J Sched (2009) 12: 595–606
DOI 10.1007/s10951-009-0107-y

A cyclic approach to large-scale short-term planning in chemical batch production

Norbert Trautmann · Christoph Schwindt

Received: 31 October 2007 / Accepted: 9 February 2009 / Published online: 19 March 2009

Abstract We deal with the scheduling of processes on a multi-product chemical batch production plant. Such a plant contains a number of multi-purpose processing units and storage facilities of limited capacity. Given primary requirements for the final products, the problem consists in dividing the net requirements for the final and the intermediate products into batches and scheduling the processing of these batches. Due to the computational intractability of the problem, the monolithic MILP models proposed in the literature can generally not be used for solving large-scale problem instances. The cyclic solution approach presented in this paper starts from the decomposition of the problem into a batching and a batch-scheduling problem. The complete production schedule is obtained by computing a cyclic subschedule, which is then repeated several times. In this way, good feasible schedules for large-scale problem instances are found within a short CPU time.

Keywords Applications · Large-scale scheduling · Process scheduling · Production scheduling

1 Introduction

In a chemical production plant, value is added to materials by successive transformation tasks such as mixing, separating, forming, or chemical reactions. We consider the case of batch production on multi-purpose equipment, which is usually chosen in make-to-order production when the product range spans over different product families with relatively small demand rates. As each product family requires a specific plant configuration, a high resource utilization and short customer lead times can only be achieved by minimizing the production makespan.

To process a batch, the inputs are first loaded into a processing unit, then the task is executed, and finally the output is unloaded from the processing unit; if subsequently a different task is executed in the processing unit, it needs to be cleaned before. Since the batch sizes are limited by the capacities of the processing units, each task is generally executed several times. The storage space, the availability, and sometimes also the shelf life of the intermediate products are limited. Given the primary requirements for a set of final products from the same product family, the short-term planning problem studied in this paper consists in computing a feasible production schedule with minimum makespan. As it has been noted by Honkomp et al. (2000), the size of industrial process scheduling problems is often a challenging issue for commercial scheduling systems. Therefore, we focus on problem instances comprising several thousands of task executions (called operations in what follows).

In this paper, we propose a cyclic approach that consists of the three phases of cyclic batching, cyclic batch-scheduling, and concatenation. In the *cyclic batching phase*, the set of operations belonging to one cycle together with the respective batch sizes and input and output proportions, and the number of cycles needed to satisfy the given primary requirements are determined. In order to keep the scheduling problem tractable, the total number of operations per cycle is limited. We formulate this cyclic batching problem as a mixed-integer nonlinear program whose size is independent of the primary requirements and for which locally optimal

N. Trautmann (✉)
Department of Business Administration, University of Bern, 3012 Bern, Switzerland
e-mail: norbert.trautmann@pqm.unibe.ch

C. Schwindt
Institute of Management and Economics, Clausthal University of Technology, 38678 Clausthal-Zellerfeld, Germany

solutions can be determined using standard software. In the *cyclic batch-scheduling* phase, we compute a subschedule by allocating the processing units, intermediates, and storage facilities over time to the operations belonging to one cycle. We present a new priority-rule based method for this problem. In the *concatenation* phase, a complete production schedule is generated by concatenating the cyclic subschedules. We show how to extract feasibility-preserving precedence relationships among the operations from the cyclic subschedule and how to generate the complete schedule in polynomial time by exploiting these relationships.

Each of the three phases of the method is performed only once. In practice, the hardness of a process scheduling problem is essentially determined by the number of operations to be scheduled, whereas the numbers of tasks, products, and processing units are typically rather small. Since the size of the cyclic batching problem is independent of the primary requirements and hence independent of the number of operations and because we can explicitly limit the size of the cyclic batch-scheduling problem to be solved, only the concatenation phase is affected by the primary requirements. Since the concatenation can be performed very efficiently, the cyclic approach enables us to shift the high dimensional part of the problem from the NP-hard scheduling problem to the polynomially solvable concatenation problem. In total, a relatively short computation time is required, and we are able to efficiently cope with large-scale problem instances. In an experimental performance analysis, we applied this cyclic approach to a set of 70 test instances, including instances with more than 3000 operations. For each instance, we obtained a better feasible solution within much less CPU time than the method of Gentner et al. (2004), which is the only reference in the open literature reporting on tests with instances of the latter dimension. From a theoretical point of view, the cyclic approach operates on a set of production schedules that is a proper subset of the set of all feasible schedules. This is due to the required cyclicity of the subschedule and the concatenation structure of the complete schedule. Our computational experience with the method indicates that the limitation to cyclic schedules only induces a minor loss of generality. The reason is that the cyclic batching problem is formulated in a flexible way allowing tasks to be executed a variable number of times in each cycle. On the other hand, the cyclic approach provides a framework within which large-scale process scheduling problems become tractable.

The remainder of this paper is organized as follows. The short-term planning problem discussed in this paper is introduced in Sect. 2. In Sect. 3, we review the related literature. In Sect. 4, the cyclic batching problem is formulated as a mixed-integer nonlinear program, and in Sect. 5 we present a conceptual model of the batch-scheduling problem. In Sect. 6, we explain how we construct the cyclic subschedule and how we compute the complete production schedule by concatenating flexible copies of this subschedule. The results of the experimental performance analysis are discussed in Sect. 7. Section 8 is devoted to concluding remarks and directions for further research.

2 Short-term planning in chemical batch production

In Sect. 2.1, we review the particular characteristics of chemical batch production on multi-product plants. In Sect. 2.2, we state the short-term planning problem. Section 2.3 introduces a practical example of a chemical batch production plant that has been provided by Kallrath (2002).

2.1 Chemical batch production on multi-product plants

A multi-product plant consists of multi-purpose processing units like heaters, filters, and reactors and storage facilities such as tanks, silos, or cooling houses. The final products are produced through a sequence of tasks on the processing units. The duration of a task is generally considered to be independent of the batch size. For executing a task, several alternative processing units may be available. In this case, the processing time of the operation may depend on the processing unit used. A multi-purpose processing unit can operate several tasks, but only one operation at a time. Between consecutive operations performed on the same processing unit, a cleaning with sequence-dependent duration may be necessary.

Each task consumes and produces one or several products. The proportions of the input products and the proportions of the output products of a task may be either fixed or variable within prescribed bounds. Certain substances, like chemically instable intermediates, are perishable and must be consumed immediately after production. Material flows can be linear, divergent, convergent, or general, including the case of recycling flows. The minimum and maximum filling levels of the processing unit used give rise to lower and upper bounds on the batch size. That is why in difference to continuous production, in batch production a task generally has to be executed several times to obtain a desired amount of output products.

A multi-purpose batch plant is often able to produce final products belonging to different product families. Each product family, however, requires a specific configuration of the plant. During a re-configuration of the plant, it is not possible to process any operation. Hence, it is important to achieve a small production makespan for the processing of the products of each family to ensure a high resource utilization and short customer lead times. The short-term planning problem considered in the following refers to the processing of one product family on a plant with a given configuration.

For simplicity, we assume that the nonperishable intermediate products are stocked in dedicated storages and that the storage capacities for the final products allow for stocking the total production amount of one product family.

2.2 Short-term planning problem

The short-term planning problem can be stated as follows. Given primary requirements for the final products, we must determine (1) the batch size, the input and the output proportions, and the number of executions for each task, (2) an assignment of the corresponding operations to the processing units, and (3) the start times of the operations in such a way that

– the given primary requirements for the final products are satisfied,
– the prescribed intervals for the batch sizes and the input and output proportions are observed,
– no processing unit executes more than one operation at a time,
– the processing units are cleaned between consecutive operations if necessary,
– a sufficient amount of each input product is available at the start of each operation,
– sufficient storage space for each output product is available at the completion of each operation,
– all perishable intermediates are consumed immediately after production, and
– the makespan is minimized.

2.3 Sample production process

In this section we describe the chemical batch production process of the case study presented by Kallrath (2002), which is based on an existing plant. For the representation we use the state-task network (STN) concept that has been introduced by Kondili et al. (1993). An STN is a bipartite directed graph which comprises three types of elements:

1. *State nodes* represent the raw materials, intermediates, and final products. They are drawn as ellipses labeled with the respective state number.
2. *Task nodes* are identified with the tasks transforming one or more input states into one or more output states. Task nodes are represented by rectangles indicating the task number.
3. *Arcs* correspond to the flow of material. If more than one input product is consumed or more than one output product is produced, the arcs are labeled with the feasible values of the input or output proportions.

Figure 1 shows the STN for the Kallrath case study with 19 products, 17 tasks, and 9 processing units.

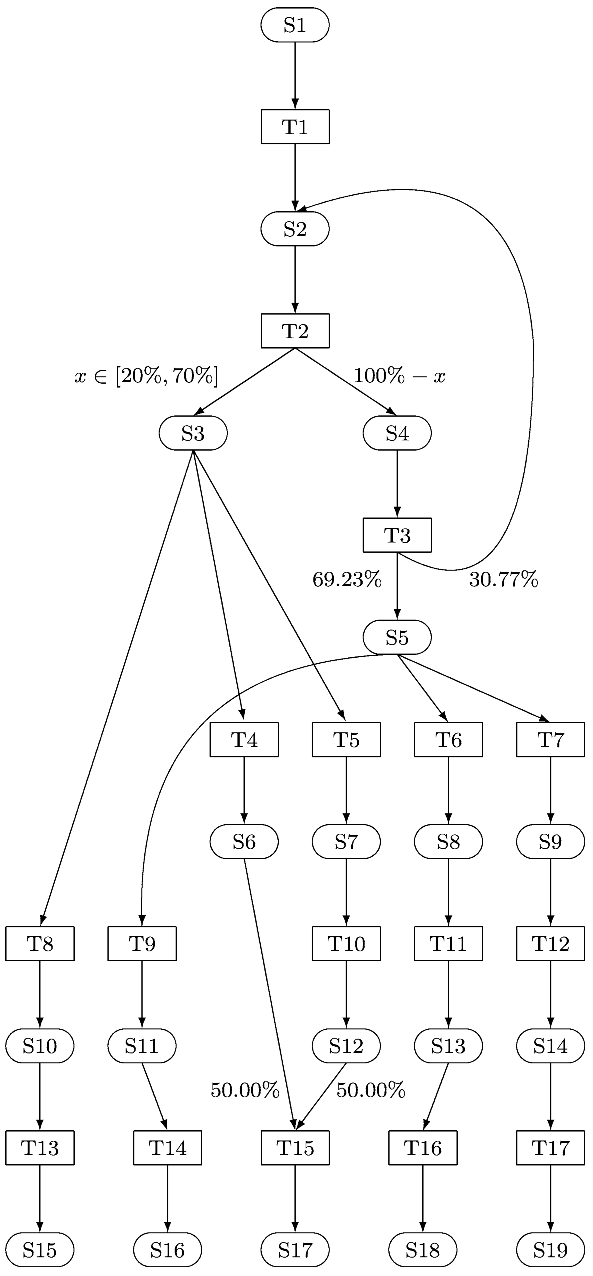

Fig. 1 State-task network of the sample production process

Table 1 lists the initial and maximum stocks of the products. Some of the intermediate products cannot be stocked, which is indicated by a maximum stock of 0. The value ∞ for the initial or maximum stock means that there is sufficient initial stock or storage capacity available.

Table 2 provides the intervals of feasible batch sizes for each processing unit. Table 3 displays the processing units in which the tasks can be executed and the corresponding processing times. Alternative processing units are available for executing tasks 10 to 14, 16, and 17.

The tasks are numbered in the order of their quality requirements. In order to guarantee the purity of the final prod-

Table 1 Initial and maximum stocks

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19
Initial stock	∞	20	20	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Maximum stock	∞	30	30	15	30	0	10	10	10	0	0	10	0	10	∞	∞	∞	∞	∞

Table 2 Minimum and maximum batch sizes

	U1	U2	U3	U4	U5	U6	U7	U8	U9
Minimum batch size	3	5	4	4	4	3	3	4	4
Maximum batch size	10	20	10	10	10	7	7	12	12

Table 3 Processing units and processing times

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17
Processing units	U1	U2	U3	U4	U4	U4	U4	U5	U5	U6/U7	U6/U7	U6/U7	U8/U9	U8/U9	U8	U8/U9	U8/U9
Processing times	4	8	4	8	8	8	8	12	12	8/10	10/12	12/12	8/12	8/12	8	12/12	12/12

ucts, each processing unit must be cleaned before passing to an operation with a higher task index. The time needed for cleaning a processing unit equals one half of the processing time of the preceding operation.

3 Related literature

In the context of supply chain management, the time-phased primary requirements to be produced in the production network are determined on the mid-term campaign planning level. Campaign planning aims at using the procurement, production, storage, and transportation facilities in the supply chain efficiently by coordinating the material flows in the network. Mixed-integer linear programming models for campaign planning can for example be found in Timpe and Kallrath (2000) and Grunow et al. (2002).

The short-term planning problem described in Sect. 2.2 has been widely discussed in the chemical engineering literature. An overview of state-of-the-art models and methods can be found in the survey papers of Floudas and Lin (2004), Burkard and Hatzl (2005), and Méndez et al. (2006). Roughly speaking, monolithic approaches (cf. Sect. 3.1) and decomposition approaches (cf. Sect. 3.2) can be distinguished.

3.1 Monolithic approaches

The monolithic solution approaches address the short-term planning problem as a whole, starting from a formulation as a mixed-integer linear program. The time horizon is divided into a given number of time intervals. In the so-called discrete-time formulations (see, e.g., Kondili et al. 1993), each interval corresponds to a time period of fixed length. In contrast, in the so-called continuous-time formulations (see, e.g., Ierapetritou and Floudas 1998 or Castro et al. 2001), the period length is chosen implicitly during the solution of the mixed-integer linear program.

The main disadvantage of all these monolithic approaches is that the CPU time required for solving real-world problems tends to be prohibitively large (cf. Maravelias and Grossmann 2004). To overcome this difficulty, Shah et al. (1993a), Blömer and Günther (2000), and others have developed different heuristics that aim at reducing the number of variables. Nevertheless, the computational burden for solving real-world problems with more than 50 operations is still very high. Whereas the case studies considered in the chemical engineering literature generally contain less than 100 operations, industrial process scheduling systems have to cope with problem instances comprising several thousands of operations.

Shah et al. (1993b), Castro et al. (2003), and Wu and Ierapetitrou (2004) address a related, but less complex process scheduling problem, where a cyclic schedule is computed, which is constantly repeated within a planning horizon of given length. The problem of determining the cyclic schedule and the number of its repetitions is tackled by monolithic approaches that are similar to the approaches mentioned above.

3.2 Decomposition approaches

Promising alternative approaches are based on decomposing the short-term planning problem into interdependent subproblems. Decomposition methods have for example been proposed by Brucker and Hurink (2000), Neumann et al.

(2002), Gentner et al. (2004), and Maravelias and Grossmann (2004).

In the procedure of Brucker and Hurink (2000), the numbers and the sizes of the batches are computed using a constructive algorithm, and the operations are scheduled on the processing units with a tabu search procedure. The authors do not consider all of the constraints mentioned in Sect. 2.1. In particular, they assume that the capacities of the storage facilities are unlimited and that each task can only be executed on one dedicated processing unit.

Maravelias and Grossmann (2004) propose to compute the numbers of batches by solving the LP relaxation of a monolithic continuous-time formulation of the short-term planning problem. For given numbers of batches, the batch sizes and the start times of the operations are determined by a branch-and-bound algorithm that uses constraint-propagation techniques.

Similarly to the approach by Brucker and Hurink, Neumann et al. (2002) decompose the problem hierarchically into a batching problem and a batch-scheduling problem. The solution of the batching problem, which can be formulated as a mixed-integer nonlinear program, provides the numbers and the sizes of all batches needed to satisfy the primary requirements. The batch-scheduling problem consists in allocating the processing units, intermediates, and storage facilities over time to the processing of the operations arising from the batching step. Neumann et al. (2002) and Schwindt and Trautmann (2004) present a branch-and-bound method and a two-phase priority-rule based method, respectively, for solving the batch-scheduling problem. A new single-phase priority-rule based method is described in Schwindt et al. (2007). The main features of the latter method are explained in Sect. 6.1. Within a reasonable amount of computation time, good feasible solutions to problem instances with up to 100 operations can be computed with all three scheduling methods.

Gentner et al. (2004) devise a decomposition of the batch-scheduling problem which partitions the set of all batches into a sequence of subsets. The assignment of the batches to the individual subsets is determined stepwise by solving a binary linear program in each iteration. Gentner et al. (2004) and Gentner (2005) computed feasible solutions to batch-scheduling instances with several thousand operations. To the best of our knowledge, their method is the only one available so far for solving such large-scale short-term planning problems. However, the required computation time still considerably increases with growing primary requirements.

The cyclic approach to be discussed in the following sections is capable to cope with large-scale instances and runs very fast and accurately. The good performance of the new approach is due to the fact that first, the NP-hard scheduling problem is tackled for a small subset of all operations only, and second, the complete production schedule is constructed in polynomial time based on the resulting subschedule. We note that a preliminary version of our approach is described in Schwindt and Trautmann (2006).

3.3 Related work in scheduling theory

In this paper, we deal with a cyclic approach to the scheduling of batch processes on multi-purpose processing units subject to constraints on the inventory levels of intermediate products. We complete this section by briefly reviewing three concepts from classical scheduling theory that are related to our work: scheduling with batching, cyclic scheduling, and scheduling with inventory constraints.

3.3.1 Scheduling with batching

In machine scheduling, a batch designates a set of jobs that have to be executed jointly. The completion time of all jobs in a batch are considered to be equal to the completion time of the entire batch. Before processing a job, the machine has to be set up, which requires a setup time that is independent of the job sequence and the batches. Depending on the way in which the duration of a batch is defined, one can distinguish between serial and parallel batching (s-batching and p-batching) problems. We speak of an s-batching problem if the duration of a batch arises from summing up the durations of all jobs. Consequently, the jobs of a batch are executed one after another. In a p-batching problem, the duration of the batch equals the largest processing time of a job in that batch, which means that all jobs of a batch are processed in parallel. In this case, the machine is called a batching machine. If the number of jobs in a batch is bounded from above, the p-batching problem is called bounded. Potts and Kovalyov (2000) provide a review on scheduling problems involving batching decisions, with a special emphasis on the design of efficient dynamic programming methods. Algorithms and complexity results for a large variety of serial and parallel single machine batching problems can be found in Brucker (2004), Chap. 8.

Processing units are similar to batching machines in that a processing unit can batchwise process a certain amount of materials and the processing time of a batch is independent of the batch size chosen. There are, however, fundamental differences between the classical p-batching problem and the batching problem occurring in short-term planning. The combinatorial nature of the former problem arises from the tradeoff between total setup and completion times. The latter batching problem essentially consists in deciding on the optimum continuous input and output proportions and batch sizes and in computing the required number of batches for each task.

3.3.2 *Cyclic scheduling*

Cyclic or periodic scheduling is concerned with steady state scheduling problems where the schedule is to be repeated over an infinite planning horizon. Cyclic schedules are typically implemented in repetitive manufacturing environments like assembly lines in mass production where the primary requirement rates are sufficiently stable over time. The objective usually consists in finding a cyclic schedule with minimum cycle time. The cycle time is equal to the time between the start of the first job in a cycle and the start of this job in the next cycle, and when the schedule is repeated infinitely many times, minimizing the cycle time is equivalent to maximizing the throughput rate. Compared to an acyclic schedule, a cyclic schedule offers the advantage of an easier shop floor control. That is why in practice, cyclic schedules are sometimes even used as baseline schedules in build-to-order production, from which order-dependent acyclic schedules are obtained by allowing minor deviations (see Pinedo 2002, Sect. 16.2). An overview of cyclic scheduling problems and related complexity results is given by McCormick and Rao (1994).

In our approach, we want to find a cyclic subschedule that is executed a finite number of times such that given primary requirements are satisfied and the makespan of the complete production schedule is minimum. This means that we solve an acyclic scheduling problem by constructing a cyclic subschedule whose repetition is terminated after a certain time. In this way, we combine the relative simplicity of computing a small cyclic subschedule with the more realistic assumptions of an acyclic scheduling problem.

3.3.3 *Scheduling with inventory constraints*

One challenge of the scheduling problem considered in this paper consists in the material-availability and the storage-capacity constraints that have to be observed for the intermediate products. A special case of those inventory constraints has been studied extensively in shop floor scheduling, where scheduling problems with finite job buffers have been considered. When the buffer capacities are limited, it may happen that machines get blocked because their output buffers are full. The storage facilities of a chemical production plant can be viewed as limited buffers that are replenished and depleted by general amounts of materials. Scheduling problems with general inventory constraints modeling material-availability and storage-capacities conditions have been considered in the context of project scheduling (see, e.g., Beck 2002, Neumann and Schwindt 2002, and Laborie 2003). It can be shown that the material-availability constraints can be expressed as storage-capacity constraints and vice versa. Furthermore, Neumann and Schwindt (2002) have shown by transformation from 3-PARTITION that the problem of finding a feasible schedule subject to inventory constraints and chain precedence relations is NP-hard for the case of a single storage facility. As a consequence, the feasibility variant of the short-term planning problem under consideration is also NP-hard and there does not exist any polynomial-time approximation algorithm with bounded performance ratio for the problem unless $\mathcal{P} = \mathcal{NP}$.

4 Cyclic batching problem

In the batching phase of our cyclic approach, we translate the given primary requirements into the set of operations belonging to one cycle together with their respective batch sizes and input and output proportions, and the number of cycles needed to satisfy the given primary requirements. Each task may be executed several times in a cycle, the numbers of executions generally being different from task to task. Moreover, in order to keep the scheduling problem tractable, we impose an upper bound on the total number of operations per cycle. To obtain a cyclic solution allowing for executing the same subschedule an arbitrary number of times, the amount of any intermediate produced within one cycle must be equal to the amount consumed.

In the following, we assume that we assign the same batch size and the same input and output proportions to all operations belonging to the same task. Accordingly, four types of quantities have to be determined:

– the batch size of each task,
– the input and output proportions of each task,
– the number of operations of each task in the cycle, and
– the number of replications of the cycle.

We formulate the batching problem as a mixed-integer nonlinear program, which can be solved using standard software for mathematical programming. The size of the model only depends on the number of tasks and the number of products and is consequently independent of the primary requirements and the number of operations to be scheduled. Let $\mathcal{T}$ be the set of all tasks and $\mathcal{P}$ be the set of all products under consideration. By $\mathcal{P}_\tau^-$ and $\mathcal{P}_\tau^+$ we denote the sets of input and output products of task $\tau \in \mathcal{T}$. The input and output proportions and the batch size of task τ are associated with the continuous decision variables $\alpha_{\tau\pi}$ and β_τ, respectively. We establish the convention that $\alpha_{\tau\pi} < 0$ for all input products $\pi \in \mathcal{P}_\tau^-$ and that $\alpha_{\tau\pi} > 0$ for all output products $\pi \in \mathcal{P}_\tau^+$ of task τ. For each task $\tau \in \mathcal{T}$, the following mass balance constraints ensure that the input products are completely transformed into the output products and that the output proportions sum up to 100 %:

$$-\sum_{\pi \in \mathcal{P}_\tau^-} \alpha_{\tau\pi} = \sum_{\pi \in \mathcal{P}_\tau^+} \alpha_{\tau\pi} = 1 \quad (\tau \in \mathcal{T}). \tag{1}$$

Now recall that proportions $\alpha_{\tau\pi}$ and batch sizes β_τ have to be chosen within prescribed intervals, say $[\underline{\alpha}_{\tau\pi}, \overline{\alpha}_{\tau\pi}]$ and $[\underline{\beta}_\tau, \overline{\beta}_\tau]$, which leads to inequalities

$$\underline{\alpha}_{\tau\pi} \le \alpha_{\tau\pi} \le \overline{\alpha}_{\tau\pi} \quad \left(\tau \in \mathcal{T},\ \pi \in \mathcal{P}_\tau^- \cup \mathcal{P}_\tau^+\right) \tag{2}$$

and

$$\underline{\beta}_\tau \le \beta_\tau \le \overline{\beta}_\tau \quad (\tau \in \mathcal{T}). \tag{3}$$

Let $\mathcal{T}_\pi^-$ and $\mathcal{T}_\pi^+$ be the sets of all tasks consuming and producing, respectively, product $\pi \in \mathcal{P}$ and let $\mathcal{P}^p \subset \mathcal{P}$ denote the set of perishable products. Then equations

$$\alpha_{\tau\pi}\beta_\tau = -\alpha_{\tau'\pi}\beta_{\tau'} \quad \left(\pi \in \mathcal{P}^p,\ \tau \in T_\pi^+,\ \tau' \in T_\pi^-\right) \tag{4}$$

guarantee that the amount of product $\pi \in \mathcal{P}^p$ produced by an operation of some task $\tau \in \mathcal{T}_\pi^+$ can immediately be consumed by an operation of any task $\tau' \in \mathcal{T}_\pi^-$ consuming product π. We note that depending on the bounds on the input and output proportions and on the batch sizes, there may exist feasible production schedules for which constraint (4) is not satisfied. In this case, however, several operations producing product π have to be completed simultaneously or several operations consuming π have to be started at the same time. In general, the latter conditions would significantly reduce the optimization space at the batch-scheduling level.

Now let ε_τ be the integer decision variable providing for task $\tau \in \mathcal{T}$ the number of operations to be executed in one cycle. By $\mathcal{P}^i \subset \mathcal{P}$ we denote the set of intermediates. In order to obtain a cyclic solution, which allows us to execute the same subschedule an arbitrary number of times, the amount of an intermediate π produced within one cycle must be equal to the amount of π consumed, i.e.,

$$\sum_{\tau \in \mathcal{T}_\pi^- \cup \mathcal{T}_\pi^+} \alpha_{\tau\pi}\beta_\tau\varepsilon_\tau = 0 \quad \left(\pi \in \mathcal{P}^i\right). \tag{5}$$

Moreover, we have to determine the number $\xi \in \mathbb{Z}_{\ge 0}$ of cycles needed to satisfy the given primary requirements for final products. Let ρ_π be the primary requirement less the initial stock of raw material or final product $\pi \in \mathcal{P} \setminus \mathcal{P}^i$. The final inventory of product π must be sufficiently large to match the primary requirements for π, i.e.,

$$\xi \sum_{\tau \in \mathcal{T}_\pi^- \cup \mathcal{T}_\pi^+} \alpha_{\tau\pi}\beta_\tau\varepsilon_\tau \ge \rho_\pi \quad \left(\pi \in \mathcal{P} \setminus \mathcal{P}^i\right). \tag{6}$$

The complexity of the resulting cyclic batch-scheduling problem mainly depends on the number of operations to be scheduled. To ensure that the problem remains tractable, the following inequality imposes an upper bound $\overline{\varepsilon}$ on the total number of operations per cycle:

$$\sum_{\tau \in \mathcal{T}} \varepsilon_\tau \le \overline{\varepsilon}. \tag{7}$$

Eventually, we formulate the objective function. Recall that our goal is to compute a feasible production schedule with minimum makespan. Therefore, at the batching level, we want to minimize the total workload $\xi \sum_{\tau \in \mathcal{T}} \overline{p}_\tau \varepsilon_\tau$ to be scheduled at the batch-scheduling level, where $\overline{p}_\tau$ denotes the mean processing time of task τ on the alternative processing units. In sum, the cyclic batching problem (C-BP) reads as follows:

$$\text{(C-BP)} \quad \begin{cases} \text{Minimize} & \xi \sum_{\tau \in \mathcal{T}} \overline{p}_\tau \varepsilon_\tau \\ \text{subject to} & (1) \text{ to } (7), \\ & \varepsilon_\tau \in \mathbb{Z}_{\ge 0} \quad (\tau \in \mathcal{T}), \\ & \xi \in \mathbb{Z}_{\ge 0}. \end{cases}$$

Problem (C-BP) represents a mixed-integer nonlinear program with the integer decision variables ξ and ε_τ and the continuous decision variables β_τ and $\alpha_{\tau\pi}$ ($\tau \in \mathcal{T}, \pi \in \mathcal{P}$). Thus, the total number of variables is independent of the primary requirements and, for typical production processes, rather small. For example, the instance of problem (C-BP) belonging to the sample production process presented in Sect. 2 contains 18 integer and 18 continuous decision variables. Computational experiments have shown that locally optimal solutions can be determined using commercial standard software within short CPU times. By applying a transformation described by Neumann et al. (2002), problem (C-BP) can be converted into a mixed-binary linear program of larger, but still moderate size.

5 Cyclic batch-scheduling problem

Solving the cyclic batching problem provides us with a set $\mathcal{O}$ of $n = \sum_{\tau \in \mathcal{T}} \varepsilon_\tau$ operations $i = 1, \dots, n$ belonging to one cycle. The cyclic batch-scheduling problem then consists in

- selecting a processing unit and
- computing a feasible start time

for each operation such that the makespan of the cyclic subschedule is minimized.

Let $\mathcal{U}$ be the set of all processing units and $\mathcal{U}_i$ be the set of those alternative units on which operation i can be carried out. For $i \in \mathcal{O}$ and $k \in \mathcal{U}_i$, the binary decision variable x_{ik} indicates whether or not operation i is assigned to unit k ($x_{ik} = 1$ or $x_{ik} = 0$, respectively). Each operation i must be executed on exactly one processing unit, i.e.,

$$\sum_{k \in \mathcal{U}_i} x_{ik} = 1 \quad (i \in \mathcal{O}). \tag{8}$$

Vector $x = (x_{ik})_{i \in \mathcal{O}, k \in \mathcal{U}_i}$ is referred to as an *assignment* of operations i to processing units k.

Let $S_i \ge 0$ be the start time of operation i. The pair (S, x) of start time vector $S = (S_i)_{i \in \mathcal{O}}$ and assignment x is called a *(sub-)schedule*. By $p_i(x)$ and $c_{ij}(x)$ we denote the processing time of operation i and the changeover time from operation i to operation j given assignment x. Moreover, let $\mathcal{O}_k(x) = \{i \in \mathcal{O} \mid x_{ik} = 1\}$ designate the set of all operations i being executed on unit k. Schedule (S, x) is called *process-feasible* if no two operations i and j are executed in parallel on a processing unit, i.e.,

$$\big[S_i, S_i + p_i(x) + c_{ij}(x)\big[\cap \big[S_j, S_j + p_j(x) + c_{ji}(x)\big[= \emptyset \quad \big(k \in \mathcal{U},\ i, j \in \mathcal{O}_k(x) : i < j\big). \tag{9}$$

Now we turn to the storage facilities. We assume that each product $\pi \in \mathcal{P}$ is stocked in a dedicated storage facility of capacity σ_π, where $\sigma_\pi = 0$ for the perishable products $\pi \in \mathcal{P}^p$, which cannot be stored. The demand $\rho_{i\pi}$ of an operation $i \in \mathcal{O}$ belonging to some task $\tau \in \mathcal{T}$ for storage capacity is equal to $\alpha_{\tau\pi}\beta_\tau$. By $\mathcal{O}_\pi^- = \{i \in \mathcal{O} \mid \rho_{i\pi} < 0\}$ and $\mathcal{O}_\pi^+ = \{i \in \mathcal{O} \mid \rho_{i\pi} > 0\}$ we denote the sets of operations consuming or producing, respectively, product π. Operations $i \in \mathcal{O}_\pi^-$ deplete the inventory of product π at their start time S_i, whereas operations $i \in \mathcal{O}_\pi^+$ replenish the inventory of product π at their completion time $S_i + p_i(x)$. Schedule (S, x) is said to be *storage-feasible* if no operation is started before all input products are available or completed before the output products can be stocked in the storage facilities. With $\rho_{0\pi}$ denoting the initial stock of product π, the material-availability and storage-capacity constraints can be formulated as

$$0 \le \rho_{0\pi} + \sum_{i \in \mathcal{O}_\pi^+ : S_i + p_i(x) \le t} \rho_{i\pi} + \sum_{i \in \mathcal{O}_\pi^- : S_i \le t} \rho_{i\pi} \le \sigma_\pi \quad (\pi \in \mathcal{P},\ t \ge 0). \tag{10}$$

A process- and storage-feasible schedule (S, x) satisfying (8) is called *feasible*. The cyclic batch-scheduling problem (C-BSP) consists in finding a feasible (sub-)schedule (S, x) with minimum makespan:

$$\text{(C-BSP)} \begin{cases} \text{Minimize} & \max_{i \in \mathcal{O}}(S_i + p_i(x)) \\ \text{subject to} & (8) \text{ to } (10), \\ & S_i \ge 0 \quad (i \in \mathcal{O}), \\ & x_{ik} \in \{0, 1\} \quad (i \in \mathcal{O},\ k \in \mathcal{U}_i). \end{cases}$$

6 Generating the production schedule

The production schedule is generated in two steps. In the first step, which is dealt with in Sect. 6.1, we compute a cyclic subschedule solving the cyclic batch-scheduling problem (C-BSP) defined in Sect. 5. In the second step, based on the cyclic subschedule we efficiently construct a complete production schedule for all cycles. This concatenation step is explained in Sect. 6.2.

6.1 Computation of the cyclic subschedule

In this section, we explain the main principle of the priority-rule based method of Schwindt et al. (2007). In a preprocessing phase of the algorithm, we at first construct an operation-on-node network. Each operation $i \in \mathcal{O}$ corresponds to one node of the network, and vice versa. The nodes are connected by arcs representing temporal constraints between the operations. Those temporal constraints arise from necessary conditions that must be observed between operations belonging to the same task or between operations producing and consuming the same intermediates. For example, assume that we have arranged the operations of a task in some arbitrary order. We define a minimum time lag between the starts of any two consecutive operations in the sequence. If the task can only be processed on one unit, the minimum time lags coincide with the duration of the task. Otherwise, the operations may overlap in time, and the minimum time lags are chosen to be equal to zero. Further temporal constraints can be generated by exploiting the input-output relationships between the tasks. If an intermediate is produced by exactly one task, we can identify minimum time lags that are necessary to the timely availability of the input materials. To this end, we separately consider each task consuming the intermediate. Starting with the first operation of this task, we calculate how many operations of the producing task must be completed before a sufficient amount of the intermediate is available to start the consuming operation. We then add a minimum time lag between the last required producing operation and the consuming operation, the time lag being equal to the duration of the producing operation. Taking into account the residual stock of the intermediate, we proceed analogously with the remaining operations of the consuming task. If an intermediate is consumed by exactly one task, we can add temporal constraints to avoid capacity overflows in a similar way, where perishable products are associated with a fictitious storage facility of capacity zero. The last step of the preprocessing phase consists in computing the set of all strong components of the network. Since any two nodes of a strong component are mutually linked by temporal constraints, it proves advantageous to schedule all operations belonging to the same strong component consecutively.

The basic principle of the scheduling method is as follows. In each iteration we schedule one eligible operation i, which is selected based on some priority values. The operations of a strong component are eligible to be scheduled if (1) all of those operations' predecessors in the network outside the strong component have already been scheduled, (2) there is enough input material available to process all operations of the strong component, and (3) there is no other strong component for which some but not all operations have been scheduled. Condition (3) ensures that all operations of a strong component are added to the schedule in successive iterations. For the selected operation i we then determine the earliest point in time t for which the temporal constraints of the network are satisfied, the operation and the necessary changeovers can be performed on one of the alternative processing units $k \in \mathcal{U}_i$, and a sufficient amount of input materials is available. Then, operation i is scheduled at time t on processing unit k, i.e., we put $x_{ik} := 1$ and $S_i := t$. The storage-capacity constraints are considered via capacity-driven provisional latest start times. At the completion time $t' = S_i + p_i(x)$ of operation i it may happen that the inventory of some output product π of operation i exceeds the respective storage capacity σ_π. In this case, in the next iterations we temporarily force eligible operations consuming product π to start no later than time t'. These capacity-driven latest start times are maintained until the capacity overflow has been removed. As a consequence it may happen that an eligible operation can no longer be scheduled because the capacity-driven latest start time is smaller than the earliest feasible start time. If no eligible operation can be selected, the current partial schedule (S, x) cannot be expanded to a feasible schedule. To break such a deadlock, we perform an unscheduling step by introducing a release date $r_i := t' - p_i(x)$ for the operation i that caused the capacity overflow and restart the priority-rule based method from scratch. The new release date ensures that in the next pass of the algorithm, operation i is scheduled at a time $S_i \geq r_i$ that does not prevent all eligible operations j from being started at a time $S_j \geq t'$. If a given maximum number of unscheduling steps were performed without having found a feasible solution, the method is terminated. Otherwise, we obtain a feasible schedule after a finite number of iterations. The unscheduling step may generate unnecessary idle times, which are removed from the schedule in a postprocessing phase, where we compute the earliest schedule satisfying all the schedule-induced precedence relationships. By randomly varying the priority values of the operations, the priority-rule based method can be used as a multi-pass procedure providing a set of feasible subschedules.

6.2 Construction of the production schedule

In the concatenation phase, the complete production schedule is generated in the following way. The computed subschedule (S, x) for executing the operations of one cycle defines a partial ordering among those operations. We represent this ordering by precedence relationships between the operations. Moreover, the last operations in a cycle give rise to release dates for the first operations of the next cycle. The start and completion times for the operations in the first cycle equal those of the subschedule computed in the cyclic batch-scheduling phase. To determine the start and completion times of the operations in the next cycle, we compute an earliest schedule for these operations subject to the precedence relationships between and the release dates for the operations. This temporal scheduling problem represents a longest path problem, which can be solved efficiently by a standard label-correcting algorithm. Thus, the concatenation of the cyclic subschedules forming the complete production schedule can be performed in polynomial time.

In detail, we proceed as follows. The subschedule (S, x) computed by the priority-rule based method induces precedence relationships between the operations i, j of one cycle being executed on the same processing unit or producing and consuming the same product. Those precedence relationships are translated into minimum start-to-start time lags δ_{ij} defining the temporal constraints

$$S_j - S_i \geq \delta_{ij}. \tag{11}$$

Inequalities (11) ensure that the schedule remains feasible when left- or right-shifting the operations. More precisely, for each pair of operations (i, j) with $S_j \geq S_i + p_i(x) + c_{ij}(x)$ and $x_{ik} = x_{jk} = 1$ for some $k \in \mathcal{U}$ we introduce the time lag $\delta_{ij} = p_i(x) + c_{ij}(x)$, which prevents the overlapping of i and j. For each pair (i, j) with $S_j \geq S_i + p_i(x)$ and $\rho_{i\pi} > 0$, $\rho_{j\pi} < 0$ for some $\pi \in \mathcal{P}$, we add the time lag $\delta_{ij} = p_i(x)$, which guarantees that at any point in time a sufficient amount of intermediate π is available. Finally, we define the time lag $\delta_{ij} = -p_j(x)$ for each pair (i, j) with $S_j + p_j(x) \geq S_i$ and $\rho_{i\pi} < 0$, $\rho_{j\pi} > 0$ for some $\pi \in \mathcal{P}$, to avoid any excess of the storage capacity of product π.

Moreover, the completion time of the last operation i that is executed on a processing unit defines a release date $r_j = S_i + p_i(x) + c_{ij}(x)$ for the first operation j in that unit in the next execution of the subschedule. Analogously, the last change in the inventory level of an intermediate π, say at time t, gives rise to a release date r_j for the first operation j that subsequently produces or consumes units of π, where $r_j = t$ if $j \in \mathcal{O}_\pi^-$ and $r_j = t - p_j(x)$ if $j \in \mathcal{O}_\pi^+$. The release dates r_j can easily be transformed into temporal constraints of type (11) by introducing a fictitious operation $i = 0$ for the production start with $S_0 = 0$ and putting $\delta_{0j} := r_j$.

The operations of each cycle are assigned to the processing units according to the assignment x of subschedule (S, x). The start times of the operations in the first cycle equal those of subschedule (S, x). The start times of the operations in the next cycle are obtained by solving a temporal scheduling problem which consists in calculating an earliest schedule for these operations subject to the schedule-induced precedence relationships and the release dates introduced. As it is well-known (see, e.g., Neumann et al. 2003, Sect. 1.3), this temporal scheduling problem can be solved efficiently by longest path calculations in the operation-on-node network belonging to the corresponding temporal constraints (11). By iteratively concatenating the ξ cycles in this way, we finally obtain the complete production schedule.

Eventually we show that the production schedule is feasible. At first we notice that, since the same assignment x of operations to processing units is applied to each cycle, the minimum time lags and release dates introduced ensure that no two operations are executed in parallel on a processing unit and that the necessary cleaning times are observed. By construction of subschedule (S, x), the material-availability and storage capacity constraints are satisfied for the first cycle. From condition (5) of problem (C-BP), it follows that after the execution of the subschedule, the inventory levels of all intermediates coincide with the respective initial inventory levels. The minimum time lags and release dates defined between the operations producing or consuming a common product imply that the sequence of start and completion times of those operations remain unchanged in all cycles. As a consequence, for each intermediate the maximum and the minimum inventory levels over time coincide in each cycle, which shows that the complete schedule also respects the material-availability and storage-capacity conditions.

7 Experimental performance analysis

We compared our cyclic approach to the decomposition method devised by Gentner et al. (2004). To evaluate the impact of the cyclic formulation of the batching problem we also included a variant of the procedure of Neumann et al. (2002) where like for the cyclic approach the batch scheduling is performed with the priority-rule based method of Sect. 6.1. For our tests, we used a test set introduced by Gentner (2005), which consists of 70 instances generated by varying the primary requirements for the final products in the example presented in Sect. 2.3. For each instance we computed a feasible solution to the cyclic batching problem using Frontline Systems' Solver package. We performed the tests on an 3.4 GHz Pentium IV PC. The results for the method of Gentner et al. were taken from Gentner (2005) and refer to a 1.4 GHz Pentium IV PC. The priority-rule based method was stopped after 60 seconds of CPU time.

The results obtained for the 70 problem instances are shown in Table 4, where "$C_{\max}$" stands for the best makespan found, "t_{cpu}" is the CPU time in seconds, and "# op.'s" designates the number of operations in the complete production schedule obtained with the cyclic method. When compared to the method of Gentner, the new method is able to find a markedly better solution for each problem instance. In addition, especially for large-scale problem instances, the required computation time is significantly smaller. Having prescribed an upper bound of $\overline{\varepsilon} = 150$ batches, between 14 and 583 seconds are required for solving the cyclic batching problem. The concatenation always takes less than one second of CPU time. The results obtained for the method of Neumann et al. demonstrate the usefulness of the cyclic decomposition. For the smallest instances the Neumann method is able to find the schedules with the best makespans. This is due to the fact that in difference to the cyclic model, the initial inventories are available to be consumed. With increasing size of the instances, however, the quality of the schedules found decreases, and for the larger instances the priority-rule based method is no longer able to find any feasible schedule within the time limit of 60 seconds.

8 Conclusions

In this paper, we have presented a cyclic approach to short-term planning in chemical batch production. Our method starts from the decomposition of the short-term planning into a batching level providing the set of operations to be executed and a batch-scheduling level at which the operations are scheduled on the processing units subject to material-availability and storage-capacity constraints. The main idea of our cyclic approach consists in formulating the batching problem as a cyclic model where the given primary requirements are produced through the repetitive execution of the same set of operations. In this way, we ensure that the resulting batch-scheduling problem can be solved within a reasonable amount of computation time by computing a subschedule for the operations of one cycle and concatenating the number of cycles needed to meet the primary requirements. The performance of the new method has been tested on a test set involving instances with several thousands of operations.

A further improvement of the cyclic method can be obtained by treating the last cycle separately. From the cyclic structure of the subschedule it follows that, at the completion of the last cycle, the inventory levels of the intermediates coincide with the initial levels. In order to make use of the initial inventories, the cyclic batching problem can be expanded by decision variables for the input and output proportions, the batch sizes, and the numbers of task executions of the last cycle. This variant of the cyclic method would prove advantageous in cases where the initial inventories could meet a significant part of the total requirements.

An important area of our future research will be the adaptation of the cyclic approach to continuous process scheduling problems where tasks are executed at constant production rates. In addition, we are developing predictive-reactive methods for the short-term planning of chemical production plants when processing times, resource availabilities, or production yields are subject to uncertainty. The different short-term planning methods will be integrated with decision models for mid-term multi-site campaign planning in the chemical industry.

Table 4 Computational results

Instance	G. (2005)		N. et al. (2002)		This paper		
	$C_{\max}$	t_{cpu}	$C_{\max}$	t_{cpu}	# op.'s	$C_{\max}$	t_{cpu}
WeKa0_0	178	18	122	61	88	128	116
WeKa0_1	352	38	235	62	176	252	113
WeKa0_2	474	53	388	62	264	376	118
WeKa0_3	612	120	–	–	352	500	119
WeKa0_4	738	209	–	–	440	624	115
WeKa0_5	906	178	–	–	528	748	122
WeKa0_6	1046	215	–	–	616	872	119
WeKa0_7	1199	323	–	–	704	996	121
WeKa0_8	1334	281	–	–	792	1120	117
WeKa0_9	1548	399	–	–	880	1244	128
WeKa0_10	1740	431	–	–	968	1368	100
WeKa0_15	2123	644	–	–	1408	1988	97
WeKa0_20	2899	1500	–	–	1848	2608	97
WeKa0_30	4416	5235	–	–	2728	3884	77
WeKa19_0	238	19	162	62	105	166	80
WeKa19_1	436	165	343	61	210	316	81
WeKa19_2	618	59	–	–	315	466	79
WeKa19_3	818	97	–	–	420	616	80
WeKa19_4	1004	179	–	–	525	766	81
WeKa19_5	1184	232	–	–	630	916	80
WeKa19_6	1384	330	–	–	735	1066	83
WeKa19_7	1570	474	–	–	840	1216	81
WeKa19_8	1806	442	–	–	945	1366	81
WeKa19_9	1946	568	–	–	1050	1516	80
WeKa19_10	2135	570	–	–	1155	1666	83
WeKa19_15	2848	1322	–	–	1680	2416	79
WeKa19_20	3811	1911	–	–	2205	3166	78
WeKa19_30	5896	6610	–	–	3255	4666	76
WeKa20_0	168	34	134	64	89	138	86
WeKa20_1	336	50	282	64	178	264	87
WeKa20_2	590	72	446	61	267	390	90
WeKa20_3	750	76	640	62	356	516	100
WeKa20_4	896	93	–	–	445	642	98
WeKa20_5	990	126	–	–	534	768	100
WeKa20_6	1138	184	–	–	623	894	95
WeKa20_7	1294	215	–	–	712	1020	95
WeKa20_8	1547	200	–	–	801	1146	96
WeKa20_9	1816	327	–	–	890	1272	94
WeKa20_10	1920	448	–	–	979	1398	94
WeKa20_15	2386	421	–	–	1424	2028	96
WeKa20_20	3604	969	–	–	1869	2658	96
WeKa20_30	5194	3255	–	–	2759	3918	75
WeKa21_0	210	17	136	61	98	144	103
WeKa21_1	382	127	–	–	196	284	100
WeKa21_2	555	67	482	61	294	424	95
WeKa21_3	728	97	640	60	392	564	91
WeKa21_4	868	152	–	–	490	704	86
WeKa21_5	1082	226	–	–	588	844	86
WeKa21_6	1224	250	–	–	686	984	83
WeKa21_7	1420	240	–	–	784	1124	82
WeKa21_8	1554	291	–	–	882	1264	85
WeKa21_9	1701	475	–	–	980	1404	85
WeKa21_10	1916	469	–	–	1078	1544	82
WeKa21_15	2545	771	–	–	1568	2244	81
WeKa21_20	3398	1415	–	–	2058	2944	82
WeKa21_30	5091	5957	–	–	3038	4344	89
WeKa22_0	190	192	144	61	102	152	327
WeKa22_1	376	85	–	–	204	290	644
WeKa22_2	558	102	–	–	306	428	298
WeKa22_3	722	120	–	–	408	566	155
WeKa22_4	930	249	–	–	510	704	239
WeKa22_5	1024	239	–	–	612	842	324
WeKa22_6	1298	255	–	–	714	980	270
WeKa22_7	1488	341	–	–	816	1118	150
WeKa22_8	1520	439	–	–	918	1256	276
WeKa22_9	1779	427	–	–	1020	1394	149
WeKa22_10	1786	647	–	–	1122	1532	221
WeKa22_15	2586	704	–	–	1632	2222	171
WeKa22_20	3172	1598	–	–	2142	2912	206
WeKa22_30	5375	7563	–	–	3162	4292	271

Acknowledgements This research has been supported by the Deutsche Forschungsgemeinschaft under Grant Schw1178/1.

References

Beck, J. C. (2002). Heuristics for scheduling with inventory: dynamic focus via constraint criticality. *Journal of Scheduling*, *5*, 43–69.

Blömer, F., & Günther, H. O. (2000). LP-based heuristics for scheduling chemical batch processes. *International Journal of Production Research*, *38*, 1029–1051.

Brucker, P. (2004). *Scheduling algorithms*. Berlin: Springer.

Brucker, P., & Hurink, J. (2000). Solving a chemical batch scheduling problem by local search. *Annals of Operations Research*, *96*, 17–36.

Burkard, R. E., & Hatzl, J. (2005). Review, extensions and computational comparison of MILP formulations for scheduling of batch processes. *Computers & Chemical Engineering*, *29*, 1752–1769.

Castro, P., Barbosa-Póvoa, A. P., & Matos, H. (2001). An improved RTN continuous-time formulation for the short-term scheduling of multipurpose batch plants. *Industrial & Engineering Chemistry Research*, *40*, 2059–2068.

Castro, P., Barbosa-Póvoa, A. P., & Matos, H. (2003). Optimal periodic scheduling of batch plants using RTN-based discrete and continuous-time formulations: a case study approach. *Industrial & Engineering Chemistry Research*, *42*, 3346–3360.

Floudas, C. A., & Lin, X. (2004). Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. *Computers & Chemical Engineering*, *28*, 2109–2129.

Gentner, K. (2005). *Dekompositionsverfahren für die ressourcenbeschränkte Projektplanung*. Aachen: Shaker.

Gentner, K., Neumann, K., Schwindt, C., & Trautmann, N. (2004). Batch production scheduling in the process industries. In J. Y. T. Leung (Ed.), *Handbook of scheduling: algorithms, models, and performance analysis* (Chap. 48). Boca Raton: CRC Press.

Grunow, M., Günther, H. O., & Lehmann, M. (2002). Campaign planning for multi-stage batch processes in the chemical industry. *OR Spectrum*, *24*, 281–314.

Honkomp, S. J., Lombardo, S., Rosen, O., & Pekny, J. F. (2000). The curse of reality: why process scheduling optimization problems are difficult in practice. *Computers & Chemical Engineering*, *24*, 323–328.

Ierapetritou, M. G., & Floudas, C. A. (1998). Effective continuous-time formulation for short-term scheduling, 1: multipurpose batch processes. *Industrial & Engineering Chemistry Research*, *37*, 4341–4359.

Kallrath, J. (2002). Planning and scheduling in the process industry. *OR Spectrum*, *24*, 219–250.

Kondili, E., Pantelides, C. C., & Sargent, R. W. H. (1993). A general algorithm for short-term scheduling of batch operations, I: MILP formulation. *Computers & Chemical Engineering*, *17*, 211–227.

Laborie, P. (2003). Algorithms for propagating resource constraints in AI planning and scheduling: existing approaches and new results. *Artificial Intelligence*, *143*, 151–188.

Maravelias, C. T., & Grossmann, I. E. (2004). A hybrid MILP/CP decomposition approach for the continuous time scheduling of multipurpose batch plants. *Computers & Chemical Engineering*, *28*, 1921–1949.

McCormick, S. T., & Rao, U. S. (1994). Some complexity results in cyclic scheduling. *Mathematical and Computer Modelling*, *20*, 107–122.

Méndez, C. A., Cerdá, J., Grossmann, I. E., Harjunkoski, I., & Fahl, M. (2006). State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Computers & Chemical Engineering*, *30*, 913–946.

Neumann, K., & Schwindt, C. (2002). Project scheduling with inventory constraints. *Mathematical Methods of Operations Research*, *56*, 513–533.

Neumann, K., Schwindt, C., & Trautmann, N. (2002). Advanced production scheduling for batch plants in process industries. *OR Spectrum*, *24*, 251–279.

Neumann, K., Schwindt, C., & Zimmermann, J. (2003). *Project scheduling with time windows and scarce resources*. Berlin: Springer.

Pinedo, M. (2002). *Scheduling: theory, algorithms, and systems*. Upper Saddle River: Prentice Hall.

Potts, C. N., & Kovalyov, M. Y. (2000). Scheduling with batching: a review. *European Journal of Operational Research*, *120*, 228–249.

Schwindt, C., & Trautmann, N. (2004). A priority-rule based method for batch production scheduling in the process industries. In D. Ahr, R. Fahrion, M. Oswald, & G. Reinelt (Eds.), *Operations research proceedings 2003* (pp. 111–118). Berlin: Springer.

Schwindt, C., & Trautmann, N. (2006). A cyclic approach to large-scale short-term planning of multi-purpose batch plants. In M. Morlock, C. Schwindt, N. Trautmann, & J. Zimmermann (Eds.), *Perspectives on operations research* (pp. 225–238). Wiesbaden: Deutscher Universitäts-Verlag.

Schwindt, C., Fink, R., & Trautmann, N. (2007). A priority-rule based method for scheduling in chemical batch production. In *Proceedings of the international conference on industrial engineering and engineering management* (pp. 1347–1351), Singapore.

Shah, N., Pantelides, C. C., & Sargent, R. W. H. (1993a). A general algorithm for short-term scheduling of batch operations, II: computational issues. *Computers & Chemical Engineering*, *17*, 229–244.

Shah, N., Pantelides, C. C., & Sargent, R. W. H. (1993b). Optimal periodic scheduling of multipurpose batch plants. *Annals of Operations Research*, *42*, 193–228.

Timpe, C. H., & Kallrath, J. (2000). Optimal planning in large multi-site production networks. *European Journal of Operational Research*, *126*, 422–435.

Wu, D., & Ierapetritou, M. (2004). Cyclic short-term scheduling of multiproduct batch plants using continuous-time representation. *Computers & Chemical Engineering*, *28*, 2271–2286.