Effective Decomposition Algorithm for Multistage Batch Plant Scheduling

Pedro M. Castro,[a] Iiro Harjunkoski,[b] Ignacio E. Grossmann[c]

[a]*Energy Systems Modeling and Optimization Unit, LNEG, 1649-038 Lisboa, Portugal*
[b]*ABB Corporate Research Center, Wallstadter Str. 59, 68526 Ladenburg, Germany*
[c]*Dept. Chemical Engineering, Carnegie Mellon University, Pittsburgh 15213, USA*

Abstract

This paper presents a new algorithm for the scheduling of batch plants with a large number of orders and sequence-dependent changeovers. Such problems are either intractable or yield poor solutions with full-space approaches. We use decomposition on the entire set of orders and derive the complete schedule in several iterations. The key idea is to allow for partial rescheduling without altering the main decisions in terms of unit assignments and sequencing, so that the complexity is kept at a manageable level. It has been implemented with a unit-specific continuous-time model and tested for different decomposition settings. The results show that a real-life 50-order, 17-unit, 6-stage problem can effectively be solved in roughly 6 minutes of computational time.

Keywords: Optimization, Continuous-time, Sequence-dependent changeovers.

1. Introduction

The vast literature in the scheduling area highlights the successful application of optimization approaches to an extensive variety of challenging problems. This important achievement comes from the remarkable advances in modeling techniques, algorithmic methods and computer hardware that have been made in the last two decades. However, there is still a significant gap between theory and practice. New academic developments are mostly tested on relatively small problems whereas real-world applications consist of hundreds of batches, dozens of equipments and long scheduling horizons. In order to make exact methods more attractive for real-world applications, efforts should be oriented towards the development of systematic techniques that allow maintaining the number of decisions at a reasonable level, even for large-scale problems. Although these techniques can no longer guarantee optimality, this may not be critical in practice due to: (i) very short time available to generate a solution; (ii) theoretical optimality can easily get lost due to the dynamic nature of industrial environments; (iii) implementing the schedule as such is often limited by the real process; (iv) only a subset of the actual scheduling goals are taken into account.

In this paper, we address the short-term scheduling of multistage batch plants with sequence-dependent changeovers. A complex algorithm is proposed that can be parameterized for the fast and efficient solution of problems of varying size. The key idea is essentially the one proposed by Röslof et al. (2001) and further explored by Méndez and Cerdá (2003). The novel aspect is the use of a multiple time grid continuous-time model, where the solution process itself overcomes the difficulty in specifying the number of event points that will be different from grid to grid, instead of a model based on sequencing variables. The algorithm is validated through the solution

of example problems of moderate size for which the optimal solution is known, in order to measure the optimality gap and difference in total computational effort.

2. Problem Definition

Given a multistage, multiproduct plant with processing stages $k \in K$, product orders $i \in I$ and units $m \in M$, the goal is to determine the assignment of orders to units and the sequence of orders in each unit so as to minimize the makespan. Orders may be subject to release (r_i) and due dates (d_i) that are enforced as hard constraints. The processing times are unit dependent ($p_{i,m}$) and the duration of sequence-dependent changeovers are given by $cl_{i,i',m}$. A particular unit can handle all orders belonging to set I_m and is allocated to a single stage, with set M_k including those belonging to stage k. Set I_k gives the orders that are handled in stage k. Unlimited intermediate storage and wait policies (UIS/UW) are assumed, while transfer times between units are negligible. The process representation is given in Fig. 1 in the form of a Resource-Task Network.

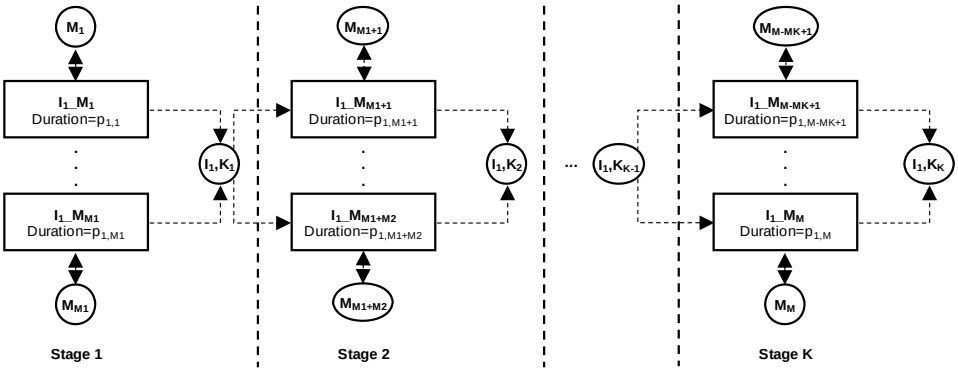

Figure 1. Schematic of a multistage multiproduct plant

3. Key Idea of Decomposition Approach

Finding a schedule for a multistage plant with parallel units involves two decision levels: (i) assigning orders to units; (ii) sequencing orders on every unit. We follow this hierarchy to set different degrees of freedom for the orders. Those being considered for the first time can be assigned to all possible units and take any position in the sequence. In contrast, previously scheduled orders have significantly less freedom. While the timing of events is allowed to change, orders cannot be reassigned to other units. Furthermore, their relative position in the sequence remains unchanged.

Let *NOS* be the number of orders to schedule per iteration. The higher the value, the fewer the iterations (set *J*) and the larger the feasible region up to that of the full-space model (*NOS*=|*I*|) so better solutions are likely to result. However, the resulting mathematical problems also become more complex and may become intractable, so there is an obvious tradeoff.

The second decision concerns order-iteration assignments. We will use the increasing slack times heuristic (MST) that prioritizes orders with a smaller time span (Eq. 1).

$$span_i^{MST} = d_i - r_i - \sum_{k \in K} \min_{\substack{m \in M_i \\ m \in M_k}} p_{i,m} \quad \forall i \in I \tag{1}$$

In the following we heavily rely on the concepts of multiple time grids and combined process and changeover tasks (Castro et al., 2006). More specifically, the execution of combined task (*i*,*i*',*m*) comprises the processing time of order *i* plus the required changeover from *i* to *i*' so that unit *m* is ready for order *i*' to immediately follow. Also, bear in mind that postulating a single time slot per task is enough to ensure generality.

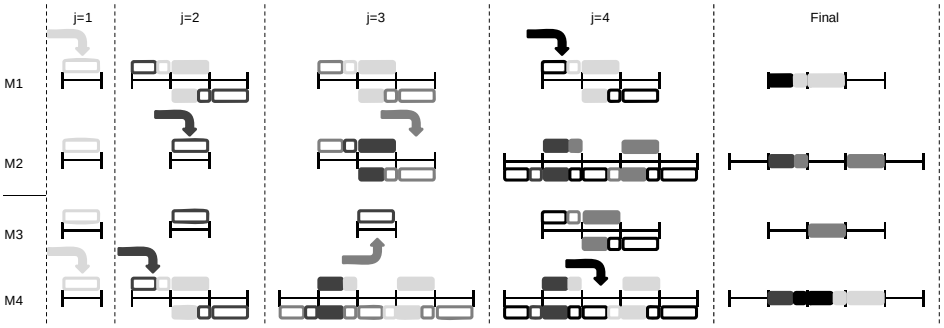

Figure 2. Illustration of decomposition algorithm for *NOS*=1.

The scheduling algorithm is illustrated in Fig.2 for *NOS*=1. In the first iteration (*j*=1), all units get a single time slot to allow for the execution of tasks of type (*i*,*i*,*m*). From *j*=1 to 2, idle units remain with a single time slot, whereas the remaining (M1 and M4) receive two more. This makes it possible to produce the new dark-grey order before or after the already assigned light-grey order. More specifically, the latter can only be executed in slot #2 with either combined task (light-grey, light-grey) or (light-grey, dark-grey). In general, we need to postulate *NOS*+1 additional slots for each previously assigned order. Depending on their relative positioning form the previous iteration ($pos_{i,m}$), such orders will be assigned to slot number $pos_{i,m}\times(NOS+1)$.
Things become increasingly more complicated as we go through iterations. For *j*=3 and M4, we need (dark-grey, light-grey)-A, (dark-grey, medium-grey); (light-grey, medium-grey) and (light-grey, light-grey)-B and, for the new order, (medium-grey, dark-grey)-C, (medium-grey, light-grey) and (medium-grey, medium-grey). It can be easily checked that all possible sequences with dark-grey before light-grey are accounted for. As an example (medium-grey)-(dark-grey)-(light-grey) is achieved with C in slot #1, A in slot #2 and B in slot #4. The procedure continues until the final schedule is obtained. It is important to highlight that there may be idle slots between orders or empty last slots and that no task extends past the duration of a slot. Although it looks like it, in M1 and M4 the first light-grey box accounts for the black-(light-grey) changeover.

4. Scheduling Algorithm

The proposed algorithm comprises two parts. In the first, constructive scheduling, the goal is to find a good initial solution, which is improved afterwards by performing a local search. In this rescheduling part, a couple of orders are released from the schedule to try to find better unit assignments or sequencing. It can be viewed as repeating the last iteration of the constructive step several times, for different order candidates.
Every iteration *j* starts with the selection of orders that are under consideration, I^{act}. As explained in section 3, the number of active time slots for unit *m* (T_m^{act}) is a function on the number of orders previously assigned to it as well as *NOS*. As illustrated in Fig. 2, any previous order can be assigned to a single time slot, determined by its position in

the sequence. New orders can be assigned to any other slot, either before previous orders or after the last one. Such information is then used to generate sets $I_{m,t}$ (orders that in unit m can be executed in slot t) and $I_{i',m,t}$ (orders that can be assigned to slot t of unit m and be followed by order i'). Finally, the last slot is the sole element of T_m^{last} .

5. Mathematical Formulation

The underlying unit-specific continuous-time formulation is essentially CT4I proposed by Castro et al. (2006). The novelty is the introduction of slack variables $S_{t,m}^1$ to allow for due dates violation, an absolute necessity when decomposition algorithms are involved. These are penalized in the objective function (Eq. 2) through weight α=10, which essentially minimizes the makespan, *MS*. 4-index binary variables $\overline{N}_{i,i',m,t}$ identify the execution of the combined tasks at time slot t, while continuous variables $C_{i,m,t}$ keep track of unit states ($C_{i,m}^0$ for the initial state). Timing variable $T_{t,m}$ gives the time of event point t in time grid m, while the transfer time of order i in stage k is $TD_{i,k}$.

$$\min MS + \sum_{m\in M_{|K|}} \sum_{t\in T_m^{active}} \alpha \cdot S_{t,m}^1 \tag{2}$$

$$C_{i,m,t} = C_{i,m}^0\Big|_{t=1} + C_{i,m,t-1}\Big|_{t\neq 1} + \sum_{\substack{i'\in I_{i,m,t-1}\\ i'\neq i}} \overline{N}_{i',i,m,t-1} - \sum_{\substack{i'\in I_m\\ i\in I_{i',m,t}}} \overline{N}_{i,i',m,t} \;\; \forall i\in I^{act}, m\in M_i, t\in T_m^{act} \tag{3}$$

$$\sum_{i\in I_m} C_{i,m}^0 \leq 1 \;\forall m\in M \tag{4}$$

$$T_{t+1,m}\Big|_{t\notin T_m^{last}} + MS\Big|_{t\in T_m^{last}} - T_{t,m} \geq \sum_{i'\in I_m} \sum_{i\in I_{i',m,t}} \overline{N}_{i,i',m,t} \times (p_{i,m} + cl_{i,i',m}) \;\forall m\in M, t\in T_m^{act} \tag{5}$$

$$T_{t,m} \geq \sum_{i'\in I_m} \sum_{i\in I_{i',m,t}} \overline{N}_{i,i',m,t} \times (r_i + \sum_{k'\in K, k'<k} \min_{m'\in M_{k'}\cap M_i} p_{i,m'}) \;\forall k\in K, m\in M_k, t\in T_m^{act} \tag{6}$$

$$T_{t,m} \leq \sum_{i'\in I_m} \sum_{i\in I_{i',m,t}} \overline{N}_{i,i',m,t} ub_{i,i',m} + S_{t,m}^1 + H(1 - \sum_{i'\in I_m} \sum_{i\in I_{i',m,t}} \overline{N}_{i,i'm,t}) \;\forall m\in M, t\in T_m^{act} \tag{7}$$

$$TD_{i,k} \geq T_{t,m} + \sum_{i'\in I_m, i\in I_{i',m,t}} \overline{N}_{i,i',m,t} p_{i,m} - H(1 - \sum_{t'\in T_m^{active}, t'\geq t} \sum_{i'\in I_m, i\in I_{i',m,t'}} \overline{N}_{i,i',m,t'}) \tag{8}$$
$$\forall k \neq |K|, m\in M_k, t\in T_m^{act}, i\in I_{m,t}$$

$$TD_{i,k-1} \leq T_{t,m} + H(1 - \sum_{t'\in T_m^{active}, t'\leq t} \sum_{i'\in I_m, i\in I_{i',m,t'}} \overline{N}_{i,i',m,t'}) \;\forall k\neq 1, m\in M_k, t\in T_m^{act}, i\in I_{m,t} \tag{9}$$

$$TD_{i,k} = TD_{i,k-1} \;\forall k\in K, i\in I^{act}, i\notin I_k \tag{10}$$

$$\sum_{m\in M_k} \sum_{t\in T_m^{act}} \sum_{i'\in I_m, i\in I_{i',m,t}} \overline{N}_{i,i',m,t} = 1 \;\forall k\in K, i\in I^{act} \cap I_k \tag{11}$$

Equation 3 gives the resource balances over the equipment states. Notice in the third term on the right-hand side that combined tasks with a single order index (i,i,m) do not need to regenerate the equipment state since no order follows. Eq. 4 limits the initial states of units to a single order. The central timing constraint is given by Eq. 5. It ensures that the difference in time between two consecutive event points must be greater than the processing time of the order being executed plus the required changeover time

for the following order. Eqs. 6-7 are the release and due date constraints, where slacks $S^1_{t,m}$ allow for violation of the due dates (note that $ub_{i,i',m}=f(d_i)$, see Castro and Novais, 2009). These are needed for all pairs (slot, unit) and not just for those penalized in the objective function (Eq. 2). The transfer time of order *i* in stage *k* (e.g. in hours) must be greater than its ending time in stage *k* and lower than its starting time in *k*+1 (Eqs. 8-9). The transfer times of orders not involved in stage *k* are equal to those in the previous stage (Eq. 10). Finally, all orders need to be executed once on every stage (Eq. 11).

6. Computational Results

The performance of the scheduling algorithm is illustrated through the solution of ten example problems. P7-P13 are taken from Castro and co-workers (2006, 2009) and can be solved to global optimality by the full-space continuous-time formulation. Their purpose is to evaluate the quality of the solution returned by the algorithm. P16 is the challenging industrial problem, provided by a pharmaceutical batch plant, with P14-P15 considering the first 30/40 orders to measure the effect of problem size on computational effort. The algorithm and underlying models were implemented in GAMS 23.2 with CPLEX 12.1 as the MILP solver. The termination criteria were a relative optimality tolerance=10^{-6} and maximum computational time per iteration=3600 CPUs on the constructive part; 60 CPUs on the rescheduling part. The hardware was a HP laptop with an Intel Core2 Duo T9300 2.5 GHz processor running Windows Vista.

The solutions obtained are listed in Table 1 for a total of 30 trials resulting from different choices of parameter *NOS*. All generated problems from the smaller instances were solved to optimality, so failure to find the global optimal solution is entirely due to the decomposition strategy. Interestingly, there was not a single successful (resulting in an optimal solution) run and solutions with due date violations were observed in 19% of the cases. Increasing the value of *NOS* leads to an average increase in solution quality, which comes as no surprise since one is working closer to full-space mode (*NOS*=|*I*|).

The focus is really on difficult large-scale problems and on obtaining near optimal solutions fast. From the last columns, one can see that reasonably good solutions can be found in roughly 6 minutes of computational time. For *NOS*=2, P14 and P15 can still be solved in less than 1 hour but for P16 it is no longer possible to solve the problems generated in the last few iterations to optimality. As a consequence, solution quality degrades from *NOS*=2 to 3, contrary to what happens for most of the smaller instances.

The major strength of the proposed algorithm is the ability to address problems systematically for a wide variety of settings. In particular, one can switch to a continuous-time model with general precedence sequencing variables. While a similar performance was observed, we did found a better solution for the most challenging problem (*NOS*=3) featuring a makespan=47.722 h, see Fig. 3. It is also important to highlight that the full-space version of the model could only find a makespan=36.121 h for P14 before running out of memory at 51,900 CPUs, a value 16% higher than the best solution found by the decomposition algorithm in considerably less time.

7. Conclusions

This paper has addressed the scheduling of large-scale multistage batch plants with sequence-dependent changeovers. A new decomposition algorithm was proposed to construct the full schedule in several iterations. The complete set of orders is handled sequentially according to the increasing slack times heuristic. Newly considered orders are given total freedom in terms of unit assignments and sequencing, while those

handled in previous iterations are allowed to change their starting times provided that their unit assignments and relative positions in the sequence are kept constant. By changing the number of orders tackled per iteration, the algorithm can effectively handle problems of different size in an efficient way and, hence, take the most of available computational resources. In particular, it successfully tacked an industrial scale problem in a few minutes of computational time.

Table 1. Computational Statistics (**Best Solution**, *Solution with Due Dates Violation*)

			Solution for NOS=			CPUs for NOS=		
Problem	(I,M,K)	Optimum	1	2	3	1	2	3
P7	(8,6,2)	542	591	595	**558**	3.89	2.26	4.10
P8	(8,6,2)	584	664	611	**587**	3.78	2.38	2.27
P9	(8,6,3)	915	*1355*	**981**	*1166*	3.93	2.34	13.0
P10	(8,6,3)	914	970	**938**	**938**	3.79	3.22	4.91
P11	(12,6,2)	233	261	**245**	261	5.86	4.01	12.9
P12	(8,8,4)	265	*374*	*507*	**275**	4.12	7.42	27.2
P13	(15,4,2)	273	314	328	**309**	7.54	5.50	21.1
P14	(30,17,6)	?	33.424	**31.018**	31.934	40.9	1607	20540
P15	(40,17,6)	?	42.056	**40.766**	42.151	99.7	3251	30409
P16	(50,17,6)	?	52.849	**48.929**	51.444	371	12690	29413

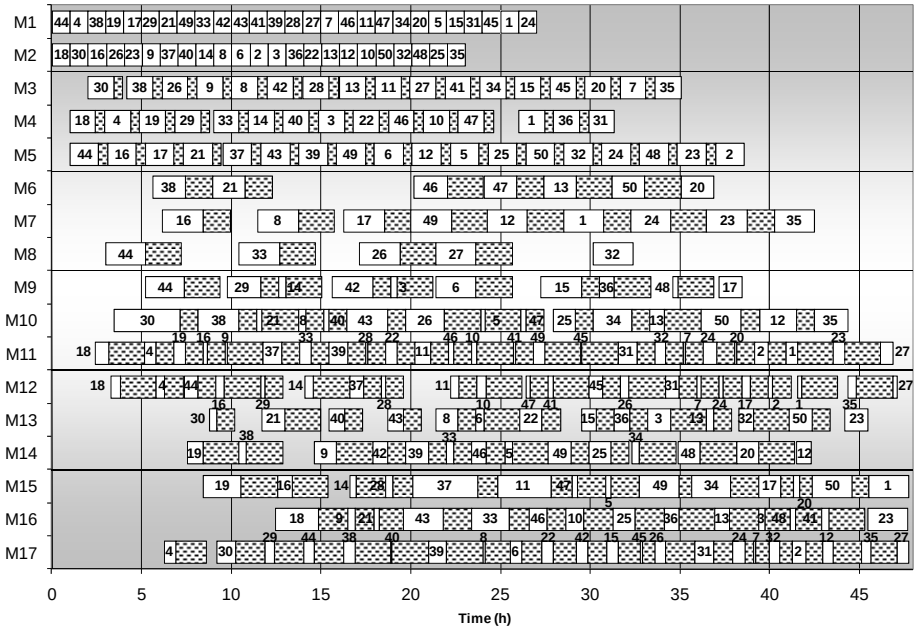

Figure 3. Best found solution for P16.

References

P. Castro , I. Grossmann, A. Novais, 2006, Two New Continuous-Time Models for the Scheduling of Multistage Batch Plants with Sequence-Dependent Changeovers. Ind. Eng. Chem. Res. 45, 6210.

P. Castro, A. Novais, 2009, Scheduling Multistage Batch Plants with Sequence-Dependent Changeovers. AIChE J. 55, 2122.

C. Méndez, J. Cerdá, 2003, Dynamic Scheduling in Multiproduct Batch Plants, Comput. Chem. Eng. 27, 1247.

J. Roslöf, I. Harjunkoski, J. Björkqvist, S. Karlsson, T. Westerlund, 2001, An MILP-based Reordering Algorithm for Complex Industrial Scheduling and Rescheduling. Comput. Chem. Eng. 25, 821.