A Novel Approach to Scheduling Multipurpose Batch Plants Using Unit-Slots

Naresh Susarla, Jie Li, and I. A. Karimi
Dept. of Chemical and Biomolecular Engineering, National University of Singapore, Singapore 117576, Singapore

DOI 10.1002/aic.12120
Published online November 23, 2009 in Wiley InterScience (www.interscience.wiley.com).

Several models for scheduling multipurpose batch plants exist in the literature. The models using unit-specific event points have shown better solution efficiency on various literature examples. This article presents a novel approach to scheduling multipurpose batch plants, which uses unit-slots instead of process-slots to manage shared resources such as material storage. We develop two slightly different models that are even more compact and simpler than that of Sundaramoorthy and Karimi, Chem Eng Sci. *2005;60:2679–2702. Although we focus on material as a shared resource, our multi-grid approach rationalizes, generalizes, and improves the current multi-grid approaches for scheduling with shared resources. Our models allow nonsimultaneous transfers of materials into and out of a batch. We show by an example that this flexibility can give better schedules than those from existing models in some cases. Furthermore, our approach uses fewer slots (event-points) on some examples than even those required by the most recent unit-specific event-based model. Numerical evaluation using literature examples shows significant gains in solution efficiency from the use of unit-slots except where the number of unit-slots required for the optimal solution equals that of process slots. We also highlight the importance of constraint sequencing in GAMS implementation for evaluating mixed-integer linear programming based scheduling models fairly.* © 2009 American Institute of Chemical Engineers *AIChE J,* 56: 1859–1879, 2010
Keywords: scheduling, multipurpose plants, batch processes, slot-based formulations, unit-slots, continuous-time

Introduction

The flexibility and versatility of batch plants in general and multipurpose batch plants (MBPs) in particular provide both opportunities and challenges for the manufacturer. MBPs use a pool of equipment and resources to produce a slate of products with varying recipes and characteristics. As the pool can be configured in a myriad of combinations and equipment and limited resources are shared among multiple products, scheduling the operation of MBPs is quite challenging and has received considerable attention in the literature. Méndez et al.,[1] Floudas and Lin,[2] and Pitty and Karimi[3] present excellent reviews of the current approaches and associated challenges. The substantial research effort over the past three decades has resulted in numerous mixed-integer linear programming (MILP) models. However, recent work[4–6] reveals that further work is needed to develop more general and efficient models.

The allocation of equipment and resources and sequencing of various tasks over time are the main considerations in most scheduling problems. Thus, an effective approach for modeling time (or time representation) is of utmost importance in a MILP model. The existing literature[1–3] has classified these approaches as slot-based, event-based, and precedence-based (or sequence-based).

The sequence-based or precedence-based representation[3] uses either direct[7–11] (immediate precedence) or indirect[12–15] (general

Correspondence concerning this article should be addressed to I. A. Karimi at cheiak@nus.edu.sg.

precedence) sequencing of pairs of tasks on units. Although it does not use constructs like "slots" or "event points" in time explicitly, it does assign times for various tasks and must prepostulate the possible numbers of batches. The handling of shared resources is not straightforward with these models.

Wagner[16] defined "order-positions" in a task-sequence for a machine-scheduling problem. He assigned one task to each position and one position for each task. Later, Ku et al.[17], Ku and Karimi[18,19] and Birewar and Grossmann[20] used "positions in a sequence" or "production slots" for scheduling multiproduct batch plants. These "positions" are essentially slots, and these models are slot-based. Their slots are in a sequence (or sequence-slots) rather than time (time-slots). These models restricted each task to one sequence-slot only, and did not use any explicit time-grid. However, they assigned timing variables such as completion times to these sequence-slots, which mapped time-slots on the time-grids of various units.

Apart from the aforementioned models, all slot-based scheduling models that we are aware of define slots as time intervals on a time-grid, or time-slots. Geoffrion and Graves[21] divided the scheduling horizon into "equal indivisible known time slots" (uniform discrete-time representation). In contrast, Sahinidis and Grossmann[22] used nonidentical, ordered time-slots of unknown variable lengths in their continuous-time formulation for planning continuous processes. More significantly, they may be the first to allow a task to span multiple consecutive time-slots, thus generalizing the assumption (one slot per task) that previous models[16–20] with sequence-slots used. Since then, several formulations[23–36] have used slots as ordered time intervals of unknown and variable lengths rather than sequence positions. Although some[23–27] restricted each task to a single slot, others[22,28–36] allowed it to span multiple slots. In our view, all are slot-based models with the former being just a special case of the latter. This is indeed the traditional view on slot-based models since the work of Sahinidis and Grossman in 1991.[22]

The scheduling literature has used two types of time slots. One is the synchronous[33,34,36] or process-slots,[26] and the other is asynchronous[25] or unit-slots.[26] In the former, all units in a process share one common set of slots. This provides a single time-axis along which shared resources such as storage, utilities, manpower, etc., can be balanced with ease. In the latter, each unit has a separate (or unit-specific) set of slots with partially or wholly independent timings. As the slot timings independently with units and are unknown in a scheduling formulation, the "order" of any resource usage during the slots is not readily known on a single time-axis. This makes the resource balance difficult.

Zentner et al.[37] used terms such as "events" and "event times" for the starts/ends of tasks in their comparison of uniform discrete-time models and nonuniform continuous-time models. Zhang and Sargent[38,39] used the same concept of "events" in their continuous-time MINLP formulation for an operational planning problem. These "events" can be viewed[2] as "starts/ends" of slots. Ierapetritou and Floudas[40] introduced the concept of "event points" in their continuous-time model. Floudas and Lin[2] classified these into global and unit-specific event points. In the former,[41,42] each event point has a unique value of time, which is the same for all units. This orders the event points on a single time-axis. This is the same[41] as synchronous or process-slots because the interval between successive event points is nothing but a process-slot. In unit-specific event points, an event point[40] is associated with multiple time instances, one for each unit (or unit-specific). Tasks corresponding to the same event point start at different times on different units. This is the same as unit-slots because the interval between the timings of two successive event points on a unit is a unit-slot. Thus, both event-based and slot-based approaches are conceptually the same, as they use time grids. While the former views grid/s in terms of ordered, distributed time points, the latter does the same in terms of ordered, variable-length time intervals.

Given the conceptual resemblance between the slot-based and event-based models, the notion of time-grid[43,44] seems clearer for classifying various scheduling approaches. Using that notion, we suggest three types of models (Figure 1), namely single-grid, multi-grid, and no-grid. Task timings are must to define a schedule, so all approaches must use timing variables (in different forms) for various activities. However, the sequence-based approach does not use a "grid" defined in terms of either "event points" or "slots". Thus, no-grid does not mean that timing variables do not exist. It just signifies that the timing variables are not associated with a grid. In that sense, one could argue it to be a no-grid approach. The models using global events or process-slots use one common time-grid for all units and resources. Thus, they are single-grid models. The models using unit-specific events or unit-slots use a separate time-grid for one or more units or resources. Thus, they are multi-grid models.[44]

Although the aforementioned classification presents our viewpoint, several different interpretations are indeed possible. In addition, some remarks are in order. Although the single-grid models do not differ significantly in modeling details, some multi-grid models do, especially in terms of variables and constraints. In this regard, we wish to highlight some differences between two groups of multi-grid models. One group has used the so-called unit-slots[24–26,35] and the other has used the so-called unit-specific events.[4,5,45,46] First, only a few models from the former have addressed shared resources. Karimi and McDonald[30] used a mix of unit-slots and nonuniform periods (process-slots), and restricted inventory balances across periods only. Lim and Karimi[25] and Erdirik-Dogan and Grossmann[35] used unit-slots, but both defined binary variables to relate the timings of resource usage across various units. In contrast, the unit-specific event-based models have handled shared resources without using any such binary variables. Second, most unit-specific event-based models define task timings as 3-index (task i, unit j, event point n) variables with unit-index being explicit[47] or implicit.[5,40,45,46] In contrast, the models using unit-slots invariably define timings as 2-index (unit j, slot k) variables with no index for task. This difference makes the model details (variables and constraints) and characters different for these two groups of models. Because tasks generally outnumber units, the event-based models need more event-time variables ($i \times j \times k$ vs. $j \times k$) and associated timing constraints.

In this work, we address the scheduling of MBPs using a multi-grid approach (using unit-slots). Several formulations of all three types exist in the literature for scheduling MBPs. While Ierapetritou and Floudas[40] used multiple grids (unit-specific event-points), Maravelias and Grossmann[41] and Sundaramoorthy and Karimi[34] used a single time-grid in their models. Although the last two models have striking similarities, there are significant modelling differences as well, which

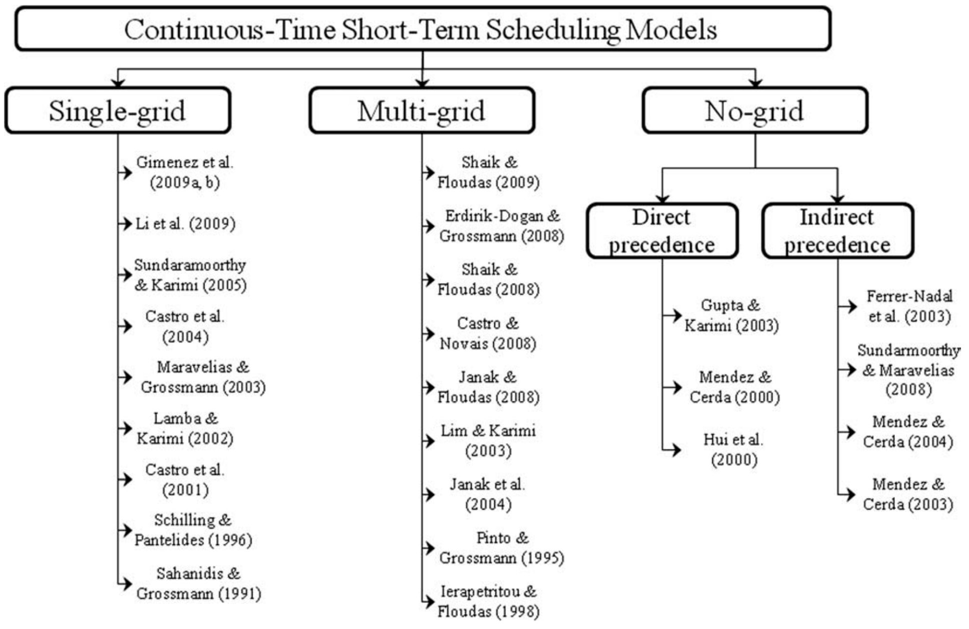

Figure 1. A classification of continuous-time scheduling models.

Sundaramoorthy and Karimi[34] have discussed in detail. Janak et al.[45] proposed a MILP formulation that improved over the previous versions of the unit-specific event-based models. They defined additional binary variables for denoting task ends. They also introduced storage tasks to address various storage configurations and even allowed a task to span multiple event-points, which was a significant departure from the earlier unit-specific event-based models. However, this formulation required many constraints and variables, and results in excessive solutions times.[5] Shaik and Floudas[46] proposed an improved version of the model of Ierapetritou and Floudas[40] using a RTN framework. Recently, Shaik and Floudas[5] have improved the model of Janak et al.[45] using a 3-index binary variable and without using storage tasks. Their model uses a user-defined parameter to allow tasks to span a given number of event-points. However, it requires one to iterate on that parameter for every possible number of event-points to reach an optimal solution. Mendez and Cerda[13] and Ferrer-Nadal et al.[15] have proposed sequence-based (no-grid) models based on indirect-pairing (general-precedence) of tasks. Ferrer-Nadal et al.[15] addressed nonzero transfer times from processing to storage units and vice-versa. While these two models do not need to prepostulate the numbers of event points or slots, as mentioned earlier, they assume the number of batches for each task.

Our specific goal in this article is to modify the earlier model of Sundaramoorthy and Karimi[34] to use unit-slots instead of process-slots. In addition, we allow various types of storage configurations[3,26] and wait policies for material states. Although we consider only "material" as a shared resource and "inventory balance" as a resource balance in this article, we use "resource" and "resource balance" as generic terms in our discussion, because we believe that our approach is seamlessly extensible to other resources such as tools, instruments, parts, utilities, manpower, etc.

Given our discussion on the conceptual similarity between the models using unit-specific events and unit-slots, this article makes the following contributions. It presents a sound and systematic approach for handling shared resources in multi-grid models. It rationalizes and improves the approach that most unit-specific event-based models have used in the literature. Our approach can also be viewed as an extension and generalization of what Castro and Novais[43] did for addressing material transfers between processing units and inter-stage unlimited-capacity storage units in multi-stage batch plants with parallel units. Lastly and more importantly, it shows the limitation of the existing and presented multi-grid approaches for addressing shared resources.

Problem Statement

A MBP has J batch processing units ($j = 1, 2, \dots, J$), performs I tasks ($i = 1, 2, \dots, I$) involving (production or consumption) S material states ($s = 1, 2, \dots, S$), and uses a dedicated storage s for each material state s. We describe the operation of MBP through the recipes[34] of various products, where a material state is any material (raw material, intermediate, waste, or final product) with distinct attributes and properties. To describe the multipurpose and specialty nature of units, we define $\mathbf{I}_j = \{i \mid \text{unit } j \text{ can process task } i\}$.

The MBP performs tasks in terms of individual batches. For each batch of task i on unit j in a MBP, we associate a batch size B_{ij}. We assume that this batch size requires a processing time of $\alpha_{ij} + \beta_{ij}B_{ij}$. Each batch will consume some material states in known proportions and produce other material states in some known proportions. We define a mass ratio (σ_{sij}) to quantify the actual amount of each material state s that a batch of task i on unit j may consume or produce. This is defined as follows.

$$\sigma_{sij} = \pm \frac{\text{Actual (not net) mass of material state } s \text{ consumed/produced by task } i \text{ on unit } j}{\text{Batch size (mass) of task } i \text{ on unit } j}$$

If task i on unit j consumes material state s, then $\sigma_{sij} < 0$; if it produces material state s, then $\sigma_{sij} > 0$; otherwise $\sigma_{sij} = 0$. Thus, the actual mass of material s associated with a batch size of B_{ij} is $|\sigma_{sij}|B_{ij}$. Note that the aforementioned mass ratio is unit-dependent. One may not need the mass ratio for a material state (e.g., a waste) in our model, if there is no need to monitor the inventory of such a material state. However, in several industrial scenarios, production schedules are constrained by the limited storage space for wastes or waste treatment capacity. Then, mass ratios will be needed for such waste materials in our formulation.

The storage of material states may involve various storage capacities and wait policies.[3,26] These are unlimited intermediate storage (UIS), limited intermediate storage (LIS), no intermediate storage (NIS), unlimited wait (UW), limited wait (LW), and zero wait (ZW). Each task on a processing unit begins (ends) with the transfers of input (output) materials into (out of) that unit from (to) appropriate storage facilities.

With this, the scheduling problem addressed in this article can be stated as:

Given:

1. Information on recipes, material states, tasks, mass ratios, etc.
2. J processing units, their suitable tasks, and limits on their batch sizes.
3. S storage units, initial inventories, and limits on their holdups.
4. Wait policy for each material state s.
5. Cost of or net revenue from each material state.

Determine:

1. Tasks and their sequence and timings on each unit.
2. Batch size of each task.
3. Inventory profiles of all material states.

Assuming:

1. Deterministic scenario with no batch/unit failures or operational. interruptions.
2. Unit-to-unit transfers are instantaneous.
3. Setup or changeover times (if any) are lumped into batch processing times.
4. Batch processing time varies linearly with batch size.
5. All processing units can hold a batch temporarily before its start and after its end.
6. Direct unit-to-unit transfer of a material while bypassing the storage is allowed.
7. The storage of material states are the only shared resources.
8. Transfers of input materials for a batch may follow any sequence.

Allowing:

1. Transfers of input (output) materials into (out of) a unit for any batch need not be simultaneous.

Aiming for:

1. Maximum revenue from the plant for a given scheduling horizon $[0, H]$, or
2. Minimum time (makespan) to produce specified demands (d_s, $s = 1, 2, \ldots, S$) of material states

Unless otherwise indicated, an index takes all its legitimate values in all the expressions or constraints in our formulation.

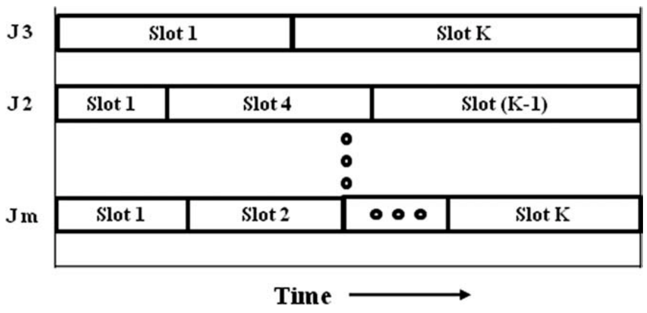

Figure 2. Design of unit-slots for our novel formulations.

MILP Formulation

To model the schedule of activities on each unit j and the storage of each material state s during the scheduling horizon, we define K ($k = 1, 2, \ldots, K$) contiguous slots (Figure 2) of unknown and arbitrary lengths. Let T_{jk} [$k = 0, 1, 2, \ldots, K$; $T_{j0} \geq 0$; $T_{jK} \leq H$; $T_{jk} \geq T_{j(k-1)}$, $1 \leq k \leq K$] denote the end time of slot k on unit j. The time before slot 1 starts is slot 0 ($k = 0$). Thus, a slot k on unit j starts at $T_{j(k-1)}$, ends at T_{jk}, and has a length $[T_{jk} - T_{j(k-1)}]$. We use T_{sk} to denote the end times of slots on storage units. While each unit/storage has K slots, the slot end times (T_{jk}/T_{sk}) and thus slot lengths vary from unit to unit. By definition,

$$T_{j(k+1)} \geq T_{jk} \quad 1 \leq j \leq J, 0 \leq k < K \tag{1a}$$

$$T_{s(k+1)} \geq T_{sk} \quad 1 \leq s \leq S, 0 \leq k < K \tag{1b}$$

Tasks and batches

We allocate the processing tasks to various slots on the processing units. Each task involves one batch, every slot must have a task, and the allocation of a task may span multiple slots. A batch of any task during $[T_{jk}, T_{j(k+1)}]$ of a slot $(k + 1)$ on unit j involves three operations in the following order.

1. Unit j idles or receives input materials for the current batch during $[T_{jk}, T_{jk} + \delta_{j(k+1)}]$, where $\delta_{jk} \geq 0$ ($1 \leq k \leq K$) is an unknown continuous variable, which also denotes the delay in the actual start of a task in slot $k + 1$ of unit j. It may receive a material any time during this interval. Each material transfer into the unit is instantaneous, but the transfers need not be simultaneous and may follow any sequence. All the transfers required for the current batch end by $T_{jk} + \delta_{j(k+1)}$. If the unit is continuing a batch from the previous slot (k), then no idling or transfers can occur and $\delta_{j(k+1)} = 0$.

2. The unit processes (reacts, crystallizes, heats, etc.) the current batch. It begins the processing at $T_{jk} + \delta_{j(k+1)}$ and ends at $T_{j(k+1)} - \theta_{j(k+1)}$, where $\theta_{jk} \geq 0$ ($1 \leq k \leq K$) is another unknown continuous variable, which also denotes the early end of a task in slot k of unit j, such that $T_{j(k+1)} - \theta_{j(k+1)} \geq T_{jk} + \delta_{j(k+1)}$. This gives us the following that replaces Eq. 1b.

$$T_{j(k+1)} \geq T_{jk} + \delta_{j(k+1)} + \theta_{j(k+1)} \quad 1 \leq j \leq J, 0 \leq k < K \tag{1c}$$

3. The unit idles or discharges the outputs from the current batch during $[T_{j(k+1)} - \theta_{j(k+1)}, T_{j(k+1)}]$. It may discharge an output material any time during this interval. The material transfers out of the unit are instantaneous, but need not be simultaneous.[48–50] All the required transfers end by $T_{j(k+1)}$. If the unit is continuing a batch into the next slot $(k + 2)$, then no idling or transfers can occur and $\theta_{j(k+1)} = 0$.

Next, we define one binary variable (ys_{ijk}) and two 0–1 continuous variables (ye_{ijk} and y_{ijk}) to denote respectively the start, end, and continuation of the allocation of a task (including the idling of a unit, which we define as task $i = 0$) on a unit-slot as follows,

$$ys_{ijk} = \begin{cases} 1 & \text{if task } i \text{ begins its allocation on unit } j \text{ in slot } (k+1) \\ 0 & \text{Otherwise} \end{cases}$$
$$1 \le j \le J, i = 0, i \in \mathbf{I}_j, 0 \le k < K$$

$$ye_{ijk} = \begin{cases} 1 & \text{if task } i \text{ ends its allocation on unit } j \text{ in slot } k \\ 0 & \text{Otherwise} \end{cases}$$
$$1 \le j \le J, i = 0, i \in \mathbf{I}_j, 1 \le k \le K$$

$$y_{ijk} = \begin{cases} 1 & \text{if task } i \text{ continues its allocation on unit } j \text{ from slot } k \text{ to } (k+1) \\ 0 & \text{Otherwise} \end{cases}$$
$$1 \le j \le J, i = 0, i \in \mathbf{I}_j, 0 \le k < K$$

ys_{ijk} refers to the start of a new allocation of task i from T_{jk}, y_{ijk} refers to the continuation of a current allocation of task i across T_{jk}, and ye_{ijk} refers to the end of a current allocation of task i at T_{jk}. If a unit j begins a new task in slot $(k + 1)$, then a new batch necessarily begins at T_{jk}. However, it is also possible that a unit ends a batch in slot k and continues with a new batch of the same task again in slot $(k + 1)$. While y_{ijK} is undefined, y_{ij0} is known and fixed. If a task i is unfinished at time zero, and must continue, then $y_{ij0} = 1$, otherwise $y_{ij0} = 0$. Thus, we allow an unfinished batch at time zero to continue. However, we do not allow any unfinished batch at the end of scheduling horizon (H). In other words, all batches must end within the scheduling horizon. If a unit j has just ended a batch of i at time zero, then we set $ye_{ij0} = 1$ and all other ye_{ij0} as zero.

One of three things must happen at every T_{jk}. A task allocation may begin, a task allocation must continue, or the unit must idle. This gives us,

$$\sum_{i=0, i\in\mathbf{I}_j} (y_{ijk} + ys_{ijk}) = 1 \qquad 1 \le j \le J, 0 \le k < K \tag{2}$$

Similarly, the allocation of a task i ends at T_{jk}, if and only if it starts/continues at $T_{j(k-1)}$, and does not continue across T_{jk}. That is,

$$ye_{ijk} = [y_{ij(k-1)} + ys_{ij(k-1)}] - y_{ijk} \qquad 1 \le j \le J, i = 0, i \in \mathbf{I}_j, 1 \le k < K \tag{3a}$$

$$ye_{ijK} = y_{ij(K-1)} + ys_{ij(K-1)} \qquad 1 \le j \le J, i = 0, i \in \mathbf{I}_j \tag{3b}$$

Batch sizes

Let $BI_{ijk} = B^{\mathrm{L}}_{ij} ys_{ijk} + \Delta BI_{ijk}$ $(1 \le j \le J,\ i \in \mathbf{I}_j,\ 0 \le k < K)$, BO_{ijk} $(1 \le j \le J,\ i \in \mathbf{I}_j,\ 1 \le k \le K)$, and b_{ijk} $(1 \le j \le J,\ i \in \mathbf{I}_j,\ 0 \le k < K)$ respectively be the amounts of task i entering, exiting, and continuing at unit j at T_{jk} or the start of slot $(k + 1)$. Here, B^{L}_{ij} is the minimum required amount of task i on unit j. If a unit j is empty at the start of the horizon, then we set $b_{ij0} = 0$, otherwise we assign an appropriate nonzero value. We require that all units be empty ($b_{ijK} = 0$) at the end of the horizon.

First, the finite capacity of unit j demands that the amounts of task i entering, continuing, and exiting at T_{jk} not exceed the maximum allowable batch size (B^{U}_{ij} of task i on unit j. Therefore, we have,

$$\Delta BI_{ijk} \le (B^{\mathrm{U}}_{ij} - B^{\mathrm{L}}_{ij}) ys_{ijk} \qquad 1 \le j \le J, i \in \mathbf{I}_j, 0 \le k < K \tag{4a}$$

$$b_{ijk} \le B^{\mathrm{U}}_{ij} y_{ijk} \qquad 1 \le j \le J, i \in \mathbf{I}_j, 1 \le k < K \tag{4b}$$

$$BO_{ijk} \le B^{\mathrm{U}}_{ij} ye_{ijk} \qquad 1 \le j \le J, i \in \mathbf{I}_j, 1 \le k \le K \tag{4c}$$

Lastly, a balance on the amount of task i over slot k gives us,

$$BO_{ijk} = b_{ij(k-1)} + [B^{\mathrm{L}}_{ij} ys_{ij(k-1)} + \Delta BI_{ij(k-1)}] - b_{ijk} \qquad 1 \le j \le J, i \in \mathbf{I}_j, 1 \le k \le K \tag{5}$$

Operation times

As described earlier, if the allocation of a task i continues across T_{jk}, then θ_{jk} for slot k and $\delta_{j(k+1)}$ for slot $(k + 1)$ must be zero. Thus,

$$\delta_{j(k+1)} \le H \sum_{i\in\mathbf{I}_j} ys_{ijk} \qquad 1 \le j \le J, 0 \le k < K \tag{6a}$$

$$\theta_{jk} \le H \sum_{i\in\mathbf{I}_j} ye_{ijk} \qquad 1 \le j \le J, 1 \le k \le K \tag{6b}$$

Next, let t_{jk} $(1 \le j \le J,\ 0 \le k \le K)$ denote the processing time remaining in completing the ongoing batch on unit j at T_{jk}. If a batch has ended at or before time zero, then we set $t_{j0} = 0$, otherwise we assign an appropriate nonzero value. If a batch ends during slot k, then the remaining batch time must be zero at T_{jk}.

$$t_{jk} \le \sum_{i\in\mathbf{I}_j} (\alpha_{ij} y_{ijk} + \beta_{ij} b_{ijk}) \qquad 1 \le j \le J, 1 \le k < K \tag{7a}$$

Using t_{jk}, we can compute the actual processing time (non-negative) of a batch during slot k as $t_{jk} + \sum_{i\in\mathbf{I}_j} [(\alpha_{ij} + \beta_{ij} B^{\mathrm{L}}_{ij}) ys_{ijk} + \beta_{ij} \Delta BI_{ijk}] - t_{j(k+1)}$, where α_{ij} and β_{ij} $(1 \le j \le J,\ i \in \mathbf{I}_j)$ are the parameters defining the linear dependence of batch processing time on batch size. Then, summing all the operation times, we obtain,

$$T_{j(k+1)} = T_{jk} + \delta_{j(k+1)} + t_{jk} + \sum_{i\in\mathbf{I}_j} [(\alpha_{ij} + \beta_{ij} B^{\mathrm{L}}_{ij}) ys_{ijk} + \beta_{ij} \Delta BI_{ijk}] - t_{j(k+1)} + \theta_{j(k+1)} \qquad 1 \le j \le J, 0 \le k < K \tag{7b}$$

Material transfers and inventory balance

When a batch begins, it must use some materials from the storage tanks, and when it ends, it must transfer some materials to them. To this end, consider a unit j receiving or delivering a material state s in slot k. Suppose that this material flow occurs in slot k' on storage unit s. Three scenarios are possible: $k' < k$, $k' = k$, and $k' > k$. For $k' < k$, we can simply introduce additional slots on storage s to make $k' = k$. For $k' > k$, we can do the same on unit j. In other words, with no loss of generality, we demand that if a unit j is receiving or delivering a material to a storage s at any time, then the unit-slots corresponding to that time on both unit j and storage s must have the same index. This is a key step that avoids the binary variables defining the relative positions of checkpoints on different units as used by Lim and Karimi.[25]

Now, consider the start of a new allocation of a batch of task i ($i \in \mathbf{I}_j$) in slot $(k + 1)$ of unit j. This batch will need materials from storage tanks s with $\sigma_{sij} < 0$, hence consider such a storage s transferring material s to unit j during slot $(k + 1)$ of unit j. As discussed earlier, we demand that this transfer must occur during $[T_{jk}, T_{jk} + \delta_{j(k+1)}]$. As argued in the previous paragraph, storage s must make this transfer during its own slot $(k + 1)$ or at time T_{sk} with no loss of generality. Since the transfer must occur between $[T_{jk}, T_{jk} + \delta_{j(k+1)}]$, we write,

$$T_{sk} \geq T_{jk} - H[1 - \sum_{i \in \mathbf{I}_j, \sigma_{sij} < 0} ys_{ijk}]$$
$$1 \leq j \leq J, s: \sum_{i \in \mathbf{I}_j, \sigma_{sij} < 0} \sigma_{sij} < 0, s: I_s^{\mathrm{U}} \text{ is limited}, 0 \leq k < K \tag{8a}$$

$$T_{sk} \leq T_{jk} + \delta_{j(k+1)} + H[1 - \sum_{i \in \mathbf{I}_j, \sigma_{sij} < 0} ys_{ijk}]$$
$$1 \leq j \leq J, s: \sum_{i \in \mathbf{I}_j, \sigma_{sij} < 0} \sigma_{sij} < 0, s: I_s^{\mathrm{U}} \text{ is limited}, 0 \leq k < K \tag{8b}$$

Note that we do not use Eqs. 8a and 8b for j and s, if j can never perform a task that consumes s, or s has an unlimited supply (e.g., raw materials) and can never experience a stock-out. $s: \sum_{i \in \mathbf{I}_j, \sigma_{sij} < 0} \sigma_{sij} < 0$ ensures the former. The summation term in Eq. 8 is to ensure that Eq. 8 holds any time a task that consumes s on unit j begins at T_{jk}.

We now use a similar argument to address the transfer of material at the end of a batch on unit j. Consider the end of a batch of task i ($i \in \mathbf{I}_j$) during slot k on unit j. Each storage s with $\sigma_{sij} > 0$ will receive some material from this batch. As done before, we demand that this transfer must occur during $[T_{j(k+1)} - \theta_{j(k+1)}, T_{j(k+1)}]$ on unit j and slot k or at time T_{sk} on storage s. Therefore, we get,

$$T_{sk} \leq T_{jk} + H[1 - \sum_{i \in \mathbf{I}_j, \sigma_{sij} > 0} ye_{ijk}]$$
$$1 \leq j \leq J, s: \sum_{i \in \mathbf{I}_j, \sigma_{sij} < 0} \sigma_{sij} > 0, 1 \leq k < K, s: I_s^{\mathrm{U}} \text{ is limited} \tag{9a}$$

$$T_{sk} \geq T_{jk} - \theta_{jk} - H\left[1 - \sum_{i \in \mathbf{I}_j, \sigma_{sij} > 0} ye_{ijk}\right]$$
$$1 \leq j \leq J, s: \sum_{i \in \mathbf{I}_j, \sigma_{sij} > 0} \sigma_{sij} > 0, 1 \leq k < K, s: I_s^{\mathrm{U}} \text{ is limited} \tag{9b}$$

Again, we do not use Eqs. 9a and 9b for j and s, if j can never perform a task that produces s, or storage s is a final product or a waste with unlimited storage capacity, and can never face an overflow. $s: \sum_{i \in \mathbf{I}_j, \sigma_{sij} > 0} \sigma_{sij} > 0$ ensures the former. The summation term in Eq. 9 is due to the same reasons explained for Eq. 8.

Equation 9b assumes UW (unlimited wait) policy. To accommodate other wait policies such as LW (limited wait) and ZW (zero wait) for intermediate material states, we need the following.

$$T_{sk} \leq T_{jk} - \theta_{jk} + \sum_{i \in \mathbf{I}_j, \sigma_{sij} > 0} [w_{ij} ye_{ijk}] - H[1 - \sum_{i \in \mathbf{I}_j, \sigma_{sij} > 0} ye_{ijk}]$$
$$1 \leq j \leq J, s: \sum_{i \in \mathbf{I}_j, \sigma_{sij} > 0} \sigma_{sij} > 0, \; 1 \leq k < K, \; s: I_s^{\mathrm{U}} \text{ is LIS/LW} \tag{9c}$$

where, w_{ij} is the maximum time that unit j can hold a batch of task i after its completion of processing.

On the basis of our aforementioned discussion and Eqs. 7–9, we have the following inventory balance for storage s.

$$I_{s1} = I_{s0} + \sum_j \sum_{i \in \mathbf{I}_s, \sigma_{sij} < 0} \sigma_{sij} \left[B_{ij}^{\mathrm{L}} ys_{ij1} + \Delta BI_{ij1}\right] \quad 1 \leq s \leq S \tag{10a}$$

$$I_{sk} = I_{s(k-1)} + \sum_j \sum_{i \in \mathbf{I}_s, \sigma_{sij} > 0} \sigma_{sij} BO_{ijk}$$
$$+ \sum_j \sum_{i \in \mathbf{I}_s, \sigma_{sij} < 0} \sigma_{sij} [B_{ij}^{\mathrm{L}} ys_{ijk} + \Delta BI_{ijk}] \quad 1 \leq s \leq S, 1 \leq k < K \tag{10b}$$

$$I_{sK} = I_{s(K-1)} + \sum_j \sum_{i \in \mathbf{I}_s, \sigma_{sij} > 0} \sigma_{sij} BO_{ijK} \quad 1 \leq s \leq S \tag{10c}$$

where, I_{sk} ($I_s^{\mathrm{L}} \leq I_{sk} \leq I_s^{\mathrm{U}}$) is the inventory of material s at T_{sk} or the end of slot k on storage s.

Note that temporary storage in processing units, instantaneous material transfers, and storage unit bypassing are key implicit assumptions in Eqs. 7–10. Equation 10 applies to the various storage capacities (LIS, UIS, and NIS) by setting I_s^{U} properly. However, we do not need Eqs. 8a and 9a for the case of UIS.

Equations 8 and 9 are the key constraints that register material transfers chronologically on storage tanks. They do not exist in a formulation that uses process-slots. While process-slots are synchronized across all units irrespective of their relation to a material state, unit-slots are synchronized across only those units that are associated with a material state. The unit-slots in our formulation will be identical to process-slots in the worst-case scenario. In such a case, a formulation using process-slots will perform better (in terms of both computational time and relaxed MILP objective) than the one using unit-slots, because of the absence of Eqs. 8 and 9.

We now argue that Eqs. 8 and 9 guarantee a valid resource (specifically mass in this case) balance. A pivotal basis underlying this guarantee is our model's ability to allow tasks to span multiple slots. As discussed earlier, this allows us insert dummy slots freely as needed on any unit. This then allows us to assume with no loss of generality that if a unit j receives or delivers a material to a storage s at any time, then the unit-slots corresponding to that transfer must have the same index on both unit j and storage s. Now, Eqs. 7 and 8 register exactly every resource exchange between a processing unit (j) and a resource on the time-grid of that resource (s). The timings of these exchanges are the endpoints (T_{sk}) of the slots on the time-grid of resource s, which are in the correct chronological order because of Eq. 1. No resource exchange occurs at points other than T_{sk}. This proves that our resource balance is valid and correct.

Furthermore, as the various operations (transfer, processing, etc.) on each unit are allocated to their appropriate slots, and sequenced properly, the overall schedule will not have any infeasibility. Note that our model does not use any example-specific arguments to derive the timing constraints. Hence, it is as general as any model that uses global event points or process slots, as far as mass as a resource is concerned. However, in contrast, most models in the literature assume simultaneous (but instantaneous) transfers of materials. As our model allows nonsimultaneous (but instantaneous) transfers of materials, it is in fact more general than the models with global event points or process slots.[34,41,42] Indeed, we show later with an example that our model can give better optimal schedules than literature models.[34,41]

Castro et al.[48] and Gimenez et al.[49,50] have also addressed the non-simultaneous transfers of materials. Castro et al.[48] defined explicit transfer tasks and assigned each material to a single transfer. In addition, they did not allow the temporary storage before the actual start of a task. The single-grid models of Gimenez et al.[49,50] and our multi-grid model do not have such limitations. δ_{jk} and θ_{jk} in our model are very similar to the slack variables ($\tilde{T}^{\mathrm{LB}}_{j,n}$ and $\tilde{T}^{\mathrm{EE}}_{j,n}$) of Gimenez et al.[49,50] However, Gimenez et al.[49,50] use several extra binary variables such as S^{I}_{jn} and S^{O}_{jn} to model the activity states of processing units, which our approach does not. Similarly, they use a binary variable Y_{ijn} to denote if a task i formally ends in unit j at T_n, which we do not. Therefore, our model uses much fewer binary variables. It detects activity states of the processing unit in an implicit manner, and is much simpler. However, Gimenez et al.[49,50] also consider preventive maintenance, changeover or set-up times, and intermediate due-dates, which we do not in this work.

As mentioned earlier, the aforementioned approach for material transfers and inventory balances may be viewed as an extension of the one used by Castro and Novais[43] for scheduling multi-stage batch plants. However, a very critical distinction between the two is that Castro and Novais[43] do not allow a task to span multiple event-points, and arbitrarily assume that each transfer between a processing and a storage unit occurs at the same event-point across two different time-grids. In contrast, we provide a rationale for the latter in light of the former. This to us is a prerequisite for the validity of resource balance in multi-grid formulations. Apart from that consideration, our model allows nonsimultaneous material transfers and applies to multi-purpose rather than multi-stage batch plants. The storage units are pooled across all units rather than being dedicated to the parallel units in the stages immediately upstream and downstream of each inter-stage storage. We consider all storage configurations (UIS, LIS, NIS) and wait policies (UW, LW, ZW) as compared to UIS/UW considered by Castro and Novais.[43] Finally, we use binary variables in Eqs. 8 and 9 to enable selective synchronization across storage and processing units.

In addition to the above, our approach can also be contrasted with the one used by Janak et al.[45] Janak et al.[45] defined an extra set of storage tasks and corresponding binary variables. In contrast, we do not have such binary variables. They also defined timing variables for the storage tasks. Their timing variables were essentially 3-index (task, implicit unit, event-point). In contrast, our timing variables are 2-index (storage unit, slot). They also used additional variables for the storage task amounts, and several extra constraints for capacity, duration, and sequencing to ensure correct material balance. In contrast, our approach is much simpler, more intuitive, and more general, and requires far fewer variables and constraints. We do not report our numerical evaluation of that model in this article, because it took excessive computation times on almost all problems. Our observation is consistent with what has been commented by Shaik and Floudas.[5]

Variable bounds and scheduling objectives

Appropriate bounds on the variables can improve solution time. All variables are nonnegative in our formulation, and the upper bounds for continuous variables are $\Delta BI_{ijk} \le B^{\mathrm{U}}_{ij} - B^{\mathrm{L}}_{ij}$, $BO_{ijk} \le B^{\mathrm{U}}_{ij}$, $b_{ijk} \le B^{\mathrm{U}}_{ij}$, $T_{jk} \le H$, $T_{sk} \le H$, and $I^{\mathrm{L}}_{s} \le I_{sk} \le I^{\mathrm{U}}_{s}$. Note that we do not impose an upper bound on t_{jk} as done by Sundaramoorthy and Karimi,[34] as it is bounded by Eq. 6a. Also, we do not impose an upper limit on the inventory of a material state that does not have a zero-wait policy at the end of the last slot K, as that inventory level will depend on what may happen after the last slot K.

Three scheduling objectives are possible. Two used in the literature are makespan and revenue. The third is net profit. The revenue from production is given by:

$$R = \sum_{k=1}^{K} \sum_{j=1}^{J} \sum_{i \in \mathbf{I}_j} \sum_{s,\sigma_{sij}>0} \sigma_{sij} v_s BO_{ijk} \tag{11}$$

where, υ_s is the price of material s.

In case of makespan minimization, H ceases to be a given parameter, and we need to satisfy a given demand (d_s) for each material s.

$$\sum_{k=1}^{K} \sum_{j=1}^{J} \sum_{i \in \mathbf{I}_j} \sum_{s,\sigma_{sij}>0} \sigma_{sij} BO_{ijk} \ge d_s \tag{12}$$

In this case, we use the following to compute makespan, even though only the first would be sufficient.

$$\mathrm{MS} \ge T_{jK} \tag{13a}$$

$$\mathrm{MS} \ge T_{sK} \tag{13b}$$

Equations 13a and 13b used together seem to give faster solutions.

The third objective of net profit is a generalization of revenue. It includes all materials rather than just those that are sold.

$$\mathrm{NP} = \sum_{k=1}^{K-1}\sum_{j=1}^{J}\sum_{i\in\mathbf{I}_j}\sum_{s:\sigma_{sij}<0}\sigma_{sij}v_s\left[B_{ij}^{\mathrm{L}}ys_{ijk}+\Delta BI_{ijk}\right] + \sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{i\in\mathbf{I}_j}\sum_{s:\sigma_{sij}>0}\sigma_{sij}v_sBO_{ijk} \quad (14)$$

This completes our first model (SLK1, Eqs. 1–9, 11 or 12–13 or 14) for scheduling MBP stated earlier. We now present a slight modification (SLK2) of SLK1.

Alternate model (SLK2)

In contrast to SLK1, we define an additional 0–1 continuous variable as follows:

$$z_{jk} = \begin{cases} 1 & \text{if unit } j \text{ ends a batch within slot } k \\ 0 & \text{Otherwise} \end{cases} \qquad 1 \le j \le J, 0 \le k < K$$

If a unit j is idle or has ended tasks at time zero, then $z_{j0} = 1$. Otherwise, $z_{j0} = 0$.

Clearly, ye_{ijk}, ys_{ijk} , and z_{jk} must satisfy,

$$z_{jk} = \sum_{i=0, i\in\mathbf{I}_j} ys_{ijk} \quad 1 \le j \le J,\ 0 \le k < K \quad (15a)$$

$$z_{jk} = \sum_{i=0, i\in\mathbf{I}_j} ye_{ijk} \quad 1 \le j \le J,\ 1 \le k < K \quad (15b)$$

For SLK2, we use the above in place of Eq. 2. Now, we rewrite Eqs. 4b,4c as follows:

$$\sum_{i\in\mathbf{I}_j} b_{ijk} \le B_j^{\mathrm{U}}[1 - z_{jk}] \qquad 1 \le j \le J,\ 1 \le k < K \quad (16a)$$

$$\sum_{i\in\mathbf{I}_j} BO_{ijk} \le B_j^{\mathrm{U}} z_{jk} \qquad 1 \le j \le J,\ 1 \le k \le K \quad (16b)$$

where, $B_j^{\mathrm{U}} = \max_{i\in\mathbf{I}_j}[B_{ij}^{\mathrm{U}}]$.

With this, SLK2 comprises Eqs. 1, 3, 4a, 16a, 16b, 5–9, and 11–15. Although SLK2 has more constraints than SLK1, it performs better, as we show later. Furthermore, this is in spite of the fact that Eqs. 4b and 4c are intuitively tighter than Eqs. 16a and 16b.

Numerical Evaluation

A fair and an unbiased comparison demands careful attention on many factors[51] such as hardware, operating system, and software. In our study, we used CPLEX 11/GAMS[52] 22.8 on a Dell precision PWS690 workstation with Intel® Xeon® 3 GHz CPU, 16 GB RAM, running Windows XP Professional x64 Edition. To solve our models for various examples, the first step is to program them in GAMS.

GAMS implementation

In GAMS, the MODEL statement defines a sequence of constraints in an optimization model. Most MILP users know very well a major pitfall that this step involves as far as the solution and comparison of MILP models are concerned. Although the sequence in the MODEL statement does not affect model statistics such as the numbers of constraints, variables, and non-zeros, the reality is that it has a profound effect on the model solution time. A change in the sequence of the constraints changes the solution time for the same model. This is no news to a researcher working with MILPs. However, more significantly, this fact enables one to manipulate the sequence of constraints in order to obtain a better solution time for a given model. Ironically, to our knowledge, this issue has not been reported or discussed yet in the process scheduling literature on MILP models. Our extensive numerical experience with a variety of scheduling models in general, and those in this work in particular, shows that it is highly unlikely for a single sequence of constraints to be the most efficient for all test examples. Not only this, it is difficult to guarantee that a given sequence will be the most efficient for all instances of a specific example. The issue in our opinion is similar to the observation of Liu and Karimi[25] that it is difficult to find a single MILP model that performs the best on all examples. In our experience, the solution times can vary by an order of magnitude with a change in the order of constraints. Table 1 lists some of the many observed results from our numerical work. Clearly, this issue is extremely critical in comparing MILP models and can easily give unsound results. However, it is not clear, if a satisfactory resolution of this issue is possible. Though it is certainly beyond the scope of this work, we strongly believe that this critical issue, in addition to those mentioned by Karimi et al.,[51] must be taken care of in any numerical evaluation of MILP models. In our numerical evaluation, we have eliminated the effect of this factor. While we are unaware of any theoretical or heuristic guidelines for determining the optimal or best possible order/sequence of constraints, we believe that every work must report the constraint sequences of all scheduling models that it tests. As a fair practice, in all MILP model comparisons, the literature models must be implemented with the same order of constraints as done in the articles that reported the models. If such sequences are not available, or the authors chose to use a different sequence, then the authors must report the appropriate sequences along with reasons for doing so. Later, we list the constraint sequences for all models tested in this work.

In addition to the above, several other factors have a significant impact on the computational performance of MILP formulations These are MIP solver, solver version,[34] solver tuning options (Table 2), example-specific fixing of variables and parameters (e.g. Big-M values), uneven fixing of variables across models, solution iterations in search of the best, etc. For instance, fixing of variables based on example-specific information exists in some work.[5,45] This can be an advantage for a model in which it is done, and disadvantage for the one in which it is not. A comparison, in which some variables are fixed in some models based on example-specific information, and not in others, can lead to unreliable assessment.

First, we solve a simple example to compare the recently published unit-specific event-based model[5] with our models (SLK-1 and SLK-2). This example highlights the need for allowing tasks to span all possible events or slots.

Table 1. **Effect of Constraint Sequence in GAMS Models on Solution Times**

Example	Objective	Demand (mu) /Horizon (h)	Model	CPU Time (s)	
				Seq-1	Seq-2
3	Max Revenue	$H = 12$ h	SLK-2	781	564
	Max Revenue	($H = 16$ h)	SK	95.41	68.64
4	Max Revenue	($H = 10$ h)	SK	57.84	45.01
5	Min MS	$d(s12) = 100$, $d(s13) = 200$ mu	SLK-2	249	701
	Min MS	$d(s12) = d(s13) = 250$ mu	SLK-2	9.7	15.6
	Min MS	$d(s12) = d(s13) = 250$ mu	SK	107	2458

Seq-1 for SLK2 and SK are the same as the sequences reported in the article.
Seq-2 for revenue maximization are:
Model SLK2 \10a-c, 1a-b, 9a-b, 8a-b, 11, 5, 4a, 16a, 16b, 7a-b, 15a-b, 3a-b\;
Model SK \2, 3, 9, 13, 5, 6, 10, 4, 11, 12, 14, 20\.
Seq-2 for makespan minimization are:
Model SLK2 \12, 15a-b, 3a-b, 5, 4a, 16a, 16b, 7a-b, 8b-a, 9b-a, 1a-b, 13a-b, 10a-c\;
Model SK \22, 2, 3, 9, 13, 5, 6, 10, 4, 11, 12, 14, 21\.

Example 1

This example consists of 4 tasks ($i1$–$i4$), 4 units ($j1$–$j4$), 6 material states ($s1$–$s6$), and 4 storages ($s2$–$s5$). Figure 3 shows the detailed recipe diagram, and Tables 3 and 4 list the complete data. We assume a scheduling horizon of 10 h and maximize revenue.

First, we use the model (SF) of Shaik and Floudas.[5] We begin with $n = 5$ (five event-points) and examine the effect of increasing Δn^5 from 0 to 3. For $\Delta n = 0$, SF gives an objective of 200.12. Increasing Δn to 1 ($\Delta n = 1$) gives the same objective, so we increase n to 6. Now, SF gives the same objective of 200.12 for $\Delta n = 0$ and $\Delta n = 1$. Any further increase up to $n = 15$ gives the same solution for $\Delta n = 0$ and $\Delta n = 1$, so we conclude 200.12 as the best solution from SF. With SLK-1 and SLK-2, we get a solution of 300 for $K = 6$ and 400 with $K = 7$. Figure 4 gives that solution. Increasing K up to 15 does not improve the objective, so we conclude this as the best solution. Clearly, SF is unable to give the best solution using this common approach of increasing slots or event-points by one. The reason is that it has an additional layer of iteration, namely Δn. Therefore, to study SF further, we tried additional values for Δn. With $n = 5$ and $\Delta n = 2$, SF gives a solution of 300. Increasing Δn to 3 also gives the same solution. For $n = 6$ and $\Delta n = 2$, SF again gives a solution of 300. But with $n = 6$ and $\Delta n = 3$, SF gives the desired solution of 400. Thus, it is not clear how one should limit the number of event-points for tasks in SF, as it is not possible to know how many event-points a task may span in any given example. It is also clear that SF requires cascaded iterations as compared to our models.

Now, we present another example to show that SLKs can give better optimal schedules than those from the unit-specific event-based model of Shaik and Floudas (SF[5]), and the single-grid models of Maravelias and Grossmann (MG[41]) and Sundaramoorthy and Karimi (SK[34]). This is simply because SLKs allow non-simultaneous material transfers, while others do not. Note that MG is indeed a reduced version of the model of Maravelias and Grossmann (MG[41]) by eliminating the constraints related to utilities/resources.

Example 2

This example involves six tasks ($i1$–$i6$), 3 units ($j1$–$j3$), eight material states ($s1$–$s8$), three storages ($s4$–$s6$), and two final products (s7 and s8) through two production routes (R1 and R2). Figure 5 shows the recipe diagram with different arcs for R1 (solid) and R2 (dotted). Tables 3 and 4 list the complete data. We maximize revenue for a scheduling horizon of 6 h.

MG, SK, and SF give an optimal solution of \$560.1 for this example. In contrast, SLKs give a better optimal solution of \$650. This is because the former do not allow nonsimultaneous material transfers. For MG, SK, and SF assume that all materials required for each batch must be transferred simultaneously at a single point in time. Tasks $i1$ and $i2$ on $j1$ and $j2$ produce $s4$ and $s5$, respectively. $i4$ needs $s4$ and $s5$ in $j3$ to produce $s7$ (a final product). Although $i1$ produces 30 kg of $s4$ in $j1$ at 4 h, $i2$ produces 10 kg of $s5$ in $j2$ at 3 h. As MG, SK, and SF require that $s4$ and $s5$ must be transferred simultaneously to start $i4$ in $j3$, $s5$ has to wait 1 h (from 3 h to 4 h), before it is used for $i4$. Thus, $j2$ should either hold $s5$ during [3 h, 4 h], or transfer it to the storage at 3 h. As S5 has a maximum capacity of 5 kg, it cannot store the 10 kg of $s5$, and $j2$ must hold $s5$ during [3 h, 4 h]. This forces $j2$ to be idle from 3 h to 4 h. SLKs on the other hand

Table 2. **Effect of Solver Options in GAMS Models on Solution Times**

Example	Objective	Demand (mu)/Horizon (h)	Model	CPU Time (s) for CPLEX Options			
				Default	Scaind = 1	Reduce = 2	Dpriind = 2
3	Max Revenue	$H = 12$	SLK-2	781	603	217	588
	Max Revenue	$H = 16$	SK	377	387	351	329
4	Min MS	$d(s8) = d(s9) = 200$	SLK-2	821	1364	720	786
5	Min MS	$H = 12$	SLK-1	422	579	267	91
	Min MS	$d(s12) = 100$, $d(s13) = 200$	SK	727	1265	724	481
	Min MS	$d(s12) = 100$, $d(s13) = 200$	MG	1930	2272	2806	359

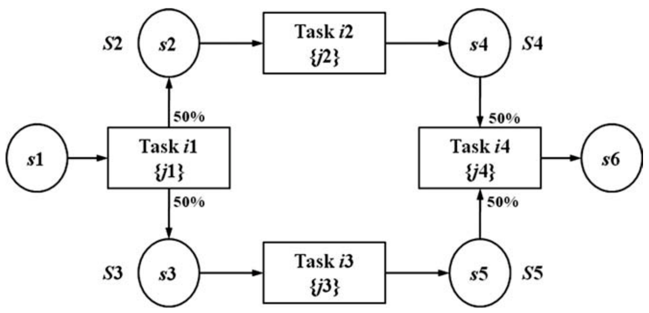

Figure 3. Recipe diagram for Example 1.

release $j2$ by allowing $s5$ to be transferred to $j3$ at 3 h. This enables $j2$ to have 3 h [3 h, 6 h] of production in SLKs instead of only 2 h [4 h, 6 h] in MG, SK, and SF. This forces MG, SK, and SF to use a lower batch size of $i5$ on $j2$ as compared to SLKs, and give an inferior solution. To ensure the validity of our results, we solved MG, SK, and SF with up to 15 slots/events. The schedules for this example are shown in Figure 6 (from SLKs) and Figure 7 (from MG, SK, and SF).

Now, we proceed to solve several cases of three more examples (Examples 2–4) from the literature to compare SLKs with MG, SK, and SF. However, as the latter do not allow nonsimultaneous material transfers, comparing them would not be fair. Therefore, we reduce SLKs by forcing simultaneous material transfers to enable a fair comparison between the five models. We do this by setting $\delta_{jk} = 0$ and eliminating θ_{jk} from SLKs. Appendix 1 lists the reduced models. The number of slots refers to the number of event points for SF and MG in all our subsequent discussion. In this work, we assumed UW policy for all examples and the following constraint sequences for the various models.

For revenue maximization:

Model SLK1 \1b, 1c, 2, 3a, 3b, 7a, A1, 9a, A3, A2, 8a, 10a–c, 5, 4a, 4c, 4b, 11\;

Model SLK2 \1b, 1c, 15a, 15b, 3a, 3b, 7a, A1, A2, 8a, A3, 9a, 10a–c, 5, 4a, 16a, 16b, 11\;

Table 3. Batch Size Data for Examples 1–5

Task	Label i	Unit	Label j	α_{ij}	β_{ij}	$B_{ij}^{L} - B_{ij}^{U}$ (mu)
			Example 1			
Task 1	1	Unit 1	j 1	1.666	0.03335	0–40
Task 2	2	Unit 2	j 2	2.333	0.08335	0–20
Task 3	3	Unit 3	j 3	0.667	0.0666	0–5
Task 4	4	Unit 4	j 4	2.667	0.00833	0–40
			Example 2			
Task 1	1	Unit 1	Unit 1	1.666	0.0778	0–30
Task 2	2	Unit 2	Unit 2	2.333	0.0667	0–10
Task 3	3	Unit 3	Unit 3	0.669	0.0777	0–30
Task 4	4	Unit 3	Unit 3	0.667	0.033325	0–40
Task 5	5	Unit 2	Unit 2	1.332	0.0556	0–30
Task 6	6	Unit 1	Unit 1	1.5	0.025	0–20
			Example 3			
Task 1	1	Unit 1	Unit 1	1.333	0.01333	0–100
		Unit 2	Unit 2	1.333	0.01333	0–150
Task 2	2	Unit 3	Unit 3	1	0.005	0–200
Task 3	3	Unit 4	Unit 4	0.667	0.00445	0–150
		Unit 5	Unit 5	0.667	0.00445	0–150
			Example 4			
Heating	H	Heater	HR	0.667	0.00667	0–100
Reaction-1	R1	Reactor 1	RR1	1.334	0.02664	0–50
		Reactor 2	RR2	1.334	0.01665	0–80
Reaction-2	R2	Reactor-1	RR1	1.334	0.02664	0–50
		Reactor-2	RR2	1.334	0.01665	0–80
Reaction-3	R3	Reactor-1	RR1	0.667	0.01332	0–50
		Reactor-2	RR2	0.667	0.00833	0–80
Separation	S	Separator	SR	1.3342	0.00666	0–200
			Example 5			
Heating-1	H1	Heater	HR	0.667	0.00667	0–100
Heating-2	H2	Heater	HR	1.000	0.01	0–100
Reaction-1	R1	Reactor 1	RR1	1.333	0.01333	0–100
		Reactor 2	RR2	1.333	0.00889	0–150
Reaction-2	R2	Reactor 1	RR1	0.667	0.00667	0–100
		Reactor 2	RR2	0.667	0.00445	0–150
Reaction-3	R3	Reactor 1	RR1	1.333	0.0133	0–100
		Reactor 2	RR2	1.333	0.00889	0–150
Separation	S	Separator	SR	2.000	0.00667	0–300
Mixing	M	Mixer 1	MR1	1.333	0.00667	20–200
		Mixer 2	MR2	1.333	0.00667	20–200

Table 4. Storage Capacities, Initial Inventories, and Material Prices for Examples 1–5

	Example 1		Example 2		Example 3		Example 4		Example 5	
Material s	Storage Capacity (mu) I_s^U	Initial Inventory (mu) I_s^0	Storage Capacity (mu) I_s^U	Initial Inventory (mu) I_s^0	Storage Capacity (mu) I_s^U	Initial Inventory (mu) I_s^0	Storage Capacity (mu) I_s^U	Initial Inventory (mu) I_s^0	Storage Capacity (mu) I_s^U	Initial Inventory (mu) I_s^0
1	UL	AA	UL	AA	UL	AA	UL	AA	UL	AA
2	10	0	UL	AA	200	0	UL	AA	UL	AA
3	15	0	UL	AA	250	0	UL	AA	100	0
4	10	0	10	0	UL	0	100	0	100	0
5	15	0	5	0	–	–	200	0	300	0
6	UL	0	10	0	–	–	150	0	150	50
7	–	–	UL	0	–	–	200	0	150	50
8	–	–	UL	0	–	–	UL	–	UL	AA
9	–	–	–	–	–	–	UL	–	150	0
10	–	–	–	–	–	–	–	–	150	0
11	–	–	–	–	–	–	–	–	UL	AA
12	–	–	–	–	–	–	–	–	UL	0
13	–	–	–	–	–	–	–	–	UL	0

UL = Unlimited; AA = Available as and when required.
Example 1: Price for $s6$ is 10 \$/mu.
Example 2: Price for $s7$ is 10 \$/mu and s8 is 5 \$/mu.
Example 3: Price for $s4$ is 5 \$/mu.
Example 4: Price for $s8$ and $s9$ is 10 \$/mu.
Example 5: Price for $s12$ and $s13$ is 5 \$/mu.

Model SK \2, 3, 4, 5, 6, 9, 10, 11, 12, 13, 14, 20\;

Model MG \11, 4, 5, 6, 12, 15–21, 3, 7, 23–30, 34–36\;

Model SF \1–17, 24, 25, 28–31, 33, 34\.

For makespan minimization:

Model SLK1 \1b, 1b, 2, 3a, 3b, 7a, A1, 9a, A3, 8a, A2, 10a–c, 5, 4a, 4c, 4b, 12, 13a-b\;

Model SLK2 \1a-b, 15a–b, 3a–b, 7a, A1, A2, 8a, A3, 9a, 10a–c, 5, 4a, 16a, 16b, 12, 13a-b\;

Model SK \22, 2, 3, 4, 5, 6, 9, 10, 11, 12, 13, 14, 21\;

Model MG \11, 4, 5, 6, 12, 15–21, 3, 7, 23–30, 34–36, 41\;

Model SF \1–17, 24, 25, 28–32\.

The aforementioned equation numbers are from Sundaramoorthy and Karimi[34] for SK, Maravelias and Grossmann[41] without resource constraints for MG, and Shaik and Floudas[5] for SF. The sequences correspond to the orders in which the equations are presented in the respective papers. As the constraint sequences of MG, SK, and SF can also be "tuned", our numerical comparison is subject to the limitation of the aforementioned assumed sequences. Furthermore, as it was tedious to solve for all possible values of Δn in SF, we restricted to $\Delta n = 0$ and $\Delta n = 1$.

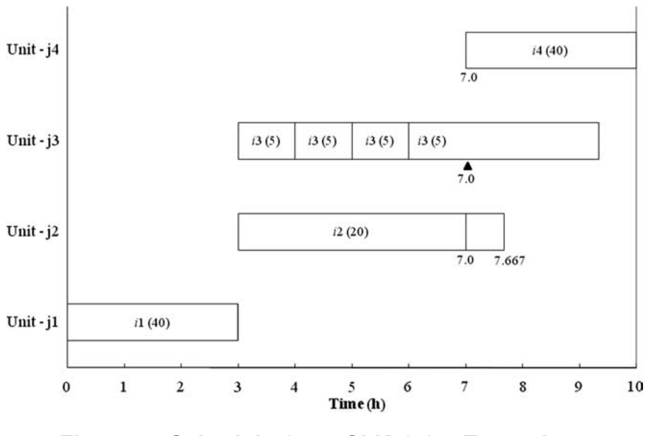

Figure 4. Schedule from SLK-2 for Example 1.

Example 3

This example[34] has been studied extensively in the literature. It involves three tasks ($i1$–$i3$), 5 units ($j1$–$j5$), four material states ($s1$–$s4$), and two storages ($S2$, $S3$). $j1$ and $j2$ can process $i1$, $j3$ can process $i2$, $j4$ and $j5$ can process $i3$. Figure 8 shows the recipe diagram. Tables 3 and 4 list the complete data.

First, consider revenue maximization for three scheduling horizons (3a: $H = 10$ h, 3b: 12 h, and 3c: 16 h). Table 5 gives the model and solution statistics. For this small problem, all models (SLK1, SLK2, SK, MG, and SF) expectedly have nearly similar statistics. For $H = 10$ h, SLK1 and SLK2 both need 6 slots ($K = 7$) to obtain the optimal solution of \$2628.2 reported in the literature. However, if the same number of slots is the same, then single-grid models (SK and MG) give better RMIP objectives than multi-grid models (SLK-1, SLK-2, SF). This is mainly because the latter use several big-M constraints to synchronize the timings on different time grids.

For 3b ($H = 12$ h) and 3c ($H = 16$ h), the multi-grid models (SLK-1, SLK-2, and SF) solve slower than SK and MG. This is because multi-grid models are unable to reduce the number of slots as compared with single-grid models, and give poor RMIP values, which clearly shows their limitation. Both SK and MG give better RMIP objectives of \$4481 and \$4563.8 respectively, while the multi-grid models give \$4951.2. While SF solves faster for $\Delta n = 0$, one would need to solve it several times to get the best solution as

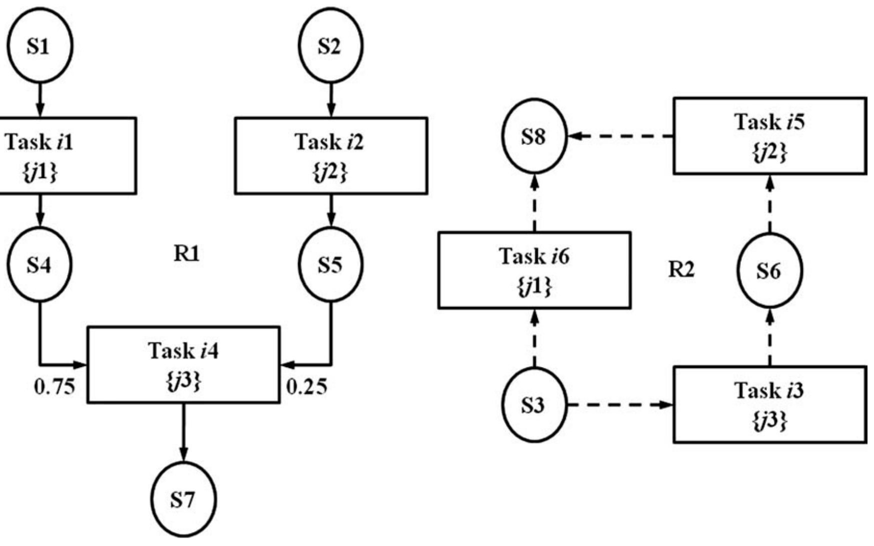

Figure 5. Recipe diagram for Example 2.

discussed earlier. Thus, it is not feasible or fair to compare the solution times of SF with those of other models.

Figure 9 gives the schedule for Example 3b from SLK2. The rectangular blocks give task durations. Start/end times of batches along with slot numbers are shown under each block, and the batch sizes are indicated as labels.

For makespan minimization, we consider two cases with different product demands, $d_4 = 2000$ mu and 4000 mu. Table 5 lists the model and solution statistics. For $d_4 = 2000$ mu (Example 3d), both single-grid and multi-grid models need 16 slots ($K = 17$) to get the optimal solution of 28.772 h. However, single-grid models (SK and MG) give a better RMIP objective of 24.7 h as compared to 24.2 h for the multi-grid models (SLK-1, SLK-2, and SF). We allow a limit of 10,000 CPU s for this example. SLK-1, SF ($\Delta n = 0$), and MG do not achieve the best solution of 28.772 h within 10,000 CPU s. However, SK converges to 0% relative gap within 5403 s of CPU time. SLK-2 does not converge within 10,000 CPU s, but attains the best solution. Here again, a single-grid model performs better than the multi-grid models due to the same number of slots.

For the more difficult case (Example 3e) of $d_4 = 4000$ mu, SLK-1, SLK-2, and SF require 23 slots ($K = 24$) to get the best makespan of 56.432 h. While SLK-2 converges in 4944 CPU s, SLK-1 does not converge in 10000 CPU s. SK and MG need 26 slots/events get the best solution, but after 10,000 CPU s. In this example, the reduction (Table 5) in the number of slots enables one multi-grid model (SLK-2) to outperform the single-grid models.

Note that the number of variables for our implementation of SF is slightly more than that reported in Shaik and Floudas,[5] because we do not fix any variables (binary or continuous) based on specific problem details for the sake of a fair comparison. In contrast, Shaik and Floudas[5] fixed some variables for their model, but not for any other model.

Example 4

This example[53] is more complex than Example 3 and has been studied extensively in the literature. Figure 10 gives the recipe diagram. It involves five tasks ($i1$–$i5$), four processing

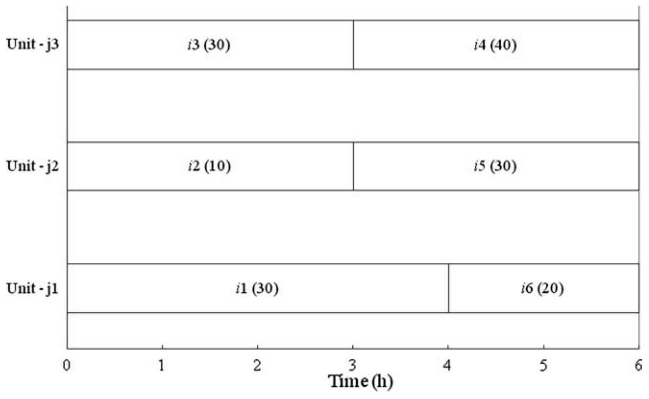

Figure 6. Schedule from SLKs for Example 2.

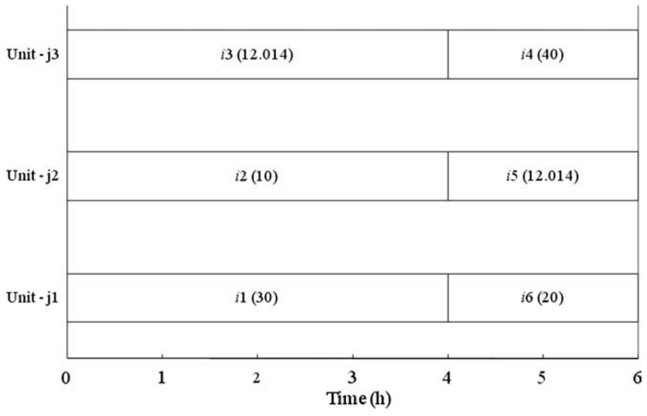

Figure 7. Schedule from MG, SK, and SF for Example 2.

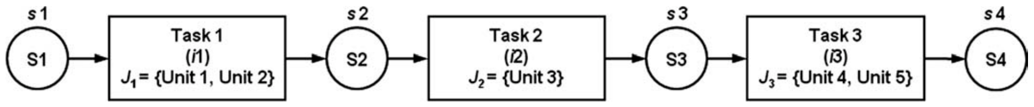

Figure 8. Recipe diagram for Example 3 (Sundaramoorthy and Karimi[34]).

units (HR, RR1, RR2, SR), nine materials (*s*1–*s*9), and four storage units (*S*4–*S*7). HR can perform *i*1, RR1 and RR2 can perform *i*2–*i*4, and SR can perform *i*5. Tables 3 and 4 list the data.

For revenue maximization, we use three scheduling horizons (4a: $H = 10$ h, 4b: 12 h, and 4c: 16h). For Example 4a, Table 6 shows that multi-grid models require one fewer slot than single-grid models. However, in spite of this, the latter give a better RMIP objective of 2690.6 vs. 2730.7. This again may be due to the absence of big-M constraints in the latter. SLK-1 and SLK-2 solve faster (28.2 s and 12 s) than SK and MG (57.8 s and 126 s).

For Example 4b, SLKs perform exceptionally well and solve an order-of-magnitude faster than SK and MG. SLKs

Table 5. Model and Solution Statistics for Examples 3

Model	*K*	*H*	CPU time (s)	Nodes	RMILP ($)	MILP ($)	Binary Variables	Continuous Variables	Constraints	Nonzeros	Relative Gap (%)
Example 3a ($H = 10$)											
SLK2	7	10	1.95	2764	4000.0	2628.2	60	359	491	1556	–
SLK1	7	10	3.20	6341	4000.0	2628.2	60	324	461	1496	–
SF											
($\Delta n = 0$)	6	10	0.17	285	3973.9	2628.2	30	119	209	639	–
($\Delta n = 1$)	6	10	1.33	1607	4000.0	2628.2	55	144	479	1539	–
SK	7	10	1.06	1090	3384.3	2628.2	60	316	300	1001	–
MG	7	10	0.88	770	3548.4	2628.2	70	309	846	2766	–
Example 3b ($H = 12$)											
SLK2	9	12	781	248,340	4951.2	3463.6	80	467	657	2082	–
SLK1	9	12	1492	600,476	4951.2	3463.6	80	422	617	2002	–
SF											
($\Delta n = 0$)	8	12	1.88	8136	4951.2	3463.6	40	157	281	865	–
($\Delta n = 1$)	8	12	585	254,877	4951.2	3463.6	75	192	657	2117	–
SK	9	12	11.6	12,480	4481.0	3463.6	80	416	408	1359	–
MG	9	12	10.3	32,092	4563.8	3463.6	90	397	1084	3879	–
Example 3c ($H = 16$)											
SLK2	12	16	10,000	1,628,804	6601.7	5038.1	110	629	906	2871	18
SLK1	12	16	10,000	1,294,912	6601.7	5038.1	110	569	851	2761	19.1
SF											
($\Delta n = 0$)	11	16	113	484,764	6601.7	5038.1	55	214	389	1204	–
($\Delta n = 1$)	11	16	10,000	1,727,132	6601.7	5038.1	105	264	909	2984	15.87
SK	12	16	377	461,037	6312.6	5038.1	110	566	570	1896	–
MG	12	16	2431	1,974,025	6332.8	5038.1	120	529	1441	5811	–
Model	*K*	*H*	CPU time (s)	Nodes	RMILP (h)	MILP (h)	Binary Variables	Continuous Variables	Constraints	Nonzeros	Relative Gap (%)
Example 3d: $d(s4) = 2000$ mu											
SLK2	17	50	10,000	328,879	24.2	28.772	160	901	1330	4203	15.8
SLK1	17	50	10,000	380,619	24.2	28.772	160	816	1250	4043	15.8
SF											
($\Delta n = 0$)	16	50	10,000	1,093,172	24.2	28.884	80	309	574	1783	6.6
($\Delta n = 1$)	16	50	10,000	769,022	24.2	28.772	155	384	1344	4443	15.8
SK	17	–	5403	3,214,852	24.72	28.772	160	816	843	2794	–
MG	17	50	10,000	1,210,125	24.7	29.5	170	750	2045	9879	8.88
Example 3*e*: $d(s4) = 4000$ mu											
SLK2	23	100	4944	1,522,250	48.5	56.432	220	1225	1828	5781	–
SLK1	23	100	10,000	1,880,663	48.5	56.432	220	1110	1718	5561	1.89
SF											
($\Delta n = 0$)	22	100	34.6	42,758	48.5	56.432	110	423	790	2461	–
($\Delta n = 1$)	22	100	8586	1,957,756	48.5	56.432	215	528	1860	6177	–
SK	26	–	10,000	31,15,485	49.11	56.432	250	1266	1329	4405	6.04
MG	26	100	10,000	562,110	49.01	56.432	260	1146	3116	19212	10.25

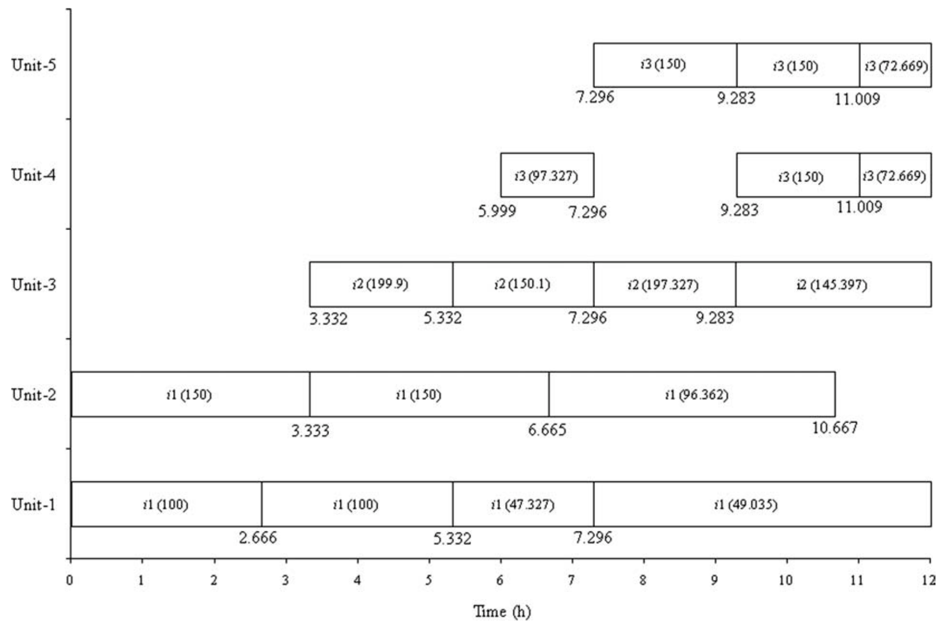

Figure 9. Maximum-revenue schedule from SLK-2 for Example 3b.

require 7 slots ($K = 8$) to get the optimal solution (Figure 11) of \$2658.5. Surprisingly, even though SF is a multi-grid model, it needs one more event-point than those in SLKs. In other words, our proposed multi-grid approach is more effective in reducing slots than SF. In contrast, the single-grid models require 10 slots/events ($K = 11$), so our approach reduces three slots for this example. Consequently, SLK-1 and SLK-2 solve much faster (38 s for SLK-2 and 62.1 s for SLK-1 vs. 3330 s for SK and 9124.5 s for MG) and require fewer binary variables (84 vs. 120 for SK and 160 for MG). SLKs also give the best RMIP objective of 3301.0 compared to 3350.5 for SF and 3343.4 for SK and MG.

For Example 4c, SLKs again perform quite well. They need only 8 slots ($K = 9$) compared to 9 slots/events ($K = 10$) for SF, SK, and MG to get the best solution of \$3738.38. This again confirms the ability of our approach to reduce slots, where SF does not. This reduction enables SLK-1 and SLK-2 to solve in only 30 s and 76.1 s, respectively, while SK and MG require 156 s and 703 s, respectively. The RMILP objective (\$4291.7.0) from SLKs is

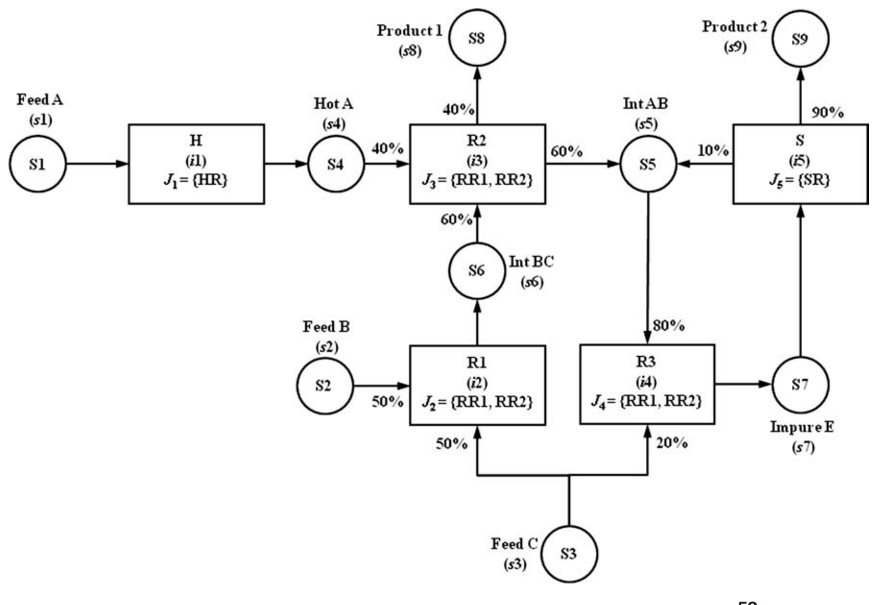

Figure 10. Recipe diagram for Example 4 (Kondili et al.[53]).

Table 6. Model and Solution Statistics for Examples 4

Model	K	H	CPU time (s)	Nodes	RMILP (\$)	MILP (\$)	Binary Variables	Continuous Variables	Constraints	Nonzeros	Relative Gap (%)
Example 4a (H = 10)											
SLK2	7	10	12.0	19,043	2730.7	1962.7	72	449	645	2319	–
SLK1	7	10	28.2	44,015	2730.7	1962.7	72	421	621	2315	–
SF											
($\Delta n = 0$)	6	10	3.19	7978	2730.7	1931.92*	48	208	439	1415	–
($\Delta n = 1$)	6	10	8.13	10,915	2730.7	1962.7	88	248	847	3150	–
SK	8	10	57.8	65,587	2690.6	1962.7	84	489	458	1686	–
MG	8	10	126	54,753	2690.6	1962.7	112	617	1468	5464	–
Example 4b (H = 12)											
SLK2	8	12	38.0	58,065	3301.0	2658.5	84	517	753	2710	–
SLK1	8	12	62.1	80,093	3301.0	2658.5	84	485	725	2706	–
SF											
($\Delta n = 0$)	8	12	53.2	137,217	3350.5	2658.5	64	274	597	1925	–
($\Delta n = 1$)	8	12	825	899,959	3350.5	2658.5	120	330	1157	4342	–
SK	11	12	3330	2,614,049	3343.4	2658.5	120	687	665	2442	–
MG	11	12	9125	3,174,288	3343.4	2658.5	160	842	2020	8467	–
Example 4c (H = 16)											
SLK2	9	16	30.0	32,531	4291.7	3738.38	96	585	861	3101	–
SLK1	9	16	76.1	62,786	4291.7	3738.38	96	549	829	3097	–
SF											
($\Delta n = 0$)	9	16	140	315,573	4438.9	3738.38	72	307	676	2180	–
($\Delta n = 1$)	9	16	3903	3,554,870	4438.9	3738.38	136	371	1312	4938	–
SK	10	16	156	96,734	4318.8	3738.38	108	621	596	2190	–
MG	10	16	703	200,592	4318.8	3738.38	144	767	1836	7410	–
Model	K	H	CPU time (s)	Nodes	RMILP (h)	MILP (h)	Binary Variables	Continuous Variables	Constraints	Nonzeros	Relative Gap (%)
Example 4d: $d(s8)$ & $d(s9) = 200$ mu											
SLK2	10	50	821	175,107	18.7	19.34	108	658	983	3517	–
SLK1	10	50	1276	152,939	18.7	19.34	108	618	947	3513	–
SF											
($\Delta n = 0$)	9	50	1.14	507	18.7	19.34	72	307	685	2199	–
($\Delta n = 1$)	9	50	1331	203,829	18.7	19.34	136	371	1321	4957	–
SK	10	–	171	55,349	18.7	19.34	108	621	604	2197	–
MG	10	50	314	36,146	18.7	19.34	160	842	1962	7767	–
Example 4e: $d(s8) = 500$ mu & $d(s9) = 400$ mu											
SLK2	22	100	10,000	302,206	47.4	47.6835	252	1474	2279	8209	0.64
SLK1	22	100	10,000	260,603	47.4	47.6835	252	1186	2195	8205	0.64
SF											
($\Delta n = 0$)	21	100	10,000	986,563	47.5	47.754	168	703	1633	5259	0.49
($\Delta n = 1$)	21	100	10,000	340,927	47.4	49.012	328	863	3181	12,109	3.34
SK	23	–	10,000	398,979	48.78	49.05	264	1479	1501	5473	0.55
MG	23	100	10,000	59,431	48.78	49.05	368	1934	4471	26,279	0.55

*Suboptimal.

also better than that (\$4318.8) from SK, MG, and SF (\$4438.9). Figure 12 shows the optimal schedule from SLK-2.

For makespan minimization, we solve for two demands, $d_8 = d_9 = 200$ mu (Example 4d) and ($d_8 = 500$, $d_9 = 400$ mu) (Example 4e). Table 6 gives the model and solution statistics. For Example 4d, SLKs perform worse than SK and MG, because the unit-slots are identical to process-slots. For Example 4e, although no model converges within 10,000 CPU s, SLKs and SF need one slot less ($K = 22$) than SK and MG ($K = 23$). However, SLKs attain a better solution of \$47.6835 than \$47.75 for SF and \$49.05 for SK and MG.

Example 5

This example[34] (Figure 13) involves six processing units (HR, RR1, RR2, SR, MR1, and MR2), seven tasks (*i*1–*i*7), 13 material states (*s*1–*s*13), and seven storage units (*S*3–*S*7, *S*9, and *S*10). Relatively, this is a more complex problem, and hence used often in the literature. It embodies many common features of an MBP such as units performing multiple tasks, multiple units suitable for a task, and dedicated units for specific tasks. It also assumes nonzero initial inventories for *s*6 and *s*7, and recycles *s*4.

For revenue maximization, we use four scheduling horizons; Example 5a: $H = 8$ h, Example 5b: $H = 10$ h, Example 5c: $H = 12$ h, and Example 5d: $H = 16$ h. From

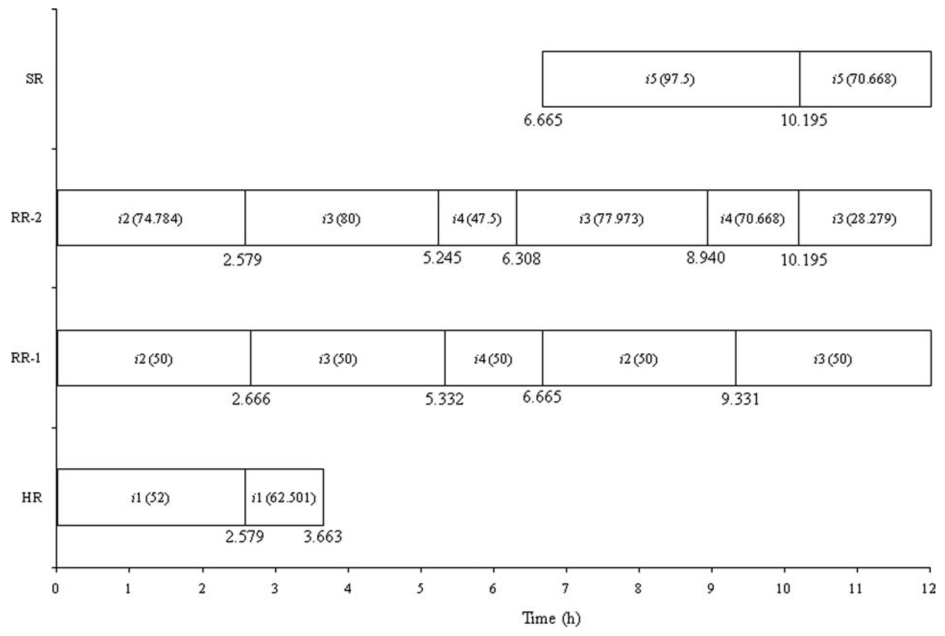

Figure 11. Maximum-revenue schedule from SLK-2 for Example 4b.

Table 7, all models require six slots/events ($K = 7$) for $H = 8$. Again, single-grid models perform better. For Example 5b ($H = 10$), although no model converges in 10,000 CPU s, SLKs and SF require fewer slots (9 vs. 10) than SK and MG. Also, SLKs and SF obtain a solution of \$2337.36 in 10,000 s compared to \$2260.9 for SK and \$2137.1 for MG. For Example 5c ($H = 12$), SLKs and SF again require fewer slots/events (8 vs. 9) than SK and MG. The RMILP objectives are also better (\$3465.6 vs. \$3867.3), and SLK-2 is significantly faster (10.1 s vs. 95.4 s for SK and 296 s for MG). SLK-1 takes a relatively long time (>400 s) to converge. For $H = 16$ h, although no model converges within 10000 CPU s, SLK-2 and SF attain a better objective (\$4241.5 vs. \$4240.83 for others).

For makespan minimization, we consider Example 5e: $d_{12} = 100$ and $d_{13} = 200$ mu, and Example 5f: $d_{12} = d_{13} = 250$ mu. For Example 5e, SLKs use up to three fewer slots (8 vs. 9 for SF and 8 vs. 11 for SK and MG). SLK-2 is

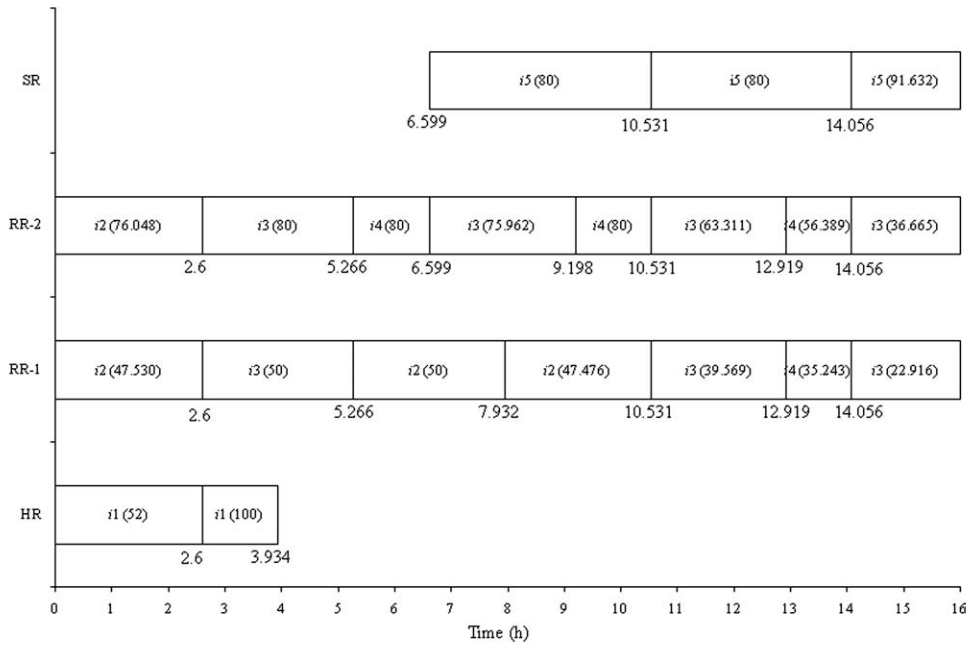

Figure 12. Maximum-revenue schedule from SLK-2 for Example 4c.

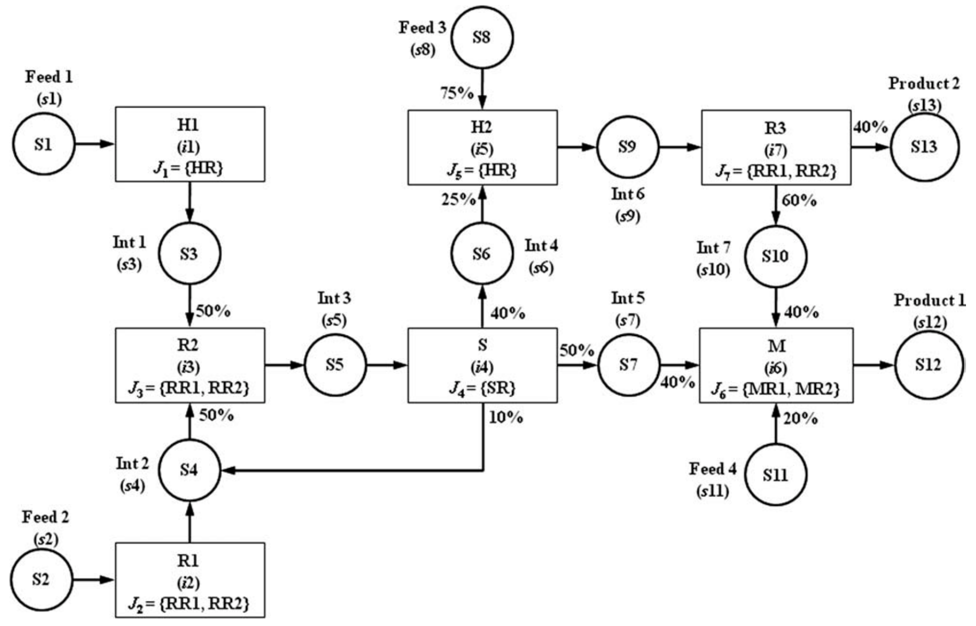

Figure 13. Recipe diagram for Example 5 (Sundaramoorthy and Karimi[34]).

significantly faster than SK and MG (249 s vs. 727 s for SK and 1930 s for MG). Figure 14 gives the detailed schedule. For Example 5f also, SLKs outperform SK and MG significantly as seen in Table 7.

Remarks

Our numerical evaluation demonstrates that our unit-slot models perform much better than some general models in the literature for both revenue maximization and makespan minimization except where unit-slots are identical to process-slots. Although SLK-1 performs nearly the same as SLK-2, SLK-2 seems to be more consistent across the limited problems considered in this work. Our models are much simpler in implementation and do not require cascaded iterations like SF. Additionally, as seen in some examples, our models require even fewer slots than the unit-specific event-based models (SF).

The main difference between our models (SLKs) and SK is of course the use of unit-slots and consequent additional variables (T_{jk} and T_{sk}) and constraints (Eqs. 7 and 8) for synchronizing transfer timings between processing units and storages. However, some additional differences are worth noting.

1. In contrast to SK, SLKs do not use the slot lengths (SL_{jk}) as variables. This eliminates the constraint that forces the sum of slot lengths to be less than H.

2. Instead of writing Eq. 5 for b_{ijk} as in SK and SLKs, we experimented with an aggregated form as follows:

$$b_{jk} = b_{j(k-1)} + \sum_{i \in \mathbf{I}_j, i>0} (B^{\mathrm{L}}_{ij} ys_{ijk} + \Delta BI_{ijk}) + \sum_{i \in \mathbf{I}_j, i>0} BO_{ijk}$$

3. Our motivation for trying the aforementioned form was to reduce the numbers of variables and constraints. However, we discovered that the above equation leads to a poor relaxation.

4. In contrast to SK, we express BI_{ijk} as $BI_{ijk} = B^{\mathrm{L}}_{ij} ys_{ij} + \Delta BI_{ijk}$ to reduce the number of constraints. Furthermore, Eqs. 4b and 4c also eliminate several constraints that SK uses.

5. We have introduced an alternate objective of profit in addition to revenue and makespan, which are well known in the literature.

6. We have also generalized the concept of stoichiometric coefficient to Mass Ratio (σ_{sij}), which captures unit dependency and enables a proper mass balance on materials involved in a batch.

7. The use of Eq. 2 is also novel compared to SK. It is mystery why SLK-2 seems to perform slightly better than SLK-1 in spite of SLK-1 using fewer variables and constraints.

Conclusion

We proposed a novel continuous-time formulation for scheduling multipurpose batch plants. Its major contributions are a fool-proof and novel use of unit-slots in managing shared resources such as materials and the flexibility to allow nonsimultaneous transfers of materials into a batch. It gives rational and logical arguments and constraints for resource balance using unit-slots. Similar to the unit-specific event-based models, it does not use the extra binary variables used by Lim and Karimi.[25] For some literature examples, our approach needs fewer slots/events than both single-grid and multi-grid models that exist in the literature. This enables significant reductions in solution times and model size, and yields tighter RMIP values. However, in problems where unit-slots are identical to process slots, the performance of multi-grid models is worse than single-grid models. Thus, it demonstrates the limitation of the current multi-grid models. Lastly, this work highlights the importance of constraint sequencing in

Table 7. Model and Solution Statistics for Examples

Model	K	H	CPU time (s)	Nodes	RMILP (\$)	MILP (\$)	Binary Variables	Continuous Variables	Constraints	Nonzeros	Relative Gap (%)
Example 5a ($H = 8$)											
SLK2	7	8	284	388,832	2751.0	1583.4	102	655	1070	3671	–
SLK1	7	8	2659	3,674,795	2751.0	1583.4	102	613	1034	3654	–
SF											
($\Delta n = 0$)	6	8	9.37	18,578	2751.0	1583.4	66	290	716	2169	–
($\Delta n = 1$)	6	8	56.8	80,223	2751.0	1583.4	121	345	1336	4628	–
SK	7	8	45.5	39,305	2560.6	1583.4	102	595	728	2207	–
MG	7	8	81.2	55,146	2560.6	1583.4	154	806	1893	6630	–
Example 5b ($H = 10$)											
SLK2	9	10	10,000	872,204	3618.6	2337.36	136	853	1428	4907	5.07
SLK1	9	10	10,000	711,493	3618.6	2337.36	136	799	1380	4886	11.6
SF											
($\Delta n = 0$)	8	10	115	1,160,988	3618.6	2292.5*	88	382	884	2847	–
($\Delta n = 1$)	8	10	10,000	2,256,858	3618.6	2337.36	165	459	1658	6187	1.68
SK	10	10	10,000	2,666,604	3473.9	2260.9*	153	874	1121	3380	10.7
MG	10	10	10,000	1,415,516	3473.9	2137.1*	220	1151	2691	10,710	23.9
Example 5c ($H = 12$)											
SLK2	8	12	10.1	3096	3465.6	3041.3	119	754	1249	4289	–
SLK1	8	12	422	236,847	3465.6	3041.3	119	706	1207	4270	–
SF											
($\Delta n = 0$)	7	12	2.11	1262	3465.6	3041.3	77	336	844	2560	–
($\Delta n = 1$)	7	12	4.44	881	3465.6	3041.3	143	402	1579	5505	–
SK	9	12	95.4	43,951	3867.3	3041.3	136	781	990	2989	–
MG	9	12	296	71,877	3867.3	3041.3	198	1036	2425	9269	–
Example 5d ($H = 16$)											
SLK2	11	16	10000	705,090	5225.9	4241.5	170	1051	1786	6143	0.29
SLK1	11	16	10000	3,042,257	5225.9	4237.61	170	985	1726	6118	1.64
SF											
($\Delta n = 0$)	10	16	1475	515,144	5225.9	4241.5	110	474	1228	3733	–
($\Delta n = 1$)	10	16	10000	1,511,596	5225.9	4241.5	209	573	2311	8136	0.01
SK	11	16	1687	283,938	5125.9	4240.83	170	967	1252	3771	–
MG	11	16	10000	739,728	5125.9	4185.24	242	1266	2957	12,232	1.69
Model	K	H	CPU time (s)	Nodes	RMILP (h)	MILP (h)	Binary Variables	Continuous Variables	Constraints	Nonzeros	Relative Gap (%)
Example 5e: $d(s\ 12) = 100$ mu & $d(s13) = 200$ mu											
SLK2	9	50	249	45,218	11.3	13.367	136	859	1448	4944	–
SLK1	9	50	2538	502,461	11.3	13.367	136	805	1400	4923	–
SF											
($\Delta n = 0$)	9	50	70.7	43,967	11.3	13.367	99	428	1112	3369	–
($\Delta n = 1$)	9	50	1867	770,883	11.3	13.367	187	516	2079	7286	–
SK	11	–	727	233,210	11.417	13.367	170	967	1264	3782	–
MG	11	50	1930	162,982	11.417	13.367	242	1267	2970	12,427	–
Example 5f: $d(s12)$ & d($s13$) = 250 mu											
SLK2	11	100	9.70	820	14.3	17.025	170	1057	1806	6180	–
SLK1	11	100	6.34	663	14.3	17.025	170	991	1746	6155	–
SF											
($\Delta n = 0$)	10	100	5.38	2236	14.3	17.199	110	474	1240	3760	–
($\Delta n = 1$)	10	100	7.30	683	14.3	17.025	209	573	2323	8163	–
SK	12	–	107	12,992	15.001	17.306	187	1060	1395	4173	–
	13	–	387	29,981	14.920	17.306	204	1153	1526	4564	–
MG	12	100	247	14,683	15.001	17.306	264	1382	3236	14,047	–
	13	100	6798	244,438	14.920	17.306	286	1497	3502	15,748	–

*Suboptimal.

GAMS implementation for evaluating MILP-based scheduling models fairly. While this particular work has not addressed all the other features of MBPs in the literature such as changeover times, semicontinuous processes, resources other than materials, etc. the proposed approach is readily extendible.

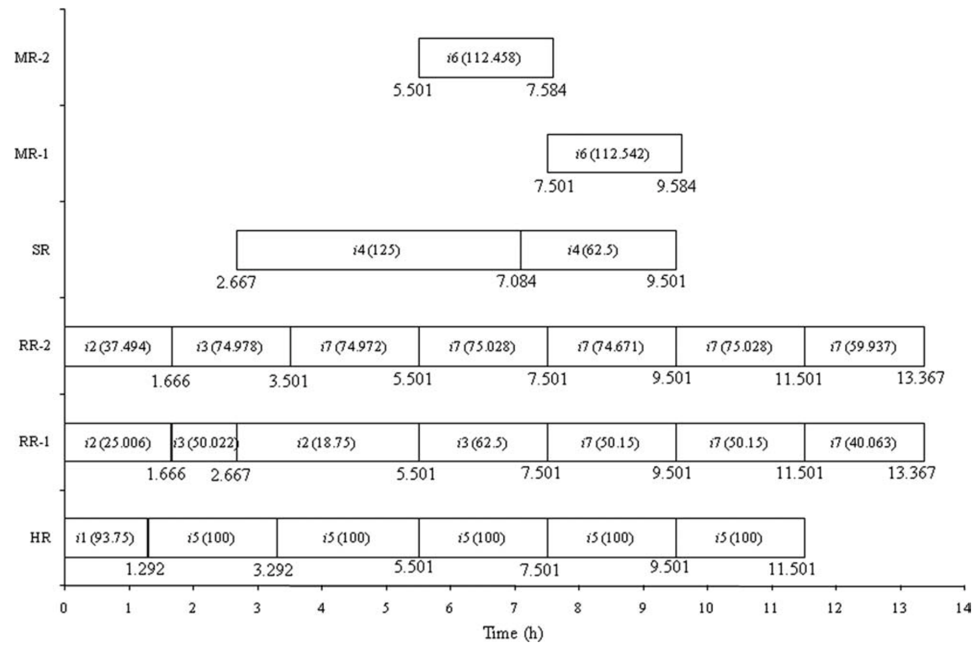

Figure 14. Minimum-makespan schedule from SLK-2 for Example 5e.

Acknowledgments

The authors would like to acknowledge financial support for this work from The Agency for Science, Technology, and Research (A*Star) under grant 052 116 0074 and National University of Singapore. The authors are indebted to the anonymous reviewers whose detailed and constructive comments helped improve this manuscript considerably. They are especially thankful to an anonymous reviewer who painstakingly produced clever examples to challenge initial versions of our model, which they improved through our revisions.

Notation

Indices

s =	material state
i =	task
j =	unit
k =	Slot

Superscripts

L =	lower limit
U =	upper limit

Sets

$\mathbf{I}_j$ =	Tasks that unit j can perform

Parameters

σ_{sij} =	Mass Ratio for material s in task i on unit j
H =	Scheduling Horizon
υ_s =	Unit price of s
α_{ij} =	Parameter for determining the processing time of task i on unit j
β_{ij} =	Parameter for determining the processing time of task i on unit j
d_s =	Minimum demand for material s

Variables

T_{jk} =	Time at which slot k on unit j ends
T_{sk} =	Time at which slot k on storage s ends
BI_{ijk} =	Amount of task i entering unit j at the start of slot k
ΔBI_{ijk} =	Amount above the minimum batch size in BI_{ijk}
b_{ijk} =	Amount of task i that resides in unit j at the end of slot k
BO_{ijk} =	Batch material output by task i at its completion within slot k
t_{jk} =	Time remaining at T_{jk} to complete the task in progress in slot k on unit j
I_{sk} =	Inventory level of s at T_{jk}
δ_{jk} =	Time period for which unit j idles in the beginning of slot k
θ_{jk} =	Time period for which unit j idles towards the end of slot k

Binary

ys_{ijk} =	1, if unit j begins a task at the beginning of slot k
0–1 =	Continuous
z_{jk} =	1, if unit j ends an ongoing task within slot k
ye_{ijk} =	1, if unit j ends a task i within slot k
y_{ijk} =	1, if unit j continues task i at time T_{jk}

Literature Cited

1. Méndez CA, Cerdá J, Grossmann IE, Harjunkoski I, Fahl M. State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Comput Chem Eng*. 2006;30:913–946.
2. Floudas CA, Xiaoxia L. Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. *Comput Chem Eng*. 2004;28:2109–2129.
3. Pitty SS, Karimi IA. Novel MILP models for scheduling permutation flowshops. *Chem Product Process Model*. 2008;3:22.
4. Janak SL, Floudas CA. Improving unit-specific event based continuous-time approaches for batch processes: integrality gap and task splitting. *Comput Chem Eng*. 2008;32:913–955.
5. Shaik M, Floudas C. Novel unified modeling approach for short-term scheduling. *Ind Eng Chem Res*. 2009;48:2947–2964.
6. Li J, Susarla N, Karimi IA, Shaik M, Floudas CA. An analysis of some unit-specific event-based models for the short-term scheduling of non-continuous processes. *Ind Eng Chem Res*. Accepted.
7. Méndez CA, Cerdá J. Optimal scheduling of a resource-constrained multiproduct batch plant supplying intermediates to nearby end-product facilities. *Comput Chem Eng*. 2000;24:369–376.
8. Hui C-W, Gupta A, van der Meulen HAJ. A novel MILP formulation for short-term scheduling of multi-stage multi-product batch plants with sequence-dependent constraints. *Comput Chem Eng*. 2000;24:2705–2717.
9. Gupta S, Karimi IA. Scheduling a two-stage multiproduct process with limited product shelf life in intermediate storage. *Ind Eng Chem Res*. 2003;42:490–508.

10. Gupta S, Karimi IA. An improved MILP formulation for scheduling multiproduct, multistage batch plants. *Ind Eng Chem Res*. 2003;42: 2365–2380.
11. Liu Y, Karimi IA. Novel continuous-time formulations for scheduling multi-stage batch plants with identical parallel units. *Comput Chem Eng*. 2007;31:1671–1693.
12. Mendez CA, Henning GP, Cerda J. An MILP continuous-time approach to short-term scheduling of resource-constrained multistage flowshop batch facilities. *Comput Chem Eng*. 2001;25:701–711.
13. Mendez CA, Cerda J. An MILP continuous-time framework for short-term scheduling of multipurpose batch processes under different operation strategies. *Opt Eng*. 2003;4:7–22.
14. Mendez CA, Cerda J. Short-term scheduling of multistage batch processes subject to limited finite resources. *Comput Aided Chem Eng*. 2004;15B:984–989.
15. Ferrer-Nadal S, Capon-Garcia E, Mendez CA, Puigjaner L. Material transfer operations in batch scheduling. A critical modeling issue. *Ind Eng Chem Res*. 2008;47:7721–7732.
16. Wagner H. An integer linear-programming model for machine scheduling. *Naval Res Logistics Q*. 1959;6(2):131–140.
17. Ku HM, Rajagopalan D, Karimi I. Scheduling in batch processes. *Chem Eng Prog*. 1987;83:35–45.
18. Ku HM, Karimi IA. Scheduling in serial multiproduct batch processes with finite interstage storage: mixed integer linear program formulation. *Ind Eng Chem Res*. 1988;27:1840–1848.
19. Ku HM, Karimi I. Scheduling in serial multiproduct batch processes with due-date penalties. *Ind Eng Chem Res*. 1990;29:580–590.
20. Birewar D, Grossmann I. Incorporating scheduling in the optimal design of multiproduct batch plants. *Comput Chem Eng*. 1989;13: 141–161.
21. Geoffrion A, Graves G. Scheduling parallel production lines with changeover costs: Practical application of a quadratic assignment/LP approach. *Oper Res*. 1976:595–610.
22. Sahinidis NV, Grossmann IE. MINLP model for cyclic multiproduct scheduling on continuous parallel lines. *Comput Chem Eng*. 1991; 15:85–103.
23. Pinto JM, Grossmann IE. Optimal cyclic scheduling of multistage continuous multiproduct plants. *Comput Chem Eng*. 1994;18:797–816.
24. Pinto JM, Grossmann IE. A continuous-time mixed-integer linear-programming model for short-term scheduling of multistage batch plants. *Ind Eng Chem Res*. 1995;34:3037–3051.
25. Lim M-F, Karimi IA. Resource-constrained scheduling of parallel production lines using asynchronous slots. *Ind Eng Chem Res*. 2003;42:6832–6842.
26. Liu Y, Karimi IA. Scheduling multistage, multiproduct batch plants with nonidentical parallel units and unlimited intermediate storage. *Chem Eng Sci*. 2007;62:1549–1566.
27. Liu Y, Karimi IA. Scheduling multistage batch plants with parallel units and no interstage storage. *Comput Chem Eng*. 2008;32:671–693.
28. Schilling G, Pantelides C. A simple continuous-time process scheduling formulation and a novel solution algorithm. *Comput Chem Eng*. 1996;20:1221–1226.
29. McDonald CM, Karimi IA. Planning and scheduling of parallel semicontinuous processes. I. Production planning. *Ind Eng Chem Res*. 1997;36:2691–2700.
30. Karimi IA, McDonald CM. Planning and scheduling of parallel semicontinuous processes. II. Short-term scheduling. *Ind Eng Chem Res*. 1997;36:2701–2714.
31. Lamba N, Karimi IA. Scheduling parallel production lines with resource constraints. I. Model formulation. *Ind Eng Chem Res*. 2002; 41:779–789.
32. Lamba N, Karimi IA. Scheduling parallel production lines with resource constraints. II. Decomposition algorithm. *Ind Eng Chem Res*. 2002;41:790–800.
33. Reddy PCP, Karimi IA, Srinivasan R. A new continuous-time formulation for scheduling crude oil operations. *Chem Eng Sci*. 2004;59:1325–1341.
34. Sundaramoorthy A, Karimi IA. A simpler better slot-based continuous-time formulation for short-term scheduling in multipurpose batch plants. *Chem Eng Sci*. 2005;60:2679–2702.
35. Erdirik-Dogan M, Grossmann IE. Slot-based formulation for the short-term scheduling of multistage, multiproduct batch plants with sequence-dependent changeovers. *Ind Eng Chem Res*. 2008;47: 1159–1163.
36. Li J, Karimi IA, Srinivasan R. Recipe determination and scheduling of gasoline blending and distribution operations. *AIChE J*. DOI: 10.1002/aic.11970.
37. Zentner M, Pekny J, Reklaitis G, Gupta J. Practical considerations in using model-based optimization for the scheduling and planning of batch/semicontinuous processes. *J Process Control*. 1994;4:259–280.
38. Zhang X. Algorithms for Optimal Scheduling using Non-Linear Models. Ph.D. Thesis, London: University of London, 1995;20:897–904.
39. Zhang X, Sargent R. The optimal operation of mixed production facilities-a general formulation and some approaches for the solution. *Comput Chem Eng*. 1996;20:897–904.
40. Ierapetritou MG, Floudas CA. Effective continuous-time formulation for short-term scheduling. I. Multipurpose batch processes. *Ind Eng Chem Res*. 1998;37:4341–4359.
41. Maravelias CT, Grossmann IE. New general continuous-time state—task network formulation for short-term scheduling of multipurpose batch plants. *Ind Eng Chem Res*. 2003;42:3056–3074.
42. Castro PM, Barbosa P, voa AP, Matos HA, Novais AQ. Simple continuous-time formulation for short-term scheduling of batch and continuous processes. *Ind Eng Chem Res*. 2004;43:105–118.
43. Castro P, Novais A. Short-term scheduling of multistage batch plants with unlimited intermediate storage. *Ind Eng Chem Res*. 2008;47:6126–6139.
44. Castro P, Grossmann I. New continuous-time MILP model for the short-term scheduling of multistage batch plants. *Ind Eng Chem Res*. 2005;44:9175–9190.
45. Janak SL, Lin X, Floudas CA. Enhanced continuous-time unit-specific event-based formulation for short-term scheduling of multipurpose batch processes: resource constraints and mixed storage policies. *Ind Eng Chem Res*. 2004;43:2516–2533.
46. Shaik MA, Floudas CA. Unit-specific event-based continuous-time approach for short-term scheduling of batch plants using RTN framework. *Comput Chem Eng*. 2008;32:260–274.
47. Shaik MA, Janak SL, Floudas CA. Continuous-time models for short-term scheduling of multipurpose batch plants: a comparative study. *Ind Eng Chem Res*. 2006;45:6190–6209.
48. Castro PM, Barbosa-Povoa AP, Matos HA, Novais AQ. Simultaneous design and scheduling of multipurpose plants using resource task network based continuous-time formulations. *Ind Eng Chem Res*. 2005;44:343–357.
49. Gimenez DM, Henning GP, Maravelias CT. A novel network-based continuous-time representation for process scheduling. I. Main concepts and mathematical formulation. *Comput Chem Eng*. 2009;33: 1511–1528.
50. Gimenez DM, Henning GP, Maravelias CT. A novel network-based continuous-time representation for process scheduling. II. General framework. *Comput Chem Eng*. 2009;33:1511–1528.
51. Karimi I, Tan Z, Bhushan S. An improved formulation for scheduling an automated wet-etch station. *Comput Chem Eng*. 2004;29:217–224.
52. Brooke A, Kendrick D, Meeraus A. *GAMS: A User's Guide*. South San Francisco, CA: The Scientific Press, 1988.
53. Kondili E, Pantelides CC, Sargent RWH. A general algorithm for short-term scheduling of batch operations. I. MILP formulation. *Comput Chem Eng*. 1993;17:211–227.

Appendix

To reduce our model to enforce simultaneous material transfers into a batch, we set $\delta_{jk} = 0$ and eliminate θ_{jk}. This leads to the following modifications of Eqs. 7b, 8b, and 9b.

$$t_{j(k+1)} \geq t_{jk} + \sum_{i \in \mathbf{I}_j} [(\alpha_{ij} + \beta_{ij} B_{ij}^{\mathrm{L}}) ys_{ijk} + \beta_{ij} \Delta BI_{ijk}] - (T_{j(k+1)} - T_{jk})$$

$$1 \leq j \leq J,\ 0 \leq k < K \quad \text{(A1)}$$

$$T_{sk} \leq T_{jk} + H[1 - \sum_{i \in \mathbf{I}_j, \sigma_{sij} < 0} ys_{ijk}]$$

$$1 \leq j \leq J, s : \sum_{i \in \mathbf{I}_j, \sigma_{sij} < 0} \sigma_{sij} < 0, s : I_s^{\mathrm{U}} \text{ is limited}, 0 \leq k < K \quad \text{(A2)}$$

$$T_{sk} \geq T_{j(k-1)} + t_{j(k-1)} + \sum_{i \in \mathbf{I}_j} [(\alpha_{ij} + \beta_{ij} B_{ij}^{\mathrm{L}}) ys_{ij(k-1)} + \beta_{ij} \Delta BI_{ij(k-1)}] - H[1 - \sum_{i \in \mathbf{I}_j,\ \sigma_{sij} > 0} ye_{ijk}]$$

$$1 \leq j \leq J,\ s : \sum_{i \in \mathbf{I}_j, \sigma_{sij} > 0} \sigma_{sij} > 0,\ 1 \leq k < K, s : I_s^{\mathrm{U}} \text{ is limited} \quad \text{(A3)}$$

All other equations remain the same as in the original model.

Manuscript received Dec. 28, 2008, and revision received Oct. 8, 2009.