AIChE

A General Framework for Process Scheduling

Arul Sundaramoorthy and Christos T. Maravelias
Dept. of Chemical and Biological Engineering, University of Wisconsin, Madison, WI, 53706

DOI 10.1002/aic.12300
Published online July 22, 2010 in Wiley Online Library (wileyonlinelibrary.com).

Existing methods for process scheduling can be broadly classified as network-based or sequential. The former are used to address problems where different batches of the same or different tasks are freely mixed or split, whereas the latter are used to address problems where batch mixing/splitting is not allowed. A framework is proposed that allows us to: (1) express scheduling problems in facilities that consist of network and sequential, as well as continuous subsystems, (2) formulate mixed-integer programming (MIP) scheduling models for such problems, and (3) solve the resulting MIP formulations effectively. The proposed framework bridges the gap between network and sequential approaches, thereby addressing the major formulation challenge in the area of process scheduling, namely, the development of a framework that can be used to address a wide spectrum of problems. © 2010 American Institute of Chemical Engineers *AIChE J,* 57: 695–710, 2011
Keywords: process scheduling, mixed-integer programming

Introduction

The increasing product customization and diversification in the chemical industry have led to the installation (or retrofit) of facilities where multiple products compete for limited resources (equipment units and utilities) and which can be operated in multiple modes. The flexibility of these so called *multiproduct* facilities allows for higher resource utilization, lower inventory costs, and better responsiveness to demand fluctuations. Nevertheless, these advantages can only be materialized if the production is planned well, a task which is hard mainly due to the increased flexibility, and, thus, multiplicity of solutions.

To address this challenge, researchers in the area of process systems engineering (PSE) have developed a number of systematic scheduling approaches, typically mixed-integer programming (MIP) formulations. These approaches can be generally classified as: (1) network-based, used to address problems where batch mixing (blending) and splitting are allowed; and (2) sequential, or ordered-based, used to address problems where batch splitting and mixing are forbidden. Hence, there is no single approach that can be used to represent and address all scheduling problems. Furthermore, there is no method for the solution of problems in processes with different restrictions on batch splitting/mixing. Consequently, one of the two major outstanding challenges in the area of scheduling is the development of a framework capable of addressing all process scheduling problems. The second major challenge is the reduction of the computational requirements of scheduling methods. In this article, we address the former by developing a general strategy for process scheduling, which is based on the representation of network-based and sequential subsystems using a common formalism, and the formulation of a computationally effective MIP model for the unified problem.

The article is structured as follows. In the next section, we review scheduling problems and solution methods and present motivation for our work. In the following two sections, we present the key concepts of our strategy and the resulting MIP formulation, respectively. We then discuss modeling extensions and solution methods, and we close with a set of illustrative problems.

Background

Process and problem classification

Process scheduling problems can, in general, be classified in terms of material routing. If the output of multiple batches

Correspondence concerning this article should be addressed to C. T. Maravelias at Christos@engr.wisc.edu.

(of the same or different task) can be mixed to form the input of a subsequent batch or the output of a single batch can be split to be consumed by more than one batches, then we have a *network* process. On the other hand, if the output of a batch can be consumed only by a single batch and the input of a batch can only be the output of another batch, then we have a *sequential* process. Sequential multiproduct processes can be further classified as multistage (flexible flow-shop), if the sequence of the various operations is the same for all products; or multipurpose (flexible job-shop), if this sequence is different among products. Figure 1 shows the different types of processes, using a traditional convention for sequential processes, which, as we will see later, is not always valid.

While most chemical processes can be treated as networks, there are situations where batch integrity has to be preserved. For example, in biotech processing it is preferable to use a single seed batch for a single production batch to ensure that the seed culture is at a certain physiological state, minimize the chances of using out-of-specifications batches, and simplify logistics. Also, in recipe-based production it is often forbidden to blend batches between two steps, even though there is no technical restriction. Note that the no batch blending and splitting restriction between two tasks does not necessarily imply constant batch size, because materials that are not modeled can be removed (e.g., solids in solid-liquid separation after a fermentation), or added (e.g., solvent). Interestingly, facilities with such restrictions most often include continuous as well as network batch processes (e.g., downstream purification steps in a biotech facility), which means that they cannot be described in terms of the aforementioned traditional classification. The previous remark suggests that the classification should be made on the basis of specific tasks rather than entire facilities; i.e., a facility is divided into subsystems where sequential and network *processing* takes place. However, as we will discuss in the following subsection current scheduling methods do not consider this aspect.

Method classification and literature review

Approaches to process scheduling follow the aforementioned *dichotomy* between network and sequential processes. Network-based approaches were developed to address problems in network facilities, while sequential (or order-based) approaches were developed to address problems in sequential facilities.

Network-Based Approaches. In their seminal articles, Kondili et al.[1] and Shah et al.[2] proposed a general framework, the so called state-task network (STN) representation, for the scheduling in network processes. Pantelides[3] later introduced the resource-task-network (RTN) representation, where all the resources (units, vessels, materials, utilities, etc.) are treated uniformly. The two aforementioned approaches adopted a *discrete-time* representation, where the scheduling horizon was partitioned into a known number of periods of equal (and known) length. They also adopted a *common*, across all units, time grid to seamlessly account for material balance equations and shared utility requirements. Common discrete-time formulations have a number of advantages: (1) account linearly for inventory and backlog

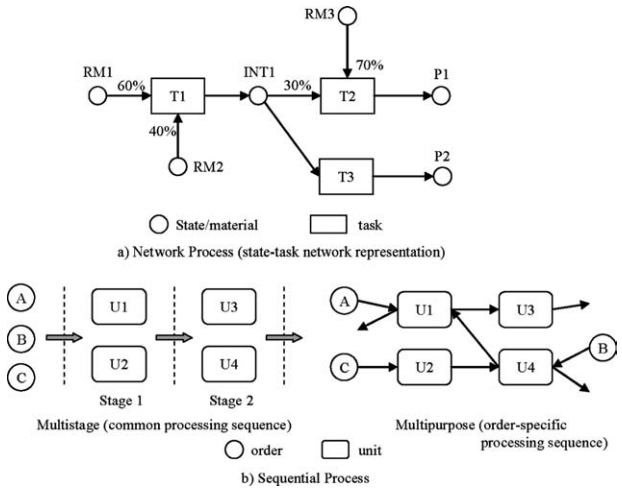

Figure 1. Network vs. sequential processes.

costs; (2) handle intermediate release and due dates at no additional computational cost, and (3) can be readily modified to model events taking place during the execution of a task, etc. However, since they may lead to large MIP formulations, several researchers developed a wide range of the so-called *continuous-time* representations, where the scheduling horizon is partitioned into a set of periods of unequal and unknown length and tasks have variable processing times.[4–10] Continuous-time formulations can be further classified into common and unit-specific time approaches. Continuous-time representations lead to smaller MIP formulations, which however are not easier to solve, due to large integrality gaps and symmetry of solutions. Finally, Maravelias[11] developed a *mixed-time* representation where the scheduling horizon is divided into periods of fixed length, but tasks are allowed to have variable processing times.

Sequential-Based Approaches. The key feature in sequential processes is that each order (or batch) moves through various production stages as a discrete entity, without splitting or mixing with other orders. To maintain such batch integrity, researchers proposed order-based (or batch-based) *sequential* MIP formulations with order-indexed decision variables for assignments to equipment units and precedence decisions. Compared to their network-based counterparts, order-based approaches are less broad: they do not typically consider storage and utility constraints, and they are based on the assumption that batching decisions are fixed. These assumptions were only recently relaxed by Maravelias and coworkers, who proposed models for the simultaneous batching and scheduling of sequential processes,[12–13] as well as models for sequential processes under storage and utility constraints.[14–15] There also exist some recent work on simultaneous batching and scheduling in single-stage sequential processes.[16] Finally, order-based approaches can be further classified based on the manner in which sequencing is performed. The two most popular methods are via (immediate or global) precedence variables[12–14,17–21] and time-grids, unit specific[22–25] and common.[15]

Figure 2 shows a schematic classification of the various scheduling approaches.[26] Note that these approaches can be further classified; e.g., precedence-based approaches can be

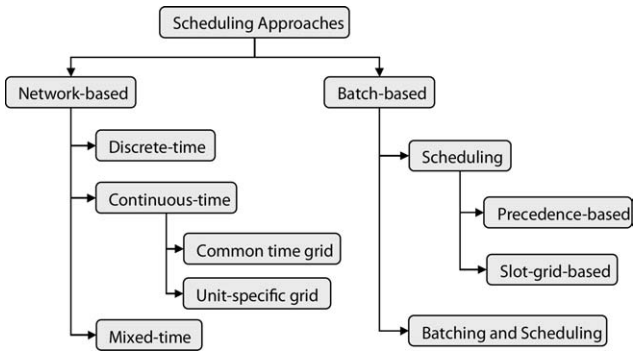

Figure 2. Classification of scheduling approaches (from Maravelias and Sung[26]).

divided into immediate and global precedence formulations. A comprehensive review of approaches to batch scheduling can be found in Mendez et al.[27]

Critical insights

One of the key concepts of network-based approaches is the definition of variables and the expression of material balance and resource constraints at each time point (period) of the grid. Interestingly, it is the satisfaction of material balance equations that also enforces feasible sequencing between tasks: if the states (or resources) consumed by one task are available, then this task can start, otherwise it cannot. Thus, network-based approaches have no constraints relating the execution of *consecutive* tasks; in fact, in the presence of recycle streams the notion of *consecutive* tasks is ill-defined. It is also important to note that network-based formulations employ variables for tasks, not specific batches of a task, and that there are no explicit constraints for the number of batches of *related* tasks. Therefore, we can loosely say that material balance constraints dictate the sequencing of tasks. As we will see in the next subsection, the outcome of this *material-centric* approach is that batch integrity cannot be maintained.

On the other hand, order-based approaches, rely heavily on the notion of an order going through the different processing stages. First, most approaches assume that the number of batches needed to satisfy demand is fixed, which means that the number of operations carried out in a schedule is known. Second, all variables are indexed by order (or batch), and all constraints are defined for orders and stages. It is important to note that this *order-centric* approach is also followed in RTN-based approaches that have been developed to address sequential problems.[24–25] In particular, orders are treated as resources consumed/produced by tasks, which means that the resource constraints essentially enforce a balance of orders through the stages. Interestingly, most order-based approaches do not employ material balance constraints. This is because the amount of material processed is considered fixed and already assigned to a set of batches. Thus, the only decisions to be made are the assignment of a set of orders (or batches) to units at each stage and the sequencing in the same unit subject to sequencing constraints between consecutive stages. Clearly, this approach cannot be used to address problems in network processes. Furthermore, as we show in the next subsection, this *order-centric* approach leads to unnecessarily large formulations.

To summarize, the key difference between approaches for network and sequential processes lies in the type of key entity that is modeled and conserved. Approaches for network processes employ materials (modeled either as states in STN or resources in RTN formulations), while approaches for sequential processes employ orders (or batches).

Limitations of existing approaches

Here we illustrate some of the limitations of the existing approaches and motivate the development of our strategy. First, consider the process shown in Figure 3a, where task R, carried out in unit U1, converts feed (F) to intermediate (I); task S, carried out in U2, converts intermediate into a final product (P); dedicated vessel V can be used for the storage of intermediate; the processing times of tasks R and S are equal to 2 h and 3 h, respectively; and minimum and maximum capacities are given in Figure 3a. The process appears to be a sequential one: each material is consumed or produced by a single task. However, enforcing that the amount of intermediate produced by a single batch of R should be consumed by a single batch of S using material balance equations of network-based formulations is impossible as shown in Figure 3b and c. In the solution shown in 3b, material balance for the intermediate is satisfied at $t = 2$ h and $t = 4$ h, but two batches of R are mixed and sent to unit U2 at $t = 4$ h. In the solution shown in Figure 3c, material balance for the intermediate is satisfied at $t = 2$ h and t = 5 h, but one batch of R is split and then processed in unit U2 as two batches at $t = 2$ h and $t = 5$ h, respectively. This simple example illustrates that traditional material balance equations used in network-based formulations are incapable of preventing batch mixing/splitting, and, therefore, network-based formulations employing material balance equations (either as

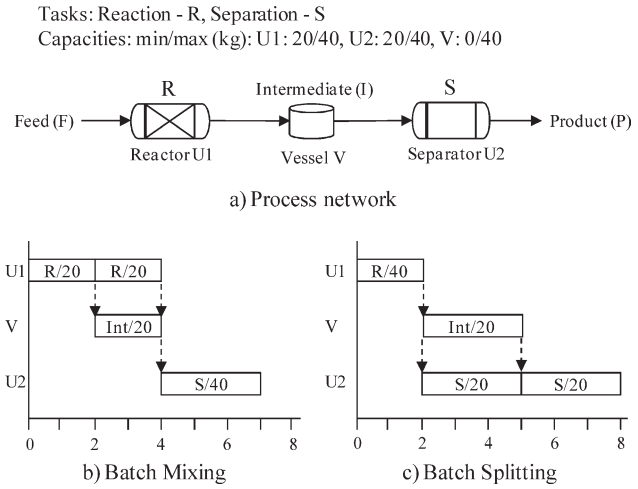

Figure 3. Motivating example.

Material balance constraints for intermediate (I) are satisfied but batches can be mixed or split. (b) $t = 2$: $S_{I,2} = S_{I,1} + B_{R,0} - B_{S,2} \Rightarrow 20 = 0 + 20$, satisfied; $t = 4$: $S_{I,4} = S_{I,3} + B_{R,2} - B_{S,4} \Rightarrow 0 = 20 + 20 - 40$, satisfied. (c) $t = 2$: $S_{I,2} = S_{I,1} + B_{R,0} - B_{S,2} \Rightarrow 20 = 0 + 40 - 20$, satisfied $t = 5$: $S_{I,5} = S_{I,4} - B_{S,5} \Rightarrow 0 = 20 - 20$, satisfied.

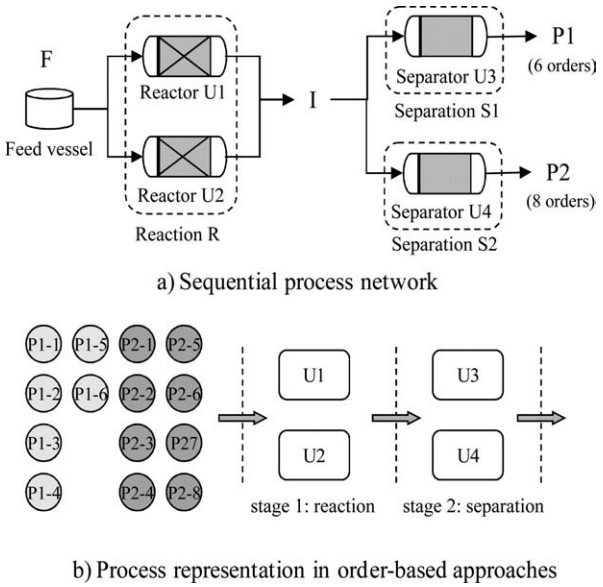

Figure 4. Representation of sequential processes.

inventory or resource balance) cannot be used to address sequential processing.

In addition to developing a framework that addresses all classes of problems, we also address a shortcoming of order-based approaches. To illustrate, we consider the process shown in Figure 4, where reaction R (carried out in U1 or U2) converts feed into an unstable intermediate I, which is in turn converted into products P1 or P2. An additional restriction is that each batch of R should be consumed by a single batch of task S1 (in U3) or S2 (in U4); i.e., the process is sequential. Let us further consider that we have a scheduling problem that involves six orders for P1 and eight orders for P2. To address this problem using one of the existing precedence-based approaches, we will have to introduce 14 orders to be processed in two stages, each one consisting of two units (see Figure 4b). The resulting MIP model would consist of assignment binary variables for 14 orders at 4 units and assignment constraints for 14 orders at 2 stages, and sequencing binaries and constraints for $14 \cdot 13 = 182$ order pairs. Note that the growth of the model is at least quadratic in the number of orders. A time-grid-based approach would employ roughly $14 \cdot 4 \cdot N$ assignment variables (where N is the number of time periods), and result in a formulation that grows linearly in the number of orders. Note that the growth in both cases is due to the manner in which orders are modeled, not due to the inherent complexity of the process, and for that matter, the complexity of the scheduling problem. This simple example illustrates a major limitation of order-based approaches, namely, the fast growth in the size of the models with the number of orders.

Proposed Approach

Our strategy is based on three major concepts. The first is the classification of the subsystems of a process into network and sequential subsystems, and the subsequent grouping of the tasks and states into different subsets to be modeled differently. The second is the unified representation of a process, comprising network and/or sequential subsystems. This is achieved by expressing sequential subsystems using a material-based (network-like) formalism with special features. The third is the development of novel constraints for the enforcement of batch integrity in sequential subsystems. The three aforementioned concepts along with the time representation we employ are discussed in the following subsections, while the MIP formulation that we develop based on these concepts is presented in the next section.

Subsystem classification

A given facility can have batch network and sequential, as well as continuous processes. For example, it is common to have a process that starts with sequential batch processing, where batch integrity must be preserved for quality control purposes, followed by network batch processing, typically to produce similar products using the same base material, but different additives (mixing), followed by continuous processing for packing. Currently, the subsystems of such facility will have to be scheduled separately, leading to suboptimal solutions. To develop a framework capable of addressing the facility as a whole, we should be able to represent all these subsystems using the same formalism. In achieving this, we generalize the well-known notions of states and tasks, which have been used only for network processes, to represent all subsystems. This implies that sequential processes are represented using states and tasks, an element of our strategy discussed in the next subsection. For now, we assume that the entire process can be grouped into network and sequential subsystems, all of which consist of tasks and states with different properties (see Figure 5). Continuous subsystems will be discussed later. Note that tasks always belong to a subsystem, whereas states connecting different subsystems do not belong to either subsystem. Also, final products are interfaced with customers, which as we will see later can be modeled as additional units.

Once we have the classification into subsystems, we characterize states, $s \in \mathbf{S}$, and tasks, $i \in \mathbf{I}$, of the process as follows:

(a) States are grouped into three types (see Figure 6a): (1) network states, $s \in \mathbf{S}^N$; (2) sequential states, $s \in \mathbf{S}^Q$; and (3) hybrid states, $s \in \mathbf{S}^H$. Network states are consumed and produced within a network subsystem; that is, by processing tasks in a network subsystem. Similarly, sequential states are

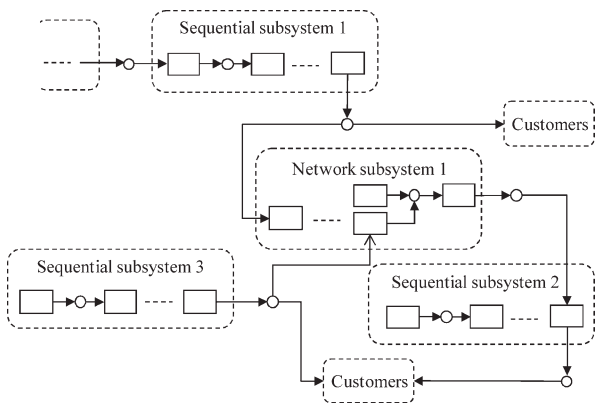

Figure 5. Subsystem classification.

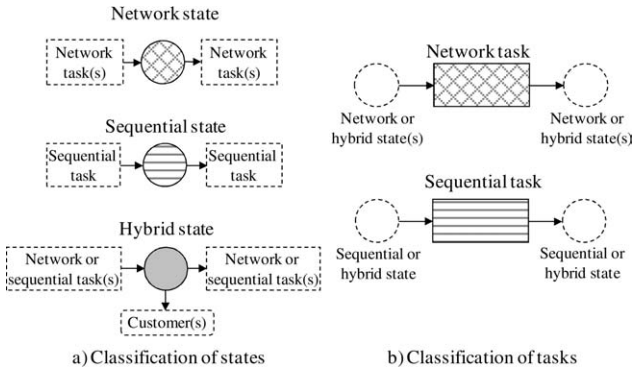

Figure 6. Proposed classification of states and tasks.

consumed and produced by tasks that belong to the same sequential subsystem. Hybrid states are states used to interface different subsystems; they can be either intermediate materials linking different subsystems, or finished materials connecting processes to customers.

(b) There are two subsets of processing tasks (see Figure 6b): (1) network tasks $i \in \mathbf{I}^N$, and (2) sequential tasks $i \in \mathbf{I}^Q$, where $\mathbf{I} = \mathbf{I}^N \cup \mathbf{I}^Q$. Network tasks consume and produce only network or hybrid states, whereas sequential tasks consume and produce only sequential or hybrid states.

Finally, we group equipment units into processing units, $j \in \mathbf{J}^P$, and storage vessels, $j \in \mathbf{J}^V$, and we introduce pseudo-units, $j \in \mathbf{J}^C$, to model customers. Note that units do not belong to a subsystem: they can be used to carry out tasks or store states belonging to different subsystems, although this assumption can be relaxed.

Material-based representation of sequential processes

In the previous subsection we assumed that sequential subsystems can be represented using tasks and states, as opposed to orders (batches) and stages that have been traditionally used in existing approaches. We now illustrate how this is achieved. We assume that a sequential subsystem is given in terms of products (which can be final products or hybrid states), and a sequence of operations. The generation of this network-like representation of such subsystem follows three steps as listed bellow:

Step 1. For each product and operation in a stage, we define a task; if two products share the same operation in one stage, then we define a single task, which means that a task may belong in the *production path* of multiple products, as is often the case.

Step 2. For each different material consumed or produced in the subsystem, we define a state. Note that if an intermediate is shared among multiple products, only one state is defined, which means that a state can be in the *path* of multiple products.

Step 3. Generate a network representation by connecting all states to the tasks consuming/producing them, where we assume that the same state cannot be produced by two tasks consuming different states (this assumption can be easily relaxed).

Based on the aforementioned procedure we obtain a network where each task has a single input/output state, and each state has a single producing task but can have multiple consuming tasks. To illustrate, we consider the sequential process introduced in Figure 4, which consists of a single operation in stage 1 (reaction R) and two operations in stage 2 (separations S1 and S2), producing two final products, P1 and P2. Since the operation in the first stage is the same for both products, we need to introduce a single "feed" state, RM, a single task, RXN, in the first stage, and a single intermediate state, INT (see Figure 7). Of course, we will need additional constraints to ensure that no batch mixing or splitting occurs for intermediate INT. This is discussed in the next subsection.

It is interesting to note here that the classification of a subsystem as network or sequential does not depend on the number of tasks producing/consuming a state. For example, in the process shown in Figure 3a, state I is produced and consumed by a single task, but the process is a *network* one because a batch produced in U1 can be split between two batches in U2 (Figure 3b), or two batches from U1 can be mixed toward a single batch in U2 (Figure 3c). On the contrary, the process in Figures 4 and 7 is *sequential* even though INT can be consumed by two tasks. In other words, the classification of a process is based on the specific restrictions on the way we treat batch input/output and not the structure of the facility, as it has often been assumed in the literature.

Before we present how batch mixing and splitting is avoided, it is worthwhile to discuss how the proposed representation can reduce model size for a sequential facility. We consider the process shown in Figures 4 and 7, with a total of 14 single-batch orders. As we have already seen, existing precedence-based approaches result in formulations of O(orders^2 · stages), while existing time-grid-based approaches will result in formulations O(orders · stages · periods). As we will see in the next section, the majority of variables and constraints of the proposed formulation are defined for tasks, units, states, and time periods. Therefore, the formulation size depends more on the complexity of the actual process (number of units, materials and different processing tasks) rather than the number of orders in a specific instance.

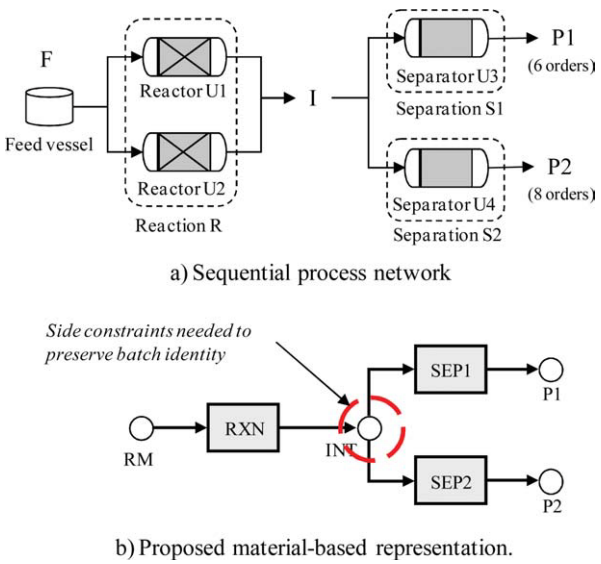

Figure 7. Proposed material-based representation of sequential processes.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

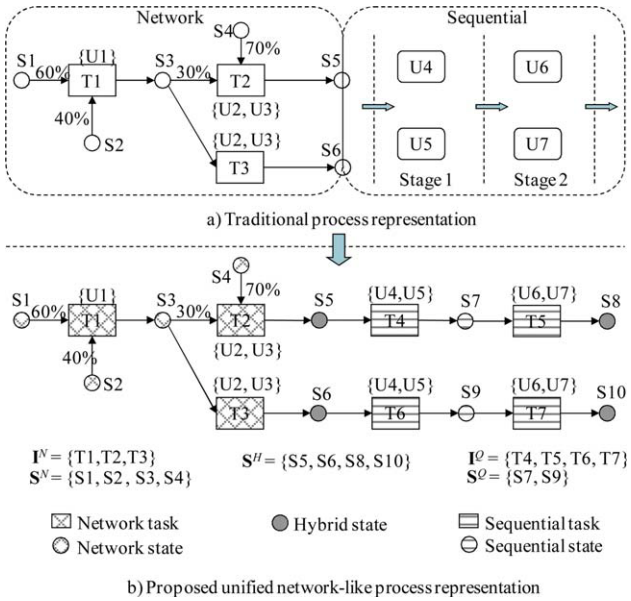

Figure 8. Illustration of proposed representation.

Shown process consists of a network (upstream) and sequential (downstream) subsystems. The former involves three tasks and six states (including S5 and S6), while the latter is represented as a network with four tasks and six states (including S5 and S6). Since S5 and S6 are in the interface of the two subsystems they are treated as hybrid states (see previous subsection), along with states S8 and S10 that correspond to final products. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

Thus, the proposed approach can lead to smaller formulations, especially when the number of orders per product becomes large. Since this number increases as we consider longer scheduling horizons, this means that the advantage of the proposed formulation becomes more pronounced as the problem becomes harder. This reduction will not be substantial in facilities where multiple products are produced and there are few orders per product. Even in these facilities, however, small reductions are possible if multiple products have some common upstream operations and are differentiated in the last few processing steps, a situation that is quite common in many sectors. The reduction in this case comes from the fact that the proposed approach employs materials (that can be common among multiple products over a few stages) rather than orders that are distinct from the first to the last stage (see Example 2).

In summary, the advantages of our representation are twofold: (1) facilitation of unified representation of general facilities which allows us to address all process scheduling problems; and (2) development of models for sequential processes that are material-centric rather than order-centric, which in turn can potentially lead to reduction in the model size. Figure 8 illustrates how a process with network and sequential subsystems can be represented as a unified network.

Batch integrity

We employ two modeling concepts to preserve batch identity around states in sequential subsystems: (1) explicitly account for connections between units and vessels,[28] and (2) disaggregate inventory variables.[15] Given is a set of processing units, $j \in \mathbf{J}^P$, and a set of storage vessels $j \in \mathbf{J}^V$. If used to perform a sequential task, then a unit can be connected to only one unit in the following operation or only one storage vessel. Similarly, a unit can be connected to only one unit or vessel from the previous operation. To achieve this, we use subsets $\mathbf{JC}_j^-/\mathbf{JC}_j^+$ of units or vessels from/to which a batch can be transferred to/from unit or vessel j. Binary variable $W_{jj't}$ denotes a connection between units/vessels j and $j' \in \mathbf{JC}_j^+$ at time point t. As a result, there exists a flow $F_{sjj't}$ of a batch of material s from j to j' at the same time point (see Figure 9).

Next, we decouple the inventory variable S_{sjt} into two inventory variables, one for the inventory level of state s in unit j just before the time point t, S_{sjt}^-; and one for inventory just after the time point t, S_{sjt}^+. Such decoupling is needed because the traditional inventory balance constraint

$$S_{sjt} = S_{sj(t-1)} + \sum_{j' \in \mathbf{JC}_j^-} F_{sj'jt} - \sum_{j' \in \mathbf{JC}_j^+} F_{sjj't} \quad \forall s, \quad j \in \mathbf{J}^V, t \quad (1)$$

cannot prevent batch mixing when two batches of the same order are stored in a vessel immediately one after the other as illustrated in Figure 3 (see also Figure 10a). To avoid such interaction between batches of the same order, we express two material balance constraints using the decoupled inventory variables (see Figure 10b)

$$S_{sjt}^- = S_{sj(t-1)}^+ - \sum_{j' \in \mathbf{JC}_j^+} F_{sjj't} \quad \forall s, \quad j \in \mathbf{J}^V, t \quad (1a)$$

$$S_{sjt}^+ = S_{sjt}^- + \sum_{j' \in \mathbf{JC}_j^-} F_{sj'jt} \quad \forall s, \quad j \in \mathbf{J}^V, t \quad (1b)$$

This ensures that the flow $F_{sjj't}$ from storage vessel j occurs just before time point t, while the flow $F_{sj'jt}$ into storage vessel j occurs just after time point t. Figure 10 illustrates how the proposed modeling does not allow batch mixing and splitting in the process shown in Figure 3.

Finally, note that we use the term *divide* to describe the batching decision (i.e., an order is divided into batches), and the term *split* to describe the situation where a batch is

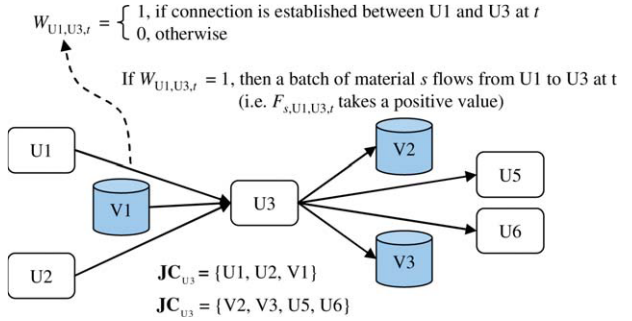

Figure 9. Illustration of unit connectivity.

Set $\mathbf{JC}_{U3}^-$ includes the units/vessels from which a batch can be transferred to U3, while $\mathbf{JC}_{U3}^+$ includes the units/vessels to which a batch can be transferred from U3. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

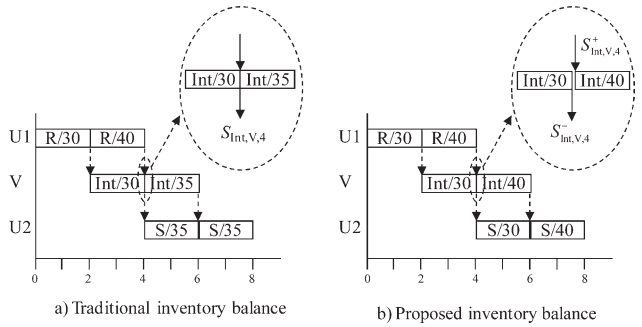

Figure 10. Effectiveness of proposed material balances.

Two batches of R with sizes of 30 kg and 40 kg are required to meet the total demand of 70 kg. The batches share vessel V sequentially (batch 1: t = 2-4; batch 2: t = 4-6). At $t = 4$, batch 1 flows from V to unit U2, whereas batch 2 flows from unit U1 to V. (a) If a single inventory balance is used, batches can *exchange* material, and (b) with decoupled flow-out and flow-in mixing is avoided.

shared by more than one task, which is not allowed in sequential processes.

Time representation

Regardless of the process structure, a common time reference grid is necessary across all units and vessels in order to enforce utility constraints. In this article, we use a discrete-time framework. The time horizon is divided into a number of T periods defined by a set of time points $t \in \mathbf{T} = \{0, 1, 2, \ldots, T\}$. Period t starts (ends) at time point $t - 1$ (t), whose length, δ, is fixed. Furthermore, as in any discrete-time approach, we scale time-related data using δ and convert them to the time interval basis: for example, $\tau_{ij} = \lceil \bar{\tau}_{ij}/\delta \rceil$, $\varepsilon_s = \lceil \bar{\varepsilon}_s/\delta \rceil$ and $\varphi_{sj} = \lfloor \bar{\varphi}_{sj}/\delta \rfloor$ where, $\bar{\tau}_{ij}$, $\bar{\varepsilon}_s$ and $\bar{\varphi}_{sj}$ are the given processing, release and due times, respectively. Note that these parameters are approximated by either rounding up or down to the nearest integer values in such a way that the resulting optimal solution is feasible.

It is important to note that the novelty of the proposed method lies in the development of the three concepts presented in this section. The selection of the time representation is a secondary issue. In fact, the proposed framework can be trivially extended to continuous-time representations. The reason we selected a discrete-time representation is because we have recently found it to be superior to its continuous-time counterparts for a wide range of problems.

Mathematical Formulation

Given is a general process consisting of network, as well as sequential subsystems, the production recipes and operation sequences for network and sequential subsystems, utility requirements and availabilities, demands and prices of products, release and due times, and production costs (processing, storage, and utility). We use the term task to denote a type of operation, and the term batch to denote a specific instance of this task in a schedule; i.e., a schedule can include multiple batches of the same task.

Our goal is to determine:

(a) the number and sizes of batches;

(b) the assignment of batches to processing units and states to storage vessels, and

(c) the sequencing and timing of assigned batches in each unit and vessel;

so as to optimize an objective that can be either an economic (profit, cost, sales, etc.) or time-related (makespan, earliness, lateness, etc.) metric.

Our formulation includes the following time-indexed binary decisions

(a) Batch assignment: $X^P_{ijt}/X^V_{sjt} = 1$ if task i or state s is assigned to unit/vessel j;

(b) Resource consumption: $R^P_{jt}/R^V_{jt} = 1$ if unit/vessel j is consumed at time point t;

(c) Connectivity: $W_{jj't} = 1$ if a connection between unit/vessel j and unit/vessel/customer j' exists;

(d) Delivery: $Z_{sjt} = 1$ if product s is delivered to customer $j \in \mathbf{J}^C$ at time point t.

We also use the following nonnegative continuous variables:

(e) $F_{sjj't}$ denotes the flow of state s from unit/vessel j to another unit/vessel/customer j'.

(f) S_{st} represents the inventory of state $s \in \mathbf{S}^N$ at time point t; S^-_{sjt}/S^+_{sjt} represents the inventory of state $s \in \mathbf{S}^Q \cup \mathbf{S}^H$ in vessel j just before/after time point t.

(g) N_{ut} represents the consumption level of utility u.

(h) B_{ijt} denotes the batch size of task i assigned to start on unit j at time point t.

We use uppercase-bold letters for sets, lowercase-italic for indices, lower case-Greek-italic for parameters, and uppercase-italic for variables.

Common constraints

Resource balance constraints for processing units and storage vessels are expressed as follows

$$R^P_{jt} = R^P_{j(t-1)} - \sum_{i \in \mathbf{I}_j} X^P_{ij(t-\tau_{ij})} + \sum_{i \in \mathbf{I}_j} X^P_{ijt} \quad \forall j \in \mathbf{J}^P, t \qquad (2)$$

$$R^V_{jt} = R^V_{j(t-1)} - \sum_{s \in \mathbf{S}^V_j} X^V_{sj(t-1)} + \sum_{s \in \mathbf{S}^V_j} X^V_{sjt} \quad \forall j \in \mathbf{J}^V, t \qquad (3)$$

where $i \in \mathbf{I}_j$ is the subset of tasks that can be carried out in unit j, and $\mathbf{S}^V_j$ is the set of states that can be stored in vessel j. We assume that storage can be discontinued at any time; i.e., $\tau = 1$.

If ξ_{iju}/ψ_{iju} denotes the fixed/variable requirement of task i in unit j for utility u, then the total utility consumption N_{ut} must satisfy availability $\theta^{\max}_u$:

$$N_{ut} = N_{u(t-1)} - \sum_{i,j \in \mathbf{J}^P_i} \left(\xi_{iju} X^P_{ij(t-\tau_{ij})} + \psi_{iju} B_{ij(t-\tau_{ij})}\right) + \sum_{i,j \in \mathbf{J}^P_i} \left(\xi_{iju} X^P_{ijt} + \psi_{iju} B_{ijt}\right) \le \theta^{\max}_u \quad \forall u, t \qquad (4)$$

where $\mathbf{J}^P_i$ is the set of processing units task i can be assigned to.

Material balances

Network States. If $\rho_{is}^{+}/\rho_{is}^{-}$ denotes the fraction of state $s \in \mathbf{S}^N$ produced/consumed by task $i \in \mathbf{I}_s^{+}/\mathbf{I}_s^{-}$, then the inventory at time point t must be equal to its inventory at the previous time point plus the total amount produced minus the total amount consumed by subsequent tasks and shipments (demand satisfaction), and should be within the maximum capacity $\gamma_s^{\max}$

$$S_{st} = S_{s(t-1)} + \sum_{i \in \mathbf{I}_s^{+}} \rho_{is}^{+} \sum_{j \in \mathbf{J}_i^{P}} B_{ij(t-\tau_{ij})} - \sum_{i \in \mathbf{I}_s^{-}} \rho_{is}^{-} \sum_{j \in \mathbf{J}_i^{P}} B_{ijt} - \sum_{j \in \mathbf{J}_s^{C}} D_{sjt} \leq \gamma_s^{\max} \quad \forall s \in \mathbf{S}^N, t \quad (5N)$$

where $\mathbf{J}_s^{C}$ is the set of customers associated with state s and D_{sjt} is the shipment of s to customer $j \in \mathbf{J}_s^{C}$ at time point t.

Sequential States. The inventory variable of sequential states is disaggregated: S_{sjt}^{-} and S_{sjt}^{+} denote the inventory levels of s in j just before and after, respectively, time point t

$$S_{sjt}^{-} = S_{sj(t-1)}^{+} - \sum_{j' \in \mathbf{JC}_j^{+} \cap \mathbf{J}_s^{-}} F_{sjj't} \quad \forall s \in \mathbf{S}^Q, j \in \mathbf{J}_s^{V}, t \quad (5Sa)$$

$$S_{sjt}^{+} = S_{sjt}^{-} + \sum_{j' \in \mathbf{JC}_j^{-} \cap \mathbf{J}_s^{+}} F_{sj'jt} \quad \forall s \in \mathbf{S}^Q, j \in \mathbf{J}_s^{V}, t \quad (5Sb)$$

where $\mathbf{J}_s^{-}/\mathbf{J}_s^{+}$ consists of the units that can consume/produce state s, vessels that store s, and customers that require state s. While vessels that store state s are included in both $\mathbf{J}_s^{+}$ and $\mathbf{J}_s^{-}$, customers that require state s are included only in $\mathbf{J}_s^{-}$. The former disaggregated variable is equal to the inventory just after t-1 minus the flow out of the vessel at t (Eq. 5Sa), while the latter is equal to the inventory just before t plus the flow into the vessel at t (Eq. 5Sb). Note that the flow terms in Eqs. 5Sa and 5Sb replace the batch-size terms in Eq. 5N, because we introduced flows to preserve batch integrity in sequential subsystems. Also, note that $S_{sjt}^{-} = S_{sjt}^{+}$ whenever the storage of s is continued in j.

Hybrid States. Hybrid states interface different subsystems (network to sequential and/or sequential to network), as well as processing subsystems and customers. Since the batch identity for a subset of hybrid states has to be preserved (e.g., states produced by sequential tasks and delivered to customers as a whole batch), we use the disaggregation of inventory variables (see Figure 10). Furthermore, since a hybrid state can be connected to network and sequential subsystems, both flow and batch-size terms should be included

$$\sum_{j \in \mathbf{J}_s^{V}} S_{sjt}^{-} = \sum_{j \in \mathbf{J}_s^{V}} S_{sj(t-1)}^{+} - \sum_{i \in \mathbf{I}_s^{-}} \rho_{is}^{-} \sum_{j \in \mathbf{J}_i^{P}} B_{ijt} - \sum_{j \in \mathbf{J}_s^{V}} \sum_{j' \in \mathbf{JC}_j^{+} \cap \mathbf{J}_s^{-}} F_{sjj't} \quad \forall s \in \mathbf{S}^H, t \quad (5Ha)$$

$$\sum_{j \in \mathbf{J}_s^{V}} S_{sjt}^{+} = \sum_{j \in \mathbf{J}_s^{V}} S_{sjt}^{-} + \sum_{i \in \mathbf{I}_s^{+}} \rho_{is}^{+} \sum_{j \in \mathbf{J}_i^{P}} B_{ij(t-\tau_{ij})} + \sum_{j \in \mathbf{J}_s^{V}} \sum_{j' \in \mathbf{JC}_j^{-} \cap \mathbf{J}_s^{+}} F_{sj'jt} \quad \forall s \in \mathbf{S}^H, t \quad (5Hb)$$

However, unlike Eqs. 5Sa and 5Sb, we sum over all the corresponding storage vessels to avoid the addition of the same amount of material to more than one vessel.

For network states we assume that we have dedicated storage vessels, therefore, we check only storage capacity in Eq. 5N. However, this assumption can be trivially relaxed to account for storage of network states to specific vessels in a manner similar to sequential and hybrid states without the decoupling of inventory variables.

Sequential subsystems

Equation 6 enforces that a sequential state $s \in \mathbf{S}^Q$ just produced or currently stored, will either be consumed or continue to be stored

$$\sum_{j \in \mathbf{J}_s^{V}} X_{sj(t-1)}^{V} + \sum_{i \in \mathbf{I}_s^{+},} \sum_{j \in \mathbf{J}_i^{P}} X_{ij(t-\tau_{ij})}^{P} = \sum_{i' \in \mathbf{I}_s^{-}, j \in \mathbf{J}_{i'}^{P}} X_{i'jt}^{P} + \sum_{j \in \mathbf{J}_s^{V}} X_{sjt}^{V} \quad \forall s \in \mathbf{S}^Q, t \quad (6)$$

Furthermore, Eq. 6 ensures that the number of batches producing a state $s \in \mathbf{S}^Q$ is the same as the number of batches consuming it, thus enforcing a batch balance constraint.

Whenever a task $i \in \mathbf{I}^Q$ is completed in unit $j \in \mathbf{J}^P$, the unit must be connected to another unit $j' \in \mathbf{JC}_j^{+}$

$$\sum_{i \in \mathbf{I}_j^{Q}} X_{ij(t-\tau_{ij})}^{P} = \sum_{j' \in \mathbf{JC}_j^{+}} W_{jj't} \quad \forall j \in \mathbf{J}^P, t \quad (7)$$

Similarly, a sequential task can start on unit j, only if another unit/vessel $j' \in \mathbf{JC}_j^{-}$ is connected to unit j

$$\sum_{i \in \mathbf{I}_j^{Q}} X_{ijt}^{P} = \sum_{j' \in \mathbf{JC}_j^{-}} W_{j'jt} \quad \forall j \in \mathbf{J}^P, t \quad (8)$$

If a sequential task $i \in \mathbf{I}^Q$ is completed on unit j at time point t, then state s produced by i must flow to a unit/vessel $j' \in \mathbf{JC}_j^{+}$:

$$\sum_{i \in \mathbf{I}_j^{Q} \cap \mathbf{I}_s^{+}} B_{ij(t-\tau_{ij})} = \sum_{j' \in \mathbf{JC}_j^{+} \cap \mathbf{J}_s^{-}} F_{sjj't} \quad \forall s, \quad j \in \mathbf{J}_s^{+} \cap \mathbf{J}^P, t \quad (9)$$

Similarly, if task i is started on j at t, then it must have consumed state s flowed from unit/vessel $j' \in \mathbf{JC}_j^{-}$:

$$\sum_{i \in \mathbf{I}_j^{Q} \cap \mathbf{I}_s^{-}} B_{ijt} = \sum_{j' \in \mathbf{JC}_j^{-} \cap \mathbf{J}_s^{+}} F_{sj'jt} \quad \forall s, \quad j \in \mathbf{J}_s^{-} \cap \mathbf{J}^P, t \quad (10)$$

Note that batch sizes of two consecutive sequential operations may vary, in which case the batch integrity can be enforced based on a key material (e.g., active pharmaceutical ingredient), using the conversion coefficients $\rho_{is}^{+}/\rho_{is}^{-}$ in Eqs. 9 and 10 in a manner similar to network tasks.

To prevent mixing of batches, vessels storing sequential states must be emptied whenever material of a sequential state flows out (i.e., the corresponding batch must be fully transferred). Similarly, the material of a sequential state has to be transferred only to an empty vessel

$$\sum_{s\in\mathbf{S}^Q} S^-_{sjt} \le \beta^{\max}_j \left(1 - \sum_{j'\in\mathbf{JC}^-_j} W_{j'jt} - \sum_{j'\in\mathbf{JC}^+_j} W_{jj't}\right) \quad \forall j\in\mathbf{J}^V, t \quad (11)$$

Note that Eq. 11 is enforced only to states whose batch integrity has to be ensured. Removal of first and second summation terms on the RHS of Eq. 11 allows batch mixing and splitting in vessels, respectively, as discussed later.

Order Delivery

To ensure that orders for the same final product are not mixed, we model orders using customers as units. In particular, each order for a state $s \in \mathbf{S}^F$ (set of final products) is associated with a pseudo-customer $j \in \mathbf{J}^C$. A customer can have many orders, provided that they correspond to distinct states. Thus, the number of customers is equal to the maximum number of orders for the same product; e.g., if five products have two orders each, then we only use two pseudo-customers. Using these state-customer (s-j) pairings to represent an order, Eq. 12 ensures that all orders are delivered only once

$$\sum_t Z_{sjt} = 1 \quad \forall s\in\mathbf{S}^F, \quad j\in\mathbf{J}^C_s \quad (12)$$

If a single order includes multiple states due on the same date, then we treat it as different state-customer pairs with the same customer and due date; and add constraints to enforce that all these orders are delivered at the same time.

Depending on the subsystem that produces the final products, there exist different cases of order delivery to customers:

(a) The final product is produced by network task(s); i.e., different batches can be mixed in the storage vessel, and/or the same batch can be used to satisfy multiple orders.

(b) The final product is produced by a sequential task, where we assume that different batches of finished product can share the storage vessel, but the same batch cannot be used to satisfy multiple orders.

In case (a), the delivery of orders to customers is enabled by allowing the corresponding delivery variable D_{sjt} in Eq. 5N to be nonzero

$$\omega_{sj} Z_{sjt} \le D_{sjt} \le \omega^{\max}_{sj} Z_{sjt} \quad \forall s\in\mathbf{S}^F, \quad j\in\mathbf{J}^C_s, t \quad (13a)$$

where ω_{sj} is the demand for order (s,j), and $\omega^{\max}_{sj}$ is an upper bound on the amount of material that can be delivered.

In case (b), demand satisfaction is enforced through flow variables

$$\omega_{sj} Z_{sjt} \le \sum_{j'\in\mathbf{JC}^-_j\cap\mathbf{J}^+_s} F_{sj'jt} \le \omega^{\max}_{sj} Z_{sjt} \quad \forall s\in\mathbf{S}^F, \quad j\in\mathbf{J}^C_s, t \quad (13b)$$

Furthermore, to allow batch mixing, but forbid batch splitting, we use Eq. 11a instead of Eq. 11:

$$\sum_{s\in\mathbf{S}^H} S^-_{sjt} \le \beta^{\max}_j \left(1 - \sum_{j'\in\mathbf{JC}^+_j} W_{jj't}\right) \quad \forall j\in\mathbf{J}^V, t \quad (11a)$$

Eq. 14 can be used to tighten the formulation

$$\sum_{j'\in\mathbf{J}^C} W_{jj't} \le 1 \quad \forall j\in\mathbf{J}^V, t \quad (14)$$

Finally, a strict restriction on batch integrity that no mixing of batches of the same final product in storage vessels, although less common in practice, can also be achieved by expressing Eq. 11 for these states instead of Eq. 11a.

Additional constraints

The amount of state s that flows from j to j' must satisfy the maximum capacity of both

$$\sum_{s;j\in\mathbf{J}^+_s, j'\in\mathbf{J}^-_s} F_{sjj't} \le \min(\beta^{\max}_j, \beta^{\max}_{j'}) W_{jj't} \quad \forall j, \quad j'\in\mathbf{JC}^+_j, t \quad (15)$$

If task i is assigned to processing unit j, then its batch size must satisfy

$$\beta^{\min}_j X^P_{ijt} \le B_{ijt} \le \beta^{\max}_j X^P_{ijt} \quad \forall i, \quad j\in\mathbf{J}^P_i, t \quad (16)$$

The inventories of sequential/hybrid states must be bounded by the maximum $\beta^{\max}_j$ size of the vessel

$$\begin{aligned} S^+_{sjt} &\le \beta^{\max}_j X^V_{sjt} \quad \forall s\in\mathbf{S}^Q\cup\mathbf{S}^H, \quad j\in\mathbf{J}^V_s, t \\ S^-_{sjt} &\le \beta^{\max}_j X^V_{sjt} \quad \forall s\in\mathbf{S}^Q\cup\mathbf{S}^H, \quad j\in\mathbf{J}^V_s, t \end{aligned} \quad (17)$$

Note that since the vessels are emptied during the transfer of sequential/hybrid states so as to ensure batch integrity, we do not enforce that the inventory be at least a minimum. However, for network states, minimum inventories can be enforced.

Forbidden assignments can be enforced by fixing the corresponding assignment variables to zero, and forbidden paths between any two units can also be enforced by fixing the corresponding connectivity variables. The release ε_s and due ϕ_{sj} times can be enforced by fixing the assignment variables to zero before the release times and by fixing the delivery variables to zero after the due times. Furthermore, sequence-dependent changeover times can be readily modeled as given in Shah et al.[2]

Finally, Eq. 18 expresses integrality and non-negativity constraints

$$\begin{aligned} &R^P_{jt}, R^V_{jt}, W_{jj't}, X^P_{ijt}, X^V_{sjt}, Z_{sjt} \in \{0,1\}; \\ &B_{ijt}, F_{sjj't}, N_{ut}, S_{st}, S^-_{sjt}, S^+_{sjt} \ge 0 \end{aligned} \quad (18)$$

The proposed MIP formulation consists of Eqs. 2 to 18.

Objective function

The proposed model can accommodate various cost- and time-related performance measures.

Profit Maximization/Cost Minimization. The maximization of profit, which is total revenue minus the total costs, is given by

$$\max \sum_{s\in\mathbf{S}^F, j'\in\mathbf{J}^C_s, t} \left(\pi_{sj'} D_{sj't} + \sum_{j\in\mathbf{JC}^-_{j'}} \pi_{sj'} F_{sjj't}\right) - \left(\sum_{i,j\in\mathbf{J}^P_i, t} \alpha_{ij} X^P_{ijt} + \sum_{s,j\in\mathbf{J}^V_s, t} \chi_{sj} X^V_{sjt} + \sum_{u,t} \zeta_u N_{ut}\right) \quad (19)$$

where π_{sj} is the price of state s for customer j; α_{ij} and χ_{sj} are fixed processing and storage costs, respectively; and ζ_u is per unit utility cost. Equation 19 can be easily modified to account for variable costs, as well as holding and backlog costs.

Makespan Minimization. The minimization of makespan (*MS*), is given by

$$\min MS \tag{20}$$

where

$$MS \geq \sum_t tZ_{sjt} \quad \forall s \in \mathbf{S}^F, \quad j \in \mathbf{J}_s^C \tag{21}$$

Total Earliness/Tardiness Minimization. The minimization of total earliness and total tardiness are given by Eqs. 22 and 23, respectively

$$\min \sum_{s \in \mathbf{S}^F, j \in \mathbf{J}_s^C} T_{sj}^E \tag{22}$$

$$\min \sum_{s \in \mathbf{S}^F, j \in \mathbf{J}_s^C} T_{sj}^R \tag{23}$$

where T_{sj}^E/T_{sj}^R denote the earliness/tardiness for order (s,j)

$$T_{sj}^E \geq \sum_{t \leq \varphi_{sj}} X_{sj't}^V + \left(\varphi_{sj} - \sum_{t \leq \varphi_{sj}} tZ_{sjt} \right) \quad \forall s \in \mathbf{S}^F, \quad j \in \mathbf{J}_s^C, \quad j' \in \mathbf{J}_s^V \tag{24}$$

$$T_{sj}^R \geq \sum_{t > \varphi_{sj}} (t - \varphi_{sj}) Z_{sjt} \quad \forall s \in \mathbf{S}^F, \quad j \in \mathbf{J}_s^C \tag{25}$$

In case of earliness minimization, some orders may be finished earlier than the actual delivery times, in which case inventory of the finished orders build up. To avoid such cases, we account for the time the finished order is stored in vessels before the delivery to customers as shown in Eq. 24.

Remarks

In the subsection entitled *Order delivery*, we started discussing how different material routing restrictions can be enforced (e.g., batch blending, but no splitting allowed for final products). In fact, our framework allows us to model all possible restrictions by simply expressing different types of constraints for the states produced and consumed by the corresponding tasks. For example, the situation where a batch can be split into two or more smaller batches, but no batch mixing is allowed can be addressed by (1) treating the state produced by the preceding task as hybrid state, and (2) allowing multiple withdrawals. The latter is achieved by removing the restriction that the vessel be emptied upon material transfer

$$\sum_{s \in \mathbf{S}^H} S_{sjt}^- \leq \beta_j^{\max} \left(1 - \sum_{j' \in \mathbf{JC}_j^-} W_{j'jt} \right) \quad \forall j \in \mathbf{J}^V, t \tag{11b}$$

Thus, an alternative interpretation of our approach that does not employ the concept of subsystems is the following: the entire process is represented via tasks and states, where the latter are classified according to the material routing restrictions they are subject to and modeled accordingly. In particular, each state can be categorized and modeled as shown in Table 1, where different constraints are used to model inventory.

Table 1. Different Types of Hybrid States and the Relevant Inventory Constraints

Material routing restriction	State type	Relevant inventory constraints
No batch mixing; no batch splitting	Sequential	5Ha, 5Hb, 17, 11
No batch mixing; batch splitting allowed	Hybrid	5Ha, 5Hb, 17, 11b
Batch mixing allowed; no batch splitting	Hybrid	5Ha, 5Hb, 17, 11a
No material routing restrictions—interfacing subsystems	Hybrid	5Ha, 5Hb, 17
No material routing restrictions	Network	5N

Extensions

In this section, we present how we can extend our MIP formulation to account for three important features: continuous processes, maintenance activities, and rescheduling. Our framework can also be extended to account for aspects such as storage of materials in processing units and transfer activities, but this discussion is beyond the scope of this article. Gimenez et al. present a thorough treatment of these aspects.[29–30] We close this section with the discussion of a set of tightening constraints that enhance the solution of the proposed MIP formulation.

Continuous processes

Facilities where batch processing is followed by downstream continuous lines (e.g., packaging) are quite common. Since our goal is to present a framework capable of addressing problems in facilities that consist of different types of processes, we incorporate the modeling of continuous processes as discussed by Maravelias.[11] Here we assume that the processing times for continuous processes are multiples of δ. To assign continuous tasks $i \in \mathbf{I}^L$ on units, we use the same binary variable X_{ijt}^P which now denotes an assignment over period $t+1$ (from time t to $t+1$) instead of an assignment starting at t and finishing at $t+\tau_{ij}$. Furthermore, we define the batch size B_{ijt} for period t which is determined by the minimum, $\upsilon_{ij}^{\min}$, and maximum, $\upsilon_{ij}^{\max}$, production rates and the length δ

$$(\upsilon_{ij}^{\min}\delta) X_{ijt}^P \leq B_{ijt} \leq (\upsilon_{ij}^{\max}\delta) X_{ijt}^P \quad \forall i \in \mathbf{I}^L, \quad j \in \mathbf{J}_i^P, t \tag{26}$$

The constraints required for continuous subsystems are Eqs. 2 to 4, 5N, 12, 13a, and 26, where we assume that batches produced by sequential processes can be either consumed partially or mixed by continuous processes. More complex restrictions, such as minimum processing time

constraints, can be modeled using existing modeling methods. Furthermore, like in network subsystems, we assume that we have dedicated storage vessels for continuous states, an assumption that can be trivially relaxed as discussed in the previous section.

Maintenance tasks

In practice, maintenance activities on units and vessels are usually scheduled over a time window $[\mu_j^S, \mu_j^E]$. These activities can be modeled by introducing pseudo tasks $i \in \mathbf{I}^M$ for units/vessels that require maintenance of duration τ_{ij}. Using binary variables X_{ijt}^M, we introduce Eqs. 27 and 28 to ensure that the maintenance task i is carried out during the given time window

$$\sum_{t \geq \mu_j^S}^{t \leq \mu_j^E - \tau_{ij}} X_{ijt}^M = 1 \quad \forall i \in \mathbf{I}^M, \quad j \in \mathbf{J}_i \tag{27}$$

$$X_{ijt}^M = 0 \quad \forall i \in \mathbf{I}^M, \quad j \in \mathbf{J}_i, \quad t < \mu_j^S / t > \mu_j^E - \tau_{ij} \tag{28}$$

Furthermore, we ensure that a unit/vessel is not assigned to other tasks/materials while the maintenance task is being carried out

$$R_{jt}^P = R_{j(t-1)}^P - \sum_{i \in \mathbf{I}_j} X_{ij(t-\tau_{ij})}^P + \sum_{i \in \mathbf{I}_j} X_{ijt}^P - \sum_{i \in \mathbf{I}_j} X_{ij(t-\tau_{ij})}^M + \sum_{i \in \mathbf{I}_j} X_{ijt}^M \quad \forall j \in \mathbf{J}^P, t \tag{29}$$

$$R_{jt}^V = R_{j(t-1)}^V - \sum_{s \in \mathbf{S}_j^V} X_{sj(t-1)}^V + \sum_{s \in \mathbf{S}_j^V} X_{sjt}^V - \sum_{i \in \mathbf{I}_j} X_{ij(t-\tau_{ij})}^M + \sum_{i \in \mathbf{I}_j} X_{ijt}^M \quad \forall j \in \mathbf{J}^V, t \tag{30}$$

Note that Eqs. 29 and 30 are generalizations of Eqs. 2 and 3.

Rescheduling

Rescheduling is commonly performed whenever processes disruptions occur or rush orders arrive. To incorporate this in our framework, we define two time points:

(a) Rescheduling point, σ^R: It is the time point at which the actual disruption occurs or rush order arrives.

(b) Restarting point, σ^S: It is the time point given by $\sigma^S = \sigma^R - \kappa$, where $\kappa = \max_{i,j \in \mathbf{J}_i^P} \tau_{ij}$.

Our goal is to reschedule starting at σ^R, subject to previously made decisions that affect the current state of the system; e.g., we have to account for tasks being executed at $t = \sigma^R$. To achieve this, we formulate a MIP model starting at $\sigma^S = \sigma^R - \kappa$, and we fix all decisions in the interval $[\sigma^R - \kappa, \sigma^R - 1]$ to be identical to the ones already made and being implemented. Thus, in our rescheduling MIP model we account for all decisions that we cannot change, but are likely to affect/constrain our rescheduling solution. We choose κ to be the maximum of processing times so that we can account for all tasks that could possibly be continued after σ^R. If $\tilde{Y}_t$ represents the vector of level values of t-indexed variables in the current solution, then we set $Y_t = \tilde{Y}_t$ over the period $[\sigma^R - \kappa, \sigma^R - 1]$. Furthermore, since all variables are fixed, constraints for $[\sigma^R - \kappa, \sigma^R - 1]$ are trivially satisfied, so we express constraints for $t = \sigma^R, \sigma^R + 1, \ldots$. Note that constraints for $t = \sigma^R$ may not be satisfied in the case of a unit break-down or other major disruption at $t = \sigma^R$.

Tightening constraints

We calculate the minimum, $l_s^{\min}$, and maximum, $l_s^{\max}$, numbers of batches required to meet the total demand amounts of each finished sequential state $s \in \mathbf{S}^Q$ as in the following

$$l_s^{\min} = \left\lceil \frac{\widehat{\omega}_s}{\hat{\beta}_s^{\max}} \right\rceil \quad \forall s \in \mathbf{S}^Q, \quad l_s^{\max} = \left\lceil \frac{\widehat{\omega}_s}{\hat{\beta}_s^{\min}} \right\rceil \quad \forall s \in \mathbf{S}^Q$$

where

$$\hat{\beta}_s^{\max} = \min_{i \in \tilde{\mathbf{I}}_s} \left\{ \max_{j \in \mathbf{J}_i^P} \left\{ \beta_j^{\max} \right\} \right\} \forall s \in \mathbf{S}^Q,$$

$$\hat{\beta}_s^{\min} = \max_{i \in \tilde{\mathbf{I}}_s} \left\{ \min_{j \in \mathbf{J}_i^P} \left\{ \beta_j^{\min} \right\} \right\} \forall s \in \mathbf{S}^Q$$

and $i \in \tilde{\mathbf{I}}_s$ is the set of tasks in the production path of s. Note that $\widehat{\omega}_s$ is the total demand for finished material in the sequential subsystem.

Then, for every sequential task $i \in \mathbf{I}^Q$, we calculate the minimum $l_i^{\min}$ and maximum $l_i^{\max}$ numbers of assignments

$$l_i^{\min} = \sum_{s: i \in \tilde{\mathbf{I}}_s} l_s^{\min} \quad \forall i \in \mathbf{I}^Q \text{ and } l_i^{\max} = \sum_{s: i \in \tilde{\mathbf{I}}_s} l_s^{\max} \quad \forall i \in \mathbf{I}^Q$$

Using these numbers, we develop the following tightening constraints that bound the number of assignments of tasks to units

$$l_i^{\min} \leq \sum_{j \in \mathbf{J}_i^P, t} X_{ijt}^P \leq l_i^{\max} \quad \forall i \in \mathbf{I}^Q \tag{31}$$

Furthermore, for the minimization of makespan, we add the following tightening constraint

$$MS \geq \sum_{i \in \mathbf{I}_j} X_{ijt}^P (t + \tau_{ij}) \quad \forall j \in \mathbf{J}^P, t \tag{32}$$

Applications and Results

We present three example problems to illustrate the application of the proposed strategy. The first one is based on the process shown in Figure 8 and discusses various extensions of our strategy. The second one considers a sequential process to highlight the reduction in formulation size caused by the proposed representation. The last example problem considers a general process that has batch as well as continuous subsystems. The data for the examples can be found in the online Supporting Information. In all the examples, we used $\delta = 1$.

Example 1: Illustrative example

We consider four different instances based on the process shown in Figure 8: (a) with no changeover times between

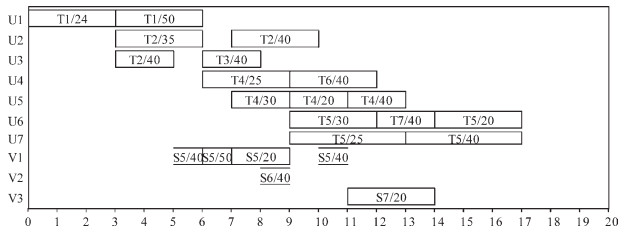

Figure 11. Gant chart of optimal solution of instance 1a.

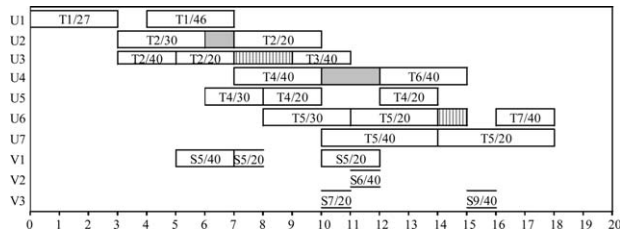

Figure 13. Gantt chart of optimal solution of instance 1c.

Maintenance time windows for units U2 and U4 are [6, 10] and [10, 15]; and maintenance task durations are 1 h and 2 h, respectively. Changeovers are denoted by ▨; maintenance tasks are denoted by ■.

tasks, (b) with changeover times, (c) with changeovers and maintenance tasks, and (d) with changeovers and rescheduling to accommodate rush orders. We consider makespan minimization as the objective function. The Gantt charts of the optimal solutions for all four instances are shown in Figures 11 to 14, where task/batch information is given.

The optimal makespan for instance (1a) is 17 h, while the addition of changeover times increased it to 18 h in instance (1b). In addition to the changeover times, instance (1c) considers scheduled maintenance tasks on units U2 and U4. The optimal makespan for this instance is the same as that for instance (1b). Finally, in instance (1d), assuming that we are executing the optimal schedule of instance (1b), a rush order of 30 kg for S10, due at 30 h, arrives at $\sigma^R = 7$ h. To accommodate this new order, we first fix the decisions over the time window $[7 - \kappa, 6]$, where $\kappa = 4$ in this example, and then solve the scheduling problem from $\sigma^S = 3$ h. The optimal makespan for this instance is 19 h, or 16 h starting from σ^S (see Figure 14).

Example 2: Sequential process

We consider a multistage (sequential) process with two stages (see Figure 15). The first stage has three processing units while the second has five units. There are nine storage vessels available: one for raw material, one for intermediates and seven for finished products. We consider three instances: (a) one order for each finished product (totaling 12 orders) with an objective of cost minimization; (b) same as instance (a), but for makespan minimization; (c) two orders for each product (totaling 24 orders) with an objective of cost minimization. The Gantt charts for the best solutions for instances (a) and (c) are given in Figures 16 and 17, respectively. The optimal makespan for instance (b) is 15 h.

Example 3: Large-scale example

The last example considers a general, bigger process that contains the downstream packaging section, which is a continuous process, besides the upstream batch process (see Figure 18). In terms of batch processing, there are five network tasks and eight sequential tasks, whereas there are 12 different continuous packaging tasks. In total, there are 25 different tasks and 28 different material states available. The plant has 15 processing units and 9 storage vessels. The objective is to minimize the total production cost required to meet 15 customer orders within due dates. The Gantt chart of the best solution is shown in Figure 19.

Computational results

Table 2 summarizes the model and solution statistics for all instances. We used GAMS 23.3/CPLEX 12.1 on a desktop computer with a 2.67 GHz Intel Core (i7-920) processor and 6.0 GB RAM running on Windows 7. We compare the results with and without using the proposed tightening constraints. The maximum resource limit for all instances is 1 h. The addition of tightening constraints improved the LP-relaxation for the cost minimization problems substantially (940.0 vs. 633.3 for 2a; 2065.0 vs. 1360.8 for 2c; and 2397.6 vs. 2287.6 for 3). Furthermore, the addition of tightening constraints enhanced the solution times by nearly three orders of magnitude for some instances (0.8 s vs. 1084 s for 1b; solved to optimality in 1.3 s vs. 29.8% gap after 3600 s for 2a; 126 s vs. 29.4% after 3600 s for 2b), and resulted in better solutions and smaller optimality gaps for bigger instances (2.4% after 3600 s vs. 35.3% after 3600 s for 2c; 4.7% after 3600 s vs. 9.2% after 3600 s for 3).

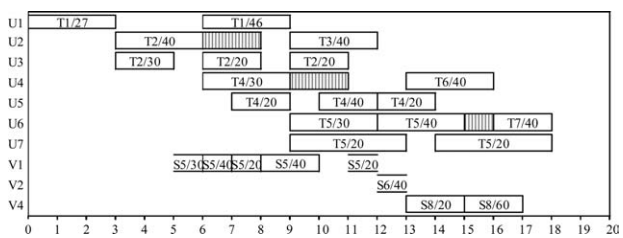

Figure 12. Gantt chart of optimal solution of instance 1b.

Changeover times are denoted by ▨.

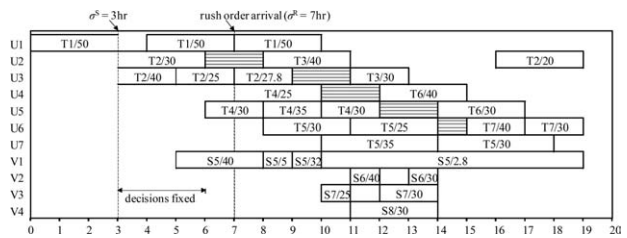

Figure 14. Gantt chart of optimal solution of example 1d.

Changeover times are denoted by ▨.

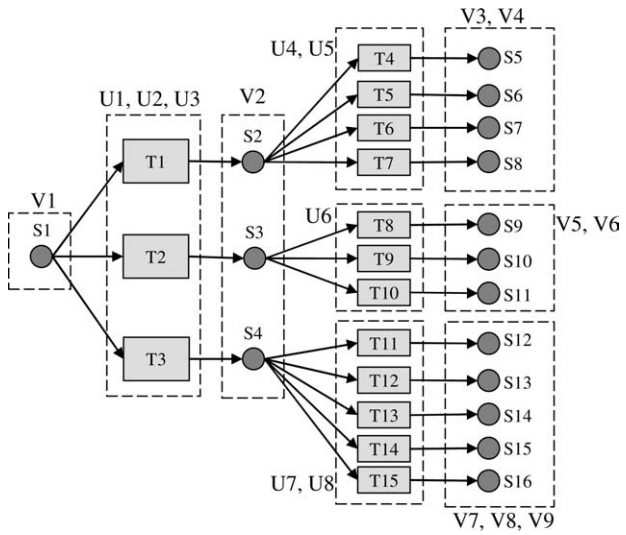

Figure 15. A multistage process for example 2.

Conclusions

In this article, we presented a strategy for the scheduling of general chemical processes. Our strategy relies on: the classification of the various subsystems of a process into network, sequential and continuous subsystems; the representation of sequential subsystems using a material based (network-like) formalism; the development of novel constraints to preserve batch integrity in sequential subsystems; and the formulation of an effective MIP formulation. Our strategy addresses the major open challenge in the formulation of scheduling problems, namely, the traditional *dichotomy* between material-based approaches for network processes and order-based approaches for sequential processes. Therefore, it facilitates the simultaneous optimization of different subsystems of a process, thereby allowing us to obtain better solutions. Furthermore, when coupled with recent developments in the modeling of a wide variety of process constraints,[29,30] it allows us to represent all scheduling problems. Hence, our future research efforts will focus on the development of effective solution methods for scheduling problems, which now appears to be the only major challenge in this area.

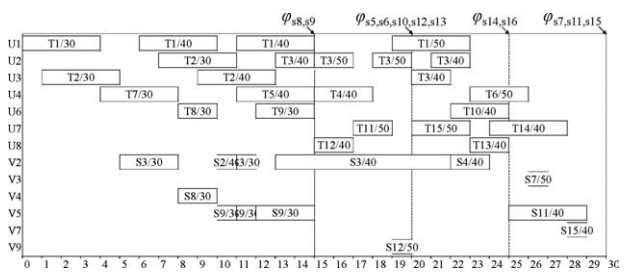

Figure 16. Gantt chart of optimal solution of instance 2a.

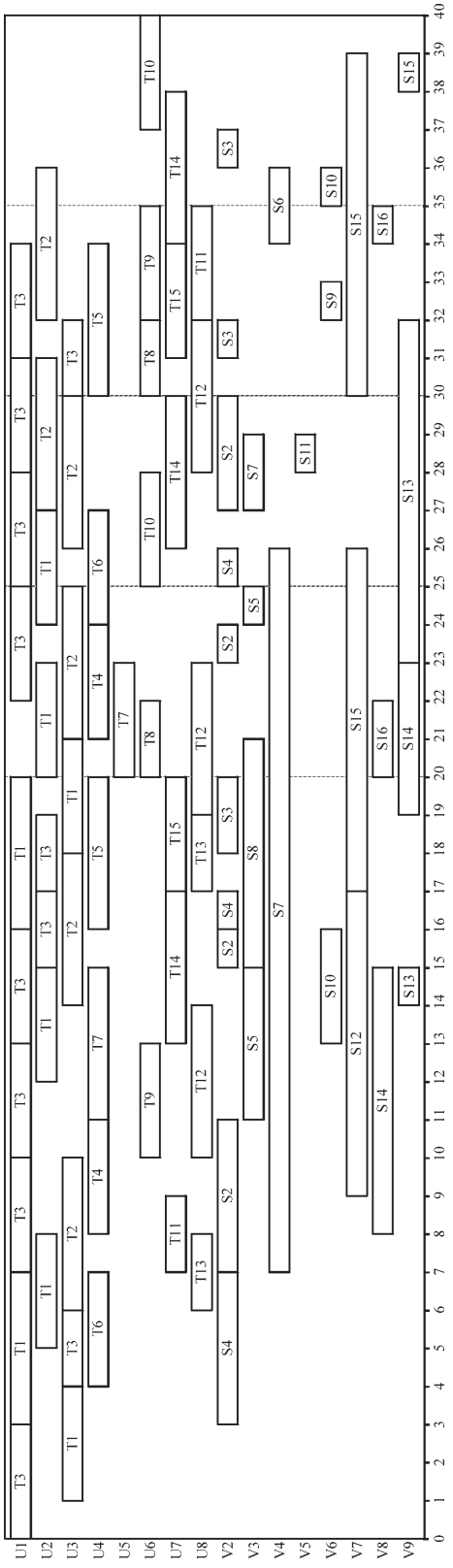

Figure 17. Gantt chart of best solution of example 2c.

For clarity purposes, the batch-size information is not given inside the boxes.

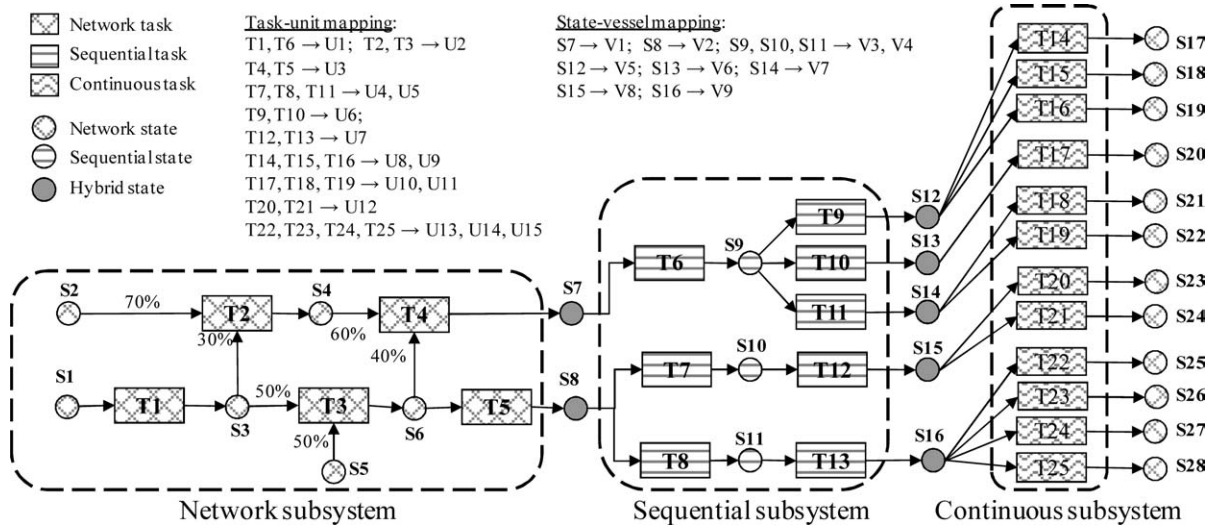

Figure 18. General process for example 3.

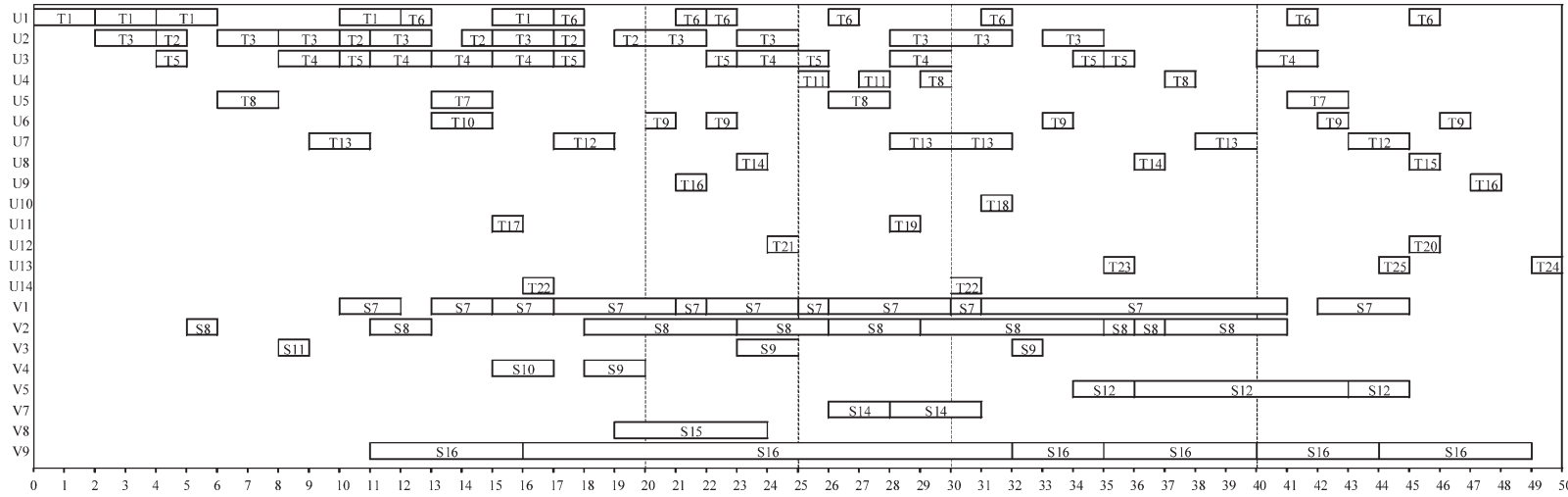

Figure 19. Gantt chart of best solution of example 3.

For clarity purposes, the batch-size information is not given inside the boxes.

Table 2. Model and Solution Statistics for Examples 1 to 3

	1a	1b	1c	1d	2a	2b	2c	3
Without tightening constraints								
Binary variables	1,176	1,176	1,185	1,026	4,102	4,402	5,966	11,565
Continuous variables	2,143	2,143	2,260	2,168	10,593	10,293	16,503	18,883
Constraints	3,536	3,894	3,980	3,533	14,490	11,339	21,140	23,476
LP-relaxation	13.3	13.3	13.3	14.9	633.3	9.1	1360.8	2287.6
Objective	17.0	18.0	18.0	16.0	940.0	17.0	2105.0	2530.0
CPU (s)	330	1084	236	2.7	3600	3600	3600	3600
Nodes	21,761	29,141	8,096	467	35,148	28,931	9,676	76,063
Integrality gap (%)	0.0	0.0	0.0	0.0	29.8	29.4	35.3	9.2
With tightening constraints								
Binary variables	1,176	1,176	1,185	1,026	4,102	4,402	5,966	11,565
Continuous variables	2,143	2,143	2,260	2,168	10,593	10,293	16,503	18,883
Constraints	3,544	3,902	3,988	3,541	14,520	11,369	21,170	23,492
LP-relaxation	13.3	13.3	13.3	14.9	940.0	9.1	2065.0	2397.6
Objective	17.0	18.0	18.0	16.0	940.0	15.0	2115.0	2530.0
CPU (s)	1.6	0.8	1.5	0.5	1.3	126	3600	3600
Nodes	56	0	0	0	0	659	3,040	58,599
Integrality gap (%)	0.0	0.0	0.0	0.0	0.0	0.0	2.4	4.7

Acknowledgments

The authors would like to acknowledge financial support from the National Science Foundation under grant CTS-0547443.

Notation

Indices/sets

$i,i' \in \mathbf{I}$ = tasks
$j,j' \in \mathbf{J}^P/\mathbf{J}^V/\mathbf{J}^C$ = processing units/storage vessels/customers
$s \in \mathbf{S}$ = states
$t \in \mathbf{T}$ = time points/periods
$u \in \mathbf{U}$ = utilities

Subsets

$\mathbf{I}^Q/\mathbf{I}^N/\mathbf{I}^L$ = processing tasks that belong to sequential/network/continuous subsystems
$\mathbf{I}^M$ = maintenance tasks
$\mathbf{I}_j^Q$ = sequential tasks that can be performed on unit j
$\mathbf{I}_s^+/\mathbf{I}_s^-$ = tasks that produce/consume state s
$\mathbf{I}_j$ = tasks that can be carried out in j
$\tilde{\mathbf{I}}_s$ = tasks in the production path of finished state $s \in \mathbf{S}^F$
$\mathbf{J}_i^P/\mathbf{J}_s^V/\mathbf{J}_s^C$ = units/vessels/customers that can process/store/consume task i/state s
$\mathbf{J}_s^-/\mathbf{J}_s^+$ = units that consume/produce or vessels that store or customers that require state s
$\mathbf{JC}_j^-/\mathbf{JC}_j^+$ = units or vessels that can be connected to/from unit or vessel j
$\mathbf{S}_j^V$ = states that can be stored in vessel j
$\mathbf{S}^Q/\mathbf{S}^N/\mathbf{S}^H$ = sequential/network/hybrid states
$\mathbf{S}^F$ = final products
$\mathbf{S}_j$ = states that can be stored in vessel j

Parameters

$\alpha_{ij}/\chi_{sj}/\zeta_u$ = cost of production/storage/utility
$\beta_j^{\min}/\beta_j^{\max}$ = minimum/maximum allowable batch size in processing unit j
δ = length of each period
ξ_{iju}/ψ_{iju} = fixed/variable amount of utility u for a batch of order i in unit/vessel j
$\varepsilon_s/\varphi_{sj}$ = release/due time for state s to customer j
$\eta_{i'ij}^P/\eta_{s'sj}^V$ = changeover times between tasks/states on units/vessels
$\gamma_s^{\max}$ = maximum storage capacity of state s
κ = maximum processing time, used in rescheduling
μ_j^S, μ_j^E = start/end of maintenance time window in j
$\upsilon_{ij}^{\min}/\upsilon_{ij}^{\max}$ = minimum/maximum production rate of i in j
$\omega_{sj}/\omega_{sj}^{\max}$ = mandatory/maximum demand amount for state s
π_{sj} = price of state s to customer j
ρ_{is}^+/ρ_{is}^- = fraction of state s produced/consumed by task i
σ^R, σ^S = time point at which disruption occurs/rescheduling starts
τ_{ij} = processing time of order i in unit j
θ_u^{max} = maximum availability of utility u

Binary variables

R_{jt}^P/R_{jt}^V = consumption of unit/vessel j at time point t
$W_{jj't}$ = connection of unit/vessel j to unit/vessel j' at time point t
X_{ijt}^P/X_{sjt}^V = assignment of task i or state s to unit/vessel j in period t
X_{ijt}^M = assignment of maintenance task to j at time point t
Z_{sjt} = delivery of state s to customer j at time point t

Nonnegative variables

B_{ijt} = batchsize of task i in unit/vessel j at time point t
D_{sjt} = delivery amount of s to customer j at time point t
$F_{sjj't}$ = flow of a batch of state s from unit/vessel j to unit/vessel j' at time point t
MS = makespan
N_{ut} = consumption level of utility u at time point t
S_{sjt}^-/S_{sjt}^+ = inventory of state $s \in \mathbf{S}\backslash\mathbf{S}^N$ in vessel j just before/after time point t
S_{st} = inventory of state $s \in \mathbf{S}^N$ at time point t
T_{sj}^E = earliness of state s to customer j
T_{sj}^R = tardiness of state s to customer j

Literature Cited

1. 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.
2. Shah N, Pantelides CC, Sargent RWH. A general algorithm for short-term scheduling of batch operations—II. Computaional issues. *Comput Chem Eng*. 1993;17:229–244.
3. Pantelides CC. Unified frameworks for the optimal process planning and scheduling. In: *Proceeding on the 2nd Conference on Foundations of Computer Aided Operations*. AIChE: New York; 1994:253–274.
4. Mockus L, Reklaitis GV. Continuous-time representation approach to batch and continuous process scheduling. 1. MINLP formulation. *Ind Eng Chem Res*. 1999;38:197–203.

5. Zhang X, Sargent RWH. The optimal operation of mixed production facilities—a general formulation and some approaches for the solution. *Comput Chem Eng*. 1996;20;897–904.
6. Giannelos NF, Georgiadis MC. A simple new continuous-time formulation for short-term scheduling of multipurpose batch processes. *Ind Eng Chem Res*. 2002;41:2178–2184.
7. Castro P, Barbosa-Povoa APFD, Matos H. An improved RTN continuous-time formulation for the short-term scheduling of multipurpose batch processes. *Ind Eng Chem Res*. 2001;40:2059–2068.
8. Ierapetritou MG, Floudas CA. Effective continuous-time formulation for short-term scheduling. 1. Multipurpose batch processes. *Ind Eng Chem Res*. 1998;37:4341–4359.
9. Maravelias, CT, Grossmann IE. New continuous-time state task network formulation for the scheduling of multipurpose batch plants. *Ind Eng Chem Res*. 2003;42(13):3056–3074.
10. 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(10):2679–2702.
11. Maravelias CT. Mixed-time representation for state-task network models. *Ind Eng Chem Res*. 2005;44:9129–9145.
12. Prasad P, Maravelias CT. Batch Selection, Assignment and sequencing in multi-stage multi-product processes. *Comput Chem Eng*. 2008;32:1106–1119.
13. Sundaramoorthy A, Maravelias CT. Simultaneous batching and scheduling in multistage multiproduct processes. *Ind Eng Chem Res*. 2008a;47:1546–1555.
14. Sundaramoorthy A, Maravelias CT. Modeling of storage in batching and scheduling of multistage processes. *Ind Eng Chem Res*. 2008b;47:6648–6660.
15. Sundaramoorthy A, Maravelias CT, Prasad P. Scheduling of multistage batch processes under utility constraints. *Ind Eng Chem Res*. 2009;48:6050–6058.
16. Castro PM, Erdiric-Dogan M, Grossmann IE. Simultaneous batching and scheduling of single stage batch plants with parallel units. *AIChE J*. 2008;54:183–193.
17. Hui C, Gupta A. A novel MILP formulation for short-term scheduling of multistage multiproduct batch plants. *Comput Chem Eng*. 2000;24(12):1611–1617.
18. Méndez CA, Henning GP, Cerdá J. An MILP continuous-time approach to short-term scheduling of resource constrained multistage flowshop batch facilities. *Comput Chem Eng*. 2001;25: 701–711.
19. Méndez CA, Cerdá J. An MILP continuous-time framework for short-term scheduling of multipurpose batch processes under different operation strategies. *Optimizn Eng*. 2003;4:7–22.
20. Gupta S, Karimi IA. Scheduling a two-stage multiproduct process with limited product shelf life in intermediate storage. *Ind Eng Chem Res*. 2003;42(3):490–508.
21. Wu J, He X. A new model for scheduling of batch process with mixed intermediate storage policies. *J Chin Inst Chem Eng*. 2004;35:381–387.
22. Pinto JM, Grossmann IE. A Continuous time mixed integer linear programming model for short term scheduling of multi-stage batch plants. *Ind Eng Chem Res*. 1995;34:3037–3051.
23. Liu Y, Karimi IA. Scheduling multistage, multiproduct batch plants with nonidentical parallel units and unlimited intermediate storage. *Chem Eng Sci*. 2007;62:1549–1566.
24. Castro PM, Grossmann IE, Novais AQ. Two new continuous-time models for the scheduling of multi-stage batch plants with sequence dependent changeovers. *Ind Eng Chem Res*. 2006;45:6210–6226.
25. Castro PM, Novais AQ. Short-term scheduling of multistage batch plants with unlimited intermediate storage. *Ind Eng Chem Res*. 2008;47:6126.
26. Maravelias CT, Sung C. A projection-based method for production planning of multiproduct facilities. *AIChE J*. 2009;55: 2614–2630.
27. Méndez CA, Cerda 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.
28. Prasad P, Maravelias CT, Kelly J. Optimization of aluminum smelter casthouse operations. *Ind Eng Chem Res*. 2006;45:7603–7617.
29. Gimenez DM, Henning GP, Maravelias CT. A novel network-based continuous-time representation for process scheduling: Part I. Main concepts and mathematical formulation. *Comput Chem Eng*. 2009;33:1511–1528.
30. Gimenez DM, Henning GP, Maravelias CT. A novel network-based continuous-time representation for process scheduling: Part II. General framework. *Comput Chem Eng*. 2009;33:1644–1660.

Manuscript received Nov. 3, 2009, and revision received Apr. 20, 2010.