European Symposium on Computer Aided Process Engineering – 15
L. Puigjaner and A. Espuña (Editors)

A Continuous-Time Formulation for Scheduling Multi-Stage Multi-product Batch Plants with Non-identical Parallel Units

Yu Liu and I. A. Karimi*
Department of Chemical & Biomolecular Engineering
National University of Singapore
4 Engineering Drive 4, Singapore 117576

Abstract

Multi-stage multi-product batch plants with non-identical parallel units are quite common in the batch chemical industry. Scheduling production in these plants optimally is a complex task that has received limited attention in the literature. In this work, we propose a new continuous time-representation using asynchronous slots and report a mixed integer linear programming (MILP) formulation for the short-term scheduling of such plants. Using several examples, we show that our formulation is simpler, tighter, and performs better than previous formulations.

Keywords: MILP; multi-product; batch plant; scheduling; makespan

1. Introduction

Scheduling of plant operations is a routine activity in all batch plants. Although the task of optimal scheduling is complex in even simpler batch plant configurations such as serial multi-product process, it is even more difficult in the case of multi-stage multi-product batch plants with non-identical parallel units. However, in spite of their industrial significance, the existing research has paid little attention to the scheduling in plants with multiple stages.

Pinto and Grossmann (1995) developed a continuous-time, slot-based MILP formulation for the short-term scheduling of multistage batch plants. They assumed sequence-independent set-ups and unconstrained resources. To assign orders to the time slots on a unit, they used tetra-index (order, slot, unit, and stage) binary variables. This increased significantly the number of binary variables for larger problems. Hui, Gupta, and Meulen (2000) presented a formulation with a set of tri-index binary variables (order, order, stage) instead of the tetra-index binary variables, i.e. (order, order, stage, unit). Although they could deal with sequence-dependent set-up times, they could solve problems with only five orders to optimality in reasonable time. Recently, Gupta and Karimi (2003) developed several improved sequence-based MILP formulations. In contrast to Pinto and Grossmann (1995), they used tri-index variables (order, order, and stage) to handle order-sequence dependencies explicitly. Compared to other sequence-

* Corresponding author. E-mail: cheiak@nus.edu.sg

based formulations in the literature, their formulations used fewer binary variables and constraints, solved faster, and gave better objective values in some cases. All these formulations are still not suitable for larger problems, and more effective formulations and algorithms are badly needed for this difficult problem.

In this paper, we develop a more effective approach for this tough scheduling problem. We begin this paper with a detailed problem description and then develop an MILP formulation for this problem. Next, we compare our model with the existing models to demonstrate the superiority of our model with three examples.

2. Problem Description

Figure 1 shows a multi-stage multi-product batch process with non-identical parallel units. The plant has S batch stages (s = 1, 2, …, S) and a total of J batch units (j = 1, 2, …, J). We define set $\boldsymbol{U}_s$ = {j | unit j is in stage s} and $m_s = |\boldsymbol{U}_s|$, i.e. $J = m_1 + m_2 + \ldots + m_S$. We assume that unlimited intermediate storage (UIS) exists between consecutive processing stages.

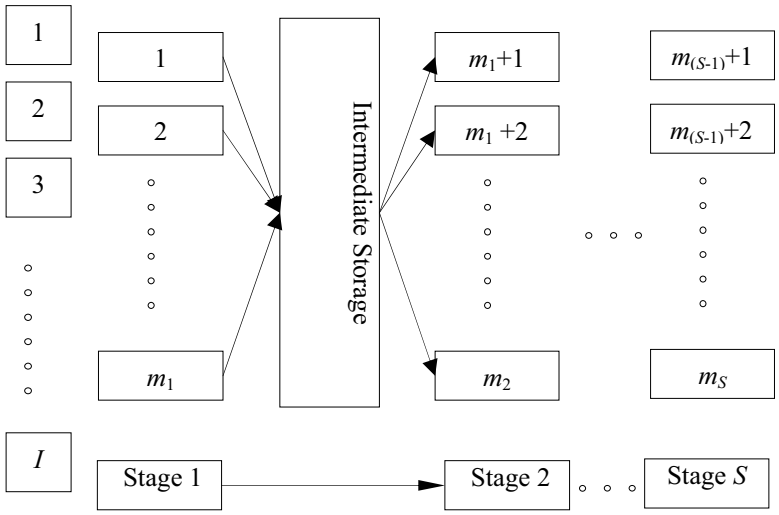

Figure 1. Structure of the scheduling problem

The plant operates in a batch mode, i.e. produces individual batches of products rather than long campaigns of identical batches. Let I denote the number of batches (i = 1, 2, …, I) that the plant must process. Some of these batches may be identical. We assume that each batch follows the sequence 1, 2, …, S of stages for processing. If a batch skips a specific stage, then its processing time is zero on all units in that stage. When parallel units are non-identical, a unit may not be able to process all batches. To accommodate these preferences, we define $\boldsymbol{I}_j$ = {i | batch i can be processed on unit j} and $\boldsymbol{J}_i$ = {j | unit j can process batch i}. A real scheduling problem could involve tens of batches, 2-5 stages, and 20-30 batch units.

In addition to the above process features, we assume the following.

1. A unit cannot process more than one batch at a time.
2. Processing is non-preemptive.
3. Processing units do not fail and processed batches are always satisfactory.

4. Start of the current scheduling period is zero time.
5. Neglect transition times.
6. The size of each batch is known *a priori*.
7. More than one unit cannot process a single batch.
8. Resources are unlimited.

With these, we can state a short-term scheduling problem for the above process as follows. Given $\boldsymbol{J}_i$, $\boldsymbol{I}_j$, $\boldsymbol{U}_s$, and the processing times of batches on suitable units, identify the units that should process each batch and the start/end times of all batches on each unit to optimize some scheduling criterion. We use makespan as the scheduling objective.

3. MILP Formulation

The main feature of our slot-based formulation is that we define time slots for each stage and each unit separately. Instead of assigning batches to unit-slots, we assign them to stage-slots and then derive their assignments to specific units later. By doing so, we reduce the number of binary variables dramatically compared with the existing slot-based formulations.

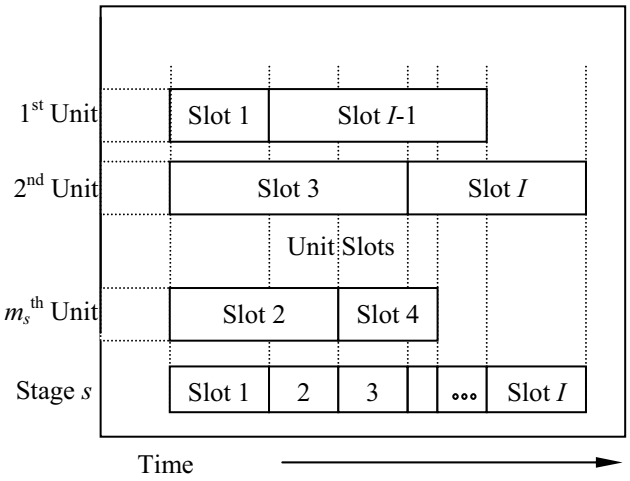

Figure 2. Definition and time-matching of slots in a stage s

We represent time (Figure 2) on each stage s and each unit j by exactly I contiguous, variable-length slots (k = 1, 2, …, I). Note that the unit-slots for a stage are asynchronous with their stage-slots and stage-slots are asynchronous across stages. Let T_{ks} be the time at which slot k ends on stage s and t_{jk} be the time at which it ends on unit j. As shown in Figure 2, the endpoint of each stage-slot matches with the endpoint of exactly one slot of some unit in that stage. For instance, $T_{3s} = t_{23}$ in Figure 2. For each stage-slot, we use the following binary variable and constraint to assign exactly one batch, which must exit the stage at the end of that slot.

$$ye_{iks} = \begin{cases} 1 & \text{if batch } i \text{ exits stage } s \text{ at time } T_{ks} \\ 0 & \text{otherwise} \end{cases}$$

$$\sum_i ye_{iks} = 1 \tag{1}$$

Since a batch must pass through each stage once and only once, we have,

$$\sum_k ye_{iks} = 1 \tag{2}$$

Having assigned a batch to each stage-slot, we now assign a unit to each batch by defining the following binary variable.

$$z_{ijs} = \begin{cases} 1 & \text{if unit } j \text{ processes batch } i \\ 0 & \text{otherwise} \end{cases} \qquad j \in \boldsymbol{U}_s$$

Note that the index s in the above is not free. Fixing j fixes s, as $j \in \boldsymbol{U}_s$. Since only one unit in each stage must process a given batch, we write the following for each stage s,

$$\sum_{j \in \boldsymbol{U}_s,\, j \in \boldsymbol{J}_i} z_{ijs} = 1 \tag{3}$$

Using the two binary variables ye_{iks} and z_{ij}, we define a 0-1 continuous variable $YZ_{ijks} = ye_{iks}z_{ijs}$. We linearize this by using,

$$\sum_{j \in U_s,\, j \in J_i} YZ_{ijks} = ye_{iks} \tag{4a}$$

$$\sum_k YZ_{ijks} = z_{ijs} \qquad j \in \boldsymbol{U}_s, j \in \boldsymbol{J}_i \tag{4b}$$

Having assigned batches to slots and units, we now determine the completion times of batches and slot times. First, slots on each stage are contiguous and ordered, so

$$T_{ks} \geq T_{(k-1)s} \qquad T_{k0} = 0 \tag{5}$$

Note that $YZ_{ijks} = 1$, only when a batch i exits a unit j at some slot k (or time t_{jk}), otherwise it is zero. Therefore, to ensure that the processing of every batch that exits a unit j at t_{jk}, we use,

$$t_{jk} \geq t_{j(k-1)} + \sum_{i \in \boldsymbol{I}_j} PT_{ij} YZ_{ijks} \qquad j \in \boldsymbol{U}_s,\ t_{j0} = URT_j \tag{6}$$

Where, URT_j is the release time of unit j, which is the earliest time at which unit j is ready to accept a new batch. Note that if a batch does not exit a unit j at t_{jk}, then $t_{jk} = t_{j(k-1)}$. Thus, only the unit-slot from which a batch leaves provides the processing time (PT_{ij}). Now, if a batch does leave a unit j at t_{jk}, then we must register this exit on the time axis of the corresponding stage as well by forcing $T_{ks} = t_{jk}$. We do this as follows.

$$T_{ks} \geq t_{jk} \qquad j \in \boldsymbol{U}_s \tag{7a}$$

$$T_{ks} \leq t_{jk} + M\left(1 - \sum_{i \in \boldsymbol{I}_j} YZ_{ijks}\right) \qquad j \in \boldsymbol{U}_s \tag{7b}$$

Where, M is a constant larger than the maximum possible makespan. Having ensured that each batch gets sufficient time for processing on a suitable unit, we must ensure that a stage s processes a batch only after stage (s–1) has completed it. To this end, we

define CT_{is} as the completion time of batch i on stage s, i.e. the time at which it exits stage s. Clearly, this is the time at which the stage-slot to which the batch is assigned ends, therefore,

$$CT_{is} = \sum_{k} T_{ks} ye_{iks}$$

Then, to ensure that a batch processes on a stage only after it has finished processing on the previous stage, we require,

$$CT_{is} \geq CT_{i(s-1)} + \sum_{j \in \boldsymbol{U}_s} PT_{ij} z_{ijs} \qquad CT_{i0} = BRT_i$$

where BRT_i is the release time of batch i.
Next, we define $TYE_{iks} = T_{ks} ye_{iks}$ to linearize the nonlinear expression for CT_{is} with the constraints below.

$$CT_{is} \geq T_{ks} - M(1 - ye_{iks}) \tag{8a}$$

$$CT_{is} \leq T_{ks} + M(1 - ye_{iks}) \tag{8b}$$

Lastly, we minimize the makespan (H) given by any one of the following constraints. Although it is sufficient to use just one constraint and most of the following constraints are redundant, we found that using all three reduced the solution time.

$$H \geq T_{ks} \tag{9a}$$

$$H \geq CT_{is} \tag{9b}$$

$$H \geq t_{jk} \tag{9c}$$

4. Model Evaluation

To evaluate this formulation and compare it with those of Pinto and Grossmann (1995) and Gupta and Karimi (2003), we use three examples. Example 1 has two batch stages ($S = 2$) and process nine batches ($I = 9$). Stage 1 has two parallel units ($\boldsymbol{U}_1 = \{1, 2\}$ and $m_1 = 2\}$ and stage 2 has three ($\boldsymbol{U}_2 = \{3, 4, 5\}$ and $m_2 = 3\}$. Table 1 gives the remaining data for this example.

Table 1. Processing time (h) of batches on units in Example 1

Batch (i)	Unit (j)				
	1	2	3	4	5
1	15	5	62	44	33
2	6	2	20	42	70
3	12	14	51	63	17
4	16	12	17	50	41
5	16	12	37	64	24
6	12	8	53	34	21
7	4	19	21	37	23
8	18	17	24	43	39
9	11	2	10	34	36

For solving the three examples, we used CPLEX 9.0 in GAMS 21.4 on an hp workstation xw6200 (Intel Xeon 3.40 GHz CPU and 3.00 GB RAM) running Windows XP. Table 2 shows the performances of and statistics for various models. Our model clearly performs much better than the existing two models. From Table 2, we see that only our model finds the optimal solution within 1000 s for all examples. Furthermore, our model also gives higher RMIP values for all the problems, which indicates that our formulation can be tighter than the other two models. In spite of using more binary variables, our model is much faster than the model of Gupta and Karimi (2003). Compared to the slot-based model of Pinto and Grossmann (1995), our model uses almost half the binary variables.

Table 2. Comparison of our model with two literature models for the examples

Example 1 (S=2, m_1=2, m_2=3, I=9)									
Model	Binary Variable	Non-zeros	Vari-ables	Cons-traints	RMIP (h)	CPU Time (s)	Relative Gap (%)	MIP (h)	Nodes
Ours	207	3669	694	832	64.11	146	0	84	35390
Gupta	152	2456	209	613	41	1000	16.67	84	409994
Pinto	360	3524	473	838	41	1000	0.53	84	619954
Example 2 (S=2, m_1=3, m_2=4, I=8)									
Ours	184	3633	721	770	22.65	23	0	34	5244
Gupta	168	3112	241	743	22	1000	17.65	34	51720
Pinto	504	4908	649	1154	22	588	0	34	406222
Example 3 (S=3, m_1=3, m_2=2, m_3=5, I=8)									
Ours	272	5273	1041	1134	103.83	72	0	148	10646
Gupta	248	4504	353	1082	66	1000	32.21	149	398600
Pinto	720	7016	929	1652	66	194	0	148	36881

5. Conclusion

An MILP formulation using a novel continuous-time asynchronous slot-based time representation was developed for scheduling multi-stage multi-product batch plants with non-identical parallel units. Evaluation on three examples shows that our formulation is simpler, tighter, and at least two times faster than the best existing formulations.

References

Hui, C. H.; Gupta, A.; Meulen, H. A. J. A novel MIP formulation for short-term scheduling of multi-stage multi-product batch plants with sequence-dependent constraints. Comput. Chem. Eng. 2000, 24, 2705.

Gupta, S.; Karimi, I. A.; An improved MILP formulation for scheduling multiproduct, multistage batch plants, Ind. Eng. Chem. Res.; 2003; 42(11); 2365.

Pinto, J. M.; Grossmann, I. E. A continuous time mixed integer linear programming model for short-term scheduling of multistage batch plants. Ind. Eng. Chem. Res. 1995, 34, 3037.