18th European Symposium on Computer Aided Process Engineering – ESCAPE 18
Bertrand Braunschweig and Xavier Joulia (Editors)

A Heuristic for the Short-Term Planning of Multi-Product Continuous Plants

Sascha Herrmann, Christoph Schwindt

Clausthal University of Technology, Operations Management Group,
Julius-Albert-Str. 2, 38678 Clausthal-Zellerfeld, Germany

Abstract

We consider the short-term production planning of multi-product continuous plants. In the literature this problem is generally modeled as a large-size monolithic mixed-integer linear or nonlinear program. In this paper we follow a decomposition approach which partitions the problem into an operations planning and an operations scheduling problem. The operations planning problem consists in assigning the production tasks to the processing units, fixing the operating conditions of the tasks, and computing the number of operations executed for each task. This problem can be formulated as a nonlinear program of moderate size containing only one integer variable. The solution to the operations planning problem provides a set of operations, which have to be scheduled on the processing units. For this operations scheduling problem we present a novel mixed-integer linear programming formulation as well as a fast priority-rule based scheduling method. In contrast to the classical monolithic approaches, our new method is able to find good feasible schedules for large-scale instances within less than five seconds.

Keywords: Short-term planning, multi-product continuous plants, real-time scheduling.

1. Introduction

Grade production in the process industries is often performed on multi-product continuous production plants. These plants consist of several multi-purpose processing units operated in continuous mode and storage facilities for stocking the states, i.e., the raw materials, intermediates, and final products. Final products arise from a sequence of chemical or physical transformations called tasks. Each task transforms one or several input states into one or several output states. Sometimes, the input and output proportions are not given in advance, but may be chosen within prescribed bounds. During the execution of a task, the material flows continuously through the processing unit, where we assume the flow rates to be constant over time. The execution of a task on a processing unit during a specified processing time with a given production rate and fixed input and output proportions is referred to as an operation.
In difference to the monolithic problem formulations known from literature (see e.g., *Giannelos and Georgiadis 2002* and *Shaik and Floudas 2007*), we follow a hierarchical decomposition approach, which has originally been proposed by *Neumann et al. 2002* for the short-term planning of batch production. The short-term planning problem quite naturally decomposes into an operations planning and an operations scheduling problem. The heuristic decomposition of the problem allows us to cope with large instances within a very short amount of time. Hence, this approach may be used for practical real-time applications.
We present computational results for a case study introduced by *Munawar et al. 2003*. Processing yields, which can be modeled as output proportions in our approach, lower

and upper bounds on the processing rates, as well as transition times are adopted from the original case study. Initially, the case study has been introduced for a cyclic demand pattern, where the objective consists in maximizing the profit generated within a given time window. As our approach is designed for short-term planning, we solve the dual of the original problem, i.e., we want to minimize the makespan needed to produce the amounts of final products obtained by Munawar et al. for the maximum-profit problem. In practice, the primary requirements generally arise from a mid-term campaign planning which considers the production and logistics activities in a distributed value chain at an aggregate level (see *Meiler et al. 2007* for a recent campaign planning approach).

2. The operations planning problem

In this section we briefly sketch some key characteristics of the operations planning problem. The purpose of operations planning is to assign the tasks to the processing units and to determine feasible operating conditions for the tasks, i.e., the input and output proportions, the flow rates, and the processing times. A formulation of an operations planning problem as a continuous nonlinear programming problem can be found in *Herrmann and Schwindt 2007*. The start of a continuous task requires time, material, and manpower. That is why the number of executions of a continuous task should remain reasonably small (cf. *Méndez and Cerdá 2002*). The model of Herrmann and Schwindt assumes that every task is executed exactly once. Due to the limited availability of storage space for the intermediates, in some cases multiple executions of a task may become necessary. We have integrated the possibility of repeating the execution of tasks into the model by introducing one integer variable ν, which indicates the number of replications of the same plan needed to satisfy the primary requirements. The objective of our operations planning model is the minimization of the workload to be scheduled on the processing units, and, as a second-order criterion, the minimization of the number ν of plan replications.

3. The operations scheduling method

A solution to the operations planning problem yields the set O of operations i with fixed processing times p_i. The operations scheduling problem consists in computing a production schedule with minimum makespan for the execution of the operations $i \in O$ on the processing units $u \in U$ subject to material-availability and storage-capacity constraints for the intermediates. By $O^u \subseteq O$ we denote the set of all operations executed on processing unit u. Let S_i be the start time of operation i and let $\tilde{S} = (S_i)_{i \in O}$ be the production schedule sought. Then the proportion $x_i(\tilde{S}, t)$ of operation i executed at time t given schedule $\tilde{S}$ equals $\max(0, \min(1, (t - S_i)/p_i))$. Between the execution of operations i and j on the same processing unit u, a cleaning of sequence-dependent duration c_{ij}^u is necessary. Moreover, for each state $s \in S$ we are given an initial stock r_0^s and a storage capacity $\overline{R}^s$. By r_i^s we denote the total requirement of operation i for storage space for state s, where we establish the convention that r_i^s is positive if s is produced and negative if s is consumed. Finally, we define O_+^s and O_-^s to be the sets of all operations producing or consuming, respectively, state s.

In the following subsections we present a formulation of the operation scheduling problem as a mixed-integer linear program and an efficient priority-rule based scheduling method for solving the operations scheduling problem.

3.1. Mixed-integer linear programming scheduling model

We exploit the fact that every scheduling problem with storage-capacity constraints can be transformed into an equivalent problem with infinite storage capacities. To eliminate the storage-capacity constraints we introduce for each state s a fictitious state s' with initial stock $r_0^{s'} = \overline{R}^s - r_0^s$ and we put $\overline{R}^s := \overline{R}^{s'} := \infty$ and $r_i^{s'} := -r_i^s$ for all operations $i \in O$.

Our operations scheduling model is based on the representation of feasible schedules devised by *Neumann et al. 2005*. For the formulation of the model we need the sets of pairs $P_1^s := O_+^s \times O_-^s$, $P_2^s := O_-^s \times O_-^s \setminus \{(i,i) \mid i \in O_-^s\}$, $P_3^s := O_+^s \times O_+^s \setminus \{(i,i) \mid i \in O_+^s\}$, $P_4^s := O_-^s \times O_+^s$, and $P^s := \cup_{k=1}^4 P_k^s$. We associate each pair $(i,j) \in P^s$ with a continuous variable x_{ij}^s and a binary variable y_{ij}^s. There always exists an optimal solution to the model such that x_{ij}^s equals the relative progress $x_j(\tilde{S},t)$ of operation j at time $t = S_i$ if $i \in O_+^s$ and at time $t = S_i + p_i$ if $i \in O_-^s$. Finally, we introduce binary variables z_{ij}^u indicating whether or not operation i is executed before operation j on unit u. The mixed-integer linear operations scheduling model then reads as follows:

$$\begin{array}{llll}
\text{Min.} & C_{\max} & & \\
\text{s.t.} & C_{\max} \geq S_i + p_i & (i \in O) & (1) \\
& S_j \geq S_i + p_i + c_{ij}^u - M(1 - z_{ij}^u) & (u \in U; i,j \in O^u : i < j) & (2) \\
& S_i \geq S_j + p_j + c_{ji}^u - M z_{ij}^u & (u \in U; i,j \in O^u : i < j) & (3) \\
& 0 \leq x_{ij}^s \leq 1 & (s \in S; (i,j) \in P^s) & (4) \\
& x_{ij}^s \geq y_{ij}^s & (s \in S; (i,j) \in P_1^s \cup P_2^s) & (5) \\
& x_{ij}^s \leq 1 - y_{ij}^s & (s \in S; (i,j) \in P_3^s \cup P_4^s) & (6) \\
& -p_j x_{ij}^s - M y_{ij}^s \leq S_j - S_i \leq -p_j + M(1 - y_{ij}^s) & (s \in S; (i,j) \in P_1^s) & (7) \\
& -p_j x_{ij}^s - M y_{ij}^s \leq S_j - S_i - p_i \leq -p_j + M(1 - y_{ij}^s) & (s \in S; (i,j) \in P_2^s) & (8) \\
& -M(1 - y_{ij}^s) \leq S_j - S_i \leq -p_j x_{ij}^s + M y_{ij}^s & (s \in S; (i,j) \in P_3^s) & (9) \\
& -M(1 - y_{ij}^s) \leq S_j - S_i - p_i \leq -p_j x_{ij}^s + M y_{ij}^s & (s \in S; (i,j) \in P_4^s) & (10) \\
& r_0^s + \sum_{j \in O^s : j \neq i} r_j^s x_{ij}^s \geq 0 & (s \in S; i \in O_+^s) & (11) \\
& r_0^s + r_i^s + \sum_{j \in O^s : j \neq i} r_j^s x_{ij}^s \geq 0 & (s \in S; i \in O_-^s) & (12) \\
& y_{ij}^s \in \{0,1\} & (s \in S; (i,j) \in P^s) & (13) \\
& z_{ij}^u \in \{0,1\} & (u \in U; i,j \in O^u : i < j) & (14)
\end{array}$$

Inequality (1) defines the production makespan to be equal to the maximum completion time of all operations. Constraints (2) and (3) prevent the overlapping of consecutive operations or cleanings on the same processing unit. Inequalities (4) ensure that proportions x_{ij}^s are between 0 and 1. The link between the continuous variables S_i and x_{ij}^s is established by constraints (5) – (10) via the binary variables y_{ij}^s. Finally, inequalities (11) and (12) formulate the material-availability constraints, which for each state s have to be satisfied at the start times S_i of the producing operations $i \in O_+^s$ and at the completion times $S_i + p_i$ of the consuming operations $i \in O_-^s$.

3.2. Priority-rule based scheduling method

The basic principle of a priority-rule based scheduling method consists in iteratively expanding a partial schedule by adding eligible operations, which are selected based on priority values. In our implementation, an operation is said to be eligible if the required amounts of all input products are available after the completion of the last operation already scheduled. The selected operation j is started at the earliest point in time t_j for which the operation can be executed on the respective processing unit and sufficient input materials are in stock during the entire execution time of this operation.

Thus far we have not taken the limited capacities of the storage facilities into account. Accordingly, the partial schedules may cause capacity overflows in the storage facilities. The storage-capacity constraints are considered via capacity-driven latest start times. If in some iteration we have generated a capacity overflow for some state s at some time t, we temporarily force all eligible operations $i \in O_-^s$ to start no later than time t. Those latest start times are maintained until the capacity overflow for state s has been removed. As a consequence it may happen that the operation j selected for being scheduled cannot be added to the partial schedule because its latest start time is strictly smaller than the earliest feasible start time t_j. In this case we perform an unscheduling step in the following way. At first, we identify the origin of the capacity conflict, i.e., the operation $i \in C \cap O_+^s$ that produced the material that cannot be stocked and hence has determined the latest start time of j. We then increase the earliest start time of i by $t_j - t$ time units by introducing a release date of $S_i + t_j - t$. Finally, we remove all operations from the partial schedule and restart the scheduling procedure. The unscheduling procedure is also invoked in case where all operations have been scheduled and the resulting complete schedule still causes capacity overflows. Algorithm 1 summarizes the procedure described above. The algorithm can be implemented as a multi-start procedure by randomly sampling the priority values of the operations.

Algorithm 1. Priority-rule based scheduling method

```
compute earliest start times $ES_i$ for all operations $i \in O$;
initialize latest start times $LS_i := \infty$ for all operations $i \in O$ and put $C := \emptyset$;
**repeat**
    **if** $C \neq O$ **then**
        determine set $E \subseteq O \setminus C$ of eligible operations;
        select eligible operation $j \in E$ with highest priority value $\pi(j)$;
        determine earliest feasible start time $t_j$ of $j$, disregarding the storage-capacity constraints;
        **if** $t_j \leq LS_j$ **then**
            put $S_j := t_j$ and $C := C \cup \{j\}$;
            update earliest and latest start times $ES_i$ and $LS_i$ of operations $i \in O \setminus C$;
        **end if**;
    **end if**;
    **if** no operation $j$ has been scheduled **then** perform an unscheduling step;
**until** $C = O$ **and** schedule $\tilde{S}$ satisfies all storage-capacity constraints;
**return** schedule $\tilde{S}$;
```

3.3. Postprocessing and concatenation of schedules

The unscheduling steps of the priority-rule based method may lead to unnecessary idle times in the complete schedule. These idle times can be removed by the following post-processing method. From the feasible complete schedule we extract precedence relationships between operations that are executed consecutively on the same processing unit as well as pegging relationships between operations producing and operations consuming the same state. Those relationships are translated into minimum and maximum time lags between the start times of the operations that guarantee the feasibility of the schedule (see *Herrmann and Schwindt 2007*). Next, we determine the earliest start schedule with respect to the minimum and maximum time lags. By construction, this earliest start schedule is feasible and left-shifted with respect to the original schedule. We then extract precedence and pegging relationships from the new schedule and proceed in the same way until a fixed point has been reached. Since the generated sequence of schedules is decreasing and bounded from below, the fixed point always exists and coincides with the limit of the sequence.

If more than one execution of the tasks is needed to satisfy the primary requirements, we construct the overall production schedule by computing a (sub-)schedule $\tilde{S}$ for one plan replication with the mixed-integer linear programming model or the priority-rule based method and by concatenating the required number ν of copies of subschedule $\tilde{S}$. To this end, we at first put the copies in a row by increasing the start times of each operation by the makespan of the preceding subschedule in the sequence. Obviously, the constructed overall schedule is feasible, since subschedule $\tilde{S}$ was feasible. Then, we apply our postprocessing procedure, which deletes the idle times in the final production schedule. As explained above, the resulting schedule is always feasible.

4. Computational experience and conclusions

We have validated our decomposition approach using various case studies from literature. In this section we report on the results obtained for the case study of *Munawar et al. (2003)* mentioned in Section 1. Since we do not consider slopping in our approach, we have increased the primary requirements for the final products by the slopping losses. In detail, this has lead to primary requirements of 8348.00 t for grade A, 10044.07 t for grade B, 2195.47 t for grade C, and 3179.76 t for grade D. The required intermediates and final products have to be produced within a scheduling horizon of 1000 h. The operations planning and operations scheduling models have been implemented under GAMS 22.5, using DICOPT 2 and CONOPT 3.1 as solvers for the operations planning model and CPLEX 10.2 as solver for the operations scheduling model. The priority-rule based scheduling method has been coded in C++. All computations have been performed on an AMD Sempron PC with 1.8 GHz clock pulse and 884 MB RAM running Linux as operating system. Solving the operations planning problem to optimality has taken 0.38 s of CPU time. For the operations scheduling with the mixed-integer linear program and with the priority-rule based method we have imposed a time limit of 9 h and 5 s, respectively. The solution of the operations planning model gives rise to $\nu = 16$ plan replications and a total workload of 4393.70 h. Starting from this solution we have created five scheduling instances, where the operations of 1, 2, 4, 8, or all 16 replications are to be scheduled jointly. Accordingly, for the first four instances the production schedule arises from concatenating 16, 8, 4, or 2 (sub-)schedules.

The results obtained for the mixed-integer linear programming formulation (MILP) and the priority-rule based method (PR) are presented in Table 1. For each of the five scheduling instances, we give the makespans $C_{\max}$ of the subschedules and of the overall production schedule. The CPU times refer to the total time needed to generate the production schedule. The results indicate that the decomposition approach performs quite well. Compared to the benchmark value of 1000 h, the makespan could be significantly improved. Based on the MILP formulation of Section 3, optimal solutions to the operations scheduling problem can be computed within reasonable time for small and medium-sized problem instances. When dealing with large-scale instances containing more than 100 operations, the MILP approach reaches its limits. In contrast, the priority-rule based scheduling method has provided very good solutions for the large-scale instances within 5 s, the small and medium-sized instances even being solved to optimality. It is worth noting that first, the makespans of the subschedules increase sublinearly with the number of operations to be scheduled and second, the makespans of the overall production schedules are nearly independent of the number of replications. The first observation suggests that the priority-rule based method scales quite well, while the second one indicates that only a minor loss in accuracy is incurred by using the concatenation approach.

		1/16/16[1]	2/8/32	4/4/64	8/2/128	16/1/256
MILP	Subschedule $C_{\max}$	73.01	121.03	223.13	—	—
	Overall $C_{\max}$	838.73	835.71	835.71	—	—
	CPU time [s]	0.28	1.67	190.52	> 9 h	> 9 h
PR	Subschedule $C_{\max}$	73.01	121.03	223.13	427.32	847.81
	Overall $C_{\max}$	838.73	835.71	835.71	835.71	847.81
	CPU time [s]	5.24	5.25	5.26	5.36	5.21

Table 1. Computational results for the case study of *Munawar et al. (2003)* obtained with the MILP scheduling model and the priority-rule based method

References

N. Giannelos, and C. Georgiadis, 2002, A novel event-driven formulation for short-term scheduling of multipurpose continuous processes, Ind. Eng. Chem. Res. 41, 2431–2439.

S. Herrmann, and C. Schwindt, 2007, Planning and scheduling continuous operations in the process industries, in H.-O. Günther, D. Mattfeld, and L. Suhl: Management logistischer Netzwerke, Physica, Heidelberg, pp. 279–299.

M. Meiler, H. Günther, and M. Grunow, 2007, Network-wide campaign planning for multi-stage processes in the chemical industry, IEEE IEEM 2007 Conference, 1352–1356.

C. Méndez, and J. Cerdá, 2002, An efficient MILP continuous-time formulation for short-term scheduling of multiproduct continuous facilities, Comp. & Chem. Eng. 26, 687–695.

S. Munawar, M. Bhushan, R. Gudi, and A. Belliappa, 2003, Cyclic scheduling of continuous multiproduct plants in a hybrid flowshop facility, Ind. Eng. Chem. Res. 42, 5861–5882.

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

K. Neumann, C. Schwindt, and N. Trautmann, 2005, Scheduling of continuous and discontinuous material flows with intermediate storage restrictions, EJOR 165, 495–509.

M. Shaik, and C. Floudas, 2007, Improved unit-specific event-based continuous-time model for short-term scheduling of continuous processes: Rigorous treatment of storage requirements, Ind. Eng. Chem. Res. 46, 1764–1779.

[1] Number of operations per task and replication / number of plan replications / total number of operations per replication