certain amount of time. Precedence constraints exist between operations which are linked due to the production structure (e.g., from input to intermediate products as well as from intermediate to output products). Every intermediate product corresponds to a storage resource k . If an operation i produces an intermediate product, r_{ik}^+ equals the number of units produced, if i consumes an intermediate product, r_{ik}^- equals the number of units consumed. The capacity \underline{R}_k is equal to the safety stock of the product and \overline{R}_k equals the maximal number of units that can be stored. For perishable products with zero shelf life time, we have $\underline{R}_k = \overline{R}_k = 0$. Finally, the cleaning times may be modeled as sequence-dependent setup times for the activities on the resources corresponding to the processing units.

In an extended model for each operation several processing alternatives may exist (differing in the required processing unit and the processing time needed). Such a situation can be modeled as a multi-mode RCPSP where each processing alternative corresponds to a mode. Furthermore, also minimal and maximal time-lags may be given between certain operations. \square

Application 1.6: High-school timetabling

In a basic high-school timetabling problem we are given m classes c_1, \dots, c_m , h teachers A_1, \dots, A_h and T teaching periods $t = 1, \dots, T$. Furthermore, we have lectures $i = 1, \dots, n$. Associated with each lecture i is a unique teacher $A_{\mu(i)}$ and a unique class $c_{\nu(i)}$. A teacher A_j may be available only in certain teaching periods. The corresponding timetabling problem is to assign the lectures to the teaching periods such that

- each class has at most one lecture in any time period,
- each teacher has at most one lecture in any time period, and
- each teacher has only to teach in time periods where he is available.

This problem may be formulated as an RCPSP with time-dependent resource profiles and n activities, where each activity corresponds to a lecture i given by teacher $A_{\mu(i)}$ for class $c_{\nu(i)}$.

Furthermore, we have $r := m + h$ resources $k = 1, \dots, m, m + 1, \dots, m + h$. The first m resources $1, \dots, m$ correspond to the classes, the last h resources $m + 1, \dots, m + h$ to the teachers A_1, \dots, A_h . We have $R_k = 1$ for $k = 1, \dots, m$. The availability of the resources $k = m + 1, \dots, m + h$ is time-dependent:

$$R_{m+j}(t) = \begin{cases} 1, & \text{if teacher } A_j \text{ is available in period } t \\ 0, & \text{otherwise.} \end{cases}$$

If activity i is a lecture for class c_l given by teacher A_j , then its resource requirement for resource k is

$$r_{ik} = \begin{cases} 1, & \text{if } k = l \text{ or } k = m + j \\ 0, & \text{otherwise.} \end{cases}$$

In a basic version of the problem one has to find a feasible schedule with $C_{\max} \leq T$. In practice, many additional constraints may have to be satisfied, e.g.

- for each class or teacher the number of teaching periods per day is bounded from above and below,
- certain lectures must be taught in special rooms,
- some pairs of lectures have to be scheduled simultaneously,
- the lectures of a class given by the same teacher should be spread uniformly over the week,
- classes or teachers should not have much idle periods on a day, etc.

□

Application 1.7: An audit-staff scheduling problem

A set of jobs J_1, \dots, J_g are to be processed by auditors A_1, \dots, A_m . Job J_l consists of n_l tasks ($l = 1, \dots, g$). There may be precedence constraints $i_1 \rightarrow i_2$ between tasks i_1, i_2 of the same job. Associated with each job J_l is a release time r_l , a due date d_l and a weight w_l .

Each task must be processed by exactly one auditor. If task i is processed by auditor A_k , then its processing time is p_{ik} . Auditor A_k is available during disjoint time intervals $[s_k^\nu, l_k^\nu]$ ($\nu = 1, \dots, m_k$) with $l_k^\nu \leq s_k^{\nu+1}$ for $\nu = 1, \dots, m_k - 1$. Furthermore, the total working time of A_k is bounded from below by H_k^- and from above by H_k^+ with $H_k^- \leq H_k^+$ ($k = 1, \dots, m$).

We have to find an assignment $\alpha(i)$ for each task $i = 1, \dots, n := \sum_{l=1}^g n_l$ to an auditor $A_{\alpha(i)}$ and to schedule the assigned tasks such that

- each task is processed without preemption in a time window of the assigned auditor,
- the total workload of A_k is bounded by H_k^- and H_k^+ for $k = 1, \dots, m$,
- the precedence constraints are satisfied,
- all tasks of J_l do not start before time r_l , and
- the total weighted tardiness $\sum_{l=1}^g w_l T_l$ is minimized.

Other features may be added to the model:

- Restricted preemption in the sense that if a task cannot be finished within one working interval of an auditor, it must be continued at the beginning of the next interval of the same auditor.
- Setup times and setup costs if an auditor moves from one job to another.
- Costs c_{ik} for assigning task i to auditor A_k . By setting $c_{ik} := \infty$ we may model that task i cannot be assigned to auditor A_k . In this situation the term $\sum_{i=1}^n c_{i\alpha(i)}$ may be added to the objective function.

Audit-staff scheduling problems may be modeled as multi-mode resource-constrained project scheduling problems with time-dependent resource profiles. For each auditor a doubly-constrained resource is introduced which on the one hand as a renewable resource is constrained by the availability profile of the auditor and, on the other hand as a non-renewable resource, by the working time bounds H_k^-, H_k^+ .

In [Figure 1.14](#) an example for an audit-staff scheduling problem with $g = 2$ jobs and $m = 3$ auditors is considered. We assume $r_1 = r_2 = 0$, $d_1 = 11$, $d_2 = 12$ and

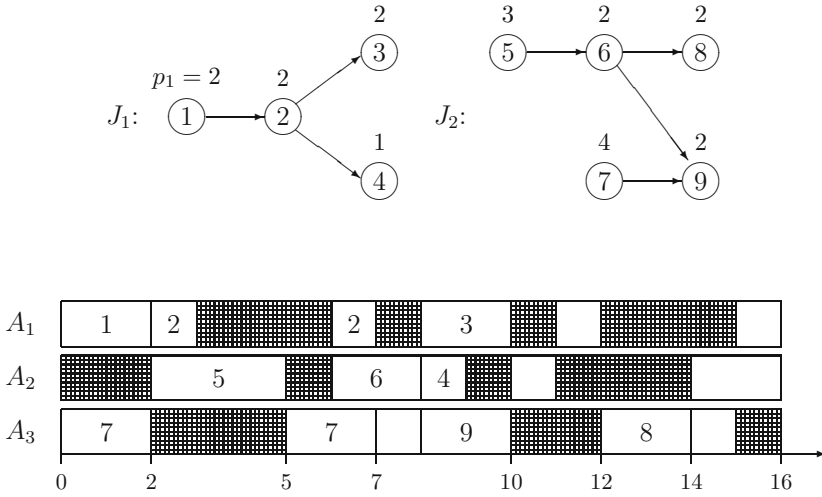


Figure 1.14: An audit staff schedule

$w_1 = 1, w_2 = 2$. Auditor A_1 is not available in the time intervals $[3, 6]$, $[7, 8]$, $[10, 11]$, $[12, 15]$, A_2 is not available in the intervals $[0, 2]$, $[5, 6]$, $[9, 10]$, $[11, 14]$, and A_3 is not available in $[2, 5]$, $[10, 12]$, $[15, 16]$. Additionally, we assume that restricted preemptions are allowed.

In Figure 1.14 also a feasible schedule is shown, where the shaded areas indicate the time periods during which the auditors are not available. The objective function value of this schedule is $w_1 T_1 + w_2 T_2 = 1 \cdot 0 + 2 \cdot (14 - 12) = 4$. \square

Application 1.8: Sports league scheduling

We consider a sports league consisting of $2n$ different teams $i = 1, \dots, 2n$ which play a single or double round robin tournament during a season. While in a single round robin tournament (SRRT) each team plays against each other team exactly once (either at home or away), in a double round robin tournament (DRRT) each team plays against each other team twice (once at home, once away). We consider so-called temporally-constrained tournaments in which the number of rounds (days for games) is equal to the minimal number of days required to schedule all games.

For a SRRT this means that $2n - 1$ rounds $t = 1, \dots, 2n - 1$ are given, in which the $\binom{2n}{2} = n(2n - 1)$ games are scheduled. Every team has to play exactly one game in each round, i.e. n games occur in each round. If no additional constraints have to be respected, a feasible SRRT schedule exists for every even number $2n$ of teams and can be constructed by using a graph model. An example for such a feasible SRRT schedule with $2n = 6$ teams and $2n - 1 = 5$ rounds can be found in Figure 1.15.

If the league consists of an odd number $2n + 1$ of teams, in each round one team has a ‘bye’, i.e. does not play. This situation may be reduced to the previous case with an even number of teams by adding a dummy team $2n + 2$. Then in each round the team playing against $2n + 2$ has a bye.

round 1	round 2	round 3	round 4	round 5
1-6	1-3	1-5	1-2	1-4
2-5	2-6	2-4	3-5	2-3
3-4	4-5	3-6	4-6	5-6

Figure 1.15: Feasible SRRT schedule with $2n = 6$ teams

For a DRRT we assume that the season is partitioned into two half series, where each half consists of $2n - 1$ rounds and each pairing has to occur exactly once in each half series. If the pairing $i - j$ is a home game for team i in the first half, it is a home game for team j in the second half. The second half series is usually not scheduled independently from the first. Often, the second series is planned complementarily to the first, i.e. the pairings of round $2n - 1 + t$ in the second half series are the same as in round t of the first half series for $t = 1, \dots, 2n - 1$ (with exchanged home teams).

An example for a feasible DRRT schedule with $2n = 6$ teams and $2(2n - 1) = 10$ rounds can be found in [Figure 1.16](#).

1	2	3	4	5	6	7	8	9	10
1-6	3-1	1-5	2-1	1-4	6-1	1-3	5-1	1-2	4-1
2-5	6-2	4-2	5-3	3-2	5-2	2-6	2-4	3-5	2-3
4-3	5-4	3-6	6-4	5-6	3-4	4-5	6-3	4-6	6-5

Figure 1.16: Feasible DRRT schedule with $2n = 6$ teams

When the season for a sports league is planned, in general many constraints have to be respected. In particular, organisational, attractiveness, and fairness constraints are important. Organisational constraints cover a set of rules which have to guarantee that all the games can be scheduled according to the regulations. Attractiveness constraints focus on what stadium visitors, television spectators and the players expect from the sequence of games (i.e. a varied, eventful, and exciting season). Finally, fairness constraints have to guarantee that no team is handicapped or favored in comparison with the competitors.

We assume that all requirements for the second half can be transformed into requirements for the first half series (e.g., if in a round of the second half the stadium for a team is unavailable, the team has to play at home in the corresponding round of the first half). Thus, it is sufficient to find a schedule for the first half series consisting of the rounds $t = 1, \dots, T := 2n - 1$ taking into account constraints for the second half series.

Such a problem may be formulated as a multi-mode RCPSP with time-dependent resource profiles as follows. Since we only plan one half series, we have $n(2n - 1)$ games which are represented by all pairs (i, j) with $i, j \in \{1, \dots, 2n\}$ and $i < j$. These pairs correspond to activities, which may be processed in two different modes (in mode H scheduling the game at the home stadium of team i , or in mode A scheduling the game at the home stadium of team j). All activities

have unit processing times in each mode. We introduce $2n$ team resources for $i = 1, \dots, 2n$ with constant capacity 1, ensuring that each team plays at most one game in each round. An activity corresponding to the game (i, j) needs one unit of the two team resources belonging to teams i and j .

In the following we describe additional constraints which can be formulated with the concepts of resources:

- **Stadium availabilities:** Due to some other events (e.g., concerts or games of other sports disciplines) some stadiums may not be available in certain rounds. We assume that each team i specifies for all rounds $t = 1, \dots, \hat{T} := 2(2n - 1)$ the values

$$S_{it} = \begin{cases} 1, & \text{if the stadium of } i \text{ is available in round } t \\ 0, & \text{otherwise.} \end{cases}$$

If $S_{it} = 0$ for some round t holds, team i cannot play a home game in round t , i.e. it has to play away.

To model such a situation we introduce $4n$ stadium resources **STADRES1(i)** (first half series) and **STADRES2(i)** (transformation of second half) for $i = 1, \dots, 2n$ with variable 0-1 resource profiles

$$\mathbf{STADRES1(i)}(t) = S_{it} \in \{0, 1\} \text{ und } \mathbf{STADRES2(i)}(t) = S_{i,t+2n-1} \in \{0, 1\}$$

for $t = 1, \dots, 2n - 1$ ensuring that the stadium availabilities $S_{i\tau}$ ($\tau = 1, \dots, \hat{T}$) are respected in the first and second half series.

An activity corresponding to the game (i, j) needs one unit of stadium resources **STADRES1(i)** and **STADRES2(j)** in mode H and one unit of **STADRES1(j)** and **STADRES2(i)** in mode A, respectively.

- **Regions:** If teams are located close to each other in a region, not all of them should play at home in a round simultaneously (since otherwise not sufficient police is available and the trains may be overcrowded). We assume that ρ regions $\mathcal{R}_1, \dots, \mathcal{R}_\rho$ are specified as subsets of teams which are located in the same region. In each round at most $\lceil \frac{|\mathcal{R}_r|}{2} \rceil$ teams from region \mathcal{R}_r may have a home game. For example, if a region contains 5 teams, we require that at most $\lceil \frac{5}{2} \rceil = 3$ of them have a home game in the same round. If, especially, two teams share the same stadium, a region containing these two teams may be used to model the situation that in no round the two teams can play at home simultaneously.

To model regions we introduce 2ρ region resources **REGRES1(r)** (first half series) and **REGRES2(r)** (transformation of second half) for regions $r = 1, \dots, \rho$ with constant capacity $\lceil \frac{|\mathcal{R}_r|}{2} \rceil$.

An activity corresponding to the game (i, j) needs one unit of region resources **REGRES1(reg[i])** and **REGRES2(reg[j])** in mode H and one unit of region resources **REGRES1(reg[j])** and **REGRES2(reg[i])** in mode A, respectively, where **reg[i]** denotes the region of team i .

- **Games with forbidden rounds:** Certain games may not be scheduled in certain rounds (e.g., games between two teams in the same region if in that region already another event takes place in this round, top games not at the beginning of the season, games between league climbers not at the end). We assume that for all such games (i, j) a subset $\hat{T}_{(i,j)} \subset \{1, \dots, T\}$ of forbidden rounds is given.

To model this case we introduce for each game (i, j) with forbidden rounds one resource $\text{FORBRES}(i, j)$ with variable 0-1 resource profile

$$\text{FORBRES}(i, j)(t) = \begin{cases} 0, & \text{if } t \in \hat{T}_{(i,j)} \\ 1, & \text{otherwise.} \end{cases}$$

Then an activity corresponding to a forbidden game (i, j) needs one unit of resource $\text{FORBRES}(i, j)$ in both modes.

- **Attractive games:** In order to distribute interesting games over the whole season, in each round the number of attractive games may not exceed a given number. We assume that all attractive games (e.g., top games or local derbies) are specified in a set AG and that a limit ag_{\max} for the number of attractive games per round is given. We introduce one resource AGRES with constant capacity ag_{\max} and each activity corresponding to an attractive game (i, j) needs one unit of resource AGRES in both modes.

- **Distribution of home and away games:** In each half series each team should play approximately half of its games at home, the other half away. Thus, at most $\lceil \frac{2n-1}{2} \rceil$ home games (away games) may be scheduled for each team in the first half. For this purpose we introduce $2n$ non-renewable home resources $\text{HOMERES}(i)$ and $2n$ non-renewable away resources $\text{AWAYRES}(i)$ for teams $i = 1, \dots, 2n$ with capacity $\lceil \frac{2n-1}{2} \rceil$.

An activity corresponding to the game (i, j) needs one unit of home resource $\text{HOMERES}(i)$ and one unit of away resource $\text{AWAYRES}(j)$ in mode H and one unit of $\text{HOMERES}(j)$ and $\text{AWAYRES}(i)$ in mode A, respectively.

Soft constraints may be:

- **Home/away preferences:** Besides the (hard) stadium unavailabilities S_{it} , teams may have preferences for home or away games in certain rounds. For example, during public festivals home games are preferred, while away games are preferred when another attractive event in the same region is already scheduled on a certain date (e.g., also games of other leagues).
- **Opponent strengths:** In order to distribute games against stronger and weaker opponents evenly over the season, for each team the opponents in two consecutive rounds should have different strengths.
- **Breaks:** Often it is not desired that teams play two home or two away games in two consecutive rounds.

Violations of all these constraints may be penalized in the objective function (e.g., by summing up penalties for all violations). \square

1.4 Reference Notes

In recent years a large number of papers has been published in the area of project scheduling. The most important results and corresponding references can be found in the books by Demeulemeester and Herroelen [58], Neumann et al. [153], Artigues et al. [7] and the handbooks [191] edited by Węglarz as well as [107] edited by Józefowska and Węglarz. Survey articles were written by Özdamar and Ulusoy [159], Herroelen et al. [100], Brucker et al. [28], Kolisch and Padman [120] and Węglarz et al. [192].

The concept of partially renewable resources was introduced by Böttcher et al. [23] and Schirmer and Drexel [173]. The discrete time/resource trade-off problem (DTRTP) was first studied by De Reyck et al. [59] and later also tackled in Demeulemeester et al. [55]. Storage resources are considered in Neumann and Schwindt [151], models for resource transfers in Krüger and Scholl [129]. The concept of mode identity is described in Salewski et al. [171]. In the literature several additional variants of resources have been introduced (e.g., so-called synchronizing resources or allocatable resources). Such variants and extensions of the RCPSP are summarized in Hartmann and Briskorn [93] as well as in Węglarz et al. [192].

For machine scheduling problems cf. the books of Błażewicz et al. [20], [21], Brucker [27], Pinedo [165], [166] and the handbook [135] edited by Leung.

Connections between packing problems and project scheduling models are considered in Hartmann [90]. In Brimberg et al. [26] aircraft maintenance problems with restricted working areas are studied. The handling of airport ground processes is discussed in Kuster and Jannach [131] as well as in Dorndorf [65]. In [65] also models and solution algorithms for gate scheduling can be found. Batch scheduling problems in process industries have been studied by Schwindt and Trautmann [175] and Neumann et al. [152].

A survey on timetabling problems is given in Schaerf [172], connections between timetabling and resource-constrained project scheduling problems are discussed in Brucker and Knust [35]. An RCPSP model for a school timetabling problem can be found in Drexel and Salewski [70], a course scheduling problem at Lufthansa was solved by Haase et al. [88]. Solution algorithms for audit-staff scheduling problems were developed in Dodin et al. [63] and Brucker and Schumacher [40]. Drexel and Knust [69] present graph- and resource-based models for sports league scheduling problems, a special problem of scheduling a table-tennis league is treated in Knust [116]. The annotated bibliography by Kendall et al. [109] provides a general overview on sports scheduling.

Additional applications of RCPSP models and solution methods can be found in Mellentien et al. [143] (scheduling the factory pick-up of new cars), Bomsdorf and Derigs [24] (movie shoot scheduling problem), Bartels and Zimmermann [14] (scheduling tests in automotive R&D projects), Lorenzoni et al. [136] (port operations).



<http://www.springer.com/978-3-642-23928-1>

Complex Scheduling

Brucker, P.; Knust, S.

2012, X, 342 p., Hardcover

ISBN: 978-3-642-23928-1