Chapter 8
Resource-Constrained Scheduling Extensions

Abstract Resource-constrained project scheduling has been a topic in both the research community and the practical oriented business magazines. This chapter presents some advanced results obtained by various research projects, extends the resource models of the previous chapter to other scheduling objectives, studies the effect of activity splitting and setup times and introduces learning effects in a resource-constrained project environment. Each part of this section can be considered as a special topic of resource-constrained project scheduling and can be easily skipped without losing overview on the general dynamic scheduling theme described throughout the book.

8.1 Introduction

The rich amount of research projects in resource-constrained project scheduling can be best illustrated by the classification schemes developed by Brucker et al. (1999) and Herroelen et al. (1999). Their intention was to classify all project scheduling related issues into a single scheme such that previous and future research efforts can be put into the right perspective, using a clear and unambiguous scheme. Both papers mention the use of the scheme as a dynamic instrument, indicating that new important issues related to project scheduling will pop up and need to get a place in the existing scheme. Up to today, the scheme is still widely used to classify new important aspects of project scheduling.

Although the scheme is not the topic of the current section, some parts of it will be outlined into more detail. This section has no intention whatsoever to give a complete overview of the state-of-the-art resource-constrained project scheduling research, but instead wants to highlight a number of resource-constrained scheduling related issues taken from literature that illustrate the importance of extending the basic project scheduling models to more realistic project settings.

M. Vanhoucke, *Project Management with Dynamic Scheduling*,
DOI 10.1007/978-3-642-25175-7_8, © Springer-Verlag Berlin Heidelberg 2012

The outline of this chapter can be summarized as follows. Section 8.2 presents three project scheduling objectives on top of the objectives presented in the previous chapter. While one extension is based on a problem from literature with a high relevance in practice, the two others are based on experience gained during consultancy projects. Section 8.3 briefly gives an overview of the literature regarding quantitative project descriptions to measure the structure of a project network and the scarceness of the resources used. Section 8.4 presents three extensions of project scheduling that have relevance in practice: the extension of the basic activity assumptions to more general settings, the use of setup times between parts of activities and the presence of learning when working with resources. Section 8.5 ends with conclusions and gives a brief sketch of future research needs.

8.2 Other Scheduling Objectives

8.2.1 Work Continuity Optimization

The previously discussed resource-constrained project scheduling techniques focused on resolving the resource conflicts by shifting activities in time. In doing so, these techniques guarantee that project activities do not use more renewable resources than available at each time period of the project life. However, these methods completely ignore the idle time of resources during the execution of the project as they do not try to schedule sets of activities that make use of a similar resource together. Though, there are numerous project examples where a subset of project activities (further referred to as an activity group) uses a common set of resources and where this set of resources is occupied from the first moment an activity from the group starts until the last activity. Therefore, so-called *work continuity constraints* (El-Rayes and Moselhi 1998) have been introduced in order to build a project schedule where the idle time of resources is minimized. A number of examples are given below:

- Spatial resources: A resource type that is not required by a single activity but rather by a group of activities. Examples are dry docks in a ship yard, shop floor space or pallets. Since the spatial resource unit is occupied from the first moment an activity from the group starts until the last activity of the group finishes, work continuity constraints can be of crucial importance (De Boer 1998).
- Time dependent cost resource: Time dependent costs (TDC) are the part of the project total cost that changes with the variation of activity times (Gong 1997). The TDC of a resource is the product of unit time cost and service time. Goto et al. (2000) have investigated the use of this concept and argue that the service time of a time dependent cost resource is the time duration starting from the first use and ending at the last. They refer to the use of a tower crane in the

construction industry and argue that the reduction of waiting times of TDC resources naturally reduces the time dependent cost.

- Repeating activities: Recognition of the drawbacks of traditional CPM network models in scheduling construction projects has led to the development of several alternative scheduling methodologies under different names, such as the Line of Balance (LOB) method or the Linear Scheduling Method (LSM). Harris and Ioannou (1998) give an excellent overview and integrate these methods into the so-called *repetitive scheduling method* (RSM), which is a practical scheduling methodology that ensures continuous resource utilization applicable to repetitive construction scheduling. A practical example of work continuity constraints in the repetitive construction industry will be discussed in Chap. 9.

Consequently, work continuity constraints are used to refer to the minimization of idle time of resources in a project. The introduction of work continuity constraints in the RCPSP leads to the *resource-constrained project scheduling problem with work continuity constraints* (RCPSPWC). This problem type involves the scheduling of project activities in order to minimize the total work continuity cost of the project subject to precedence relations and a predefined project deadline. The word 'constraint' is somewhat misleading and is only used to be in line with the terminology used in the literature. The resource idle time minimization is guaranteed by minimizing the total resource cost of the schedule that consists of the work continuity cost for each activity group g (which consists of the activity set $N^g \subset N$). This latter cost can be minimized by minimizing the time-span between the first and last activity of the activity group. Indeed, the resources are needed from the start of the first activity and will only be released at the completion of the last activity of the activity group. Consequently, the start times of all intermediate activities have no influence on the idle time of this resource and therefore do not influence the total work continuity cost. The following parameters are necessary to describe the scheduling problem formulation:

G	Set of activity groups, index g ($g = 1, \ldots, \|G\|$)
N^g	Set of activities of activity group g ($N^g \subset N$) (require a common set of resources (work continuity constraints))
SG^g	Earliest start of all activities of activity group g. (involves the start of the use of the resource)
FG^g	Earliest finish of all activities of activity group g (the resource can only be released after this time moment)
c_w^g	Work continuity cost of activity group g (cost per time unit for the set of resources of the activity group g)

A conceptual formulation for the RCPSPWC can be given as follows:

$$\text{Minimize} \sum_{g=1}^{|G|} c_w^g (FG^g - SG^g) \tag{8.1}$$

subject to

$$s_i + d_i \leq s_j \qquad \forall (i,j) \in A \tag{8.2}$$

$$\sum_{i \in S(t)} r_{ik} \leq a_k \qquad k = 1, \ldots, K; t = 1, \ldots, T \tag{8.3}$$

$$SG^g = \min_{i \in N^g} s_i \qquad g = 1, \ldots, |G| \tag{8.4}$$

$$FG^g = \max_{i \in N^g} (s_i + d_i) \qquad g = 1, \ldots, |G| \tag{8.5}$$

$$s_1 = 0 \tag{8.6}$$

$$s_n \leq \delta_n \tag{8.7}$$

$$s_i \in int^+ \qquad i = 1 \ldots, n \tag{8.8}$$

The objective in Eq. 8.1 minimizes the weighted time-span between the first and last activity of each activity group and hence, minimizes the total cost of work continuity. The constraint set given in Eq. 8.2 maintains the finish-start precedence relations among the activities. The limited availability of the renewable resources is modeled by Eq. 8.3. Equations 8.4 and 8.5 are introduced to model the start and finish time, respectively, for each activity group. These start and finish times determine the length of use of the resource and hence, the work continuity. Equation 8.6 forces the dummy start activity to start at time zero and Eq. 8.7 limits the project duration to a negotiated project deadline. Equations 8.8 ensure that the activity start times assume nonnegative integer values.

Figure 8.1 displays the optimal resource-feasible schedule for the RCPSPWC for the example project of Table 7.1. This project is subject to traditional renewable resources with a limited availability of ten units (see column resource use) as well as two resource groups (A and B). Some activities have been grouped to an activity group, since they rely on a common resource group (A or B) that is subject to work

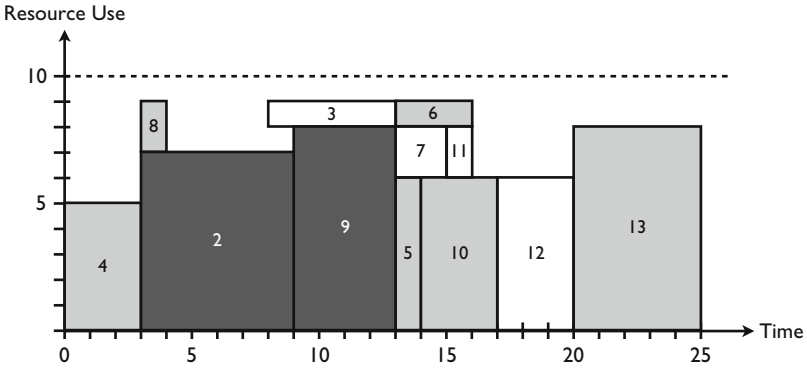


Fig. 8.1 Feasible resource graph with work continuity optimization

continuity constraints. More precisely, activities 2 and 9 have been grouped to an activity group since they use similar equipment tools of class A, while activities 3, 7, 11 and 12 need tools from group B during their execution, and hence, belong to a second activity group. The optimal resource graph with a minimal value for the resource idle time for resource groups A and B shows that activities 2 and 9 are grouped and activities 3, 7, 11 and 12 are grouped within the precedence and resource constraints. The time-span for the first activity group $N^1 = \{2, 9\}$ equals $\max(3 + 6, 9 + 4) - \min(3, 9) = 10$ while the time-span for the second activity group $N^2 = \{3, 7, 11, 12\}$ equals $\max(8 + 5, 13 + 2, 15 + 1, 17 + 3) - \min(8, 13, 15, 17) = 12$. Assume a cost of idle time of $c_w^1 = €10$ and $c_w^2 = €20$, then the total objective function cost equals $100 + 240 = €340$.

Work continuity constraints are particularly relevant in repetitive scheduling environments to minimize the idle time of various resources, which is discussed hereafter.

Repetitive Scheduling

Construction projects are often characterised by repeating activities that have to be performed from unit to unit. Highway projects, pipeline constructions and high-rise buildings, for example, commonly require resources to perform the work on similar activities that shift in stages. Indeed, construction crews perform the work in a sequence and move from one unit of the project to the next. This is mainly the result of the subdivision of a general activity (e.g. carpentry) into specific activities associated with particular units (e.g. carpentry at each floor of a high-rise building).

The repetitive processes of these construction projects can be classified according to the direction of successive work along the units. In *horizontal repetitive projects* the different processes are performed horizontally, as seen in pipeline construction or paving works. These construction projects are often referred to as continuous repetitive projects or linear projects due to the linear nature of the geometrical layout and work accomplishment. When progress is performed vertically, we refer to *vertical repetitive projects*, among which high-rise building construction is the classical example. Rather than a number of activities following each other linearly, these construction projects involve the repetition of a unit network throughout the project in discrete steps. It is therefore often referred to as discrete repetitive projects. Kang et al. (2001) argue that construction projects can consist of both horizontal and vertical repetitive processes among several multi-storey structures and refer to this type as *multiple repetitive projects*.

In the following two paragraphs, two fictitious project examples are used to illustrate the relevance of work continuity constraints in construction scheduling. The first example is taken from Harris and Ioannou (1998) to schedule a repetitive project to minimize crew idle time. In a second example, the complete trade-off between project duration and work continuity is illustrated. Chapter 9 describes a last example of a real-life project that aims at the construction of a tunnel at the Westerschelde in the Netherlands.

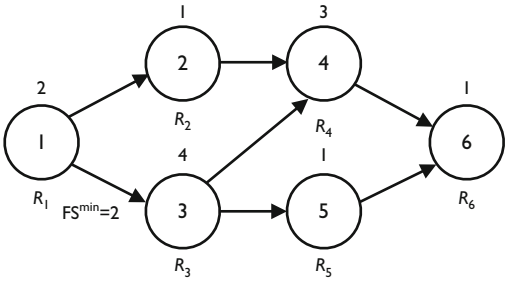


Fig. 8.2 An example project with six repeating activities (Source: Harris and Ioannou (1998))

Example 1 (A six-unit repetitive project). Figure 8.2 displays an illustrative activity-on-the-node project network for the activities in the first unit, as published in Harris and Ioannou (1998). Each of these six activities has a duration, denoted above the node and needs to use a certain resource R_i, denoted below the node. The solid arcs are technological precedence relations between the activities. The default value for each precedence relation is a finish-start relationship with a minimal time-lag of zero (i.e. $FS^{\min} = 0$), unless indicated otherwise. In this example, a minimal time-lag of two time units between activity 1 and 3 is specified (this is indicated as a 'lead time' in Harris and Ioannou (1998)). The construction of a project schedule with repeating networks can be done by the so-called *Repetitive Scheduling Method* (RSM).

Figure 8.3 displays the complete network for a project with six repeating units, each having the six discrete activities of Fig. 8.2. The dashed arcs link similar activities from unit to unit and are used to represent resource availability constraints.

Since it is assumed that the work to be done in units 3 and 4 for activity 1 is twice the work to be done in unit 1, the durations of activities 13 and 19 have been doubled. Moreover, a minimal time-lag of five is added between units 3 and 4 (i.e. $FS^{\min}_{14,20} = 5$). This planned interruption in resource continuity is to meet some known or predicted circumstance. Harris and Ioannou (1998) mention that the delivery of materials by a subcontractor's truck is sufficient to completing only three units, and consequently, a work break period is needed after unit 3. Remark that this repetitive project does not have an activity 3 in unit 5, which is a characteristic of an atypical project. To that purpose, the duration of activity 27 is set to zero, which is similar to the deletion of this activity from the project network.

The purpose is to construct a feasible project schedule with a minimal value for the resource idle time for all resources R_1 to R_6. The algorithm developed by Vanhoucke (2006b) reports an idle time value of 5 with a total project duration of 30. This project duration is in this case equal to the critical path length. The start times reported by the algorithm are equal to $s_0 = 0$, $s_1 = 0$, $s_2 = 6$, $s_3 = 4$, $s_4 = 11$, $s_5 = 19$, $s_6 = 24$, $s_7 = 2$, $s_8 = 7$, $s_9 = 8$, $s_{10} = 14$, $s_{11} = 20$, $s_{12} = 25$, $s_{13} = 4$, $s_{14} = 8$, $s_{15} = 12$, $s_{16} = 17$, $s_{17} = 21$, $s_{18} = 26$, $s_{19} = 8$, $s_{20} = 14$, $s_{21} = 16$, $s_{22} = 20$, $s_{23} = 22$, $s_{24} = 27$, $s_{25} = 12$, $s_{26} = 15$, $s_{27} = 20$, $s_{28} = 23$, $s_{29} = 23$, $s_{30} = 28$, $s_{31} = 14$, $s_{32} = 16$, $s_{33} = 20$, $s_{34} = 26$, $s_{35} = 24$, $s_{36} = 29$ and $s_{37} = 30$. In Fig. 8.4, an RSM diagram based on these start times is shown, which is similar to the diagram given by Harris and Ioannou (1998). The vertical axis shows the work to be

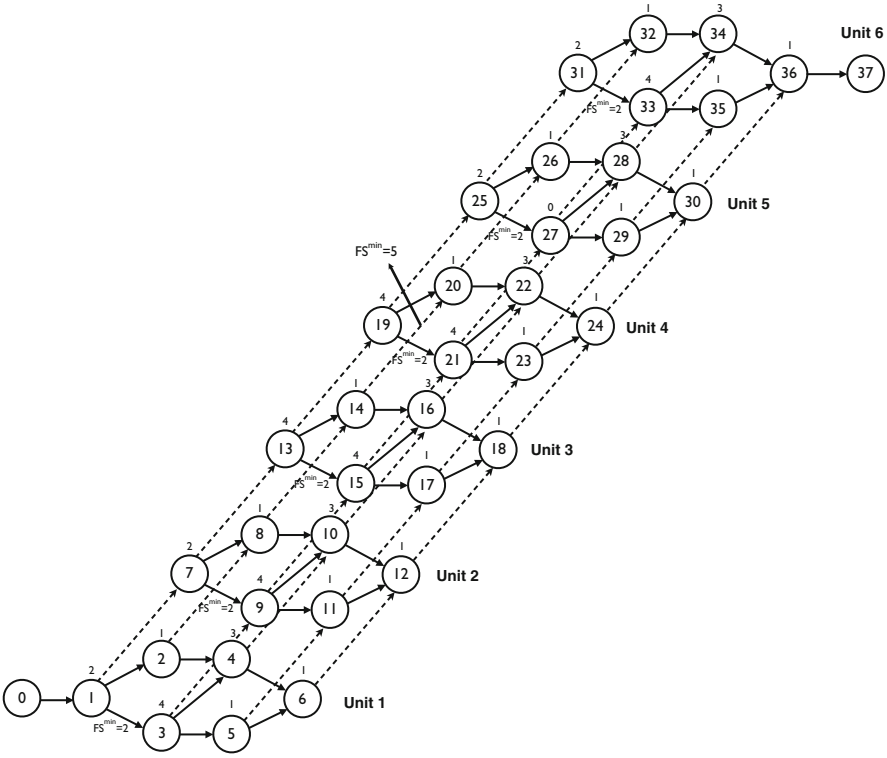


Fig. 8.3 The repetitive project network of Fig. 8.2 with six units

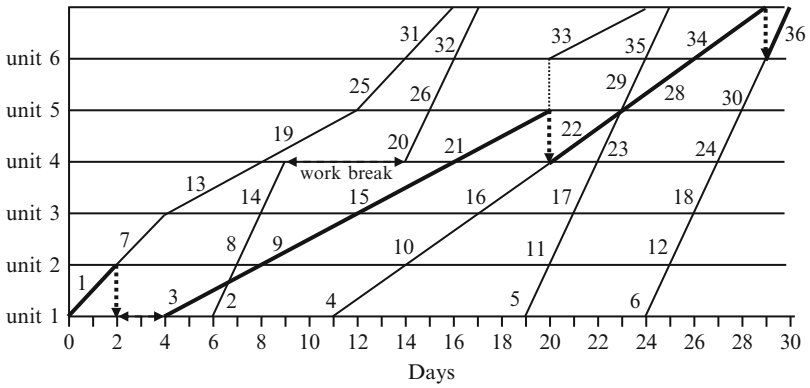


Fig. 8.4 RSM diagram for a six units project of Fig. 8.2

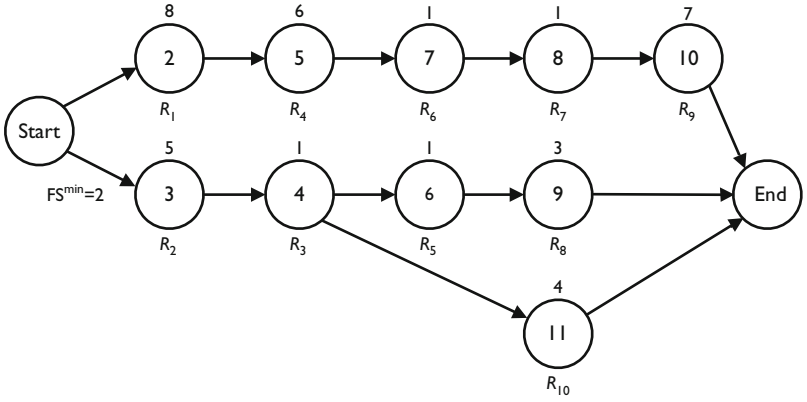


Fig. 8.5 An example project with ten repeating activities

done in the different units, the horizontal axis denotes the time line and the numbers next to the lines refer to the activity numbers of the project network of Fig. 8.3. The slope of each line is equal to the unit production rate, i.e. the number of repetitive units that can be accomplished by a resource during a unit of time. Consequently, it can be calculated as the inverse of the duration of that activity at that unit. Total project duration equals 30 days and the resource idle time amounts to 5 days (i.e. equal to the work break between activity 14 and 20 in the network of Fig. 8.2). The minimal time-lags between activities 1 and 3 at all units do not affect the resource idle time. The line in bold is the so-called *controlling sequence* and determines the length of the project duration. Obviously, this controlled sequence is similar to the critical path and critical chain concepts discussed in the previous chapters. This example illustrates that the general project scheduling approach can be easily applied and translated to more specific sectors (in this case, the RSM method is mainly used in the construction sector) without changing the general techniques and principles discussed earlier. More details on the repetitive scheduling method are outside the scope of this book.

Example 2 (Trade-off between work continuity and project duration). In the first example, the schedule with the best work continuity cost, expressed as the solution with the minimal idle time for all resources shifting between units, has a length equal to the critical path length. However, in reality, there is often a trade-off between idle time reduction and project duration, leading to a project schedule that can exceed its minimal critical path (or critical chain) duration. Indeed, thanks to a longer project duration, there are more degrees of freedom to group certain activities that require the same set of resources.[1] In this section, the use of the so-called horizon-varying scheduling approach (Vanhoucke, 2006b) is illustrated on the project example of Fig. 8.5. In this figure, a unit network is displayed with ten nondummy activities.

[1] In Chap. 9, it will be shown that there is a trade-off between the project duration and the idle time of expensive freezing machines for the construction of a tunnel under the Westerschelde.

In Table 8.1, the activity durations from units 1 to 5 are displayed. In this example, it is assumed that the crew productivity increases along the units, known as the learning effect of crews.

Figure 8.6 displays the complete trade-off profile between the total project duration and work continuity by means of the black bars. This is the result of a horizon-varying approach starting from the critical path length of 43 to a project duration of 55. A critical path length of 43 corresponds to a resource idle time of 32 time units due to waiting times of resources between units. Increasing the project deadline results in a lower resource idle time thanks to lower waiting times between units. A project duration of 55 time units corresponds to a minimal resource idle time between units. Note that the minimal resource idle time equals zero since only zero time-lags are involved. This trade-off profile can be used as a decision tool to determine an optimal level of resource idle time in the schedule. By assigning costs to both resource idle time and project duration, the optimal point in the complete profile with an associated project duration and idle time level can be determined. Assume that c_r is used to denote the cost per unit resource idle time and c_d is used to denote the cost per time unit that has to be paid during each day of the project duration. Consequently, the total cost of a schedule with total planned project duration PD and corresponding resource idle time (*idle*) equals $c_t = c_r * idle + c_d * \text{PD}$. Figure 8.6 reports the total cost c_t by four lines depending on the values for c_r and c_d.

Table 8.1 Activity durations of the activities of Fig. 8.5 for five units

Unit/activity	2	3	4	5	6	7	8	9	10	11
Unit 1	8	5	1	6	10	1	1	3	7	4
Unit 2	7	4	1	5	8	1	1	2	6	3
Unit 3	6	3	1	4	7	1	1	1	5	2
Unit 4	5	2	1	3	6	1	1	1	4	1
Unit 5	4	1	1	2	5	1	1	1	3	1

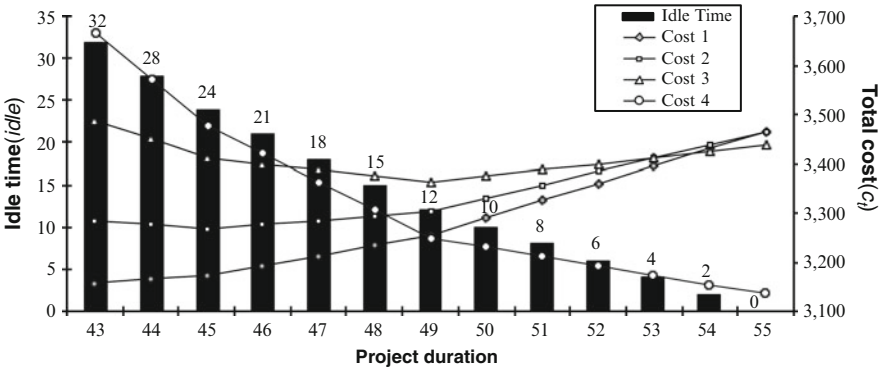


Fig. 8.6 Trade-off between work continuity and project deadline

The optimal project duration and the corresponding level of idle time depend on both the values for c_r and c_d. Each unit increase in the project duration involves an extra cost c_d while the total cost will be decreased by c_r times the idle time reduction due to the project duration increase. Consequently, a project duration increase is only beneficial as long as c_d/c_r is smaller than the (negative) slope of the crew idle time curve as displayed in Fig. 8.6. As an example, the black bars of Fig. 8.6 have 3 different values for the slope, i.e. 4 between 43 and 45, 3 between 45 and 49 and 2 from 49 onwards. Consequently, four different solutions can be optimal, depending on the cost values c_d and c_r:

- $\frac{c_d}{c_r} > 4$: It is never beneficial to increase the project duration, and the optimal solution equals the critical path length 43. This is displayed in Fig. 8.6 by the curve labelled 'Cost 1' with $c_d = 63$ and $c_r = 14$ which has its lowest point at project duration 43.
- $3 < \frac{c_d}{c_r} \leq 4$: It is beneficial to increase the project duration up to 45. This is displayed by the curve labelled 'Cost 2' with $c_d = 63$ and $c_r = 18$.
- $2 < \frac{c_d}{c_r} \leq 3$: It is beneficial to increase the project duration up to 49. This is displayed by the curve labelled 'Cost 3' with $c_d = 62.5$ and $c_r = 25$.
- $\frac{c_d}{c_r} \leq 2$: A maximal increase in the project duration leads to the lowest cost. This is displayed by the curve labelled 'Cost 4' with $c_d = 57$ and $c_r = 38$ with a minimal cost for a project duration of 55.

As a summary, the optimal project duration always coincides with a slope breakpoint in the trade-off curve between idle time and project duration, i.e. the optimal project duration will lie at the points with project durations equal to 43, 45, 49 or 55, depending on the costs c_d and c_r.

8.2.2 Quality Dependent Time Slots

Quality-dependent time slots refer to predefined time windows where certain activities can be executed under ideal circumstances (optimal level of quality) while outside these time windows, there is a loss of quality due to detrimental effects. The purpose is to select a quality-dependent time slot for each activity, resulting in a minimal loss of quality.

The topic of this section is based on real-life project data aiming at scheduling an R&D project from the bio-technology sector with genetically manipulated plants. In this project, several activities need to be scheduled in the presence of limited resources and severe quality restrictions. More precisely, some activities need to be executed preferably within certain predefined periods, referred to as quality-dependent time slots. Although the execution is also possible outside these predefined intervals, it is less desirable since it leads to a decrease in quality.

It is assumed that each activity has several predefined quality-dependent time slots, from which one has to be selected. The selection of a time slot must be done before the start of the project (i.e. during the scheduling phase). Given a fixed set

of time slots per activity, the target is then to select a time slot and to schedule the project such that the loss in quality will be minimized. Each activity has a duration $d_i (1 \leq i \leq n)$ and a number of quality-dependent time windows $nr(i)$. Each window l of activity i ($1 \leq i \leq n$ and $1 \leq l \leq nr(i)$) is characterized by a time-interval $[q_{il}^-, q_{il}^+]$ of equal quality, while deviations outside that interval result in a loss of quality. Note that the time slot $[q_{il}^-, q_{il}^+]$ is used to refer to a window with optimal quality and can be either an interval or a single point in time. The quality deviation of each activity i can be computed as $Q_i^{loss} = \max(q_{il}^- - s_i; s_i - q_{il}^+; 0)$ and depends on the selection of the time window l, with s_i the start time of activity i.

To that purpose, a binary decision variable needs to be introduced in the conceptual model, which determines the selection of a specific time interval for each activity i, as follows:

$$\begin{aligned} y_{il} \quad & = 1, \text{ if time interval } l \text{ has been selected for activity } i, \\ & = 0, \text{ otherwise} \end{aligned}$$

q_{il}^{opt} is used to denote the minimal activity cost associated with a fixed and optimal level of quality for each time window l of activity i. q_{il}^{extra} is used to denote the loss in quality per time unit deviation from the time interval and consequently, the total cost of quality equals $\sum_{i=1}^{n} \sum_{l=1}^{nr(i)} (q_{il}^{opt} + q_{il}^{extra} Q_i^{loss}) y_{il}$. Note that $nr(1) = nr(n) = 1$, since nodes 1 and n are dummy activities with $q_{11}^- = q_{11}^+$ and $q_{n1}^- = q_{n1}^+$. Moreover, the dummy start activity must be forced to start at time instance zero. The project needs to be finished before a negotiated project deadline δ_n, i.e. $q_{n1}^- = q_{n1}^+ = \delta_n$. Consequently, setting $q_{n1}^{extra} = \infty$ denotes that the project deadline can not be exceeded (a hard constraint), while $q_{n1}^{extra} < \infty$ means that the project deadline can be exceeded at a certain penalty cost (soft constraint). The *resource-constrained project scheduling problem with quality-dependent time slots* (RCPSPQTS) can be conceptually formulated as follows:

$$\text{Minimize} \sum_{i=1}^{n} \sum_{l=1}^{nr(i)} (q_{il}^{opt} + q_{il}^{extra} Q_i^{loss}) y_{il} \tag{8.9}$$

subject to

$$s_i + d_i \leq s_j \qquad \forall (i, j) \in A \tag{8.10}$$

$$\sum_{i \in S(t)} r_{ik} \leq a_k \qquad k = 1, \ldots, K; t = 1, \ldots, T \tag{8.11}$$

$$Q_i^{loss} \geq \sum_{l=1}^{nr(i)} q_{il}^- y_{il} - s_i \qquad i = 1 \ldots, n \tag{8.12}$$

$$Q_i^{loss} \geq s_i - \sum_{l=1}^{nr(i)} q_{il}^+ y_{il} \qquad i = 1 \ldots, n \tag{8.13}$$

$$\sum_{l=1}^{nr(i)} y_{il} = 1 \qquad i = 1 \ldots, n \tag{8.14}$$

$$s_1 = 0 \tag{8.15}$$

$$s_i, Q_i^{loss} \in int^+ \qquad i = 1 \ldots, n \tag{8.16}$$

$$y_{il} \in 0, 1 \qquad i = 1 \ldots, n; l = 1, \ldots, nr(i) \tag{8.17}$$

where $S(t)$ denotes the set of activities in progress in period $]t-1, t]$. The objective in Eq. 8.9 minimizes the total quality cost of the project (i.e. the fixed cost within the selected time window plus the extra cost of quality loss due to deviations from that interval). The constraint set given in Eq. 8.10 maintains the finish-start precedence relations among the activities. Equations 8.11 represent the renewable resource constraints and the constraint sets in Eq. 8.12 and 8.13 compute the deviation between the activity start time and the selected time window. Equations 8.14 represent the time window selection and forces the model to select a single time window for each activity. Equation 8.15 forces the dummy start activity to start at time zero and Eq. 8.16 ensure that the activity start times as well as the time window deviations assume nonnegative integer values. Equations 8.17 ensure that the time window selection variable is a binary (0/1) variable. Remark that the quality loss function measuring the quality decrease due to a deviation from the ideal time window l can be of any form (such as stepwise functions, convex functions, etc). However, Eqs. 8.9–8.17 assume, without loss of generality, a linear quality deviation function.

Although the first real-life application of this scheduling problem type was the scheduling of a genetically manipulated plants project, there are numerous other examples where predefined time-windows need to be selected before the execution of the project. The following four examples illustrate the possible generalization of multiple quality-dependent time windows to other project environments:

- Perishable items. The project scheduling problem with quality dependent time slots is a typical example of a scheduling environment where items (e.g. plants) are perishable. Many project activities consist of tests on growing plants where the quality is time-dependent since there is an optimal time interval of consumption. Earlier consumption is possible, at a cost of a loss in quality, since the plants are still in their ripening process. Later consumption results in loss of quality due to detrimental effects.
- State-of-nature dependencies. In many projects, the performance of some activities might depend on the state of nature. In this case, a predefined set of possible start times depending on the state of nature are linked with possible execution times of the activity and the deviation from these time windows is less desirable (resulting in higher costs or quality loss) or even completely intolerable. A spectacular example, where six different quality-dependent time slots due to an external state of nature reason, occurred in the Rosetta mission. In early

January 2003, an Ariane-5 rocket carrying the ESAs Rosetta Spacecraft was launched from Kourou, French Guiana. The main objective of the Rosetta mission was a rendez-vous with a Comet called Wirtanen. Therefore, the spacecraft had to gather momentum three times in the gravity fields of Mars (26/08/2005) and the Earth (28/11/2005 and 28/11/2007) in order to get into the outer regions of the planetary system. The planet constellation required for that purpose could only last for a short period. As a consequence, the launch window was only open for 6 specific days between January 13 and 31, 2003 (i.e. six different time slots). Afterwards, the comet Wirtanen could no longer be reached. Besides the restricted span of launch dates, there was also a tight limit on the time of day at which Rosetta could leave earth. Because the earth rotates, Kourou had to be correctly positioned in relation to the direction in which the spacecraft had to head off, on the first part of its interplanetary journey. The daily time span was about 20 min. So the ultimate deadline was the date of the first possible constellation.

- Multiple activity milestones. The project scheduling literature with activity due dates (milestones) has been restricted to considering projects with predefined due dates. In reality, milestones are the results of negotiations, rather than simply dictated by the client of the project. Indeed, due dates, including earliness and tardiness penalty costs for possible deviations, are agreed upon by the client and the contractor (and possibly some subcontractors). This results in a set of possible due dates for each activity, rather than a single predefined due date. The objective is then to select a due date for each activity such that the total earliness/tardiness penalty costs will be minimized.
- Time-dependent resource cost. In many projects, the cost of (renewable) resources heavily depends on the time of usage. The aforementioned time-switch constraints (see Chap. 2) are a typical and extreme example of time-dependent resource costs, since they restrict the execution of activities to predefined time intervals (work periods) without any possibility to deviate. However, when activities are allowed to deviate from their original work periods, these time slots can be considered as quality-dependent time slots (in this case, the cost of overtime). Indeed, it is often possible to deviate from an activity pattern by adding more (expensive) resources to an activity in the predefined rest period. The work periods can be considered as the time slots while the rest periods are periods outside these slots in which the activity can be executed at an additional cost.

This project scheduling type with limited resources and quality-dependent time slots can be easily illustrated by means of an example project in Fig. 8.7. The number above each node is used to denote the activity duration d_i while the number below the node represents its requirement r_{i1} for a single renewable resource with availability $a_1 = 10$. Table 8.2 shows the $(q_{il}^{opt}, q_{il}^{-} = q_{il}^{+}, q_{il}^{extra})$ data for each quality-dependent time slot l. For the sake of clarity, it is assumed that $q_{il}^{-} = q_{il}^{+}$, i.e. the quality-dependent time slots are a single point in time. Moreover, q_{il}^{extra} is assumed to be equal for each interval l. Each activity of the example project

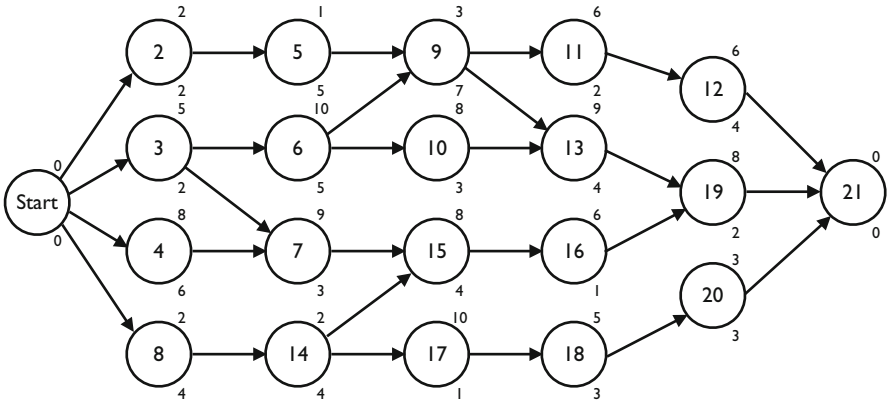


Fig. 8.7 An example project with quality-dependent time slots

Table 8.2 The quality dependent time slot data for Fig. 8.7

	q_{i1}^{opt}	$q_{i1}^{-} = q_{i1}^{+}$	q_{i1}^{extra}	q_{i2}^{opt}	$q_{i2}^{-} = q_{i2}^{+}$	q_{i2}^{extra}
1	–	–	–	–	–	–
2	0	0	4	–	–	–
3	0	1	19	2	4	19
4	0	0	6	–	–	–
5	0	9	2	2	19	2
6	0	5	1	–	–	–
7	0	8	8	–	–	–
8	0	8	16	2	18	16
9	0	15	1	–	–	–
10	3	15	6	–	–	–
11	0	18	6	–	–	–
12	0	37	8	–	–	–
13	0	23	2	–	–	–
14	0	20	2	–	–	–
15	0	22	6	–	–	–
16	0	30	1	–	–	–
17	0	23	4	–	–	–
18	0	23	3	2	33	3
19	0	36	7	–	–	–
20	0	19	6	2	39	6
21	–	–	–	–	–	–

belongs to one of the following categories. An activity can be subject to single (activities 12 and 17) or multiple quality-dependent time slots (activities 3, 5, 8, 18 and 20). Activities can also have the requirement to be scheduled as-soon-as-possible (ASAP; activities 2, 4, 6, 7, 9, 10, 11 and 13) or as-late-as-possible (ALAP; activities 14, 15, 16 and 19). This can be incorporated in the network by

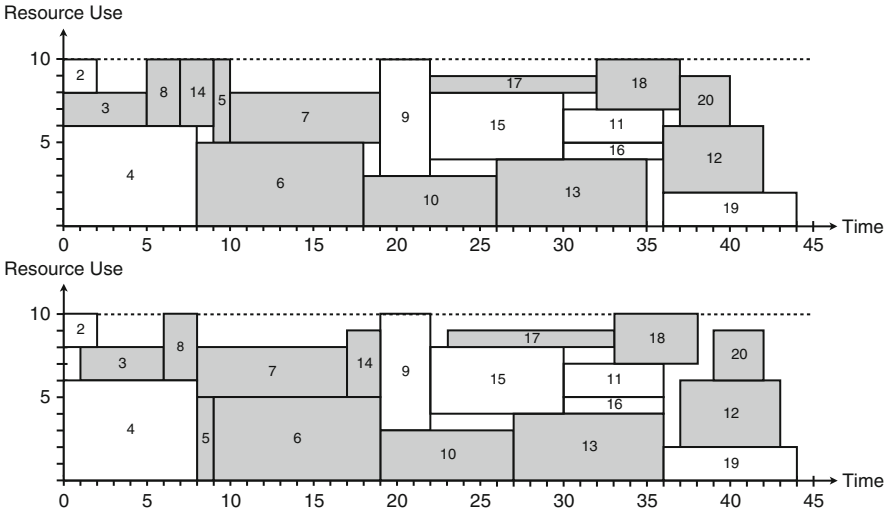


Fig. 8.8 The RCPSP schedule with minimal time (*top*) and quality-dependent time slots (*bottom*)

adding a single quality-dependent time slot with $q_{il}^- = q_{il}^+ = es_i$ (earliest start time of activity i) or $q_{il}^- = q_{il}^+ = ls_i$ (latest start time of activity i) to force an ASAP or ALAP constraint, respectively. Deviations from these requirements will be penalized by per time unit. The project deadline δ_n equals 44 time units.

Figure 8.8 displays the schedules found by solving the RCPSP without (i.e. minimization of project time) and with the quality-dependent time slots. The activities highlighted in dark grey are the activities that are scheduled at a different time instance between the two schedules. Activities 3, 6, 8, 10, 12, 13, 14, 17, 18 and 20 have been scheduled later than the classical RCPSP schedule, while activities 5 and 7 have been scheduled earlier.

8.2.3 Resource Availability Cost Problem

The *Resource Availability Cost Problem* (RACP) is a special case of the project scheduling problem, which can be considered as a combination of scheduling features discussed in the previous chapter. It is very much related to the resource leveling problem of Sect. 7.3.4 since it considers the use of resources in the objective and contains the time minimization problem of Sect. 7.3.2 as a subproblem. Despite its practical relevance, the scheduling problem has received relatively less attention than other scheduling problems, which might be attributed to its inherent scheduling complexity. The resource availability cost problem involves the construction of a project schedule within a predefined project deadline δ_n such that the total cost of the

resources is found. It is assumed that the total cost $c_k(a_k)$ of a resource k depends on its availability a_k, regardless whether the resource is used by activities or not, with $c_k(a_k^1) < c_k(a_k^2)$ if $a_k^1 < a_k^2$. Consequently, the cheapest resource availability cost $c_k(a_k)$ is determined by the assigned availability a_k for each resource type k. The problem implicitly assumes that a resource is assigned to a project for the complete duration, leading to an overall cost, which depends on the amount of units assigned to the project. The RACP can be formulated as follows:

$$\text{Minimize} \sum_{k=1}^{K} c_k(a_k) \tag{8.18}$$

subject to

$$s_i + d_i \leq s_j \qquad \forall (i,j) \in A \tag{8.19}$$

$$\sum_{i \in S(t)} r_{ik} \leq a_k \qquad k = 1, \ldots, K; t = 1, \ldots, T \tag{8.20}$$

$$s_1 = 0 \tag{8.21}$$

$$s_n \leq \delta_n \tag{8.22}$$

$$s_i \in int^+ \qquad i = 1 \ldots, n \tag{8.23}$$

The objective in Eq. 8.18 is a discrete nondecreasing resource cost function that minimizes the total cost of the necessary resources, specified by their availability. All other constraints are similar to the resource-constrained project scheduling problems of Sect. 7.3.2 of the previous chapter. The difference, however, is that the resource availability a_k of Eq. 8.20 is now a decision variable instead of a predefined input parameter.

The construction of an RACP schedule consists of two steps: an availability assignment step followed by a project scheduling step. Indeed, once the availability a_k is determined for each resource type k, leading to a total cost as specified in the objective function, a resource-constrained project schedule needs to be constructed as discussed in Sect. 7.3.2. When the total duration of this schedule is lower than or equal to the predefined project deadline δ_n, the schedule is feasible, otherwise not. The feasible schedule with the lowest total cost, as a result of the weighted sum of resource costs with their availabilities, is the best possible schedule that can be found.

It has been mentioned that the use of resources often increases the complexity of a scheduling problem. It goes without saying that the RACP is also a complex problem since it consists of an iterative solution approach for the RCPSP. In order to reduce the computational effort when finding a feasible project schedule, the lower bound calculations of Sect. 7.4.2 can be easily applied as alternatives for the computational burdensome algorithms to construct a schedule with a minimal

project duration. When the lower bound gives a minimal project duration that exceeds the project deadline δ_n, there is no need to construct a project schedule with the current combination of resource availabilities a_k. In doing so, the often time consuming RCPSP step can be avoided and another a_k value for at least one resource k is necessary. Further details on search procedures for the RACP are outside the scope of this book and can be found in Möhring (1984) and Demeulemeester (1995).

The RACP can be best illustrated by means of a project example using the activity-on-the-node network of Fig. 7.1. It is assumed that the project uses two resource types, i.e. resource type 1 and 2 of Table 7.2. It is assumed that the total resource cost $c_k(a_k)$ is a linear nondecreasing function of a_k with a per unit cost of $c_1(a_k = 1) = 3$ and $c_2(a_k = 1) = 2$ for resource types 1 and 2 and the project deadline is set to $\delta_n = 25$. The maximum resource demand is equal to 8 and 17 for resource types 1 and 2, respectively, requested by activities 9 and 10. Moreover, it can be shown that the minimum required resource availability is equal to 8 units for resource type 1 in order to be able to construct a resource feasible project schedule within the deadline of 25. This can be obtained by constructing an optimal resource-constrained project schedule (see Sect. 7.3.2) with a project deadline $\delta_n = 25$ only taking the first resource into account (the availability of resource type 2 is set to infinity). Likewise, resource type 2 needs at least 23 units within the given project deadline. Consequently, the search for the optimal combination between a_1 and a_2 starts at 8 and 23 units.

Table 8.3 enumerates eight possible combinations using the approach of Demeulemeester (1995). This approach stipulates that only efficient cost points needs to be evaluated in a strict order, starting from a low cost combination (denoted by START with a cost of 70) in increasing steps until the project duration is smaller than or equal to the project deadline (denoted by STOP). The exact sequence can be found by following the iso-cost curves denoted by the dashed lines with a cost slope equal to $\frac{c_1}{c_2} = \frac{3}{2} = 1.5$. Figure 8.9 illustrates the principle on the project example, starting with the minimal resource requirements $a_1 = 8$ and $a_2 = 23$ with a total resource cost of 41 and finishing at the best possible solution with $a_1 = 9$ and $a_2 = 25$. The dots represent a_1 and a_2 combinations and the dashed lines represent the iso-cost curves that determine the order in which the dots will be evaluated.

Table 8.3 Iterative search for the best possible resource availability combination

	a_1	a_2	Cost	Time	Feasible?
START →	8	23	70	28	no
	8	24	72	27	no
	9	23	73	28	no
	8	25	74	27	no
	9	24	75	27	no
	10	23	76	27	no
	8	26	76	27	no
	9	25	77	25	yes → STOP

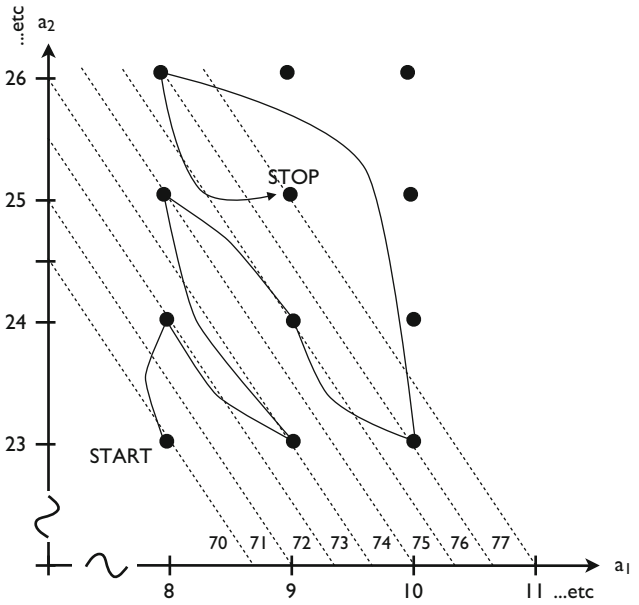


Fig. 8.9 The efficient cost curves for two resource types and the search for the best resource availabilities

8.3 Quantitative Project Descriptions

Quite a number of measures have been proposed in the literature to describe the characteristics of a project. A clear distinction can be made between measures capturing information about the size and the topological structure of the network and measures that are related to the different resources allocated to the project. In the following sections, the commonly used measures published in the project scheduling literature are briefly discussed, without going into mathematical details. Parts of it have already been discussed in a previous part of the book (Sect. 5.5.2). The reader is free to skip this section without running the risk to lose grip on the overall dynamic scheduling picture discussed throughout the various chapters of this book.

8.3.1 Network Topology

Numerous measures to indicate a network's topology have been presented in the literature. This section gives a brief overview of (some of) these measures used in the literature and/or software tools. A complete overview and mathematical details

are outside the scope of this chapter. For a more detailed discussion of the network characteristics, the reader is referred to the references mentioned in this section.

In Demeulemeester et al. (2003), three well-known complexity measures to describe the topological structure of a network are used in a project network generator in order to test and validate project scheduling algorithms. These network measures are briefly described along the following lines:

- The *Coefficient of Network Complexity* (CNC) is probably the easiest measure to describe the topology of a project network and is calculated as the total number of precedence relations (arcs) over the total number of project activities (nodes) in the network (Davies, 1974; Davis, 1975; Kaimann, 1974, 1975; Pascoe, 1966).
- The *Order Strength* (OS) is defined as the number of precedence relations (not only the direct relations but also including the transitive ones) divided by the theoretical maximum number of precedence relations (Dar-El, 1973; Herroelen and De Reyck, 1999; Kao and Queyranne, 1982; Mastor, 1970; Tavares, 1999; Thesen, 1977). The maximum number of possible precedence relations in a project network is equal to $\frac{n(n-1)}{2}$ with n the number of nondummy activities. This is the most widely used measure in the literature and has been used in phase transition studies to show regions where resource-constrained project scheduling problems are very complex (Herroelen and De Reyck, 1999).
- The *Complexity Index* (CI) was originally defined for activity-on-the-arc networks as the so-called reduction complexity and basically measures the closeness of a network to a series-parallel directed graph. More precisely, it calculates the minimum number of node reductions that, along with series and parallel reductions, allow to reduce a two-terminal acyclic network to a single edge (Bein et al., 1992). Further details are outside the scope of this book.

Tavares et al. (1999) have proposed six network topology measures to describe the design and structure of an activity-on-the-node project network and they have been redefined by Vanhoucke et al. (2008) to use them in a comparative network study. Four of these indicators have been used in a dynamic project scheduling study of Vanhoucke and Vandevoorde (2007b). These four indicators have been rescaled and lie between 0 and 1, inclusive, denoting the two extreme structures. The logic behind each indicator is straightforward, as follows:

- The *Serial/Parallel* (SP) indicator measures the closeness of a network to a serial or parallel network. More precisely, when $\text{SP} = 0$, all activities are in parallel, and when $\text{SP} = 1$, the project is represented by a complete serial network. Between these two extreme values, networks can be generated close to a serial or parallel network. The SP indicator determines the maximal number of levels of the network, defined as the longest chain (in terms of the number of serial activities) in the network. This indicator has already been mentioned in Sect. 5.5.2 of the schedule risk analysis chapter.
- The *Activity Distribution* (AD) indicator measures the distribution of project activities along the levels of the project, and hence, the width of the network. When $\text{AD} = 0$, all levels contain a similar number of activities, and the number

of activities are uniformly distributed over all levels. When AD = 1, there is one level with a maximal number of activities, and all other levels contain a single activity.

- The *Length of Arcs* (LA) indicator measures the length of each precedence relation (i, j) in the network as the difference between the level of the end activity j and the level of the start activity i. When LA equals 0, the network has many precedence relations between two activities on levels far from each other such that the activity can be shifted further in the network. When LA equals 1, many precedence relations have a length of one, resulting in activities with immediate successors on the next level of the network and with little freedom to shift.
- The *Topological Float* (TF) indicator measures the topological float of a precedence relation as the number of levels each activity can shift without violating the maximal level of the network (as defined by SP). TF = 0 when the network structure is 100% dense and no activities can be shifted within its structure with a given SP value. A network with TF = 1 consists of one chain of activities without topological float (they define the maximal level and, consequently, the SP value) while the remaining activities have a maximal float value (which equals the maximal level, defined by SP, minus 1).

A more extensive discussion of the topology measures falls outside the scope of this book. Further mathematical details can be found in Vanhoucke et al. (2008), and illustrative examples are given by Vanhoucke (2010a).

8.3.2 Resource Scarceness

Several measures to describe the resource scarceness have been introduced in the literature. The scarceness of project resources can be measured along two dimensions. A first dimension is related to the number of resources used by the project activities and is measured as the density of the resource demand matrix r_{ik} (demand of activity i for resource k) in order to specify whether an activity uses a particular resource or not. This is done by computing the resource factor RF or the resource use RU. Secondly, the quantity in which these resources are used, measured by the amount of resource demand relative to its availability for each activity, plays an important role in the resource scarceness. This second dimension is measured by the resource strength RS or the resource-constrainedness RC. These two sets of resource scarceness indicators are briefly described along the following lines:

Average number of project resources used:

- The *Resource Factor* (RF) reflects the average portion of resource types requested per activity and consequently measures the density of the matrix r_{ik}. It simply scans for each activity/resource combination whether the resource is requested by the activity or not and calculates the average portion for all

resources requested by all activities, which obviously results in a percentage of resource use (Pascoe, 1966; Cooper, 1976; Alvarez-Valdes and Tamarit, 1989). It is used in studies by Kolisch et al. (1995) and Schwindt (1995), among others.
- The *Resource Use* (RU) is a similar measure. However, it does not calculate the use of resources as a percentage for all resources used in the project, but instead, varies between zero and the number of resource types available and measures for each activity the number of resource types used for the project. It is used in a study by Demeulemeester et al. (2003).

Average amount of resource use:

- The *Resource Strength* (RS) can be calculated for a resource type k as $\frac{a_k - r_k^{\min}}{r_k^{\max} - r_k^{\min}}$ where a_k denotes the total availability of renewable resource type k, $r_k^{\min}$ is equal to the maximum requested amount for resource type k for all activities and $r_k^{\max}$ denotes the peak demand of resource type k in the resource-unconstrained earliest start schedule. The resource strength RS was first introduced by Cooper (1976), then later used by Alvarez-Valdes and Tamarit (1989) and finally redefined by Kolisch et al. (1995). The measure is criticized by various authors (Elmaghraby and Herroelen, 1980; Herroelen and De Reyck, 1999) since it cannot be considered as a pure resource measure due to the incorporation of network topology information (during the calculation of $r_k^{\max}$, an earliest start schedule is built).
- The *Resource Constrainedness* (RC) measures the average quantity of each resource type k required by all project activities, divided by its availability. The measure has been introduced by Patterson (1976) and is used in several resource complexity studies.

8.3.3 Relevance

The interest in project topology and resource measures dates back to Elmaghraby and Herroelen (1980) who draw attention to the need for project datasets that span the full range of problem complexity. Ever since, many researchers have followed this advice, leading to project network generators with resource generation capabilities using one or more of the measures described earlier (Demeulemeester et al., 1993; Kolisch et al., 1995; Schwindt, 1995; Agrawal et al., 1996; Tavares, 1999; Drexl et al., 2000; Demeulemeester et al., 2003; Akkan et al., 2005; Vanhoucke et al., 2008). Varying network topology structures and resource measures during the generation of fictitious project data is necessary due to the complexity of resource-constrained project scheduling. This is also illustrated by the complexity dimension of the project mapping Fig. 1.4 where it is said that the introduction of resources increases the project scheduling complexity. The incorporation of these measures in software tools can be interesting for the user and/or project manager for several reasons, for example:

- Quantitative description of the project: Having (quantitative) knowledge about the characteristics of the project (network and resources) can be helpful in selecting tools and techniques for project scheduling, risk analysis and project control purposes. In Chap. 5, it has been briefly shown that the reliability of sensitivity measures obtained by a schedule risk analysis depends on the use of network topology measures. Chapter 13 gives empirical evidence based on a simulation study that the efficiency of project control depends on the network topology of the project and that these quantitative measures can be used to select the most appropriate tracking method for the project under study.
- Automatic generation of project data: When using a software tool, many users initially try to use the functionalities of the tool without having real data. The generation of fictitious project data is an easy and powerful tool to let the user start immediately with the software to gain experience with all its features even before entering real project data. The generation of fictitious data can be best done under a user-defined design, having control over the network structure and the scarceness of project resources.
- Resource scarceness in multi-project environments: Knowledge about the scarceness of resources is particularly interesting in a multi-project environment. As an example, in Chap. 10, it will be briefly discussed that the size of project and feeding buffers should not only depend on the risk of the activities in the critical chain or feeding chains, but also on the scarceness of the project resources. The higher the resource scarceness, the more likely the project duration will increase when more projects enter the project portfolio.

8.4 Extra Scheduling Features

In Sect. 7.5 of Chap. 7, some commonly used extensions to the basic resource-constrained project scheduling techniques have been briefly discussed. However, the range of possible extensions to the basic resource-constrained project scheduling problems is wide and diverse and depends on the needs and wishes from industry as well as the state-of-the-art developments in research. This section gives an overview of three extensions that have an important impact on real-life project scheduling and is far from complete. Results are based on research projects done at Ghent University in collaboration with various companies.

8.4.1 Activity Assumptions

Most, if not all, project scheduling software tools aim at the construction of resource feasible schedules in order to minimize the total lead time of the project, known as the RCPSP as presented in Chap. 7. Hence, an activity-on-the-node project network with a list of activities with their corresponding precedence relations and resource

requirements needs to be given as an input. However, various activity assumptions need to be made by the user in order to construct a feasible schedule. In a research project done by Vanhoucke and Debels (2008), the impact of the work content option, the possibility of activity preemption and the presence of fast tracking is studied on the resource utilization and project duration of a project. Figure 8.10 gives an overview of the three activity assumptions tested in the experimental study for a project activity with a total work content of 9 man-days (for example, the duration $d_i = 3$ and a single renewable resource requirement $r_{i1} = 3$), which will be briefly described along the following lines.

- Fixed duration or fixed work: The basic *resource-constrained project scheduling problem* RCPSP assumes that each activity i consists of a deterministic work content W_{ik} for each resource type k, and imposes a fixed duration d_i and a fixed renewable resource requirement r_{ik} on its execution. The extension to the *discrete time/resource trade-off problem* (DTRTP) still assumes a fixed work content but allows variable activity durations. As an example, the activity of Fig. 8.10 still has a fixed work content W_{i1} of 9 for the single resource type 1, but can now be executed under different scenarios. Note that many commercial software tools pay a lot of attention to this activity assumption and call for the well-considered use of this activity option before the construction of a schedule (see e.g Uyttewaal 2005 for examples). The choice between fixed durations or fixed work has been discussed previously in Sect. 7.5.2.
- The presence of activity preemption: The basic RCPSP assumes that each activity, once started, will be executed until its completion. The extension to the *preemptive resource-constrained project scheduling problem* (PRCPSP) allows activities to be preempted at any integer time instant and restarted later on at no additional cost, and has been investigated in the literature as an option to

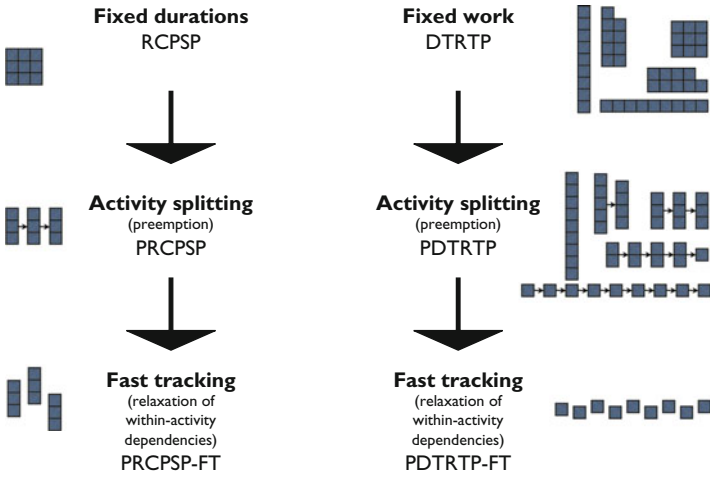


Fig. 8.10 Resource-constrained project scheduling under various activity assumptions

further reduce the total project lead time. In many project scheduling software tools, the option of activity splitting can be made before the construction of a resource-feasible schedule and has an effect on the number of execution scenarios, as displayed in Fig. 8.10. In theory, the DTRTP can also be extended to its preempted version (the *preemptive discrete time/resource trade-off problem* (PDTRTP)), but no results are available on this problem type in the literature.

- The effect of fast tracking: Fast tracking is a scheduling technique used to reduce the total project lead time during project execution even further. When projects are fast-tracked, it usually indicates the compression of a project schedule by doing certain activities in parallel that would normally be done in a sequence. Hence, it violates the precedence relations between activities, which implies activity execution at incomplete information. In the study, the impact of within-activity fast tracking is investigated, which allows the execution of preemptive subparts of an activity in parallel. The fast tracking option removes precedence relations between subparts of preempted activities and increases the number of execution scenarios. The within-activity fast tracking option is inspired by the idea that activities are often executed by groups of resources (with a fixed availability), but the total work can often be done by multiple groups (in parallel). The *preemptive resource-constrained project scheduling problem with fast tracking* (PRCPSP-FT) assumes preemptive activities with fixed durations, which results in d_i parallel subactivities for each original activity, each with a resource requirement r_{ik}. The *preemptive discrete time/resource trade-off problem with fast tracking* (PDTRTP-FT) assumes variable activity durations (under a fixed work content) and allows the preemptive and parallel execution of each subactivity with a duration and resource requirement equal to 1, as shown in the bottom part of Fig. 8.10.

The results of the study can be summarized as follows. First, allowing preemption in the RCPSP has almost no effect on both the lead time and the resource utilization when the resource availability a_k is fixed over the complete project life time horizon. Hence, the task splitting option of project scheduling software, which results in preemptive and often less clear schedules, is no good alternative to improve the schedule quality. Second, the shift from fixed duration activities to fixed work content activities (DTRTP), however, has a major effect on both the lead time (an improvement with approximately 21%) and the resource utilization (from approximately 75% to 94% or more). Hence, the fixed work option should be carefully considered as a default option, since it has a major beneficial effect on the schedule quality. Third, within-activity fast tracking turns out to have a beneficial effect on the fixed duration activities (PRCPSP-FT), leading to approximately 15% lead time improvement and an 88% resource utilization, but the extra benefits when using fixed work activities (PDTRTP-FT) are relatively small compared to the very efficient schedules found by the DTRTP. Hence, allowing fixed work activities already results in a very efficient schedule, making the within-activity fast tracking a redundant alternative to improve schedule quality.

8.4.2 Setup Times

In the previous section, it has been shown that activity preemption seldom has a significant impact on the total project duration compared to the RCPSP duration, but, on the contrary, activity preemption in combination with activity fast-tracking of these preempted subparts of activities can lead to large project duration reductions. However, preemptions in activities often come at a certain price. Indeed, the study of the previous section did not take preparation or setup work into account and simply assumed that activities can be started, preempted and/or fast-tracked at no extra time or cost. However, in daily scheduling activities, it is noticed that most project planners make a distinction in an activity duration between the actual work (in days) and the time needed to prepare the actual work (currently called setup time) such as installation time, releasing resources from other sites, etc. Figure 8.11a displays details about a fictitious activity with a total duration d_i of 6 days (horizontal direction of the activity), consisting of 1 setup day and 5 remaining days of actual work. The project activity needs to be executed by teams of two people working together (vertical axis of the activity). Part b of the figure shows that activity preemption results in a renewed setup time due to the interruption in time. Consequently, the total setup time amounts to 2 days while the actual work remains 5 days (which is known in the literature as preemption-resume). The activity fast-tracking option of part c is the result of assigning two teams (each containing two people working in parallel) on the interruptive parts of the activity, and hence, the two preemptive parts can now be executed in parallel. Consequently, this fast tracking option removes precedence relations between subparts of preempted activities and increases the number of execution scenarios for each activity of the project.

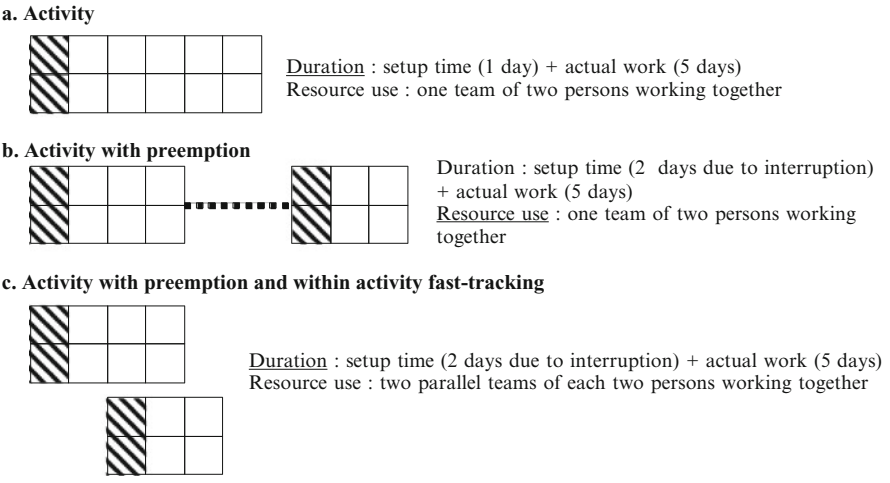


Fig. 8.11 Resource-constrained project scheduling with setup times

Vanhoucke (2008c) has studied the impact of these setup times on the PRCPSP-FT as option c of Fig. 8.11 where activity preemptions and/or within-activity fast trackings are only allowed at the expense of an extra setup cost. Hence, the original defined activity durations d_i consist of both a single setup time t_i for starting the activity and a remaining processing time. The setup time component includes activity preparations such as equipping, resetting, changing, positioning, cleaning and warming up (Mika 2006). This setup time is added to the total duration each time the activity is interrupted. This problem formulation is highly relevant for projects where multiple resource units (e.g. teams of people or a combination of machines) are assigned to project activities. In this case, activity resource requirements are often defined as the minimal amount of resource units required to perform the activity, and hence, the duplication of this minimal amount of resource units allows the fast tracking of subparts of activities. Obviously, the duplication and corresponding fast tracking decision often involve extra setup time, as discussed before.

The author has performed an experimental study on randomly generated project data with up to 30 project activities and showed the relative decrease in the project duration when allowing activity preemption and fast tracking. The project networks have been generated such that they have different values for the network topology and resource scarceness. The network topology is determined by the amount of precedence relations between activities and leads to project networks with a lot of parallel to a lot of serial activities. The resource scarceness is measured by the average resource demand of the activities relative to its availability and has an influence on the scheduling complexity (vertical axis of Fig. 1.4). Each project requires four renewable resource types. In order to compare the resulting schedules with the optimal RCPSP schedules, it was assumed that part of the activity durations can be considered as the unavoidable setup time before the initial subactivity. Activity setup times have been generated under five settings as 0%, 25%, 50%, 75% or 100% of the original activity duration minus one. Consequently, the activity setup times and remaining activity durations have been calculated as follows: the author subtracted the generated setup time from the original activity duration to calculate the remaining activity duration. Hence, the sum of the activity setup time and its remaining duration was always equal to the original duration of the project network instance and the remaining duration is minimum one. This approach allows to measure the impact of preemptive fast tracking with setup times on the schedule quality by comparing it with the RCPSP project duration. The test set leads to three different scenarios, as follows:

1. If the setup time of each activity is set at $t_i = 0\%$, then the remaining duration for each activity is equal to the duration of the RCPSP instances. Since there are no setup times, the problem boils down to the PRCPSP-FT described in the previous section.
2. If the setup times for the activities are set at a value t_i from 25% to 75%, then the remaining durations of the activities are greater than 1 and lower than their

original duration. The minimal project duration will lie between the RCPSP-FT and the PRCPSP minimal project duration.
3. If the setup time of each activity is set at $t_i = 100\%$, then each remaining activity duration is equal to 1. In this case, activity preemption is impossible, and hence, the problem boils down to the basic RCPSP.

The author has investigated the impact of the different settings for the setup time, each split up to projects with a different network topology and resource scarceness, and have measured the average relative project duration improvement when introducing activity preemption and fast tracking with setup times, relatively compared to the minimal RCPSP solution as described in Chap. 7. A detailed analysis of the results of the experiment is outside the scope of this book. However, the main results and conclusions can be summarized as follows:

- Effect of the size of the setup times: The higher the value for the setup times, the less beneficial it is to preempt activities and hence, the closer the problem resembles the basic RCPSP. It is worth mentioning that the option to fast track has a major effect on the project duration, even with high values for the setup times. As an example, the PRCPSP-FT with setup times up to 75% of the original duration still leads to average improvements varying from 0.11% to 1.53% on average (depending on the scarceness of resources). For smaller setup times, this percentage can go up to 6.40%. This illustrates that the presence of relatively high setup times does not prevent project duration reductions when allowing activity preemption and/or fast tracking. Hence, if technical restrictions allow a within-activity fast tracking, even within the presence of relatively high setup costs, it is still beneficial to allow activity preemption as a technique to reduce the project duration.
- Effect of the network topology: The topology of the project network has a clear effect on the impact of setup times on the project duration. A network with higher Serial/Parallel (SP) value (measured by a higher amount of precedence relations in the project network resulting in a more serial network, see Sect. 8.3.1) benefits more from activity preemption and fast tracking than less dense networks that already allow a high degree of flexibility thanks to the high degree of potential parallelism between project activities.
- Effect of the resource scarceness: The scarceness of resources has a negative effect on the total project duration. Obviously, higher resource scarceness results in a highly complex scheduling problem due to the relatively little room to schedule the projects, which prevents further project duration improvements by preempting or fast tracking project activities.

8.4.3 Learning

Minimizing the total project duration during project scheduling is an important goal in today's competitive industrial environment (see Sect. 7.3.2 where the objective is the minimization of time). In project management, the project baseline schedule is

used as a benchmark and point of reference during the project's progress. Activity start and finish times are often seen as milestones and are used to follow up the progress of the project. However, the presence of learning effects can dramatically change the project baseline schedule, leading to changes in the activity durations, their start and finish times and consequently the total project duration.

The presence and importance of learning in project management is based on the observation that in most projects, human resources are a critical factor in the scheduling process. Not only their availability, but also their productivity will influence the project duration. One of the main reasons why the productivity of a human resource varies over time is because of the effect of learning, which indicates the process of acquiring experience while performing similar activities leading to an improvement of the worker's skill. As a measurable result of learning, the time required to perform the next activities decreases. The mathematical modeling of learning effects is outside the scope of this chapter, and the reader is referred to papers written by Wright (1936), Yelle (1979) and Nembhard and Uzumeri (2000) for general learning concepts, and to Shtub et al. (1996), Amor and Teplitz (1998), Ash and Smith-Daniels (1999) and Heimerl and Kolisch (2010) for learning in a project scheduling setting.

In a study by Van Peteghem and Vanhoucke (2010b), three different project baseline schedules are compared to each other in order to investigate the effect of activity learning on the project duration. Each schedule has a specific purpose and is constructed under different assumptions. The construction and interpretation of the three schedules will be explained by the use of a fictitious illustrative example project. The example project contains eight nondummy activities and two dummy activities to represent the start and end of the project. Moreover, it is assumed that all activities can be scheduled according to the fixed work option, where each activity's work content is displayed above each node of the activity-on-the-node network of Fig. 8.12a. These numbers are expressed in man-hours, and do not contain any learning effects. Consequently, the scheduling problem is assumed to be a *discrete time/resource trade-off problem* (DTRTP) as discussed in Sect. 8.4.1 where the work content W_{ik} for each resource type k implies the choice of an activity duration d_i and a fixed renewable resource requirement r_{ik} for its execution. The availability of the single renewable resource is equal to 10. Table 8.5 gives for each activity an overview of the allowable duration/resource pairs, the best duration/resource combination selected for each of the three schedules S^O, S^L and S^R and the activity start time of the activity.[2]

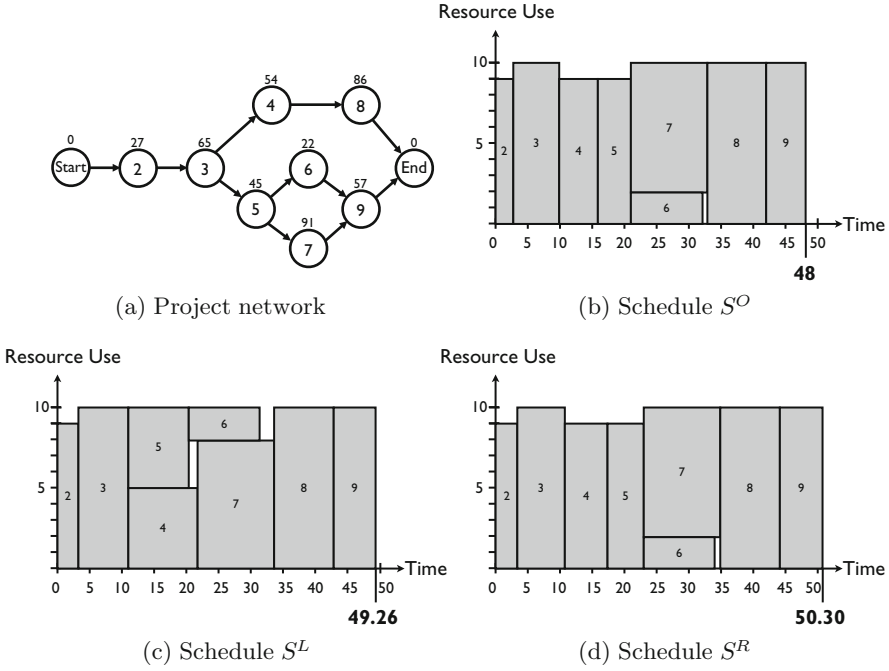
Activity Learning: The incorporation of learning in the allowable activity combinations has an effect on the total work content of the activities which may vary along the choice of the activity (d_i, r_{ik}) combination. The general idea is that

[2]The activity durations, their start times and the corresponding project duration for each schedule depend on the degree of learning for each activity. The calculations and construction of the schedule are outside the scope of this section and are determined by an algorithm developed by Van Peteghem and Vanhoucke (2010a).

teams with a lot of people (low d_i values and high r_{ik} values) have little time to learn while teams with only few people (high d_i values and low r_{ik} values) have much more time to learn. This general principle is illustrated in Table 8.4 for the six duration/resource combinations of activity 7 displayed in Table 8.5. The table shows that the incorporation of learning effects leads to longer durations for teams with a lot of people (ten people) and smaller durations for teams with only few people (below ten people). Obviously, the specific values for the durations and work content of the activity with learning depend on the learning rate and initial efficiency of the team, but these calculations are outside the scope of this chapter. In the remainder of this section, it is assumed that the time estimates without learning are artificial estimates made by the project manager, and the learning durations are the real durations that occur in reality. A comparison between three schedules will analyze the impact of incorporating the learning effects during the construction of the baseline schedule and the impact on the project duration when this is not done.

Three Baseline Schedules: The three project baseline schedules can be described along the following lines.

- S^O: The optimal schedule without incorporation of the learning effects. Figure 8.12b shows the optimal solution found by solving the DTRTP without activity learning effects, resulting in a project duration of 48 time units.



(a) Project network (b) Schedule S^O

(c) Schedule S^L (d) Schedule S^R

Fig. 8.12 Different schedules for an example project (Source: Van Peteghem and Vanhoucke (2010b))

Table 8.4 Illustrative learning effects on activity/resource combinations of activity 7 of Table 8.5

Team	No learning		Learning	
	d_7	W_{71}	d_7	W_{71}
Team 1: 10 people	10	100	10.06	100.60
Team 2: 9 people	11	99	10.93	98.37
Team 3: 8 people	12	96	11.78	94.24
Team 4: 7 people	13	91	12.63	88.41
Team 5: 6 people	16	96	15.13	90.78
Team 6: 5 people	19	95	17.56	87.80
Average	5.67	28.33		

- S^L: The optimal schedule with the incorporation of the learning effects. Figure 8.12c shows the optimal schedule solved by the DTRTP where each activity was subject to activity learning, leading to a total project duration of 49.26 time units. Consequently, the introduction of learning effects has led to a project duration increase of 2% which means that ignoring learning during the construction of a baseline schedule would lead to an underestimate of the project duration by 2%.
- S^R: The optimal schedule without learning effect using activity duration with learning. Figure 8.12d displays the so-called *realistic schedule* where the activity sequence in the schedule has been found by solving the DTRTP without activity learning, but where the activity durations have afterwards been replaced by their learning duration. The underlying assumption made is that the project manager is not aware of the existence of learning effects during the construction of a baseline schedule and therefore starts executing the project as shown in S^O. However, activities will take more or less time than originally planned due to the existence of learning effects. Therefore, S^R is the schedule where the sequence of the activities is defined by the start times of the different activities in the optimal schedule S^O, but where the activity durations are changed to the durations where a learning effect is considered. In the example, this results in a project schedule with a project duration of 50.30 time units. Obviously, this project duration should always be equal to or larger than the project duration of the S^L schedule, since in the latter, the learning effects have been incorporated in advance.

Effect of Learning: In the study, the impact of learning effects on the project baseline schedule is investigated from three different angles. First, the impact of learning effects on the optimal project duration is measured to test the relevance and importance of predicting these effects in advance. Second, the impact of learning during project progress is tested to measure the change in the total project duration when learning effects are ignored during the baseline scheduling phase. Finally, project progress with and without learning effects is compared to reveal the beneficial effect of incorporating learning effects during the early project stages.

Table 8.5 Activity durations, resource requirements and schedule information for the example project

ID	Allowable (d_i, r_{ik}) combinations	Selected (d_i, r_{ik}) combinations			Start time		
		S^O	S^L	S^R	S^O	S^L	S^R
Start	(0,0)	(0,0)	(0,0)	(0,0)	0	0	0
2	(3,9), (4,7), (5,6), (6,5), (7,4), (9,3)	(3,9)	(3.54,9)	(3.54,9)	0	0	0
3	(7,10), (8,9), (9,8), (10,7), (11,6), (13,5)	(7,10)	(7.38,10)	(7.38,10)	3	3.54	3.54
4	(6,9), (7,8), (8,7), (9,6), (11,5), (14,4)	(6,9)	(10.93,5)	(6.46,9)	10	10.92	10.92
5	(5,9), (6,8), (7,7), (8,6), (9,5), (12,4)	(5,9)	(9.18,5)	(5.51,9)	16	10.92	17.37
6	(3,8), (4,6), (5,5), (6,4), (8,3), (11,2)	(11,2)	(10.93,2)	(10.93,2)	21	20.10	22.88
7	(10,10), (11,9), (12,8), (13,7), (16,6), (19,5)	(12,8)	(11.78,8)	(11.78,8)	21	21.84	22.88
8	(9,10), (10,9), (11,8), (13,7), (15,6), (18,5)	(9,10)	(9.19,10)	(9.19,10)	33	33.63	34.67
9	(6,10), (7,9), (8,8), (9,7), (10,6), (12,5)	(6,10)	(6.46,10)	(6.46,10)	42	42.81	43.85
End	(0,0)	(0,0)	(0,0)	(0,0)	48	49.26	50.30

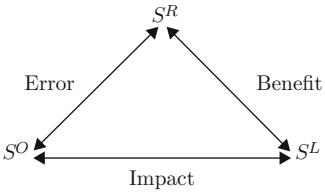


Fig. 8.13 Comparison of three project baseline schedules to measure the influence of activity learning

In order to analyze these three research questions, the authors have compared the three schedules S^O, S^L and S^R to each other, as graphically displayed in Fig. 8.13. The underlying idea of each comparison is briefly outlined along the following lines.

1. Impact of learning: The project baseline schedule without learning effects S^O and the baseline schedule with learning effects S^L are compared in order to investigate the *impact* of the introduction of learning effects during the project scheduling phase and to determine the driving variables of the differences between the project durations of both schedules.
2. Margin of error: The proposed schedule S^O is compared to the realistic schedule S^R in order to discover the potential *margin of error* made during project progress (S^R) when the learning effects have been ignored during the project scheduling phase (S^O) but observed afterwards during project progress. The smaller the deviation between both solutions is, the less important it is to spend time and effort to predict the learning effects in advance in order to incorporate them in the project schedule during baseline schedule construction.
3. Benefits of early knowledge of learning effects: The realistic schedule S^R is compared to the learning schedule S^L in order to measure the *benefits* that can possibly be made when learning effects are detected in early stages of the project progress. Indeed, the S^L takes all the learning effects into account when constructing a baseline schedule, and optimally assigns the teams to the activities. The S^R schedule, however, follows the timetable proposed by the S^O but observes longer or shorter activity durations due to learning. When learning is observed (S^R), the remaining part of the work yet to be done could be rescheduled taking learning into account (i.e. the remaining work is scheduled

according to the S^L scheduling approach). This rescheduling can be done under two scenarios. In a first scenario, it is implied that teams cannot be changed and hence only learning can be taken into account on the fixed activity durations. In a second scenario, it is assumed that teams can also be changed for the remaining work, leading to new duration/resource combinations with learning.

Research Results: The main results and conclusions of the study can be summarized as follows:

- Impact of learning on the project baseline schedule: In this computational experiment, the authors have investigated the impact of learning effects on the project schedule and its corresponding duration and searched for drivers of differences between baseline schedules with and without activity learning. The results can be briefly summarized as follows:
  - Learning rate: The degree of activity learning has a significant impact on the project duration. The faster the resources learn, the higher the project duration reductions in the S^L schedule relative to the S^O schedule. The experiments have also shown that low initial efficiencies of resources when starting to work on an activity can be quickly recuperated by the learning effects, leading to improvements up to 35% compared to the S^O schedule.
  - Network topology: Results have shown that activity learning has especially beneficial effects when many activities can be executed in parallel. In these cases, many activities are performed in parallel and have therefore more degrees of freedom to be scheduled with a relatively longer duration within the predefined project deadline. Obviously, when working longer on an activity, the benefits of learning can be fully exploited. The inverse is true for more serial networks where activities have on average shorter durations to guarantee a similar project duration, which prevents the resources to learn a lot within each activity execution.
  - Team assignment flexibility: When more duration/resource combinations can be chosen for each activity (see Table 8.5), there is more flexibility and room for project duration improvement when learning effects are incorporated. Consequently, the higher the number of allowable duration/resource combinations for the activities, the larger the differences are between the S^O and the S^L schedules.
- Margin of error during project progress: The S^O project schedule is assumed to be the baseline schedule proposed to the client and/or is used to set milestones without being aware that the learning effect will occur during project execution. However, efficiency improvements or deficiencies (i.e. learning) might occur during project progress, which affect the activity durations and the total project duration. This leads to a project progress captured by the S^R schedule, which might deviate from the original S^O baseline schedule. The test experiments have shown that prior information about the learning effects during the scheduling phase will often generate a competitive advantage in terms of the accuracy of

the project schedule and the promised project duration towards the client. As an illustration, tests have shown that only 7.53% of the project schedules has an absolute deviation smaller than 1%. Most projects (approximately 78%) have a deviation of more than 10%. These tests clearly have shown that the original baseline schedule S^O is often not an accurate prediction of the real project progress since learning effects will often lead to high deviations between the proposed project duration of S^O and the observed project duration given by the S^R schedule.

- Benefits of early knowledge of learning effects: The computational experiments have shown that a timely incorporation of learning effects leads to significant project duration reductions. It could be shown that the project duration reductions after rescheduling are significantly larger when changes in the duration/resource combinations are allowed, illustrating that changes in the team member assignments largely affect the efficiency gain which can be obtained. As an example, incorporating learning effects after a quarter of the project progress can lead to 55% of the maximum improvement when team member assignments are allowed to be modified. This value drops to approximately 37% and 13% of the maximum improvement, in case the incorporation of learning is done at respectively 50% and 75% of the project progress. The maximum improvement is calculated as the project duration reduction when learning effects are incorporated in advance (i.e. at 0% of the project progress).

8.5 Conclusions

This chapter extends the lessons learned from the resource-constrained project scheduling methods discussed in Chap. 7 to more advanced methods and model formulations in order to be used in more practical oriented settings.

First, three different scheduling objectives have been discussed in detail based on features taken from literature or detected in real-life projects. The presence of work continuity aims at the minimization of idle time of bottleneck resources and can often be used in projects with repeating activities. The use of quality dependent time slots is based on a real-life project in the biotechnology sector and assumes that activities can ideally be scheduled in certain time-slots and are subject to penalties (costs, detrimental effects, etc. . . .) outside these time slots. The resource availability cost problem is a well-known extension of the basic resource-constrained project scheduling problem (RCPSP, see Sect. 7.3.2) but assumes that the availability of resources is not fixed but can be set based on the cost of the resources.

In a second part of this chapter, some basic quantitative project descriptions have been presented to measure the topology of a project network and the scarceness of the resources used in the project and their relevance in practice is briefly shown.

Finally, three extra scheduling features have been described that allow to make the project schedule more realistic. The relaxation of the often strict activity assumptions has led to the formulation of various but related resource-constrained

project scheduling problems. The presence of setup times between activities and learning effects of the resources working on these activities has been investigated and their impact on the project duration has been shown based on different research studies.

In the next chapter, the resource-constrained project scheduling problem with work continuity constraints will be illustrated on a real-life tunnel construction project performed in the Netherlands.