OR Spektrum (2000) 22: 501–524

OR Spektrum

Batch scheduling in process industries: an application of resource–constrained project scheduling*

Batch–Scheduling in der Prozeßindustrie: eine Anwendung der ressourcenbeschränkten Projektplanung

Christoph Schwindt and Norbert Trautmann

Institut für Wirtschaftstheorie und Operations Research, Universität Karlsruhe, 76128 Karlsruhe, Germany (e-mail: {schwindt,trautmann}@wior.uni-karlsruhe.de)

Received: October 15, 1999 / Accepted: March 21, 2000

Abstract. The paper deals with batch scheduling problems in process industries where final products arise from several successive chemical or physical transformations of raw materials using multi–purpose equipment. In batch production mode, the total requirements of intermediate and final products are partitioned into batches. The production start of a batch at a given level requires the availability of all input products. We consider the problem of scheduling the production of given batches such that the makespan is minimized. Constraints like minimum and maximum time lags between successive production levels, sequence–dependent facility setup times, finite intermediate storages, production breaks, and time–varying manpower contribute to the complexity of this problem. We propose a new solution approach using models and methods of resource–constrained project scheduling, which (approximately) solves problems of industrial size within a reasonable amount of time.

Zusammenfassung. Die Arbeit behandelt Batch–Scheduling–Probleme in der Prozeßindustrie. In mehreren aufeinanderfolgenden chemischen oder physikalischen Transformationsschritten werden aus Rohstoffen auf Mehrzweckanlagen Endprodukte hergestellt. Wird die Anlage im Batch–Modus betrieben, so werden die Gesamtbedarfe an Zwischen- und Endprodukten in Chargen unterteilt. Der Produktionsbeginn einer Charge auf einer Stufe erfordert die Verfügbarkeit aller Eingangsstoffe. Wir betrachten das Problem der Ablaufplanung für die Chargenpro-

* The authors are very grateful for many helpful comments and suggestions of two anonymous referees and the editor Hans–Otto Günther. This research is supported in part by the SAP AG, Walldorf, and in part by the Deutsche Forschungsgemeinschaft under Grant Ne 137/4.

Correspondence to: N. Trautmann

duktion mit dem Ziel der Zykluszeitminimierung. Nebenbedingungen wie zeitliche Mindest- und Höchstabstände zwischen aufeinanderfolgenden Produktionsstufen, reihenfolgeabhängige Umrüstzeiten von Betriebsmitteln, kapazitiv begrenzte Zwischenlager, Produktionspausen und die zeitlich schwankende Personalverfügbarkeit tragen zur Komplexität dieses Problems bei. Wir schlagen einen neuen Lösungsansatz auf der Grundlage von Modellen und Methoden der ressourcenbeschränkten Projektplanung vor, mit dessen Hilfe Probleminstanzen industrieller Größe in angemessener Rechenzeit näherungsweise gelöst werden können.

Key words: Resource–constrained project scheduling – Batch scheduling – Process industries

Schlüsselwörter: Ressourcenbeschränkte Projektplanung – Batch–Scheduling – Prozeßindustrie

1 Introduction

In process industries, e.g. the chemical, pharmaceutical, food, or paper industry, the production of final products typically takes place in several successive chemical or physical transformation processes (also called *tasks*) such as heating, filtration, or packaging. With respect to the material flow, we distinguish between continuous production and batch (discontinuous) production mode (cf. Blömer and Günther, 1998). Continuous production is characterized by a continuous flow of material, which, e.g., is typical of oil industry. In the following, we focus on process industries operating in batch production mode, where we assume that the input of a task is consumed at its start and the output arises at its completion. Usually, the production is processed in batch mode if small amounts of a large number of products are required. This offers the advantage of a high flexibility with respect to product variety and the demand of the market. The combination of a task and a respective production quantity is called *batch*. The production of a batch is referred to as *operation*. Notice that a task may be performed more than once, resulting in several corresponding operations. In what follows, we assume the batch sizes belonging to the individual operations to be predetermined by technology, regulations, or as output of a higher planning level. Moreover, it is typical of chemical batch processes that processing times are independent of the respective batch sizes. That is why in practice, batch sizes often arise from the capacities of reactors or tanks.

The problem of batch scheduling considered in this paper now consists of allocating scarce resources over time to a given set of operations such that the production is completed within a minimum amount of time (*makespan*). This objective is particularly important in batch production, where a large number of different products are processed on multi–purpose equipment (cf. Blömer and Günther, 1998). In this case, the plant is configured according to the required final products. Before processing the next set of final products, the plant has to be reconfigured, which requires the completion of all operations.

A variety of technological and organizational constraints have to be taken into account: Between different production levels, minimum and maximum time lags

have to be observed resulting from quarantine or shelf–life times. A task generally takes up different types of resources: *manpower*, *processing units* with sequence–dependent setup times (e.g. reactors or filters), and *storages* (e.g. tanks or silos). All these resources are available in limited capacity only, where the manpower is not necessarily constant over time. Break calendars specify time intervals during which specific tasks cannot be processed. Tasks which can be interrupted (e.g. packaging) have to be resumed immediately after break. Other tasks (e.g. heating) cannot be interrupted at all.

There is an extensive literature dealing with production planning and scheduling problems in process industries. Basic issues are discussed in Applequist et al. (1997), Blömer and Günther (1998), Kallrath and Wilson (1997), Reklaitis (1996), Rippin (1993), and Schürbüscher et al. (1992). Allweyer et al. (1994) and Blömer and Günther (1999) survey different types of chemical batch processes found in practice. Most of the solution approaches proposed in literature are based on MIP formulations, cf. e.g. Birewar and Grossmann (1990), Blömer (1999), Blömer and Günther (2000), Burkhard et al. (1998b), Dedopoulos and Shah (1995), Dimitriadis et al. (1997), Kim et al. (1996), Kondili et al. (1993), Mockus and Reklaitis (1997), Moon (1996), Pinto and Grossmann (1996), or Shah et al. (1993a,b). Branch–and–bound procedures can be found in Ku and Karimi (1990), Pekny et al. (1993), and Schilling and Pantelides (1996). Metaheuristics are presented in Brucker and Hurink (1999), Cartwright and Long (1993), Das et al. (1990), and Fortemps et al. (1996). Further heuristic methods are devised by Artiba and Tahon (1992), Dessouky et al. (1996), He et al. (1996), and Kudva et al. (1994). Some of these approaches additionally include the problem of determining batch sizes. For a detailed literature review, we refer to Blömer (1999). A survey of commercial software packages for scheduling that are also applicable to process industries is given in Benoy et al. (1994).

Even if the flexibility of MIP modelling allows an easy integration of additional constraints, no paper covers all the issues discussed here, in particular general minimum and maximum time lags, time–varying manpower, and break calendars. The peculiarity of our approach is the integration of all constraints within the framework of resource–constrained project scheduling, which offers the advantage of efficiently modelling and (approximately) solving even large–scale problems. In particular, our approach is independent of the chosen time grid which seems to be the main problem with time–indexed MIP formulations. The generation scheme is based on the relaxation of resource scarcity, which is also the basic principle of the project scheduling algorithms presented in Bartusch et al. (1988), De Reyck and Herroelen (1998), Möhring et al. (1998), and Schwindt (1998) coping with the project duration problem subject to general temporal and (manpower) resource constraints. Further approaches for this project duration problem can be found in Dorndorf et al. (1998) and Franck (1999). An overview of recent developments in resource–constrained project scheduling is given in Brucker et al. (1999) and Neumann and Zimmermann (1999).

The remainder of this paper is organized as follows: In Section 2 we illustrate all constraints of our problem using a case study which is based on Westenberger and Kallrath (1995). In Section 3 we present a resource–constrained project scheduling

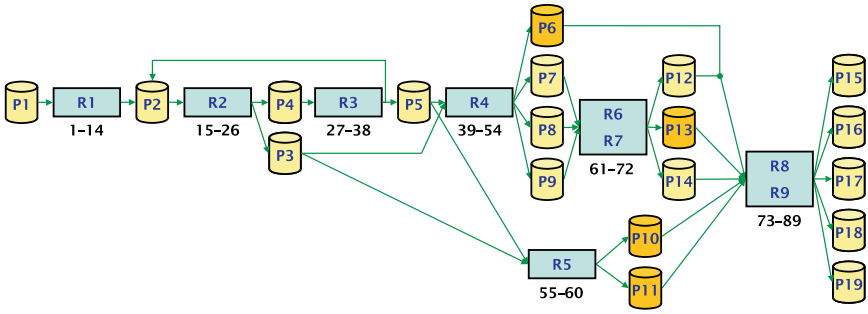

Fig. 1. Example of a production structure

model including the temporal, the resource, and the calendarization constraints. Section 4 sketches a generation scheme using concepts of project scheduling as building blocks. We demonstrate the efficiency of our algorithm on the basis of a benchmark instance introduced by Westenberger and Kallrath (1995) including 89 operations and 28 resources. Section 5 briefly discusses some extensions which typically occur when dealing with practical problems, e.g. alternative non–identical resources and further objective functions. Section 6 is devoted to concluding remarks and directions for further research.

2 Example

Westenberger and Kallrath (1995) present a case study that covers most of the features contributing to the complexity of batch scheduling problems in process industries. The papers by Ahleff (1995), Blömer (1999), Blömer and Günther (1998, 1999, 2000), Burkhard et al. (1998a,b), and Rosenau (1996) have been inspired by this case study and show that it is extremely difficult to obtain feasible schedules within an affordable amount of time when taking into account all the given constraints. Figure 1 illustrates the production structure, which consists of storages for 19 products $P1$ to $P19$ and nine processing units $R1$ to $R9$. The storages and processing units are linked by divergent, convergent, as well as cyclic material flows. Accordingly, bills of materials are analytic (e.g. oil industry), synthetic (e.g. pharmaceutical industry), or cyclic (e.g. catalysts). The primary requirements for the final products $P15, \ldots, P19$ are given by demand vector $(30, 30, 40, 20, 40)$ (cf. task 4 in Westenberger and Kallrath, 1995).

The problem is to explode the primary requirements into batches (batching problem) and to schedule these batches on scarce resources (batch scheduling problem). The above approaches perform batching and batch scheduling simultaneously. Our (heuristic) approach is as follows. By a simple heuristic, the production of the final products is decomposed into 89 operations with the characteristics as shown in Table 1. The principle idea of this procedure is to start with the final products and to choose the batch sizes according to capacities of processing units and storages and the batch sizes of consuming tasks. In what follows we emphasize on the resulting batch scheduling problem.

Table 1. Operations settings

Tasks	Operations	Alternative processing units	No of workers	Duration [min, max]	Materials consumed	Materials produced
1	$1, \ldots, 14$	$R1$	1	$[2, 2]$	$P1$: 10	$P2$:10
2	$15, \ldots, 26$	$R2$	1	$[4, 4]$	$P2$: 15	$P3$:5, $P4$:10
3	$27, \ldots, 38$	$R3$	1	$[2, 2]$	$P4$: 10	$P2$:3, $P5$:7
4	$39, \ldots, 42$	$R4$	1	$[4, 4]$	$P3$: 5	$P6$:5
5	$43, 44$	$R4$	1	$[4, 4]$	$P3$: 7	$P7$:7
5	45	$R4$	1	$[4, 4]$	$P3$: 6	$P7$:6
6	$46, 47, 48$	$R4$	1	$[4, 4]$	$P5$: 7	$P8$:7
7	$49, \ldots, 54$	$R4$	1	$[4, 4]$	$P5$: 7	$P9$:7
8	$55, 56, 57$	$R5$	1	$[6, 6]$	$P3$: 10	$P10$:10
9	$58, 59, 60$	$R5$	1	$[6, 6]$	$P5$: 10	$P11$:10
10	$61, 62$	$R6$ $R7$	1	$[5, 5]$	$P7$: 7	$P12$:7
10	63	$R6$ $R7$	1	$[5, 5]$	$P7$: 6	$P12$:6
11	$64, 65, 66$	$R6$ $R7$	1	$[6, 6]$	$P8$: 7	$P13$:7
12	$67, \ldots, 72$	$R6$ $R7$	1	$[6, 6]$	$P9$: 7	$P14$:7
13	$73, 74, 75$	$R8$ $R9$	2	$[6, \infty]$	$P10$: 10	$P15$:10
14	$76, 77, 78$	$R8$ $R9$	2	$[6, \infty]$	$P11$: 10	$P16$:10
15	$79, \ldots, 82$	$R8$ $R9$	2	$[4, 12]$	$P6$: 5, $P12$:5	$P17$:10
16	$83, 84, 85$	$R8$ $R9$	2	$[6, \infty]$	$P13$: 7	$P18$:7
17	$86, \ldots, 89$	$R8$ $R9$	2	$[6, \infty]$	$P14$: 10	$P19$:10

Each operation takes up one processing unit and a given number of workers (cf. Table 1). In Fig. 1, the processing units are labelled with the operations executed in.

Intermediate products $P6$, $P10$, $P11$, and $P13$ cannot be stored. In many applications, some materials can be stored for a given *shelf–life time* only, e.g. fresh milk or glue. Some operations require a *quarantine time* before subsequent operations can be started, e.g. the time hot chocolate needs for cooling. Moreover, for some of the operations a *release date* for start or a *deadline* for completion may be imposed. Thus, prescribed *time lags* between individual operations have to be taken into account.

For the storable materials, the inventory is bounded from below by the safety stock and from above by some storage capacity (cf. Table 2). Generally, storage space may be limited for raw materials, intermediate products, final products, and some kinds of waste. For simplicity, we assume that sufficient capacity is available to store the required raw material $P1$ and final products $P15$ to $P19$, that there is a sufficient initial inventory of $P1$, and that there is no initial stock of $P15$ to $P19$. Products $P2$ and $P5$ are chemically identical and share the same storage tank. In practice, constraints on homogeneous storages (e.g. tanks) as well as on heterogeneous storages (e.g. cold stores) have to be considered. A task may deplete and/or replenish storages. In what follows, we assume that depletion occurs at the start and that replenishment occurs at the completion of an operation. Precedences

Table 2. Inventories

	$P1$	$P2$	$P3$	$P4$	$P5$	$P7$	$P8$	$P9$	$P12$	$P14$	$P15$	$P16$	$P17$	$P18$	$P19$
Initial	140	10	10	0	10	0	0	0	0	0	0	0	0	0	0
Minimum	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Maximum	∞	30	30	15	30	10	10	10	10	10	∞	∞	∞	∞	∞

Table 3. Setup times

Task	4	5	6	7	8	9	10	11	12	13	14	15	16	17
4	0	4	4	4										
5	0	0	4	4										
6	0	0	0	4										
7	0	0	0	0										
8					0	6								
9					0	0								
10							0	5	5					
11							0	0	6					
12							0	0	0					
13										0	3	3	3	3
14										0	0	3	3	3
15										0	0	0	2	2
16										0	0	0	0	3
17										0	0	0	0	0

between tasks resulting from the production structure are taken into account by the constraints on the minimum inventories of input products, an approach which offers more flexibility compared to the linking of operations of consecutive levels by minimum time lags (cf. e.g. Aquilano and Smith, 1980; Günther, 1992; Neumann and Schwindt, 1997). Such a linking would require a fixed matching between producing and consuming tasks.

The individual operations can be carried out on *multi–purpose processing units* (cf. Frauendorfer and Königsperger, 1996), i.e. facilities which can operate several kinds of tasks (but only one at a time). At some production levels there are parallel processing units, e.g. $R6$ and $R7$. These parallel processing units are not necessarily identical, i.e., the processing time and further process properties may depend on the particular processing unit used. For the moment, we assume alternative processing units to be identical; the case of alternative non–identical processing units (cf. Westenberger and Kallrath, 1995) will be discussed in Section 5.

In order to guarantee purity of products, cleaning of a processing unit between two successive tasks becomes necessary if tasks differ in their produced materials. As processing units $R1$, $R2$, and $R3$ operate only one task each, the corresponding setup times are 0. Table 3 shows the setup times between tasks for processing units $R4, \ldots, R9$ as given by Westenberger and Kallrath (1995). Note that these cleanings cannot be modelled by means of time lags as their durations depend on the product sequence.

Table 4. Number of available workers in each shift

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
[6 a.m., 2 p.m.[	8	8	8	8	8	4	4
[2 p.m., 10 p.m.[	8	8	8	8	8	4	4
[10 p.m., 6 a.m.[	4	4	4	4	4	4	4

In addition to Westenberger and Kallrath (1995), we assume that besides processing units also *workers* are needed for processing tasks, e.g. for operating a processing unit or checking the quality of an intermediate product. Each operation requires one or two workers (cf. Table 1). In this example, all workers have the same qualification, i.e., each worker can process every task. In most practical applications, workers of different skills have to be considered. The workers operate in two day shifts and one night shift which differ in the number of available workers (cf. Table 4). The planning period is supposed to start at 6 a.m. of a Monday.

We assume that there is a break for meals during the 6th hour of each day shift. We have to distinguish between tasks which may be interrupted by these *production breaks* (tasks being performed manually, e.g. packaging) and non–interruptible tasks (tasks using expensive equipment or chemical reactions). In our example, we assume that operations 73 to 78 may be interrupted by production breaks, whereas the remaining operations must be processed without any interruption. Note that some tasks can be processed even during a production break, e.g. some chemical reactions. Thus, we have to assign individual break calendars to each operation. We assume that the final operations 73 to 89 cannot be processed during a break, while the other operations may be carried out independently of breaks.

The duration of some tasks is not necessarily known in advance, but there may be a minimum and a maximum duration prescribed. Operations 79 to 82 mixing products $P6$ and $P12$ have to start immediately when a batch of the non–storable product $P6$ becomes available, and have to be "executed" for the prescribed minimum duration. Subsequently, the output $P17$ can stay in the respective processing unit for a certain time, which is given by the difference of the maximum and the minimum duration. Variable processing times offer an additional degree of freedom if intermediate storage space is limited or succeeding operations are linked by shelf–life times.

3 Model

3.1 Basic concepts

Suppose that n operations numbered from 1 to n have to be scheduled. We model the processing of the operations as a resource–constrained project (cf. e.g. Brucker et al., 1999; Neumann and Schwindt, 1997) that consists of a set $V = \{0, 1, \ldots, n, n+1\}$ of activities which require resources and which are connected by prescribed minimum and maximum time lags. Each operation is identified with exactly one real activity $i \in \{1, \ldots, n\}$ of the project and gives rise to two events, the start

event i^S that occurs at time $S_i \in \mathbb{R}_{\geq 0}$ and the completion event i^C that occurs at time $C_i \in \mathbb{R}_{\geq 0}$ with $C_i \geq S_i$. Dummy activity 0 represents the production start and dummy activity $n+1$ corresponds to the production end, both regarded as activities and as events. We assume the production to start at time zero, i.e. $S_0 := 0$. The makespan of the production equals C_{n+1}.

$\widetilde{V} := \{0, 1^S, 1^C, \ldots, n^S, n^C, n+1\}$ represents the set of all events. With T_e being the time of occurrence of event $e \in \widetilde{V}$ we have $T_{i^S} = S_i$ and $T_{i^C} = C_i$ ($i \in \{1, \ldots, n\}$). A vector

$$T = (T_e)_{e \in \widetilde{V}} := (S_0, S_1, \ldots, S_n, C_1, \ldots, C_n, C_{n+1})$$

of start and completion times is termed *schedule*.

Batch scheduling refers to the problem of determining a schedule such that all given constraints are observed and the makespan C_{n+1} is minimized. The following subsections are devoted to the (conceptional) modelling of temporal, resource, and break constraints of the batch scheduling problem.

3.2 Temporal constraints

As we mentioned in Section 2, minimum and maximum time lags may be given between starts or completions of activities $i, j \in V$. We denote a minimum (maximum) time lag between corresponding events $e, f \in \widetilde{V}$ by $d_{ef}^{min} \in \mathbb{Z}_{\geq 0}$ ($d_{ef}^{max} \in \mathbb{Z}_{\geq 0}$). Thus, a maximum time lag between the completion of activity i and the start of activity j is denoted by $d_{i^C j^S}^{max}$.

In what follows, we partition the set of activities into the set of (break–) interruptible activities V^{bi} and the set of non–interruptible activities V^{ni} with $0, n+1 \in V^{ni}$. For each activity i, we have a *minimum duration* $p_i^{min} \in \mathbb{Z}_{\geq 0}$ and a *maximum duration* $p_i^{max} \in \mathbb{Z}_{\geq 0} \cup \{\infty\}$ with $p_i^{max} \geq p_i^{min}$, where the term "duration" refers to the time p_i during which an activity is actually in progress. If activity $i \in V^{bi}$ is interrupted, the difference $C_i - S_i$ between completion and start times is larger than duration p_i. For the production start and the production end, we have $p_0^{min} = p_0^{max} = p_{n+1}^{min} = p_{n+1}^{max} = 0$. There also may be tasks with unbounded maximum duration. In this case, we have $p_i^{max} = \infty$. Since for real non–interruptible activities $i \in V^{ni}$ the processing time p_i coincides with $C_i - S_i$, the requirement $p_i^{min} \leq p_i \leq p_i^{max}$ gives rise to a minimum time lag $d_{i^S i^C}^{min} := p_i^{min}$ and for $p_i^{max} < \infty$ to a maximum time lag $d_{i^S i^C}^{max} := p_i^{max}$ between the start and the completion of i. The case of interruptible activities is dealt with in Subsection 3.4.

In project scheduling, it is customary to represent the project in question by a network. To this end we assign event e to node e and identify node e with event e ($e \in \widetilde{V}$). Accordingly, the network contains two nodes for each activity $i \in \{1, \ldots, n\}$. Additionally, we assign node 0 ($n+1$) to the production start 0 (the production end $n+1$). Arc set E consists of arcs $\langle e, f \rangle$ between events $e, f \in \widetilde{V}$ which are linked by time lags: If there is a prescribed minimum time lag d_{ef}^{min} between events $e, f \in \widetilde{V}$, we introduce an arc $\langle e, f \rangle$ with weight $\delta_{ef} := d_{ef}^{min}$. In case of a maximum time lag d_{ef}^{max} between events $e, f \in \widetilde{V}$, we have an arc $\langle f, e \rangle$

with weight $\delta_{fe} := -d_{ef}^{max}$. The *project network* N consists of node set $\widetilde{V}$, arc set E, and arc weights δ, $N = \langle \widetilde{V}, E; \delta \rangle$ for short.

Since no task can be started before the production start, we have $S_i \geq 0$ for all $i \in V$. If there is no path in N from 0 to i^S ($i \in V$) of nonnegative length, we add an arc $\langle 0, i^S \rangle$ with weight $\delta_{0i^S} = 0$. Similarly, it has to be ensured that $C_i \leq C_{n+1}$ holds for all $i \in V$ because all tasks have to be completed before the production end: If there is no path in N from i^C to $n+1$ ($i \in V$) of nonnegative length, we introduce an arc $\langle i^C, n+1 \rangle$ with weight $\delta_{i^C,n+1} = 0$.

A schedule T is called *time–feasible* iff it satisfies

$$T_f \geq T_e + \delta_{ef} \quad (\langle e, f \rangle \in E).$$

Due to the occurrence of maximum time lags, network N generally contains cycles. It is well–known that a time–feasible schedule exists exactly if N does not contain any cycle of positive length (cf. Bartusch et al., 1988).

We now consider some particular time lags which are used for modelling constraints introduced in Section 2:

- The output of activity i must be consumed by activity j after a shelf–life time $s_i \in \mathbb{Z}_{\geq 0}$ at the latest: $d_{i^C j^S}^{min} := 0$, $d_{i^C j^S}^{max} := s_i$.
- The output of activity i can be consumed by activity j after a quarantine time $q_i \in \mathbb{Z}_{\geq 0}$ at the earliest: $d_{i^C j^S}^{min} := q_i$.
- Activity i can be started at a release date $r_i \in \mathbb{Z}_{\geq 0}$ at the earliest: $d_{0i^S}^{min} := r_i$.
- Activity i has to be completed by a deadline $\overline{d}_i \in \mathbb{Z}_{\geq 0}$: $d_{0i^C}^{max} := \overline{d}_i$.

Further applications where minimum and maximum time lags are needed for modelling can be found in Neumann and Schwindt (1997), Schwindt (1998), and Trautmann (1999).

Figure 2 shows a detail of project network N which corresponds to the example of Section 2. Activities 61, 64, and 67 have a fixed duration. Activities 79, 83, and 86 have a variable duration, where for activities 83 and 86 the duration is not bounded from above. Notice that activity 64 produces intermediate product $P13$ that cannot be stored, i.e., which has a shelf–life time of 0. Thus, a corresponding consuming activity, e.g. operation 83, has to start immediately at the completion of activity 64. In what follows, we consider the (partial) schedule

$$\begin{aligned} T' :&= (S_{61}, S_{64}, S_{67}, S_{79}, S_{83}, S_{86}, C_{61}, C_{64}, C_{67}, C_{79}, C_{83}, C_{86}) \\ &= (18, 22, 22, 22, 28, 30, 23, 28, 28, 28, 34, 36). \end{aligned}$$

As can easily be verified, schedule T' is time–feasible.

3.3 Resource constraints

Each task requires one or several workers with special skills and takes up processing units and/or storage units. *Workers* with the same skills are grouped to *manpower pools* that are modelled as so–called *renewable resources*. Let $\mathcal{R}^\rho$ denote the set of renewable resources. A resource $k \in \mathcal{R}^\rho$ is called renewable if the availability

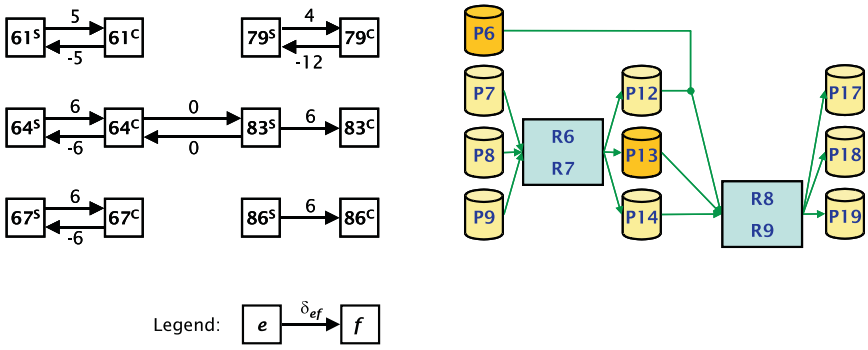

Fig. 2. Detail of the project network and the production structure

of its (limited) *capacity* $R_k^\rho \in \mathbb{Z}_{>0}$ is independent of its previous utilization. The capacity of a renewable resource equals the number of workers in the corresponding manpower pool. For each activity $i \in V$, we have a *demand* $r_{ik}^\rho \in \mathbb{Z}_{\geq 0}$ for resource $k \in \mathcal{R}^\rho$. That means that r_{ik}^ρ units of resource k are taken up during the execution of activity i, where we suppose that i is in progress from time S_i inclusively to time C_i exclusively. The demand of an activity coincides with the required number of workers of the corresponding pool. For the fictitious activities 0 and $n+1$, we have $r_{0k}^\rho = r_{n+1,k}^\rho = 0$ ($k \in \mathcal{R}^\rho$).

For the case where the capacity of a renewable resource is not constant over time, we proceed as follows (cf. Bartusch et al., 1988). Let the piecewise constant function $R_k^\rho(t)$ give the available capacity of resource $k \in \mathcal{R}^\rho$ at time $t \geq 0$ and let $R_k^\rho := \max_{t \geq 0} R_k^\rho(t)$ be the maximum resource capacity available. We suppose $R_k^\rho(t)$ to be continuous from the right and to have a finite number ν of jump discontinuities at $t = \tau_1, \ldots, \tau_\nu$. Adjacent time intervals $[\tau_{\mu-1}, \tau_\mu[$ and $[\tau_\mu, \tau_{\mu+1}[$ differ in the available capacity of at least one renewable resource. This is modelled by a set V' of dummy activities. For each interval with $R_k^\rho(\tau_{\mu-1}) < R_k^\rho$ for some $k \in \mathcal{R}^\rho$ where $\tau_0 := 0$, we introduce one corresponding dummy activity $i \in V'$. We set $S_i := \tau_{\mu-1}$, $C_i := \tau_\mu$, and $r_{ik}^\rho := R_k^\rho - R_k^\rho(S_i)$.

For a given schedule T, we define the *active set*

$$\mathcal{A}^\rho(T,t) := \{i \in V \cup V' \mid S_i \leq t < C_i\}$$

of real and dummy activities in progress at time $t \geq 0$. A *manpower–feasible schedule* T has to satisfy

Table 5. Dummy activities for the first week

i	91	92	93	94	95
S_i	16	40	64	88	112
C_i	24	48	72	96	168
r_{i1}^ρ	4	4	4	4	4

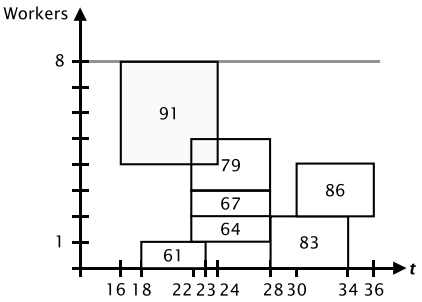

Fig. 3. Pool requirements

$$\sum_{i \in \mathcal{A}^\rho(T,t)} r_{ik}^\rho \leq R_k^\rho \quad (k \in \mathcal{R}^\rho; t \geq 0). \tag{1}$$

In the example of Section 2, we assumed all workers to have the same skills, i.e., we group the workers to one manpower pool, which is modelled as one renewable resource $k = 1$ with capacity $R_1^\rho = 8$. We need five dummy activities per week as given in Table 5 (recall that the planning period starts at Monday 6 a.m.). As can be seen from Fig. 3, schedule T' is not manpower–feasible.

Next, we turn to *processing units*. Identical processing units are grouped to pools of processing units, where each pool is modelled as a so–called *setup resource* which in contrast to a renewable resource has to be set up between subsequent operations. Demeulemeester and Herroelen (1995), Günther (1992), and Neumann and Schwindt (1997) discuss sequence–independent setups between activities of a project. The case where the need for setups depends on the activity sequence is treated in Kolisch (1995). In this paper, we deal with the more general case of sequence–dependent setup times.

The set of setup resources is denoted by $\mathcal{R}^\sigma$. Processing units are identical if they can operate the same tasks and have the same processing and setup times, e.g. $R6$ and $R7$ in the example in Section 2. The capacity $R_k^\sigma \in \mathbb{Z}_{>0}$ of a setup resource $k \in \mathcal{R}^\sigma$ equals the number of processing units in the corresponding pool. In what follows, we suppose that no task requires more than one processing unit of a pool of processing units, i.e., the demand r_{ik}^σ of resource $k \in \mathcal{R}^\sigma$ by activity $i \in V$ is either 0 or 1.

Between activity i and that activity j ($i, j \in V$) which is processed immediately after i on the same unit of resource $k \in \mathcal{R}^\sigma$, a *sequence–dependent setup time* $\vartheta_{ij}^k \in \mathbb{Z}_{\geq 0}$ has to be observed. For simplicity, we set $\vartheta_{0i}^k := \vartheta_{i,n+1}^k := 0$ ($i \in V$; $k \in \mathcal{R}^\sigma$) and suppose that the input products for activity j need not to be completed when the processing unit is set up between activities i and j.

Given a schedule T, let σ_{ik} denote the index of the activity that is processed immediately after activity i on the same unit of resource $k \in \mathcal{R}^\sigma$ if such an assignment is possible. If i is the last activity to be processed on a unit of resource k, we set $\sigma_{ik} := n + 1$. Let $V_k^\sigma := \{i \in V \mid r_{ik} = 1\}$ denote the set of activities being processed on a processing unit of setup resource $k \in \mathcal{R}^\sigma$. A schedule is *process–feasible* if indices σ_{ik} can be chosen such that

$$S_{\sigma_{ik}} \geq C_i + \vartheta^k_{i\sigma_{ik}} \; (i \in V^\sigma_k;\; k \in \mathcal{R}^\sigma). \tag{2}$$

With $\overline{d}$ denoting an upper bound on the minimum makespan or representing the planning horizon, a given schedule T satisfies (2) iff there is an assignment of variables $y^k_{ij} \in \{0,1\}$ observing the following constraints:

$$S_j \geq C_i + \vartheta^k_{ij} - \overline{d}(1 - y^k_{ij}) \qquad (k \in \mathcal{R}^\sigma; i,j \in V^\sigma_k) \tag{3}$$

$$\sum_{i \in V^\sigma_k \cup \{0\}} y^k_{ij} = 1 \qquad (k \in \mathcal{R}^\sigma; j \in V^\sigma_k) \tag{4}$$

$$\sum_{j \in V^\sigma_k \cup \{n+1\}} y^k_{ij} = 1 \qquad (k \in \mathcal{R}^\sigma; i \in V^\sigma_k) \tag{5}$$

$$\sum_{j \in V^\sigma_k} y^k_{0j} \leq R^\sigma_k \qquad (k \in \mathcal{R}^\sigma) \tag{6}$$

$$y^k_{ij} \in \{0,1\} \qquad (k \in \mathcal{R}^\sigma; i,j \in V^\sigma_k \cup \{0, n+1\}) \tag{7}$$

For $i,j \in V^\sigma_k$, $y^k_{ij} = 1$ indicates that activity j is processed immediately after activity i on the same unit of resource k, which incurs a setup time of ϑ^k_{ij}, i.e., $\sigma_{ik} = \sum_{j \in V} j y^k_{ij}$.

Theorem 1 *A schedule T is process–feasible if and only if for each resource $k \in \mathcal{R}^\sigma$, there are values y^k_{ij} satisfying (3)– (7).*

Proof. Sufficiency: (6) implies that up to R^σ_k activities can be started on resource k. Let I denote the set of these initial activities. With $C_0 = 0$ and $\vartheta^k_{0i} = 0$ ($i \in V$), (3) holds for all activities $i \in I$. Due to $r^\sigma_{ik} \in \{0,1\}$ ($k \in \mathcal{R}^\sigma$, $i \in V$) we have $\sum_{i \in I} r^\sigma_{ik} \leq R^\sigma_k$. From (5) it follows that for each activity $i \in I$, there is exactly one activity j with $y^k_{ij} = 1$, and due to (3) j cannot start before the changeover between i and j has been completed. On the other hand, (4) says that an activity $j \notin I$ has exactly one direct predecessor $i \neq 0$. Due to (3), i and the changeover between i and j have to be completed before j starts on the processing unit i has been processed on, and no additional resource capacity is needed.
Necessity: Obvious. □

For a given schedule T and a resource $k \in \mathcal{R}^\sigma$, a feasible assignment for y^k_{ij} ($i,j \in V, k \in \mathcal{R}^\sigma$) can be found by solving the following assignment problem: We define a set of sources $V^R := V^\sigma_k \cup \{0_1, \ldots, 0_{R^\sigma_k}\}$ and a set of sinks $V^S := V^\sigma_k \cup \{(n+1)_1, \ldots, (n+1)_{R^\sigma_k}\}$. We introduce an edge $[i,j]$ between nodes i and j ($i \in V^R, j \in V^S$) if the time between the completion of activity i and the start of activity j is not less than the time ϑ^k_{ij} needed for the setup of the processing unit, i.e. if $S_j \geq C_i + \vartheta^k_{ij}$ holds. Additionally, we add edges between each node $0_r \in V^R$ ($r \in \{1, \ldots, R^\sigma_k\}$) and each node $j \in V^S$ as well as between each node $(n+1)_s \in V^S$ ($s \in \{1, \ldots, R^\sigma_k\}$) and each node $i \in V^R$. These edges ensure that each activity may be the first or the last activity that is processed on a processing unit or that a processing unit may be not used.

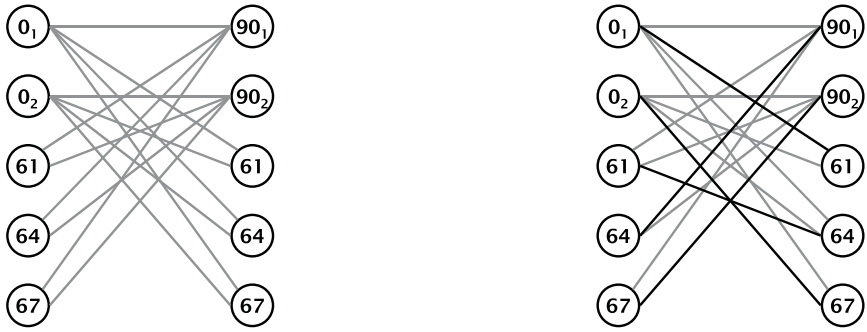

Fig. 4. Assignment problems for process–infeasible schedule T' and a process–feasible schedule

The edge weights do not affect the test of setup–feasibility. For a given schedule T, however, we can get an assignment of the activities to the individual processing units of a given type with minimal total setup time by choosing the values ϑ_{ij}^k as weights of the edges $[i, j]$ ($i \in V^R, j \in V^S$).

In our example, processing units $R1$ to $R9$ are modelled by setup resources $k \in \mathcal{R}^\sigma := \{2, \ldots, 8\}$. As we assumed processing units $R6$ and $R7$ to be identical, both are modelled by one resource $7 \in \mathcal{R}^\sigma$ with $R_7^\sigma = 2$. The same holds for processing units $R8$ and $R9$ which are gathered to resource $8 \in \mathcal{R}^\sigma$ with $R_8^\sigma = 2$. For activities 61, 64, and 67, the setup times on resource 7 are given as follows: $\vartheta_{61,64}^7 = \vartheta_{61,67}^7 = 5$, $\vartheta_{64,67}^7 = 6$, and $\vartheta_{ij}^7 = 0$ otherwise (cf. Table 3). The left graph in Fig. 4 depicts the corresponding assignment problem for schedule T'. As $C_j < S_i + \vartheta_{ij}^7$ for all $i, j \in \{61, 64, 67\}$, no edges are introduced between the corresponding nodes and thus the assignment problem is not solvable. By delaying the start of activity 64 up to $C_{61} + \vartheta_{61,64}^7 = 28$, the corresponding assignment problem gets solvable (cf. the right graph in Fig. 4).

Finally, we turn to *storages* that are modelled as so–called *cumulative resources* (cf. Neumann, 1999; Schwindt, 1998, 1999). A resource k is called cumulative if its availability results from previous utilization and is bounded from below (by a safety stock) and from above (by a capacity). In contrast to the well–known *nonrenewable resources* (cf. Brucker et al., 1999), an activity may deplete *and* replenish cumulative resources. Let $\mathcal{R}^\gamma$ denote the set of cumulative resources. For each resource $k \in \mathcal{R}^\gamma$, there are a prescribed minimum inventory $\underline{R}_k^\gamma \in \mathbb{Z} \cup \{-\infty\}$ (the safety stock), and a prescribed maximum inventory $\overline{R}_k^\gamma \in \mathbb{Z} \cup \{\infty\}$ (the storage capacity). For each activity $i \in V$, we have a *demand* $r_{ik}^\gamma \in \mathbb{Z}$ for resource $k \in \mathcal{R}^\gamma$. If $r_{ik}^\gamma > 0$, the demand is termed *replenishment* and occurs at time C_i; if $r_{ik}^\gamma < 0$, the demand is termed *depletion* and occurs at time S_i. r_{0k}^γ can be regarded as the initial stock of resource $k \in \mathcal{R}^\gamma$. For $n+1$, we have $r_{n+1,k}^\gamma = 0$ ($k \in \mathcal{R}^\gamma$).

With $V_k^{\gamma-} := \{i \in V \mid r_{ik}^\gamma < 0\}$ denoting the set of activities depleting resource $k \in \mathcal{R}^\gamma$ and $V_k^{\gamma+} := \{i \in V \mid r_{ik}^\gamma > 0\}$ being the set of activities replenishing resource $k \in \mathcal{R}^\gamma$, we define

$$\mathcal{A}_k^\gamma(T, t) := \{i \in V_k^{\gamma-} \mid S_i \leq t\} \cup \{i \in V_k^{\gamma+} \mid C_i \leq t\}$$

Table 6. Cumulative resource requirements

i	$1,\ldots,14$	$15,\ldots,26$	$27,\ldots,38$	$46,\ldots,54$	$58,59,60$
$r^{\gamma}_{i,10}$	10	-15	3	0	0
$r^{\gamma}_{i,13}$	0	0	7	-7	-10
$r^{\gamma}_{i,28}$	10	-15	10	-7	-10

to be the *active set* of activities that depleted or replenished resource $k \in \mathcal{R}^{\gamma}$ by time t for a given schedule T. A *storage–feasible schedule* satisfies

$$\underline{R}^{\gamma}_k \leq \sum_{i \in \mathcal{A}^{\gamma}_k(T,t)} r^{\gamma}_{ik} \leq \overline{R}^{\gamma}_k \qquad (k \in \mathcal{R}^{\gamma}; t \geq 0). \tag{8}$$

Obviously, $\underline{R}^{\gamma}_k \leq \sum_{i \in V} r^{\gamma}_{ik} \leq \overline{R}^{\gamma}_k$ for all $k \in \mathcal{R}^{\gamma}$ is a sufficient and necessary condition on the existence of a storage–feasible schedule (cf. Neumann, 1999).

Storages can be modelled as follows: A *homogeneous storage*, where a single product is stored, corresponds to a cumulative resource k with $\underline{R}^{\gamma}_k = 0$ provided that no safety stock is required and $\overline{R}^{\gamma}_k$ equals the storage capacity in product units. For $r^{\gamma}_{ik} < 0$, $-r^{\gamma}_{ik}$ coincides with the number of product units consumed by operation i. For $r^{\gamma}_{ik} > 0$, r^{γ}_{ik} gives the number of product units produced by operation i. A *heterogeneous storage*, where Π different products numbered from 1 to Π are stored (e.g. products $P2$ and $P5$ in the example in Section 2), can be modelled by $\Pi + 1$ cumulative resources $k_1, \ldots, k_{\Pi+1}$. Resource k_{π} is assigned to product π ($1 \leq \pi \leq \Pi$). Resource $k_{\Pi+1}$ ensures that the capacity of the storage is not exceeded. An operation i depleting or replenishing the heterogeneous storage by $-r_{ik_{\pi}}$ or $r_{ik_{\pi}}$ units of product $\pi \in \{1, \ldots, \Pi\}$, respectively, requires $r_{ik_{\pi}}$ units of resources k_{π} and $k_{\Pi+1}$. We have $\underline{R}^{\gamma}_{k_{\pi}} = 0$ if there are no safety stocks and $\overline{R}^{\gamma}_{k_{\pi}}$ equals the storage capacity in product units ($\pi \in \{1, \ldots, \Pi + 1\}$).

In our example, all products are stored in some storage facilities where products $P2$ and $P5$ share the same storage. Thus, we have 20 cumulative resources in set $\mathcal{R}^{\gamma} := \{9, \ldots, 28\}$. Resources 9 to 27 correspond to individual products $P1$ to $P19$ (cf. Table 2). Resource 28 keeps the total stock of products $P2$ and $P5$, and we have $\underline{R}^{\gamma}_{28} = 0$ and $\overline{R}^{\gamma}_{28} = 30$. Table 6 shows the demands for cumulative resources 10, 13 and 28 (for brevity, only activities using these resources are listed).

Let us assume that the inventories of products $P12$ and $P14$ at time 18 are 3 and 5, respectively. The start of activity 79 at time 22 causes a shortage of product $P12$ of two units. At the completion of activity 61 at time 23 seven units of $P12$ are added. With the completion of activity 67 at time 28, the inventory of product $P14$ is 12 units whereas the capacity of the corresponding storage is 10 units. At time 30, activity 86 starts and consumes 10 units of $P14$. Schedule T' is not storage–feasible (cf. Fig. 5).

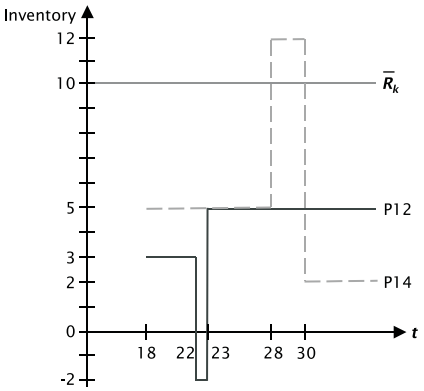

Fig. 5. Inventory levels

3.4 Break constraints

As mentioned in Section 2, some tasks may be interrupted by a break. An interrupted task has to be resumed immediately after the end of the break. Furthermore, we assume that material is left on the processing unit(s) during the break and that there is no setup needed before resuming the task. That implies that in particular it is not allowed to interrupt a task in order to start or proceed with another one.

A *break calendar* b_i of an activity $i \in V$ assigns a value $b_i(t) \in \{0, 1\}$ to all $t \geq 0$ with

$$b_i(t) = \begin{cases} 1, \text{if activity } i \text{ can be processed at time } t \\ 0, \text{otherwise.} \end{cases}$$

Without loss of generality, we assume that functions b_i are piecewise continuous.
The requirement that activities $i \in V^{ni}$ must not be interrupted can be formulated as follows:

$$\int_{S_i}^{C_i} (1 - b_i(t))dt = 0 \quad (i \in V^{ni}). \tag{9}$$

The processing time $p_i = C_i - S_i - \int_{S_i}^{C_i}(1 - b_i(t))dt$ of activity $i \in V$ must not be smaller than its minimum duration and must not be larger than its maximum duration, i.e.,

$$p_i^{min} \leq C_i - S_i - \int_{S_i}^{C_i} (1 - b_i(t))dt \leq p_i^{max} \quad (i \in V^{bi}). \tag{10}$$

Recall that for activities $i \in V^{ni}$, the corresponding inequalities $p_i^{min} \leq C_i - S_i \leq p_i^{max}$ have been expressed by minimum and maximum time lags between events i^S and i^C (see Subsection 3.2).

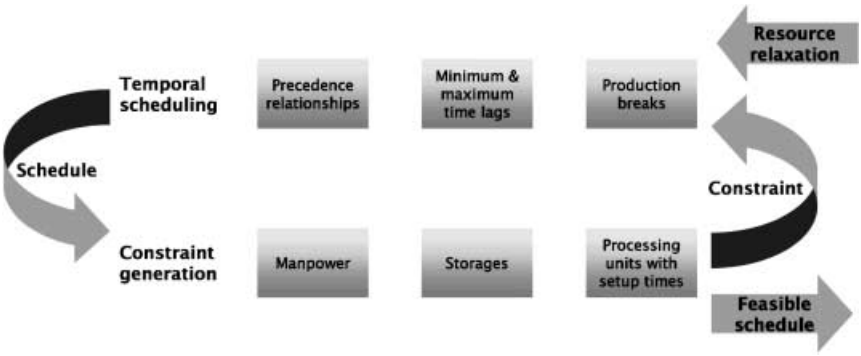

Fig. 6. Temporal scheduling and constraint generation for components of the batch scheduling problem

Equations

$$b_i(S_i) = 1 \quad (i \in V) \tag{11}$$

say that no activity can be simultaneously started and interrupted.

A schedule satisfying conditions (9), (10), and (11) is called *break–feasible*. A time–, manpower–, process–, storage–, and break–feasible schedule is called *feasible*.

4 Project scheduling

In this section, we briefly sketch the generation scheme of an (exact) branch–and–bound algorithm for solving the resource–constrained project scheduling problem constructed in Section 3. The constraints that make the problem hard are the scarcity of resources, i.e. the limited availability of manpower and processing units, the minimum inventory of intermediate products, and the capacity of storages. By relaxing the corresponding constraints (1), (2), and (8) we obtain a *temporal scheduling* problem which can be solved efficiently by a polynomial longest path algorithm of type label correcting, where additionally start and completion times are appropriately delayed until all calendarization constraints (9), (10), and (11) are met (cf. Franck, 1999; Trautmann, 1999). Violations of the resource constraints which occur in an optimal solution to this *resource relaxation* can be avoided by introducing appropriate precedence relationships between sets of events (*constraint generation*, cf. Fig. 6).

In what follows, we use the concept of *disjunctive precedence constraints* between sets of start and completion events. A disjunctive precedence constraint $A \to B$ between set A and set B says that no event $f \in B$ can take place before the occurrence of at least one of the events $e \in A$:

$$\min_{f \in B} T_f \geq \min_{e \in A} T_e. \tag{12}$$

Now let us assume that the schedule T under consideration is manpower–infeasible, i.e. $\sum_{i \in \mathcal{A}^\rho(T,t)} r_{ik}^\rho > R_k^\rho$ for some renewable resource $k \in \mathcal{R}^\rho$ at some time t. The conflict on the overloaded resource can be resolved by choosing set A in inequality (12) to be the set of completion events of an (inclusion–maximal) subset of the active set which can be executed simultaneously on resource k. Set B of delayed events is set to be $\mathcal{A}^\rho(T,t) \setminus A$. For the manpower conflict between activities 61, 64, 67, 79, and 91 at time 22 in schedule T', a corresponding constraint would be $S_{79} \geq \min(C_{61}, C_{64}, C_{67}, C_{91})$.

In case of an overloaded setup resource $k \in \mathcal{R}^\sigma$, we proceed analogously where (12) is replaced by $\min_{j^S \in B} T_{j^S} \geq \min_{i^C \in A}(T_{i^C} + \vartheta_{ij}^k)$. For details of how choosing sets A and B we refer to Trautmann (1999). The conflict between activities 61, 64, and 67 on processing units R6 and R7 (resource 7) illustrated in Fig. 4 can be resolved by introducing relationship $S_{64} \geq \min(C_{61} + 5, C_{67} + 5)$ delaying the start of activity 64 up to the earliest point in time where activity 61 or activity 67 is completed and resource 7 is set up for processing activity 64.

The cumulative resource constraints may be violated either by falling below the safety stock or by exceeding the storage capacity. Let us first consider the case where at some time $t \geq 0$ the inventory of a storage k falls below the safety stock. Then set A can be chosen to be an (inclusion–maximal) set of completion events which are *not* contained in the active set (i.e. which have not occurred by time t) and which belong to activities replenishing resource k. For set B, we have an inclusion–minimal subset of the start events of depleting activities $j \in \mathcal{A}_k^\gamma(T,t)$ with $\sum_{j \in \mathcal{A}_k^\gamma(T,t) \setminus B} r_{jk} \geq \underline{R}_k$.

The case where at time t the inventory exceeds the capacity of storage k can be dealt with as follows. We choose A to be an (inclusion–maximal) set of start events depleting resource k and occuring after time t. Accordingly, set B is an inclusion–minimal subset of the set of completion events of replenishing activities of the active set with $\sum_{j \in \mathcal{A}_k^\gamma(T,t) \setminus B} r_{jk} \leq \overline{R}_k$.

We consider the example shown in Fig. 5. The shortage of product $P12$ can be avoided by introducing inequality $S_{79} \geq C_{61}$ and thus delaying the withdrawal of five units of $P12$ up to the completion of activity 61 producing $P12$. The stock excess of product $P14$ can be resolved by delaying the completion of activity 67 up to the start of activity 86, i.e. $C_{67} \geq S_{86}$.

The temporal scheduling (TS) problem consists of finding a time– and break–feasible schedule with minimum project duration satisfying a given set of disjunctive precedence constraints. An optimal solution to problem TS can be determined by a fixed point algorithm which exploits the existence of a unique componentwise minimal schedule observing the temporal, the break, and the disjunctive precedence constraints (cf. Schwindt, 1998; Franck, 1999; Trautmann, 1999).

The schedule generation scheme for the resource–constrained project scheduling problem is depicted in Fig. 7. The first TS problem coincides with the resource relaxation. If it is solvable, we compute a time– and break–feasible schedule T minimizing C_{n+1}. If T is manpower–infeasible, process–infeasible, or storage–infeasible, we determine appropriate sets of events A and B as described above and add the corresponding disjunctive precedence constraint $A \to B$ to problem TS (constraint generation). Temporal scheduling and the constraint generation are

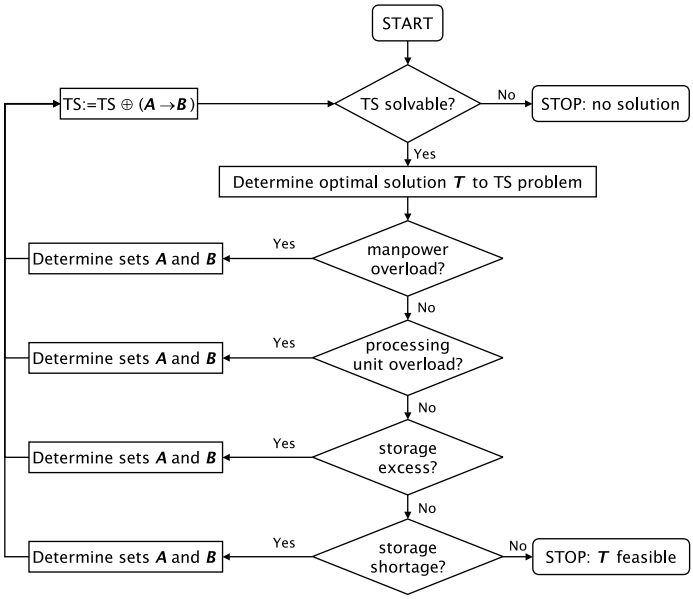

Fig. 7. Generation scheme

repeated until either the TS problem does no longer possess a feasible solution or the computed optimal solution T represents a feasible schedule.

A branch–and–bound algorithm based on the generation scheme of Fig. 7 has been implemented in C under MS–Visual C++ 6.0. Batch sizes for task 4 of the case study by Westenberger and Kallrath (1995) with the original primary requirements have been determined by the batching heuristic mentioned in Section 2 (cf. Table 1). The resulting batch scheduling problem corresponds to the example described in Section 2 without manpower pools, production breaks, variable processing times, and heterogeneous storages, but with alternative non–identical processing units (cf. Sect. 5). This batch scheduling problem has been (approximately) solved using a truncated beam search version of the branch–and–bound algorithm. On a Pentium II–333 PC with 128 MB memory operating under MS–Windows NT 4.0, we obtained a makespan of 92 units of time within one minute. Blömer (1999) and Blömer and Günther (1998, 1999, 2000) obtained feasible solutions for variants of this instance with different primary requirements. Some variants could be solved to optimality. Relaxations of this instance have been treated by Ahleff (1995), Burkhard et al. (1998b), and Rosenau (1996). For the batch scheduling problem described in Section 2, we got a makespan of 143 units of time within one minute.

5 Extensions

Often tasks can be carried out in several *alternative modes* which differ in processing times, production breaks, and manpower, processing unit, and storage demand. In that case, we have to model the processing of the operations as a *multi–mode*

resource–constrained project. In addition to the model described in Section 3, the use of multiple modes enables to deal with the following settings:

- Use of alternative non–identical processing units. In the example of Section 2, we assumed alternative processing units to be identical. In the case study of Westenberger and Kallrath (1995), the alternative processing units $R6$, $R7$ and $R8$, $R9$ are not identical but differ in processing and setup times. Thus, for each operation that requires one of these processing units, two alternative modes have to be defined that differ in duration and resource requirements.
- Often, practical applications necessitate to distinguish manpower of different skills or experiences, e.g. if there are tasks that an engineer completes within half of the time needed by a trainee.
- Alternative storages for identical input or output materials at different locations require alternative modes for the corresponding consuming/producing operations.
- Additionally, alternative storages allow to model alternative product structures, e.g. the use of alternative input materials.

Batch scheduling then corresponds to the problem of determining a start time, a completion time, and a mode for each operation such that all given constraints are observed and the makespan C_{n+1} is minimized. A corresponding model and a solution procedure which is based on the approach by Heilmann (2000) can be found in Trautmann (1999). Other approaches for multi–mode resource–constrained project duration problems can be found in De Reyck and Herroelen (1999), Franck (1999), Kolisch (1995), Sprecher (1994), and Sprecher and Drexl (1998).

Especially in the case of make–to–order production, the main emphasis is generally put on meeting *due dates* for the delivery of customer orders rather than on minimizing the makespan. These delivery dates correspond to due dates d_i of activities $i \in V$. Let $\max(0, C_i - d_i)$ be the *tardiness* of activity $i \in V$. The total tardiness $\sum_{i \in V} \max(0, C_i - d_i)$ or the maximum tardiness $\max_{i \in V} \max(0, C_i - d_i)$ are further objectives which have been dealt with in literature. Sometimes it is expedient to consider a non–negative linear combination of these functions. All these objective functions belong to the class of regular functions, that is, the objective function value does not decrease with increasing completion times C_i ($i \in V$) (cf. Neumann et al., 2000; Sprecher et al., 1995). An important non–regular objective function is the sum of sequence–dependent setup times [see Trautmann (1999) for an approach that is based on the model presented in Section 3].

6 Conclusions and further research

We have modelled batch scheduling problems in process industries within the framework of resource–constrained project scheduling. The proposed solution procedure is based on the relaxation of the constraints limiting the availability of manpower, processing units, the safety stock constraints, and the storage capacity constraints. This relaxation corresponds to the problem of minimizing the makespan subject to minimum and maximum time lags and break calendars. In the course of a branch–and–bound algorithm, infeasibilities occurring in the resulting schedule

are resolved by introducing precedences between sets of conflicting operations. Task 4 of the benchmark instance by Westenberger and Kallrath (1995) has been solved to feasibility by a simple batching heuristic and the subsequent solution of the corresponding batch scheduling problem by a truncated version of the branch–and–bound algorithm within one minute on a personal computer.

In practice, input data like processing times, material, or manpower availability are often subject to change (cf. Kanakamedala et al., 1994) and thus the production schedule may become infeasible. *Reactive planning* aims at finding a feasible schedule while keeping the system nervousness as small as possible, i.e. a feasible schedule sharing a maximum number of properties (e.g. start times or operation sequences) with the original schedule. Using the presented relaxation–based approach, additional constraints arising from altered input data can easily be integrated.

Another important area of future research is the generalization to the modelling of continuous or half–continuous material flow by relaxing the assumption of infinite depletion and replenishment rates of the storages. In contrast to batch mode production, here the problem of finding appropriate operation delays arises when resolving cumulative resource infeasibilities.

Appendix
List of symbols

$\mathcal{A}_k^\gamma(T,t)$	set of activities that have used resource $k \in \mathcal{R}^\gamma$ by time $t \geq 0$
$\mathcal{A}^\rho(T,t)$	set of activities being processed at time $t \geq 0$
$b_i(t)$	indicates whether or not activity i can be in progress at time t
C_i	completion time of activity i
d_i	due date of activity i
$\overline{d}_i$	deadline of activity i
d_{ef}^{min}	minimum time lag between events $e, f \in \widetilde{V}$
d_{ef}^{max}	maximum time lag between events $e, f \in \widetilde{V}$
δ_{ef}	arc weight between nodes $e, f \in \widetilde{V}$
E	set of arcs
i^C	completion event of activity i
i^S	start event of activity i
n	number of operations / activities
$n+1$	production end
p_i	duration of activity i
p_i^{max}	maximum duration of activity i
p_i^{min}	minimum duration of activity i
q_i	quarantine time of activity i
r_i	release date of activity i

r_{ik}^{γ}	inventory of resource $k \in \mathcal{R}^{\gamma}$ consumed or replenished by activity i
r_{ik}^{ρ}	usage of resource $k \in \mathcal{R}^{\rho}$ by activity i
r_{ik}^{σ}	usage of resource $k \in \mathcal{R}^{\sigma}$ by activity i
$\mathcal{R}^{\gamma}$	set of cumulative resources
$\mathcal{R}^{\rho}$	set of renewable resources
$\mathcal{R}^{\sigma}$	set of setup resources
$\underline{R}_k^{\gamma}$	minimum inventory of resource $k \in \mathcal{R}^{\gamma}$
$\overline{R}_k^{\gamma}$	maximum inventory of resource $k \in \mathcal{R}^{\gamma}$
R_k^{ρ}	capacity of resource $k \in \mathcal{R}^{\rho}$
R_k^{σ}	capacity of resource $k \in \mathcal{R}^{\sigma}$
s_i	shelf–life time of activity i
S_i	start time of activity i
σ_{ik}	direct successor of activity i on the same unit of setup resource $k \in \mathcal{R}^{\sigma}$
ϑ_{ij}^{k}	setup time between activities i and j on a unit of setup resource $k \in \mathcal{R}^{\sigma}$
T	schedule
T_e	occurrence time of event $e \in \widetilde{V}$
V	set of activities
V'	set of dummy activities
$\widetilde{V}$	set of events
V^{bi}	set of interruptible activities
V^{ni}	set of non–interruptible activities
$V_k^{\gamma-}$	set of activities consuming inventory of resource $k \in \mathcal{R}^{\gamma}$
$V_k^{\gamma+}$	set of activities replenishing inventory of resource $k \in \mathcal{R}^{\gamma}$
V_k^{σ}	set of activities using resource $k \in \mathcal{R}^{\sigma}$
0	production start

References

Ahleff J (1995) Analyse von mathematischen Modellen zur Reihenfolgeplanung in der Prozeßindustrie. Master thesis, RWTH Aachen

Allweyer T, Loos P, Scheer AW (1994) An empirical study on scheduling in the process industries. Heft 109, Institut für Wirtschaftsinformatik, Universität Saarbrücken

Aquilano NJ, Smith DE (1980) A formal set of algorithms for project scheduling with critical path scheduling / material requirements planning. Journal of Operations Management 1: 57–67

Applequist G, Samikoglu O, Pekny J, Reklaitis GV (1997) Issues in the use, design and evolution of process scheduling and planning systems. ISA Transactions 36: 81–121

Artiba A, Tahon C (1992) Production planning knowledge–based systems for pharmaceutical manufacturing lines. European Journal of Operational Research 61: 18–29

Bartusch M, Möhring R, Radermacher FJ (1988) Scheduling project networks with resource constraints and time windows. Annals of Operations Research 16: 201–240

Benoy M, Dewilde P, Voet M, Herroelen W (1994) Finite scheduling: State–of–the–market. Ernst & Young Management Consultants, Brussels

Birewar DB, Grossmann IE (1990) Simultaneous production planning and scheduling in multiproduct batch plants. Industrial & Engineering Chemistry Research 29: 570–580

Blömer F (1999) Produktionsplanung und -steuerung in der chemischen Industrie – Ressourceneinsatzplanung von Batchprozessen auf Mehrzweckanlagen. Gabler, Wiesbaden

Blömer F, Günther HO (1998) Scheduling of a multi–product batch process in the chemical industry. Computers in Industry 36: 245–259

Blömer F, Günther HO (1999) Numerical evaluation of MILP models for scheduling chemical batch processes. Discussion paper 1999/05, Technical University of Berlin

Blömer F, Günther HO (2000) LP–based heuristics for scheduling chemical batch processes. International Journal of Production Research 38: 1029–1051

Brucker B, Drexl A, Möhring R, Neumann K, Pesch E (1999) Resource–constrained project scheduling: notation, classification, models, and methods. European Journal of Operational Research 112: 3–41

Brucker P, Hurink J (1999) Solving a chemical batch scheduling problem by local search. Osnabrücker Schriften zur Mathematik, Heft 215, University of Osnabrück

Burkhard RE, Hujter M, Klinz B, Rudolf R, Wennink M (1998a) A process scheduling problem arising from chemical production planning. Optimization Methods & Software 10: 175–196

Burkhard RE, Kocher M, Rudolf R (1998b) Rounding strategies for mixed integer programs arising from chemical production planning. Yugoslav Journal of Operations Research 8: 9–23

Cartwright HM, Long A (1993) Simultaneous optimization of chemical flowshop sequencing and topology using genetic algorithms. Industrial & Engineering Chemistry Research 32: 2706–2713

Das H, Cummings PT, LeVan MD (1990) Scheduling of serial multiproduct batch processes via simulated annealing. Computers & Chemical Engineering 14: 1351–1362

Dedopoulos IT, Shah N (1995) Optimal short–term scheduling of maintenance and production for multipurpose plants. Industrial & Engineering Chemistry Research 34: 192–201

Demeulemeester, EL, Herroelen, WS (1996) Modelling setup times, process batches and transfer batches using activity network logic. European Journal of Operational Research 89: 355–365

De Reyck B, Herroelen W (1998) A branch–and–bound procedure for the resource–constrained project scheduling problem with generalized precedence constraints. European Journal of Operational Research 111: 152–174

De Reyck B, Herroelen W (1999) The multi–mode resource–constrained project scheduling problem with generalized precedence relations. European Journal of Operational Research 119: 538–556

Dessouky YM, Roberts CA, Dessouky MM, Wilson G (1996) Scheduling multi–purpose batch plants with junction constraints. International Journal of Production Research 34: 525–541

Dimitriadis AD, Shah N, Pantelides CC (1997) RTN–based rolling horizon algorithms for medium term scheduling of multipurpose plants. Computers & Chemical Engineering S21: S1061-S1066

Dorndorf U, Pesch E, Phan Huy T (1998) A time–oriented branch–and–bound algorithm for project scheduling with generalized precedence constraints. Technical Report, Department BWL III, University of Bonn

Fortemps P, Ost C, Pirlot M, Teghem J, Tuyttens D (1996) Using metaheuristics for solving a production scheduling problem in a chemical firm – a case study. International Journal of Production Economics 47: 13–26

Franck B (1999) Prioritätsregelverfahren für die ressourcenbeschränkte Projektplanung mit und ohne Kalender. Shaker, Aachen

Frauendorfer K, Königsperger E (1996) Concepts for improving scheduling decisions: an application in the chemical industry. International Journal of Production Economics 46–47: 27–38

Günther HO (1992) Netzplanorientierte Auftragsterminierung bei offener Fertigung. OR Spektrum 14: 229–240

He DW, Kusiak A, Artiba A (1996) A scheduling problem in glass manufacturing. IIE Transactions 28: 129–139

Heilmann R (2000) Ressourcenbeschränkte Projektplanung im Mehr–Modus–Fall. Gabler, Wiesbaden

Kallrath J, Wilson JM (1997) Business optimisation using mathematical programming. Macmillan Press, Houndmills

Kanakamedala KB, Reklaitis G, Venkatasubramanian V (1994) Reactive schedule modification in multipurpose batch chemical plants. Industrial & Engineering Chemistry Research 33: 77–90

Kim M, Jung JH, Lee IB (1996) Optimal scheduling of multiproduct batch processes for various intermediate storage policies. Industrial & Engineering Chemistry Research 35: 4058–4066

Kolisch R (1995) Project scheduling under resource constraints: efficient heuristics for several problem classes. Physica, Heidelberg

Kondili E, Pantelides CC, Sargent RWH (1993) A general algorithm for short–term scheduling of batch operations – I. MILP Formulation. Computers & Chemical Engineering 17: 211–227

Ku HM, Karimi I (1990) Completion time algorithms for serial multiproduct batch processes with shared storage. Computers & Chemical Engineering 14: 49–69

Kudva G, Elkamel A, Pekny JF, Reklaitis GV (1994) Heuristic algorithm for scheduling batch and semi–continuous plants with production deadlines, intermediate storage limitations and equipment changeover costs. Computers & Chemical Engineering 18: 859–875

Lawler EL (1976) Combinatorial optimization: networks and matroids. Holt, Rinehart, and Winston, New York

Mockus L, Reklaitis GV (1997) Mathematical programming formulation for scheduling of batch operations based on nonuniform time discretization. Computers & Chemical Engineering 21: 1147–1156

Möhring R, Stork F, Uetz M (1998) Resource constrained project scheduling with time windows: a branching scheme based on dynamic release dates. Technical Report 596/1998, Department of Mathematics, Technische Universität Berlin

Moon S, Park S, Lee WK (1996) New MILP models for scheduling of multiproduct batch plants under zero–wait policy. Industrial & Engineering Chemistry Research 35: 3458–3469

Neumann K (1999) Project scheduling with limited cumulative resources: modelling and basic concepts and results. Proceedings of IEPM'99, Glasgow

Neumann K, Nübel H, Schwindt C (2000) Active and stable project scheduling. Mathematical Methods of Operations Research (to appear)

Neumann K, Schwindt C (1997) Activity–on–node networks with minimal and maximal time lags and their application to make–to–order production. OR Spektrum 19: 205–217

Neumann K, Zimmermann J (1999) Methods for the resource–constrained project scheduling problem with regular and nonregular objective functions and schedule–dependent time windows. In: Węglarz J (ed) Project scheduling: recent models, algorithms, and applications, pp 261–287. Kluwer, Boston

Pekny JF, Miller DL, Kudva GK (1993) An exact algorithm for resource constrained sequencing with application to production scheduling under an aggregate deadline. Computers & Chemical Engineering 17: 671–682

Pinto M, Grossmann IE (1996) An alternate MILP model for short term scheduling of batch plants with preordering constraints. Industrial & Engineering Chemistry Research 35: 338–342

Pinto M, Grossmann IE (1998) Assignment and sequencing models for the scheduling of process systems. Annals of Operations Research 81: 443–466

Reklaitis GV (1996) Overview of scheduling and planning of batch process operations. In: Reklaitis GV, Sunol AK, Rippin, DWT, Hortaçsu Ö (eds) Batch processing systems engineering, pp 660–705. Springer, Berlin Heidelberg New York

Rippin DWT (1993) Batch process systems engineering: A retrospective and prospective review. Computers & Chemical Engineering 17: S1–S13

Rosenau B (1996) Bearbeitung eines Losgrößenproblems aus der chemischen Industrie mit Schnittebenenenverfahren. Diploma thesis, University of Marburg

Schilling G, Pantelides CC (1996) A simple continuous–time process scheduling formulation and a novel solution algorithm. Computers & Chemical Engineering 20: S1221–S1226

Schürbüscher D, Metzner W, Lempp P (1992) Besondere Anforderungen an die Produktionsplanung und -steuerung in der chemischen und pharmazeutischen Industrie. Chemie–Ingenieur–Technik 64: 334– 341

Schwindt C (1998) Verfahren zur Lösung des ressourcenbeschränkten Projektdauerminimierungsproblems mit planungsabhängigen Zeitfenstern. Shaker, Aachen

Schwindt C (1999) A branch–and–bound algorithm for the project duration problem subject to temporal and cumulative resource constraints. Proceedings of IEPM'99, Glasgow

Shah N, Pantelides CC, Sargent RWH (1993a) A general algorithm for short–term scheduling of batch operations – II. Computational issues. Computers & Chemical Engineering 17: 229–244

Shah N, Pantelides CC, Sargent RWH (1993b) Optimal periodic scheduling of multi–purpose batch plants. Annals of Operations Research 42: 193–228

Sprecher A (1994) Resource–constrained project scheduling – exact methods for the multi–mode case. Springer, Berlin Heidelberg New York

Sprecher A, Drexl A (1998) Multi–mode resource–constrained project scheduling by a simple, general, and powerful sequencing algorithm. European Journal of Operational Research 107: 431–450

Sprecher A, Kolisch R, Drexl A (1995) Semi–active, active, and non–delay schedules for the resource–constrained project scheduling problem. European Journal of Operational Research 80: 94–102

Trautmann N (1999) Batch scheduling in process industries – an application of resource–constrained project scheduling. Report WIOR–557, Institut für Wirtschaftstheorie und Operations Research, University of Karlsruhe

Westenberger H, Kallrath J (1995) Formulation of a job shop problem in process industry. Unpublished working paper, Bayer AG, Leverkusen and BASF AG, Ludwigshafen

Zentner MG, Reklaitis GV (1996) An interval–based mathematical model for the scheduling of resource–constrained batch chemical processes. In: Reklaitis GV, Sunol AK, Rippin, DWT, Hortaçsu Ö (eds) Batch processing systems engineering, pp 779–807. Springer, Berlin Heidelberg New York