Valid Inequalities Based on Demand Propagation for Chemical Production Scheduling MIP Models

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

DOI 10.1002/aic.14021
Published online February 5, 2013 in Wiley Online Library (wileyonlinelibrary.com)

Although several mixed-integer programming (MIP) models have been proposed for the scheduling of chemical manufacturing facilities, the development of solution methods for these formulations has received limited attention. In this article, we develop a constraint propagation algorithm for the calculation of lower bounds on the number and size of tasks necessary to satisfy given demand. These bounds are then used to express four types of valid inequalities which greatly enhance the computational performance of the MIP scheduling model. Specifically, the addition of these inequalities leads to reductions in the computational requirements of more than three orders of magnitude, thereby allowing us to address medium-sized problems of industrial relevance. Importantly, the proposed methods are applicable to a wide range of problem classes and time-indexed MIP models for chemical production scheduling. © 2013 American Institute of Chemical Engineers *AIChE J*, 59: 872–887, 2013

Keywords: chemical production scheduling, mathematical programming

Introduction

Chemical production involves the conversion of raw materials (feedstocks) into final products, both of which are most often fluids (gases or liquids). Each production task converts a set of input materials into a set of output materials and is carried out in equipment units that are capable of processing different amounts of materials. Although there are chemical facilities with a wide range of processing characteristics and constraints, there are two major types of chemical production environments, *sequential* and *network*.[1,2] In sequential environments, it is assumed that production tasks have a single input and a single output material; the input material of a task is produced by a single upstream task; and the output of a task is consumed by a single downstream task. Thus, materials from different tasks are not mixed, and the output of a task cannot be split to be consumed by multiple tasks. Also, it is implicitly assumed that each material is produced and consumed by at most one task. In the process systems engineering (PSE) literature, these facilities are termed either as multistage, if the routing through the stages is the same for all batches/products, or multipurpose, if there are batch/product-specific routings. In network environments, tasks may consume or produce multiple materials, and a material can be produced and consumed by different types of tasks. Also, material coming from different tasks can be mixed in a storage vessel, the output of a task can be consumed by multiple downstream tasks, and material can also be recycled.

Interestingly, the methods that have been developed to address chemical production scheduling problems follow for the most part the aforementioned classification. Early work focused on sequential facilities where a batch of raw material has to go through multiple production stages to be converted to the final product. In other words, the processing of fluids is represented as the processing of a discrete batch that goes through different processing stages; amounts of materials are not monitored and no material balances are included. We note that this type of modeling is possible because splitting, mixing, and recycling are not allowed. We term these approaches as *batch-based*. Early models considered fixed number and size of batches (for known demand), which meant that the batching problem was solved before scheduling.[3–5] Also, it was typically assumed that there are no storage and utility constraints. Batch-based models were first extended to account for special cases of storage constraints[6] and utility requirements,[7] and more recently, to account for the simultaneous batching and scheduling,[8,9] with general storage[10] and utility constraints.[11]

Clearly, batch-based methods cannot be used to address problems in network facilities where mixing, splitting, and recycling are allowed; materials can be produced and consumed by different tasks; and the sizes of the tasks are optimization decisions. To address these problems, researchers developed what we term *material-based* models, where the amounts of material processed by a task, as well as the inventories of materials, are explicitly modeled. Another key modeling aspect of these approaches is that a time reference grid is defined, and (1) the start/end times of tasks are mapped onto this grid and (2) inventory constraints are expressed at the time points of the grid. The pioneering work of Pantelides and coworkers[12–14] used a discrete-time grid. Since then, a wide range of formulations have been proposed.[15–24] Recently, researchers have started to look into the structure of material-based scheduling models,[25–27] as

Additional Supporting Information may be found in the online version of this article.

Correspondence concerning this article should be addressed to C. Maravelias at maravelias@wisc.edu.

well as work toward the development of solution methods.[28–32] Despite these efforts, however, both commercial products and specialized models and methods remain computationally inefficient for larger instances.

Accordingly, the goal of this article is to address this challenge through the development of general solution methods for material-based formulations. Specifically, we first develop an algorithm that allows us to calculate tight lower bounds on the number of batches of each task and the total amount of material processed by each task that are necessary to satisfy the given demand. These lower bounds, which are calculated from instance data within seconds, are then used to write four classes of strong valid inequalities; that is, constraints that reduce the feasible space of the linear programming (LP) relaxation of mixed-integer programming (MIP) scheduling models, and therefore speedup their solution, without cutting off any feasible integer solution. Although our methods can be applied to a wide range of MIP models, we choose to apply them to discrete-time models because a recent study suggests that they are computationally more effective than their continuous-time counterparts,[33] they can be easily extended to account for different processing characteristics and constraints,[34] and they are used as the backbone in a wide range of chemical production scheduling tools as well as in in-house tools developed by chemical companies.[35–37]

The article is structured as follows. First, we introduce a network-inspired representation of chemical facilities, present a formal statement of the problem we consider and a widely used MIP scheduling model, and close with a motivating example. Second, we present the constraint propagation algorithm for the calculation of the lower bounds on the number of batches and the total material processed. Third, we present four types of valid inequalities, as well as extensions for problems with intermediate due times. Finally, we perform an extensive computational study including more than 70 problem instances. We use lowercase italic letters for indices, uppercase bold letters for sets, lowercase Greek letters for parameters, and uppercase italics for optimization variables.

Background

Chemical production scheduling notation

A chemical facility transforms a set of raw materials into a set of valuable products through a sequence of processing tasks carried out in equipment units. A processing task converts a set of input materials (raw materials or intermediates) into a set of outputs (intermediates or final products) according to specified conversion coefficients. Note that the term *task* is used to describe a type of operation (e.g., the conversion of A to B via reaction A $\rightarrow$ B is a task), which implies that a schedule may include multiple executions (or instances) of the same task. A task can in general be carried out in multiple units of unequal capacity, which implies that the number of batches of a task necessary to satisfy given demand is not known before optimization. Thus, the term scheduling in the PSE literature has been used to describe a problem which includes three levels of decisions: (1) the determination of the number (and size) of batches to be performed, (2) the assignment of batches to processing units, and (3) the timing of batches so that at most one batch is executed on a unit.

We assume that a chemical facility consists of: (1) processing units, where tasks are executed; and (2) storage vessels, where materials are stored. Processing units and storage vessels have capacity limitations, and each processing unit (storage vessel) is capable of processing (storing) a subset of tasks (materials). We do not consider other shared resources such as utilities (e.g., electricity) and labor. Figure 1a shows the process flow diagram (PFD) of a simple chemical facility, where raw material S1 is converted into intermediate S2 in unit U1, and the latter is converted to either product S3 or S4 in unit U2 or U3. Note that the PFD shows the physical equipment of the facility, but not necessarily the tasks that take place on the units. Also, units can perform multiple tasks; for example, units U2 and U3 can convert S2 to S3 or convert S2 to S4.

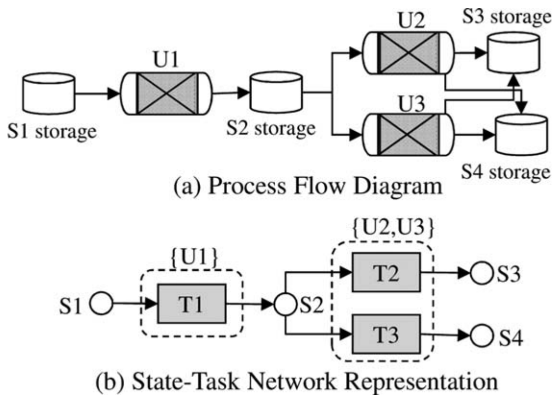


Figure 1. Process flow diagram (physical layout) and state-task network representation (procedural layout) of a facility.

Problem statement

To study scheduling problems, we have to use a representation that includes not only the physical resources, but also the different tasks that can be carried out. In this article, we adopt the state-task network (STN) representation,[12] which is based on the following concepts:

a. Materials (states): materials include feeds, intermediates, and final products and are represented by circles.

b. Tasks: tasks are operations that produce and consume materials and are depicted with rectangles.

c. Units: unary resources for the execution of tasks; multiple units may be able to process a single task, and a single unit may be able to process multiple tasks.

Materials and tasks are connected by arrows, referred to as streams, showing the flow of material. If a task consumes (produces) a material, an arrow points from the material (task) to the task (material). Figure 1b shows the STN representation for a scheduling problem in the facility shown in Figure 1a. Note that units are shown implicitly through the task-unit compatibility information given by the dotted lines. The structure of the network is defined in terms of the following sets (see Figure 2a):

Indices/Sets

$i,i' \in \mathbf{I}$ tasks

$j \in \mathbf{J}$ processing units

$s,s' \in \mathbf{S}$ materials

Subsets

$\mathbf{I}_s^+/\mathbf{I}_s^-$ tasks producing/consuming material s

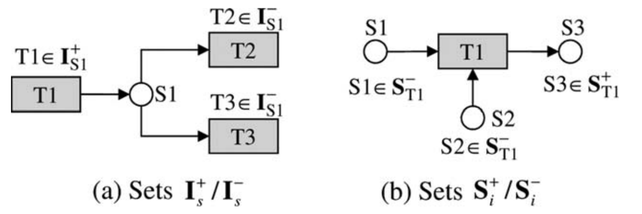
$\mathbf{J}_i^P$ processing units that can process task i

$\mathbf{S}_i^+/\mathbf{S}_i^-$ materials produced/consumed by task i

$\mathbf{S}^F$ final products

We are also given the following parameters:

ξ_{st} = amount of raw material s delivered at time t (>0) or customer demand for final product s at time t (<0); ξ_{s0} is the initial inventory of material s, including intermediates.



$T1 \in \mathbf{I}_{S1}^{+}$
$T2 \in \mathbf{I}_{S1}^{-}$
$T3 \in \mathbf{I}_{S1}^{-}$
T1
T2
T3
S1
S1
T1
S2
S3
$S1 \in \mathbf{S}_{T1}^{-}$
$S2 \in \mathbf{S}_{T1}^{-}$
$S3 \in \mathbf{S}_{T1}^{+}$
(a) Sets $\mathbf{I}_s^+ / \mathbf{I}_s^-$
(b) Sets $\mathbf{S}_i^+ / \mathbf{S}_i^-$

Figure 2. Illustration of general sets.

$\beta_j^{\min}/\beta_j^{\max}$=minimum/maximum capacity of unit j

ρ_{is}=fraction of material s produced (>0) or consumed (<0) by task i

τ_{ij}=processing time for task i in unit j

$\gamma_s^{\max}$ = maximum inventory of material s

In Figure 2a, S1 is consumed by T2 and T3 and produced by T1; therefore, $\mathbf{I}_{S1}^{-}$={T2, T3} and $\mathbf{I}_{S1}^{+}$={T1}. In Figure 2b, T1 produces S3 and consumes S2 and S1, so $\mathbf{S}_{T1}^{+}$={S3} and $\mathbf{S}_{T1}^{-}$={S1,S2}. In general, a plus (+) superscript indicates production, and a minus (−) indicates consumption. If T2 and T3 can both be processed in either U2 or U3, then $\mathbf{J}_{T2}^{P}=\mathbf{J}_{T3}^{P}$={U2, U3}.

Structure of scheduling constraints

Regardless of the nature of the time representation and the details of the formulation, there are three processing constraints that have to be enforced[27]:

1. Unit utilization: a unit cannot perform more than one task at a time.
2. Material balance: the inventory of a material in a storage vessel has to be nonnegative and less than the capacity of the vessel.
3. Batch-size: the size of a processing task (batch-size) has to be between a nonzero lower and an upper bound (semicontinuous constraint).

The main difference between the various material-based formulations lies in the modeling of the first constraint. In discrete-time models, it is enforced through the widely used one-machine clique constraint, whereas in continuous-time formulations more complex sets of constraints are used. The second constraint is equivalent to a flow balance constraint used in the network representation of production planning formulations.[38] Finally, the third constraint results in variable lower/upper bound constraints: if a task is executed, then the amount processed should be within a lower and upper bound; otherwise it is zero. The size of a task should be greater than or equal to the lower bound for safety and controllability reasons, as well as due to recipe constraints.

Most of the aforementioned types of constraints have been studied extensively. First, valid inequalities and specialized solution methods have been developed for time-indexed single-machine problems.[39–41] Also, many constraint propagation algorithms have been developed for this constraint.[42–45] Second, lot-sizing problems have material balances that are similar to those in network scheduling problems, and adding valid inequalities based on these material balances and bill-of-materials information results in a significantly tighter formulation.[46–48] Finally, valid inequalities have been derived from variable upper bound constraints in the context of fixed-charge network problems.[49,50] However, scheduling problems with variable lower bound constraints have not been studied sufficiently. Burkard and Hatzl[29] bound the number of batches of each task and the number of batches produced of each material by solving a small MIP for every task and an LP for every material, whereas Janak and Floudas[30] propose bounding the number of batches and cumulative production for each task by solving an MIP bounding problem for every task. Accordingly, in this article, we study how the variable lower bound constraints can be used to develop strong valid inequalities for MIP chemical production scheduling problems.

Discrete-time model

To illustrate the underlying concepts and perform computational studies, we introduce the widely used MIP formulation of Shah et al.[13] We use this model because a recent computational study showed that discrete-time models are more effective for large-scale problems and, most importantly, can be easily extended to account for a range of processing restrictions and characteristics (e.g., intermediate due dates, holding and utility costs, variable resource requirements during task execution) at almost no computational cost.[33] However, we highlight that our methods are applicable to all time-indexed MIP models for chemical production scheduling that account for variable batch sizes.

Time points are indexed by t. The formulation includes one family of binary variables:

X_{ijt}=is one if unit j processes task i starting at time t

and the following nonnegative continuous variables:

B_{ijt}=batch size of task i processed in unit j starting at time t

S_{st}=inventory of material s at time t

The model consists of unit utilization (Eq. 1), unit capacity (Eq. 2), inventory capacity (Eq. 3), and material balance (Eq. 4) constraints

$$\sum_{i\in\mathbf{I}_j}\sum_{t'\geq t-\tau_{ij}+1}^{t} X_{ijt'} \leq 1 \qquad \forall j,t \tag{1}$$

$$\beta_j^{\min} X_{ijt} \leq B_{ijt} \leq \beta_j^{\max} X_{ijt} \qquad \forall i, j\in\mathbf{J}_i^P, t \tag{2}$$

$$S_{st} \leq \gamma_s^{\max} \qquad \forall s,t \tag{3}$$

$$S_{st}=S_{s(t-1)}+\sum_{i\in\mathbf{I}_s^+}\rho_{is}\sum_{j\in\mathbf{J}_i^P}B_{ij(t-\tau_{ij})}+\sum_{i\in\mathbf{I}_s^-}\rho_{is}\sum_{j\in\mathbf{J}_i^P}B_{ijt}+\xi_{st} \qquad \forall s,t \tag{4}$$

Variables X_{ijt} are fixed to zero for the $\tau_{ij}-1$ time points before the end of the time horizon.

Motivating example

We consider an instance of the facility introduced in Figure 1, with a demand for 90 kg of S3 and 25 kg of S4. For each unit, the capacity (min-max) and processing cost of a task in that unit are: 25–60 kg and \$10 for U1; 40–50 kg and \$25 for U2; and 35–45 kg and \$30 for U3. Our objective is to minimize cost. We generated a model for this instance based on the formulation presented in the previous subsection. The minimum cost for the LP relaxation is \$76.7, while the number of batches (i.e., the sum of the X_{ijt} variables) of tasks T1, T2, and T3 are 1.9, 1.8, and 0.5, respectively (see Table 1).

However, if we take into account the capacities of the units, we can infer that more batches will be required in any feasible solution. T2 must produce 90 kg of S3 to satisfy demand; because the maximum batch size is 50 kg, this requires at least two batches. Similarly, T3 must produce at least 25 kg

Table 1. Effect of Tightening on the Number of Batches and Cost for the Network in Figure 1

	No. of Batches			Min. Cost ($)
	T1	T2	T3	
Optimal solution	3	2	1	105
Minimum No. of batches				
B&H method	2	2	1	
Proposed method	3	2	1	
LP relaxation				
No tightening	1.9	1.8	0.5	76.7
B&H method	2.2	2	1	96.7
Proposed method	3	2	1	105

Lower bounds on the number of batches are calculated using our proposed method and the method from Burkard and Hatzl[29] (B&H).

of S4, but, because the minimum batch size is 35 kg, T3 must produce at least 35 kg in at least one batch. Together, T2 and T3 require a minimum of 125 kg (=90+35) of S2, which T1 produces in at least three batches. Using this constraint-propagation procedure, we calculate lower bounds on the number of batches and cumulative production for each task.

Table 1 shows the effect of tightening on the example in Figure 1. Without tightening, the minimum cost for the LP relaxation is $76.7. With the proposed method, the cost of the LP relaxation is $105. Interestingly, the minimum cost and number of batches of the LP relaxation, after the addition of the bounds we calculated, match the optimal MIP solution. Our goal is to develop a method that allows us to systematically calculate similar bounds in general production networks, including facilities with recycle streams. Note that using the tightening methods proposed by Burkard and Hatzl[29] improves the cost of the LP relaxation to $96.7. Burkard and Hatzl do not consider minimum unit capacities, so their method requires T1 to produce only 115 kg (=90+25) of S2 in two batches. The tightening methods of Janak and Floudas[30] find a lower bound on the number of batches which is as tight as ours, but require solving a MIP for every task which is often as time consuming as solving the original MIP model. Also, their approach relies on a specific continuous-time model. Our proposed methods do not require solving any auxiliary MIPs, are valid for any network, and can be used with any model using a time grid.

Constraint Propagation Algorithm

Our constraint propagation algorithm calculates parameters which are then used to formulate the valid inequalities. The algorithm depends on the structure of the process network. We classify networks into four categories according to Figure 3:

1. networks with no loops (Figure 4a);

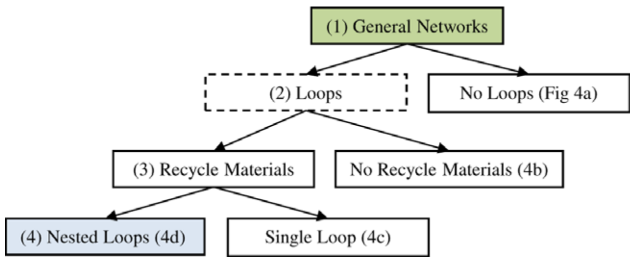
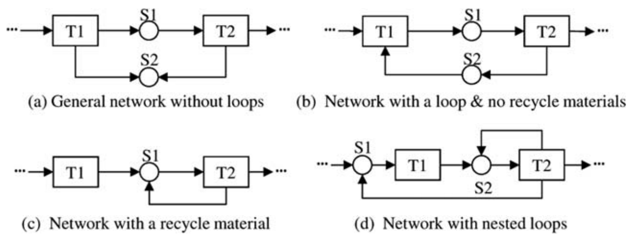


Figure 3. Classification of networks.

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



T1
S1
T2
S2
(a) General network without loops
(b) Network with a loop & no recycle materials
(c) Network with a recycle material
(d) Network with nested loops

Figure 4. Examples of categories of networks.

2. networks with loops but no recycle materials (Figure 4b);
3. networks with recycle materials in a single loop (Figure 4c); and
4. networks with a recycle material and multiple nested loops (Figure 4d).

A loop is any closed path (moving only in the direction of the stream arrows) within the network, and a recycle material is a material in a loop that can be produced by multiple tasks. Each colored box in Figure 3 has its own specific tightening procedures that are described in the indicated subsection. *Subsection General networks* describes the general methods used by all networks. Networks are first divided into networks with and without loops. If there are loops, then the *subsection Networks with loops* describes the additional tightening methods. Loops are further divided into loops with and without recycle materials, and *subsection Networks with recycle materials* explains more tightening methods for networks with recycle materials. When a loop has a recycle material, it is classified as having either a single loop or a nested loop, which is described in *subsection Networks with nested loops*. Next, we present a general algorithm for applying the tightening methods to any network. Then, we describe an alternative approach for networks with recycle materials that also works well when multiple tasks can produce a single material. Finally, we present remarks and extensions.

General networks

Backward Propagation. For backward propagation, we introduce the following parameters:

ω_s=lower bound on the amount of material s required to meet final demand

$\mu_i/\tilde{\mu}_i$ = lower bound on the production of task i required to meet final demand, $\mu_i \le \tilde{\mu}_i$

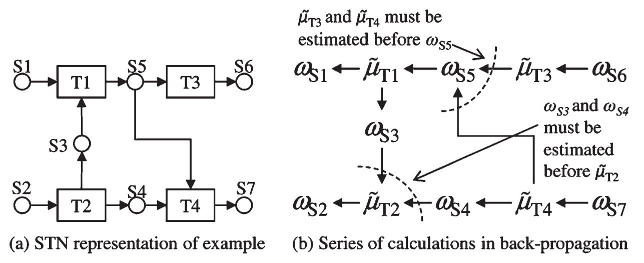
ν_{is}=lower bound on the amount of material s that must be produced by task i

First, we need to find the minimum amount required for all materials, ω_s, and the minimum production for all tasks, $\tilde{\mu}_i$. We calculate these parameters sequentially by backward propagating demand for final products. We estimate ω_s once $\tilde{\mu}_i$ is known for all tasks consuming material s (all $i \in \mathbf{I}_s^-$). Similarly, we need to know ω_s for all materials produced by task i (all $s \in \mathbf{S}_i^+$) before we calculate $\tilde{\mu}_i$ (see Figure 5).

We calculate ω_s starting with the final products

$$\omega_s = \begin{cases} -\sum_t \xi_{st} & s \in \mathbf{S}^F \\ -\sum_{i \in \mathbf{I}_s^-} \rho_{is}\tilde{\mu}_i - \xi_{s0} & s \notin \mathbf{S}^F \end{cases} \tag{5}$$

In the RHS of Eq. 5, the top expression is the total customer demand for final products, whereas the bottom expression is the amount of intermediate s required by all tasks consuming it minus any initial inventory. If the initial



$\tilde{\mu}_{T3}$ and $\tilde{\mu}_{T4}$ must be estimated before ω_{S5}
ω_{S3} and ω_{S4} must be estimated before $\tilde{\mu}_{T2}$
(a) STN representation of example
(b) Series of calculations in back-propagation

Figure 5. Example of backward propagation of demand.

From demand for S6 and S7, we calculate ω_{S6} and ω_{S7}; next, we calculate $\tilde{\mu}_{T3}$ and $\tilde{\mu}_{T4}$, followed by ω_{S5} and ω_{S4}. We cannot find $\tilde{\mu}_{T2}$ yet because ω_{S3} is not known, so we calculate $\tilde{\mu}_{T1}$, followed by ω_{S3} and ω_{S1}. Finally, we calculate $\tilde{\mu}_{T2}$ and ω_{S2}.

inventory exceeds the amount required, ω_s will be negative and the process does not need to produce material s.

The minimum amount of material s that must be produced by task i, v_{is}, is introduced as an intermediate parameter. Although ω_s can be negative, v_{is} must be at least zero

$$v_{is}=\begin{cases} \max\{\omega_s,0\} & s\in\mathbf{S}^{ST}, i\in\mathbf{I}_s^+ \\ 0 & s\in\mathbf{S}^{MT}\setminus\mathbf{S}^R, i\in\mathbf{I}_s^+ \end{cases} \tag{6}$$

where $\mathbf{S}^{ST}$ is the set of materials produced by a single task, $\mathbf{S}^{MT}$ is the set of materials that can be produced by multiple tasks, and $\mathbf{S}^R$ is the set of recycle materials. Equation 6 does not consider recycle materials, which we discuss later. If multiple tasks can produce a material, we assume that any single task can meet the full demand; therefore, the minimum production of that material by each task is zero, and, for these materials, an alternative approach that gives better results is described later.

After calculating v_{is} for all materials produced by a task, we calculate μ_i

$$\mu_i=\max_{s\in\mathbf{S}_i^+}\left\{\frac{v_{is}}{\rho_{is}}\right\} \tag{7}$$

The term inside the brackets is the total amount task i must process to meet the demand for s. We take the maximum over all materials produced by task i to ensure the task satisfies demand for all materials. At this point, we must find the minimum attainable production amount, $\tilde{\mu}_i\geq\mu_i$

$$\tilde{\mu}_i=\mu_i+\Delta\mu_i \tag{8}$$

where $\Delta\mu_i$ is the minimum amount that must be added to μ_i to reach an attainable production amount and is discussed below. Parameter $\tilde{\mu}_i$ provides a tighter lower bound on the required production of task i. We backward propagate demand until ω_s and $\tilde{\mu}_i$ are known for all materials and tasks.

Attainable Production Amounts. After finding μ_i, we need $\Delta\mu_i$ to calculate $\tilde{\mu}_i$. When only one unit can process a task, it is straightforward to find the range of attainable production amounts for any number of batches and to check if the required production is in one of those attainable ranges (see example in Figure 6). If μ_i falls in an attainable region, demand can be met exactly, and $\Delta\mu_i$ is zero; otherwise, $\Delta\mu_i$ is the distance between μ_i and the start of the next attainable range.

However, when multiple units can process a task, we must find and check attainable ranges for every possible combination of batches in units. For example, if units U1 and U2 can process a task, we need to check 1 batch in U1, 0 in U2; 0 in U1, 1 in U2; 1 in U1, 1 in U2; and so forth. To reduce the number of combinations, we find an upper bound on the number of batches in a particular unit, $\varepsilon_j^{\max}$, by dividing μ_i by the largest possible batch unit j can process and rounding up

$$\varepsilon_j^{\max}=\lceil\frac{\mu_i}{\beta_j^{\max}}\rceil \qquad \forall j\in\mathbf{J}_i^P \tag{9}$$

When the number of batches for a particular unit is at its upper bound, we are guaranteed the attainable range meets or exceeds demand with just one unit, and the number of batches in all other units is set to zero to eliminate more combinations. In total, there are K_i ranges to check

$$K_i=\prod_{j\in\mathbf{J}_i^P}\left(\varepsilon_j^{\max}\right)+|\mathbf{J}_i^P| \tag{10}$$

For each range, $k\in\{1,2,\ldots,K_i\}$, there is a unique combination of ε_j^k, where ε_j^k gives the number of batches in unit j for range k. The first term in Eq. 10 is the number of ranges with $\varepsilon_j^k=0,1,2,\ldots(\varepsilon_j^{\max}-1)$ for all $j\in\mathbf{J}_i^P$, and the second term is the number of ranges with $\varepsilon_j^k=\varepsilon_j^{\max}$ for one $j\in\mathbf{J}_i^P$ and $\varepsilon_j^k=0$ for all other $j\in\mathbf{J}_i^P$. We loop over all $k\in\{1,2,\ldots, K_i\}$ and check if μ_i falls into an attainable range. If, for task i

$$\sum_{j\in\mathbf{J}_i^P}\varepsilon_j^k\beta_j^{\min}\leq\mu_i\leq\sum_{j\in\mathbf{J}_i^P}\varepsilon_j^k\beta_j^{\max} \tag{11}$$

for some k, the minimum required production can be met exactly, and $\Delta\mu_i$ is zero. The LHS of Eq. 11 gives the lower bound of the attainable range, and the RHS gives the upper bound. If μ_i is not attainable for any range, we perform another loop over all k to find the range that is able to meet production requirements and whose lower bound is closest to μ_i

$$\Delta\mu_i^k=\begin{cases}\sum_{j\in\mathbf{J}_i^P}\varepsilon_j^k\beta_j^{\min}-\mu_i & \text{if}\sum_{j\in\mathbf{J}_i^P}\varepsilon_j^k\beta_j^{\min}\geq\mu_i \\ \infty & \text{if}\sum_{j\in\mathbf{J}_i^P}\varepsilon_j^k\beta_j^{\min}<\mu_i\end{cases} \tag{12}$$

The top line sets $\Delta\mu_i^k$ equal to the excess amount produced (the lower bound on the range minus μ_i) if the combination of units is at least able to meet demand. The second line sets $\Delta\mu_i^k$ to infinity when the combination's capacity is too small. Once all ranges have been checked, $\Delta\mu_i$ is calculated

$$\Delta\mu_i=\min_k\left\{\Delta\mu_i^k\right\} \tag{13}$$

As an example, we consider a task that can be processed in U1 or U2 and must process 55 kg (see Figure 7). We find $\varepsilon_{U1}^{\max}=3$ and $\varepsilon_{U2}^{\max}=2$ from Eq. 9; and K_i=8 from Eq. 10. All ε_j^k for $k\in\{1,2,\ldots, 8\}$ are shown on the left side of Figure 7. The attainable ranges are shaded (Eq. 11). Because μ_i= 55

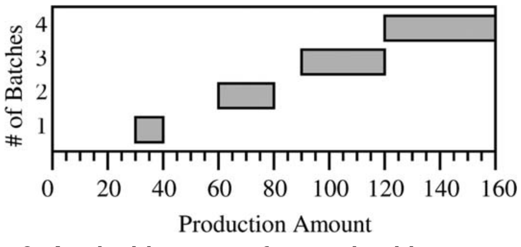


Figure 6. Attainable ranges for a unit with a capacity of 30–40 kg.

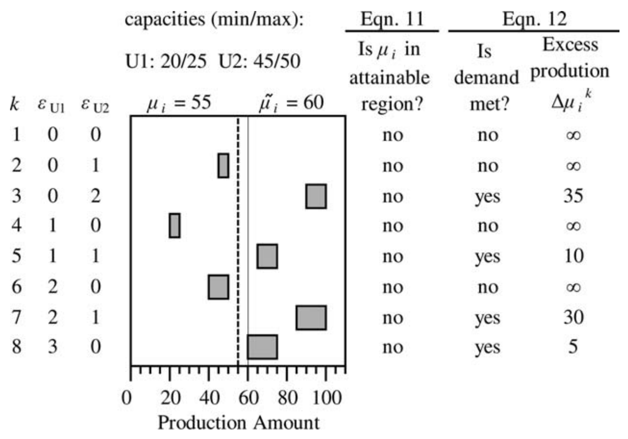


Figure 7. Example of calculation of the minimum attainable production amount.

does not fall in a gray range, Eq. 11 is never satisfied. The minimum excess material produced by each combination of units is the distance between μ_i and the start of the next shaded range (Eq. 12). For this example, the smallest excess production is 5, so $\Delta\mu_i$ is 5, and $\tilde{\mu_i}$ is 60.

The algorithm for finding $\Delta\mu_i$ is as follows:

```
1. Calculate $\varepsilon_j^{\max}$ and $K_i$ and set $\varepsilon_j^k$.
2. For $k \in \{1,2,\ldots, K_i\}$
3.     If Eq. 11 is true
4.         Set $\Delta\mu_i = 0$ and stop
5.     end
6. end
7. For $k \in \{1,2,\ldots, K_i\}$
8.     Calculate $\Delta\mu_i^k$ (Eq. 12)
9. end
10. Calculate $\Delta\mu_i$ (Eq. 13)
```

The attainable production amount, $\tilde{\mu}_i$, is now calculated using Eq. 8.

Networks with loops

When there are loops in a network, simple backward propagation will not work. We use tear streams to break the loop with one tear stream in every loop. A tear stream consists of a task, referred to as the tear task, and a material produced by that task, referred to as the tear material. Although any stream can be chosen as the tear stream, some choices are more convenient. If a loop has a task that produces a material outside the loop, this task should be used as the tear task; otherwise, any task can be chosen. The sets $\mathbf{I}^T$ and $\mathbf{S}^T$ contain all tear tasks and materials, and $\mathbf{I}_l^L$ and $\mathbf{S}_l^L$ give all tasks and materials in loop l.

Figure 8 provides an example for propagating demand for a network with a loop. We write the value of μ_i inside the box representing task i, ω_s inside the circle representing material s, and ν_{is} next to the stream connecting task i to material s. For simplicity, all examples have one unit per task, and capacities are given for each task. The loop is shown in bold. We chose T3 as the tear task because it produces S4 outside the loop, and S5 is the tear material. We initialize $\nu_{T3,S5}$ for the tear stream to zero (Figure 8b). Now, we can estimate $\tilde{\mu}_{T3}$ as soon as ω_{S4} is known. We backward propagate demand as follows: $\omega_{S4}=50 \Rightarrow \tilde{\mu}_{T3}=100 \Rightarrow \omega_{S3}=100 \Rightarrow \tilde{\mu}_{T2}=100 \Rightarrow \omega_{S2}=100 \Rightarrow \tilde{\mu}_{T1}=110$ ($\mu_{T1}=100$ and $\Delta\mu_{T1}=10$) $\Rightarrow \omega_{S1}=110–45=65 \Rightarrow \tilde{\mu}_{T4}=65 \Rightarrow \omega_{S5}=65*0.8=52$. We stop after calculating $\omega_{S5}=52$ for the tear material (Figure 8c). We need a minimum of 52 kg of S5, but T3 only produces 50 kg (=100*0.5). Therefore, we set $\nu_{T3,S5}=52$, and delete ω_s and μ_i for all other materials and tasks (Figure 8d). We repeat backward propagation using the new value $\nu_{T3,S5}=52$ (Figure 8e). Now, T3 produces the required 52 kg of S5.

In general, we begin by initializing ν_{is} for all tear streams to zero. We backward propagate demand until ω_s is estimated for a tear material. We update ν_{is} for the tear stream using Eq. 6 and compare it to the production of the tear task. If production can meet demand ($\nu_{is} \le \rho_{is}\tilde{\mu}_i$), there is no problem, and we continue with the backward propagation. If demand is greater than production ($\nu_{is} > \rho_{is}\tilde{\mu}_i$), we reset all other ω_s and $\tilde{\mu_i}$, and restart the backward propagation using the new initial value for the tear stream. If, on the second pass, demand for a tear material still exceeds production, the problem is infeasible with the given initial inventories.

Networks with recycle materials

Previously, when multiple tasks could produce a single material, we assumed that each task was capable of producing the full amount, and, therefore, each task had a minimum production of zero (Eq. 6). However, when a material $s \in \mathbf{S}^{MT}$ is part of a loop, we can find a better estimate for ν_{is}. We refer to these materials as recycle materials, $\mathbf{S}^R \subseteq \mathbf{S}^{MT}$. Note that the fraction of a material that is recycled is less than one; in other words, if we start out with some amount of the recycle material and run only tasks in the loop, we will never end up with more of the recycle material than what we had initially.

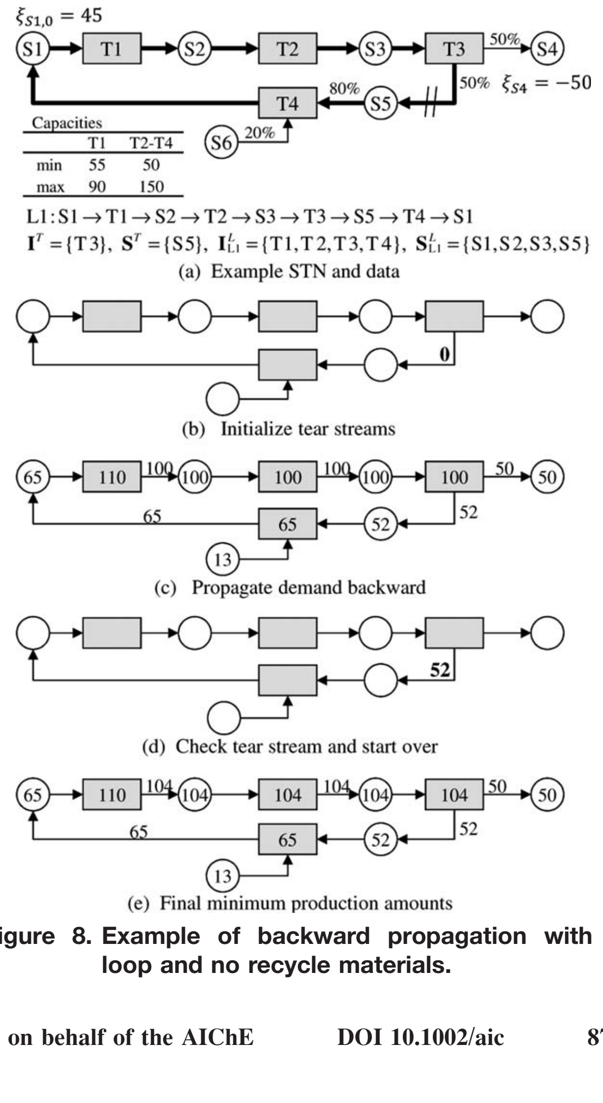

$\xi_{S1,0} = 45$
S1 T1 S2 T2 S3 T3 S4
50%
50% $\xi_{S4} = -50$
T4 80% S5
S6 20%
Capacities
T1 T2-T4
min 55 50
max 90 150
$L1: S1 \to T1 \to S2 \to T2 \to S3 \to T3 \to S5 \to T4 \to S1$
$\mathbf{I}^T = \{T3\}$, $\mathbf{S}^T = \{S5\}$, $\mathbf{I}_{L1}^L = \{T1, T2, T3, T4\}$, $\mathbf{S}_{L1}^L = \{S1, S2, S3, S5\}$
(a) Example STN and data
0
(b) Initialize tear streams
65 110 100 100 100 100 100 100 50 50
65 65 52 52 13
(c) Propagate demand backward
52
(d) Check tear stream and start over
65 110 104 104 104 104 104 104 50 50
65 65 52 52 13
(e) Final minimum production amounts

Figure 8. Example of backward propagation with a loop and no recycle materials.

We introduce the following new parameters for networks with recycle materials:

ω_s^R=upper bound on the amount of material s that can be produced when only the minimum amount of a recycle material, $\omega_{s'}$, is available

μ_i^R=upper bound on the production of task i when only the minimum amount of a recycle material is available

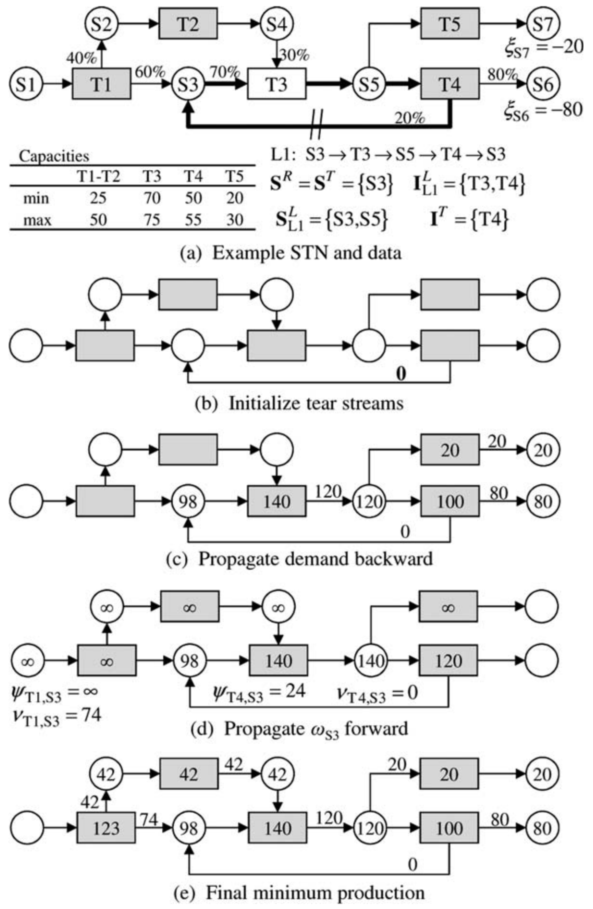
ψ_{is}=upper bound on the amount of recycle material s produced by task i when only the minimum amount of material s is available

Example. For the network in Figure 9, we start by selecting the tear stream in exactly the same way as above. We choose T4 $\rightarrow$ S3 as the tear stream and initialize $\nu_{\text{T4,S3}}$ to zero. The backward propagation continues until we calculate ω_{S3}=98 in Figure 9c. There is a maximum amount of S3 that T4 (T1) can produce without increasing the required production of T1 (T4) (see Proposition 1); this amount is given by the parameter ψ_{is}, which we estimate by propagating ω_{S3} forward (Figure 9d). In the forward propagation, we start out with ω_{S3}=98 kg of S3, which means T3 can produce at most μ_{T3}^R=140 kg (=98/0.7) of S5 (assuming there is enough S4). T5 needs $\tilde{\mu}_{\text{T5}}$=20 kg of S5 (from Figure 9c) and the remaining 120 kg (=140−20) of S5 are available for T4. Therefore, T4 produces up to $\psi_{\text{T4,S3}}$=24 kg (=120*0.2) of S3. Because ω_{S3}=98 kg of S3 are needed in all (from Figure 9c), we conclude that T1 must produce the remaining 74 kg (=98−24), and $\nu_{\text{T1,S3}}$=74. T1 is capable of producing all 98 kg of S3 ($\psi_{\text{T1,S3}}=\infty$ with unlimited initial inventory of feeds), so T4 is not required to produce any S3, and $\nu_{\text{T4,S3}}$=0. Because S3 is also a tear material, we must ensure that the production of S3 by T4 exceeds $\nu_{\text{T4,S3}}$; because it does ($0\leq 0.2*120$), we continue with the backward propagation until we have calculated ω_s and $\tilde{\mu}_i$ for all materials and tasks (Figure 9e).

General Methods for Recycle Materials. Backward propagation for networks with recycle materials starts by initializing ν_{is} for tear streams to zero. As with standard loops, when ν_{is} is calculated for a tear stream, we check if production exceeds demand. After calculating ω_s for a recycle material, we label this material s* and find the maximum amount that can be recycled. The minimum amount of the labeled recycle material that is required to meet demand, ω_{s*}, is propagated forward around the loop to find ω_s^R and $\mu_i^R$for all materials and tasks, respectively. For all materials and tasks outside the loop, we set $\omega_s^R$and μ_i^R to infinity; this ensures no feasible solutions are cutoff and assumes that these materials are produced starting from feeds with an unlimited supply. Once μ_i^R is known for all tasks producing a material ($i\in\mathbf{I}_s^+$), we calculate ω_s^R. We can calculate ω_s^R for the labeled recycle material immediately

$$\omega_s^R=\begin{cases} \omega_s+\xi_{s0} & \text{for s}=s* \\ \sum_{i\in\mathbf{I}_s^+}\rho_{is}\mu_i^R+\xi_{s0} & \text{if s} \in \bigcup_{l\in\mathbf{L}_{s*}^S}\mathbf{S}_l^L \text{ and } \max_{l\in\mathbf{L}_s^S}\left\{|\mathbf{I}_s^+\cap\mathbf{I}_l^L|\right\}=1\end{cases} \tag{14}$$

The first expression gives ω_s^R for the labeled recycle material s*. The second expression is for all other materials in the loop except materials produced by multiple tasks (other recycle materials) within any loop containing s*. The first term in the condition in the second line gives all materials in any loop containing s*, and the second term is the number of tasks within the loop that produce material s. If there is another recycle material inside the loop, there are nested



S1
S2
S3
S4
S5
S6
S7
T1
T2
T3
T4
T5
40%
60%
70%
30%
80%
20%
$\xi_{S7}=-20$
$\xi_{S6}=-80$

Capacities	T1-T2	T3	T4	T5
min	25	70	50	20
max	50	75	55	30

L1: S3→T3→S5→T4→S3
$\mathbf{S}^R=\mathbf{S}^T=\{\text{S3}\}$ $\mathbf{I}_{\text{L1}}^L=\{\text{T3,T4}\}$
$\mathbf{S}_{\text{L1}}^L=\{\text{S3,S5}\}$ $\mathbf{I}^T=\{\text{T4}\}$
(a) Example STN and data
0
(b) Initialize tear streams
98
140
120
120
20
20
20
100
80
80
0
(c) Propagate demand backward
∞
140
120
$\psi_{\text{T1,S3}}=\infty$
$\nu_{\text{T1,S3}}=74$
$\psi_{\text{T4,S3}}=24$
$\nu_{\text{T4,S3}}=0$
(d) Propagate ω_{S3} forward
42
123
74
(e) Final minimum production

Figure 9. Example of backward propagation with a single recycle material in a single loop.

loops in the network, and Case 1 of the next subsection discusses how to find ω_s^R. The total amount of material s available ω_s^R, is the amount produced by all tasks plus the initial inventory. Once ω_s^R is known for all materials consumed by task i ($s\in\mathbf{S}_i^-$), we find μ_i^R

$$\mu_i^R=\min_{s\in\mathbf{S}_i^-}\left\{\frac{\omega_s^R+\sum_{i'\in\mathbf{I}_s^-,i'\neq i}\rho_{i's}\tilde{\mu}_{i'}}{-\rho_{is}}\right\}\qquad \text{if } i\in\bigcup_{l\in\mathbf{L}_{s*}}\mathbf{I}_l^L$$
$$\text{and}\quad \max_{l\in\mathbf{L}_{s*}}\left\{|\mathbf{S}_i^-\cap\mathbf{S}_l^L|\right\}=1 \tag{15}$$

Equation 15 gives μ_i^R for any task inside the loop except for tasks that consume multiple materials within any loop containing s*, which are discussed in Case 2 of the next subsection. The first term in the condition gives all tasks in any loop containing material s* and the second term gives the number of materials within the loop that are consumed by task i. In the numerator, the minimum amount of material s required for other tasks is subtracted to give the maximum amount of material s available for task i. Dividing by $-\rho_{is}$ gives the total amount task i needs to process to consume all of the available s. Taking the minimum over all materials consumed by task i ensures there is enough of every material. We do not round μ_i^R to an attainable production amount; rounding up will result in a weaker bound, and rounding down may cutoff feasible solutions. Once μ_i^R is known for all tasks producing the labeled recycle material, we find the maximum amount produced by each task

$$\psi_{is*} = \rho_{is*}\mu_i^R \qquad \forall i \in \mathbf{I}_{s*}^+ \tag{16}$$

Finally, we calculate the minimum amount of material s* produced by each task

$$v_{is*} = \max\left\{0, \omega_{s*} - \sum_{i' \in \mathbf{I}_{s*}^+ \setminus \{i\}} \psi_{i's*}\right\} \qquad \forall i \in \mathbf{I}_{s*}^+ \tag{17}$$

The most a task can produce of a recycle material is ψ_{is} and the least is zero. If $v_{is} > \psi_{is}$, there is not enough initial inventory, and the problem is infeasible. If the demand cannot be satisfied without a particular task, Eq. 17 gives the minimum that task must produce. If a single task or combination of tasks can satisfy the demand, all other tasks do not need to supply any material, and the final term in Eq. 17 is less than zero.

Proposition 1. There is a maximum amount of each recycle material that can be produced by each task ψ_{is}, without increasing the minimum required production of other tasks producing that material.

Proof. If ω_s is the minimum amount of recycle material s required to meet final demand, and ψ_{is} is the maximum amount of material s that can be produced by task i when starting with ω_s, then v_{is} is the amount of s produced by task i and

$$v_{is} = \max\left\{0, \omega_s - \sum_{i' \in \mathbf{I}_s^+, i' \neq i} \psi_{i's}\right\} \qquad \forall i \in \mathbf{I}_s^+.$$

Let ζ_{is} be the maximum fraction of material s that is recycled by task i. If an additional amount of material s, $\delta_i \geq 0$, is produced by task i, the total amount of material s now required is at least $\omega_s + \sum_{i \in \mathbf{I}_s^+} \frac{\delta_i}{\zeta_{is}}$, and the maximum amount produced by all tasks is $\sum_{i \in \mathbf{I}_s^+} \psi_{is} + \delta_i$. The total amount of s that needs to be produced by task i is

$$v'_{is} = \max\left\{0, \omega_s + \sum_{i' \in \mathbf{I}_s^+} \frac{\delta_{i'}}{\zeta_{i's}} - \sum_{i' \in \mathbf{I}_s^+, i' \neq i} (\psi_{i's} + \delta_{i'})\right\} \qquad \forall i \in \mathbf{I}_s^+$$

because $\zeta_{is} \leq 1$, $\sum_{i \in \mathbf{I}_s^+} \frac{\delta_i}{\zeta_{is}} \geq \sum_{i \in \mathbf{I}_s^+, i \neq i'} \delta_i$, and $v'_{is} \geq v_{is}$. Therefore, all tasks producing material s are at their minimum production when all other tasks recycle at most ψ_{is}. □

Networks with nested loops

Not all networks with multiple loops need the additional procedures described in this section. There are two cases with multiple loops where the forward propagation will not work.

Case 1. When multiple tasks in a single loop produce another recycle material, the forward propagation described previously fails when calculating ω_s^R for that material. For the network in Figure 10a, S3 is produced by T3 and T2, which both belong to loop L3 containing recycle material S2. We propagate the demand as described previously until it is time to propagate ω_{S2} forward and calculate ω_{S3}^R in Figure 10b. We need μ_{T2}^R and μ_{T3}^R to calculate ω_{S3}^R, but ω_{S3}^R is needed to calculate μ_{T3}^R (Figure 10c). During the forward propagation, T2 produces at most $\mu_{\text{T2}}^R = 120$ kg of S3. Previously (Figure 10b), we determined that T2 must produce a

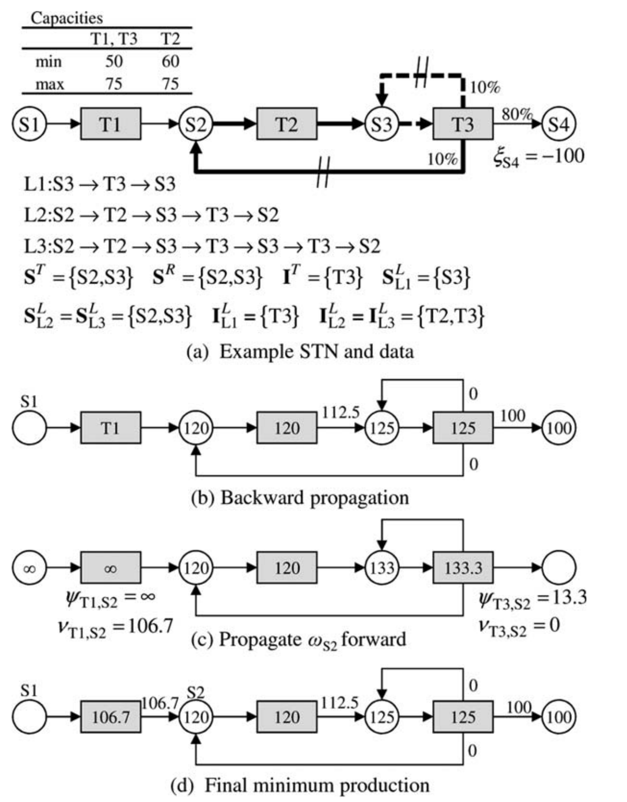

Capacities
T1, T3
T2
min
50
60
max
75
75
S1
T1
S2
T2
S3
T3
S4
10%
80%
10%
$\xi_{\text{S4}} = -100$
L1:S3 → T3 → S3
L2:S2 → T2 → S3 → T3 → S2
L3:S2 → T2 → S3 → T3 → S3 → T3 → S2
$\mathbf{S}^T = \{\text{S2,S3}\}$ $\mathbf{S}^R = \{\text{S2,S3}\}$ $\mathbf{I}^T = \{\text{T3}\}$ $\mathbf{S}_{\text{L1}}^L = \{\text{S3}\}$
$\mathbf{S}_{\text{L2}}^L = \mathbf{S}_{\text{L3}}^L = \{\text{S2,S3}\}$ $\mathbf{I}_{\text{L1}}^L = \{\text{T3}\}$ $\mathbf{I}_{\text{L2}}^L = \mathbf{I}_{\text{L3}}^L = \{\text{T2,T3}\}$
(a) Example STN and data
S1
T1
120
120
112.5
125
125
100
100
0
0
(b) Backward propagation
∞
∞
120
120
133
133.3
$\psi_{\text{T1,S2}} = \infty$
$v_{\text{T1,S2}} = 106.7$
$\psi_{\text{T3,S2}} = 13.3$
$v_{\text{T3,S2}} = 0$
(c) Propagate ω_{S2} forward
S1
106.7
106.7
S2
120
120
112.5
125
125
100
100
0
0
(d) Final minimum production

Figure 10. Example of backward propagation with multiple recycle materials in the same loop.

minimum of $\tilde{\mu}_{\text{T2}} = 112.5$ kg of S3, meaning T2 can now produce an extra 7.5 kg (=120−112.5). If T3 processes all additional 7.5 kg of S3, it will produce another 0.75 kg (=7.5*0.1) of S3. Processing the additional 0.75 kg gives 0.075 kg (=0.75*0.1) more of S3; this is a geometric series that eventually converges to 8.33 kg [=7.5/(1−0.1)] more of S3. We add the extra 8.33 kg to the original 125 kg of S3 needed to meet the demand to give $\omega_{\text{S3}}^R = 133.3$ (Figure 10c). Demand propagation continues until ω_s and $\tilde{\mu}_i$ are known for all materials and tasks (Figure 10d).

The example is generalized to any process network. We will refer to the first labeled recycle material as s* (the one for which we are trying to find v_{is}) and the second recycle material as s** (the one for which we are trying to find ω_s^R). The total amount of material s** available is

$$\omega_s^R = \left(\sum_{i \in \mathbf{I}_s^+ \mathbf{I}^{\text{RNC}}} \left(\rho_{is}^+ \mu_i^R - v_{is}\right)\right)\left(\frac{1}{1-\zeta_s}\right) + \omega_s + \xi_{s0} \qquad \text{if}$$

$$s \in \bigcup_{l \in \mathbf{L}_{s*}^S} \mathbf{S}_l^L \quad \text{and} \max_{l \in \mathbf{L}_{s*}^S} \left\{\left|\mathbf{I}_s^+ \cap \mathbf{I}_s^L\right|\right\} \geq 2 \tag{18}$$

where $\mathbf{I}^{\text{RNC}}$ is the set of tasks for which μ_i^R has not been calculated, and ζ_s is the maximum fraction of material s that can be recycled by the loop containing only tasks in $\mathbf{I}^{\text{RNC}}$. The algorithm provides a method for keeping track of $\mathbf{I}^{\text{RNC}}$. Equation 18 gives ω_s^R for any materials excluded from Eq. 14. The first term is the additional amount available during the forward propagation from all tasks for which μ_i^R has already been calculated. Dividing by $1-\zeta_s$ gives the total additional amount of s** produced during recycling. Finally, the original ω_s and the initial inventory are added to give ω_s^R.

Parameter ζ_s can be estimated when there is only one loop containing s^{**} but not s^*; that is, $l\in\{l: s^{**}\in\mathbf{S}_l^L, s^*\notin\mathbf{S}_l^L\}$

$$\zeta_s=\frac{\prod_{i\in\mathbf{I}_l^L, s'\in\mathbf{S}_i^+\cap\mathbf{S}_l^L}\rho_{is'}}{\prod_{i\in\mathbf{I}_l^L, s'\in\mathbf{S}_i^-\cap\mathbf{S}_l^L}(-\rho_{is'})} \quad \text{for} \quad s\in\mathbf{S}^R \tag{19}$$

The numerator gives the fraction of material remaining in the loop after task i. The denominator is the fraction of the material consumed by task i that is already in the loop. Equation 19 is only valid when there is only one loop containing s** but not s*. Equation 18 is valid for any network, but ζ_s needs to be estimated.

Case 2. When a task inside a loop consumes multiple materials in a loop, the forward propagation fails. In Figure 11, T2 consumes S2 and S3 which both belong to loop L2. In Figure 11b, demand has been backward propagated to calculate the demand for recycle material S2. The demand needs to be propagated forward, but we cannot calculate μ_{T2}^R until ω_{S2}^R and ω_{S3}^R are known, and ω_{S3}^R cannot be calculated until μ_{T2}^R is known (Figure 11c). During the forward propagation, we have $\omega_{\mathrm{S2}}^R=108$ kg of S2 available. If there is enough S3, T2 can process at most 120 kg. We ignore S3 in Eq. 15, and set $\mu_{\mathrm{T2}}^R=120$. We continue propagating demand (Figure 11d).

The procedure is easily generalized to any process network

$$\mu_i^R=\min_{s\in\mathbf{S}_i^-\backslash\mathbf{S}^{\mathrm{RNC}}}\left\{\frac{\omega_s^R+\sum_{i'\in\mathbf{I}_s^-, i'\neq i}\rho_{i's}\tilde{\mu}_{i'}}{-\rho_{is}}\right\}$$
$$\text{if } i\in\bigcup_{l\in\mathbf{L}_{s*}^S}\mathbf{I}_l^L \quad \text{and} \quad \max_{l\in\mathbf{L}_{s*}^S}\left\{|\mathbf{S}_i^-\cap\mathbf{S}_l^L|\right\}\geq 2 \tag{20}$$

where $\mathbf{S}^{\mathrm{RNC}}$ is the set of materials for which ω_s^R has not been calculated; these materials are ignored when calculating μ_i^R. The algorithm provides a way to keep track of $\mathbf{S}^{\mathrm{RNC}}$. Equation 20 gives μ_i^R for any tasks in the loop excluded from Eq. 15.

Complete algorithm

The complete algorithm combines the tightening procedures for all network types. For each of the four categories, the relevant portions of the algorithm in Figure 12 are colored as in Figure 3. For convenience, we define the new sets:

$\mathbf{S}_s^{SL}$=materials in any loop containing material s, $\mathbf{S}_s^{SL}=\bigcup_{l\in\mathbf{L}_s^S}\mathbf{S}_l^L$

$\mathbf{I}_s^{SL}$=tasks in any loop containing material s, $\mathbf{I}_s^{SL}=\bigcup_{l\in\mathbf{L}_s^S}\mathbf{I}_l^L$

$\mathbf{I}^{\mathrm{NC}}/\mathbf{I}^{\mathrm{RNC}}$=tasks for which μ_i /μ_i^R is not known

$\mathbf{I}^A/\mathbf{I}^{RA}$=tasks available ($\omega_s/\omega_s^R$ has been calculated for all $s\in\mathbf{S}_i^+/\mathbf{S}_i^-$) for calculating μ_i/μ_i^R

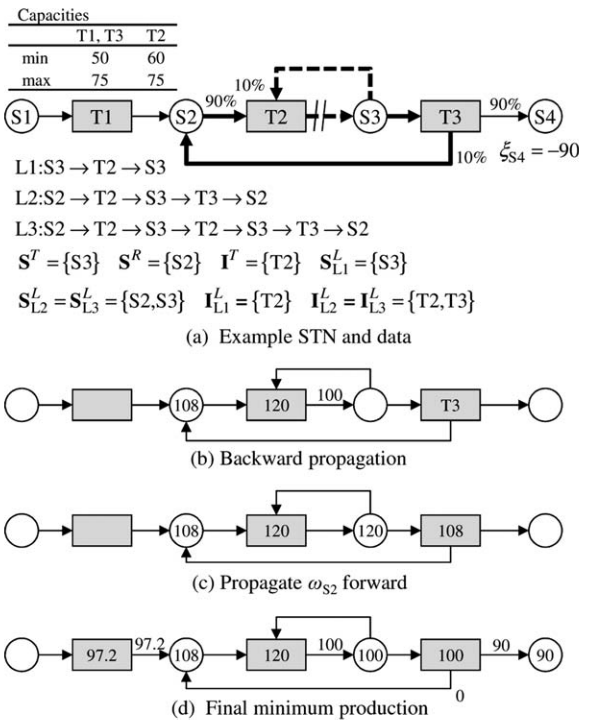
$\mathbf{S}^{\mathrm{NC}}/\mathbf{S}^{\mathrm{RNC}}$ = materials for which ω_s/ω_s^R is not known

$\mathbf{S}^A/\mathbf{S}^{RA}$=materials available ($\mu_i/\mu_i^R$ has been calculated for all $i\in\mathbf{I}_s^-/\mathbf{I}_s^+$) for calculating ω_s/ω_s^R

$\mathbf{S}_i^C$=materials for which v_{is} has been calculated for task i

$\mathbf{S}^{\mathrm{RC}}$=recycle materials for which we need to perform a forward propagation

In the aforementioned algorithm, we use the parameter t_s to keep track of how many times tear material $s\in\mathbf{S}^T$ has been updated. As mentioned previously, if the tear material is updated more than once, the instance is infeasible.



Capacities
T1, T3 T2
min 50 60
max 75 75
S1
T1
S2
T2
S3
T3
S4
10%
90%
90%
10%
$\xi_{S4}=-90$
L1:S3→T2→S3
L2:S2→T2→S3→T3→S2
L3:S2→T2→S3→T2→S3→T3→S2
$\mathbf{S}^T=\{S3\}$ $\mathbf{S}^R=\{S2\}$ $\mathbf{I}^T=\{T2\}$ $\mathbf{S}_{L1}^L=\{S3\}$
$\mathbf{S}_{L2}^L=\mathbf{S}_{L3}^L=\{S2,S3\}$ $\mathbf{I}_{L1}^L=\{T2\}$ $\mathbf{I}_{L2}^L=\mathbf{I}_{L3}^L=\{T2,T3\}$
(a) Example STN and data
108
120
100
T3
(b) Backward propagation
108
120
120
108
(c) Propagate ω_{S2} forward
97.2
97.2
108
120
100
100
100
90
90
0
(d) Final minimum production

Figure 11. Example of backward propagation with a recycle material and multiple loops.

Materials produced by multiple tasks

When multiple tasks can produce a material, Eq. 6 gives v_{is}=0 for each task, which may mean μ_i is zero for upstream tasks that must produce some material. In Figure 13, S4 is produced from T2 or T3, so each task has a minimum production of zero. However, T2 and T3 both require S2, which is produced by T1. If we used Eq. 6 to find v_{is} in Step 4 of the algorithm, we find that μ_{T1}=0, which is not a tight lower bound. Instead, we can solve a simple linear program (LP$_i$) to find v_{is}

$$\begin{aligned}&\min Q_i\\ &\text{s.t.}\quad \xi_{s0}+\sum_{i'\in\mathbf{I}_s^+}\rho_{i's}Q_{i'}\geq-\sum_{i'\in\mathbf{I}_s^-}\rho_{i's}Q_{i'}\forall s \qquad \forall i\\ &\qquad\sum_{i'\in\mathbf{I}_s^+}\rho_{i's}Q_{i'}\geq\omega_s\ \forall s\notin\mathbf{S}^{NC}\end{aligned} \tag{LP$_i$}$$

where Q_i is a positive variable for the total amount of material task i produces in a particular solution of (LP$_i$). The first constraint requires that, for each material, the amount produced plus any initial inventory is greater than the amount consumed. The second constraint enforces that the amount produced of a material must exceed ω_s and is only written for materials for which ω_s is known. The objective is to minimize the amount produced by task i. When it is time to find v_{is} in the algorithm, we solve (LP$_i$) for task i and then multiply the optimal objective value by ρ_{is} to get v_{is}. When a task (other than a tear task) produces multiple materials, we only need to solve (LP$_i$) once and multiply the optimal value by ρ_{is} for each material $s\in\mathbf{S}_i^+$. For tear tasks that produce multiple materials, we solve (LP$_i$) twice, first to calculate v_{is} for any nontear materials produced by the task and second to update v_{is} for the tear material.

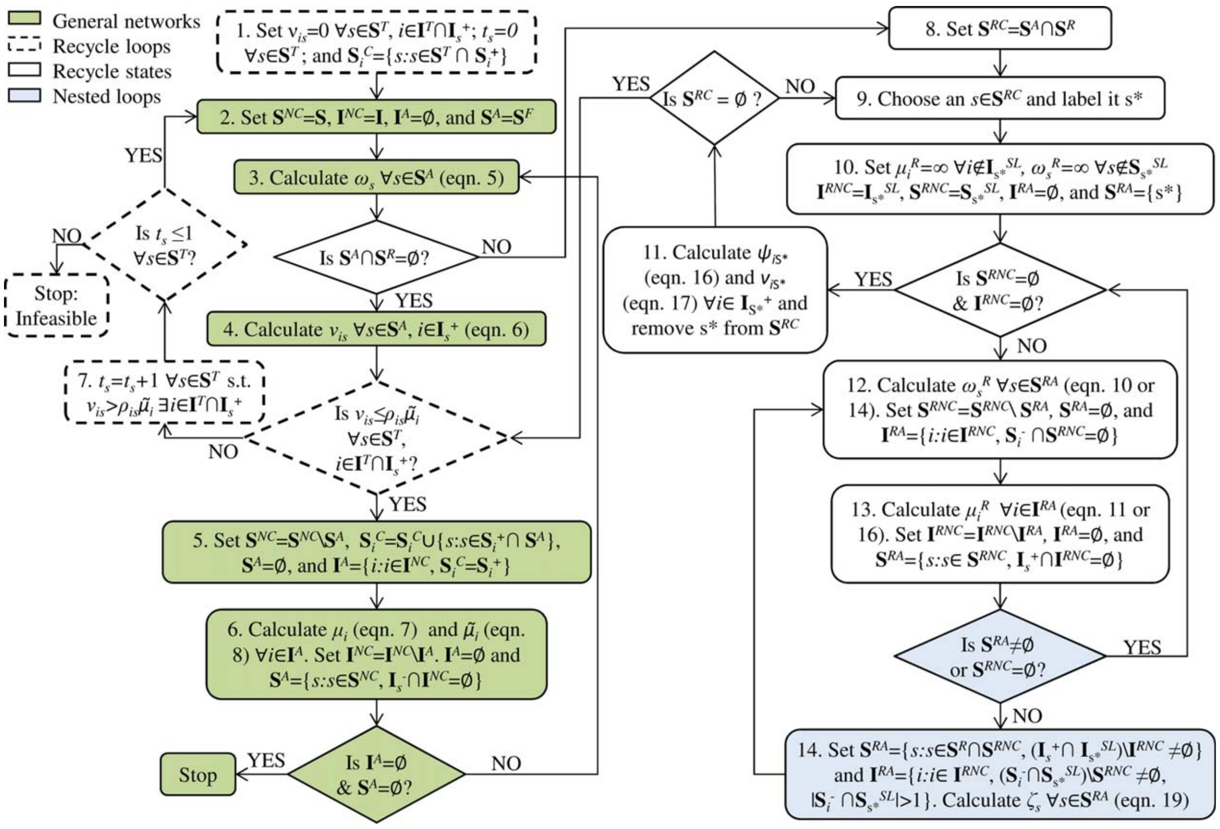

General networks
Recycle loops
Recycle states
Nested loops
1. Set $v_{is}=0$ $\forall s\in\mathbf{S}^T$, $i\in\mathbf{I}^T\cap\mathbf{I}_s^+$; $t_s=0$ $\forall s\in\mathbf{S}^T$; and $\mathbf{S}_i^C=\{s:s\in\mathbf{S}^T\cap\mathbf{S}_i^+\}$
2. Set $\mathbf{S}^{NC}=\mathbf{S}$, $\mathbf{I}^{NC}=\mathbf{I}$, $\mathbf{I}^A=\emptyset$, and $\mathbf{S}^A=\mathbf{S}^F$
3. Calculate ω_s $\forall s\in\mathbf{S}^A$ (eqn. 5)
Is $\mathbf{S}^A\cap\mathbf{S}^R=\emptyset$?
4. Calculate v_{is} $\forall s\in\mathbf{S}^A$, $i\in\mathbf{I}_s^+$ (eqn. 6)
Is $v_{is}\le\rho_{is}\tilde{\mu}_i$ $\forall s\in\mathbf{S}^T$, $i\in\mathbf{I}^T\cap\mathbf{I}_s^+$?
5. Set $\mathbf{S}^{NC}=\mathbf{S}^{NC}\backslash\mathbf{S}^A$, $\mathbf{S}_i^C=\mathbf{S}_i^C\cup\{s:s\in\mathbf{S}_i^+\cap\mathbf{S}^A\}$, $\mathbf{S}^A=\emptyset$, and $\mathbf{I}^A=\{i:i\in\mathbf{I}^{NC},\mathbf{S}_i^C=\mathbf{S}_i^+\}$
6. Calculate μ_i (eqn. 7) and $\tilde{\mu}_i$ (eqn. 8) $\forall i\in\mathbf{I}^A$. Set $\mathbf{I}^{NC}=\mathbf{I}^{NC}\backslash\mathbf{I}^A$. $\mathbf{I}^A=\emptyset$ and $\mathbf{S}^A=\{s:s\in\mathbf{S}^{NC},\mathbf{I}_s^-\cap\mathbf{I}^{NC}=\emptyset\}$
Is $\mathbf{I}^A=\emptyset$ & $\mathbf{S}^A=\emptyset$?
Stop
Is $t_s\le 1$ $\forall s\in\mathbf{S}^T$?
Stop: Infeasible
7. $t_s=t_s+1$ $\forall s\in\mathbf{S}^T$ s.t. $v_{is}>\rho_{is}\tilde{\mu}_i$ $\exists i\in\mathbf{I}^T\cap\mathbf{I}_s^+$
Is $\mathbf{S}^{RC}=\emptyset$?
8. Set $\mathbf{S}^{RC}=\mathbf{S}^A\cap\mathbf{S}^R$
9. Choose an $s\in\mathbf{S}^{RC}$ and label it s*
10. Set $\mu_i^R=\infty$ $\forall i\notin\mathbf{I}_{s*}^{SL}$, $\omega_s^R=\infty$ $\forall s\notin\mathbf{S}_{s*}^{SL}$ $\mathbf{I}^{RNC}=\mathbf{I}_{s*}^{SL}$, $\mathbf{S}^{RNC}=\mathbf{S}_{s*}^{SL}$, $\mathbf{I}^{RA}=\emptyset$, and $\mathbf{S}^{RA}=\{s*\}$
11. Calculate ψ_{is*} (eqn. 16) and v_{is*} (eqn. 17) $\forall i\in\mathbf{I}_{s*}^+$ and remove s* from $\mathbf{S}^{RC}$
Is $\mathbf{S}^{RNC}=\emptyset$ & $\mathbf{I}^{RNC}=\emptyset$?
12. Calculate ω_s^R $\forall s\in\mathbf{S}^{RA}$ (eqn. 10 or 14). Set $\mathbf{S}^{RNC}=\mathbf{S}^{RNC}\backslash\mathbf{S}^{RA}$, $\mathbf{S}^{RA}=\emptyset$, and $\mathbf{I}^{RA}=\{i:i\in\mathbf{I}^{RNC},\mathbf{S}_i^-\cap\mathbf{S}^{RNC}=\emptyset\}$
13. Calculate μ_i^R $\forall i\in\mathbf{I}^{RA}$ (eqn. 11 or 16). Set $\mathbf{I}^{RNC}=\mathbf{I}^{RNC}\backslash\mathbf{I}^{RA}$, $\mathbf{I}^{RA}=\emptyset$, and $\mathbf{S}^{RA}=\{s:s\in\mathbf{S}^{RNC},\mathbf{I}_s^+\cap\mathbf{I}^{RNC}=\emptyset\}$
Is $\mathbf{S}^{RA}\neq\emptyset$ or $\mathbf{S}^{RNC}=\emptyset$?
14. Set $\mathbf{S}^{RA}=\{s:s\in\mathbf{S}^R\cap\mathbf{S}^{RNC},(\mathbf{I}_s^+\cap\mathbf{I}_{s*}^{SL})\backslash\mathbf{I}^{RNC}\neq\emptyset\}$ and $\mathbf{I}^{RA}=\{i:i\in\mathbf{I}^{RNC},(\mathbf{S}_i^-\cap\mathbf{S}_{s*}^{SL})\backslash\mathbf{S}^{RNC}\neq\emptyset, |\mathbf{S}_i^-\cap\mathbf{S}_{s*}^{SL}|>1\}$. Calculate ζ_s $\forall s\in\mathbf{S}^{RA}$ (eqn. 19)
YES
NO

Figure 12. General algorithm for propagating demand through a network.

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

When we use this method, we solve (LP$_i$) to find v_{is} for all tasks in the network. This method also provides an alternative to the methods for recycle materials. For more complicated networks with nested loops, this method may give tighter bounds and may be much simpler to use. Demand is still backward propagated according to the algorithm in Figure 12 with two changes: (1) we calculate v_{is} in Step 4 of the algorithm with (LP$_i$) instead of with Eq. 6, and (2) we skip (or answer yes to) all steps involving recycle materials (Steps 8–14).

Remarks and extensions

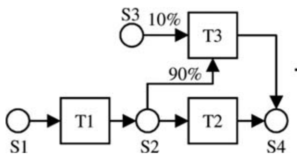
Our algorithm is applicable to all types of network facilities, including facilities with multiple tasks producing or consuming the same material(s) and multiple units capable of performing the same task(s). The only inputs to the algorithm are customer demand, conversion coefficients, unit capacities, and the network structure, which means that the algorithm is applicable to problems with a wide range of characteristics and constraints, including processes with variable processing times, utility constraints, changeovers, and so forth.

The backward propagation methods work for continuous processes as well. We can model a continuous process using a task with a duration of one time period with many of these tasks occurring sequentially while the continuous process is running. If $\theta_{ij}^{\min}/\theta_{ij}^{\max}$ is the minimum/maximum production rate of continuous task i, and Δt is the duration of a period, then the minimum/maximum "batch-size" of task i will be $\theta_{ij}^{\min}\Delta t/\theta_{ij}^{\max}\Delta t$. Thus, if the duration of a run of a continuous task is assumed to be a multiple of a time period, parameters $\theta_{ij}^{\min}\Delta t/\theta_{ij}^{\max}\Delta t$ can be used to calculate, essentially, the minimum total duration a continuous task should be executed. If a continuous task is allowed to run for less than a time period, then the minimum "batch-size" is zero, which means that all batch sizes are attainable and the methods for finding $\tilde{\mu}_i$ should not be applied.

Also, material-based models have been extended to address problems in sequential production environments as well as facilities that combine sequential and network environments.[34] Thus, our methods, which are applicable to all aforementioned models, have the potential to benefit a wide range of tools and models.

Valid Inequalities

We write valid inequalities after calculating $\tilde{\mu}_i$ and ω_s for all tasks and materials. The inequalities only require time-indexed variables and can be written for all types of material-based MIP formulations that use time-indexed variables. We find the minimum number of batches processed by a



Method for v_{is}	$\tilde{\mu}_i$ T1	T2	T3	ω_s S1	S2	S3	S4
Eqn. 2	0	0	0	0	0	0	50
(LP)	45	0	0	45	0	0	50

Figure 13. Comparison of the two methods for finding v_{is}: (1) using Eq. 6 and (2) solving (LP).

Customers demand 50 kg of S4 and all tasks are processed in a unit with a capacity of 0–50 kg.

task λ_i, by dividing the required production by the largest possible size of a single batch of task i and rounding up

$$\lambda_i = \left\lceil \frac{\tilde{\mu}_i}{\max_{j \in \mathbf{J}_i^P} \left\{ \beta_j^{\max} \right\}} \right\rceil \tag{21}$$

Note that for continuous tasks λ_i is the minimum number of time periods the task needs to run. The minimum number of batches provides a lower bound for the sum of the assignment variables

$$\sum_{j \in \mathbf{J}_i^P, t} X_{ijt} \geq \lambda_i \qquad \forall i \tag{22}$$

When multiple tasks produce a material, Eq. 22 may not provide a tight bound. Instead, we find bounds for the minimum number of batches of all tasks producing a material, κ_s. Again, we divide the required amount of each material by the largest possible amount of that material that can be produced in a single batch and round up

$$\kappa_s = \left\lceil \frac{\omega_s}{\max_{i \in \mathbf{I}_s^+, j \in \mathbf{J}_i^P} \left\{ \rho_{is} \beta_j^{\max} \right\}} \right\rceil \tag{23}$$

Now, κ_s provides a bound for the assignment variables for all tasks producing material s

$$\sum_{i \in \mathbf{I}_s^+, j \in \mathbf{J}_i^P, t} X_{ijt} \geq \kappa_s \qquad \forall s \in \mathbf{S}^{MT} \tag{24}$$

Equation 24 does not provide new information for materials produced by a single task. Valid inequalities 22 and 24 have the same form as those proposed by Burkard and Hatzl[29] and Janak and Floudas,[30] but the bounds from the different methods may be different.

When a task can be processed in units with very different capacities, Eq. 22 and 24 may not provide tight bounds. Instead, we use the maximum batch-size and minimum production requirements to find stronger general inequalities

$$\sum_{j \in \mathbf{J}_i^P, t} \beta_j^{\max} X_{ijt} \geq \hat{\mu}_i \qquad \forall i \tag{25}$$

where $\hat{\mu}_i$ is the tightest bound for Eq. 25. Previously, we calculated $\tilde{\mu}_i$ based on $\beta_j^{\min}$. Because Eq. 25 is based on $\beta_j^{\max}$, $\tilde{\mu}_i$ does not provide a tight bound. We find $\hat{\mu}_i$ by performing another loop over all $k \in \{1,2,\ldots, K_i\}$ with the same ε_j^k used to calculate the attainable production amount

$$\hat{\mu}_i^k = \begin{cases} \sum_{j \in \mathbf{J}_i^P} \varepsilon_j^k \beta_j^{\max} & \text{if } \mu_i \leq \sum_{j \in \mathbf{J}_i^P} \varepsilon_j^k \beta_j^{\max} \\ \infty & \text{otherwise} \end{cases} \tag{26}$$

The top expression sets $\hat{\mu}_i^k$ to the production amount at the end of attainable range k if the range is large enough to meet demand. The second expression sets $\hat{\mu}_i^k$ to infinity when the attainable range is too small to meet demand. After checking all ranges, we set $\hat{\mu}_i$ to the smallest of all $\hat{\mu}_i^k$

$$\hat{\mu}_i = \min_k \left\{ \hat{\mu}_i^k \right\} \tag{27}$$

The LHS of Eq. 25 can only take on a discrete set of values when X_{ijt} is binary; these values are given by $\sum \varepsilon_j \beta_j^{\max}$ where ε_j is an integer and are the same as the production amounts at the end of the attainable ranges shown in Figure 7. Therefore, all possible values for the LHS occur at

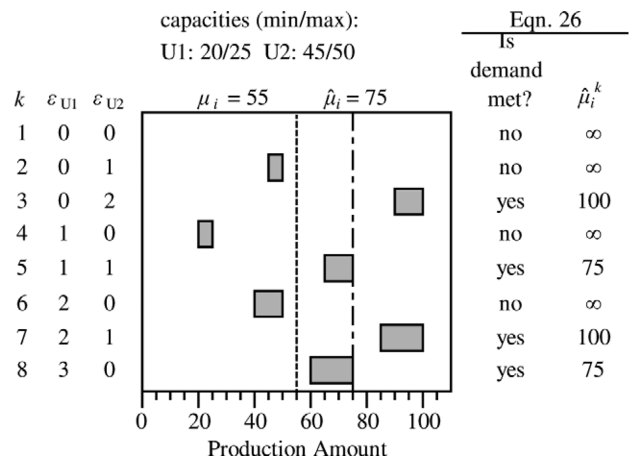


Figure 14. Example of $\hat{\mu}_i$ calculation for the example in Figure 7.

The LHS of Eq. 25 must belong to the set {0, 25, 50, 75…}. Because the minimum production is $\tilde{\mu}_i$=60, the LHS of Eq. 25 must be at least 75 in any feasible solution.

the end of an attainable range. To find the value of $\hat{\mu}_i$ that gives the tightest bound, we use the production amount at the end the attainable region that is closest to but greater than $\tilde{\mu}_i$ (see Figure 14). In general, $\hat{\mu}_i$ is the smallest production amount at the end of an attainable range ($\sum \varepsilon_j^k \beta_j^{\max}$) that is at least able to meet demand ($\tilde{\mu}_i$).

Equation 25 can also be written for materials produced by multiple tasks

$$\sum_{i \in \mathbf{I}_s^+ j \in \mathbf{J}_i^P, t} \rho_{is} \beta_j^{\max} X_{ijt} \geq \omega_s \qquad \forall s \in \mathbf{S}^{MT} \tag{28}$$

Figure 15 illustrates the effect of the valid inequalities on the feasible region of the LP relaxation of the model for the example in Figures 7 and 14. The total demand is 55, μ_i=55, $\tilde{\mu}_i$=60, and $\hat{\mu}_i$=75. The lines show Eqs. 22 and 25 with three RHS values: μ_i, $\tilde{\mu}_i$, and $\hat{\mu}_i$. Equation 22 requires at least two batches. Improving (increasing) the RHS of constraint 25 improves the LP relaxation of the model.

When backlogging is not allowed, we use due times to tighten the formulation further. Now, all parameters include a time index (μ_{st}, ω_{st}, etc.) and are zero for times when no orders are due. For every due time, we add all earlier orders (ξ_{st}) to get φ_{st} and calculate the other parameters as before. Instead of summing the assignment variable over the entire horizon, we sum it only over the time available to start the task

$$\sum_{j \in \mathbf{J}_i^P} \sum_{t'=0}^{t-\tau_{ij}} X_{ijt'} \geq \lambda_{it} \qquad \forall i, t \tag{29}$$

For times when no orders are due, all parameters are zero, and Eq. 29 is not written. We can also write constraints 24, 25, and 28 to consider due times by changing the times over which the assignment variables are summed. For the example in Figure 1, suppose customer C1 demands 30 kg of S3 at t = 6 and 25 kg of S4 at t = 9, and customer C2 demands 60 kg of S3 at t = 9. The total final demands up to t = 6 and up to t = 9 are calculated. Table 2 lists order sizes (φ_{st}), required productions for materials (ω_{st}) and tasks ($\tilde{\mu}_{it}$), and assignment bounds (λ_{it}).

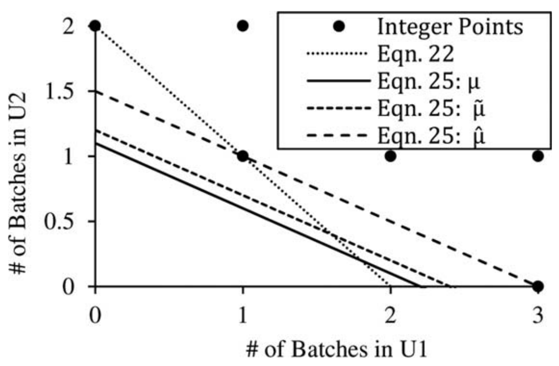


Figure 15. Effect of valid inequalities on feasible region for the example in Figure 7 and 14.

Computational Study

Problem instances

To determine how well the valid inequalities work, we test 72 instances with different objectives, networks, problem features, and time horizons. Each instance is run with different tightening formulations. Specifically, we compare the two types of constraints based on the number of batches (Eqs. 22 and 24) and on the minimum production (Eqs. 25 and 28). We also look at the effectiveness of the extensions for instances with due times.

We consider two objectives: minimization of processing cost and makespan. The processing cost depends on the task and unit, but not on the batch size

$$\text{cost} = \sum_{t,i,j\in \mathbf{J}_i^P} \alpha_{ij} X_{ijt} \tag{30}$$

$$\text{MS} \geq \sum_{i\in \mathbf{I}_j} X_{ijt}\left(t+\tau_{ij}\right) \qquad \forall j \in \mathbf{J}^P, t \tag{31}$$

We use four networks, N1–N4, to determine the effect of recycle streams and materials produced by multiple tasks on the effectiveness of the valid inequalities (Figure S1 in Supporting Information). Process and order data are also given in Supporting Information (Tables S1–S4). Networks N1 and N4 have no recycle streams, but network N4 has a material produced by multiple tasks. Networks N2 and N3 have a recycle stream. Network N2 is taken from Kondili et al.[12]

We consider three problem classes with different features for each network. Problem (a) has all orders due at the end of the time horizon. To determine the impact of including due times in the valid inequalities, problem (b) has intermediate due times. To compare the two types of valid inequalities, unit capacities are changed for problem (c) so at least one task can be processed by units with different capacities. We solve each instance with three time horizons: 40, 80, and 120 h. The order size and due time scale with the time horizon to ensure all time horizons lead to schedules that are similarly loaded. Starting with a 40-h horizon, the order size and due time are doubled for an 80-h horizon and tripled for a 120-h horizon.

Table 2. Parameter Values When due Times are Used to Tighten the Formulation

	Order (ϕ_{st})		ω_{st}	μ_{it}			Assignment Bounds (λ_{it})		
t	S3	S4	S2	T1	T2	T3	T1	T2	T3
6	30	0	35	35	35	0	1	1	0
9	90	25	125	125	90	35	3	2	1

Table 3. Constraint Sets in the Seven Formulations

Set	Minimum No. of Batches (Eqs. 22 and 24)	Minimum Production (Eqs. 25 and 28)	Due Times (Eq. 29)
F1			
F2	X		
F3		X	
F4	X	X	
F5	X		X
F6		X	X
F7	X	X	X

We consider seven sets of valid inequalities (Table 3). All formulations contain Eqs. 1–3. Formulation F1 has no valid inequalities. F2 includes the constraints based on the minimum number of batches. Set F3 contains the constraints based on the required production. F4 combines F2 and F3. Finally, sets F5–F7 are the same as sets F2–F4 but with due times. Only problem (b) is run with formulations F5–F7.

All problems are solved using GAMS 23.7/CPLEX 12.3 on a computer with 6 GB of RAM and a 2.67-GHz Intel Core (i7-920) processor running on Windows 7. We use a resource limit of 1800 s. Model and solution statistics for all problems are given in Supporting Information (Tables S5–S10). For all instances, implementing the demand propagation algorithm takes an average of 0.26 s with a maximum time of 4.3 s when using Eq. 6 to calculate v_{is}, and an average of 2.4 s with a maximum time of 11.9 s when using (LP$_i$) to calculate v_{is}.

Results

Figure 16 is a performance profile for formulations F1–F4. For each instance, the computational time for each formulation is divided by the fastest time over all formulations for that instance to give r. The vertical axis is the probability a formulation will solve a problem within s (on the horizontal axis) times the fastest formulation. For cost minimization, the untightened formulation solves only 17% of the problems within 15 times the fastest formulation but F4 solves 94% of the problems within the same time. Using the valid inequalities based on the minimum production (in either F3 or F4) give the best results. For makespan minimization, F3 performs the best, but tightening is generally less effective and does not always improve solution time.

Figure 17 compares the four formulations for the two objectives. Each point is the average computational time or optimality gap over all networks, problem classes, and time horizons (36 runs per point). The valid inequality based on the number of batches (F2) is the least effective, and F3 and F4 are about equivalent. For cost minimization, tightening increases the fraction of problems solved to optimality and decreases the computational time, whereas for makespan minimization, it does not have a large impact on the computational time. Because makespan minimization appears to be an easier problem and tightening is not as effective, we will focus on cost minimization for the remainder of the section.

Figure 18 compares the four formulations for the four networks averaged over the three problem types and three time horizons (nine runs per point). All tightened formulations perform better than F1, and F3 and F4 are the most effective. When there are recycle streams, as in N2 and N3, F2 does not perform as well. Figure 19 shows results for the three problem

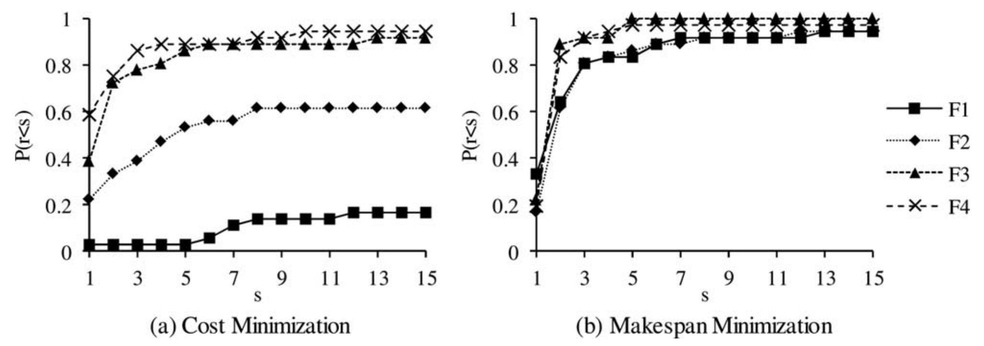


Figure 16. Performance profiles comparing tightening formulations F1–F4.

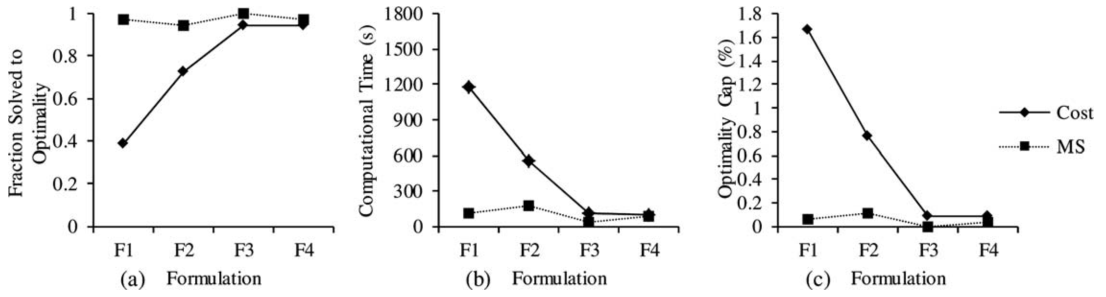


Figure 17. Results for the two objectives.

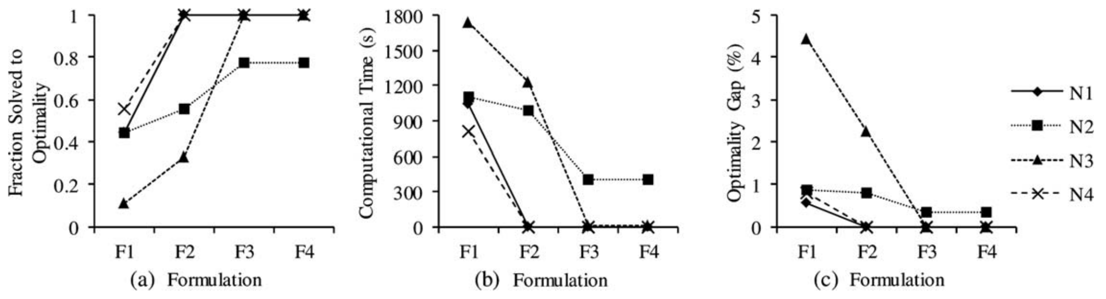


Figure 18. Results for four process networks (cost minimization).

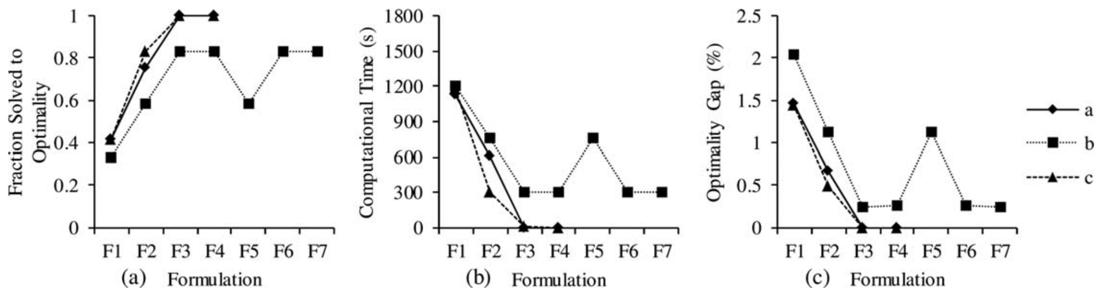


Figure 19. Results for the three problem classes (cost minimization).

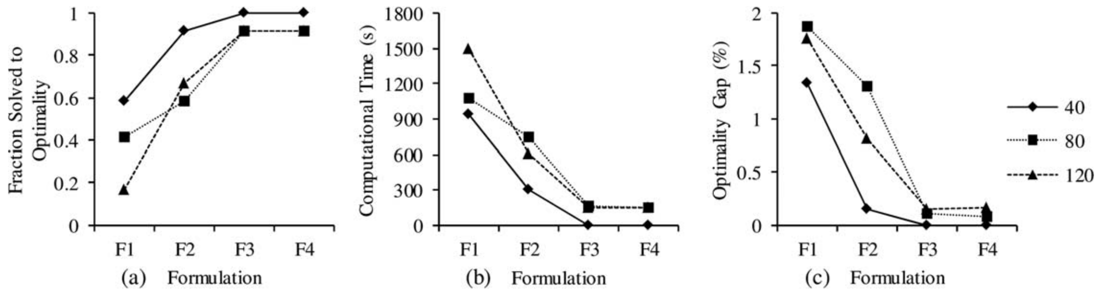


Figure 20. Results for three time horizons (cost minimization).

Table 4. Summary of Tightening Results with Average Computational Time or Optimality Gap

	Cost Minimization				Makespan Minimization			
Category	1	2	3	4	1	2	3	4
No. of problems	14	20	2	0	34	1	0	1
F1	189 s	1.3%	2.1%	–	61 s	2.0%	–	302 s
F4	1.4 s	2.4 s	1.5%	–	32 s	18 s	–	1.1%

classes averaged over the four networks and three time horizons (12 runs per point). For all problem classes, tightening based on the minimum number of batches (F2 and F5) is less effective than using the minimum production requirements (F3 and F6), but using both types of inequalities in F4 and F7 is sometimes better. When the problem includes due times in type (b), adding the due times to the valid inequalities (F5–F7) has little impact on the solution time. Finally, in Figure 20 we show results for the different time horizons, where each point is an average over all problems classes and networks (12 runs per point). As expected, increasing the time horizon decreases the fraction of problems solved to optimality, but using tightening increases the fraction solved so only 8% of the problems solved with a 40-h horizon are not solved with 80- or 120-h horizons. We also note that the average computational requirement and optimality gap decrease notably.

Table 4 shows aggregate results for formulations F1 and F4. Problems are grouped into four categories: (1) those that are solved to optimality by both formulations, (2) those that are solved to optimality only by F4, (3) those that are never solved to optimality, and (4) those that are solved to optimality only by F1. The average computational time or optimality gap is given for the problems in each of the four categories. For cost minimization, 14 problems are in the first category, and tightening reduces the computational time from 189 to 1.4 s. Category 2 contains 20 problems, and the tightened formulation solves these problems in an average of 2.4 s whereas, without tightening, there is an average optimality gap of 1.3% after 1800 s. For the two problems in Category 3, tightening reduces the average optimality gap from 2.1 to 1.5%. There are no problems in the final category. For makespan minimization, 34 problems fall in the first category, and the average computational time is 61 s without tightening and 32 s with tightening. There is one problem each in Categories 2 and 4.

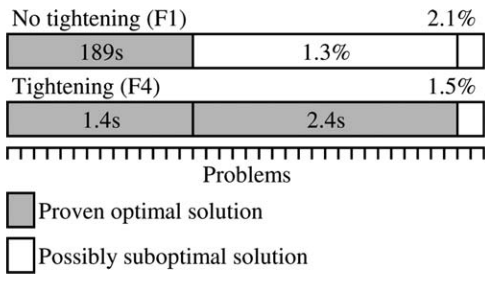


Figure 21. Comparison of average optimality gap and average solution time for formulations F1 and F4.

Model and solution statistics

Table 5 lists model and solution statistics for four representative instances for cost minimization and a 40-h time horizon. F2–F4 only add a moderate number of equations, about one or two per task. F5–F7 add many more constraints, about one or two per task per due time. The objective for the LP relaxation improves as valid inequalities are added and is always best when both types of valid inequalities are used.

Long solution times

Figure 21 presents a summary of the computational results comparing the average optimality gap after 1800 s and the solution time on a problem-by-problem basis for formulations F1 and F4 for cost minimization. For the 14 problems that are solved to optimality with both formulations, the proposed methods reduce the average computational time from 189 to 1.4 s (a speedup of more than two orders of magnitude). Twenty problems that cannot be solved to optimality by F1 within 1800 s (average optimality gap=1.3%) are solved to optimality by F4 with an average computational time of only 2.4 s, which represents a speedup of more than three orders of magnitude.

Importantly, even this type of speedup is an underestimation of the actual enhancement our methods lead to when

Table 5. Model and Solution Statistics for Representative Problems for Cost Minimization and a 40-h Horizon

	Problem 1a				Problem 2b						
Formulation	F1	F2	F3	F4	F1	F2	F3	F4	F5	F6	F7
Discrete vars	230	230	230	230	312	312	312	312	312	312	312
Continuous vars	878	878	878	878	1206	1206	1206	1206	1206	1206	1206
Constraints	1149	1154	1154	1159	1354	1360	1360	1366	1430	1429	1505
LP relaxation	311.14	320	320	320	424.66	430.16	431.66	431.66	441.65	443.56	443.56
Objective	320	320	320	320	490	490	490	490	490	490	490
Nodes	6615	0	0	0	31,996,77	3,441,882	20,019	1668	2,734,482	747	615
CPU time (s)	2.75	0.09	0.09	0.09	1800.02	1800	18.97	3.45	1800.02	2.32	2.39
Gap (%)	0	0	0	0	1.87	1.87	0	0	1.87	0	0
	Problem 3c				Problem 4a						
Discrete vars	427	427	427	427	467	467	467	467			
Continuous vars	1501	1501	1501	1501	1338	1338	1338	1338			
Constraints	1846	1854	1854	1862	1723	1731	1731	1739			
LP relaxation	253.46	294.43	305.67	309	328.58	355.92	352.13	357.58			
Objective	310	310	310	310	380	380	380	380			
Nodes	2,830,545	56,924	0	0	2,668,183	502	0	0			
CPU time (s)	1251.22	68.89	0.09	0.09	1800	0.64	0.09	0.09			
Gap (%)	0	0	0	0	2.54	0	0	0			

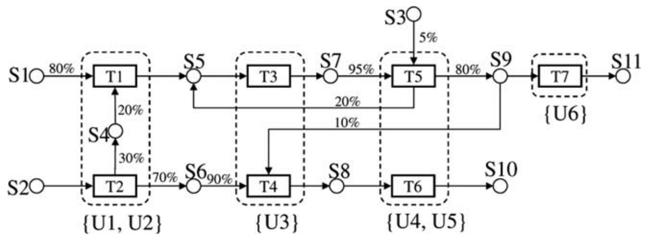


Figure 22. Network for the instance that is solved with a long solution time.

applied to hard instances. To illustrate the point, we solved an instance of Network 3 with a 120-h time horizon using formulation F1 with a resource limit of 16 h (57,600 s) (Figure 22). After 16 h, the optimality gap was 2.22%. The same instance is solved to optimality in 5.8 s using F4, a four orders of magnitude improvement. We observed that many of the instances that did not solve using F1 within the 1800 s resource limit require a much longer time to solve.

Conclusions

We presented methods for the effective solution of chemical production scheduling problems using time-indexed MIP models. We developed a demand propagation algorithm for the calculation of parameters that are used to generate valid inequalities. Specifically, we calculate lower bounds on (1) the number of batches of each task in any feasible solution, (2) the number of batches of tasks producing a material, (3) the total amount processed by one task, and (4) the amount of each material that has to be produced. The algorithm exploits minimum batch-size restrictions (variable lower bound constraints), which are common in chemical manufacturing, but have received limited attention in the literature. Furthermore, it is applicable to facilities with recycling of material, a class of problems that cannot be addressed using more standard methods based on bill-of-material information. Most importantly, the demand propagation algorithm has minimal computational requirements (less than 1 min of CPU time), even for complex large-scale manufacturing facilities. Using the aforementioned bounds, we generated four types of valid inequalities, which lead to dramatic computational improvements. The new valid inequalities based on the minimum production are more effective than those based on the number of batches, while using both types of constraints leads to the most effective solution. We also observed that the valid inequalities are especially effective for longer time horizons. The proposed methods, which are applicable to all time-indexed MIP scheduling models for chemical manufacturing, can potentially lead to significant computational improvements in a wide range of commercial optimization-based tools.

Notation

Sets

Indices/Sets

$i,i' \in \mathbf{I}$ =	tasks
$j \in \mathbf{J}$ =	processing units
$s,s' \in \mathbf{S}$ =	materials
$t,t' \in \mathbf{T}$ =	time points
k =	range
$l \in \mathbf{L}$ =	loops

Subsets

Sets for any network:

$\mathbf{I}_s^+/\mathbf{I}_s^-$ =	tasks producing/consuming material s
$\mathbf{I}_j$ =	tasks that can be performed by processing unit j
$\mathbf{J}_i^P$ =	processing units that can process task i
$\mathbf{S}^{ST}$ / $\mathbf{S}^{MT}$ =	materials produced by a single/multiple tasks
$\mathbf{S}_i^+/\mathbf{S}_i^-$ =	materials produced/consumed by task i
$\mathbf{S}^F$ =	final products

Sets for networks with loops:

$\mathbf{I}_l^L/\mathbf{S}_l^L$ =	tasks/materials in loop l
$\mathbf{I}^T/\mathbf{S}^T$ =	tear tasks/materials
$\mathbf{S}^R$ =	recycle materials (materials in a loop that are produced by multiple tasks)
$\mathbf{L}_s^S$ =	loops containing material s

Sets used in the algorithm:

$\mathbf{S}_s^{SL}$ =	materials in any loop containing material s
$\mathbf{I}_s^{SL}$ =	tasks in any loop containing material s
$\mathbf{I}^{NC}/\mathbf{I}^{RNC}$ =	tasks for which μ_i/μ_i^R is not known
$\mathbf{I}^A/\mathbf{I}^{RA}$ =	tasks available for calculating μ_i/μ_i^R
$\mathbf{S}^{NC}/\mathbf{S}^{RNC}$ =	materials for which ω_s/ω_s^R is not known
$\mathbf{S}^A/\mathbf{S}^{RA}$ =	materials that are available for calculating ω_s/ω_s^R
$\mathbf{S}_i^C$ =	materials for which v_{is} has been calculated for task i
$\mathbf{S}^{RC}$ =	recycle materials for which we need to perform a forward propagation

Parameters

Problem data:

α_{ij} =	cost of carrying out task i in unit j
$\beta_j^{\min}/\beta_j^{\max}$ =	minimum/maximum capacity of unit j
$\gamma_s^{\max}$ =	maximum inventory of material s
ξ_{st} =	amount of raw material s delivered at time t (>0) or customer demand for material s at time t (<0)
ρ_{is} =	fraction of material s produced (>0) or consumed (<0) by task i
τ_{ij} =	processing time for task i in unit j

Parameters calculated by tightening methods:

$\Delta\mu_i$ =	minimum amount added to μ_i to reach an attainable production amount
ε_j^k =	number of batches in unit j for range k
$\varepsilon_j^{\max}$ =	maximum number of batches in unit j needed to satisfy final demand
ζ_s =	maximum fraction of material s that can be recycled
κ_s =	lower bound on the number of batches of tasks producing material s
λ_i =	lower bound on the number of batches of task i
$\mu_i/\tilde{\mu}_i$ =	lower bound on the production of task i required to meet final demand
μ_i^R =	upper bound on the production of task i when only the minimum amount of a recycle material is available
v_{is} =	lower bound on the amount of material s that must be produced by task i
ϕ_{st} =	total amount of material s due up to time t
ψ_{is} =	upper bound on the amount of recycle material s produced by task i when only the minimum amount of material s is available
ω_s =	lower bound on the amount of material s required to meet final demand
ω_s^R =	upper bound on the amount of material s that can be produced when only the minimum amount of a recycle material is available

Variables

Binary variables:

X_{ijt} =	is one if unit j processes task i starting at time t

Continuous nonnegative variables:

B_{ijt} =	batch-size of task i processed in unit j starting at time t
S_{st} =	inventory of material s at time t
Q_i =	total amount of material task i produces in a particular solution of (LP).

Acknowledgment

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

Literature Cited

1. Mendez 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(6–7):913–946.
2. Maravelias CT. General framework and modeling approach classification for chemical production scheduling. *AICHE J*. 2012;58(6):1812–1828.
3. Egli UM, Rippin DWT. Short-term scheduling for multiproduct batch chemical plants. *Comput Chem Eng*. 1986;10(4):303–325.
4. Reklaitis GV. Review of Scheduling of Process Operation. *AIChE Symp. Ser* 1982;78(214):119–133.
5. Pinto JM, Grossmann IE. Assignment and sequencing models for the scheduling of process systems. *Anna Oper Res*. 1998;81:433–466.
6. Mendez CA, Cerda J. An MILP Continuous-time framework for short-term scheduling of multipurpose batch processes under different operation strategies. *Optim Eng*. 2003;4(1–2):7–22.
7. 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(4–6):701–711.
8. Prasad P, Maravelias CT. Batch selection, assignment and sequencing in multi-stage multi-product processes. *Comput Chem Eng*. 2008;32(6):1106–1119.
9. Sundaramoorthy A, Maravelias CT. Simultaneous batching and scheduling in multistage multiproduct processes. *Ind Eng Chem Res*. 2008;47(5):1546–1555.
10. Sundaramoorthy A, Maravelias CT. Modeling of storage in batching and scheduling of multistage processes. *Ind Eng Chem Res*. 2008;47(17):6648–6660.
11. Sundaramoorthy A, Maravelias CT, Prasad P. Scheduling of multistage batch processes under utility constraints. *Ind Eng Chem Res*. 2009;48(13):6050–6058.
12. Kondili E, Pantelides CC, Sargent RWH. A general algorithm for short-term scheduling of batch-operations .1. Milp formulation. *Comput Chem Eng*. 1993;17(2):211–227.
13. Shah N, Pantelides CC, Sargent RWH. A general algorithm for short-term scheduling of batch-operations .2. Computational issues. *Comput Chem Eng*. Feb 1993;17(2):229–244.
14. Pantelides CC. Unified frameworks for optimal process planning and scheduling. In: Second Conference on Foundations of Computer Aided Process Operations, 1994, Snowmass, CO, CACHE Publications.
15. 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(6–7):897–904.
16. Schilling G, Pantelides CC. A simple continuous-time process scheduling formulation and a novel solution algorithm. *Comput Chem Eng*. 1996;20:S1221–S1226.
17. Ierapetritou MG, Floudas CA. Effective continuous-time formulation for short-term scheduling. 1. Multipurpose batch processes. *Ind Eng Chem Res*. 1998;37(11):4341–4359.
18. Mockus L, Reklaitis GV. Continuous time representation approach to batch and continuous process scheduling. 1. MINLP formulation. *Ind Eng Chem Res*. 1999;38(1):197–203.
19. Castro P, Barbosa-Povoa APFD, Matos H. An improved RTN continuous-time formulation for the short-term scheduling of multipurpose batch plants. *Ind Eng Chem Res*. 2001;40(9):2059–2068.
20. 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(13):3056–3074.
21. 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.
22. Maravelias CT. Mixed-time representation for state-task network models. *Ind Eng Chem Res. Nov* 2005;44(24):9129–9145.
23. Giannelos NF, Georgiadis MC. A simple new continuous-time formulation for short-term scheduling of multipurpose batch processes. *Ind Eng Chem Res*. 2002;41(9):2178–2184.
24. 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(9):1511–1528.
25. Pochet Y, Warichet F. A tighter continuous time formulation for the cyclic scheduling of a mixed plant. *Comput Chem Eng*. 2008;32(11):2723–2744.
26. Maravelias CT, Papalamprou K. Polyhedral results for discrete-time production planning MIP formulations for continuous processes. *Comput Chem Eng*. 2009;33(11):1890–1904.
27. Maravelias CT. On the combinatorial structure of discrete-time MIP formulations for chemical production scheduling. *Comput Chem Eng*. 2012;38:204–212.
28. Maravelias CT, Grossmann IE. Minimization of the makespan with a discrete-time state-task network formulation. *Ind Eng Chem Res*. 2003;42(24):6252–6257.
29. Burkard RE, Hatzl J. Review, extensions and computational comparison of MILP formulations for scheduling of batch processes. *Comput Chem Eng*. 2005;29(8):1752–1769.
30. 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(4–5):913–955.
31. Kelly JD, Zyngier D. Hierarchical decomposition heuristic for scheduling: coordinated reasoning for decentralized and distributed decision-making problems. *Comput Chem Eng*. 2008;32(11):2684–2705.
32. Ferris M, Maravelias CT, Sundaramoorthy A. Simultaneous batching and scheduling using dynamic decomposition on a grid. *INFORMS J Comput*. 2009;21(3):398–410.
33. Sundaramoorthy A, Maravelias CT. Computational study of network-based mixed-integer programming approaches for chemical production scheduling. *Ind Eng Chem Res*. 2011;50(9):5023–5040.
34. Sundaramoorthy A, Maravelias CT. A general framework for process scheduling. *AICHE J*. 2011;57(3):695–710.
35. Kelly JD, Mann JL. Crude oil blend scheduling optimization: an application with multimillion dollar benefits—Part 2—The ability to schedule the crude oil blendshop more effectively provides substantial downstream benefits. *Hydrocarbon Process*. 2003;82(7):72–79.
36. Kelly JD, Zyngier D. Multi-product inventory logistics modeling in the process industries In: Chaovalitwongse W, Furman KC, Pardalos PM, editors. Optimization and Logistics Challenges in the Enterprise, New York: Springer, 2009;61–95.
37. Wassick JM. Enterprise-wide optimization in an integrated chemical complex. *Comput Chem Eng*. 2009;33(12):1950–1963.
38. Pochet Y, Wolsey LA. Production Planning by Mixed Integer Programming. New York, Berlin: Springer, 2006.
39. Wolsey LA. Valid inequalities for 0-1 Knapsacks and Mips with generalized upper bound constraints. *Discrete Appl Math*. 1990;29(2–3):251–261.
40. van den Akker JM, van Hoesel CPM, Savelsbergh MWP. A polyhedral approach to single-machine scheduling problems. *Math Program*. 1999;85(3):541–572.
41. van den Akker JM, Hurkens CAJ, Savelsbergh MWP. Time-indexed formulations for machine scheduling problems: Column generation. *INFORMS J Comput*. 2000;12(2):111–124.
42. Hooker J. Logic-Based Methods for Optimization : Combining Optimization and Constraint Satisfaction. New York: John Wiley & Sons, 2000.
43. Baptiste P, Le Pape C, Nuijten W. Constraint-Based Scheduling : Applying Constraint Programming to Scheduling Problems. Boston: Kluwer Academic, 2001.
44. Van Hentenryck P, Michel L. Constraint-Based Local Search. Cambridge, Mass: MIT Press, 2005.
45. Hooker JN. Planning and scheduling by logic-based benders decomposition. *Oper Res*. 2007;55(3):588–602.
46. Pochet Y, Wolsey LA. Lot-sizing with constant batches—formulation and valid inequalities. *Math Oper Res*. 1993;18(4):767–785.
47. Wolsey LA. MIP modelling of changeovers in production planning and scheduling problems. *Eur Jf Oper Res*. 1997;99(1):154–165.
48. Miller AJ, Wolsey LA. Tight formulations for some simple mixed integer programs and convex objective integer programs. *Math Program*. 2003;98(1–3):73–88.
49. Nemhauser GL, Wolsey LA. Integer and Combinatorial Optimization. New York: Wiley, 1988.
50. Wolsey LA. Integer Programming. New York: Wiley, 1998.

Manuscript received Oct. 5, 2012, and revision received Dec. 11, 2012.