Solution Procedures for Production Planning and Detailed Scheduling in Process Industries

Christoph Schwindt

Institut für Wirtschaftstheorie und Operations Research
University of Karlsruhe
D–76128 Karlsruhe, Germany
e–mail: schwindt@wior.uni–karlsruhe.de

Abstract. In this paper we consider solution procedures for short–term planning in process industries operating in batch production mode. The planning problem is decomposed into a batching and a batch scheduling problem. Batching is concerned with the partitioning of total requirements for intermediate and final products into individual batches. In the course of batch scheduling, those batches are allocated to scarce production and storage resources over time. The batching problem can be formulated as a nonlinear mixed–integer programming problem. The batch scheduling problem corresponds to a hybrid shop floor scheduling problem with sequence–dependent cleanings and finite–capacity buffers. For solving the batching problem we propose a transformation into a mixed–binary linear program. The batch scheduling problem is solved by a truncated branch–and–bound method using concepts from resource–constrained project scheduling. An experimental performance analysis based on a collection of benchmark instances from literature shows that short–term planning problems of practical size can be approximately solved within a short amount of time. A generalized version of the batch scheduling algorithm has been integrated into the Advanced Planning System SAP APO.

Key words: Operations planning and scheduling, process industries, Advanced Planning Systems, batching and batch scheduling, mixed–integer programming.

1 Introduction

In process industries, each final product results from a sequence of *tasks* representing physical and chemical transformations of raw materials and intermediate products. In the following, we focus on *batch production*, where all materials flows are discontinuous.

The combination of a task with the quantities of input and output products for one execution of the task is referred to as a *batch*. One task generally corresponds to several batches of the same type. Each batch is processed on one *processing unit* (e.g., a reactor, a heater, or a filter). One and the same batch may be performed on different alternative processing units. For a given task, the *batch size* corresponds to the total quantity of all input products loaded into the processing unit. The batch size has to be chosen within a given range.

The following peculiarities differentiate batch production in process industries from the manufacturing of piece–goods. The processing times of tasks are independent of the respective batch sizes. A processing unit has to be cleaned before executing the next operation when switching to a higher–quality output product or in case of idle times. Certain intermediate substances

are perishable and thus cannot be stored. Other intermediate products have to be stocked in dedicated storage facilities of finite capacity. Recycling of intermediate products implies that a certain amount of the production is not available for matching the primary requirements.

The short–term planning problem consists of allocating processing units and storage facilities over time to the production of given primary requirements for final products such that all operations are completed within a minimum *makespan*. In the context of Advanced Planning Systems, the short–term planning problem is decomposed into a production planning and a detailed scheduling problem (cf. Trautmann, 2001). This decomposition fits the organizational firm structure while markedly decreasing the severe computational requirements incurred by solving the entire short–term planning problem at one time (see Silver et al., 1998, Sect. 13.3). The production planning phase generates appropriate batches (*batching problem*), and the detailed scheduling phase sequences those batches on the processing units subject to inventory constraints (*batch scheduling problem*). The batching problem can be formulated as a nonlinear mixed–integer program, and the batch scheduling problem can be viewed as a resource–constrained project scheduling problem with sequence–dependent cleanings and storage resources (cf. Neumann, 2001).

In the remainder of the paper we proceed as follows. In Section 2 we devise a transformation of the nonlinear batching problem into a mixed–binary linear programming problem. Section 3 is devoted to a branch–and–bound algorithm for solving the batch scheduling problem. This algorithm is based on the relaxation of the resource constraints and enumeration of alternatives for resolving resource–infeasibilities in the resulting schedules. The experimental performance analysis of Section 4 indicates that the combination of both algorithms can be used for solving large–scale problems within a reasonable amount of time. Compared to methods which have been proposed in literature recently, the quality of feasible solutions is improved and the computational effort is markedly reduced.

2 Solving the batching problem

First we recall some notation from Neumann (2001). Let $\mathcal{T}$ be the set of all tasks and let β_τ and ε_τ be the batch size and number of batches for task $\tau \in \mathcal{T}$. By $\mathcal{P}_\tau^-$ and $\mathcal{P}_\tau^+$ we denote the set of input and output products, respectively, of task τ. $\mathcal{P}_\tau := \mathcal{P}_\tau^- \cup \mathcal{P}_\tau^+$ is the set of all input and output products of task τ, and $\mathcal{P} := \cup_{\tau\in\mathcal{T}}\mathcal{P}_\tau$ is the set of all products considered. In addition to β_τ and ε_τ, the proportions $-\alpha_{\tau\pi} > 0$ of all input products $\pi \in \mathcal{P}_\tau^-$ and the proportions $\alpha_{\tau\pi} > 0$ of all output products $\pi \in \mathcal{P}_\tau^+$ have to be determined for all tasks $\tau \in \mathcal{T}$. Batch sizes β_τ and proportions $\alpha_{\tau\pi}$ have to be chosen within prescribed intervals $[\underline{\beta}_\tau, \overline{\beta}_\tau]$ and $[\underline{\alpha}_{\tau\pi}, \overline{\alpha}_{\tau\pi}]$. The number of batches ε_τ is bounded by the maximum number $\overline{\varepsilon}_\tau$ of batches for task τ which can be executed within the planning horizon.

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 $\mathcal{P}^p \subseteq \mathcal{P}$ be the set of perishable products. Then equations $\alpha_{\tau\pi}\beta_\tau = -\alpha_{\tau'\pi}\beta_{\tau'}$ for all $\pi \in \mathcal{P}^p$ and all $(\tau, \tau') \in \mathcal{T}_\pi^+ \times \mathcal{T}_\pi^-$ ensure that the amount of product π produced by one batch of some task $\tau \in \mathcal{T}_\pi^+$ can immediately be consumed by any task $\tau' \in \mathcal{T}_\pi^-$ consuming π.

By ρ_π we denote the primary requirements minus the initial stock of π. For recycled products

$\pi \in \mathcal{P}$, ρ_π is augmented by a residual stock after the completion of the last batches. The inventory of product π after the completion of all batches equals $\sum_{\tau\in\mathcal{T}_\pi} \alpha_{\tau\pi}\beta_\tau\varepsilon_\tau$, where $\mathcal{T}_\pi := \mathcal{T}_\pi^- \cup \mathcal{T}_\pi^+$. This amount must be sufficiently large to match the requirements ρ_π. The final stock $\sum_{\tau\in\mathcal{T}_\pi} \alpha_{\tau\pi}\beta_\tau\varepsilon_\tau - \rho_\tau$ of product π must not exceed the given storage capacity σ_π for π.

With $\overline{p}_\tau$ being the mean processing time of task $\tau \in \mathcal{T}$ on the alternative processing units, the batching problem can be formulated as follows (cf. Neumann, 2001):

$$
\text{(BP)} \quad \left\{ \begin{array}{lll}
\text{Min.} & \displaystyle\sum_{\tau\in\mathcal{T}} \overline{p}_\tau\varepsilon_\tau & (1) \\
\text{s.t.} & \underline{\alpha}_{\tau\pi} \le \alpha_{\tau\pi} \le \overline{\alpha}_{\tau\pi} \qquad (\tau \in \mathcal{T}, \pi \in \mathcal{P}_\tau) & (2) \\
& \displaystyle\sum_{\pi\in\mathcal{P}_\tau^+} \alpha_{\tau\pi} = -\sum_{\pi\in\mathcal{P}_\tau^-} \alpha_{\tau\pi} = 1 \qquad (\tau \in \mathcal{T}) & (3) \\
& \underline{\beta}_\tau \le \beta_\tau \le \overline{\beta}_\tau \qquad (\tau \in \mathcal{T}) & (4) \\
& \alpha_{\tau\pi}\beta_\tau = -\alpha_{\tau'\pi}\beta_{\tau'} \qquad (\pi \in \mathcal{P}^p, (\tau,\tau') \in \mathcal{T}_\pi^+ \times \mathcal{T}_\pi^-) & (5) \\
& \rho_\pi \le \displaystyle\sum_{\tau\in\mathcal{T}_\pi} \alpha_{\tau\pi}\beta_\tau\varepsilon_\tau \le \rho_\pi + \sigma_\pi \qquad (\pi \in \mathcal{P}) & (6) \\
& 0 \le \varepsilon_\tau \le \overline{\varepsilon}_\tau \qquad (\tau \in \mathcal{T}) & (7) \\
& \varepsilon_\tau \in \mathbb{Z} \qquad (\tau \in \mathcal{T}) & (8)
\end{array} \right.
$$

Proposition 1. *NP–hardness of batching*

The feasibility problem for (BP) is NP–hard in the strong sense even if $\mathcal{P}^p = \emptyset$, $\sigma_\pi = \infty$ for all $\pi \in \mathcal{P}$, and $\mathcal{P}_\tau = 1$, $\overline{\varepsilon}_\tau = \infty$, $\underline{\beta}_\tau = \overline{\beta}_\tau$ 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 $\tau_{\lambda\mu}$ transforming π_λ into π_μ with $\underline{\beta}_{\lambda\mu} = \overline{\beta}_{\lambda\mu} = s(\lambda)$ (see Figure 1). 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 $\mu = 1, \ldots, \nu$. Since $\lambda \in A_\mu \Leftrightarrow \varepsilon_{\tau_{\lambda\mu}} = 1$, this condition is met exactly if the 3–PARTITION instance is a yes–instance. □

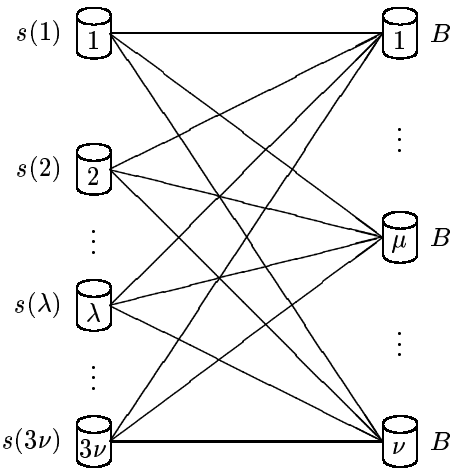

Figure 1: Batching problem to the proof of Proposition 1

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 worse still, poses a serious algorithmic problem. That is why we develop a formulation of the batching problem as an equivalent mixed–binary linear 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.

To linearize the constraints 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 (2) on proportions $\alpha_{\tau\pi}$ can be written as

$$\underline{\alpha}_{\tau\pi}\sum_{\pi'\in\mathcal{P}_\tau^+}\xi_{\tau\pi'} \le \xi_{\tau\pi} \le \overline{\alpha}_{\tau\pi}\sum_{\pi'\in\mathcal{P}_\tau^+}\xi_{\tau\pi'} \qquad (\tau\in\mathcal{T}, \pi\in\mathcal{P}_\tau) \tag{9}$$

The equations

$$\sum_{\pi'\in\mathcal{P}_\tau^+}\xi_{\tau\pi'} = -\sum_{\pi'\in\mathcal{P}_\tau^-}\xi_{\tau\pi'} \qquad (\tau\in\mathcal{T}) \tag{10}$$

correspond to the mass balance constraints (3) and the batch size constraints (4) now read

$$\underline{\beta}_\tau \le \sum_{\pi\in\mathcal{P}_\tau^+}\xi_{\tau\pi} \le \overline{\beta}_\tau \qquad (\tau\in\mathcal{T}) \tag{11}$$

Similarly, the batch size coupling conditions (5) for perishable products can be formulated as

$$\xi_{\tau\pi} = -\xi_{\tau'\pi} \qquad (\pi\in\mathcal{P}^p, (\tau,\tau')\in\mathcal{T}_\pi^+\times\mathcal{T}_\pi^-) \tag{12}$$

In order to eliminate the products $\xi_{\tau\pi}\varepsilon_\tau$ occurring in the inventory constraints (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

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

The number ε_τ of batches for task τ is represented by the sum of $\overline{\varepsilon}_\tau$ binary variables θ_τ^μ, 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., $\xi_{\tau\pi} > 0$):

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

If $\xi^\mu_{\tau\pi} > 0$, then $\xi^\mu_{\tau\pi} \leq \overline{\alpha}_{\tau\pi}\overline{\beta}_\tau\theta^\mu_\tau$ implies $\theta^\mu_\tau > 0$. For $\xi^\mu_{\tau\pi} = 0$, $\xi_{\tau\pi} - (1-\theta^\mu_\tau)\overline{\alpha}_{\tau\pi}\overline{\beta}_\tau \leq \xi^\mu_{\tau\pi}$ provides $\theta^\mu_\tau = 0$. $\theta^\mu_\tau = 1$ and inequality $\xi_{\tau\pi} - (1-\theta^\mu_\tau)\overline{\alpha}_{\tau\pi}\overline{\beta}_\tau \leq \xi^\mu_{\tau\pi}$ imply $\xi^\mu_{\tau\pi} = \xi_{\tau\pi}$. For $\theta^\mu_\tau = 0$, it follows from $\xi^\mu_{\tau\pi} \leq \overline{\alpha}_{\tau\pi}\overline{\beta}_\tau\theta^\mu_\tau$ that $\xi^\mu_{\tau\pi} = 0$. Hence, constraints (13) enforce the equivalences $\theta^\mu_\tau = 1 \Leftrightarrow \xi^\mu_{\tau\pi} > 0 \Leftrightarrow \xi^\mu_{\tau\pi} = \xi_{\tau\pi}$. Constraints (14) show the analogous conditions for tasks τ and input products π (i.e., $\xi_{\tau\pi} < 0$):

$$\left.\begin{array}{rcl} \xi_{\tau\pi} & \leq & \xi^\mu_{\tau\pi} \leq 0 \\ \underline{\alpha}_{\tau\pi}\overline{\beta}_\tau\theta^\mu_\tau & \leq & \xi^\mu_{\tau\pi} \leq \xi_{\tau\pi} - (1-\theta^\mu_\tau)\underline{\alpha}_{\tau\pi}\overline{\beta}_\tau \end{array}\right\} \qquad (\tau \in \mathcal{T}, \pi \in \mathcal{P}^-_\tau, \mu = 1,\ldots,\overline{\varepsilon}_\tau) \tag{14}$$

The linear ordering

$$\theta^1_\tau \geq \theta^2_\tau \geq \cdots \geq \theta^{\overline{\varepsilon}_\tau}_\tau \qquad (\tau \in \mathcal{T}) \tag{15}$$

of the binary variables θ^μ_τ associated with one and the same task τ ensures that the first ε_τ binary variables $\theta^1_\tau,\ldots,\theta^{\varepsilon_\tau}_\tau$ equal one. This condition is not necessary regarding the equivalence of both batching models. However, it reduces the number of feasible solutions considerably.

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

$$\rho_\pi \leq \sum_{\tau\in\mathcal{T}_\pi}\sum_{\mu=1}^{\overline{\varepsilon}_\tau}\xi^\mu_{\tau\pi} \leq \rho_\pi + \sigma_\pi \qquad (\pi \in \mathcal{P}) \tag{16}$$

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

$$(\widetilde{\mathrm{BP}}) \quad \left\{\begin{array}{ll} \text{Min.} & \displaystyle\sum_{\tau\in\mathcal{T}}\overline{p}_\tau\sum_{\mu=1}^{\overline{\varepsilon}_\tau}\theta^\mu_\tau \\ \text{s.t.} & (9) \text{ to } (16) \\ & \theta^\mu_\tau \in \{0,1\} \qquad (\tau \in \mathcal{T}, \mu = 1,\ldots,\overline{\varepsilon}_\tau) \end{array}\right.$$

Westenberger and Kallrath (1995) describe a real–life short–term planning problem from chemical industry which covers all the features discussed in Section 1 (for details see Trautmann, 2001). The corresponding batching problem $(\widetilde{\mathrm{BP}})$ 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.

3 Solving the batch scheduling problem

The problem of scheduling the batches on the processing units subject to inventory constraints can be modelled as a resource–constrained project scheduling problem (cf. Neumann, 2001), where the operations $i \in \widetilde{V} = \{1,\ldots,n\}$ correspond to batches for tasks $\tau(i)$, the renewable resources $k \in \mathcal{R}^\rho$ of capacity 1 correspond to processing units, and the cumulative resources $k \in \mathcal{R}^\gamma$ with safety stock $\underline{R}_k = 0$ and capacity $\overline{R}_k = \sigma_\pi$ correspond to storage facilities keeping products $\pi \in \mathcal{P}$. The requirements for cumulative resources k by operations i are $r_{ik} = \alpha_{\tau(i),\pi}\beta_{\tau(i)}$.

Again, we first list some notation from Neumann (2001). For each operation $i \in \widetilde{V}$ and each renewable resource $k \in \mathcal{R}^\rho_i$ on which i can be carried out, the binary assignment variable x_{ik}

indicates whether or not i is executed on k. Processing and cleaning times depend on assignment $x = (x_{ik})_{i \in \widetilde{V}, k \in \mathcal{R}_i^\rho}$.

Besides an assignment x, we seek a *schedule*, that is, a vector $S = (S_i)_{i \in V}$ of start times S_i for operations $i \in V := \{0, 1, \ldots, n, n+1\}$, where the fictitious operations 0 and $n+1$ represent the project beginning at time $S_0 = 0$ and the project termination at time S_{n+1}. Between the start of certain operations i and j, assignment–dependent minimum and maximum time lags $d_{ij}^{min}(x)$ and $d_{ij}^{max}(x)$ are prescribed. Those time lags can be represented by a project network N with node set V, arc set E, and arc weights $d_{ij}(x)$ for $(i,j) \in E$.

A cleaning between two successive operations i and j on renewable resource k becomes necessary precisely if $\tau(j) > \tau(i)$ or $S_j > S_i + p_i(x)$, where the tasks are assumed to be numbered according to increasing product quality. Let $O_k(S,x)$ be the set of all pairs (i,j) such that $i \neq j$, $S_i \leq S_j$, and i and j are executed on k, i.e., $x_{ik} = x_{jk} = 1$. $O_k(S,x)$ can be partitioned into the set $C_k(S,x)$ containing all pairs (i,j) for which k has to be cleaned if j is processed after i and the set $\overline{C}_k(S,x)$ of pairs for which j must be started immediately after the completion of i.

For $k \in \mathcal{R}^\gamma$, let

$$\mathcal{A}_k(S,x,t) := \{0\} \cup \{i \in \widetilde{V} \mid r_{ik} < 0, S_i \leq t\} \cup \{i \in \widetilde{V} \mid r_{ik} > 0, S_i + p_i(x) \leq t\} \tag{17}$$

be the set of operations i that deplete or replenish cumulative resource k by time $t \geq 0$. Then $r_k(S,x,t) := \sum_{i \in \mathcal{A}_k(S,x,t)} r_{ik}$ is the inventory of product π stocked in cumulative resource k at time t. The batch scheduling problem now can be stated as follows (cf. Neumann, 2001):

$$\text{(BSP)} \quad \left\{ \begin{array}{lll} \text{Min.} & S_{n+1} & & (18) \\ \text{s.t.} & \displaystyle\sum_{k \in \mathcal{R}_i^\rho} x_{ik} = 1 & (i \in \widetilde{V}) & (19) \\ & S_0 = 0,\ S_j \geq S_i + d_{ij}(x) & ((i,j) \in E) & (20) \\ & \left. \begin{array}{l} S_j \geq S_i + p_i(x) + c_i(x), \text{ if } (i,j) \in C_k(S,x) \\ S_j = S_i + p_i(x), \text{ if } (i,j) \in \overline{C}_k(S,x) \end{array} \right\} & (k \in \mathcal{R}^\rho) & (21) \\ & \underline{R}_k \leq r_k(S,x,t) \leq \overline{R}_k & (k \in \mathcal{R}^\gamma, t \geq 0) & (22) \\ & x_{ik} \in \{0,1\} & (i \in \widetilde{V}, k \in \mathcal{R}_i^\rho) & (23) \end{array} \right.$$

A schedule S is called process–feasible if S fulfills constraints (21) and storage–feasible if S satisfies inequalities (22).

By transformation from the flow shop problem $F||C_{\max}$ it can easily be shown that the feasibility problem for (BSP) is NP–hard in the strong sense. Next, we briefly sketch the generation scheme of a branch–and–bound algorithm for solving the resource–constrained project scheduling problem (BSP).

The constraints that make the problem intractable arise from the scarcity of renewable and cumulative resources. By relaxing the corresponding constraints (19), (21), and (22) we obtain a *temporal scheduling problem*, where $d_{ij}(x)$ in (20) is replaced with $\min\{d_{ij}(x) \mid x \text{ satisfies (19)}\}$. The temporal scheduling problem represents a longest path problem in project network N and can be solved by standard network flow algorithms.

An optimal solution to this relaxation may be infeasible for (BSP) due to two reasons: 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 (19) is violated, or one of the resource constraints (21) and (22) may not be met (when checking inequalities (22), we replace $p_i(x)$ and $c_i(x)$ in (17) with $\min\{p_i(x) \mid x \text{ satisfies (19)}\}$ and $\min\{c_i(x) \mid x \text{ satisfies (19)}\}$, respectively). In the former case, we select some $k \in \mathcal{R}_i^\rho$ for processing i by putting $x_{ik} := 1$ and replace $\mathcal{R}_i^\rho$ with $\{k\}$. Violations of the resource constraints can be avoided by introducing appropriate time lags between some operations and adding the corresponding arcs to the project network N.

By assigning renewable resources to the execution of operations and adding time lags we obtain a new (BSP) with a tighter relaxation in an expanded project network N. The selection of renewable resources and the addition of time lags is continued until assignment x is complete and the temporal scheduling yields a feasible schedule S or until the temporal scheduling problem is unsolvable because N contains a cycle of positive length. Figure 2 shows a generation scheme where violations of the resource constraints (21) and (22) are resolved in chronological order and a renewable resource is selected each time schedule S satisfies all resource constraints with respect to current assignment x.

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 put $x_{ik} := 1$ for $k \in \mathcal{R}_i^\rho$.
END (∗ FOR ∗)
REPEAT
  Calculate longest path lengths ℓ_{0i} from 0 to all nodes i in the (expanded) project network N.
  IF N contains cycle of positive length **THEN** terminate (∗ no feasible solution is found ∗)
  ELSE set schedule $S := (\ell_{0i})_{i=0}^{n+1}$.
  Determine earliest point in time $t \in \{S_1, S_1 + p_1, \ldots, S_n, S_n + p_n\}$ for which constraints (21) or (22) are not satisfied.
  IF $t < \infty$ **THEN** add appropriate arc (i, j) with weight $d_{ij}(x)$ to (expanded) project network N.
  ELSE (∗ S is feasible with respect to x ∗)
    Determine earliest point in time t at which some operation i with $|\mathcal{R}_i^\rho| > 1$ is started.
    IF $t < \infty$ **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 ∗)

Figure 2: Generation scheme for batch scheduling

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) $\tau(j) > \tau(i)$ and $S_j < S_i + p_i(x) + c_i(x)$, or

(b) $\tau(j) \leq \tau(i)$ and $S_j < S_i + p_i(x)$, or

(c) $\tau(j) \leq \tau(i)$ and $S_i + p_i(x) < S_j < S_i + p_i(x) + c_i(x)$.

Case (a) can be dealt with by considering the two alternatives depicted in Figure 3a, 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 $d_{ji}^{min}(x) := p_j(x)$ saying that operation i cannot be started before operation j has been completed (note that in this case, cleaning of k is not necessary). 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 Figure 3b). 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 Figure 3c). 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 $d_{ij}^{max}(x) := p_i(x)$.

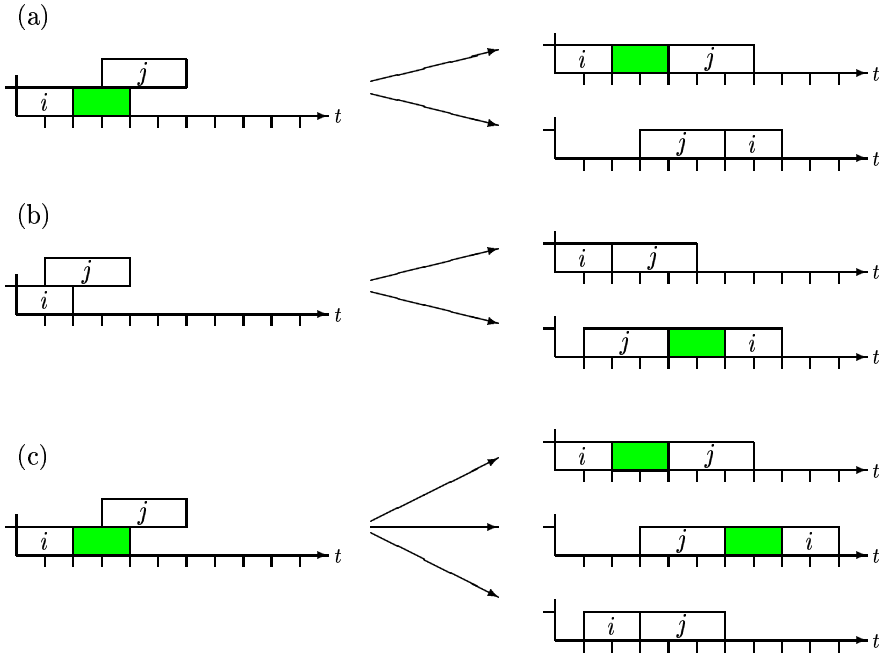

Figure 3: How to resolve process–infeasibility

The cumulative resource constraints (22) 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. Figure 4a). We then delay the start 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 4b. 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 the optimal batches for the Westenberger–Kallrath (WK) example has been approximately solved using a truncated version of the branch–and–bound algorithm (beam search). On a Pentium personal computer with 800 MHz clock pulse operating under MS Windows 2000, we

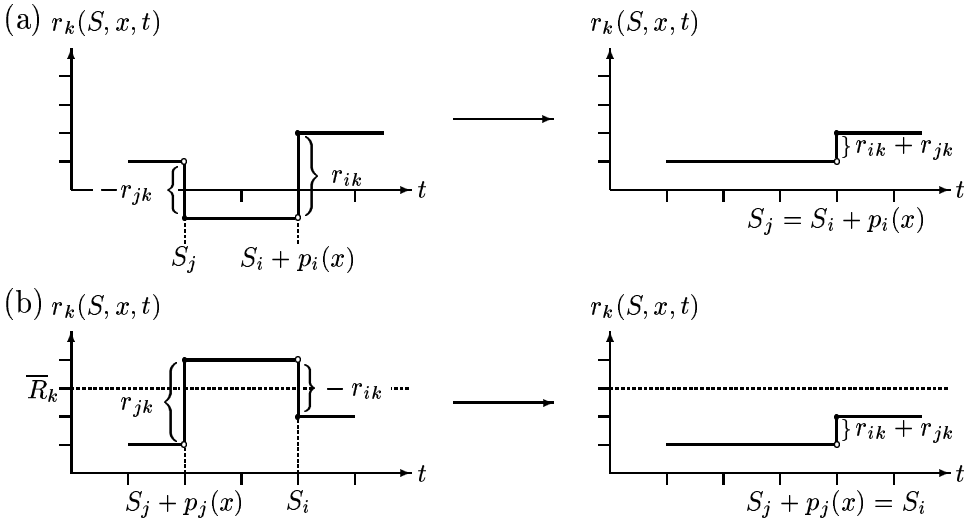

Figure 4: How to resolve storage–infeasibility

have obtained a feasible schedule with a makespan of 88 units of time within an imposed running time limit of 56 seconds. Thus, approximately solving the entire short–term planning problem has taken one minute of computing time. The schedule found is the best solution known thus far for the WK example.

4 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) for evaluating different solution procedures which are based on a monolithic mixed–integer linear programming formulation of the short–term planning problem. We compare our decomposition approach (B+BS) with the best of those methods, the time grid heuristic (TGH).

For each instance, we have imposed a running time limit of one minute, have solved the corresponding batching problem to optimality, and have allotted the remaining computation time to the beam search procedure for the batch scheduling problem. For each instance, Table 1 shows the makespans and CPU times belonging to methods TGH and B+BS. Column "best" 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 B+BS method indicates that either a provably optimal solution has been found or the best solution known until now could be improved.

The results displayed in Table 1 indicate that compared to 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 the B+BS procedure 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).

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

[a] in seconds on a Pentium–266 PC, truncated after one hour if feasible solution could be found
[b] in seconds on a Pentium–800 PC, truncated after one minute

Table 1: Computational results

References

[1] Blömer F. and H.O. Günther (1998). Scheduling of a multi–product batch process in the chemical industry. *Computers in Industry* 36, 245–259.

[2] Garey, M.R. and D. Johnson (1979). *Computers and Intractability.* Freeman, New York.

[3] Günther, H.O. (1999). Personal communication.

[4] Neumann, K. (2001). Modelling of production planning and detailed scheduling for process industries. *Proceedings of IEPM '01.*

[5] Silver, E.A., Pyke, D.F., and Peterson, R. (1998). *Inventory Management and Production Planning and Scheduling.* John Wiley, New York.

[6] Trautmann, N. (2001). Components of Advanced Planning Systems for process industries. *Proceedings of IEPM '01.*

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