OR Spectrum (2002) 24: 251–279

OR Spectrum

Advanced production scheduling for batch plants in process industries*

Klaus Neumann, Christoph Schwindt, and Norbert Trautmann

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

Abstract. An Advanced Planning System (APS) offers support at all planning levels along the supply chain while observing limited resources. We consider an APS for process industries (e.g. chemical and pharmaceutical industries) consisting of the modules network design (for long–term decisions), supply network planning (for medium–term decisions), and detailed production scheduling (for short–term decisions). For each module, we outline the decision problem, discuss the specifics of process industries, and review state–of–the–art solution approaches. For the module detailed production scheduling, a new solution approach is proposed in the case of batch production, which can solve much larger practical problems than the methods known thus far. The new approach decomposes detailed production scheduling for batch production into batching and batch scheduling. The batching problem converts the primary requirements for products into individual batches, where the workload is to be minimized. We formulate the batching problem as a nonlinear mixed–integer program and transform it into a linear mixed–binary program of moderate size, which can be solved by standard software. The batch scheduling problem allocates the batches to scarce resources such as processing units, workers, and intermediate storage facilities, where some regular objective function like the makespan is to be minimized. The batch scheduling problem is modelled as a resource–constrained project scheduling problem, which can be solved by an efficient truncated branch–and–bound algorithm developed recently. The performance of the new solution procedures for batching and batch scheduling is demonstrated by solving several instances of a case study from process industries.

Key words: Detailed production scheduling – Process industries – Batch production – Advanced Planning Systems (APS)

* This research has been supported in part by the Deutsche Forschungsgemeinschaft under Grant Ne 137/4 and in part by the SAP AG, Walldorf.
Correspondence to: K. Neumannn

1 Introduction

In process industries, e.g. chemical or pharmaceutical industries, final products are typically produced through several successive chemical or physical transformation processes (also called *tasks*) such as heating, filtration, or packaging. A task requires different types of production resources: a *processing unit* (e.g. a reactor or filter) and *storage facilities* for input and output products (e.g. tanks or silos). All resources are available in limited capacity only.

Different software vendors such as SAP, AspenTech, i2 Technologies, or J.D. Edwards offer Advanced Planning Systems (APS) that support all planning sections along the supply chain while observing limited resources. In this paper, we concentrate on the production section of the supply chain. Long–term decisions are made in the *network design* phase, where plant locations are determined and production systems are configured. Mid–term decisions are made in the *supply network planning* phase, which provides the primary requirements for the final products to be produced at individual plants on the basis of *demand planning* data. The short–term allocation of individual production resources over time to the production of the primary requirements is performed during the *detailed production scheduling* phase.

The purpose of this paper is to present a new solution approach to the detailed production scheduling phase. At first we put the detailed production scheduling phase into the framework of an APS. Based on a generic architecture of an APS, we state the decision problems arising in the network design, supply network planning, and detailed production scheduling phases, discuss the specifics of process industries, and briefly review state–of–the–art solution approaches.

The new approach to detailed production scheduling is discussed in the main part of the paper. The approach is tailored to the minimization of some regular objective function (i.e. a function nondecreasing in the start times of operations) like makespan or mean tardiness in the case of batch production, where all material flows are discontinuous. The basic idea is to decompose the detailed production scheduling problem into a batching and a batch scheduling problem. *Batching* converts the primary requirements for products into individual batches. A *batch* combines a task and a production quantity. The time needed for processing the batch is independent of the production quantity. The objective of batching is to minimize the workload, i.e. the total amount of work to be performed on the processing units. Constraints result from given bounds on batch sizes and proportions of input and output products, storage capacities, perishable products, and the given primary requirements. We formulate the batching problem as a nonlinear mixed–integer program, which is subsequently transformed into a linear mixed–binary program. The processing of a batch is called an *operation*. *Batch scheduling* allocates scarce resources like processing units, workers, and intermediate storage facilities to the operations arising from the batching step such that the regular objective function of detailed production scheduling is minimized. We model the batch scheduling problem as a resource–constrained project scheduling problem, where an operation corresponds to an activity of the project. Perishable products give rise to minimum and maximum time lags between activities. Each processing unit rep-

resents a renewable resource of unit capacity with sequence–dependent cleaning times. Alternative processing units are modelled by alternative execution modes for the activities. Storage facilities represent so–called cumulative resources. After a detailed description of the project scheduling model, we show how to solve the project scheduling problem using a truncated branch–and–bound algorithm. The basic idea of this algorithm is to relax the (hard) resource constraints and to replace them by a disjunction of additional precedence constraints between activities.

The detailed production scheduling problem and the specifics of process industries are explained by an example from chemical industry. This example is part of a case study presented by Westenberger and Kallrath (1995) and will be called *WK example* in what follows. In the WK example, we consider a batch production plant producing five different final products. The problem is to find a feasible production schedule with minimum makespan for given primary requirements. We compare the decomposition approach proposed in this paper with the best solution approach known from literature so far by solving several instances that result from varying the primary requirements.

The remainder of this paper is organized as follows: Section 2 is concerned with the architecture of APS for process industries, the individual modules, and the information flow between different modules. Section 3 illustrates the detailed production scheduling problem in chemical industry using the WK example. The decomposition of detailed production scheduling into batching and batch scheduling is explained in Section 4. The latter subproblems and respective solution methods are discussed in Sections 5 and 6. Section 7 provides the results of the experimental performance analysis. In Section 8 we show how to integrate further constraints such as renewable resources of capacity greater than one (several identical processing units or manpower), calendars, or campaigns into the batch scheduling procedure. Section 9 is devoted to concluding remarks and directions for further research.

2 Components of Advanced Planning Systems

Advanced Planning Systems offer support for long–term, mid–term, and short–term planning of the supply chain, which consists of the sections procurement, production, distribution, and sales. In contrast to enterprise resource planning (ERP) systems like SAP R/3, limited availability of resources is taken into account in all planning phases. A monolithic supply chain planning is impracticable due to the complexity and size of the individual decision problems. Furthermore, the different planning horizons motivate a division into long–term, mid–term, and short–term planning. This is the reason why Advanced Planning is generally decomposed into planning modules where the interdependencies between the modules have to be respected.

Figure 1 shows the structure of a generic APS (cf. Meyr et al., 2000) consisting of modules for the phases network design, demand planning, supply network planning, detailed production scheduling, distribution & transport planning, demand fulfillment & available to promise (ATP), and inventory management. This structure reflects the common hierarchy of planning tasks that is implemented in the

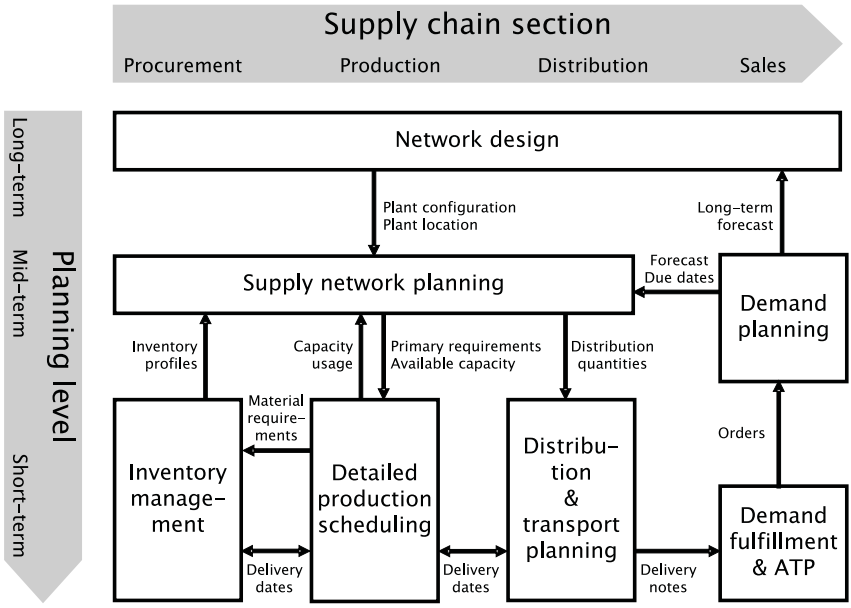

Fig. 1. Structure of an APS

APS offered by different software vendors such as SAP, AspenTech, i2 Technologies, or J.D. Edwards. Each box in Figure 1 corresponds to a module of the APS. The arcs between boxes represent information flows between the modules. In what follows, we concentrate on the production section in the supply chain consisting of the modules network design, supply network planning, and detailed production scheduling. We briefly discuss each module and state the corresponding decision problem, where we place special emphasis on specific requirements arising in process industries. The shorter the planning horizon, the greater is the impact of those specifics on the planning tasks.

Basic concepts of APS and the architecture of commercial APS are presented in Knolmayer et al. (2002) and Stadtler and Kilger (2000). For special requirements of process industries we refer to Applequist et al. (1997), Honkomp et al. (2000), and Pinto and Grossmann (1998).

2.1 Network design

Network design is concerned with the physical structure of the supply chain and thus deals with planning locations of plants and warehouses or other storage facilities and with designing the layout of the individual plants. Typically, the planning horizon ranges from three to ten years.

The *facility location problem* can be described as follows (cf. Tsiakis et al., 2001). We consider a network comprising possible locations for plants and warehouses, fixed locations of customer zones, and transportation facilities between

individual locations (e.g. road networks, shipping lanes, sea routes, and air corridors). Long–term forecasts for potential sales of product groups in the customer zones are given. The objective is to determine the number, locations, and capacities of plants and warehouses such that the total annualized cost of the network is minimized. The cost comprises infrastructure cost related to establishing a plant or a warehouse at a certain location, production cost at plants, material handling cost at warehouses, and transportation cost. Binary decision variables indicate whether a warehouse or a plant is built at a certain location, whether a plant supplies a warehouse, or whether a warehouse is assigned to a customer zone. For each plant and each warehouse, a continuous decision variable is introduced representing the respective production or storage capacity. Different types of constraints have to be observed: Network structure constraints say that a link between a plant and a warehouse or a warehouse and a customer zone can exist only if the corresponding warehouses and plants are built. In addition, a flow of material can take place only if the corresponding link is established. Furthermore, so–called single sourcing constraints (cf. Holmberg et al., 1999) may be imposed, according to which a customer zone must be assigned to a single warehouse. Material balance constraints ensure that the production rates at the individual plants must coincide with the flow of the products from the plants to the warehouses, and that the amount of a product received by a customer zone matches the corresponding demand. Further constraints arise from lower and upper bounds on production, storage, and transportation capacities. Solution approaches to the facility location problem known from literature are based on formulations as linear mixed–integer programs combined with heuristic methods, standard solvers and problem decomposition, e.g. Benders' decomposition. For an overview of recent literature, we refer to Tsiakis et al. (2001) and Vidal and Goetschalckx (1997). Issues of network design and plant configurations are also treated in Kallrath (2002b).

The *layout design problem* is solved for each plant individually and involves the spatial arrangement of processing units and intermediate storage facilities. If the plant under consideration consists of a single floor, a typical problem is to divide the given set of equipment into groups of units that reflect the partitions created by aisles or corridors such that the intersection flow cost are minimized. Jayakumar and Reklaitis (1994) show how to model this problem as a graph partitioning problem and present a corresponding heuristic solution procedure. Constraints arise from process sequences, preferred directions of material flow, as well as safety and other restrictions. If the plant consists of multiple floors, the analogous problem is to divide the given set of equipment into subsets and additionally assign each of the subsets to a single level. In this case, the cost associated with the flow between two processing units or storage facilities depends on the distance in between and may be different for horizontal, downward, and upward flow. Jayakumar and Reklaitis (1996) formulate this problem as a nonlinear integer program and convert it to a linear mixed–integer program. For a detailed literature review on chemical plant layout problems, we review to Jayakumar and Reklaitis (1994, 1996).

In process industries, network design includes, for each processing unit, the decision whether it is operated in continuous, semi–continuous, or batch production mode. Plants operating in continuous production mode are dedicated to one product

group, e.g. refineries in oil industry. In semi–continuous production mode, the input products are loaded continuously into the processing unit and the output products arise discontinuously or vice versa. Batch production mode is typical of chemical or pharmaceutical industries if small amounts of a large number of products are required, which corresponds to the increasing trend to low–volume production of speciality goods in customer–specific packaging. In what follows, we will concentrate on process industries operating in batch production mode. The different products are processed on multi–purpose equipment, where the production plant is configured according to the required final products. Before processing the next set of final products, the plant has to be reconfigured. This requires the termination of all operations.

Network design is based on long–term changes in the product program, sales figures, and production technologies. Thus, main input factors are long–term forecasts for potential sales of product groups in specific regions, which result from the **demand planning** module. Advanced Planning Systems perform demand planning in three steps (cf. Wagner, 2000): First, statistical forecasting captures the main characteristics of time series of the past and creates forecasts using time–series analysis and causal models (cf. Silver et al., 1998). Second, information is added to the time series that had not been taken into account within statistical forecasting, e.g. life cycles of product groups (cf. Wright and Goodwin, 1998). Third, collaboration between different functional areas (sales, production, procurement) is supported in order to get a consensus on the forecast.

2.2 Supply network planning

The goal of supply network planning is an efficient utilization of the production and storage capacities determined by the network design. The planning horizon has to cover at least one seasonal cycle to be able to balance all demand peaks. It is divided into weekly or monthly *time buckets*. Due to changes in input data, supply network planning is incrementally updated, e.g. in weeks or months, in order to take updated information like actual inventory profiles, capacity usages, and new demand data into consideration. Because of the problem size, it is often necessary to concentrate on bottleneck resources.

The decision variables of supply network planning are purchasing, production, and transportation quantities of final products, and the amount of additionally required capacities (e.g. overtime) in individual time buckets. The objective is to minimize the cost of production, transportation, and inventory holding, cost for delayed or cancelled fulfillment of customer orders, and cost for additional capacities. Constraints arise from given delivery times for customer orders, perishability of products, production, storage and transportation capacities, calendars, and lower and upper bounds on batch sizes. Supply network planning settles capacity bottlenecks by producing in earlier or later periods, at alternative sites or in alternative production modes, by working overtime, by buying products from external suppliers, or by late or cancelled delivery.

Solution approaches to supply network planning problems known from literature are based on formulations as mixed–integer programs, cf. e.g. Kallrath (1999,

2000), Oxe (1997), and Timpe and Kallrath (2000). Kallrath (1999) describes a model that includes the determination of so–called *campaigns*. A campaign consists of several batches of the same task that must be executed consecutively. Typically only campaigns of a minimum size can be produced. Between different campaigns, a sequence–dependent setup–time has to be considered. The problem of scheduling campaigns in distributed chemical batch plants is studied in Berning et al. (2002). For a survey on campaign planning in chemical industry we refer to Papageorgiou and Pantelides (1996). Grunow et al. (2002) present a three-stage decomposition approach to campaign planning and scheduling.

If only one plant is considered, supply network planning is similar to classical master production scheduling (MPS) with capacity constraints (cf. Nahmias, 1997). Note that certain features like cancelling delivery are not considered in MPS.

Supply network planning is based on weekly or monthly forecasts for at most one year including fixed customer orders, the effects of mid–term marketing events, and promotions on sales. This information has to be considered in the statistical forecasting step of demand planning. The customer orders are registered in the **demand fulfillment & ATP** module, which also determines a first due date for each order (cf. Kilger and Schneeweiss, 2000). Further input factors of supply network planning are the plant configuration, i.e. the plant layout, and the plant locations. Both are determined in the network design module.

The results of supply network planning are inputs to other modules. **Detailed production scheduling** is based on primary requirements for final products and on production capacities (remember that the production capacities for the time buckets result from supply network planning). **Distribution & transportation planning** refers to the distribution quantities computed.

2.3 Detailed production scheduling

Detailed production scheduling deals with the short–term allocation of resources over time to the production of the primary requirements determined by supply network planning. Thus, the problem is to explode the primary requirements into batches and to schedule those batches on scarce resources. The secondary requirements for raw materials arising from the batches determined are then reported to the **inventory management**, where appropriate lot sizes for purchasing are to be found.

Detailed production scheduling is performed individually for each plant. The planning horizon corresponds to the bucket size of supply network planning. The goal is to compute a feasible production plan that minimizes a regular objective function (i.e. a function nondecreasing in the start times of operations). The objective of makespan minimization is particularly important in the context of batch production mode in order to enable flexible responses to demand changes, especially if the whole plant has to be reconfigured before producing another set of final products. Another regular objective function that is often considered in practice is the mean tardiness with respect to given due dates for the delivery of customer–ordered final products.

Batches are processed on *multi–purpose processing units*, which can operate several kinds of tasks. At some production levels, there may be alternative processing units. In general, the processing time depends on the particular processing unit used. Recall, however, that in batch production mode the execution time of a task is independent of the respective batch size. For each task, a lower and an upper bound on the respective batch size are given. The minimum batch size generally arises from a minimum filling level, whereas the maximum batch size coincides with the capacity of the corresponding processing unit. A special feature of process industries is the need for cleaning processing units in order to guarantee purity of products. The cleaning times often depend on the sequence of batches and the quality of the products to be produced. In certain applications, a processing unit must also be cleaned when no other batch is started immediately after the completion of the preceding batch. At any point in time, at most one batch or cleaning operation can be executed on one and the same processing unit.

Each task consumes and produces different products. In batch production mode, the input of a task is consumed at its start and the output arises at its completion. Certain intermediate products can be buffered in *storage facilities* of given capacity. Other intermediate products are perishable and cannot be stored and thus must be consumed without any delay. In that case, producing tasks must be assigned to consuming tasks such that no perishable product is in stock at any time.

The storage facilities and processing units of a batch plant are linked by divergent, convergent, or cyclic material flows. Accordingly, recipes are analytic (e.g. in basic materials industry), synthetic (e.g. in pharmaceutical industry), or cyclic (e.g. if catalysts are present). For details we refer to Loos (1997). For each task, the number of executions and the production quantities have to be chosen such that the quantity produced of each product is sufficient to match its gross requirement. Note that in the case of cyclic material flows, some residual of the products belonging to a cycle cannot be used for production. Proportions of input and output goods of tasks are often fixed. However, the input and output proportions of a task may also vary within given bounds, that is, the output can be allotted to different products according to the specific demand situation.

Most solution procedures known from literature are based on formulations of the detailed production scheduling problem as a time–indexed mixed–integer program depending on the time grid chosen, see e.g. Blömer and Günther (1998, 2000), Burkard et al. (1998), Kondili et al. (1993), Pinto and Grossmann (1995, 1998), and Shah (1998). For a literature review we refer to Blömer and Günther (1998) and Shah (1998). Grunow et al. (2002) and Timpe (2002) successfully apply hybrid approaches involving mixed–integer and constraint programming.

3 A case study from chemical industry

Based on an existing plant, Westenberger and Kallrath (1995) have presented a case study dealing with detailed production scheduling in chemical industry. Here, we consider the third out of six “tasks” (see Kallrath, 2002a) where additionally the cleaning times are included. The data of the case study and a bibliogra-

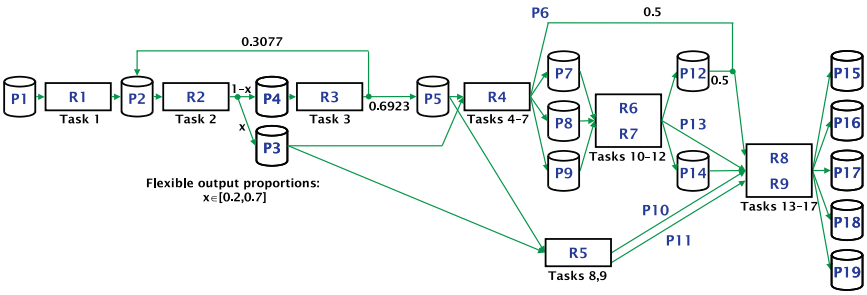

Fig. 2. Production flow of WK example

phy of papers referring to the WK example are available at http://www.wior.uni-karlsruhe.de/neumann/forschung/wk95/wk95.html.

Figure 2 illustrates the production flow of the WK example, which consists of 15 storage facilities for the storable products $P1, \ldots, P5, P7, P8, P9, P12, P14, \ldots, P19$ and 9 multi–purpose processing units (reactors) $R1$ to $R9$. A representation of the production flow as a *state–task–network* can be found in Blömer and Günther (2000). Intermediate products $P6$, $P10$, $P11$, and $P13$ cannot be stocked. The storage facilities and processing units are linked by divergent, convergent, or cyclic material flows. There are primary requirements of 30, 30, 40, 20, and 40 units for the final products $P15$ to $P19$ that arise from supply network planning. The primary requirements have to be produced within 6 days with 24 hours of working time each. The problem is to explode the primary requirements into batches and to schedule these batches on the scarce production resources such that the makespan is minimized.

At some production levels, there are alternative processing units, e.g. $R6$ and $R7$. Table 1 summarizes lower and upper bounds on batch sizes, processing and cleaning times for processing units, and the materials consumed and produced by each task. The time unit is one hour. Recall that all processing times are independent of the respective batch sizes. Each task uses one processing unit. In Figure 2, the processing units are labelled with the tasks executed on them. The tasks are numbered according to increasing quality of output products, i.e., cleaning of a processing unit after the completion of a task becomes necessary if one passes to a task with a higher number or if there is no task on the processing unit which is started immediately after the completion of the preceding one.

Each task consumes and produces materials as given in Table 1. Tasks 2 and 3 produce two output products each and task 15 consumes two input products. The respective proportions of the output products are shown in Figure 2. The output proportions of task 2 are variable. $100x\%$ of the total output is allotted to product $P3$ and the remaining part to product $P4$, where x can vary between 0.2 and 0.7.

For storable products, the inventory is bounded from above (cf. Table 2). Sufficient capacity is available to store the required raw material $P1$ and final products $P15$ to $P19$. There is sufficient initial stock of $P1$, and there is no initial stock of $P15$ to $P19$. At each point in time, the inventories of products $P1$ to $P19$ must be nonnegative.

Table 1. Task settings of WK example

Task	Batch size $[\min, \max]$	Alternative processing units	Alternative processing times	Cleaning times	Materials consumed	Materials produced
1	[3,10]	$R1$	2	1	$P1$	$P2$
2	[5,20]	$R2$	4	2	$P2$	$P3$, $P4$
3	[4,10]	$R3$	2	1	$P4$	$P2$, $P5$
4	[4,10]	$R4$	4	2	$P3$	$P6$
5	[4,10]	$R4$	4	2	$P3$	$P7$
6	[4,10]	$R4$	4	2	$P5$	$P8$
7	[4,10]	$R4$	4	2	$P5$	$P9$
8	[4,10]	$R5$	6	3	$P3$	$P10$
9	[4,10]	$R5$	6	3	$P5$	$P11$
10	[3,7]	$R6$ / $R7$	4 / 5	2 / 2.5	$P7$	$P12$
11	[3,7]	$R6$ / $R7$	5 / 6	2.5 / 3	$P8$	$P13$
12	[3,7]	$R6$ / $R7$	6 / 6	3 / 3	$P9$	$P14$
13	[4,12]	$R8$ / $R9$	4 / 6	2 / 3	$P10$	$P15$
14	[4,12]	$R8$ / $R9$	4 / 6	2 / 3	$P11$	$P16$
15	[4,12]	$R8$	4	2	$P6$, $P12$	$P17$
16	[4,12]	$R8$ / $R9$	6 / 6	3 / 3	$P13$	$P18$
17	[4,12]	$R8$ / $R9$	6 / 6	3 / 3	$P14$	$P19$

Table 2. Bounds on inventories of WK example

	$P1$	$P2$	$P3$	$P4$	$P5$	$P7$	$P8$	$P9$	$P12$	$P14$	$P15$	$P16$	$P17$	$P18$	$P19$
Initial stock	∞	20	20	0	20	0	0	0	0	0	0	0	0	0	0
Max. stock	∞	30	30	15	30	10	10	10	10	10	∞	∞	∞	∞	∞

4 Decomposition of detailed production scheduling

In contrast to long–term and mid–term planning, short–term planning has to take into account a large number of constraints that are specific of process industries. That is why detailed production scheduling requires tailor–made models and solution procedures. The particular feature of our approach for detailed production scheduling is the decomposition into a *batching problem* (BP) and a *batch scheduling problem* (BSP). Batching provides the number and sizes of batches for all tasks, and the subsequent batch scheduling assigns a processing unit and an execution time interval to each batch. In Section 5, we show how to model (BP) as a nonlinear mixed–integer program and how to transform it into a linear mixed–integer program of moderate size that can be solved using standard software, e.g. CPLEX or XPRESS–MP. The essential decision variables of (BSP) represent the start times of the operations. Efficient solution procedures for resource–constrained project scheduling can be used for solving large instances of (BSP), see Section 6. A similar decomposition approach has been used by Brucker and Hurink (2000) for solving a special case of detailed production scheduling.

The number of decision variables of the time–indexed MIP models for detailed production scheduling from literature depends on the time grid chosen and is not polynomial in the number of operations. This is the reason why, in contrast to our approach, only relatively small instances can be solved by the latter models.

Obviously, the material flow establishes a causal relationship between the batching and batch scheduling problems (BP) and (BSP). This means that the optimal start times of operations depend on the optimal number and sizes of batches and vice versa. Consequently, when solving at first (BP) and then the corresponding (BSP) we are not assured to obtain an optimal solution to the original detailed production scheduling problem. Due to the limited capacity of intermediate storages, the decomposition method may even fail in finding any feasible solution. As will be shown in the performance analysis of Section 7, the decomposition nevertheless proves useful for heuristically solving problem instances of practical size.

5 Batching

Batching deals with converting the primary requirements for products into sets of batches for each task. The goal is to minimize the workload to be scheduled. Constraints result from given bounds on batch sizes and the number of task executions and from limited storage capacities. In Section 5.1, we present a nonlinear mixed–integer formulation of the batching problem, a preliminary version of which can be found in Neumann et al. (2001a). In Section 5.2, we transform that nonlinear program into a linear mixed–binary program of polynomially related size.

5.1 Formulation as a nonlinear mixed–integer program

Let $\mathcal{T}$ be the set of all tasks and let β_τ be the batch size and ε_τ be the number of batches for task $\tau \in \mathcal{T}$ executed within the given planning horizon $[0, \overline{d}]$ for detailed production scheduling. Thus, $\beta_\tau \varepsilon_\tau$ is the amount produced by task τ. Prescribed minimum and maximum batch sizes $\underline{\beta}_\tau$ and $\overline{\beta}_\tau$, respectively, for task τ provide the constraints

$$\underline{\beta}_\tau \leq \beta_\tau \leq \overline{\beta}_\tau \qquad (\tau \in \mathcal{T}) \tag{1}$$

Let $\mathcal{U}_\tau$ be the set of alternative processing units on which task τ can be carried out and $p_{\tau k}$ be the processing time of task τ on processing unit $k \in \mathcal{U}_\tau$. Then $\overline{\varepsilon}_\tau := \sum_{k \in \mathcal{U}_\tau} \overline{d}/p_{\tau k}$ is an upper bound on the number of executions of tasks τ, and we have the constraints

$$\left.\begin{array}{l} 0 \leq \varepsilon_\tau \leq \overline{\varepsilon}_\tau \\ \varepsilon_\tau \in \mathbb{Z} \end{array}\right\} \quad (\tau \in \mathcal{T}) \tag{2}$$

A task may have several input and output products, e.g., task 2 in the WK example (cf. Section 3). For task $\tau \in \mathcal{T}$, let $\alpha_{\tau\pi} > 0$ be the proportion of output

product π and $-\alpha_{\tau\pi} > 0$ be the proportion of input product π. If π is neither input nor output product of task τ, we set $\alpha_{\tau\pi} := 0$. Obviously,

$$\sum_{\pi\in\mathcal{P}_\tau^+} \alpha_{\tau\pi} = -\sum_{\pi\in\mathcal{P}_\tau^-} \alpha_{\tau\pi} = 1 \qquad (\tau \in \mathcal{T}) \tag{3}$$

where $\mathcal{P}_\tau^+$ and $\mathcal{P}_\tau^-$ are the sets of output and input products, respectively, of task τ. Let $\underline{\alpha}_{\tau\pi}$ and $\overline{\alpha}_{\tau\pi}$ be given lower and upper bounds on $\alpha_{\tau\pi}$. Then

$$\underline{\alpha}_{\tau\pi} \le \alpha_{\tau\pi} \le \overline{\alpha}_{\tau\pi} \qquad (\tau \in \mathcal{T}, \pi \in \mathcal{P}_\tau) \tag{4}$$

where $\mathcal{P}_\tau := \mathcal{P}_\tau^+ \cup \mathcal{P}_\tau^-$ is the set of products which are either input or output of task τ. If $\alpha_{\tau\pi}$ is fixed, we have $\underline{\alpha}_{\tau\pi} = \overline{\alpha}_{\tau\pi}$.

Next, we consider inventory constraints. For a product $\pi \in \mathcal{P}_\tau$, $\alpha_{\tau\pi}\beta_\tau\varepsilon_\tau$ or $-\alpha_{\tau\pi}\beta_\tau\varepsilon_\tau$ is the total amount of π produced or consumed, respectively, by task τ. Let ρ_π be the given primary requirement minus the initial stock for product π. Consider a recycled product π which belongs to a cycle in the product structure and is the output product of a task τ inside the cycle and of a different task τ' outside the cycle, e.g., product $P2$ in the WK example (see Fig. 2). In that case the amount of product π recycled after the last execution of task τ necessarily remains on stock because it is not consumed. To guarantee that a sufficient quantity of product π is available as input of consuming tasks $\tau'' \in \mathcal{T}_\pi^-$, ρ_π has to be enlarged by the maximum residual stock of π. To meet all primary requirements, it then has to hold that

$$\sum_{\tau\in\mathcal{T}_\pi} \alpha_{\tau\pi}\beta_\tau\varepsilon_\tau \ge \rho_\pi \qquad (\pi \in \mathcal{P}) \tag{5}$$

where $\mathcal{T}_\pi$ is the set of tasks producing or consuming product π and $\mathcal{P} = \cup_{\tau\in\mathcal{T}}\mathcal{P}_\tau$ is the set of all products. $\sum_{\tau\in\mathcal{T}_\pi} \alpha_{\tau\pi}\beta_\tau\varepsilon_\tau - \rho_\pi$ is the final inventory of product π after satisfying the primary and secondary requirements. This final inventory must not exceed the given capacity σ_π of the storage facility for π. That is,

$$\sum_{\tau\in\mathcal{T}_\pi} \alpha_{\tau\pi}\beta_\tau\varepsilon_\tau \le \rho_\pi + \sigma_\pi \qquad (\pi \in \mathcal{P}) \tag{6}$$

If product π cannot be stored, then $\sigma_\pi := 0$.

The last constraints refer to perishable products, which cannot be stored. Let $\mathcal{T}_\pi^+$ and $\mathcal{T}_\pi^-$ be the sets of tasks producing and consuming, respectively, product π. We assume that for a perishable product π, the amount produced by a batch of a task $\tau \in \mathcal{T}_\pi^+$ must equal the amount consumed by a batch of any task $\tau' \in \mathcal{T}_\pi^-$, that is,

$$\alpha_{\tau\pi}\beta_\tau = -\alpha_{\tau'\pi}\beta_{\tau'} \qquad (\pi \in \mathcal{P}^p, (\tau,\tau') \in \mathcal{T}_\pi^+ \times \mathcal{T}_\pi^-) \tag{7}$$

where $\mathcal{P}^p$ is the set of perishable products. This means that in contrast to the formulation by Neumann et al. (2001), we do not consider solutions to the batching problem where batches of several tasks $\tau \in \mathcal{T}_\pi^+$ and batches of several tasks $\tau' \in \mathcal{T}_\pi^-$ are matched in such a way that the total amount of product π produced equals the

total amount consumed. The reason why we restrict ourselves to the one–to–one correspondence (7) between tasks $\tau \in \mathcal{T}_\pi^+$ and $\tau' \in \mathcal{T}_\pi^+$ is that otherwise, the completion of several batches producing π and the start of several batches consuming π would have to occur simultaneously. The latter requirement would considerably reduce the set of feasible solutions to the corresponding batch scheduling problem. Moreover, in practice the implementation of schedules where several operations must be *completed* simultaneously is generally impossible because already small differences between predicted and realized processing times lead to infeasibility.

Let $\overline{p}_\tau := \sum_{k \in \mathcal{U}_\tau} p_{\tau k}/|\mathcal{U}_\tau|$ be the mean processing time of task τ on any of the alternative processing units. To minimize the workload, the objective function to be minimized is chosen to be the total mean processing time $\sum_{\tau \in \mathcal{T}} \overline{p}_\tau \varepsilon_\tau$ (recall that the processing time of a batch is independent of the batch size). The batching problem then takes the form

$$\left.\begin{array}{ll} \text{Min.} & \displaystyle\sum_{\tau \in \mathcal{T}} \overline{p}_\tau \varepsilon_\tau \\ \text{s.t.} & (1) \text{ to } (7) \end{array}\right\} \qquad \text{(BP)}$$

(BP) represents a nonlinear mixed–integer program with the integral decision variables ε_τ and the continuous decision variables β_τ and $\alpha_{\tau\pi}$ ($\tau \in \mathcal{T}$, $\pi \in \mathcal{P}_\tau$). Next we show the NP–hardness of batching problem (BP).

Proposition. The feasibility problem for (BP) is NP–hard even if $\mathcal{P}^p = \emptyset$, $\sigma_\pi = \infty$ for all $\pi \in \mathcal{P}$, and $\overline{\varepsilon}_\tau = \infty$, $\underline{\beta}_\tau = \overline{\beta}_\tau$ for all $\tau \in \mathcal{T}$.

Proof. We provide a polynomial transformation from 3–PARTITION (cf. Garey and Johnson, 1979). Let $B \in \mathbb{N}$ be a bound and let A be a set of 3ν elements λ with associated sizes $s(\lambda) \in \mathbb{N}$ such that $B/4 < s(\lambda) < B/2$ and $\sum_{\lambda \in A} s(\lambda) = \nu B$. The question is whether or not A can be partitioned into ν sets $A_1, \ldots, A_\nu$ such that $\sum_{\lambda \in A_\mu} s(\lambda) = B$ for all $\mu = 1, \ldots, \nu$.

For each $\lambda \in A$, we introduce a raw material π_λ with initial stock $s(\lambda)$, and each index $\mu = 1, \ldots, \nu$ is assigned to a final product π_μ with primary requirement B. For each raw material π_λ and each final product π_μ, we define one task $\tau_{\lambda\mu}$

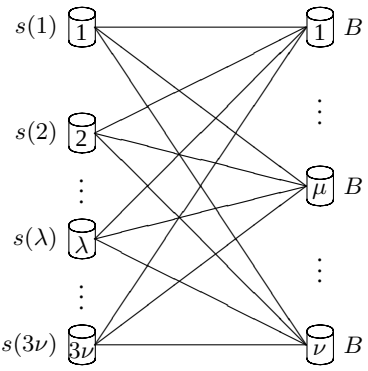

Fig. 3. Batching problem of the proof to the Proposition

transforming π_λ into π_μ with $\underline{\beta}_{\lambda\mu} = \overline{\beta}_{\lambda\mu} = s(\lambda)$ (see Fig. 3). Because of the scarcity of raw materials, at most one batch of tasks $\tau_{\lambda\mu}$ can be executed for each $\lambda \in A$. Since the cumulative requirements νB for all final products equal the total inventory $\sum_{\lambda\in A} s(\lambda)$ of all raw materials, exactly one batch consuming product π_λ is executed for each $\lambda \in A$, say, a batch of task $\tau_{\lambda\mu}$. There is a feasible batching precisely if $\sum_{\lambda\in A} s(\lambda)\varepsilon_{\tau_{\lambda\mu}} = B$ holds for all $\mu = 1, \ldots, \nu$. Assuming that $\lambda \in A_\mu$ if and only if $\varepsilon_{\tau_{\lambda\mu}} = 1$, this condition is met exactly if the 3–PARTITION instance is a yes–instance. □

It can easily be shown that the feasible region belonging to the continuous relaxation of problem (BP) is generally nonconvex. Thus, the computation of an optimal solution to (BP) is not only intractable from the complexity point of view but even worse, it poses a serious algorithmic problem. Thus in the following subsection, we develop a formulation of the batching problem as an equivalent linear mixed–binary program. The number of binary variables in the latter problem equals the upper bound $\sum_{\tau\in\mathcal{T}} \overline{\varepsilon}_\tau$ on the number of batches which can be carried out within the planning horizon.

5.2 Formulation as a linear mixed–binary program

To obtain an equivalent linear formulation of (BP), we first introduce the continuous variables

$$\xi_{\tau\pi} = \alpha_{\tau\pi}\beta_\tau \qquad (\tau \in \mathcal{T}, \pi \in \mathcal{P}_\tau)$$

representing the negative amount of product π consumed or the amount of product π produced, respectively, by a batch of task τ. Since the batch size of a task equals the sum of all input quantities, we have

$$\beta_\tau = \sum_{\pi\in\mathcal{P}_\tau^+} \xi_{\tau\pi} \qquad (\tau \in \mathcal{T})$$

Thus, the lower and upper bound constraints (4) on proportions $\alpha_{\tau\pi}$ can be written as

$$\underline{\alpha}_{\tau\pi} \sum_{\pi'\in\mathcal{P}_\tau^+} \xi_{\tau\pi'} \le \xi_{\tau\pi} \le \overline{\alpha}_{\tau\pi} \sum_{\pi'\in\mathcal{P}_\tau^+} \xi_{\tau\pi'} \qquad (\tau \in \mathcal{T}, \pi \in \mathcal{P}_\tau) \tag{8}$$

The equations

$$\sum_{\pi'\in\mathcal{P}_\tau^+} \xi_{\tau\pi'} = -\sum_{\pi'\in\mathcal{P}_\tau^-} \xi_{\tau\pi'} \qquad (\tau \in \mathcal{T}) \tag{9}$$

correspond to the mass balance constraints (3), and the batch size constraints (1) now read

$$\underline{\beta}_\tau \le \sum_{\pi\in\mathcal{P}_\tau^+} \xi_{\tau\pi} \le \overline{\beta}_\tau \qquad (\tau \in \mathcal{T}) \tag{10}$$

Similarly, the batch size coupling conditions (7) for perishable products can be formulated as

$$\xi_{\tau\pi} = -\xi_{\tau'\pi} \qquad (\pi \in \mathcal{P}^p, (\tau, \tau') \in \mathcal{T}_\pi^+ \times \mathcal{T}_\pi^-) \tag{11}$$

In order to eliminate the nonlinear term $\alpha_{\tau\pi}\beta_\tau\varepsilon_\tau = \xi_{\tau\pi}\varepsilon_\tau$ occurring in the inventory constraints (5) and (6), we replace $\xi_{\tau\pi}\varepsilon_\tau$ with the sum of $\overline{\varepsilon}_\tau$ continuous variables $\xi_{\tau\pi}^\mu$ ($\mu = 1, \ldots, \overline{\varepsilon}_\tau$), where

$$\xi_{\tau\pi}^\mu = \begin{cases} \xi_{\tau\pi}, \text{ if } \mu \le \varepsilon_\tau \\ 0, \quad \text{otherwise} \end{cases}$$

The number ε_τ of batches for task τ is represented by the sum of $\overline{\varepsilon}_\tau$ binary variables θ_τ^μ ($\mu = 1, \ldots, \overline{\varepsilon}_\tau$), which equal one exactly if $\mu \le \varepsilon_\tau$, that is,

$$\theta_\tau^\mu = \begin{cases} 1, \text{ if } \sum_{\pi \in \mathcal{P}_\tau^+} \xi_{\tau\pi}^\mu > 0 \\ 0, \text{ otherwise} \end{cases}$$

The linking between variables $\xi_{\tau\pi}^\mu$ and θ_τ^μ can be achieved as follows if π is output product of task τ (i.e., $\xi_{\tau\pi} > 0$):

$$\left.\begin{aligned} 0 \;&\le\; \xi_{\tau\pi}^\mu \;\le\; \xi_{\tau\pi} \\ \xi_{\tau\pi} - (1 - \theta_\tau^\mu)\overline{\alpha}_{\tau\pi}\overline{\beta}_\tau \;&\le\; \xi_{\tau\pi}^\mu \;\le\; \overline{\alpha}_{\tau\pi}\overline{\beta}_\tau\theta_\tau^\mu \end{aligned}\right\} \qquad \begin{array}{c}(\tau \in \mathcal{T}, \pi \in \mathcal{P}_\tau^+, \\ \mu = 1, \ldots, \overline{\varepsilon}_\tau)\end{array} \tag{12}$$

If $\xi_{\tau\pi}^\mu > 0$, then $\xi_{\tau\pi}^\mu \le \overline{\alpha}_{\tau\pi}\overline{\beta}_\tau\theta_\tau^\mu$ implies $\theta_\tau^\mu > 0$. For $\xi_{\tau\pi}^\mu = 0$, inequality $\xi_{\tau\pi} - (1 - \theta_\tau^\mu)\overline{\alpha}_{\tau\pi}\overline{\beta}_\tau \le \xi_{\tau\pi}^\mu$ provides $\theta_\tau^\mu = 0$. $\theta_\tau^\mu = 1$ and inequality $\xi_{\tau\pi} - (1 - \theta_\tau^\mu)\overline{\alpha}_{\tau\pi}\overline{\beta}_\tau \le \xi_{\tau\pi}^\mu$ imply $\xi_{\tau\pi}^\mu = \xi_{\tau\pi}$. For $\theta_\tau^\mu = 0$, it follows from $\xi_{\tau\pi}^\mu \le \overline{\alpha}_{\tau\pi}\overline{\beta}_\tau\theta_\tau^\mu$ that $\xi_{\tau\pi}^\mu = 0$. Hence, constraints (12) provide the equivalences

$$\theta_\tau^\mu = 1 \quad \Leftrightarrow \quad \xi_{\tau\pi}^\mu > 0 \quad \Leftrightarrow \quad \xi_{\tau\pi}^\mu = \xi_{\tau\pi}$$

Constraints (13) show the analogous conditions for tasks τ and input products π (i.e., $\xi_{\tau\pi} < 0$):

$$\left.\begin{aligned} \xi_{\tau\pi} \;&\le\; \xi_{\tau\pi}^\mu \;\le\; 0 \\ \underline{\alpha}_{\tau\pi}\overline{\beta}_\tau\theta_\tau^\mu \;&\le\; \xi_{\tau\pi}^\mu \;\le\; \xi_{\tau\pi} - (1 - \theta_\tau^\mu)\underline{\alpha}_{\tau\pi}\overline{\beta}_\tau \end{aligned}\right\} \qquad \begin{array}{c}(\tau \in \mathcal{T}, \pi \in \mathcal{P}_\tau^-, \\ \mu = 1, \ldots, \overline{\varepsilon}_\tau)\end{array} \tag{13}$$

The linear ordering

$$\theta_\tau^1 \ge \theta_\tau^2 \ge \cdots \ge \theta_\tau^{\overline{\varepsilon}_\tau} \qquad (\tau \in \mathcal{T}) \tag{14}$$

of the binary variables θ_τ^μ associated with one and the same task τ ensures that precisely the first ε_τ binary variables $\theta_\tau^1, \ldots, \theta_\tau^{\varepsilon_\tau}$ equal one. The latter condition is not necessary for reducing the nonlinear mixed–integer program (BP) to a linear mixed–binary program. However, it reduces the size of the feasible region considerably without loss of generality.

Now we are ready to formulate the inventory constraints (5) and (6) as linear inequalities:

$$\rho_\pi \le \sum_{\tau \in \mathcal{T}_\pi} \sum_{\mu=1}^{\overline{\varepsilon}_\tau} \xi_{\tau\pi}^\mu \le \rho_\pi + \sigma_\pi \qquad (\pi \in \mathcal{P}) \tag{15}$$

The mixed–binary programming formulation of the batching problem is then as follows:

$$\left.\begin{array}{ll} \text{Min.} & \displaystyle\sum_{\tau\in\mathcal{T}} \overline{p}_\tau \sum_{\mu=1}^{\overline{\varepsilon}_\tau} \theta_\tau^\mu \\ \text{s.t.} & (8) \text{ to } (15) \\ & \theta_\tau^\mu \in \{0,1\} \qquad (\tau \in \mathcal{T}, \mu = 1,\ldots,\overline{\varepsilon}_\tau) \end{array}\right\} \quad (\widetilde{\text{BP}})$$

$(\widetilde{\text{BP}})$ represents a linear mixed–binary program with continuous decision variables $\xi_{\tau\pi}$ ($\tau \in \mathcal{T}$, $\pi \in \mathcal{P}_\tau$), $\xi_{\tau\pi}^\mu$ ($\tau \in \mathcal{T}$, $\pi \in \mathcal{P}_\tau$, $\mu = 1,\ldots,\overline{\varepsilon}_\tau$), and binary decision variables θ_τ^μ ($\tau \in \mathcal{T}$, $\mu = 1,\ldots,\overline{\varepsilon}_\tau$).

The batching problem $(\widetilde{\text{BP}})$ belonging to the WK example (cf. Section 3) with 876 continuous and 768 binary variables has been solved to optimality within 4 seconds using CPLEX 6.0 on an 800 MHz Pentium personal computer. The corresponding optimal solution gives rise to 78 operations to be scheduled on the processing units in the batch scheduling step.

6 Batch scheduling

In this section, we deal with scheduling the processing of the batches, i.e. the operations. Recall that each batch is given by the corresponding task τ, the batch size β_τ, and the proportions $\alpha_{\tau\pi}$. Scheduling means that the start time of each operation and the processing unit on which it is to be carried out have to be determined. The objective is to minimize a regular objective function, where temporal and resource constraints have to be taken into account. In Sections 6.1 and 6.2, we show how the batch scheduling problem can be modelled as a resource–constrained project scheduling problem. In Section 6.3 we are concerned with a solution procedure which exploits concepts of resource–constrained project scheduling. Preliminary versions of the latter procedure have been devised by Schwindt and Trautmann (2000) and Neumann et al. (2001a).

6.1 Batch scheduling and project scheduling

We briefly discuss some basic concepts from project scheduling related to our batch scheduling problem. For more details we refer to Brucker et al. (1999) or Neumann et al. (2001b). Suppose that n operations $1,\ldots,n$ have to be scheduled. We additionally introduce two dummy operations 0 and $n+1$ representing the beginning and termination, respectively, of the production process, which is regarded as a *project*. $\widetilde{V} := \{1,\ldots,n\}$ is the set of *real operations*, and $V := \{0,1,\ldots,n+1\}$ is the set of all operations. In project scheduling, the term *activity* is often used instead of operation. In our problem of detailed production scheduling, we have $n = \sum_{\tau\in\mathcal{T}} \varepsilon_\tau$ and $\widetilde{V} = \cup_{\tau\in\mathcal{T}} V_\tau$, where V_τ with $|V_\tau| = \varepsilon_\tau$ is the set of all operations or batches of task τ.

Let $S_i \geq 0$ be the *start time* of operation i. In particular, we set $S_0 := 0$ and we have $S_{n+1} \leq \overline{d}$, where S_{n+1} coincides with the makespan. $S = (S_i)_{i\in V}$ is called

a *schedule*. The *processing time* of operation i is denoted by p_i and the *cleaning time* after operation i by c_i. In particular, $p_0 = p_{n+1} = 0$ and $c_0 = c_{n+1} = 0$.

In project scheduling, *temporal constraints* for operations of the type $S_j \geq S_i + d_{ij}$ $((i,j) \in E)$ with $E \subseteq V \times V$ are generally prescribed. If $d_{ij} \geq 0$, then $d_{ij}^{min} := d_{ij}$ represents a *minimum time lag* between the start of operations i and j. If $d_{ij} < 0$, then $d_{ji}^{max} := -d_{ij}$ is a *maximum time lag* between the start of operations j and i. In particular, the maximum time lag $d_{0,n+1}^{max} = \overline{d} = -d_{n+1,0}$ guarantees that inequality $S_{n+1} \leq \overline{d}$ is satisfied. Moreover, the following time lags between real operations $i, j \in \widetilde{V}$ may occur in detailed production scheduling, where $\tau(i)$ denotes the task corresponding to operation $i \in \widetilde{V}$.

(a) Operation j has to wait for the completion of operation i because both operations are carried out on the same processing unit or because an output product of $\tau(i)$ represents an input product of $\tau(j)$. Then we set $d_{ij}^{min} := p_i$. If the processing unit has to be cleaned before processing operation j, we have $d_{ij}^{min} := p_i + c_i$.
(b) Operation j cannot be completed until the start of some operation i frees some storage space required for an output product of $\tau(j)$. We set $d_{ji}^{max} := p_j$.
(c) Operation i provides a perishable product that has to be consumed without delay by some operation j. Then we set $d_{ij}^{min} := d_{ij}^{max} := p_i$.

The time lags discussed in (a) and (b) will be introduced in the course of the solution procedure presented in Section 6.3.

It is well–known that an *activity–on–node project network* N with node set V, arc set E, and arc weights d_{ij} can uniquely be assigned to the project in question (see Neumann and Schwindt, 1997). If there is no path in N from node 0 to node $i \in V$, $i \neq 0$, with nonnegative length, we add an arc $(0,i)$ with weight $d_{0i} = 0$. If there is no path in N from node $i \in V$, $i \neq n+1$, to node $n+1$ whose length is at least $p_i + c_i$, we introduce an arc $(i, n+1)$ with weight $d_{i,n+1} = p_i + c_i$. The latter arcs ensure that all operations and cleanings are completed by time S_{n+1}. For simplicity, for the arc set and network enlarged in this way, we again use the symbols E and N, respectively.

A schedule S that satisfies

$$\left.\begin{array}{l} S_0 = 0 \\ S_j \geq S_i + d_{ij} \qquad ((i,j) \in E) \end{array}\right\} \tag{16}$$

is called *time–feasible*. It is well–known that a time–feasible schedule exists exactly if N does not contain any cycle of positive length.

6.2 Resource–constrained project scheduling

We now turn to the different resources. First, we deal with **processing units**. Each processing unit represents a *renewable resource* of capacity 1. Let $\mathcal{R}^\rho$ be the set of all renewable resources and $\mathcal{R}_i^\rho$ be the set of those alternative renewable resources on which operation i can be carried out. Note that $\mathcal{R}_i^\rho = \mathcal{U}_\tau$ for all $i \in V_\tau$. In project scheduling, a pair (i,k) with $k \in \mathcal{R}_i^\rho$ corresponds to an alternative *execution mode*

of operation i (cf. e.g. De Reyck and Herroelen, 1999; Drexl et al., 1997; Heilmann, 2001). For $i \in \widetilde{V}$ and $k \in \mathcal{R}_i^\rho$, let

$$x_{ik} := \begin{cases} 1, \text{ if operation } i \text{ is carried out on resource } k \\ 0, \text{ otherwise} \end{cases}$$

Then we have

$$\left.\begin{array}{ll} x_{ik} \in \{0,1\} & (i \in \widetilde{V}, k \in \mathcal{R}_i^\rho) \\ \displaystyle\sum_{k \in \mathcal{R}_i^\rho} x_{ik} = 1 & (i \in \widetilde{V}) \end{array}\right\} \tag{17}$$

because each real operation i must be carried out on exactly one of the alternative processing units for task $\tau(i)$. Note that a processing unit can operate several different tasks (but only one at a time). Thus, we may have $\mathcal{R}_i^\rho \cap \mathcal{R}_j^\rho \neq \emptyset$ and $x_{ik} = x_{jk} = 1$ for $k \in \mathcal{R}_i^\rho \cap \mathcal{R}_j^\rho$, even if $\tau(i) \neq \tau(j)$. We speak of an *assignment* x of processing units to operations if for all $i \in \widetilde{V}$ and $k \in \mathcal{R}_i^\rho$, the binary variables x_{ik} are specified. An assignment x satisfying (17) is called *complete*.

Since a task and thus all of its operations can be carried out on more than one alternative processing unit, the processing and cleaning times generally depend on the processing unit selected. Let $p_{\tau k}$ be again the processing time of task τ and $c_{\tau k}$ be the cleaning time after performing task τ on processing unit k. Then

$$\begin{aligned} p_i(x) &:= \sum_{k \in \mathcal{R}_i^\rho} p_{\tau(i),k} x_{ik} \quad \text{and} \\ c_i(x) &:= \sum_{k \in \mathcal{R}_i^\rho} c_{\tau(i),k} x_{ik} \end{aligned}$$

represent the processing time and cleaning time, respectively, for operation $i \in \widetilde{V}$ on the processing unit selected. Recall that $p_i = c_i = 0$ for $i = 0, n+1$. We assume that the inequality $c_{\tau k} \leq p_{\tau' k} + c_{\tau' k}$ is satisfied for all τ, τ' with $k \in \mathcal{U}_\tau \cap \mathcal{U}_{\tau'}$, which says that the time needed for cleaning processing unit k after an execution of some task τ does not exceed the processing time of any task τ' on k plus the cleaning time of k after τ'. Since the weight d_{ij} of arc $(i,j) \in E$ in project network N may depend on p_i, p_j, or c_i (see Section 6.1) and the latter quantities generally depend on assignment x, we rewrite constraints (16) as

$$\left.\begin{array}{ll} S_0 = 0 & \\ S_j \geq S_i + d_{ij}(x) & ((i,j) \in E) \end{array}\right\} \tag{18}$$

The cleaning of processing units generally depends on the sequence of operations to be performed. Let $P_k \subseteq \widetilde{V} \times \widetilde{V}$ denote the set of operation pairs (i,j) for which passing from i to j requires a cleaning of processing unit k. For the WK example from Section 3, we have $P_k = \{(i,j) \mid k \in \mathcal{R}_i^\rho \cap \mathcal{R}_j^\rho,\ \tau(j) > \tau(i)\}$. We suppose that relation P_k is *negatively transitive*, i.e., $(h,j) \in P_k$ implies $(h,i) \in P_k$ or $(i,j) \in P_k$ for all $i \in \widetilde{V}$. The latter condition means that when the transition from operation h to operation j requires a cleaning, then the cleaning of k between

h and j cannot be avoided by processing an intermediate operation i. Furthermore, we assume that in order to guarantee purity of products a processing unit must also be cleaned between operations i and j with $(i,j) \notin P_k$ if there is an idle time between the completion of i and the start of j, i.e., $S_j > S_i + p_i(x)$. To formulate the conditions necessitating a cleaning, we introduce the following relation in the set $\widetilde{V}$ of real operations given a processing unit k, a schedule S, and an assignment x:

$$O_k(S,x) := \{(i,j) \in \widetilde{V} \times \widetilde{V} \mid i \neq j, k \in \mathcal{R}_i^\rho \cap \mathcal{R}_j^\rho, S_i \leq S_j, x_{ik} = x_{jk} = 1\}$$

It holds that $(i,j) \in O_k(S,x)$ if $S_j \geq S_i$ and both operations i and j are carried out on processing unit k. Moreover, we define

$$C_k(S,x) := \{(i,j) \in O_k(S,x) \mid (i,j) \in P_k \text{ or } S_j > S_i + p_i(x)\}$$
$$\overline{C}_k(S,x) := O_k(S,x) \setminus C_k(S,x)$$

$C_k(S,t)$ is the set of all pairs (i,j) for which resource k has to be cleaned if operation j is carried out after operation i. $\overline{C}_k(S,t)$ is the set of all pairs (i,j) where operation j has to be started immediately after the completion of operation i. A schedule S is then called *process–feasible* with respect to assignment x if

$$\left.\begin{array}{l} S_j \geq S_i + p_i(x) + c_i(x), \text{ if } (i,j) \in C_k(S,x) \\ S_j = S_i + p_i(x), \text{ if } (i,j) \in \overline{C}_k(S,x) \end{array}\right\} \qquad (k \in \mathcal{R}^\rho) \tag{19}$$

Second, we deal with **storage facilities**, which represent so–called *cumulative resources* (cf. Neumann and Schwindt, 1999). For each nonperishable product $\pi \in \mathcal{P} \setminus \mathcal{P}^p$, there is one cumulative resource keeping its inventory. Let $\mathcal{R}^\gamma$ be the set of all cumulative resources. For each $k \in \mathcal{R}^\gamma$, there are a prescribed minimum inventory $\underline{R}_k$ (safety stock) and a given maximum inventory $\overline{R}_k$ (storage capacity). Each operation $i \in V$ has a demand r_{ik} for resource $k \in \mathcal{R}^\gamma$. If $r_{ik} \geq 0$, the inventory of resource k is replenished by r_{ik} units at time $S_i + p_i(x)$. If $r_{ik} < 0$, the inventory is depleted by $-r_{ik}$ units at time S_i. Let π be the product stocked in storage facility k. Then $\underline{R}_k = 0$, $\overline{R}_k = \sigma_\pi$, and $r_{ik} = \alpha_{\tau(i),\pi}\beta_{\tau(i)}$ for $i \in \widetilde{V}$. Moreover, r_{0k} represents the initial stock of product π and $r_{n+1,k} = 0$.

Let $V_k^+ := \{i \in V \mid r_{ik} > 0\}$ and $V_k^- := \{i \in V \mid r_{ik} < 0\}$ be the sets of operations replenishing and depleting, respectively, the inventory of $k \in \mathcal{R}^\gamma$. Given schedule S and assignment x,

$$\mathcal{A}_k(S,x,t) := \{i \in V_k^+ \mid S_i + p_i(x) \leq t\} \cup \{i \in V_k^- \mid S_i \leq t\}$$

is called the *active set* of activities replenishing or depleting resource $k \in \mathcal{R}^\gamma$ by time $t \in [0,\overline{d}]$. Schedule S is said to be *storage–feasible* with respect to assignment x if

$$\underline{R}_k \leq \sum_{i \in \mathcal{A}_k(S,x,t)} r_{ik} \leq \overline{R}_k \qquad (k \in \mathcal{R}^\gamma, 0 \leq t \leq \overline{d}) \tag{20}$$

Clearly, there exists a storage–feasible schedule only if $\underline{R}_k \leq \sum_{i \in V} r_{ik} \leq \overline{R}_k$ for all $k \in \mathcal{R}^\gamma$ (cf. Neumann and Schwindt, 1999).

A schedule which is time–, process–, and storage–feasible with respect to some assignment x is called *feasible* with respect to x. A pair (S, x) is called a *feasible solution* if x is a complete assignment and if S is feasible with respect to x. The batch scheduling problem consists of finding a feasible solution that minimizes a regular objective function f and can be formulated as follows:

$$\left.\begin{array}{ll} \text{Min.} & f(S) \\ \text{s.t.} & (17) \text{ to } (20) \end{array}\right\} \qquad \text{(BSP)}$$

The decision variables of batch scheduling problem (BSP) are the continuous variables S_i ($i \in V$) and the binary variables x_{ik} ($i \in \widetilde{V}, k \in \mathcal{R}^\rho$). By transformation from the flow shop problem $F||C_{\max}$ it can easily be shown that the feasibility problem for (BSP) is NP–hard. Next, we briefly sketch the generation scheme of a branch–and–bound algorithm for solving the resource–constrained project scheduling problem (BSP).

6.3 Solving the batch scheduling problem

The constraints that make the batch scheduling problem (BSP) intractable arise from the scarcity of renewable and cumulative resources. By relaxing the corresponding constraints (17), (19), and (20) we obtain a *temporal scheduling problem*, where $d_{ij}(x)$ in (18) is replaced with $d_{ij} = \min\{d_{ij}(x) \mid x \text{ satisfies } (17)\}$ (i.e., for each arc $(i, j) \in E$, we replace the mode-dependent arc weight $d_{ij}(x)$ by the minimum weight d_{ij} of (i, j) with respect to all complete mode assignments). The temporal scheduling problem represents a longest path problem in the (single-mode) project network N with arc weights d_{ij} and can be solved by standard network algorithms (cf. Ahuja et al., 1993). Since the temporal scheduling problem is a relaxation of problem (BSP), the longest path length from node 0 to node $n + 1$ in N provides a lower bound on the minimum makespan of feasible schedules.

An optimal solution to this relaxation may be infeasible for (BSP) due to two reasons: (i) there may be operations $i \in \widetilde{V}$ for which no renewable resource $k \in \mathcal{R}_i^\rho$ has been selected so far and thus equation (17) is violated, or (ii) one of the resource constraints (19) or (20) may not be met. When checking inequalities (20), we replace $p_i(x)$ in the definition of active set $\mathcal{A}_k(S, x, t)$ with $\min\{p_i(x) \mid x \text{ satisfies } (17)\}$. In case (i), we select some $k \in \mathcal{R}_i^\rho$ for processing i and replace $\mathcal{R}_i^\rho$ with $\{k\}$. This means that from now on for any assignment x satisfying (17) we have $x_{ik} = 1$ and $x_{ik'} = 0$ for all $k' \in \mathcal{R}_i^\rho \setminus \{k\}$. In case (ii), violations of the resource constraints can be avoided by introducing appropriate time lags between some operations and adding the corresponding arcs to project network N.

By assigning renewable resources to the execution of operations and adding time lags, we obtain a new problem of type (BSP) with a tighter relaxation belonging to a reduced set of complete assignments and an expanded project network again denoted by N. The selection of renewable resources and the addition of time lags is continued until assignment x is complete and the solution to the temporal scheduling problem provides a feasible schedule S or until the temporal scheduling problem is unsolvable because N contains a cycle of positive length. Figure 4

FOR $i = 1, \ldots, n$ **DO**
  IF $|\mathcal{R}_i^\rho| > 1$ **THEN** set $x_{ik} := 0$ for all $k \in \mathcal{R}_i^\rho$;
  ELSE set $x_{ik} := 1$ for all $k \in \mathcal{R}_i^\rho$;
END $(*$ FOR $*)$
REPEAT
  Compute longest path lengths ℓ_{0i} from 0 to all nodes $i \in V$ in the (expanded) project network N;
  IF N contains cycle of positive length **THEN** terminate;
  ELSE set schedule $S := (\ell_{0i})_{i=0}^{n+1}$;
  Scan $\{S_1, S_1 + p_1, \ldots, S_n, S_n + p_n\}$ for the earliest point in time t for which constraints (19) or (20) are not satisfied;
  IF there is such a time t **THEN** select appropriate arc (i, j) with weight $d_{ij}(x)$ and add (i, j) to (expanded) project network N;
  ELSE $(*\, S$ is feasible with respect to $x\, *)$
    Scan $\{S_1, \ldots, S_n\}$ for the earliest point in time t at which some operation i with $|\mathcal{R}_i^\rho| > 1$ is started;
    IF there is such a time t **THEN** select some renewable resource $k \in \mathcal{R}_i^\rho$ and set $x_{ik} := 1$, $\mathcal{R}_i^\rho := \{k\}$;
    ELSE RETURN (S, x); $(*\, (S, x)$ is feasible solution $*)$
  END $(*$ IF $*)$
END $(*$ REPEAT $*)$

Fig. 4. Generation scheme for batch scheduling

shows a generation scheme of a branch–and–bound algorithm, where violations of the resource constraints (19) and (20) are resolved in chronological order and a renewable resource is selected each time schedule S satisfies the resource constraints (19) or (20) with respect to current assignment x. The complete branch–and–bound algorithm is obtained from the generation scheme by first, branching over the arcs (i, j) to be added to N and the renewable resources $k \in \mathcal{R}_i^\rho$ to be selected for processing the real activities $i \in \widetilde{V}$ and second, fathoming enumeration nodes for which the longest path length in N is not less than the makespan of the best schedule found.

We now consider in more detail how to determine appropriate time lags for resolving resource conflicts. Let us assume that schedule S is not process–feasible with respect to x. Then there exist pairs $(i, j) \in O_k(S, x)$ such that

(a) $(i, j) \in P_k$ and $S_j < S_i + p_i(x) + c_i(x)$, or
(b) $(i, j) \notin P_k$ and $S_j < S_i + p_i(x)$, or
(c) $(i, j) \notin P_k$ and $S_i + p_i(x) < S_j < S_i + p_i(x) + c_i(x)$.

We first assume that $(j, i) \in P_k$. Case (a), where $(i, j) \in P_k$, can be dealt with by considering the two alternatives depicted in Figure 5a, where the shaded boxes correspond to cleanings between operations i and j. First, we may introduce the minimum time lag $d_{ij}^{min}(x) := p_i(x) + c_i(x)$ delaying operation j up to the point in time where the cleaning of resource k is terminated. Second, the conflict can be settled by adding the minimum time lag $d_{ji}^{min}(x) := p_j(x) + c_j(x)$ saying that operation i cannot be started before operation j has been completed (note that in this

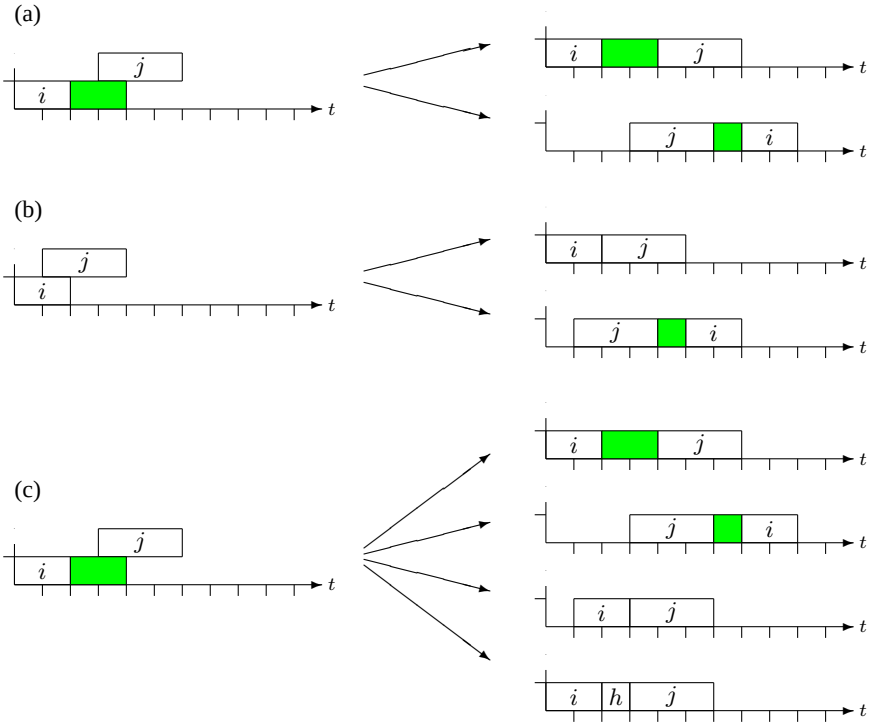

Fig. 5. How to resolve process–infeasibility

case, cleaning of k is also necessary). We now consider the case where $(i,j) \notin P_k$. Case (b) can be handled by interchanging the roles of operations i and j: The two alternatives consist of introducing the minimum time lags $d_{ij}^{min}(x) := p_i(x)$ or $d_{ji}^{min}(x) := p_j(x)+c_j(x)$ (see Fig. 5b). The conditions of case (c) say that there is an idle time between the completion of operation i and the start of operation j which is not sufficiently large for cleaning k (see Fig. 5c). Then the first alternative is to delay j up to the end of the cleaning after the execution of i, i.e., $d_{ij}^{min}(x) := p_i(x)+c_i(x)$. Analogously, i may be delayed up to the end of the cleaning after the execution of j, i.e., $d_{ji}^{min}(x) := p_j(x)+c_j(x)$. A third alternative consists of avoiding the cleaning of k before j starts by introducing the maximum time lag $d_{ij}^{max}(x) := p_i(x)$. Finally, the cleaning of k between i and j can also be avoided by delaying some operation h with $k \in \mathcal{R}_h^\rho$, $S_h < S_i + p_i(x)$, $(i,h) \notin P_k$, $(h,j) \notin P_k$, and $p_h(x) < c_i(x)$ up to the completion of operation i, i.e., $d_{ih}^{min} := p_i(x)$.

If $(j,i) \notin P_k$, no cleaning is required when passing from j to i, i.e., minimum time lag d_{ji}^{min} in cases (a), (b), and (c) is equal to $p_j(x)$.

The cumulative-resource constraints (20) are violated if the inventory of some product falls below zero or exceeds the storage capacity. At first, we consider the case where at some time $t \geq 0$ the inventory of a product π kept in cumulative resource k is negative. We choose an operation i with $S_i + p_i(x) > t$ and $r_{ik} > 0$ and an operation j with $S_j \leq t$ and $r_{jk} < 0$ (cf. Fig. 6a). We then delay the start

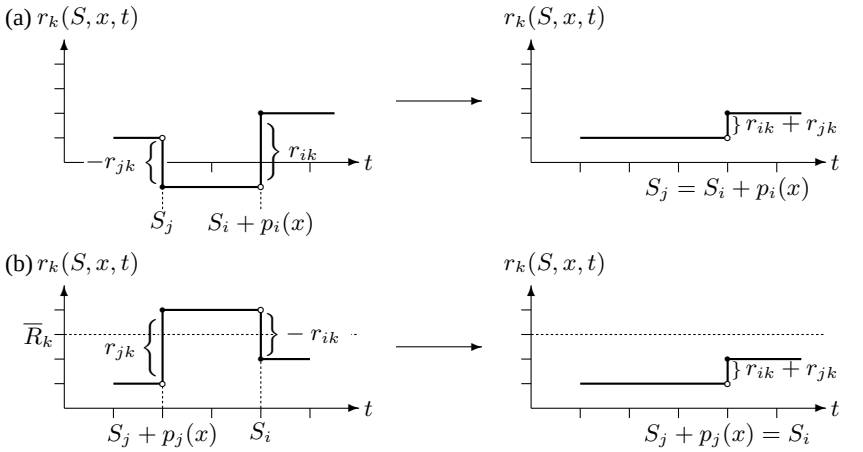

Fig. 6. How to resolve storage–infeasibility

of operation j (i.e. the depletion of k) up to the completion of operation i (i.e. the replenishment of k) by introducing the minimum time lag $d_{ij}^{min}(x) := p_i(x)$. The case where at time $t \geq 0$ the inventory of π exceeds the capacity of k is illustrated in Figure 6b. We select an operation i with $S_i > t$ and $r_{ik} < 0$ and an operation j with $S_j + p_j(x) \leq t$ and $r_{jk} > 0$. The completion of operation j (i.e. the replenishment of k) is delayed up to the start of operation i (i.e. the depletion of k) by introducing the maximum time lag $d_{ji}^{max}(x) := p_j(x)$.

A branch–and–bound algorithm based on the above generation scheme has been implemented in C under MS–Visual C++ 6.0. The batch scheduling problem with 78 operations resulting from an optimal solution to batching problem ($\widetilde{\text{BP}}$) for the WK example has been approximately solved using a *beam search* procedure, that is, a truncated version of the branch–and–bound algorithm. The basic idea of beam search is to select, at each enumeration node, a given number of most promising child nodes for further branching and to fathom the remaining child nodes (cf. Pinedo, 1995). On a Pentium personal computer with 800 MHz clock pulse operating under MS Windows 2000, we have obtained a feasible solution with a makespan of 88 units of time within an imposed running time limit of 56 seconds. Thus, including the 4 seconds for solving ($\widetilde{\text{BP}}$), approximately solving the entire detailed production scheduling problem has taken one minute of computing time. The solution found is the best known thus far for the WK example.

7 Experimental performance analysis

In this section, we are going to test the decomposition approach using 22 instances which have been generated by varying the primary requirements for the final products of the WK example. This test set has been used by Blömer and Günther (1998) as well as Blömer (1999) for evaluating different solution procedures which are based on a monolithic mixed–integer linear programming formulation of the de-

tailed production scheduling problem. We compare our decomposition approach (denoted by B+BS) with the most effective of those methods, the so–called time grid heuristic (TGH). In the performance analysis by Blömer (1999), the cleaning times of tasks 1 to 12 have been assumed to be equal to the respective processing times. Furthermore, the initial stock of products $P2$, $P3$, and $P5$ has been chosen to be equal to 10. For comparison purposes, we have used the modified cleaning times and initial stocks as well.

For each instance, we have imposed a running time limit of one minute to B+BS, have solved the corresponding batching problem to optimality using CPLEX 6.0, and have allotted the remaining computation time to the beam search procedure for the batch scheduling problem. For each instance, Table 3 shows the makespans and CPU times belonging to methods TGH and B+BS. Column "Best makespan" refers to the best feasible solution which could be obtained by some of the heuristics based on mixed–integer linear programming (the data for those procedures have been communicated by Günther, 1999). An asterisk after the makespan for the method B+BS indicates that either a solution proven to be optimal has been found or the best solution known thus far could be improved. The last two columns compare the makespans belonging to methods TGH and B+BS in the case where cleaning of processing units is ignored.

The results displayed in Table 3 indicate that in comparison with the time grid heuristic, the decomposition method generally (for 21 out of 22 instances) finds a better solution within a markedly shorter amount of time. In particular, the results for the four largest instances 19 to 22 show that, compared to the monolithic approaches, procedure B+BS scales quite well. It is worth noting that all batching problems could be solved to optimality, where the computation times vary from 2 seconds (instance 1) to 31 seconds (instance 11). The heuristics based on mixed–integer linear programming perform considerably better when cleanings are neglected. However, the decomposition method still finds solutions of comparable quality within a shorter amount of time.

The good performance of the decomposition method is mainly due to the strong correlation between the workload to be scheduled (i.e., the objective function of the batching problem) and the makespan that can be achieved in the course of batch scheduling. The makespan can generally be decreased only by reducing the workload to be processed in an *active chain* of the schedule found consisting of critical operations whose delay would lead to an increase in the makespan. As a rule, the percentage of critical operations grows as the utilization of the processing units increases. That is why we guess that the scarcer the renewable resources and thus the harder the instances are, the smaller is the loss of accuracy when using the decomposition method.

8 Supplements

The following additional features, which frequently occur when coping with real–world problems, have been integrated into the beam search procedure for batch scheduling:

Table 3. Computational results

Instance	Primary requirements	TGH ms[a]	TGH time[b]	Best ms	B+BS ms	B+BS time[c]	TGH ms[d]	B+BS ms[d]
1	(20,20,20,0,0)	36	1110	36	36 ∗	3	30	30
2	(20,20,0,20,0)	42	2247	38	38 ∗	13	34	35
3	(20,20,0,0,20)	42	2487	38	38 ∗	17	34	34
4	(20,0,20,20,0)	48	1550	38	39	60	35	33
5	(20,0,20,0,20)	44	1778	36	41	60	34	32
6	(20,0,0,20,20)	48	3605	48	43 ∗	60	41	42
7	(0,20,20,20,0)	52	2587	42	38 ∗	60	35	36
8	(0,20,20,0,20)	48	3123	42	39 ∗	60	36	36
9	(0,20,0,20,20)	54	3607	54	53 ∗	60	44	45
10	(0,0,20,20,20)	60	3607	56	50 ∗	60	41	42
11	(10,10,20,20,30)	68	3605	66	66	60	48	53
12	(30,20,20,10,10)	60	3605	52	52	60	42	40
13	(10,20,30,20,10)	64	3604	61	50 ∗	60	47	46
14	(18,18,18,18,18)	66	3606	66	57 ∗	60	48	48
15	(15,15,30,30,45)	148	3622	112	114	60	78	72
16	(45,30,30,15,15)	124	3628	76	80	60	58	62
17	(15,30,45,30,15)	112	3621	88	91	60	71	69
18	(27,27,27,27,27)	124	3631	88	91	60	72	73
19	(20,20,40,40,60)	208	5152	208	135 ∗	60	100	92
20	(60,40,40,20,20)	184	3638	184	100 ∗	60	70	74
21	(20,40,60,40,20)	184	3643	124	112 ∗	60	82	85
22	(36,36,36,36,36)	214	3635	172	134 ∗	60	88	88

[a] Makespan.
[b] CPU time in seconds on a Pentium–266 PC including memory freeing, enumeration stopped after one hour if feasible solution could be found.
[c] CPU time in seconds on a Pentium–800 PC, stopped after one minute.
[d] Without cleanings.

(a) 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. Workers with the same skills are grouped to a pool that is modelled as a renewable resource whose capacity equals the number of workers in the corresponding pool. For the case where the number of available workers is not constant over time, we refer to Schwindt and Trautmann (2000). Möhring and Uetz (2001) have applied resource–constrained project scheduling models and methods to the problem of scheduling tasks with time–varying manpower demand.

(b) If several *identical processing units* are available, those processing units can also be grouped to a pool and modelled as a renewable resource. The process–feasibility of a schedule can then be checked by solving an assignment problem. The basic idea is to examine whether the operations $i \in \widetilde{V}$ can be assigned to the individual processing units $k \in \mathcal{R}_k^\rho$ such that constraints (19) are fulfilled. Grouping identical processing units considerably reduces the number of alternative modes of the activities and thus offers advantages in computation

performance, especially for large instances. For details we refer to Schwindt and Trautmann (2000), who deal with the case of identical processing units that do not require cleaning after idle times.

(c) *Break calendars* provide time intervals (breaks) where some resources are not available. Some activities may be interrupted during a break, others must not be interrupted. The corresponding temporal scheduling problem can be solved efficiently by a polynomial longest path algorithm of type label correcting, where start and completion times of activities are appropriately delayed until all calendar constraints are met (cf. Neumann et al., 2001b; Trautmann, 2001b).

(d) A *quarantine time* $q \geq 0$ says that a product can be consumed q units of time after its production at the earliest. A *shelf life time* $s \geq 0$ implies that a product must be consumed s units of time after its production at the latest. Since one and the same product may be produced by several operations and may be consumed by several operations, it is generally not possible to model quarantine and shelf life times by minimum and maximum time lags between producing and consuming operations. Schwindt and Trautmann (2002) show how to integrate quarantine and shelf life times into the batch scheduling model discussed in Section 6 using fictitious cumulative resources and fictitious activities.

(e) *Campaigns* are used for grouping operations belonging to similar products in order to reduce cleaning times. Once the first operation of a campaign has been started, it is not allowed to start any operation not belonging to the campaign on any processing unit used until the last operation of the campaign on this processing unit has been completed. This leads to an additional type of resource conflicts. For details we refer to Trautmann (2001a).

9 Conclusions

An Advanced Planning System has been considered comprising the modules network design, supply network planning, and detailed production scheduling. The decision problems related to the first two modules have been sketched briefly. The module of detailed production scheduling has been discussed in detail for batch production in process industries. A new approach to solving the corresponding optimization problem has been proposed, which consists of decomposing the problem into batching and batch scheduling. The batching problem can be reduced to a linear mixed–binary program of moderate size and solved by standard software. The batch scheduling problem can be modelled as a resource–constrained project scheduling problem and solved by an efficient beam search procedure. The new approach is markedly superior to the monolithic solution methods known thus far. This is also demonstrated by an experimental performance analysis based on a case study from chemical industry.

Important areas of future research are, for example, the development of an iterative solution procedure for detailed production scheduling, where the batching and the batch scheduling problems are solved alternately, and of efficient decomposition methods for (approximately) solving very large instances with thousands of operations. Moreover, new solution methods for the module of supply network

planning should be developed whose performance is comparable to that of the new approach to detailed production scheduling presented.

References

Ahuja RK, Magnanti TL, Orlin JB (1993) Network flows. Prentice Hall, Englewood Cliffs

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

Berning G, Brandenburg M, Gürsoy K, Mehta V, Tölle FJ (2002) An integrated system solution for supply chain optimization in the chemical process industry. OR Spectrum 24, No. 4 (to appear)

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 (2000) LP–based heuristics for scheduling chemical batch processes. International Journal of Production Research 35:1029–1052

Brucker P, Drexl A, Möhring RH, 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 (2000) Solving a chemical batch scheduling problem by local search. Annals of Operations Research 96:17–36

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

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

Drexl A, Hartmann S, Sprecher A (1997) An exact algorithm for project scheduling with multiple modes. OR Spektrum 19:195–203

Garey MR, Johnson DS (1979) Computers and intractability. Freeman, New York

Grunow M, Günther HO, Lehmann M (2002) Campaign planning for multi–stage batch processes in the chemical industry. OR Spectrum 24:281–314

Günther HO (1999) Personal communication

Heilmann R (2001) Resource–constrained project scheduling: A heuristic for the multi–mode case. OR Spektrum 23:335–357

Holmberg K, Roennqvist M, Yuan D (1999) An exact algorithm for the capacitated facility location problems with single sourcing. European Journal of Operational Research 113:544–559

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

Jayakumar S, Reklaitis GV (1994) Chemical plant layout via graph partitioning — I. single level. Computers & Chemical Engineering 18:441–458

Jayakumar S, Reklaitis G (1996) Chemical plant layout via graph partitioning — II. multiple levels. Computers & Chemical Engineering 20:563–578

Kallrath J (1999) Mixed–integer nonlinear programming applications. In: Ciriani T, Gliozzi S, Johnson E, Tadei R (eds) Operational research in industry, pp 42–76. Ichor Business Books, West Lafayette

Kallrath J (2000) Mixed integer optimization in the chemical process industry — experience, potential and future perspectives. Transactions of the Institution of Chemical Engineers 78:809–822

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

Kallrath J (2002b) Combined strategic and operational planning — A MILP success story in chemical industry. OR Spectrum 24:315–341

Kilger C, Schneeweiss L (2000) Demand fulfillment and ATP. In: Stadtler H, Kilger C (eds) Supply chain management and advanced planning, pp 135–148. Springer, Berlin Heidelberg New York

Knolmayer G, Mertens P, Zeier A (2002) Supply chain management based on SAP systems. Springer, Berlin Heidelberg New York

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

Loos, P (1997) Produktionslogistik in der chemischen Industrie. Gabler, Wiesbaden

Meyr H, Wagner M, Rohde J (2000) Structure of advanced planning systems. In: Stadtler H, Kilger C (eds) Supply chain management and advanced planning, pp 75–78. Springer, Berlin Heidelberg New York

Möhring RH, Uetz M (2001a) Scheduling scarce resources in chemical engineering. In: Kischka P, Leopold–Wildburger U, Möhring RH, Radermacher FJ (eds) Models, methods and decision support in management, pp 195–210. Physica, Heidelberg

Nahmias S (1997) Production and operations analysis. Irvin, Homewood

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, Schwindt C (1999) Project scheduling with inventory constraints. Report WIOR–572, University of Karlsruhe

Neumann K, Schwindt C, Trautmann N (2001a) Short–term planning of batch plants in process industries. In: Kischka P, Leopold–Wildburger U, Möhring RH, Radermacher FJ (eds) Models, methods and decision support in management, pp 211–226. Physica, Heidelberg

Neumann K, Schwindt C, Zimmermann J (2001b) Project scheduling with time windows and scarce resources. Lecture Notes in Economics and Mathematical Systems, Vol 508. Springer, Berlin Heidelberg New York

Oxe G (1997) Reducing overcapacity in chemical plants by linear programming. European Journal of Operational Research 97:337–347

Papageorgiou LG, Pantelides CC (1996) Optimal campaign planning/scheduling of multipurpose batch/semicontinuous plants. 1. Mathematical formulation. Industrial & Engineering Chemical Research 35:488–509

Pinedo M (1995) Scheduling: theory, algorithms and systems. Prentice Hall, Englewood Cliffs

Pinto M, Grossmann IE (1995) A continuous mixed integer linear programming model for short term scheduling of multistage batch plants. Industrial & Engineering Chemical Research 34:3037–3051

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

Schwindt C, Trautmann N (2000) Batch scheduling in process industries: An application of resource–constrained project scheduling. OR Spektrum 22:501–524

Schwindt C, Trautmann N (2002) Storage problems in batch scheduling. In: Chamoni P, Leisten R, Martin A, Minnemann J, Stadtler H (eds) Operations Research Proceedings 2001, pp 213–217. Springer, Berlin Heidelberg New York

Shah N (1998) Single- and multisite planning and scheduling: current status and future challenges. In: Pekny J, Blau GE (eds) Foundations of computer–aided process operations, pp 75–90. Amer. Inst. Chem. Eng., New York

Silver EA, Pyke DF, Peterson R (1998) Inventory management and production planning and scheduling. Wiley, New York

Stadtler H, Kilger C (2000) Supply chain management and advanced planning. Springer, Berlin Heidelberg New York

Timpe C (2002) Solving planning and scheduling problems with combined integer and constraint programming. OR Spectrum 24, No. 4 (to appear)

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

Trautmann N (2001a) Anlagenbelegungsplanung in der Prozessindustrie. Gabler, Wiesbaden

Trautmann N (2001b) Calendars in project scheduling. In: Fleischmann B, Lasch R, Derigs U, Domschke W, Rieder U (eds) Operations Research Proceedings 2000. Springer, Berlin Heidelberg New York

Tsiakis P, Shah N, Pantelides CC (2001) Design of multi–echelon supply chain networks under demand uncertainty. Industrial & Engineering Chemical Research 40:3585–3604

Vidal CJ, Goetschalckx M (1997) Strategic production–distribution models: A critical review with emphasis on global supply chain models. European Journal of Operational Research 98:1–18

Wagner M (2000) Demand planning. In: Stadtler H, Kilger C (eds) Supply chain management and advanced planning, pp 97–115. Springer, Berlin Heidelberg New York

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

Wright G, Goodwin P (1998) Forecasting with judgment. Wiley, New York