AIChE Letter: Process Systems Engineering

Tightening Methods for Continuous-Time Mixed-Integer Programming Models for Chemical Production Scheduling

Andres F. Merchan, Sara Velez, and Christos T. Maravelias
Dept. of Chemical and Biological Engineering, University of Wisconsin-Madison, 1415 Engineering Dr., Madison, WI 53706

DOI 10.1002/aic.14249
Published online October 30, 2013 in Wiley Online Library (wileyonlinelibrary.com)

Significance

Important advances in modeling chemical production scheduling problems have been made in recent years, yet effective solution methods are still required. We use an algorithm that uses process network and customer demand information to formulate powerful valid inequalities that substantially improve the solution process. In particular, we extend the ideas recently developed for discrete-time formulations to continuous-time models and show that these tightening methods lead to a significant decrease in computational time, up to more than three orders of magnitude for some instances. © 2013 American Institute of Chemical Engineers *AIChE J*, 59: 4461–4467, 2013

Keywords: Mixed-integer programming, demand propagation, valid inequalities

Introduction

Research efforts in the area of chemical production scheduling have been primarily focused on the development of alternative formulations to ensure both generality and computational efficiency.[1,2] Starting from the work of Pantelides and coworkers,[3–5] several general mixed-integer programming (MIP) models, relying on material balances and time grids, have been proposed to (1) reduce computational requirements,[6–14] and (2) account for constraints on storage, utilities, changeovers, connectivity, material transfers, material-handling restrictions, and combined production environments.[15–19] Nevertheless, significantly less attention has been directed to solution methods for general MIP scheduling models.

To address the computational challenge, various researchers have studied the structure of MIP chemical production scheduling models,[20,21] used decomposition-based algorithms,[22–27] exploited parallel computing tools,[28,29] developed reformulations,[30–32] and developed tightening methods based on valid inequalities.[31,33–35]

An example of the latter is the work of Maravelias and coworkers[35,36] where process network information (recipes and unit capacities) and customer demand are used to calculate a set of parameters (the minimum amount of each material and the minimum production goal for each task that are required to meet the given demand), which are then used to generate strong valid inequalities. This strategy was implemented in discrete-time models leading to dramatic computational time reductions, up to four orders of magnitude for many instances. Discrete-time models have a number of advantages, including (1) the linear modeling of inventory and utility costs, (2) the treatment of intermediate raw material deliveries and final product orders at no additional computational cost, (3) the straightforward modeling of events during the execution of tasks, and (4) the modeling of time-varying utility availability and pricing using no new variables or constraints. Furthermore, a recent study showed that discrete-time models are in general faster than their continuous-time counterparts.[37] Nevertheless, continuous-time models are likely to be preferred in some specific problems (e.g., in problems with sequence-dependent changeovers), so it is important to develop methods that enhance the solution of this type of models. Accordingly, the purpose of this research note is to extend the aforementioned methods[36] from discrete-time to continuous-time formulations and assess their effectiveness.

Background

Problem statement

In the process systems engineering literature, chemical production scheduling refers to the problem of finding an optimal schedule to operate a facility that transforms raw

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

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

© 2013 American Institute of Chemical Engineers

Table 1. Sets and Parameters for STN Representation

Sets	Parameters	
$\mathbf{I}=\{i : i \text{ is a task}\}$	τ^F_{ij}/τ^V_{ij}	Fixed/variable components of the processing time for task $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$
$\mathbf{J}=\{j : j \text{ is a unit}\}$		
$\mathbf{J}_i=\{j \in \mathbf{J} : \text{unit } j \text{ can process task } i\}$	$\beta^{max}_j/\beta^{min}_j$	Maximum/minimum batch size for unit $j \in \mathbf{J}$
$\mathbf{K}=\{k : k \text{ is a material}\}$		
$\mathbf{I}^+_k=\{i \in \mathbf{I} : \text{task } i \text{ produces material } k\}$	γ_k	Storage capacity for material $k \in \mathbf{K}$
$\mathbf{I}^-_k=\{i \in \mathbf{I} : \text{task } i \text{ consumes material } k\}$	ρ_{ik}	Fraction of material $k \in \mathbf{K}$ produced (>0) or consumed (<0) by task $i \in \mathbf{I}$

materials into added value products by executing a set of tasks in processing units. In order to define such a schedule, it is necessary to determine the number and size of the batches for each task, as well as the assignment of batches to units, and the timing of those batches. The systematic study of the scheduling problem requires an abstract representation that includes the different elements of the chemical facility. In this work, we use the state-task network (STN) representation,[3] in which nodes correspond to tasks and materials (states), whereas arcs identify streams that define the consumption or production of a given material. Unit compatibility and other resources are defined via implicit mappings. The main assumptions we make in this work are (1) no material handling restrictions (i.e., batch integrity requirement that prevents mixing/splitting) are present, (2) a task cannot be interrupted once it has started (no preemption), and (3) the problem data are deterministic and fixed over the scheduling horizon. The sets and parameters required to model a chemical facility through the STN representation under the assumptions introduced above are summarized in Table 1. We use uppercase italics for variables, uppercase bold letters for sets, lowercase italics for indices, and lowercase Greek letters for parameters.

Continuous-time models

In general, a material-based global continuous-time model is based on a grid with $(N-1)$ time periods of unknown length that are shared by all units. The task-unit assignment is mapped onto this grid using the set of time points $\mathbf{N}=\{n \in \mathbb{Z} : 1 \le n \le N\}$. The main binary variables used denote starting/finishing (X_{ijn}/Y_{ijn}) of a task $i \in \mathbf{I}$ in unit $j \in \mathbf{I}$ at time $n \in \mathbf{N}$. A given model has to enforce three processing constraints, namely, equipment utilization, material balances, and capacity constraints. Most models enforce the second requirement through a network flow balance constraint and the third by essentially modeling the batch-size variable as semicontinuous. However, the assignment constraint allows for different strategies to be used; in discrete-time formulations a clique constraint is used,[4] whereas in continuous-time models, different approaches have been tried. This work includes three representative global continuous-time models, with the aim of studying the effect of different formulations on computational performance. The models are taken from the works of Maravelias and Grossman,[12] Sundaramoorthy and Karimi,[14] and Giménez et al.[16] However, we expect that similar enhancements can be achieved when these methods are applied to any continuous-time model. Complete formulations for each model are included online in the Supporting Information.

Since our main goal is to use the demand-based algorithm of Velez et al.[36] two types of objective functions are studied, makespan and cost minimization

$$\min MS \tag{1}$$

$$\min \sum_{i\in\mathbf{I}}\sum_{j\in\mathbf{J}_i}\sum_{n\in\mathbf{N}}\left(\alpha^F_{ij}X_{ijn}+\alpha^V_{ij}BS_{ijn}\right) \tag{2}$$

where $MS \in \mathbb{R}_+$ represents the makespan, $BS_{ijn} \in \mathbb{R}_+$ denotes the batch size of task $i \in \mathbf{I}$ beginning in unit $j \in \mathbf{J}_i$ at time $n \in \mathbf{N}$, and $\alpha^F_{ij}/\alpha^V_{ij}$ are the fixed/variable components of the processing cost for running task $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$. Demand requirements are met by enforcing constraint (3)

$$S_{kN} \ge \xi_k \qquad \forall k \in \mathbf{K} \tag{3}$$

where $S_{kN} \in \mathbb{R}_+$ is a continuous variable that represents the inventory level of material $k \in \mathbf{K}$ at point time $n \in \mathbf{N}\cup\{0\}$, and ξ_k is the demand of material $k \in \mathbf{K}$ at the end of the horizon. Intermediate due dates can also be handled by this approach.[36]

Backward Propagation Algorithm and Valid Inequalities

The proposed methodology consists of (1) using the network structure, unit capacities, recipes and demand information to calculate four parameters, and (2) defining valid inequalities that use these parameters to tighten the original formulations.[36] The first parameter ω_k, defines the minimum amount of material k that is required to meet the given demand. The second parameter μ_i, gives the minimum production that task i needs to yield in order to satisfy customer demand. A third parameter κ_k, is introduced to represent the minimum number of batches required to produce material k. Finally, a fourth parameter λ_i, defines the minimum number of batches of task i. Figure 1 illustrates how the available information is used to calculate the parameters involved in the tightening constraints. Figure 2 shows how the demand information is propagated backwards to calculate the important parameters for a small illustrative network.

Two different types of tightening constraints can be defined. The minimum number of batches processed by a given task provides a lower bound for the sum of assignment variables as defined by Eq. 4. When a task is carried out in units with different capacities, then Eq. 5, based on the minimum production requirement for a given task, leads to tighter formulations

$$\sum_{j\in\mathbf{J}_i}\sum_{n\in\mathbf{N}}X_{ijn} \ge \lambda_i \qquad \forall i \in \mathbf{I} \tag{4}$$

$$\sum_{j\in\mathbf{J}_i}\sum_{n\in\mathbf{N}}\beta^{\max}_j X_{ijn} \ge \mu_i \qquad \forall i \in \mathbf{I} \tag{5}$$

When a material is produced by multiple tasks, Eq. 6 lower-bounds the number of all batches producing material k,

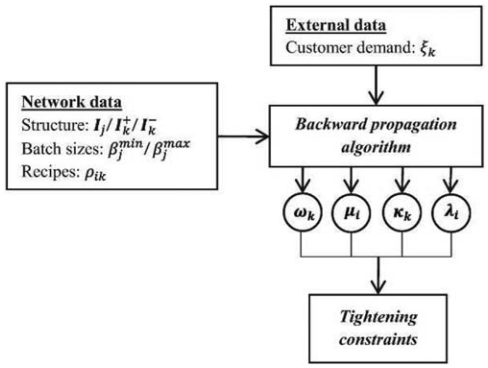

Figure 1. Simplified flow chart for tightening methods of Velez et al.[36]

It shows the different parameters that are calculated and how they define valid inequalities.

and if these tasks are carried out in units with different capacities, Eq. 7 provides additional tightening

$$\sum_{i\in\mathbf{I}_k^+}\sum_{j\in\mathbf{J}_i}\sum_{n\in\mathbf{N}} X_{ijn} \geq \kappa_k \qquad \forall k \in \mathbf{K}^{MT} \tag{6}$$

$$\sum_{i\in\mathbf{I}_k^+}\sum_{j\in\mathbf{J}_i}\sum_{n\in\mathbf{N}} \rho_{ik}\beta_j^{\max} X_{ijn} \geq \omega_k \qquad \forall k \in \mathbf{K}^{MT} \tag{7}$$

where $\mathbf{K}^{MT} = \{k \in \mathbf{K}: k \text{ is produced by multiple tasks}\}$.

The tightening constraints defined previously can be used separately or combined. In this work, we define three distinct formulations. Formulation 1 only uses Eqs. 4 and 6 based on the minimum number of batches; formulation 2 only uses Eqs. 5 and 7 based on minimum production requirements; and formulation 3 uses all four equations. Other combinations of equations proved to be ineffective to tighten the original formulation. In addition, Eqs. 4–7 are written for the starting binary variable X_{ijn}, but it is also possible to define the tightening constraints by replacing X_{ijn} with the finishing binary variable Y_{ijn}. Moreover, if we assume that every task that starts must finish during the scheduling horizon, the same constraint can be written twice, once for X_{ijn} and once for Y_{ijn}. In this work, we distinguish these possibilities as *variants* of a given formulation. Variant A corresponds to using X_{ijn} variant B uses Y_{ijn} and variant C uses both.

Computational Study

Problems, formulations, instances and runs

We follow the general ideas presented in Sundaramoorthy and Maravelias[37] to define, classify and identify every computational run in this study.

A *problem* type is defined as a combination of an objective function OBJ and a specific feature FEAT and is represented by the pair [OBJ].[FEAT]. As stated earlier, we consider makespan (MS), and cost (CT) minimization. The two features in which we are interested are constant and variable processing times (CPT, VPT). We consider four different problems: MS.CPT, MS.VPT, CT.CPT and CT.VPT.

A *problem formulation* is a unique MIP representation of a given problem, once the model (MOD), tightening constraints and variant have been selected. For each of the three continuous-time models under study (M&G, S&K, GH&M), we consider 10 problem formulations, which include the original, identified as F0, and the combinations of valid inequalities and variants described above, represented by F#X, where # corresponds to the formulation (1–3) and X defines the variant (A#C). The representation for a problem formulation is therefore [MOD].[F#X].[OBJ].[FEAT]. In total, we study 120 problem formulations.

An *instance* of a formulation is obtained by fixing the process network (PN#) and the horizon (H#). Four different networks found in literature, some of them modified, are considered: PN1[3], PN2 and PN4[38], and PN3[19]. In all the instances we assume that material transfer is instantaneous, each material has a dedicated storage vessel and there are no changeovers or utility requirements. Any of these assumptions can be removed and the proposed methods will still be valid. Figures of the process networks and data can be found in the Supporting Information. We consider horizon values of 24 and 36 h. A total of 960 instances are studied and labeled as [MOD].[F#X].[OBJ].[FEAT].[PN#].[H#].

A *run* of a particular instance is an attempt to solve it once we determine the optimal number of time points N^*, defined as the first of three consecutive values for which the optimal value of the objective function remains unchanged. In this work, this number was determined iteratively using a trial formulation. We include runs with ($N^* + 1$) and ($N^* + 2$) points to further analyze the behavior of a given formulation. A

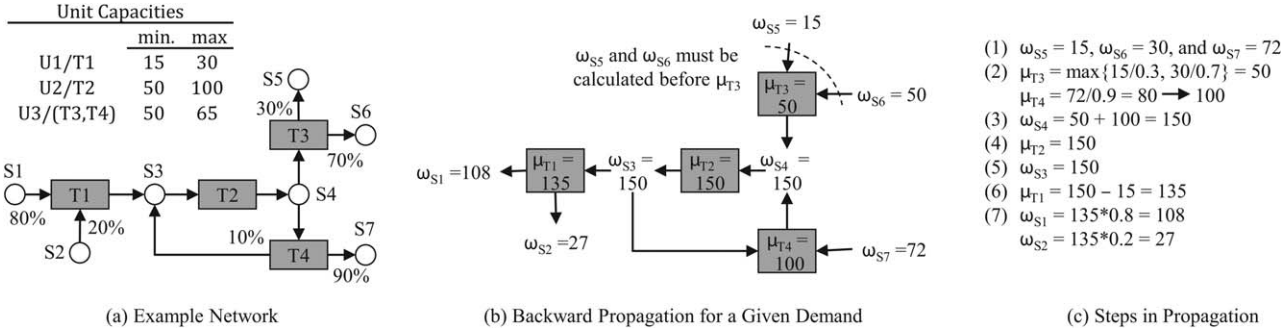

Figure 2. Backward propagation algorithm applied to a small network with recycle of material.

The steps in the propagation as follows: (1) Customer demands are given for final products. (2) T3 must produce enough of materials S5 and S6; T4 must produce enough of material S7 (at least 80), but unit U3 can produce 50-60 in 1 batch, 100-130 in 2 batches, and so on; therefore, μ_{T4} is increased from 80 to 100. (3) S4 is needed for both T3 and T4. (4) T2 must produce enough of material S4. (5) S3 is needed by T2. (6) T4 can supply up to 15 (= 150*0.1) of S3, so the remaining comes from T1. (7) S1 and S2 are needed by T1.

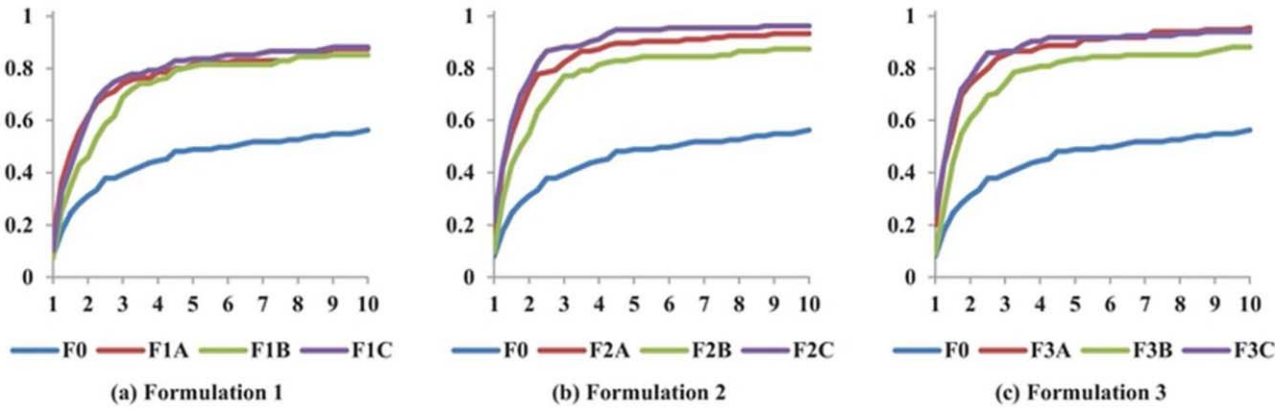

Figure 3. Performance charts for all four problems {MS.CPT, MS.VPT, CT.CPT, CT.VPT}.

Includes original formulation (F0) and proposed tightened formulations based on (a) Minimum number of batches (F1), (b) Minimum production requirements (F2), and (c) Combined minimum requirements (F3). The vertical axis is the fraction of instances solved in less than the time for the fastest instance multiplied by a given value on the horizontal axis. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

run is labeled as [MOD].[F#X].[OBJ].[FEAT].[PN#].[H#].[N#] and a total of 2,880 runs are included in this work.

Cumulative results

Solutions for the runs described above were obtained using CPLEX 12.5 on GAMS 24.0 on a computer with a 2.8 GHz Intel Core i7–930 processor and 8 GB of RAM, running a 64-bit Windows 7 operating system. Default CPLEX settings were used unless specified otherwise. We used a time limit of 30 min (1,800 CPU seconds).

In the interest of providing a general conclusion on the effectiveness of the proposed tightening methods, a performance assessment is made first by comparing the tightened formulations to the original formulations across all problems, models and instances. Figure 3 shows performance charts for each formulation, including the three variants and the original formulation as a reference. It is clear that the tightened formulations are much better than the original, which is only able to solve around 55% of the instances within one order of magnitude of the solution time for the fastest instance. Tightened formulation 1, which includes Eqs. 4 and 6 is able to move this percentage up to 90%, whereas formulation 2, introducing Eqs. 5 and 7 and formulation 3, including all Eqs. 4–7, are able to increase it to 95%. This suggests that including constraints based on the minimum total amount a task is required to process and the total amount of material that needs to be produced is more effective than using constraints obtained from simply bounding the number of batches for a single task and material. Additional testing (statistics not shown) showed that Eq. 5 is the most effective. In terms of variants, we notice that variant B, based on the finishing binary variable is less effective than the other two.

In order to discern the impact of the proposed tightening methods on each objective function, we separate the results based on problems. Figure 4 presents the performance results

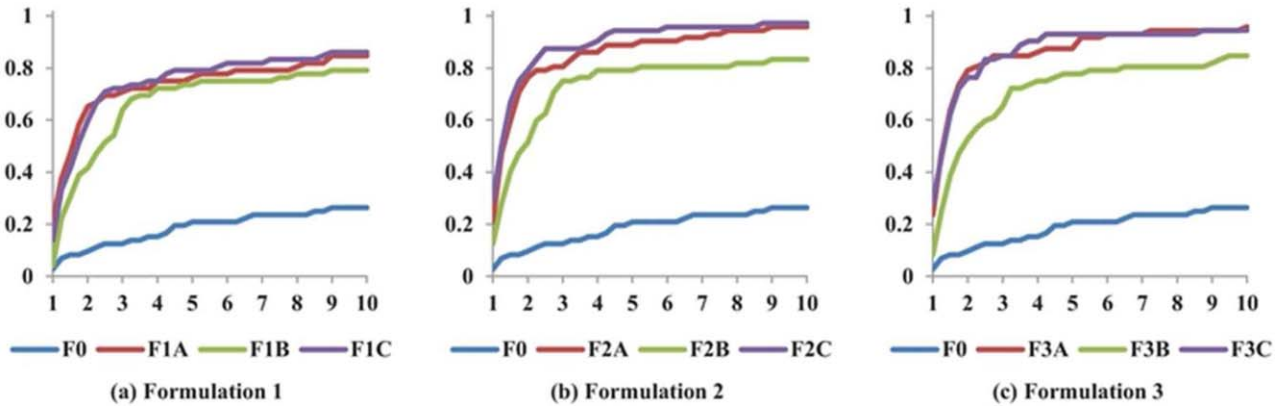

Figure 4. Performance charts for problems {CT.CPT, CT.VPT}.

Includes original formulation (F0) and proposed tightened formulations based on (a) Minimum number of batches (F1), (b) Minimum production requirements (F2), and (c) Combined minimum requirements (F3). The vertical axis is the fraction of instances solved in less than the time for the fastest instance multiplied by a given value on the horizontal axis. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

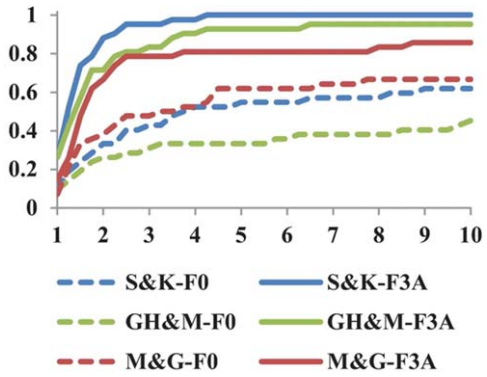

Figure 5. Performance charts for all problems with different models including formulations Fo and F3A.

The vertical axis is the fraction of instances solved in less than the time for the fastest instance multiplied by a given value on the horizontal axis. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

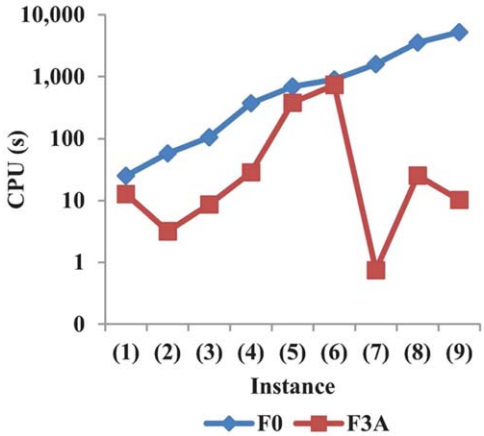

Figure 6. Computational time for nine instances run with optimal number of points.

Instances ranked in increasing order of difficulty for formulation F0. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

for cost minimization, including both features, constant and variable processing time. Tightened formulations solve an additional 70% of the instances compared to the original formulation, which only solves around 25% of the instances within one order of magnitude of the solution time for the fastest instance. The results for makespan (not shown) are completely different, showing less than 10% improvement in the number of instances solved.

Figure 4 also confirms that the formulations that include the minimum production requirements, Eqs. 5 and 7, are more effective. In addition, no significant difference between formulations 2 and 3 is evident after this comparison.

Although not the primary goal of this work, we now compare the behavior of the models under study when the tightening methods are used. Figure 5 shows the performance profiles for the three models, before and after the valid inequalities with formulation 3 and variant A are added. All models benefit from the tightening constraints, but to different extents. Model GH&M exhibits the largest improvement with an additional 50% of the instances solved to optimality within one order of magnitude of the fastest instance, followed by model S&K with 40% and M&G with less than 20%. However, model S&K is the only one that solves all the instances in less than 4 times the solution time of the fastest instance. Interestingly model M&G is the best model for the original formulation but the worst after the tightening constraints are included.

Table 2. Instances Selected for Solution Time Analysis

Number	Instance	N^*
(1)	GH&M.F3A.MS.VPT.PN2.H24	13
(2)	S&K.F3A.CT.CPT.PN1.H36	21
(3)	S&K.F3A.CT.VPT.PN1.H24	11
(4)	M&G.F3A.CT.CPT.PN4.H36	11
(5)	M&G.F3A.MS.CPT.PN4.H24	10
(6)	GH&M.F3A.MS.CPT.PN3.H24	7
(7)	M&G.F3A.CT.CPT.PN1.H24	15
(8)	GH&M.F3A.CT.VPT.PN2.H36	17
(9)	S&K.F3A.CT.VPT.PN3.H24	7

Selected instances

To further illustrate the computational advantages of the proposed methods, we present detailed results for particular instances. Following the analysis presented in the previous subsection, we use formulation 3, variant A (F3A) for all the selected instances because it led to the best average results. We chose nine instances, which cover all models, networks, and horizon values. Using the identification introduced above, Table 2 summarizes the instances studied and their optimal number of points. Complete statistics for these instances are included online in the Supporting Information. To better assess the computational enhancement, we do not use a resource limit of 1,800 CPU seconds.

Figure 6 presents the results in terms of computational time for each instance for both the original and the tightened formulation. It confirms the general behavior derived from the performance charts, in which a clear improvement was introduced by using the valid inequalities derived from the proposed algorithm. We observe that the instances in which makespan is the objective function do not improve as much as the ones in which cost minimization is the objective. On the other hand, for instances where cost is minimized we observe significant improvements ranging from one- to four-orders of magnitude. The behavior exhibited by Figure 6 is representative of the enhancements we observed in other instances.

Conclusions

In this research note, we presented tightening methods for general material-based continuous-time models for the solution of chemical production scheduling problems. In particular, we extended the ideas presented by Velez et al.[36] for discrete-time formulations to take into account the demand information that is available as an input for the solution of

the MIP model. Using the algorithm and valid inequalities discussed in their work, we were able to achieve significant improvement for three representative continuous-time models found in the literature.

Specifically, we were able to show that cost minimization consistently benefits from the introduction of the tightening methods, in contrast to makespan minimization, for which only a few instances exhibited any improvement. Moreover, we also showed that constraints based on the minimum production amounts required by a particular task are more effective than the ones based on minimum number of batches. In addition, we determined that the variants do not introduce a significant difference, although it seems to be better to use those that include the starting binary variable.

The results obtained for continuous-time models are comparable to those reported by Velez et al.[36] for discrete-time formulations. The introduction of the proposed tightening methods led to order-of-magnitude improvements in computational time; in some instances up to four orders of magnitude were achieved. In both time frameworks, we found that cost minimization, which is probably the most widely used objective in practice for short-term scheduling, benefits the most from our methodologies. An important difference, however, is that makespan minimization was found to be easier than cost minimization in discrete-time models, so the methods were most effective for the *harder* problem, whereas for continuous-time formulations cost minimization is easier than makespan minimization.

Since the calculation of the parameters included in the algorithm only requires information on network topology and customer demand and this information is available *a priori*, it is possible to extend our methodologies to address problems that include several common features in scheduling problems, such as utilities and shared resources, changeovers, shared storage, special storage policies, and noninstantaneous material transfers. Moreover, given that the formulation of the valid inequalities themselves is based on a starting binary variable which is present in every time-grid-based model, we expect our methods to work with all continuous-time models found in literature.

It is important to note that the calculation of the parameters required to formulate the tightening constraints is done in very short times (<10 s) compared to the computational times for solving the MIP itself. Thus, it is guaranteed that the running time for the algorithm is negligible in the overall solution process. In addition, the number of additional constraints introduced by the algorithm is also very small compared with the number of equations defining the model. Therefore, we expect our methods will almost always lead to net improvements in the solution process.

In summary, we showed that by using the demand propagation algorithm and defining appropriate valid inequalities, a great enhancement in the solution of chemical production scheduling problems is achieved. Our methods are applicable to all continuous-time models and can be readily used in problems with a broad range of processing characteristics and constraints.

Acknowledgments

The authors acknowledge financial support from the National Science Foundation under Grant CBET-1066206.

Notation

Indices and sets

$i \in \mathbf{I}$ = tasks
$i \in \mathbf{j}$ = units
$k \in \mathbf{K}$ = materials
$n \in \mathbf{N}$ = time points
$\mathbf{I}_k^+/\mathbf{I}_k^-$ = tasks that produce/consume material k
$\mathbf{J}_i$ = units that can perform task i
$\mathbf{K}^{MT}$ = materials that are produced by multiple tasks

Parameters

$\alpha_{ij}^F/\alpha_{ij}^V$ = fixed/variable components of the processing cost for running task i in unit j
$\beta_j^{max}/\beta_j^{min}$ = maximum/minimum batch size for unit j
γ_k = storage capacity for material k
η = horizon
κ_k = lower bound on the number of batches of tasks producing material k
λ_i = lower bound on the number of batches of task i
μ_i = lower bound on the production of task i required to meet final demand
ξ_k = demand of material k at the end of the horizon
ρ_{ik} = fraction of material k produced/consumed by task i
τ_{ij}^F/τ_{ij}^V = fixed/variable components of the processing time for task i in unit j
ω_k = lower bound on the amount of material k required to meet final demand

Binary variables

X_{ijn}/Y_{ijn} = one if task i begins/finishes in unit$j$ at time n

Continuous variables

BS_{ijn} = batch size of task i that begins in unit j at time n
S_{kn} = inventory level of material k at point time n

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. 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.
4. Shah N, Pantelides CC, Sargent RWH. A general algorithm for short-term scheduling of batch-operations.2. Computational issues. *Comput Chem Eng*. 1993;17(2):229–244.
5. Pantelides CC. Unified Frameworks for Optimal Process Planning and Scheduling. Paper presented at: 2nd Conference on Foundations of Computer Aided Process Operations; 1994, 1994; Snowmass, CO.
6. Schilling G, Pantelides CC. A simple continuous-time process scheduling formulation and a novel solution algorithm. *Comput Chem Eng*. 1996;20:S1221–S1226.
7. 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.
8. Mockus L, Reklaitis GV. Continuous time representation approach to batch and continuous process scheduling. 2. Computational issues. *Ind Eng Chem Res.* 1999;38(1): 204–210.
9. 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.
10. Giannelos NF, Georgiadis MC. A novel event-driven formulation for short-term scheduling of multipurpose continuous processes. *Ind Eng Chem Res.* 2002;41(10): 2431–2439.
11. 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.
12. 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.
13. Maravelias CT. Mixed-time representation for state-task network models. *Ind Eng Chem Res.* 2005;44(24):9129–9145.
14. 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.
15. BarbosaPovoa APFD, Pantelides CC. Design of multipurpose plants using the resource-task network unified framework. *Comput Chem Eng.* 1997;21:S703–S708.
16. 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.
17. Gimenez DM, Henning GP, Maravelias CT. A novel network-based continuous-time representation for process scheduling: Part II. General framework. *Comput Chem Eng.* 2009;33(10):1644–1660.
18. Kelly JD, Zyngier D. Multi-product inventory logistics modeling in the process industries In: Chaovalitwongse W, Furman KC, Pardalos PM, eds. Optimization and Logistics Challenges in the Enterprise. New York: Springer; 2009:61–95.
19. Sundaramoorthy A, Maravelias CT. A general framework for process scheduling. *AIChE J.* 2011;57(3):695–710.
20. Maravelias CT, Papalamprou K. Polyhedral results for discrete-time production planning MIP formulations for continuous processes. *Comput Chem Eng.* 2009;33(11): 1890–1904.
21. Maravelias CT. On the combinatorial structure of discrete-time MIP formulations for chemical production scheduling. *Comput Chem Eng.* 2012;38:204–212.
22. Bassett MH, Pekny JF, Reklaitis GV. Decomposition techniques for the solution of large-scale scheduling problems. *AIChE J.* 1996;42(12):3373–3387.
23. Papageorgiou LG, Pantelides CC. Optimal campaign planning scheduling of multipurpose batch semicontinuous plants.2. A mathematical decomposition approach. *Ind Eng Chem Res.* 1996;35(2):510–529.
24. Harjunkoski I, Grossmann IE. Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. *Comput Chem Eng.* 2002;26(11):1533–1552.
25. Wu D, Ierapetritou MG. Decomposition approaches for the efficient solution of short-term scheduling problems. *Comput Chem Eng.* 2003;27(8-9):1261–1276.
26. Maravelias CT. A decomposition framework for the scheduling of single- and multi-stage processes. *Comput Chem Eng.* 2006;30(3):407–420.
27. Castro PM, Harjunkoski I, Grossmann IE. Greedy algorithm for scheduling batch plants with sequence-dependent changeovers. *AIChE J.* 2011;57(2):373–387.
28. Ferris MC, Maravelias CT, Sundaramoorthy A. Simultaneous batching and scheduling using dynamic decomposition on a grid. *INFORMS J Comput Sum* 2009;21(3): 398–410.
29. Velez S, Maravelias CT. A branch-and-bound algorithm for the solution of chemical production scheduling MIP models using parallel computing. *Comput Chem Eng.* 2013;55(0):28–39.
30. Sahinidis NV, Grossmann IE. Reformulation of multiperiod MILP models for planning and scheduling of chemical processes. *Comput Chem Eng.* 1991;15(4):255–272.
31. 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.
32. Velez S, Maravelias CT. Reformulations and branching methods for mixed-integer programming chemical production scheduling models. *Ind Eng Chem Res.* 2013;52(10):3832–3841.
33. 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.
34. 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.
35. Velez S, Maravelias CT. Mixed-integer programming model and tightening methods for scheduling in general chemical production environments. *Ind Eng Chem Res.* 2013;52(9):3407–3423.
36. Velez S, Sundaramoorthy A, Maravelias CT. Valid inequalities based on demand propagation for chemical production scheduling MIP models. *AIChE J.* 2013; 59(3):872–887.
37. 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.
38. Papageorgiou LG, Pantelides CC. Optimal campaign planning scheduling of multipurpose batch semicontinuous plants.1. Mathematical formulation. *Ind Eng Chem Res.* 1996;35(2):488–509.

Manuscript received July 25, 2013, revision received Sept. 18, 2013,