6

Applications

The present chapter is concerned with applications of the concepts developed in Chapters 1 to 5 to production planning problems in the manufacturing and process industries, to the evaluation of investment projects, and to resource allocation problems that are subject to different kinds of uncertainty.

In Section 6.1 we discuss how scheduling problems arising in make-to-order assembly environments can be modelled as resource-constrained project scheduling problems. For different product structures, we consider the definition of appropriate minimum and maximum time lags ensuring a non-preemptive execution of overlapping operations.

Section 6.2 is devoted to a hierarchical three-stage approach to small-batch production planning using resource allocation methods from project management. The approach comprises the master production scheduling, multi-level lot sizing, and temporal plus capacity planning stages. At all levels, the scarcity of resources is taken into account, which differentiates this approach from most production planning and control systems used in practice. The lacking integration of capacity aspects is the essential reason for the generally poor performance of the latter systems.

When scheduling batch plants in the process industries, a variety of technological peculiarities have to be taken into account. In contrast to manufacturing, the batch processing times are mostly independent of the batch size and the intermediate products must be stocked in dedicated storage facilities. In addition, intermediate products may be perishable and to guarantee the purity of output products, the processing units have to be cleaned between the execution of certain operations. In Section 6.3 we deal with a two-phase method for production scheduling in the process industries, which decomposes the problem into a batching and a batch scheduling problem. For given primary requirements, the batching phase provides the numbers and sizes of the batches to be produced. Subsequently, the batches are scheduled on the processing units in the batch scheduling phase. The batching problem can be formulated as a mixed-integer linear program of polynomial size. By using the concepts of renewable and cumulative resources in combination with the

supplements from Chapter 5, the batch scheduling problem can be modelled as a resource-constrained project scheduling problem.

In practice, it is customary to evaluate investment projects based on the net present value criterion. The maximum net present value of a time-constrained investment project can, e.g., be computed by using the steepest descent method for convexifiable objective functions discussed in Chapter 3. In literature, however, it is commonly accepted that often the discount rate to be applied (i.e., the required rate of return) cannot be determined with sufficient accuracy. Moreover, the project deadline may be subject to negotiations between the investor and his customers. In Section 6.4 we show how using the steepest descent approach, the project net present value can be represented as a function of the discount rate and project deadline. On the basis of this function, investment projects with uncertain discount rate can be evaluated for a variable project deadline.

Throughout our previous discussion we have supposed that data such as activity durations, time lags, and resource requirements are deterministic quantities. Clearly, this is a simplifying assumption, which nevertheless is justified in many cases when the project data can be forecast reliably and small deviations from schedule do not seriously affect the execution of the project. Sometimes, however, the latter conditions are not met, in particular when coping with long-term projects like in the building industry or with production scheduling problems where machines and equipment may be subject to disruption. It is then expedient to take uncertainty into account already when scheduling the project or to adapt the schedule in a suitable fashion during its implementation. In Section 6.5 we propose two deterministic strategies for coping with uncertainty in project management. The anticipative approach consists in scheduling the project in a way that the impact of perturbations is minimized. Alternatively or additionally, one may use a reactive approach, where the project is rescheduled after each disruption and the objective is to minimize the changes with respect to the previous schedule.

6.1 Make-to-Order Production Scheduling

We consider the processing of a given set of customer orders in a multi-level make-to-order manufacturing environment, where no inventories are built up for future sale. At first, we recall some basic concepts from materials requirements planning (see, e.g., Nahmias 1997, Sect. 6.1). We assume that each final product consists of several subassemblies, which in turn may contain several components from lower production levels. Let P^f be the set of all final products ordered and let P be the set of all (intermediate or final) products l under consideration. Generally speaking, the product structure of a firm can be represented as a *gozinto graph* $G = (P, A, a)$ with node set P. Arc set A contains an arc (l, l') weighted by *input coefficient* $a_{ll'} \in \mathbb{N}$ if $a_{ll'}$ units of

product l are directly installed into one unit of product l'. P^f coincides with the set of all sinks of G.

Now let $x_l \in \mathbb{Z}_{\geq 0}$ denote the gross requirements for products $l \in P$. The gross requirements x_l for final products $l \in P^f$ are equal to the primary requirements d_l given by the customer orders. The gross requirements x_l for intermediate products l can easily be obtained by a *bill of materials explosion*, i.e., by solving the system of linear equations $x_l = d_l + \sum_{(l,l') \in A} a_{ll'} x_{l'}$ $(l \in P)$. Since there are no stocks available, the gross requirements x_l coincide with the amounts q_l of products l to be manufactured.

Each product $l \in P$ must be processed on machines of different types k in a prescribed order, which is given by the *process plan* of product l. Several identical machines of each type k (k-machines, for short) may be available. The processing of a batch of product l on a k-machine is referred to as an *operation*, which is denoted by kl. The execution of operation kl requires a (sequence-independent) *setup time* ϑ_{kl} during which the machine is occupied. For what follows we assume that no items of product l are needed for installing the machine. In addition, we suppose that the production is performed according to a *single-lot strategy*, i.e., all units of a product are processed in one batch of size x_l. The latter assumption is generally met in make-to-order production since each product is manufactured in response to a customer order, and splitting the batches would incur additional setup times without saving considerable holding cost. Hence, the *processing time* of operation kl is

$$p_{kl} = \vartheta_{kl} + x_l u_{kl} \tag{6.1}$$

where $u_{kl} \in \mathbb{N}$ is the unit processing time needed for producing one item of product l on a k-machine.

The make-to-order production scheduling problem consists in finding an operation schedule such that no two operations overlap in time on a machine, the operation sequences given by the process plans are observed, a sufficient amount of input products is available during the execution of each operation, and some objective function (e.g., the makespan) is minimized. In the following, we show how the production scheduling problem can be modelled as a resource-constrained project scheduling problem with renewable and cumulative resources. The model is based on the previous work by Günther (1992) and Neumann and Schwindt (1997).

For each operation kl we introduce one real activity, also denoted by kl, whose duration p_{kl} is given by (6.1). A machine type k is identified with a renewable resource $k \in \mathcal{R}^\rho$. Resource capacity R_k equals the number of k-machines available. Each activity kl requires one unit of resource k.

Project network N is obtained by exploding each node $l \in P$ of gozinto graph G into the respective (directed) path from the initial operation kl to the terminal operation $k'l$ in the process plan of product l (see Figure 6.1). The arcs $(l, l') \in A$ are then replaced by arcs $(k'l, k''l')$ linking the terminal operation $k'l$ of l with the initial operation $k''l'$ of l'. Moreover, all initial

operations of products at the lowest production level are connected with the project beginning event 0, and the terminal operations of the final products are connected with the project termination event $n+1$. Finally, backward arc $(n+1, 0)$ is added.

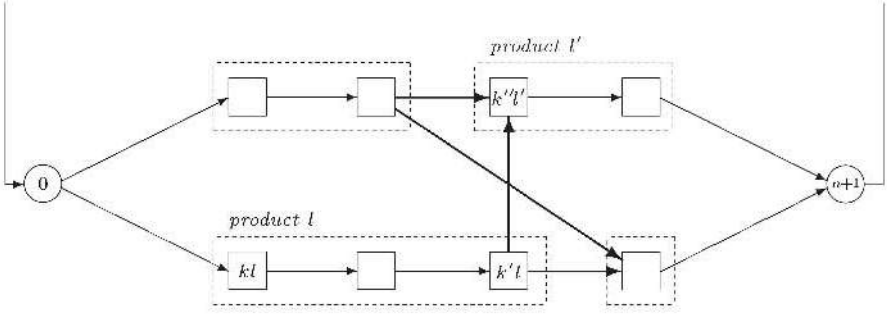

Fig. 6.1. Project network without arc weights arising from gozinto graph

We proceed by assigning weights $\delta_{kl,k'l'}$ to the arcs $(kl, k'l')$ of N. The arcs emanating from node 0 are weighted with 0 and the arcs terminating at node $n+1$ are weighted with the duration of the respective initial node. The weight $\delta_{n+1,0} = -\overline{d}$ is chosen to be the negative maximum makespan allowed. Now let kl and $k'l$ be two consecutive operations in the process plan of product l. At first, we consider the case where $u_{kl} \leq u_{k'l}$, which is depicted in Figure 6.2. Clearly, we may start the execution of operation $k'l$ when the preceding operation kl has been completed. From Figure 6.2 it can be seen, however, that much time can be saved if we allow for *overlapping operations*. The processing of the first item of product l on the k'-machine can then be started as soon as the first item on the k-machine has been completed without causing any idle time on the k'-machine. Hence, instead of adding a precedence constraint between kl and $k'l$, we introduce a time lag of $\delta_{kl,k'l} = \vartheta_{kl} + u_{kl} - \vartheta_{k'l} < p_{kl}$ units of time between the starts of operations kl and $k'l$. The time lag ensures that at any point in time where $k'l$ is in progress, a sufficient amount of product l has already been processed on the k-machine. Note that, as shown in Figure 6.2, time lag $\delta_{kl,k'l}$ may even become negative, in which case we have a maximum time lag of $-\delta_{kl,k'l}$ units of time between operations $k'l$ and kl.

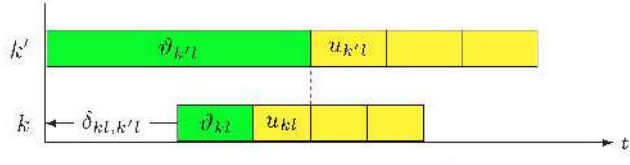

Fig. 6.2. Overlapping operations kl and $k'l$ with $u_{kl} \leq u_{k'l}$

The case where $u_{kl} > u_{k'l}$ is illustrated in Figure 6.3. Here, starting operation $k'l$ at the completion of the first item of product l on the k-machine would mean that after the processing of the first item on the k'-machine, the required second item from the k-machine is not finished. Thus, we synchronize both operations in a way that the *last* item on the k'-machine is processed after the completion of operation kl, i.e., $\delta_{kl,k'l} = \vartheta_{kl} + x_l u_{kl} - (x_l - 1)u_{k'l} - \vartheta_{k'l}$.

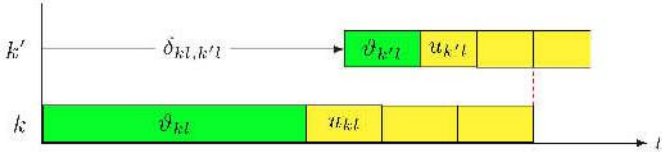

Fig. 6.3. Overlapping operations kl and $k'l$ with $u_{kl} > u_{k'l}$

In sum, between two consecutive operations kl and $k'l$ belonging to one and the same product $l \in P$, we introduce the time lag

$$\delta_{kl,k'l} = \begin{cases} \vartheta_{kl} + u_{kl} - \vartheta_{k'l}, & \text{if } u_{kl} \leq u_{k'l} \\ \vartheta_{kl} + x_l u_{kl} - (x_l - 1)u_{k'l} - \vartheta_{k'l}, & \text{otherwise} \end{cases} \tag{6.2}$$

which is the smallest lapse of time that guarantees that operation $k'l$ need not be interrupted because no items are available.

In practice, it is often expedient to transfer items in batches from one machine to another. The *transportation lot size* $y_l \in \mathbb{N}$ for product $l \in P$ is then specified by the size of pallets or containers used for the transport of l. In addition, we suppose all machines of a given type k to be grouped in a k-shop, where $t_{kk'} \in \mathbb{Z}_{\geq 0}$ denotes the *transfer time* from the k- to the k'-shop (implicitly, we have supposed until now that $y_l = 1$ and $t_{kk'} = 0$ for all products l and all machine types k, k'). Formula (6.2) can easily be adapted to the case of general transportation lot sizes and transfer times by noting that the items now arrive at the k'-shop in transfer batches of size y_l. If $u_{kl} \leq u_{k'l}$, this means that the first batch is conveyed $\vartheta_{kl} + y_l u_{kl}$ units of time after the start of kl, whereas for $u_{kl} > u_{k'l}$, y_l items of product l remain to be processed on the k'-machine after the completion of kl. In both cases, the respective transfer time $t_{kk'}$ must be included. We then obtain the following formula for the time lag $\delta_{kl,k'l}$ between consecutive operations:

$$\delta_{kl,k'l} = \begin{cases} \vartheta_{kl} + y_l u_{kl} - \vartheta_{k'l} + t_{kk'}, & \text{if } u_{kl} \leq u_{k'l} \\ \vartheta_{kl} + x_l u_{kl} - (x_l - y_l)u_{k'l} - \vartheta_{k'l} + t_{kk'}, & \text{otherwise} \end{cases} \tag{6.3}$$

Note that $y_l = x_l$ corresponds to nonoverlapping product processing, where (6.3) provides the same value for both cases $u_{kl} \leq u_{k'l}$ and $u_{kl} > u_{k'l}$.

Next, we consider the transition from the terminal operation kl in the process plan of a product l to the initial operation $k'l'$ in the process plan of a succeeding product l' with $(l, l') \in A$. We assume that l' is the only

product containing items of product l. In particular, the latter assumption is always fulfilled if the product structure is linear or convergent, i.e., if the gozinto graph G is an intree. The case of *common parts*, which are installed into different products l', is studied below. To simplify writing, we establish the convention that $y_l/a_{ll'}$ is integral, which means that all items of product l needed for the production of one unit of product l' are transferred at the same time. The first item of product l' cannot be processed before $a_{ll'}$ items of product l have been completed on the k-machine. If the time $a_{ll'}u_{kl}$ needed for producing $a_{ll'}$ items of l is less than or equal to unit processing time $u_{k'l'}$, we can start the processing of l' as soon as the first transfer batch of product l has been conveyed from the k- to the k'-shop. Otherwise, we start the processing of the last $y_l/a_{ll'}$ items of product l' on machine k' after the last transfer of y_l items of product l from the k- to the k'-shop. Hence, the time lag $\delta_{kl,k'l'}$ between terminal operation kl and initial operation $k'l'$ is chosen to be

$$\delta_{kl,k'l'} = \begin{cases} \vartheta_{kl} + y_l u_{kl} - \vartheta_{k'l'} + t_{kk'}, & \text{if } a_{ll'}u_{kl} \leq u_{k'l} \\ \vartheta_{kl} + a_{ll'}x_{l'}u_{kl} - (x_{l'} - \frac{y_l}{a_{ll'}})u_{k'l'} - \vartheta_{k'l'} + t_{kk'}, & \text{otherwise} \end{cases} \tag{6.4}$$

Note that formula (6.3) may be interpreted as the special case where $l' = l$ and $a_{ll'} := 1$.

We now turn to general product structures containing common parts $l \in P$. The presence of common parts leads to an *assignment sequence problem*, where we have to decide on the order in which completed items of product l are allotted to succeeding products l'. For a given assignment sequence, appropriate time lags may then be computed in analogy to the case of a convergent product structure. For details we refer to Neumann and Schwindt (1997). In the latter reference, a procedure for finding a suitable *block-structured* assignment sequence has been devised, where all items allotted to one and the same product l' are processed consecutively. For that case, time lags $\delta_{kl,k'l'}$ can again be written in closed form.

Alternatively, common parts can be dealt with by introducing cumulative resources. This approach, which has not been considered by Neumann and Schwindt (1997), offers the prospect of being independent of an assignment sequence to be specified in advance. Let $l \in P$ be some common part. We again consider the case where all items of l being assigned to some product are processed one after another, and for simplicity we assume that l is installed into two products, say, l' and l''. At first, we identify product l with a cumulative resource $l \in \mathcal{R}^\gamma$ with zero safety stock $\underline{R}_l$ and infinite storage capacity $\overline{R}_l$. We then decompose a copy of operation kl into two auxiliary operations kl' and kl'' with durations $p_{kl'} = a_{ll'}x_{l'}u_{kl}$ and $p_{kl''} = a_{ll''}x_{l''}u_{kl}$ to be executed on the same fictitious k-machine (which must be represented by a separate renewable resource $\hat{k}$ with capacity $R_{\hat{k}} = 1$). To ensure that after the setup of the k-machine, operations kl' and kl'' are processed in parallel with operation kl, we add the time lags $\delta_{kl,kl'} = \delta_{kl,kl''} = \vartheta_{kl}$, $\delta_{kl',kl} = p_{kl'} - p_{kl}$, and $\delta_{kl'',kl} = p_{kl''} - p_{kl}$. Because kl' and kl'' cannot overlap, they must be

processed consecutively without any delay in between. The *start* events of both operations kl' and kl'' replenish the cumulative resource by $a_{ll'}x_{l'}$ and $a_{ll''}x_{l''}$ units, and the start events of initial operations $k'l'$ and $k''l''$ in the process plans of products l' and l'' deplete the inventory of l by $a_{ll'}x_{l'}$ and $a_{ll''}x_{l''}$ units. Eventually, we introduce time lags $\delta_{kl',k'l'}$ and $\delta_{kl'',k''l''}$ of type (6.4) between the auxiliary operations kl' and kl'' and the respective initial operations $k'l'$ and $k''l''$, which guarantee that a sufficient amount of product l is available when starting operations $k'l'$ and $k''l''$.

6.2 Small-Batch Production Planning in Manufacturing Industries

In this section we review a capacity-oriented hierarchical planning method for small-batch multi-level production planning in manufacturing industries, which has been proposed by Neumann and Schwindt (1998). An earlier version of this approach is described in Franck et al. (1997). We consider the three planning stages *capacitated master production scheduling*, *multi-level lot sizing*, and *temporal plus capacity planning* (in the original paper, an additional *fine planning* stage has been included). The optimization problems arising at the capacitated master production scheduling and temporal plus capacity planning stages can be formulated as resource-constrained project scheduling problems. Alternative approaches to hierarchical production planning have, e.g., been devised by Carravilla and de Sousa (1995), Schneeweiß (1995), Drexl and Kolisch (1996), Schneeweiß (2003), Ch. 6, and Kolisch (2001*b*), Ch. 4. Elements of capacity-oriented production planning and control systems have been discussed in Drexl et al. (1994).

At the stage of **capacitated master production scheduling**, a *master production schedule* (MPS) has to be determined, which translates the primary requirements for final products into monthly production orders for final products and main components such that the workload of work centers is as smooth as possible over time. An even utilization of the work centers helps to avoid expensive capacity adjustment measures and facilitates the determination of feasible solutions at subsequent planning stages, where explicit resource constraints have to be taken into account. The planning horizon of this first stage is usually about one year comprising twelve periods of one month each.

For the final products the amounts to be produced and corresponding month-precise delivery dates are given by the customer orders. We assume that all customer orders must be met on time. From the order quantities of final products and the product structure of the company, the gross requirements for main components at lower production levels can be computed by a bill of materials explosion. To obtain the net requirements, we subtract the corresponding available stocks.

To schedule the production of the final products and main components (referred to as *main products* in what follows), we model the problem of determining an appropriate MPS as a *resource levelling problem* with, e.g., the total squared utilization cost as objective function. To this end, we first define a project with renewable resources for each individual customer order. The production of the net requirement for each main product i of such a customer order is regarded as an activity i of the project. The duration p_i of activity i results from summing up the setup and processing times for product i and the components of product i at lower production levels. To obtain the minimum time lag d_{ij}^{min} between the start of activity i and the start of any subsequent activity j in the product structure, some buffer for waiting times arising when scheduling the components of all production levels has to be added to p_i. This time buffer can be estimated by using concepts from queueing theory (see Söhner 1995, Ch. 3). The renewable resources required for carrying out the activities of the project coincide with the respective work centers involved. The resource requirements of product i are assumed to be distributed uniformly over the execution time p_i of activity i.

The project networks for all customer orders are then joined together to make a *multi-project network* by adding the project beginning and termination nodes 0 and $n+1$ and connecting nodes 0 and $n+1$ with all initial and terminal activities, respectively, of the individual project networks. The backward arc $(n+1, 0)$ corresponding to the project deadline $\overline{d}$ is weighted by $-\overline{d} = -\Delta$, where Δ denotes the planning horizon (typically about one year). A delivery date $\overline{d}_i$ for some product i can be modelled by a maximum time lag $d_{0i}^{max} = \overline{d}_i - p_i$ between the project start and the start of activity i.

The objective function of the resource levelling problem can be chosen to be any of the objective functions dealt with in Subsection 2.3.2. A solution S to the resource levelling problem provides month-precise milestones for the production of the gross requirements for the main products.

At the stage of **multi-level lot sizing**, the main products are decomposed into intermediate products for which weekly production quantities (also called *lots* or *batches*) are computed. In the lot sizing model, the planning horizon of roughly three months is divided into periods of one week each. The production orders for the main products, which define the primary requirements of the lot sizing model, are given by the MPS.

The production of the intermediate products requires several resources. Each resource corresponds to a group of machines. The processing of a product on a resource necessitates a setup of the resource, which takes a setup time and incurs a setup cost. Additional costs arise from stocking products. Setup and processing times are given in time units (for example, hours). For a given resource, the *aggregate* per-period availability corresponds to the workload in time units which can be executed by the machines of the corresponding group within one period. The objective is to determine lots for the intermediate products such that no backlogging occurs, the per-period availabilities of all resources are observed in all periods, and the sum of setup and inventory

holding costs is minimized. This problem represents a *multi-level capacitated lot sizing problem*, for which Tempelmeier and Derstroff (1996) have developed the following Lagrangean-based heuristic. By relaxing the inventory balance and capacity constraints, a decomposition of the original problem into several single-level uncapacitated lot sizing problems of the classical Wagner-Whitin type is obtained, which can be solved efficiently by dynamic programming (cf. Wagelmans et al. 1992). Violations of the relaxed constraints are taken into consideration via a Lagrangean penalty function, whose multipliers are iteratively updated in the course of a subgradient optimization procedure.

Intermediate products may be further broken down into individual components. At the stage of **temporal plus capacity planning**, the production of those components has to be scheduled on groups of identical machines for each week (period of the lot sizing stage). The weekly gross requirements for the individual components can be found by a bill of materials explosion from the lots for intermediate products computed at the lot sizing stage. Since all lots have to be processed within one week, we aim at minimizing the maximum completion time of all operations, i.e., the makespan. As has been shown in Section 6.1, the latter production scheduling problem can be modelled as a project duration problem with renewable and cumulative resources.

Since at the lot sizing stage, only aggregate per-period capacities of resources have been taken into account, it may happen that the makespan found at the temporal plus capacity planning stage exceeds the deadline of one week. In that case, we have to re-perform lot sizing such that the size of at least one lot is reduced. This can be achieved by decreasing the aggregate capacity of resources whose capacity has been violated, which corresponds to a feedback mechanism originally proposed by Lambrecht and Vanderveken (1979) for the special case of a job shop environment.

6.3 Production Scheduling in the Process Industries

In this section we are concerned with production scheduling in the process industries, where similarly to the case of manufacturing dealt with in Section 6.1, final products arise from several successive transformations of intermediate products. In contrast to manufacturing, however, where a limited number of piece goods are processed on machines, in the process industries the transformations are performed by chemical reactions of bulk goods, liquids, or gases on *processing units* such as reactors, heaters, or filters. The transformation of input products into output products on a dedicated processing unit is called a *task*. Each task may consume several input products and may produce several output products, whose amounts may be chosen within prescribed bounds. Perishable products must be consumed in the space of a given shelf life time, which may be equal to zero. In the latter case, the intermediate product cannot be stocked. In addition, the storable intermediate products must be stocked in dedicated *storage facilities* like tanks or silos.

That is why storage problems play an important role in the process industries (see, e.g., Schwindt and Trautmann 2002). Further peculiarities encountered in the process industries are cyclic product structures, sequence-dependent cleaning times on processing units, and large processing times, which may necessitate the explicit consideration of breaks like night-shifts or weekends.

Throughout this section we assume that the production is operated in *batch mode*, which means that at the beginning of a task, the input products are loaded into the processing unit, and the output becomes available at the termination of the task. The case of continuous production mode can be dealt with by using the concept of continuous cumulative resources introduced in Section 5.4. As a rule, the production is organized according to batch mode if small amounts of a large number of final products are required (whereas the continuous production mode is typical of basic materials industry such as oil or dyestuff industries). The combination of a task and the corresponding quantity produced is called a *batch*. An *operation* corresponds to the processing of a batch. Since the batch sizes are limited by the capacity of the processing units, a task may be performed more than once, resulting in several corresponding operations. In contrast to manufacturing, the processing times of operations are generally independent of the respective batch sizes.

The production scheduling problem to be dealt with consists in allocating processing units and storage facilities over time to the production of given primary requirements such that all operations are completed within a minimum *makespan*. This objective is particularly important in batch production, where often a large number of different products are processed on multi-purpose equipment (cf. Blömer and Günther 1998). In this case, the production plant is configured according to the set of production orders released. Before processing the next set of production orders, the plant has generally to be rearranged, which requires the completion of all operations.

There is an extensive literature dealing with production scheduling in the process industries. Most of the solution approaches discussed are based on time-indexed or continuous-time mixed-integer programming formulations of the problem, cf. e.g., Kondili et al. (1993), Pinto and Grossmann (1995), Blömer and Günther (1998, 2000), or Burkard et al. (1998). For a detailed review of literature, we refer to Blömer (1999), Sect. 4.2, and Schwindt and Trautmann (2000).

The special feature of the approach by Neumann et al. (2001), which we shall discuss in what follows, is the decomposition of the production scheduling problem into a *batching* and a *batch scheduling* problem. A similar technique has been used by Brucker and Hurink (2000) for solving a related production scheduling planning problem. This decomposition offers the prospect of markedly decreasing the severe computational requirements incurred by solving the entire production scheduling problem at once. The batching phase generates appropriate batches, which in the course of the batch scheduling phase are subsequently scheduled on the processing units subject to inventory constraints. The batching problem can be formulated as a mixed-integer lin-

ear program. The batch scheduling problem can be viewed as a multi-mode resource-constrained project scheduling problem with renewable and cumulative resources, sequence-dependent changeover times, and calendars.

We first deal with the **batching problem**. Batching converts the given primary requirements for final products into individual batches for tasks, where the objective is to minimize the workload, i.e., the total amount of work to be performed on the processing units. For each task we determine a collection of batches such that all primary requirements can be satisfied, there is sufficient capacity for stocking the residual inventories after the completion of all operations, the prescribed bounds on the batch sizes are observed, and the workload to be processed is minimum.

We are going to formulate the batching problem as a mixed-integer linear program (see Schwindt 2001 and Neumann et al. 2002). Let T be the set of all tasks s, and let U be the set of all processing units k. $U_s \subseteq U$ is the set of all processing units on which task s can be executed. By p_{ks} we designate the processing time of task s on processing unit $k \in U_s$. The mean processing time of task s on any processing unit $k \in U_s$ is $\overline{p}_s = \sum_{k \in U_s} p_{ks}/|U_s|$, and $\nu_s = \lceil \sum_{k \in U_s} \overline{d}/p_{ks} \rceil$ is an upper bound on the number of batches for task s which can be executed in the planning period $[0, \overline{d}]$. For each task $s \in T$, a lower bound $\underline{q}_s$ and an upper bound $\overline{q}_s$ on the batch size are given. The lower bound generally arises from technological or economical requirements, whereas the upper bound equals the capacity of the respective processing units.

By P we again denote the set of all products l to be produced, and d_l is the primary requirement for product l. Each storable product $l \in P$ is stocked in a dedicated storage facility of capacity c_l. For simplicity we assume that there are no initial stocks of products l, that a sufficient amount of raw materials is available, and that no safety stocks have to be taken into account. Each product $l \in P$ arises as output of some tasks $s \in T$, and each intermediate product $l \in P$ is also input to some other tasks $s' \in T$. The analogue to the input coefficients in manufacturing are the input and output proportions $-1 \leq a_{ls} \leq 1$, which provide the proportions of products l in the input or output, respectively, of task s. We have $a_{ls} < 0$ if l is an input product of s and $a_{ls} > 0$ if l is an output product of task s. For products l that are neither consumed nor produced by task s, we set $a_{ls} := 0$. For what follows we assume that the proportions a_{ls} cannot be varied (Neumann et al. 2002 have considered the general case of flexible input and output proportions).

The batching problem can now be formulated by introducing, for each task $s \in T$, ν_s continuous variables $q_s^\mu \geq 0$ ($\mu = 1, \ldots, \nu_s$) with the following meaning. If the number of batches for task s is greater than or equal to μ, q_s^μ provides the size of the μ-th batch and $q_s^\mu = 0$, otherwise. In addition, we need binary variables x_s^μ with $x_s^{\mu+1} \leq x_s^\mu$ ($\mu = 1, \ldots, \nu_s - 1$), where $x_s^\mu = 1$ indicates that there exists a μ-th batch for task s and $x_s^\mu = 0$, otherwise. The total workload to be processed then equals $\sum_{s \in T} \overline{p}_s \sum_{\mu=1}^{\nu_s} x_s^\mu$ (recall that the processing time of a batch is independent of the batch size). The linking between

variables q_s^μ and x_s^μ can be achieved by the inequalities $q_s^\mu/\overline{q}_s \leq x_s^\mu \leq q_s^\mu/\underline{q}_s$, which at the same time ensure that the batch sizes are between the lower and upper bounds $\underline{q}_s$ and $\overline{q}_s$.

$a_{ls}q_s^\mu$ is the increase in the inventory of product l after one execution of task s (which is negative if l is an input product of s). The quantity of product l remaining on stock after the execution of all batches equals $\sum_{s\in T} a_{ls} \sum_{\mu=1}^{\nu_s} q_s^\mu$, which must not be less than the primary requirements d_l for product l. On the other hand, the residual amount of product l after the delivery of the demands must not exceed the storage capacity c_l for product l.

In sum, the batching problem can be stated as the following mixed-integer linear program:

$$\left.\begin{array}{lll} \text{Minimize} & \displaystyle\sum_{s\in T} \overline{p}_s \sum_{\mu=1}^{\nu_s} x_s^\mu & \\ \text{subject to} & d_l \leq \displaystyle\sum_{s\in T} a_{ls} \sum_{\mu=1}^{\nu_s} q_s^\mu \leq d_l + c_l & (l \in P) \\ & q_s^\mu/\overline{q}_s \leq x_s^\mu \leq q_s^\mu/\underline{q}_s & (s \in T,\ \mu = 1,\ldots,\nu_s) \\ & x_s^{\mu+1} \leq x_s^\mu & (s \in T,\ \mu = 1,\ldots,\nu_s - 1) \\ & x_s^\mu \in \{0,1\} & (s \in T,\ \mu = 1,\ldots,\nu_s) \\ & q_s^\mu \geq 0 & (s \in T,\ \mu = 1,\ldots,\nu_s) \end{array}\right\} \quad (6.5)$$

A feasible solution (q, x) to batching problem (6.5) provides a set of operations to be scheduled on the processing units. For each task $s \in T$, we have $\sum_{\mu=1}^{\nu_s} x_s^\mu$ corresponding operations.

We now turn to the **batch scheduling problem**, which consists in allocating the resources to the operations over time such that the processing of all batches is completed within a minimum amount of time, i.e., the makespan is minimized. A variety of technological and organizational constraints have to be taken into account. A task generally requires different types of resources: processing units with sequence-dependent cleaning times, input products, and storage facilities for output products. The availability of these resources is limited by capacities and inventories. Break calendars specify time intervals during which specific tasks cannot be processed. Certain tasks can be suspended during a break (e.g., packaging), whereas other tasks (e.g., chemical reactions) cannot be interrupted at all. Some tasks may be executed on alternative processing units differing in speed and cleaning times. Finally, there may be perishable intermediate products, which cannot be stored. In what follows we develop a resource-constrained project scheduling model for the batch scheduling problem, which has been discussed in Neumann et al. (2003*b*), Sect. 2.16 (see also Schwindt and Trautmann 2000 and Neumann et al. 2002, who have proposed similar models for batch scheduling).

Analogously to the case of make-to-order production dealt with in Section 6.1, the execution of all operations can be viewed as a project, where

the makespan to be minimized corresponds to the project duration S_{n+1}. For each operation we introduce one real activity $i \in V^a$. The activity durations p_i are equal to the processing times of the corresponding tasks. In addition, we introduce two events $g, h \in V^e$ for each operation i, representing the start and the completion of i. Minimum and maximum time lags $d_{ig}^{min} = d_{ig}^{max} = 0$ and $d_{ih}^{min} = d_{ih}^{max} = p_i$ ensure that g occurs at the start and h at the completion of i.

Each operation is executed on a processing unit. We combine identical processing units to form a pool. Each pool is modelled as a renewable resource $k \in \mathcal{R}^\rho$. Processing units are identical if they can operate the same tasks with the same processing and cleaning times. The requirement r_{ik} of activity i for resource k equals 1 if operation i is carried out on a processing unit of pool k and 0, otherwise. The resource capacity R_k is equal to the number of processing units in the corresponding pool.

The cleaning times between consecutive operations on a processing unit can be modelled by introducing sequence-dependent changeover times between the activities (cf. Section 5.2). The changeover time ϑ_{ij}^k between two activities i and j on renewable resource $k \in \mathcal{R}^\rho$ equals the cleaning time after operation i if j requires a cleaning of resource k. When checking the changeover-feasibility of some schedule S, the lower capacities of all arcs in the flow network equal 0 or 1 because $r_{ik} = 1$ holds for all activities i requiring resource k. Hence, the corresponding minimum-flow problem can be solved in $\mathcal{O}(n|\theta^k(S)|)$ time by augmenting path algorithms (cf. Ahuja et al. 1993, Sect. 6.5).

Certain operations cannot be in progress during breaks. We model breaks by introducing an activity calendar b_i for each real activity $i \in V^a$ (cf. Section 5.1). If operation i cannot be processed during breaks, $b_i(t) = 0$ exactly if time t falls into a break. For the remaining activities $i \in V^a$, we have $b_i(t) = 1$ for all $t \in [0, \overline{d}]$.

Some tasks $s \in T$ can be executed on alternative processing units $k \in U_s$ belonging to different pools. For each corresponding activity i, we introduce one execution mode m_i for each alternative processing unit operation i can be executed on (cf. Section 5.3). The requirements for renewable resources as well as the durations and changeover times then refer to individual execution modes instead of activities.

Intermediate storage facilities can be modelled as (discrete) cumulative resources. We identify each intermediate product l to be stocked with one cumulative resource $l \in \mathcal{R}^\gamma$ with safety stock $\underline{R}_l = 0$ and storage capacity $\overline{R}_l = c_l$. The requirements of start and completion events $g, h \in V^e$ of operations i for resource l can be determined as follows. Assume that operation i corresponds to the μ-th execution of task s. If l is an input product of task s, i.e., $a_{ls} < 0$, then $r_{gl} = a_{ls} q_s^\mu$. If l is an output product of s, i.e. $a_{ls} > 0$, we have $r_{hl} = a_{ls} q_s^\mu$. Note that the integrality of resource requirements $r_{gl}, r_{hl} \in \mathbb{Z}$ may necessitate a subsequent scaling of all requirements and storage capacities by some factor $c \in \mathbb{Q}$, which does not affect the time complexities of the solution algorithms discussed.

Finally, we turn to perishable intermediate products. We only consider the case where a perishable product must be consumed immediately. The case of general shelf life times can be modelled by introducing auxiliary events and cumulative resources (see Schwindt and Trautmann 2002). Let l be a perishable output product produced by some operation i. Then there must exist some operation j that immediately consumes the amount of l arising at the completion of operation i. This can be ensured by introducing a minimum and a maximum time lag $d_{ij}^{min} = d_{ij}^{max} = p_i$ pulling the start of j to the completion of i, provided that there is a one-to-one correspondence between operations producing and consuming perishable products. The latter requirement can easily be integrated into the batching problem and is generally met in practice because otherwise small deviations of the realized from the predicted processing times would most often imply the loss of perishable substances. If the condition is not met, the immediate consumption of a perishable intermediate product can be enforced by introducing a corresponding cumulative resource l with $\underline{R}_l = \overline{R}_l = 0$.

Table 6.1, which is taken from Neumann et al. (2003*b*), Sect. 2.16, summarizes the input data of a batch scheduling problem and their respective counterparts in the resource-constrained project scheduling model.

Table 6.1. Batch scheduling vs. project scheduling

Batch scheduling	Project scheduling
Operations	Activities
Makespan	Project duration
Pools of identical processing units	Renewable resources
Cleaning times	Sequence-dependent changeover times
Breaks	Activity calendars
Alternative processing units	Multiple execution modes
Intermediate storage facilities	Cumulative resources
Perishable intermediate products	Minimum and maximum time lags, cumulative resources

Based on the above decomposition of the production scheduling problem into batching and batch scheduling, Schwindt and Trautmann (2000) have been able to provide a feasible solution to a benchmark problem from industry submitted by Westenberger and Kallrath (1995) for the first time (see also Kallrath 2002). The latter case study covers most of the features occurring in the production scheduling problem of batch plants. Neumann et al. (2002) have shown that the decomposition approach also compares favorably with monolithic time-indexed mixed-integer linear programming formulations of the problem.

6.4 Evaluation of Investment Projects

In this section we discuss a parametric optimization procedure, which has been proposed by Schwindt and Zimmermann (2002) for evaluating investment projects with respect to different project deadline and discount rate scenarios (see also Zimmermann and Schwindt 2002). Project managers are frequently confronted with the problem to decide whether some given investment project should be performed or to select one out of several mutually exclusive investment projects from a given portfolio. For the assessment of investments, the net present value criterion is well-established in research and practice (see, e.g., Brealey and Myers 2002, Ch. 5). In classical preinvestment analysis, investments are specified by a stream of payments, i.e., a series of payments with associated payment times. Given a stream of payments and a proper discount rate, the net present value of the project is obtained by summing up all payments discounted to the project beginning (case (a) in Figure 6.4, where exogenous parameters are written in italics).

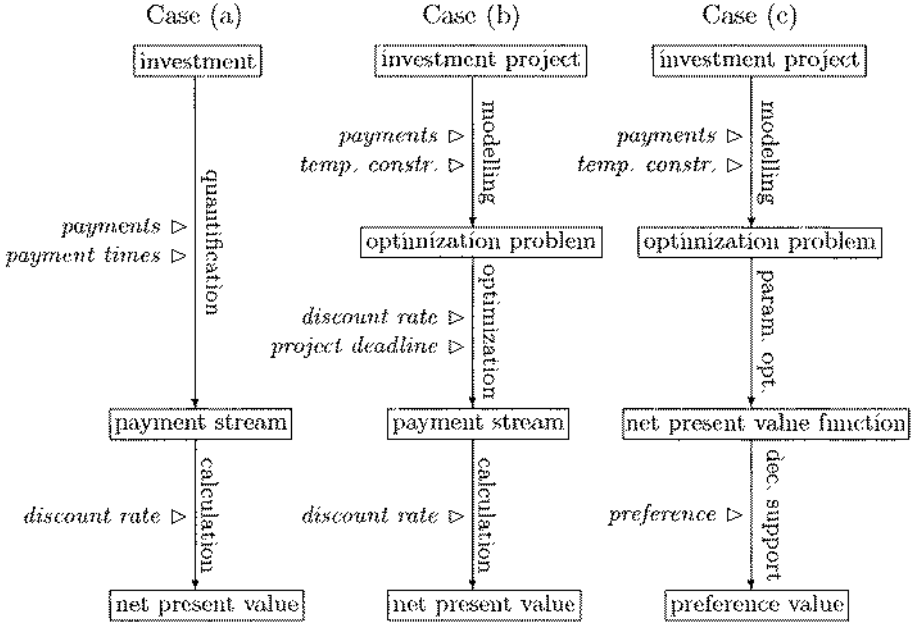

Fig. 6.4. Evaluation of investments and investment projects

In case of *investment projects*, the payment times are no longer given in advance but are subject to optimization. An investment project consists of a set of events each of which is associated with a payment. Moreover, there are prescribed minimum and maximum time lags between the occurrence of events. Thus, the stream of payments results from maximizing the net present

value of the project subject to the temporal constraints that are given by the minimum and maximum time lags (case (b) in Figure 6.4).

The formulation of the latter optimization problem presupposes the knowledge of the *required rate of return* (i.e., the discount rate) for discounting the payments and the specification of a *maximum project duration* (i.e., the project deadline). When dealing with real investments in material goods like in the building industry, however, often neither is the proper discount rate to be applied known with sufficient accuracy nor is the project deadline fixed when the investment project must be evaluated. The required rate of return is a theoretical quantity and can only be estimated (see Brealey and Myers 2002, Ch. 23). The project deadline generally arises from negotiations between the investor performing the project and his customers. The *parametric optimization approach* by Schwindt and Zimmermann (2002) provides the maximum project net present value as a function of the discount rate and project deadline chosen. The resulting net present value curve can then serve as a basis for the decision of the investor, which depends on his individual risk preference (case (c) in Figure 6.4).

Let V^e be the set of project events, including the project beginning 0 and the project termination $n+1$, the start events of project activities, and milestones at the completion of subprojects. The project events and the corresponding prescribed time lags among them can be represented by an *event-on-node network* $N = (V^e, E, \delta)$ with node set V^e, arc set E, and arc weights δ_{ij} for $(i,j) \in E$ (see Subsection 1.1.2). Each event $i \in V^e$ belonging to an activity start is associated with a (negative) disbursement $c_i^f < 0$ for bought-in supplies or outside services. Progress payments $c_j^f > 0$ arise when subprojects with milestones $j \in V^e$ have been finished. Progress payments generally refer to the direct cost which is incurred by the activities of the corresponding subproject (cf. Daynand and Padman 1997).

Given a discount rate $\alpha > 0$ and a project deadline $\overline{d}$, the *time-constrained net present value problem* reads as follows, where the project deadline $\overline{d}$ is specified by arc $(n+1,0) \in E$ with weight $\delta_{n+1,0} = -\overline{d}$:

$$\left.\begin{array}{ll} \text{Maximize} & C^\alpha(S) := \displaystyle\sum_{i \in V^e} c_i^f e^{-\alpha S_i} \\ \text{subject to} & S_j - S_i \geq \delta_{ij} \quad ((i,j) \in E) \\ & S_0 = 0 \end{array}\right\} \tag{6.6}$$

Let $S_T^{\overline{d}}$ denote the feasible region of problem (6.6) for project deadline $\overline{d}$ and let C^* be the corresponding optimal objective function value. A time-feasible schedule S with maximum net present value $C^\alpha(S) = C^*$ can be determined by the steepest descent method from Subsection 3.2.2, where $f(S) = -C^\alpha(S)$.

Until now we have assumed that discount rate α and project deadline $\overline{d}$ are exogenous parameters. As we have mentioned above, however, the proper discount rate α can only be estimated and the project deadline $\overline{d}$ may be subject to negotiations. Thus, for an adequate evaluation of the investment

project it is necessary to know the project net present value for a range of relevant values of α and $\overline{d}$. In the sequel, we describe a parametric optimization procedure that determines the maximum project net present value C^* as a function of discount rate α and project deadline $\overline{d}$. This algorithm extends a method by Grinold (1972), who has studied the dependency between C^* and the project deadline $\overline{d}$. Clearly, since $\mathcal{S}_T^{\overline{d}'} \supseteq \mathcal{S}_T^{\overline{d}}$ if $\overline{d}' \geq \overline{d}$, C^* is nondecreasing in $\overline{d}$.

The following considerations are based on two basic observations that derive from the study of schedule sets and objective functions in Sections 2.2 and 2.3. First, since the net present value objective function is linearizable, there always exists an optimal solution S to the time-constrained net present value problem (6.6) that is a vertex of the feasible region $\mathcal{S}_T^{\overline{d}}$ of (6.6). Second, each vertex S of $\mathcal{S}_T^{\overline{d}}$ can be represented by a spanning tree $G = (V^e, E_G)$ of project network N.

The basic idea for computing *net present value function*

$$C^* : [0, \infty[\times]ES_{n+1}, \infty[\longrightarrow \mathbb{R}$$

with $C^*(\alpha, \overline{d}) = \max\{C^\alpha(S) \mid S \in \mathcal{S}_T^{\overline{d}}\}$ is to cover its domain by a finite number of sets $\mathcal{M}$ such that on each of those sets, function C^* can be specified in closed form. Clearly, C^* is a closed-form function on subsets $\mathcal{M}$ of its domain where the active constraints for optimal schedules S are the same for all $(\alpha, \overline{d}) \in \mathcal{M}$ (and thus optimal schedules S can be represented by one and the same spanning tree G of N). For given spanning tree G, we call an $\subseteq$-maximal connected set $\mathcal{M}$ with the latter property a *validity domain* of G. Now let U_{ij} be the node set of the subtree which results from G by deleting arc (i,j) and does not contain node 0. Recall that arc $(i,j) \in E_G$ is called a forward arc if it is oriented in direction of the unique path from node 0 to node j in G, and a backward arc, otherwise (see Section 4.1). In addition, let $C_{ij}^\alpha(S)$ be the net present value of the events from set U_{ij} given schedule S. The following four remarks indicate how to compute the validity domains $\mathcal{M}$ of spanning trees G belonging to optimal schedules.

Remarks 6.1.

(a) Given α and $\overline{d}$, vertex S of $\mathcal{S}_T^{\overline{d}}$ is optimal if and only if there exists a spanning tree G representing S such that for each arc $(i,j) \in E_G$, it holds that $C_{ij}^\alpha(S) \geq 0$ if arc (i,j) is a forward arc and $C_{ij}^\alpha(S) \leq 0$ if (i,j) is a backward arc. The latter condition is equivalent to the requirement that there does not exist a feasible ascent direction z at S (see Subsection 3.2.2). Hence, the set U of all events that are shifted in time when modifying project deadline $\overline{d}$ coincides with set $U_{n+1,0}$ if backward arc $(n+1,0) \in E_G$ and is empty, otherwise.

(b) Given discount rate $\alpha > 0$, a spanning tree G belonging to an optimal vertex S of $\mathcal{S}_T^{\overline{d}}$ does not change when modifying project deadline $\overline{d}$, until a temporal constraint $S_j - S_i \geq \delta_{ij}$ with $(i,j) \notin E_G$ becomes active.

This property is immediate from the binary monotonicity of objective function C^α (see Subsection 2.3.1).

(c) Given deadline $\overline{d}$, a spanning tree G belonging to an optimal vertex S of $\mathcal{S}_T^{\overline{d}}$ does not change when modifying discount rate α, until for some arc $(i,j) \in E_G$, net present value $C_{ij}^\alpha(S)$ changes in sign. This property is a consequence of (a).

(d) When modifying deadline $\overline{d}$ for fixed spanning tree G, it follows from (b) that the deadline for which a new temporal constraint becomes active is independent of discount rate α. Symmetrically, when modifying discount rate α for fixed G, the discount rate for which $C_{ij}^\alpha(S) = 0$ for some $(i,j) \in E_G$ does not depend on deadline $\overline{d}$. This can be seen as follows. Let S' be an optimal vertex belonging to spanning tree G and deadline $\overline{d}'$. Then for given arc $(i,j) \in E_G$,

$$\begin{aligned} C_{ij}^\alpha(S') &= \sum_{h \in U_{ij}} c_h e^{-\alpha S'_h} \\ &= \sum_{h \in U_{ij} \setminus U} c_h e^{-\alpha S_h} + \sum_{h \in U_{ij} \cap U} c_h e^{-\alpha(S_h + \overline{d}' - \overline{d})} \\ &= \sum_{h \in U_{ij} \setminus U} c_h e^{-\alpha S_h} + e^{-\alpha(\overline{d}' - \overline{d})} \sum_{h \in U_{ij} \cap U} c_h e^{-\alpha S_h} \end{aligned} \tag{6.7}$$

Now recall that set U either coincides with set $U_{n+1,0}$ or is void. As a consequence, nodes i and j necessarily belong to the same set U or $V \setminus U$ unless $(i,j) = (n+1,0)$. For $U = U_{n+1,0}$ we have $U_{ij} = U$ if $(i,j) = (n+1,0)$, $U_{ij} \subset U$ if $i,j \in U$, and $U_{ij} \cap U = \emptyset$ if $i,j \notin U$. This means that independently of set U and arc (i,j) we have (1) $U_{ij} \cap U = \emptyset$ or (2) $U_{ij} \setminus U = \emptyset$. In case (1), it follows from (6.7) that $C_{ij}^\alpha(S') = C_{ij}^\alpha(S)$ and in case (2), equation (6.7) provides $C_{ij}^\alpha(S') = e^{-\alpha(\overline{d}' - \overline{d})} C_{ij}^\alpha(S)$. In sum, for each $(i,j) \in E_G$ it holds that $C_{ij}^\alpha(S') = 0$ precisely if $C_{ij}^\alpha(S) = 0$.

For a given spanning tree G, the net present value function C^* takes the form

$$\begin{aligned} C^*(\alpha, \overline{d}) &= \sum_{i \in V^c \setminus U} c_i^f e^{-\alpha S_i} + \sum_{i \in U} c_i^f e^{-\alpha(S_i + \overline{d} - S_{n+1})} \\ &= \sum_{i \in V^c \setminus U} c_i^f e^{-\alpha S_i} + e^{-\alpha(\overline{d} - S_{n+1})} \sum_{i \in U} c_i^f e^{-\alpha S_i} \end{aligned} \tag{6.8}$$

where schedule S is any optimal schedule that is specified by spanning tree G and U is the set of all events shifted when modifying $\overline{d}$. If $(n+1,0) \notin E_G$ and thus $U = \emptyset$, function C^* is constant in project deadline $\overline{d}$.

The amount by which $\overline{d}$ can be increased until a temporal constraint becomes binding is

$$\sigma^+(G) = \min\{S_j - S_i - \delta_{ij} \mid (i,j) \in E \setminus \{(n+1,0)\} \,;\, i \in U,\ j \in V^c \setminus U\}$$

Analogously, for the amount by which $\overline{d}$ can be decreased, we obtain

$$\sigma^{-}(G) := \min\{S_j - S_i - \delta_{ij} \mid (i,j) \in E : i \in V^e \setminus U,\ j \in U\}$$

The spanning tree G' for an optimal schedule S' to project deadline $\overline{d}' = \overline{d} + \sigma^{+}(G)$ or $\overline{d}' = \overline{d} - \sigma^{-}(G)$ can be constructed without re-performing any optimization by first, adding the arc (i,j) for the new active temporal constraint to G and second, deleting an oppositely directed arc (g,h) with minimum absolute net present value $|C^{\alpha}_{gh}(S')|$ in the resulting (undirected) cycle in G.

We now turn to the problem of finding the smallest discount rate $\alpha' > \alpha$ where some $C^{\alpha'}_{ij}(S)$ with $(i,j) \in E_G$ changes in sign. Let $\alpha^1_{ij} < \alpha^2_{ij} < \cdots < \alpha^{\nu}_{ij}$ denote all discount rates $\alpha_{ij} > \alpha$ such that for given optimal schedule S,

$$C^{\alpha_{ij}}_{ij}(S) = \sum_{h \in U_{ij}} c^f_h e^{-\alpha_{ij} S_h} = 0 \tag{6.9}$$

Each of those discount rates α_{ij} corresponds to an internal rate of return for the payment stream given by payments c^f_h for $h \in U_{ij}$ and schedule S. Thus, discount rates α_{ij} can be determined by one of the standard algorithms for the calculation of internal rates of return (see, e.g., Zheng and Sun 1999). The following condition is necessary and sufficient for a change in sign of $C^{\alpha'}_{ij}(S)$ at $\alpha' = \alpha_{ij}$, where $\Delta\alpha := \min\{\alpha^1_{ij} - \alpha, \alpha^2_{ij} - \alpha^1_{ij}, \ldots, \alpha^{\nu}_{ij} - \alpha^{\nu-1}_{ij}\}$:

$$\operatorname{sign} \sum_{h \in U_{ij}} c^f_h e^{-(\alpha_{ij} - \Delta\alpha/2) S_i} \neq \operatorname{sign} \sum_{h \in U_{ij}} c^f_h e^{-(\alpha_{ij} + \Delta\alpha/2) S_i} \tag{6.10}$$

For each set U_{ij}, we calculate the smallest discount rate $\alpha_{ij} > \alpha$ for which (6.9) and (6.10) are satisfied. An optimal schedule S' for discount rate $\alpha' := \min_{(i,j) \in E_G} \alpha_{ij}$ then results from delaying events $h \in U_{ij}$ by $\sigma^{+}(G)$ time units if (i,j) is a forward arc of G or from putting events $h \in U_{ij}$ forward by $\sigma^{-}(G)$ time units if (i,j) is a backward arc of G. The corresponding spanning tree G' can be determined analogously to the case where deadline $\overline{d}$ is varied.

In summary, each spanning tree G is valid for a rectangular set $\mathcal{M}$ which can be specified by the bottom left corner $(\alpha, \overline{d})$ and the top right corner $(\alpha', \overline{d}')$ (see Figure 6.5). For given α and $\overline{d}$, the values of α' and $\overline{d}'$ can be computed as described above. On the validity domain $\mathcal{M}$ for G, the net present value function can be written in the closed form (6.8).

Algorithm 6.1 provides a procedure for computing all bottom-left corners $(\alpha, \overline{d})$ of sets $\mathcal{M}$ along with the corresponding spanning trees G that uniquely define function C^* on sets $\mathcal{M}$. Q is a list of triples $(\alpha, \overline{d}, G)$ sorted according to nondecreasing discount rates α, ties being broken on the basis of increasing deadlines $\overline{d}$. For convenience we again put $\min \emptyset := \infty$. How to find an appropriate initial discount rate $\alpha^0 > 0$ is described in Schwindt and Zimmermann (2002).

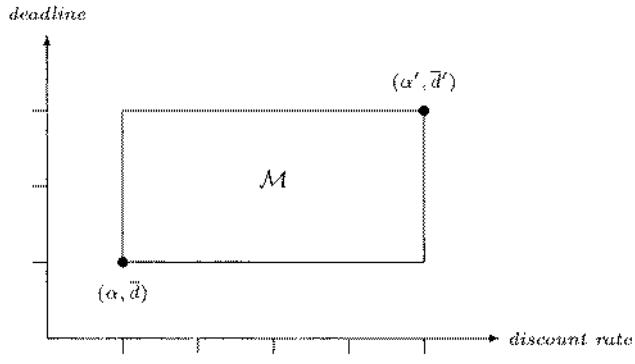

Fig. 6.5. Validity domain $\mathcal{M}$ for spanning tree G

Algorithm 6.1. Computation of net present value function

Input: Event-on-node network $N = (V^e, E, \delta)$, cash flows c_i^f for all events $i \in V^e$, initial discount rate $\alpha^0 > 0$.
Output: Set $\mathcal{C}$ of triples $(\alpha, \overline{d}, G)$.

```
compute spanning tree G belonging to optimal schedule for d̄ = ES_{n+1} and α = α^0;
initialize list Q := {(0, ES_{n+1}, G)} and set of triples C := {(0, ES_{n+1}, G)};
while Q ≠ ∅ do
  delete first element (α, d̄, G) and all other elements (α, ·, G) from list Q;
  repeat
    determine α' := min{α_ij | (i,j) ∈ E_G, α_ij satisfies (6.9) and (6.10)};
    if α' < ∞ then
      construct the spanning tree G' belonging to α' and d̄;
      add triple (α', d̄, G') to Q and C;
    if σ^+(G) < ∞ then
      set d̄' := d̄ + σ^+(G);
      construct the spanning tree G' belonging to α and d̄';
      if there is no triple (·, d̄', G') in list Q then add triple (α, d̄', G') to C;
      set d̄ := d̄' and G := G';
    else put d̄' := ∞;
  until d̄' = ∞;
return C;
```

6.5 Coping with Uncertainty

In this section we propose two deterministic strategies for coping with uncertainty in resource allocation problems. When executing a project, unforeseen downtimes of resources, staff time off, reworking time, late delivery of raw materials or bought-in parts, or imprecise time and resource estimations may cause considerable deviations from the schedule determined. Basically, there are two ways of taking uncertainty into account when performing the resource

allocation. First, we may *anticipate* deviations from predictive data by including the knowledge of uncertainty into the scheduling decisions. Second, when implementing the schedule, we may also *react* on disruptions in a way minimizing the impact of the required adaptations. An overview of different approaches to project scheduling under uncertainty is given by Demeulemeester and Herroelen (2002), Chs. 9 and 10, and Herroelen and Leus (2005), including stochastic, fuzzy, robust, and reactive project scheduling. Robust and reactive methods for project or production scheduling are reviewed in Herroelen and Leus (2004*b*) and Aytug et al. (2005), respectively.

Substantial work has been done in the area of (anticipative) *stochastic project scheduling problem*, where activity durations are modelled as stochastic variables and one attempts to minimize the expected value of a *regular* or *locally regular* objective function (see Möhring 2000 and Uetz 2003 for overviews and Stork 2001 for an in-depth treatment of the project duration problem). Algorithms for stochastic project scheduling are based on the concept of scheduling policies, which may be regarded as a specific application of the theory of stochastic dynamic programming to scheduling problems. Various classes of policies have been developed, which show a different behavior with respect to robustness and computational requirements. In principle, the policies studied define (ordinary or disjunctive) precedence relationships inducing feasible strict orders in the set of real activities. During the project execution, an activity is started as soon as all of its predecessors in that strict order have been completed. If we deal with arbitrary nonregular objective functions, those policies cannot be applied because it is generally no longer optimal to start activities at their earliest time- and resource-feasible start times.

An alternative **anticipative approach** to coping with uncertainty in planning problems refers to the concept of *robust plans* (see Scholl 2001, Ch. 4). We say that a plan is robust if it tends to require only minor revisions during its implementation. In project scheduling, a robust schedule may be defined on the basis of the *free floats* of activities as follows. For the moment we assume that only temporal constraints have to be observed. How to integrate resource constraints into this approach will be explained below. In Subsection 1.1.3 we have defined the concepts of early and late free floats with respect to the earliest and latest schedules. When implementing a schedule S, we may regard the activities as being "frozen", i.e., $ES_i = LS_i = S_i$ for all $i \in V$. In that case, the *early free float* EFF_i of an activity $i \in V$ with respect to schedule S is the maximum amount of time by which the start of activity i can be delayed given that any other activity j can be begun at its previous start time S_j. Symmetrically, the *late free float* LFF_i with respect to schedule S is the maximum amount of time by which the start of activity i can be advanced given that any other activity j can be begun at its previous start times S_j. Hence, a schedule that maximizes the (weighted) sum of all early and late free floats contains the maximum (weighted) temporal buffers for shifting activities in time without affecting the start times of any other activity. Such a schedule

can be computed by minimizing the convex objective function f of the *total weighted free float problem* (see Subsection 2.3.1) with

$$f(S) = \sum_{i \in V} w_i^f \left(\max_{(j,i) \in E} [S_j + \delta_{ji}] - \min_{(i,j) \in E} [S_j - \delta_{ij}] \right)$$

where weights $w_i^f \in \mathbb{N}$ can be chosen to reflect the degree of uncertainty with respect to start time S_i. The time-constrained total weighted free float problem can be transformed into a time-constrained project scheduling problem with a linear objective function $\widetilde{f}$ by introducing, for each $i \in V$, two auxiliary activities i' and i'' where (1) $S_{i'} \geq S_j + \delta_{ji}$ for all $(j,i) \in E$ and (2) $S_{i''} \leq S_j - \delta_{ij}$ for all $(i,j) \in E$. Conditions (1) and (2) can be expressed via additional arcs (j,i') with weights $\delta_{ji'} = \delta_{ji}$ for all $(j,i) \in E$ and arcs (i'',j) with weights $\delta_{i''j} = \delta_{ij}$ for all $(i,j) \in E$. Linear objective function $\widetilde{f}$ is then given by $\widetilde{f}(S) = \sum_{i \in V} w_i^f (S_{i'} - S_{i''})$. A similar model for time-constrained robust project scheduling under a probabilistic scenario has been studied by Herroelen and Leus (2004*a*). The objective function considered in the latter paper is the expected weighted deviation in start times between the realized schedule and baseline schedule S, where it is assumed that no activity i is started before its predictive start time S_i.

We now drop our assumption of infinite resource availability. In the presence of resource constraints, the free floats depend on the way in which resource conflicts are resolved. Similarly to stochastic project scheduling, the conflict resolution strategy can be specified as a feasible relation ρ in the set V of all activities (cf. Subsections 2.1.1 and 2.1.2). The feasibility of ρ implies that in combination with the temporal constraints, the precedence constraints among real activities and among events given by pairs $(i,j) \in \rho$ guarantee that the resource constraints are satisfied. Thus, by substituting project network N into relation network $N(\rho)$ and minimizing f on relation polytope $\mathcal{S}_T(\rho)$, we obtain a feasible robust schedule that maximizes the total weighted free float for the given set of precedence constraints. It remains to show how to determine a feasible relation ρ such that the minimizer S of f on $\mathcal{S}_T(\rho)$ also minimizes f on the set $\mathcal{S}$ of all feasible schedules, i.e., such that S is a feasible schedule with maximum total weighted free float. First, we notice that when replacing project network N with relation network $N(\rho)$, the objective function to be minimized turns into $f^\rho : \mathcal{S}_T \to \mathbb{R}$ with

$$f^\rho(S) = \sum_{i \in V} w_i^f \left(\max_{(j,i) \in E \cup \rho} [S_j + \delta_{ji}^\rho] - \min_{(i,j) \in E \cup \rho} [S_j - \delta_{ij}^\rho] \right)$$

where δ_{ji}^ρ and δ_{ij}^ρ denote the weights of arcs (j,i) and (i,j) in relation network $N(\rho)$. Thus, the objective function to be minimized explicitly depends on ρ. From the definition of f^ρ it follows that $f^\rho(S) \leq f^{\rho'}(S)$ for all $S \in \mathcal{S}_T$ if ρ' is an extension of ρ. In addition, it obviously holds that $\mathcal{S}_T(\rho') \subseteq \mathcal{S}_T(\rho)$ if $\rho' \supseteq \rho$. That is why the feasible relation ρ sought can be chosen among the $\subseteq$-minimal feasible relations. The latter relations can be generated by using

a modification of the relaxation-based enumeration scheme given by Algorithm 3.3. At each iteration it is checked whether or not the relation ρ we branch from is feasible by finding minimum (s,t)-flows in specific relation-induced flow networks belonging to the induced preorder $\theta = \Theta(D(\rho))$ (cf. Subsections 2.1.1 and 2.1.2). Instead of breaking up forbidden active sets $\mathcal{A}(S,t)$ we then break up maximum (s,t)-cuts U in the respective flow networks (for details see Sections 4.1 and 5.2, where we have applied similar techniques).

The problem where for given schedule S the resource allocation (i.e., an appropriate (s,t)-flow) is to be determined in such a way that the expected weighted deviation between the realized schedule and schedule S is minimized has been investigated by Leus and Herroelen (2004).

The **reactive approach** is as follows. Assume that we want to minimize some regular or convexifiable objective function like the project duration, the total tardiness cost, the total inventory holding cost, or the net present value of the project. At first, we determine an optimal schedule S for objective function f by using the relaxation-based enumeration scheme. We then start performing the project according to schedule S. Each time the schedule becomes infeasible due to the breakdown of resources or overrun on activity durations, we determine a new schedule S' that first, complies with the updated constraints and second, resembles as much as possible previous schedule S. The reason is that we want to avoid disruptions in the project execution that may arise from substantial modifications of the schedule. The resemblance between schedules S and S' may, e.g., be measured by the (weighted Manhattan) distance

$$\Delta(S',S) = \|S' - S\| := \sum_{i \in V} w_i |S'_i - S_i|$$

where $w_i \in \mathbb{N}$ are integers specifying the cost for shifting the start of activity $i \in V$ by one unit of time. $\Delta(S',S)$ coincides with the objective function value $\widetilde{f}(S')$ of schedule S' if $\widetilde{f}$ is chosen to be the *total earliness-tardiness cost* function with due dates d_i being equal to completion times $S_i + p_i$ and $w_i^e = w_i^t = w_i$ for all $i \in V$. Thus, we may determine a new feasible schedule S' by minimizing the total earliness-tardiness cost with respect to previous schedule S and put $S := S'$. In case of frequent schedule revisions, the computational effort for rescheduling the project can be markedly decreased by providing, in addition to schedule S, a feasible relation ρ in set V. We then minimize the total earliness-tardiness cost $\widetilde{f}$ on the intersection of relation polytope $\mathcal{S}_T(\rho)$ with the updated feasible region $\mathcal{S}'$. An appropriate $\subseteq$-minimal feasible relation ρ can be determined in the same way as for the anticipative approach.