A Continuous-Time Tightened Formulation for Single-Stage Batch Scheduling with Sequence-Dependent Changeovers

Pablo A. Marchetti and Jaime Cerdá*

INTEC (Universidad Nacional del Litoral-CONICET), Güemes 3450, 3000, Santa Fe, Argentina

This work presents a new mixed-integer linear programming (MILP) continuous-time approach for the short-term scheduling of single-stage multiproduct batch plants with parallel units and sequence-dependent changeovers. It uses a unit-specific precedence-based representation, combined with effective, nontrivial tightening constraints, to develop a very efficient problem formulation. The additional cuts account for the updated information provided by allocation and sequencing binary variables to systematically reduce the solution space of the corresponding LP at every node of the enumeration tree. In this way, close bounds for key variables like makespan, task earliness, and task starting/completion times are generated and continually improved throughout the search in order to accelerate the node pruning process. Alternative problem objectives like the minimum total earliness or the shortest makespan can be managed. To make a thorough comparison with previous continuous-time scheduling approaches, several benchmark examples have been solved. Results show that the proposed approach usually presents the best computational performance.

1. Introduction

An efficient utilization of manufacturing resources is an extremely important issue in the operation of multiproduct batch plants to cut production costs and increase annual profits and customer satisfaction. The scheduling function is the one concerned with the proper allocation of production resources to processing tasks. It aims to timely reach the specified production targets with a minimum resource capacity. Therefore, efficient tools helping schedulers to develop the best short-term schedule of multiproduct batch plants are of great interest in most manufacturing companies. Several solution methodologies have been proposed for different types of batch scheduling problems. An extensive review of the state of the art can be found in Floudas and Lin[1] and Méndez et al.[2] Overall, exact solution methods using mathematical programming models have received most of the research attention. Depending on the way the set of batches and their sizes are defined, two types of scheduling methodologies have been developed: monolithic and sequential approaches.[2] The first group tackles the full batching/scheduling problem. In addition to allocation and sequencing decisions, a monolithic approach must also choose the set of batches (size and number) that undergo each required processing task (lot-sizing problem). Because the size of any batch can change along the processing network, batch mixing/splitting operations are needed and material balances should be incorporated in the problem formulation. The STN (state-task network) representation, introduced in the seminal paper of Kondili et al.,[3] is generally used by monolithic approaches to describe the production recipes. An alternative representation is the resource-task network (RTN), where the concept of resource is unified to explicitly include equipment units (Pantelides[4]). In contrast, sequential scheduling methods are not concerned with the batching problem since they assume that the set of batches for the entire process has been previously adopted. Therefore, batch size, release times, due dates, and any other related information are problem data. By just handling linear production recipes, the treatment of mixing/splitting operations and material recycles can be avoided and, consequently, sequential models never include mass balance constraints.

Another way of classifying the scheduling methodologies is based on the time representation being applied. On one hand, a uniform time discretization approach assumes that the scheduling horizon has been divided into several time periods of equal duration, and the start/completion times of the tasks must occur at period ends. Several monolithic methodologies using the discrete time representation have been proposed to solve batch scheduling problems.[3−6] On the other hand, continuous-time representations assume that the time events can take place at any moment throughout the scheduling horizon. Timing decisions are explicitly given in terms of continuous variables denoting the exact times at which the events happen. Continuous-time representations can be grouped into three different types: global time point formulations, unit-specific time event approaches, and precedence-based models. Global time point formulations all assume a common variable time grid for all shared resources and are based on STN[7−9] or RTN process representations.[10−13] In such formulations, variable batch sizes and batch-size dependent resource requirements are allowed and tasks are required to start or end just at the time points. Since more time points are needed as the set of batches becomes larger, the number of global time points is a major multiplier of the problem size. Thus, batch scheduling problems with more than 10−12 time points usually need a considerable CPU solution time.[9]

An alternative methodology associates the starting time of tasks to unit-specific time events.[14−17] This time event representation allows the same time point to have different values for different units, thus producing a smaller model size with regard to previous approaches. A similar idea has been previously proposed by Pinto and Grossmann[18,19] by introducing the notion of time slots as predefined time intervals of unknown duration. In this case, each unit has associated a set of preordered time slots to which the batches must be allocated, and the number of slots and the slot time limits can vary with the unit. Non-common slots become a feasible alternative because the formulation was restricted to sequential processes without resources different from equipment and, consequently, no mass

* To whom correspondence should be addressed. E-mail address: jcerda@intec.unl.edu.ar.

10.1021/ie701774w CCC: $40.75
Published on Web 12/05/2008

balance equation is needed. A recent monolithic slot-based approach relying on the notion of synchronous time slots has been introduced to account for additional manufacturing resources (manpower, utilities) on a STN-based formulation.[20]

Other continuous-time formulations have been developed to handle sequential processes (linear recipes) based on the immediate or general precedence concepts. On this kind of approach, specific variables that represent the starting/completion times of a given task are defined instead of using time points at which any processing task can either begin or end. The immediate precedence representation (either unit-specific[21] or not[22,23]) uses 0−1 variables to handle the decision of performing one task immediately before another, and handles sequence-dependent changeovers in a straightforward manner. The same advantage can be found on the general precedence scheme introduced by Méndez et al.;[24] however, in this case the precedence relation has been extended to include not only the immediate predecessor but also all the tasks processed before in the same unit. In addition, assignment and sequencing decisions are decoupled and only one binary variable is required to sequence every pair of processing tasks (i, i'). This leads to an MILP mathematical formulation with a better computational performance than other continuous time approaches for batch scheduling with sequence-dependent changeovers.

In recent papers, Castro and Grossmann[25,26] introduced a multiple time grid approach. The new time-continuous representation was shown to be very competitive for tackling single-stage scheduling problems, because it uses a unit-specific time grid. A significant number of examples were solved to compare the performance of their single- and multiple-stage formulations with other existing discrete/continuous MILP and constraint programming (CP) approaches.

Overall, discrete time representations have a very tight feasible region and present a small integrality gap for problems with sequence-independent changeovers.[2,25] The required CPU time to solve them remains low as long as the model size remains manageable. However, if the scheduling horizon of the problem on hand is rather long, or the number of significant digits on the problem data is important, then a greater number of time intervals is required and the computational performance rapidly worsens. In the latter case, the problem data are usually rounded-off to prevent from largely increasing the number of time intervals. As a result, a poor problem representation may be obtained and the best schedule found is probably nonoptimal or even infeasible. Sequence-dependent changeovers can be managed by discrete formulations but at the expense of significantly increasing the model size and the required computational cost. In turn, continuous-time formulations present larger integrality gaps and more loosely feasible regions. However, some representations have shown to have a better computational performance for certain objective functions or for problems with some specific features. In monolithic continuous-time formulations with either global time points or unit-specific events, the number of points/events must be determined through a rather costly iterative process that consists of repeatedly adding another time point and solving the resulting problem formulation until the best schedule found shows no improvement. This procedure cannot guarantee the optimality of the solution found. On the other hand, the performance of precedence-based models is somewhat deteriorated by the use of big-M sequencing constraints, especially when the makespan is the objective function to be minimized. They produce an increase in the integrality gap (the difference between the optimal values of the relaxed RMIP and the original MILP problems), thus making it harder to find the optimal schedule through a branch-and-cut algorithm. Generally, issues such as low number of binary variables and lack of big-M constraints have been cited as key model features to bound the computational cost.

This work presents an enhanced mathematical formulation for the scheduling of single-stage multiproduct batch plants with sequence-dependent changeovers. It is based on the underlying idea that a suitable time representation, combined with effective, nontrivial tightening constraints ("cuts"), can lead to a MILP problem formulation with a very good computational performance. Such a combination permits to obtain better bounds from the RMIP model solved at each node of the tree and thus accelerate the node pruning process. In order to derive effective cuts, a continuous-time unit-dependent precedence-based mathematical formulation has been developed. Nontrivial cuts that take into account the information provided by allocation and sequencing variables during the search (either if they were branched or not by the solver algorithm) were also incorporated in the problem formulation. Thus, valid tight bounds for key variables such as makespan, task earliness, and task starting/completion times that continually improve throughout the search are generated. The proposed approach accounts for batch release times, unit ready times, batch due dates, and sequence-dependent setup times. Alternative problem objectives were used, such as the minimization of the overall earliness or the makespan. Several benchmark problems have been solved to make a thorough comparison with the performance of previous scheduling approaches, including the general precedence scheme of Mendez et al.[24] Future work will generalize the proposed formulation to tackle multistage batch plant scheduling problems.

This work is organized as follows. In section 2, the scheduling problem to be tackled is properly defined, and the model assumptions are presented. Section 3 introduces the new mathematical model and the special tightening constraints involving just allocation variables or a mix of allocation and sequencing variables. Special emphasis is made on setting close upper bounds for the task completion times when the overall earliness is minimized. Section 4 presents the computational results for a set of benchmark examples, and the conclusions are included in section 5.

2. Problem Statement

The problem of short-term scheduling of single-stage multiproduct batch plants with parallel production units can be stated as follows. Given:

(a) a single-stage multiproduct batch plant with multiple parallel units $j \in J$,

(b) a set of single-batch orders $i \in I$ to be completed within the scheduling horizon,

(c) the release time rt_i and the due date dd_i for each batch $i \in I$,

(d) the set of available processing units $J_i \subset J$ and the constant processing time pt_{ij} at each one for every batch i,

(e) the sequence-dependent setup times $\tau_{ii'j}$,

(f) the equipment unit ready times ru_j, and

(g) the specified time horizon H.

The problem goal is to determine a production schedule that completes all batch orders within their time limits, while satisfying assignment and sequencing constraints and optimizing a given schedule criterion, such as the makespan or the overall weighted earliness.

To derive the proposed mathematical formulation for single-stage batch scheduling, the following assumptions are made:

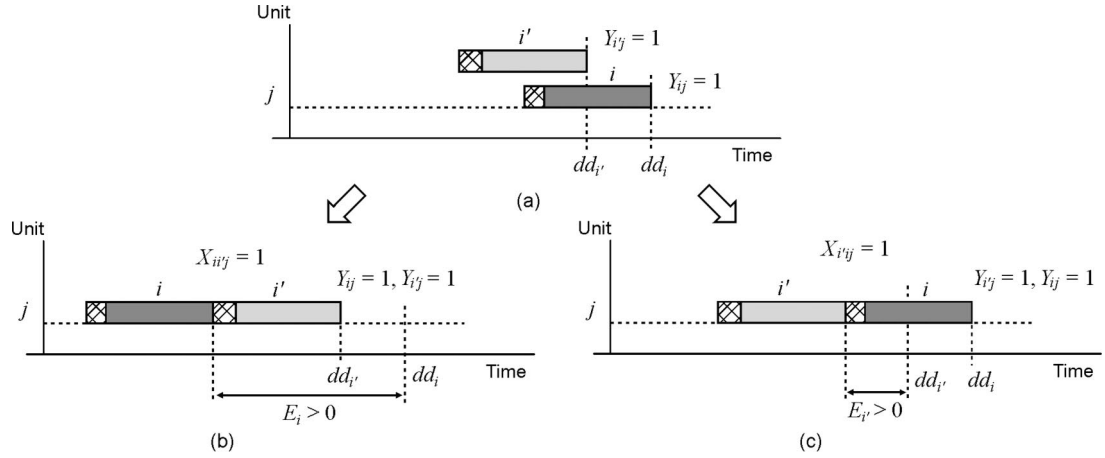

Figure 1. The necessary earliness of a task to meet other due date constraints.

1. Model parameters are all deterministic.
2. Transition times between any pair of tasks are computed by adding two components: a non-sequence-dependent (su_{ij}) and a sequence-dependent ($\tau_{ii'j}$) setup time.
3. Processing times dominate sequence-dependent changeover times (i.e., the longest changeover time never exceeds the shortest processing time).

3. Mathematical Model

3.1. Problem Constraints. 3.1.1. Assignment of Processing Units to Tasks. Let us define the binary variable Y_{ij} to represent the decision of allocating the equipment unit j to task i. Because a single unit must be assigned to every required task, then constraint (1) must be satisfied.

$$\sum_{j \in J_i} Y_{ij} = 1 \qquad \forall\, i \in I \tag{1}$$

3.1.2. Relationships among Assignment and Sequencing Decisions. To choose the task processing sequence at every equipment unit, a set of unit-specific sequencing variables $X_{ii'j}$ is defined. They are based on the general precedence notion introduced by Méndez et al.,[24] because they describe the relative ordering of any pair of tasks in the queue of any unit $j \in J_{ii'}$. In our formulation, however, a different set of sequencing variables for each equipment item rather than a single one for the entire single-stage multiproduct facilty is considered. Since the unit index j is incorporated in the domain of the variable $X_{ii'j}$, the resulting approach can be regarded as a unit-dependent general-precedence based scheduling methodology.

$$X_{ii'j} = \begin{cases} 1 & \text{if task } i \text{ is processed before task } i' \text{ in unit } j \\ 0 & \text{otherwise} \end{cases}$$
$$\forall\, i, i' \in I,\ j \in J_{ii} : (i \neq i)$$

In other words, a different binary variable $X_{ii'j}$ is to be defined not only for every possible permutation of any pair of tasks (i, i') potentially sharing an equipment unit but also for each eligible unit $j \in J_{ii'}$. By explicitly including the equipment information on the definition of $X_{ii'j}$, additional tightening constraints that produce better lower bounds on the optimal value of the objective function can be derived. In this way, the convergence rate of the MILP solution algorithm can be significantly improved with regard to the general precedence-based scheduling approach,[24] despite the larger number of unit-specific sequencing variables $X_{ii'j}$.

Since either $X_{ii'j}$ or $X_{i'ij}$ must be equal to 1 whenever the pair of tasks (i, i') has been allocated to the same unit $j \in J_{ii'}$ ($Y_{ij} + Y_{i'j} = 2$), then constraint (2), which relates assignment and sequencing variables, must be incorporated in the problem formulation.

$$Y_{ij} + Y_{i'j} - 1 \leq X_{ii'j} + X_{i'ij} \qquad \forall\, i,i' \in I, j \in J_{ii'} : (i \neq i') \tag{2}$$

3.1.3. Task Sequencing Constraints. Task-sequencing constraints (3) are needed to prevent from task overlapping when two different tasks $i,i' \in I$ are processed in the same unit $j \in J_i \cap J_{i'}$. The allocation time of unit j to any one of the tasks $i \in I_j$ also includes the sequence-dependent setup time ($\tau_{ii'j} + su_{i'j}$).

$$C_i + \sum_{j \in J_{ii'}} (\tau_{ii'j} + su_{ij}) Y_{i'j} \leq S_{i'} + H\left(1 - \sum_{j \in J_{ii'}} X_{ii'j}\right)$$
$$\forall\, i,i' \in I : (i \neq i') \text{ and } (J_{ii'} \neq \varnothing) \tag{3}$$

Equation (3) is just defined for any pair of tasks potentially sharing an equipment unit. It provides a lower bound on the value of the starting time $S_{i'}$. In contrast to the definition of the sequencing variables, the domain of eq (3) does not include the equipment subscript j. As a result, a much lower number of sequencing constraints is required with regard to the general precedence approach of Méndez et al.[24]

3.1.4. Task Starting/Completion Times. The starting time of task i (S_i) can be derived from its completion time C_i by subtracting its processing time pt_{ij} on the allotted unit j.

$$S_i = C_i - \sum_{j \in J_i} pt_{ij} Y_{ij} \qquad \forall\, i \in I \tag{4}$$

Moreover, the constraints (5) and (6) introduce additional bounds on the start and the completion time of task $i \in I$, respectively. On one hand, task i must start not before either its release time rt_i or the ready time of the allotted unit j (ru_j). The higher of both is used as a lower bound for S_i. On the other hand, task i must be completed before its promised due date.

$$S_i \geq \sum_{j \in J_i} \mathrm{Max}[rt_i, ru_j + su_{ij}] Y_{ij} \qquad \forall\, i \in I \tag{5}$$

$$C_i \leq dd_i \qquad \forall\, i \in I \tag{6}$$

3.1.5. Task Earliness. Alternatively, the due date constraint for a processing task i can be written as follows:

$$E_i \geq dd_i - C_i \qquad \forall\, i \in I \tag{7}$$

where the non-negative variable E_i stands for the earliness of task i.

3.1.6. Makespan. The schedule makespan (MK) denoting the time required to complete all the processing tasks is given by

the largest task completion time. Constraint (8) will be considered only if MK is the selected problem goal to be minimized.

$$MK \geq C_i \qquad \forall\, i \in I \tag{8}$$

3.2. Tightening Constraints Involving Assignment Variables. When the problem goal is to minimize the makespan, the values of the assignment variables partially (non-integer Y_{ij}) or completely ($Y_{ij} = 1$) allocating units to batches constitute valuable information to derive much tighter lower bounds on the optimal MK while executing the MILP solution algorithm. Because the processing time is usually much longer than the setup time for any task and the best schedule features a very low equipment idle time in real-life problems, the overall workload of an equipment unit then can be well-approximated by just considering the run times of the allocated tasks. In this way, the additional constraints (9) providing a tight lower bound of the makespan for single-stage batch scheduling problems, can be derived in a straightforward manner.

$$\mathrm{ru}_j^* + \sum_{i \in I_j} (\mathrm{su}_{ij} + \mathrm{pt}_{ij}) Y_{ij} \leq MK \qquad \forall\, j \in J \tag{9}$$

where $\mathrm{ru}_j^* = \mathrm{Max}\,[\mathrm{ru}_j, \mathrm{Min}_{\,i \in I_j}[\mathrm{rt}_i - \mathrm{su}_{ij}]]$ is a better estimation of the jth-unit ready time, because it also considers the release times of the candidate tasks for unit j. If changeover times are not sequence-dependent and the minimum makespan is the problem goal, the highest LHS of constraints (9) will provide a close estimation of the optimal MK value. The effectiveness of the valid cuts (9) can be measured through the increase of the lower bound provided by the corresponding LP at every node of the implicit enumeration tree. They are said to be valid cuts because they never exclude integer solutions from the problem feasible space.

If setup times are sequence-dependent and not negligible, they can be easily taken into account by replacing eq (9) by the set of constraints given in eq (10). The summation in eq (10) now includes the lowest possible sequence-dependent setup time $\sigma_{ij}^{\mathrm{Min}}$ for every task $i \in I_j$ in unit j, as defined by eq (11). Since the first task in the queue of unit j just features a setup time su_{ij}, then the value of the LHS of eq (10) should be reduced by the highest possible $\sigma_{ij}^{\mathrm{Min}}$, in order to develop a valid cut that ensures the optimality of the solution.

$$\mathrm{ru}_j^* - \underset{i \in I_j}{\mathrm{Max}}[\sigma_{ij}^{\mathrm{Min}}] + \sum_{i \in I_j} (\sigma_{ij}^{\mathrm{Min}} + \mathrm{su}_{ij} + \mathrm{pt}_{ij}) Y_{ij} \leq MK \qquad \forall\, j \in J \tag{10}$$

where

$$\sigma_{ij}^{\mathrm{Min}} = \underset{i' \in I_j\,:\, i' \neq i}{\mathrm{Min}} [\tau_{i'ij}] \qquad \forall\, i \in I, j \in J_i \tag{11}$$

Cuts (9)−(10) just involve assignment variables Y_{ij}. When the production schedule featuring the minimum makespan is sought, then the mathematical model to be solved will include the set of equations {(1)−(6), (8), and (9) or (10)}. In the next section, additional cuts based on information provided by assignment and sequencing variables are presented.

3.3. Valid Cuts Involving Sequencing Variables. New linear tightening constraints, in terms of the model variables $X_{ii'j}$, providing good estimates for the earliest start time of task i (EST_i), the latest completion time (LCT_i), and the makespan (MK), are going to be developed.

3.3.1. Lower Bound for the Task Starting Time S_i. Let us define the variable EST_i as the earliest starting time of task i, i.e., a lower bound on the value of S_i. A tight estimation of EST_i requires knowledge of which tasks precede task i in the waiting queue of the alloted unit, i.e., the information provided by the sequencing variables $X_{i'ij}$. Equation (12) provides the EST_i value, in terms of the unit-specific sequencing variables.

$$EST_i = \sum_{i' \in I\,:\, i' \neq i} \sum_{j \in J_{ii'}} (\sigma_{i'j}^{\mathrm{Min}} + \mathrm{su}_{i'j} + \mathrm{pt}_{i'j}) X_{i'ij} - \underset{i' \in I_j}{\mathrm{Max}}[\sigma_{i'j}^{\mathrm{Min}}] \leq S_i \qquad \forall\, i \in I \tag{12}$$

Since the first task processed in unit j features only a setup time $\mathrm{su}_{i'j}$, the RHS of eq (12) should be reduced by the highest possible $\sigma_{i'j}^{\mathrm{Min}}$. As the solution algorithm progresses and processing tasks become gradually allocated to units by setting the related assignment variables to one, the sequencing variables $X_{i'ij}$, either branched or not, will also increase to 1 by the effect of the constraints (2), thus generating better lower bounds on the starting time S_i. The constraint (12) on the value of S_i can be used when either the makespan or the overall earliness is minimized.

3.3.2. Lower Bound for the Makespan. A tightening lower bound for MK can be generated by adding to the value of C_i the total setup and processing times of the tasks executed after task i in the same unit. This improvement on the lower bound of MK can be achieved using the information provided by the sequencing variables $X_{ii'j}$.

$$C_i + \sum_{i' \in I\,:\, i' \neq i} \sum_{j \in J_{ii'}} (\sigma_{i'j}^{\mathrm{Min}} + \mathrm{su}_{i'j} + \mathrm{pt}_{i'j}) X_{ii'j} \leq MK \qquad \forall\, i \in I \tag{13}$$

Cut (13) can replace constraint (8) in the mathematical formulation when the makespan is the problem objective to be minimized. Compared to cut (13), the impact of eq (9) or eq (10) on the computational cost is much stronger.

3.3.3. Close Upper Bound for the Task Completion Time C_i. From constraint (6), the due date of task i (dd_i) can be regarded as an upper bound of the completion time C_i, i.e., the latest completion time of task i (LCT_i). Sometimes, however, the completion of task i must occur before dd_i, so that other tasks $i' \neq i$ following task i on the same processing queue can be finished on time ($C_{i'} \leq \mathrm{dd}_{i'}$). In such cases, task i will feature a necessary earliness ($E_i > 0$) and a tighter estimation of LCT_i, lower than dd_i, can be established through the information provided by the sequencing variables related to task i.

3.3.3.1. Illustrating the Necessary Earliness of a Task. Figure 1a shows a simple example where a pair of tasks (i, i') has been allocated to the same unit j and $\mathrm{dd}_i > \mathrm{dd}_{i'}$. In addition, the allocation time of unit j to task i: $\mathrm{PT}_{ij} = (\tau_{i'ij} + \mathrm{su}_{ij} + \mathrm{pt}_{ij})$ is larger than the difference ($\mathrm{dd}_i - \mathrm{dd}_{i'}$). In Figure 1b, task i is processed before task i' in unit $j \in J_{ii'}$ and, therefore, $X_{ii'j} = 1$. To timely complete task i', the latest completion time of task i (LCT_i) must be earlier than dd_i. Then, $LCT_i \leq \mathrm{dd}_i - E_i$ and a first estimation of the threshold value of E_i is given by $\underline{E}_i = \mathrm{PT}_{i'j} + (\mathrm{dd}_i - \mathrm{dd}_{i'})$. A larger value of E_i would result if task i' does not directly succeed task i. The opposite case is considered in Figure 1c, where task i' is processed before task i and $X_{i'ij} = 1$. Because $\mathrm{PT}_{ij} > (\mathrm{dd}_i - \mathrm{dd}_{i'})$, then $LCT_{i'}$ will be less than $\mathrm{dd}_{i'}$: $LCT_{i'} \leq \mathrm{dd}_{i'} - E_{i'}$, with $\underline{E}_{i'} = \mathrm{PT}_{ij} - (\mathrm{dd}_i - \mathrm{dd}_{i'})$. If task i is shorter and $\mathrm{PT}_{ij} \leq (\mathrm{dd}_i - \mathrm{dd}_{i'})$, then $LCT_{i'} = \mathrm{dd}_{i'}$ and the value of $E_{i'}$ will be driven to zero.

3.3.3.2. Multiple Contributions to the Necessary Earliness of a Task. Let us now assume that task i preceeds either task i' (case a) or task i'' (case b) in the waiting queue of unit $j \in J_{ii'i''}$. In addition, $\mathrm{dd}_{i'} < \mathrm{dd}_i < \mathrm{dd}_{i''}$ and $\mathrm{PT}_{i''j} > (\mathrm{dd}_{i''} - \mathrm{dd}_i)$. From Figure 2a, it follows that a first estimate of the threshold earliness of task i to meet the due date $\mathrm{dd}_{i'}$ is $\underline{E}_i =$

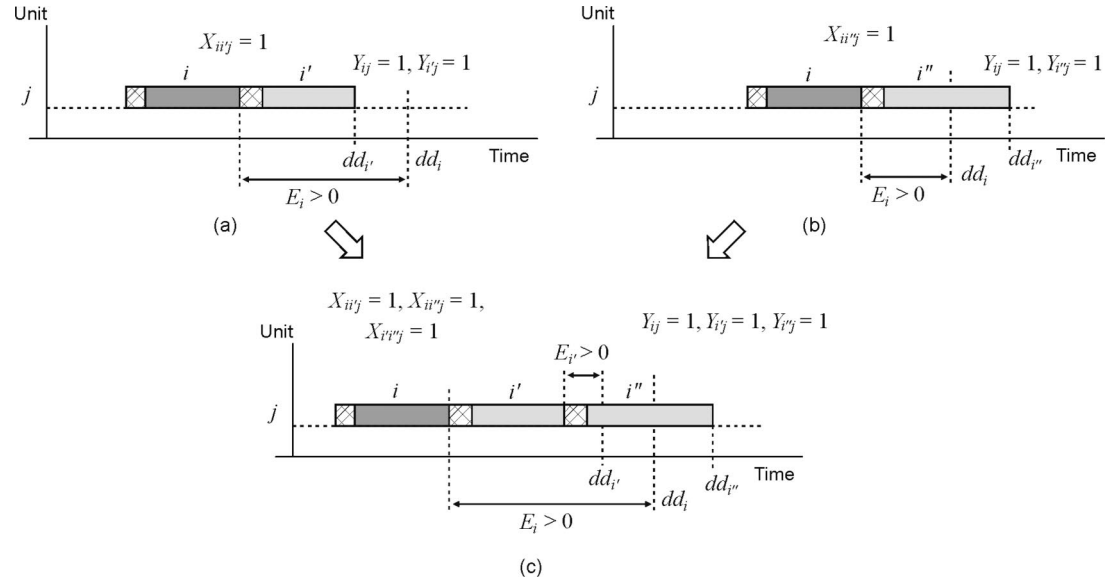

Figure 2. The necessary earliness of a task to meet due dates of two succeeding operations.

$PT_{i'j} + (dd_i - dd_{i'})$ whenever $X_{ii'j} = 1$. In turn, Figure 2b indicates that the threshold value of E_i to meet the due date $dd_{i''}$ whenever $X_{ii''j} = 1$ is $\underline{E}_i = PT_{i''j} - (dd_{i''} - dd_i)$.

In Figure 2c, task i precedes both tasks i' and i'' in the processing queue of unit $j \in J_{ii'i''}$ ($X_{ii'j} = X_{ii''j} = 1$). Therefore, such tasks (i' and i'') will require task i in unit j to be completed earlier than dd_i to meet their due date constraints. The latest completion time of task i then will be given by $LCT_i \le dd_i - \underline{E}_i$, with $\underline{E}_i = PT_{i'j} + [PT_{i''j} - (dd_{i''} - dd_i)]$. If the controlling task is task i', it is shown that E_i is still greater than $\underline{E}_i$ (see the Appendix). Such a higher threshold value of E_i, which is caused by a pair of succeeding operations instead of a single one, can be regarded as the result of adding properly defined contributions of both tasks i' and i'' to the earliness of task i.

3.3.3.3. The Earliness Contribution Adding Rule. To derive a general expression for the earliness contribution adding rule, let us introduce the parameter $\beta_{ii'j}$ as the threshold contribution of a succeeding task i' to the earliness of task i, i.e., a lower bound on the contribution of task i' to the value of E_i. Based on the due dates (dd_i, $dd_{i'}$) and a conservative estimation for the allocation time of unit j to task i' ($\underline{PT}_{i'j}$), three different types of tasks $i' \neq i$ have been considered to define the value of the earliness contribution $\beta_{ii'j}$:

Type a: $IA_i = \{\forall i' \in I (i' \neq i) \mid dd_{i'} \le dd_i \text{ and } J_{ii'} \neq \varnothing\}$

Type b: $IB_i = \{\forall i' \in I (i' \neq i) \mid dd_{i'} > dd_i \text{ and } \underline{PT}_{i'j} > (dd_{i'} - dd_i)$ for some $j \in J_{ii'}\}$

Type c: $IC_i = \{\forall i' \in I (i' \neq i) \mid i' \notin (IA_i \cup IB_i)\}$

where $\underline{PT}_{i'j} = \sigma_{i'j}^{Min} + su_{i'j} + pt_{i'j}$. The set $IA_i \subseteq I$ represents all the tasks $i' \neq i$ with $dd_{i'} \le dd_i$ that may share some unit $j \in J_{ii'}$ with task i. In turn, the set IB_i comprises every task $i' \neq i$ with $dd_{i'} > dd_i$ and $\underline{PT}_{i'j} > (dd_{i'} - dd_i)$ for some unit $j \in J_{ii'}$. Therefore, only the equipment units $j \in JB_{ii'} = \{j \in J_{ii'} \mid \underline{PT}_{i'j} > (dd_{i'} - dd_i)\}$ will be considered to calculate the contribution of task $i' \in IB_i$ to the necessary earliness of task i (E_i). The remaining tasks are contained in IC_i. The threshold contribution $\beta_{ii'j}$ of a succeeding task i' of any type to the value of E_i, whenever $X_{ii'j} = 1$, is given by eq (14):

$$\beta_{ii'j} = \begin{cases} \underline{PT}_{i'j} & \text{if } i' \in IA_i, j \in J_{ii'} \quad \text{(Type a)} \\ \underline{PT}_{i'j} - (dd_{i'} - dd_i) & \text{if } i' \in IB_i, j \in JB_{ii'} \quad \text{(Type b)} \\ 0 & \text{if } i' \in IC_i, j \in J_{ii'} \quad \text{(Type c)} \end{cases} \tag{14}$$

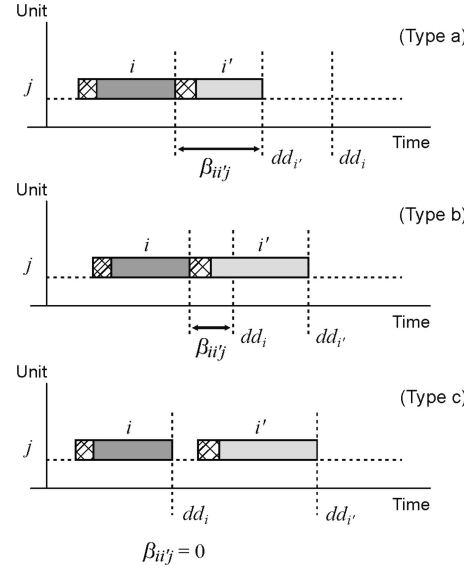

Figure 3. Threshold contribution of task i' to the necessary earliness of task i.

Moreover, the three types of tasks with the corresponding earliness contribution $\beta_{ii'j}$ are illustrated in Figure 3. To develop the expression of $\beta_{ii'j}$, it has been assumed that all due date constraints must be satisfied ($C_{i'} \le dd_{i'}$, $i' \in I$). Notice that the value of $\beta_{ii'j}$ can be computed before the problem formulation is solved.

The parameter $\beta_{ii'j}$ represents the threshold contribution of task i' to the earliness of task i, regardless of the number of tasks succeeding task i in the processing queue of unit j. Therefore, the earliness contribution adding rule providing a conservative estimation of E_i will be given by:

$$\underline{E}_i = \sum_{k \in IA_i} \sum_{j \in J_{ik}} \beta_{ikj} X_{ikj} + \sum_{/ \in IB_i} \sum_{j \in JB_{i/}} \beta_{i/j} X_{i/j} \qquad \forall i \in I \tag{15}$$

In Figure 2c, task i' belongs to the set IA_i, while $i'' \in IB_i$, since $j \in JB_{ii''}$. Moreover, the due date constraint for task i'' sets the value of E_i. From eq (15), a good, conservative estimate of the necessary earliness of task i ($\underline{E}_i$) to meet the due dates of the succeeding tasks is given by:

$$\underline{E}_i = \beta_{ii'j} X_{ii'j} + \beta_{ii''j} X_{i''j} = \underline{PT}_{i'j} + \underline{PT}_{i''j} - (dd_{i''} - dd_i) \tag{16}$$

Because $E_i = \underline{PT}_{i'j} + [\underline{PT}_{i''j} - (dd_{i''} - dd_i)]$ in Figure 2c, then $E_i = \underline{E}_i$. Equation (15) still provides a valid lower bound of E_i if the controlling task is i'. In such a case, it can be found that $E_i \geq \underline{PT}_{i''j} + \underline{PT}_{i'j} + (dd_i - dd_{i'})$ and $\underline{E}_i$ is still a valid bound (see Appendix). From the threshold earliness of task i ($\underline{E}_i$), a better estimation of its latest completion time (LCT_i) can be derived:

$$C_i \leq LCT_i = dd_i - \underline{E}_i = dd_i - \sum_{k \in IA_i} \sum_{j \in J_{ik}} \beta_{ikj} X_{ikj} + \sum_{\ell \in IB_i} \sum_{j \in JB_{il}} \beta_{i\ell j} X_{i\ell j} \quad \forall i \in I \quad (17)$$

Equation (17) will substitute eq (6) to get a tighter problem formulation. In the Appendix, it has also been proven that the earliness of task i always satisfies the following condition:

$$E_i \geq \sum_{k \in IA_i} \sum_{j \in J_{ik}} \underline{PT}_{kj} X_{ikj} + \sum_{\ell \in IB_i} \sum_{j \in JB_{i\ell}} \underline{PT}_{\ell j} X_{i\ell j} - \max_{\ell \in IB_i} [(dd_\ell - dd_i) \sum_{j \in JB_{i\ell}} X_{i\ell j}] \quad \forall i \in I \quad (18)$$

so as to meet the due dates of all succeeding tasks, whatever the type and number of succeeding operations. From eqs (14) and (15), a general expression for the threshold earliness of task i ($\underline{E}_i$) can be written as follows:

$$\underline{E}_i = \sum_{k \in IA_i} \sum_{j \in J_{ik}} \underline{PT}_{kj} X_{ikj} + \sum_{\ell \in IB_i} \sum_{j \in JB_{i\ell}} \underline{PT}_{\ell j} X_{i\ell j} - \sum_{\ell \in IB_i} (dd_\ell - dd_i)(\sum_{j \in JB_{i\ell}} X_{i\ell j}) \quad (19)$$

However,

$$\sum_{\ell \in IB_i} (dd_\ell - dd_i)(\sum_{j \in JB_{i\ell}} X_{i\ell j}) \geq \max_{\ell \in IB_i} [(dd_\ell - dd_i)(\sum_{j \in JB_{i\ell}} X_{i\ell j})] \quad (20)$$

Therefore, $E_i \geq \underline{E}_i$ and consequently $\underline{E}_i$ as given by eqs (14) and (15) always represents a threshold value for the earliness of task i. When the overall earliness is to be minimized, the mathematical model will at least include the set of equations {(1)−(7), (14)−(15), and (17)}. To reduce the earliness of task i and, consequently, the value of $\underline{E}_i$, it is expected that those binary variables $X_{ii'j}$ related to tasks i' with low $\beta_{ii'j}$ would be favored to be 1 at the optimal schedule. During the execution of the branch-and-cut search, eqs (15) and (17) will generate an increasing lower bound for E_i as the solution algorithm progresses and the nonzero sequencing variables $X_{ii'j}$ gradually rise to 1.

3.3.4. Further Tight Estimations for the Latest Completion Time of Task *i*. Let us assume that tasks $i', i'' \in IB_i$ are performed in unit $j \in J_{ii'i''}$ also assigned to task i. Figure 4a shows the threshold contributions of tasks (i', i'') to the earliness of task i. As shown by eqs (14) and (15), both contributions $\beta_{ii'j} = \underline{PT}_{i'j} - (dd_{i'} - dd_i)$ and $\beta_{ii''j} = \underline{PT}_{i''j} - (dd_{i''} - dd_i)$ result from adopting the due date dd_i as the reference time. Therefore, the completion time of task i (C_i) should never exceed the upper bound: $C_i \leq dd_i - (\beta_{ii'j} + \beta_{ii''j})$, as depicted in Figure 4b. If task i' would belong to IA_i, just the value of $\beta_{ii'j}$ will change to $\beta_{ii'j} = \underline{PT}_{i'j}$, but the same expression can be applied to estimate the LCT_i value.

However, tighter estimations of LCT_i can be obtained by adopting a reference time t different from dd_i. In Figure 5, the due date $dd_{i'}$ has been chosen as the new reference time ($t = dd_{i'} > dd_i$). Obviously, new threshold contributions to E_i, which are now referred to time t and denoted by $\beta_{ti'j}$ and $\beta_{ti''j}$, are to be defined. As shown in Figure 5a, their values will be given by $\beta_{ti'j} = \underline{PT}_{i'j} - (dd_{i'} - t)$ and $\beta_{ti''j} = \underline{PT}_{i''j} - (dd_{i''} - t)$. Since $t = dd_{i'}$, then $\beta_{ti'j} = \underline{PT}_{i'j}$. By choosing a reference time greater than dd_i, some additional tasks such as $i^{\#} \in IB_{i'}$ may have to start before $t = dd_{i'}$ to be completed on time; i.e., at time $dd_{i^{\#}}$ (see Figure 5a). Even a task $i^{\#}$ belonging to the subset IC_i may contribute to the earliness of task i by anticipating the starting time of some other tasks ($i', i'' \in IB_i$ (see Figure 5b). By choosing $t = dd_{i'}$, the earliness contribution from task $i^{\#}$, which is now given by $\beta_{ti^{\#}j} = \underline{PT}_{i^{\#}j} - (dd_{i^{\#}} - t)$, should also be considered. Because $C_i \leq dd_i < t$, it follows from Figure 5b that:

$$C_i + (\beta_{ti'j} + \beta_{ti''j} + \beta_{ti^{\#}j}) \leq t$$

For a reference time $t \neq dd_i$, a general expression for the threshold contribution of task $k \neq i$ to the value of E_i is given by

$$\beta_{tkj} = \begin{cases} \underline{PT}_{kj} & (\text{if } dd_k \leq t) \\ \underline{PT}_{kj} - (dd_k - t) & (\text{if } dd_k > t \text{ and } \underline{PT}_{kj} > dd_k - t) \\ 0 & (\text{otherwise}) \end{cases} \quad (21)$$

Equation (21) can be derived from eq (14) by changing the due date dd_i by a reference time $t \neq dd_i$. Let us define the time sets DD_i, DD_i^+, and DD_i^-:

$$DD_i = \{dd_k \mid k \in I, dd_k \neq dd_i, J_{ik} \neq \varnothing\}$$
$$DD_i^+ = \{t \in DD_i \mid t > dd_i\}$$
$$DD_i^- = \{t \in DD_i \mid t < dd_i\} \qquad i \in I$$

The threshold estimation of LCT_i with regard to time $t > dd_i$ is given by the RHS of eq (22):

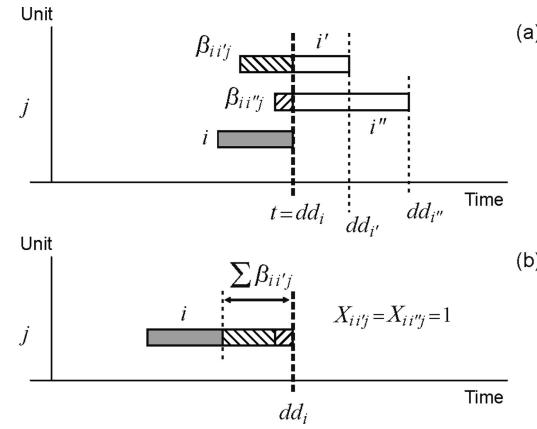

Figure 4. Threshold contributions to the value of E_i, relative to the reference time $t = dd_i$.

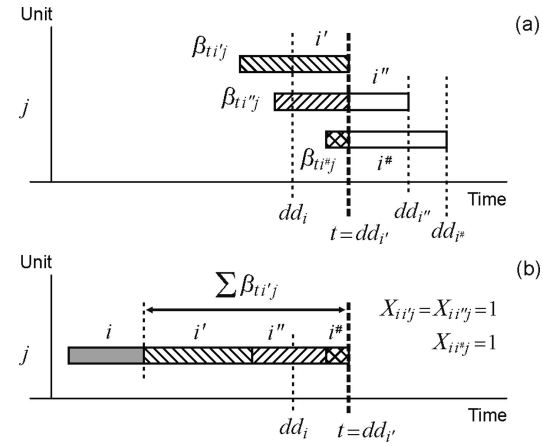

Figure 5. Threshold contributions to the value of E_i with regard to time $t > dd_i$.

$$C_i \le t - \sum_{k\in I\,:\,k\neq i}\sum_{j\in J_{ik}} \beta_{tkj} X_{ikj} \qquad \forall\, i\in I,\ t\in \mathrm{DD}_i^{+} \qquad (22)$$

where the summation is extended to any task k different from task i (see Figure 5b). By replacing t by dd_i, eq (22) reduces to eq (17). From Figures 4a and 5a, it follows that the threshold contribution of every task $\ell \in \mathrm{IB}_i$ becomes higher when a reference time t later than dd_i is selected. Often, a tighter estimation of LCT_i can be obtained by adopting $t > \mathrm{dd}_i$, as shown in Figures 4b and 5b. This is so because the indirect contribution of tasks $k \in \mathrm{IC}_i$ to the earliness of task i can also be considered. Indeed, any task i' queued after task i in the same unit can be the controlling task of i and, consequently, any due date $\mathrm{dd}_{i'} \in \mathrm{DD}_i^{+}$ can be chosen as the new reference time t. From the tasks featuring $\beta_{tkj} > 0$, only those performed after task i in the same unit will really contribute to the value of E_i. A valid upper bound of C_i is still obtained by adopting $t = \mathrm{dd}_{i'}$, even if $X_{ii'j} = 0$.

So far, we show the advantage of using $t > \mathrm{dd}_i$. However, the selection of a reference time $t < \mathrm{dd}_i$ may also provide a tighter estimation of LCT_i (see Figure 6a). In Figure 6b, task i' featuring $\mathrm{dd}_{i'} < \mathrm{dd}_i$ is performed after task i in the same unit j, and the due date $\mathrm{dd}_{i'}$ has been adopted as the new reference time instead of dd_i. According to eq (14), the threshold contribution of task i' to the value of E_i is $\beta_{ii'j} = \underline{\mathrm{PT}}_{i'j}$, although the earliness of task i is really equal to $\underline{\mathrm{PT}}_{i'j} + (\mathrm{dd}_i - \mathrm{dd}_{i'})$. Interestingly, the threshold contribution of task i' to E_i for this reference time $t = \mathrm{dd}_{i'}$ is still $\beta_{ti'j} = \underline{\mathrm{PT}}_{i'j}$ by eq (21). When a reference time $t < \mathrm{dd}_i$ is used, Figure 6b shows that the following condition holds:

$$C_i + \beta_{ti'j} \le t$$

which can be rewritten as follows:

$$C_i + \beta_{ti'j} \le \mathrm{dd}_i - (\mathrm{dd}_i - t)$$

Because $t < \mathrm{dd}_i$, the term $(\mathrm{dd}_i - t)$ is positive and a tighter upper bound of C_i can be achieved. However, the inequality $C_i + \beta_{ti'j} \le \mathrm{dd}_i - (\mathrm{dd}_i - t)$ holds only if some task i' with $\mathrm{dd}_{i'} = t \in \mathrm{DD}_i^{-}$ really succeeds task i in the queue of unit $j \in J_{ii'}$ (see Figure 6b). To guarantee this condition, a non-negative continuous variable w_{it} is defined as follows:

$$\sum_{j\in J_{ii'}} X_{ii'j} \le w_{it} \qquad \forall\, i\in I,\ t\in \mathrm{DD}_i^{-},\ i'\in I(t): J_{ii'} \neq \varnothing \quad (23)$$

where

$$I(t) = \{k \in I \mid \mathrm{dd}_k = t\}$$

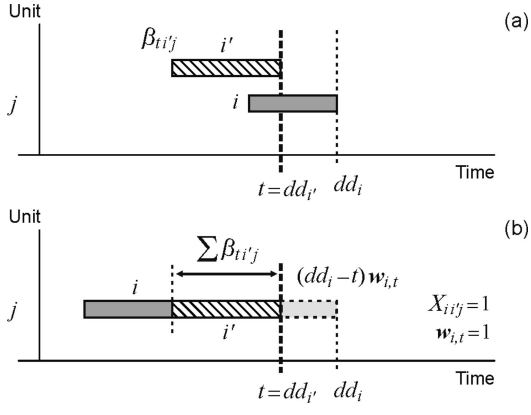

Figure 6. Threshold contributions to the value of E_i with regard to time $t < \mathrm{dd}_i$.

According to eq (23), w_{it} will be equal to 1 only if some task i' featuring a due date $\mathrm{dd}_{i'} = t \in \mathrm{DD}_i^{-}$ is processed after task i in unit $j \in J_{ii'}$. Therefore, a valid upper bound of C_i will be:

$$C_i \le \mathrm{dd}_i - (\mathrm{dd}_i - t)w_{it} - \sum_{k\in I\,:\,k\neq i}\sum_{j\in J_{ik}} \beta_{tkj} X_{ikj} \qquad \forall\, i\in I,\ t\in \mathrm{DD}_i^{-} \qquad (24)$$

If there are several tasks $i' \in I(t)$ with a common due date $t \in \mathrm{DD}_i^{-}$, any of them with $X_{ii'j} = 1$ can make the LHS summation of eq (23) equal to 1. If so, the second term on the RHS of eq (24) becomes nonzero and a tighter estimation of LCT_i can be generated. The better estimation of LCT_i provided by the set of equations (21)−(24) permits one to greatly accelerate the node pruning process. When the overall earliness is to be minimized, the mathematical model will include the set of equations {(1)−(7), (17), and (22)−(24). Equations (14)−(15) and (17) represent a rather small, very effective subset of tightening constraints. In some cases, it may be more convenient to only include such valid cuts in the problem formulation.

3.4. Objective Functions. The improved model with tightening constraints can be efficiently used to undertake problems with the following alternative problem goals.

3.4.1. Overall Weighted Earliness. When the overall earliness is to be minimized, eq (25) must be used. In this case, enough production capacity is assumed to be available to complete all batches before their due dates, i.e., constraint (7) is satisfied. The positive weight coefficient α_i represents the unit inventory cost for batch i.

$$\text{Minimize} \sum_{i\in I} \alpha_i E_i \qquad (25)$$

However, the minimum overall earliness is equivalent to maximizing the summation of batch completion times. In this case, the problem goal is given by

$$\text{Minimize} \sum_{i\in I} \alpha_i(\mathrm{dd}_i - C_i) \equiv \text{Maximize} \sum_{i\in I} \alpha_i C_i \qquad (26)$$

3.4.2. Makespan. If the minimization of the makespan is pursued, the mathematical model should include constraint (8) and the problem goal will be given by eq (27).

$$\text{Minimize } MK \qquad (27)$$

4. Computational Results

In this section, several examples are presented to illustrate the computational advantage of the proposed formulation with regard to other batch scheduling methodologies. Different sets of tightening constraints were included, depending on the selected problem goal: minimum makespan or least total earliness. By doing that, the proposed approach clearly outperforms other continuous-time MILP methodologies when they are applied to batch scheduling problems with sequence-dependent changeovers. In particular, a comparison with the sequencing model of Méndez et al.[24] shows that our unit-dependent general precedence framework requires a much lower CPU time as the problem size increases, although a larger number of binary variables is needed. Such a higher efficiency of the MILP solution algorithm comes from the better threshold value of the objective function and the tighter bounds on key variables provided by the valid cuts {(9) or (10), (17), (22)−(24)} added to the problem formulation.

The computational results reported in this section have been obtained on a Pentium IV 2.8 GHz machine, with ILOG OPL

Table 1. Data for Example 1

order	due date (days)	Processing Time (days) U_1	U_2	U_3	U_4	order	due date (days)	Processing Time (days) U_1	U_2	U_3	U_4
1	15	1.538			1.194	21	30	7.317		3.614	
2	30	1.500			0.789	22	20			0.864	
3	22	1.607			0.818	23	12			3.624	
4	25			1.564	2.143	24	30			2.667	4.000
5	20			0.736	1.017	25	17	5.952		3.448	4.902
6	30	5.263			3.200	26	20	3.824			1.757
7	21	4.865		3.025	3.214	27	11	6.410			3.937
8	26			1.500	1.440	28	30	5.500			3.235
9	30			1.869	2.459	29	25				4.286
10	29		1.282			30	26				2.154
11	30		3.750		3.000	31	22		1.569		1.363
12	21		6.796	7.000	5.600	32	18			2.698	3.654
13	30	11.250			6.716	33	15		2.147		
14	25	2.632			1.527	34	10	3.265		2.658	
15	24	5.000			2.985	35	10	3.480			2.550
16	30	1.250			0.783	36	14		2.258		
17	30	4.474			3.036	37	24		2.145	2.194	
18	30		1.429			38	16	2.365			1.850
19	13		3.130		2.687	39	22	2.030			
20	19	2.424		1.074	1.600	40	23		1.890		
	unit setup time	0.180	0.175		0.237		unit setup time	0.180	0.175		0.237

Table 2. Product Families for Example 1B

family	batches
F_1	O_1, O_2, O_3, O_5, O_{10}, O_{16}, O_{20}, O_{22}
F_2	O_4, O_8, O_9, O_{14}, O_{18}, O_{26}, O_{31}
F_3	O_7, O_{23}, O_{24}, O_{30}, O_{33}, O_{34}, O_{36}, O_{37}, O_{38}, O_{40}
F_4	O_6, O_{11}, O_{15}, O_{17}, O_{19}, O_{32}, O_{35}
F_5	O_{12}, O_{13}, O_{21}, O_{25}, O_{27}, O_{28}, O_{29}, O_{39}

Table 3. Sequence-Dependent Setup Times for Example 1B

	$\tau_{f,f'}$ F_1	F_2	F_3	F_4	F_5
F_1	0.104	0.127	0.178	0.192	0.217
F_2	0.122	0.115	0.266	0.229	0.291
F_3	0.191	0.214	0.175	0.304	0.424
F_4	0.350	0.205	0.328	0.184	0.400
F_5	0.357	0.423	0.348	0.284	0.205

Studio 3.7 software, using the embedded CPLEX v. 9.0 mixed-integer optimizer. A CPU time limit of 1 h was defined. Unless otherwise indicated, a relative gap tolerance of 1.0×10^{-6} was adopted, and the big-M parameter was made equal to the horizon length: $H = \mathrm{Max}_{i \in I}\{dd_i\}$.

4.1. Example 1. Example 1 involves a single-stage plastic compounding plant with four extruders running in parallel. This industrial problem, introduced by Pinto and Grossmann[18] and later studied by Ierapetritou, Hené, and Floudas,[27] aims to optimally schedule up to 29 batch orders. Méndez and Cerdá[28] tackled an expanded instance of Example 1 with 40 batches but assuming a dynamic scheduling scenario, with a general precedence-based MILP rescheduling approach. Recently, Castro and Grossmann[25] efficiently solved some of the larger instances of Example 1.

Order due dates and unit-dependent processing and setup times for the 40-batch instance of Example 1 can be found in Table 1. Two cases, called Example 1A and Example 1B, were studied. In Example 1A, the setup time of any task i is not dependent on the task i' previously processed. In contrast, sequence-dependent changeover times are considered in Example 1B. Let us assume that five families of products are manufactured in the plant and there is a different changeover time for each ordered pair of families. Then, the set of batches for Example 1B can be gathered into five groups F_1−F_5 (see Table 2), with a sequence-dependent setup time $\tau_{f,f'}$ between a generic family f and its succeeding family f', given in Table 3. Moreover, changeover times between consecutive families are not dependent on the assigned equipment item.

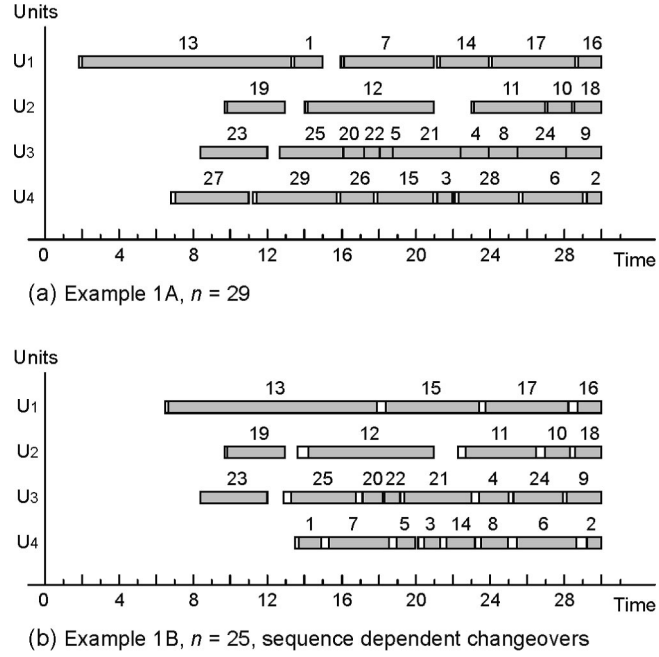

Figure 7. Optimal schedules for two instances of Example 1 found with the proposed model.

Several instances of Examples 1A and 1B involving an increasing number of batches, ranging from 12 to 40, have been solved in order to show the computational limit of the proposed methodology. Results obtained by applying both the approach of Méndez et al.[24] and the proposed formulation with valid cuts are compared in Tables 4−8. In both cases, the same computing platform and the same ILOG-software and CPLEX-solver versions were used. The makespan and the overall earliness were alternatively selected as the problem objectives.

4.1.1. Makespan. Examples 1A and 1B were solved by first assuming that the makespan is the problem goal to be minimized. To illustrate the effect of the tightening constraints involving assignment variables, only constraints (9) or (10) were

incorporated in the general precedence-based sequencing model of Méndez et al.[24] Such constraints can be used for either non-sequence-dependent or sequence-dependent changeover times. Moreover, due dates have been ignored to evaluate the effectiveness of the proposed constraints to find the short-term schedule with the shortest makespan.

For the sake of comparison, the general precedence models without (i.e., Méndez et al.[24]) and with the valid cuts (9)–(10) have been alternatively applied and the results are shown in Tables 4 and 5, respectively. It can be observed that the proposed constraints make the computational performance of the approach always much faster. For Example 1A, instances with up to 40 batches are scheduled in less than a second. For sequence-dependent changeover problems with up to 29 batches, optimal or near-optimal solutions can be found in a few CPU seconds, since the relative gap decreases significantly faster because of the tightening constraints.

4.1.2. Overall Earliness. To evaluate the impact of the tightening constraints {(17), (22)−(24)} providing a threshold value for the earliness of task i and a closer upper bound for C_i, several instances of Examples 1A and 1B were solved again; however, this time the least overall earliness has been the selected problem objective. The results found by alternatively using the general precedence model of Méndez et al.,[24] called M1, and the proposed unit-dependent general precedence formulation with either the constraint (17) or the full set of tightening constraints {(17), (22)−(24)}, called M2 and M3, respectively, have all been included in Tables 6−8.

Direct comparison between models M1 and M2 for $n = 20$ batches to measure the effect of cut (17) shows an improvement in the computational time as large as 47:1 for the sequence-independent problems (Example 1A) and 24:1 for the sequence-dependent problems (Example 1B). When the number of batches increases, despite the fact that it comprises a higher number of constraints, the proposed model M3 presents the best computational performance and it can efficiently handle problems of up to 35 batches for the sequence-independent case. In contrast, problems with $n \geq 22$ exceed the computational time limit of 1 h if the approach of Méndez et al.[24] is applied. Comparison of the formulations M2 and M3 shows that the best CPU times are obtained with model M3 for $n > 20$, because the additional constraints (22)−(24) help in providing better lower bounds of the objective function during the branch-and-cut search. When smaller values of n are considered, the additional constraints produce a small tightening effect, and the CPU improvement almost vanishes because of the larger model size. Gantt charts of the optimal schedules found with formulation M3 for both Example 1A with $n = 29$ batches and Example 1B with $n = 25$ batches are shown in Figure 7.

The tightening effect of constraints {(17), (22)−(24)} really is dependent on the existence of nonzero $X_{ii'j}$ variables. In fact, the lower bound provided by the LP relaxation at the root node of the enumeration tree is usually rather poor. However, the allocation of units to tasks gradually forces some sequencing variables by eq (2) to be 1. Therefore, the proposed constraints for the earliness minimization problem do not have a notorious effect until assignments of batches to units are at least partially made. As a result, the increase in the lower bound of the problem value during the search is slower than that observed for the minimum-makespan problem.

In order to further analyze the performance of the proposed formulation with regard to other previous approaches, a comparison with both the discrete-time approach (F1) and the new multiple time-grid formulation (F3) of Castro and Grossmann[25] is also made. Such MILP models assume non-sequence-dependent changeovers. The best solutions reported by Castro and Grossmann[25] for some instances of Example 1A using formulations F1 and F3 have been included in Table 9. Similar to this work, those results were obtained on a Pentium IV 2.8 GHz machine running the commercial solver GAMS/CPLEX 9.0. To keep the problem model on reasonable sizes, the number of time points used by formulation F1 was reduced to 3001, 601, and 301 for problem instances with 12, 29, and 40 batches, respectively. Therefore, problem data were rounded and the optimality of the solution can no longer be guaranteed. The best solutions found by the discrete time approach are different from those provided by continuous-time formulations, because of this rounding procedure. Moreover, the number of time points is

Table 4. Minimum-Makespan Solutions for Example 1 using the Model of Méndez et al.[24]

				Example 1A: Sequence-Independent Setup Times				Example 1B: Sequence-Dependent Setup Times			
n	binary variables	continuous variables	constraints	objective function	relative gap (%)	CPU time (s)	number of nodes	objective function	relative gap (%)	CPU time (s)	number of nodes
12	82	25	214	8.428		19.03	94365	8.645		8.36	39350
16	140	33	382	12.353	2.43	3600[a]	8893218	12.854		1188.50	3421982
18	161	37	444	13.985		2872.81	7166701	14.633	27.07	3600[a]	8708577
20	201	41	558	15.268	22.62	3600[a]	6282059	15.998	21.95	3600[a]	6570231

[a] Resource limit exceeded.

Table 5. Minimum-Makespan Solutions for Example 1 by Incorporating the Valid Cuts 9−10

				Example 1A: Sequence-Independent Setup Times				Example 1B: Sequence-Dependent Setup Times			
n	binary variables	continuous variables	constraints	objective function	relative gap (%)	CPU time (s)	number of nodes	objective function	relative gap (%)	CPU time (s)	number of nodes
12	82	25	218	8.428		0.05	12	8.645		0.05	15
16	140	33	386	12.353		0.03	1	12.854		0.09	44
18	161	37	448	13.985		0.11	27	14.611		40.36	116413
20	201	41	562	15.268		0.14	21	15.998		183.56	417067
22	228	45	622	15.794		0.20	49	16.396		167.09	359804
25	286	51	792	18.218		0.42	110	19.064[a]		79.25	109259
29	382	59	1064	23.302		0.61	82	24.723[a]		5.92	5385
35	532	71	1430	26.683		0.97	90				
40	625	81	1656	28.250		0.91	34				

[a] Relative gap tolerance = 0.01.

Table 6. Minimum-Earliness Solutions for Example 1 using the Model of Méndez et al.[24]

				Example 1A: Sequence-Independent Setup Times				Example 1B: Sequence-Dependent Setup Times			
n	binary variables	continuous variables	constraints	objective function	relative gap (%)	CPU time (s)	number of nodes	objective function	relative gap (%)	CPU time (s)	number of nodes
12	82	24	214	1.026		0.03	22	1.376		0.01	12
16	140	32	382	9.204		1.30	3668	11.647		2.70	8301
18	161	36	444	16.496		38.48	84843	18.773		55.77	123666
20	201	40	558	17.073		77.78	148101	19.131		81.38	159388
22	228	44	618	22.815	1.68	3600[a]	4385294	27.754	8.33	3600[a]	3616973
25	286	50	788	29.430	49.63	3600[a]	2720801	40.541	57.68	3600[a]	3435213

[a] Resource limit exceeded.

Table 7. Minimum Earliness for Example 1 using the Proposed Model with eq (17)

				Example 1A: Sequence-Independent Setup Times				Example 1B: Sequence-Dependent Setup Times			
n	binary variables	continuous variables	constraints	objective function	relative gap (%)	CPU time (s)	number of nodes	objective function	relative gap (%)	CPU time (s)	number of nodes
12	191	24	245	1.026		0.02	1	1.376		0.03	1
16	351	32	437	9.204		0.30	78	11.647		0.56	404
18	408	36	508	16.496		1.19	785	18.773		1.20	853
20	519	40	639	17.073		1.63	807	19.131		3.27	2178
22	574	44	721	22.815		9.11	6598	27.754		90.75	75569
25	738	50	916	29.430		91.14	57741	37.216	15.25	3600[a]	2006319
29	1001	58	1238	59.896	21.11	3600[a,b]	1399091				

[a] Resource limit exceeded. [b] Best possible solution = 47.254.

Table 8. Minimum Earliness for Example 1 using the Proposed Model with eqs (17) and (22)−(24)

				Example 1A: Sequence-Independent Setup Times				Example 1B: Sequence-Dependent Setup Times			
n	binary variables	continuous variables	constraints	objective function	relative gap (%)	CPU time (s)	number of nodes	objective function	relative gap (%)	CPU time (s)	number of nodes
12	191	67	365	1.026		0.03	1	1.376		0.03	1
16	351	104	636	9.204		0.53	59	11.647		0.72	173
18	408	117	727	16.496		1.23	216	18.773		1.39	396
20	519	156	946	17.073		2.30	358	19.131		4.78	1101
22	574	169	1058	22.815		4.02	865	27.754		15.34	5059
25	738	216	1355	29.430		17.53	3023	37.216		1468.53	291899
29	1001	270	1819	59.896		161.88	26139	75.067	7.99	3600[a,c]	634587
35	1353	363	2561	86.595		436.41	41317				
40	1566	439	3047	126.249	8.01	3600[a,b]	209689				

[a] Resource limit exceeded. [b] Best possible solution = 116.14. [c] Best possible solution = 69.068.

Table 9. Comparison of the Results for Example 1A with those reported by Castro and Grossmann[25]

	Example 1A: Sequence-Independent Setup Times							
n	binary variables	continuous variables	constraints	$\|T\|$	RMIP solution	solution found	CPU time (s)	number of nodes
				Discrete Time Formulation				
12	53955	12045	12017	3001	1.03	1.03[a]	7.35	0
29	23190	2405	2434	601	60.183	61.35[a]	300	892
40	14189	1205	1245	301	125.49	133.9	3600[b,c]	29878
				Multiple Time Grid Formulation				
12	120	21	65	5	0	1.026	0.38	470
29	614	45	154	11	37.12	59.896	209	183206
40	1018	57	201	14	85.29	126.927	3600[b,d]	1257613
				Proposed Formulation (M3)				
12	191	67	365		0.828	1.026	0.03	1
29	1001	270	1819		15.896	59.896	161.88	26139
40	1566	439	3047		39.968	126.249	3600[b,e]	209689

[a] A nonoptimal solution. [b] Resource limit exceeded. [c] Best possible solution = 130.47. [d] Best possible solution = 112.14. [e] Best possible solution = 116.14.

also a parameter in the multiple time grid formulation of Castro and Grossmann.[25] As a result, it requires an iterative process that is stopped when no improvement on the objective function is obtained by increasing the number of time points in just one unit. As reported by Castro and Grossmann,[25] for $n = 29$, the same optimal schedule featuring an overall earliness of 59.896 days was found in 209 and 258 s by adopting 11 and 12 time points, respectively. The same optimum was discovered by our approach in 161.88 s (see Figure 7a and Table 8). In addition to requiring at least (209 + 258) = 467 s, the multiple time

Table 10. Problem Sizes and References for Examples 2−6

		Reference	
Example	problem size	Harjunkoski and Grossmann[29]	Castro and Grossmann[25]
2	12 batches, 3 units	single-stage 3 data set 2	P2
3	15 batches, 5 units	single-stage 4 data set 2	P4
4	20 batches, 5 units	single-stage 5 data set 2	P6
5	25 batches, 5 units		P7
6	30 batches, 5 units		P10

grid formulation is unable to rigorously guarantee that the optimal schedule has been discovered. Since it does not require an additional model parameter, the proposed model with eqs (17) and (22)−(24) provides the optimal schedule at once and no iterative procedure is to be executed. Despite the difference in model size, the most restrained formulation M3, which involves the constraints (17) and (22)−(24), generally shows a better computational performance than previous methods. Larger improvements relative to scheduling approaches that also handle sequence-dependent changeovers were obtained.

4.2. Examples 2−6. The next five examples were taken from Castro and Grossmann[25] and involve an increasing number of batches and equipment units. Some of these examples have first been introduced by Harjunkoski and Grossmann.[29] Problem data for Examples 2−4 can be found in Appendix A of Harjunkoski and Grossmann.[29] Data for the remaining examples are reported by Castro and Grossmann.[25] All of them assume negligible setup times. The problem sizes and names used to refer to Examples 2−6 are shown in Table 10. The problem data for Examples 2−6 are all integers and the time horizon is rather short; therefore, one can expect that discrete formulations will feature the best computational performance. Nevertheless, the proposed approach still arises as a very good option and the best one among the continuous-time scheduling methodologies.

4.2.1. Overall Earliness. If the total earliness is minimized, then the tightening constraints (17) and (22)−(24) are added to the mathematical formulation. In Table 11, the computational results found for Examples 2−6 are compared with those obtained with the discrete time (F1), multiple time grid (F3), and sequence-based (F4) formulations, all reported in Castro and Grossmann.[25] The sequence-based formulation considered in Table 11 is a continuous-time representation introduced by Jain and Grossmann[30] that involves a set of sequencing variables $y_{ii'}$ ($i \neq i'$) to indicate that batch i is performed before batch i' in some unit m, whenever $y_{ii'} = 1$. Although the underlying idea is similar to the general precedence concept of Méndez et al.,[24] the model of Jain and Grossmann[30] includes an additional constraint relating allocation and sequencing decisions. Results for the model of Jain and Grossmann[30] used for comparison in Table 11 are those found by Castro and Grossmann[25] on a Pentium IV 2.8 GHz machine using the commercial solver GAMS/CPLEX 9.0.

From Table 11, it follows that the proposed MILP tightened representation arises as the best continuous-time approach for batch scheduling problems, only dominated by the discrete time formulation on examples with a rather short time horizon. Generally, our formulation requires the lowest CPU time among the continuous-time methodologies, regardless of the model size. For Example 2, it requires only 44% of the CPU time needed by the multiple time grid formulation of Castro and Grossmann,[25] and it is 24 times faster than the sequence-based approach of Jain and Grossmann.[30] When the number of batches and units increases, such as in Example 3, the best solution is discovered and proven to be the optimal one in only 25.9 s, versus more than 3000 s required by the other continuous-time formulations. If the number of batches further increases to 30, then the computational performance of our approach becomes deteriorated and no optimal solution is proven after 3600 s. Notice that the number of time slots in the multiple time grid formulation must be selected through an iterative procedure. However, only the CPU time required by the iteration before the last one is reported. Neither the overall time required by all iterations nor the time required by the last iteration $|T| + 1$ is included. For example, the multiple time grid formulation requires 6 time points on each unit to schedule a set of 12 batches within 8.66 s in Example 2. The solution found is said to be the optimal one because it has the same objective value that the solution provided by the discrete time formulation. However, it can be expected that the required CPU time when the same problem is solved with 7 time points will at least be of the same order of magnitude and, therefore, the total computational cost will double the reported value. Moreover, it may happen that the 7-time-point problem cannot be solved within the CPU time limit and the optimal value for 7 time points is indeed unknown. In such a case, the optimality of the reported solution could not be proven.

Interestingly, the number of binary variables of the proposed approach is at least one order of magnitude higher than the number of variables required by the other continuous time formulations included in Table 11. Moreover, it can be found that the relaxed RMIP solution of the proposed tightened formulation is equal to zero for Examples 2−6. However, the additional cuts (14), (15), (17), and (21)−(24) allow to enhance the lower bound of the objective function as the number of branches in the implicit enumeration tree increases and, therefore, an improved computational performance is achieved.

Some comments on the discrete approach need to be made. As pointed out by Castro and Grossmann,[25] the larger size of the discrete formulation caused by the increase on the number of batches from 12−20 in Examples 2−4 to 25−30 in Examples 5 and 6 was, to some extent, compensated by shortening the scheduling horizon in the later examples, thus requiring fewer time intervals. In addition to that, the problem data had only two significant digits and, consequently, an interval length of $\delta = 1$ can be adopted. As a result, the optimal solutions found were proven to be optimal because no approximation of the problem data was needed. However, the discrete time formulation will present a worse behavior when the time horizon becomes longer or the number of significant digits of the problem data requires a quite granulated δ. To overcome this situation and maintain problem tractability, an increase on the interval length δ is needed to cut the number of intervals; therefore, the problem data must be rounded and the solution optimality is no longer guaranteed.

4.2.2. Minimizing the Overall Earliness with Sequence-Dependent Changeovers. Examples 2 and 3 are revisited to show that batch scheduling problems with sequence-dependent changeovers can be solved with similar computational efficiency. Since the MILP models proposed by Castro and Grossmann[25] all assume sequence-independent setup times, no comparison with those approaches could be included. Six groups or families of batches have been defined in Table 12, and sequence-dependent setup times for every ordered pair of groups are listed in Table 13. Families G_1−G_5 are used to solve Example 2, and the remaining group (G_6) is added when examining Example 3.

The computational results obtained by applying both the approach of Méndez et al.[24] and the proposed formulation with valid cuts are compared in Table 14. They all were found using

Table 11. Computational Results for Earliness Minimization (Examples 2−6)

solution approach	binary variables	continuous variables	constraints	|T|	solution found	CPU time (s)	number of nodes
			Example 2				
discrete time[a]	8914	1144	1156	381	98	1.56	[b]
multiple time grid[a]			[b]	6	98	8.66	[b]
sequence-based[a]			[b]		98	95.0	[b]
proposed formulation	432	78	536		98	3.88	1497
			Example 3				
discrete time[a]	16950	1856	1871	371	24	0.70	0
multiple time grid[a]	325	26	101	5	24	3600[c,d]	2685905
sequence-based[a]	285	31	2976		24	3226	649506
proposed formulation	1125	89	1012		24	25.91	7057
			Example 4				
discrete time[a]	23320	1906	1926	381	84[e]	1.23	0
multiple time grid[a]	530	31	126	6	84[e]	3600[c,d]	1926341
proposed formulation	2000	126	1745		98	1987.7	193933
			Example 5				
discrete time[a]	13548	1471	1496	294	51	0.98	0
proposed formulation	3125	315	3048		51	1589.4[f]	160654
			Example 6				
discrete time[a]	16753	1471	1501	294	298	2.09	0
proposed formulation	4500	454	4408		469	3600[c,d]	50890

[a] As reported by Castro and Grossmann.[25] [b] Unreported data. [c] Resource limit exceeded. [d] Best possible solution = 0. [e] A nonfeasible solution. [f] Solution found using the proposed model with eq (17).

Table 12. Families of Batches for Examples 2 and 3

family	batches
G_1	O_1, O_2, O_3
G_2	O_4, O_5
G_3	O_6, O_7
G_4	O_8, O_9, O_{10}
G_5	O_{11}, O_{12}
G_6	O_{13}, O_{14}, O_{15}

Table 13. Sequence-Dependent Setup Times for Every Ordered Pair of Groups at Examples 2 and 3

	Sequence-Dependent Setup Time, $\tau_{G,G'}$					
	G_1	G_2	G_3	G_4	G_5	G_6
G_1	2	5	3	2	4	2
G_2	3	3	4	6	3	5
G_3	5	2	1	2	2	6
G_4	3	6	3	2	3	4
G_5	2	5	2	4	1	3
G_6	4	6	6	5	4	3

the same computing platform and the same CPLEX-solver version that were mentioned at the beginning of this section. It is simple to conclude that our tightening formulation presents a much higher computational efficiency. For both examples, the handling of additional sequence-dependent changeovers somewhat deteriorates the minimum overall earliness, as follows from the results shown in Tables 11 and 14. However, the CPU time required by the tightening model just increases from 3.88 s to 6.89 s in Example 2 and from 25.91 s to 38.06 s in Example 3. Moreover, the number of variables and constraints does not vary at all. In short, the proposed formulation can effectively manage sequence-dependent setup times.

5. Conclusions

This work presents an efficient scheduling tool for single-stage multiproduct facilities with sequence-dependent changeovers. It is a continuous-time sequencing approach based on a MILP formulation that mostly relies on the general precedence notion of Méndez et al.[24] Assignment and sequencing decisions related to a particular task i have been decoupled, and the later ones permit to select the tasks processed before task i in the same equipment item. In the proposed model, however, a different set of sequencing variables for each unit rather than a single one for the entire facility has been defined, i.e., a unit-specific general precedence scheme. By explicitly including the equipment index in the domain of the sequencing variables, additional nontrivial tightening constraints producing better lower bounds on the optimal values of alternative objective functions (such as makespan or overall earliness) or key variables (such as task starting/completion times) have been developed. They are expressed in terms of allocation and/or sequencing variables. On one hand, the values of the assignment variables that are allocating units to batches constitute valuable information to derive tight lower bounds on the optimal makespan during the search procedure. Because the processing time is usually much longer than the setup time for any task, then the overall workload of every equipment unit can be very well approximated by just considering the run times of the allocated tasks. On the other hand, the completion of a particular task i must often occur well before the specified due date dd_i, so that other tasks succeeding task i in the same processing queue can be finished on time. In such cases, task i will feature a necessary earliness ($E_i > 0$) and a tighter estimation of the latest completion time (LCT_i) lower than dd_i can be established using the information provided by the sequencing variables related to task i.

By incorporating those valid cuts in the problem formulation, the integrality gap is strongly reduced as the solution algorithm progresses and the nonzero binary variables at the optimum gradually increase to 1. In this way, the quality of the lower bound provided by the corresponding LP at every node of the enumeration tree is continually improved. They are demonstrated to be valid cuts that never exclude integer solutions from the feasible space. Several instances of six large examples involving up to 40 batches and 5 parallel units have been addressed. Despite the larger number of sequencing variables, the proposed approach with the valid cuts clearly outperforms other MILP continuous-time methodologies when they are applied to batch scheduling problems with sequence-dependent changeovers.

Table 14. Computational Results for Examples 2 and 3 with Sequence-Dependent Changeovers

Example	binary variables	continuous variables	constraints	minimum earliness	CPU time (s)	number of nodes
Formulation of Méndez et al.[24]						
2	102	24	444	111	282.47	808997
3	180	30	1110	33	3600[a,b]	3358784
Proposed Formulation						
2	432	78	536	111	6.89	3194
3	1125	89	1012	33	38.06	10731

[a] Resource limit exceeded. [b] Best possible solution = 0.

Appendix: A Valid Lower Bound on the Earliness of a Batch

Proposition: The threshold earliness value $\underline{E}_i$ given by constraint (15) is always a lower bound on the earliness of batch $i \in I$.

Proof: To make the proof, four different cases will be considered. In the first three cases, it will be initially assumed that task i processed in unit $j \in J_{i,i1,i2}$ is followed by a pair of succeeding tasks ($i1$, $i2$) on the jth queue (i.e., $T_i = \{i1, i2\}$).

Moreover, tasks ($i1$, $i2$) will belong to either IA_i or IB_i. Therefore, some contribution on the earliness of task i is expected from such tasks ($i1$, $i2$). In contrast, succeeding tasks $i' \in IC_i$ also processed in unit j are ignored. A fourth case is finally analyzed where some earliness of task i is indirectly caused by an element of IC_i. For the sake of simplicity, the setup times are assumed to be negligible. The proof for $|T_i| = 2$ is then generalized for an arbitrary number of succeeding tasks. In the analysis, two elements of T_i are important: (a) the controlling task i_c, and (b) the last task i_n on the subsequence T_i. The controlling task i_c is the first task on the subsequence T_i that is completed just in time: $C_c = dd_c$. Moreover, unit j does not remain idle at all from C_i to C_c. In turn, $C_n = \max_{i' \in T_i} C_{i'}$.

A1: Case I. Every task succeeding task i on the queue of unit j has a due date not later than dd_i (i.e., $T_i \subseteq IA_i$). Let us assume that the sequence of succeeding tasks is given by $T_i = \{i1, i2\}$. Then,

$$\text{(a)} \quad X_{i,i1,j} = X_{i,i2,j} = 1$$

$$\text{(b)} \quad dd_{i1} < dd_{i2} \le dd_i$$

Case I.1: Task $i1$ controls the necessary earliness of task i to meet all due-dates constraints.

In Figure A1, tasks ($i1$, $i2$) have been arranged by increasing due dates so as to decrease the necessary earliness E_i. Because $i1$ is the controlling task, unit j does not remain idle waiting for the next batch to be processed until the completion of task $i1$. After that, some idle time may be observed. Then,

$$E_i \ge pt_{i1j} + pt_{i2j} + (dd_i - dd_{i2})$$

The threshold value for E_i given by eq (15) is

$$\underline{E}_i = pt_{i1j} + pt_{i2j}$$

Since $dd_i - dd_{i2} \ge 0$, then $E_i \ge \underline{E}_i$.

Case I.2: Task $i2$ controls the necessary earliness of task i to meet the due dates of all tasks succeeding task i in unit j (see Figure A2).

In this case, unit j is not idle until the due date of the controlling task dd_{i2}. Then, the necessary earliness of task i is,

$$\underline{E}_i = pt_{i1j} + pt_{i2j} + (dd_i - dd_{i2})$$

and the threshold value of E_i given by eq (15) is $\underline{E}_i = pt_{i1j} + pt_{i2j}$. Because $dd_i - dd_{i2} \ge 0$, then $E_i \ge \underline{E}_i$. Unique expressions of E_i and $\underline{E}_i$ for Case I can then be written as follows:

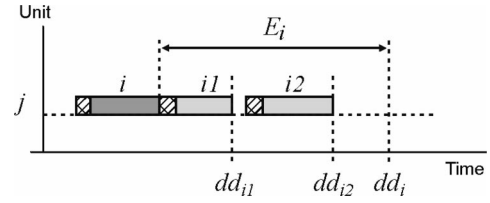

Figure A1. Necessary earliness of task i for Case I.1.

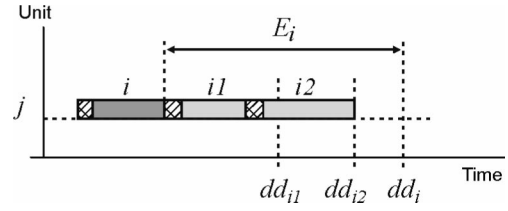

Figure A2. Necessary earliness of task i for Case I.2.

$$E_i \ge pt_{i1j} + pt_{i2j} + (dd_i - dd_{i2})$$

$$\underline{E}_i = pt_{i1j} + pt_{i2j} \le E_i$$

Generalization for multiple succeeding tasks: Let us assume that task i is processed in unit j and $T_i \subseteq IA_i$ denotes the sequence of tasks succeeding task i. Because $i' \in T_i$, then $X_{ii'j} = 1$ and the expressions of E_i and $\underline{E}_i$ for Case I are given by

$$E_i \ge \sum_{i' \in T_i} pt_{i'j} X_{ii'j} + (dd_i - dd_n) X_{ii_n j}$$

$$\underline{E}_i \ge \sum_{i' \in T_i} pt_{i'j} X_{ii'j}$$

where dd_n is the due date of the last task in the sequence T_i. Since $dd_i - dd_n > 0$, it follows that $E_i \ge \underline{E}_i$. Because the processing unit performing task i is unknown before solving the problem formulation, more general expressions of E_i and $\underline{E}_i$ for Case I can be written as follows:

$$E_i \ge \sum_{i' \in IA_i} \sum_{j \in J_{ii'}} pt_{i'j} X_{ii'j} + \underset{i' \in IA_i}{\text{Min}}[(dd_i - dd_{i'})(\sum_{j \in J_{ii'}} X_{ii'j})]$$

$$\underline{E}_i \ge \sum_{i' \in IA_i} \sum_{j \in J_{ii'}} pt_{i'j} X_{ii'j}$$

A2: Case II. Every task succeeding task i on the queue of unit j has a due date later than dd_i and belongs to the set IB_i. Assuming that $T_i = \{i1, i2\} \subseteq IB_i$, then the following conditions are satisfied:

$$\text{(a)} \quad X_{i,i1,j} = X_{i,i2,j} = 1$$

$$\text{(b)} \quad dd_i < dd_{i1} < dd_{i2}$$

$$\text{(c)} \quad pt_{i1j} > (dd_{i1} - dd_i)$$

$$\text{(d)} \quad pt_{i2j} > (dd_{i2} - dd_i)$$

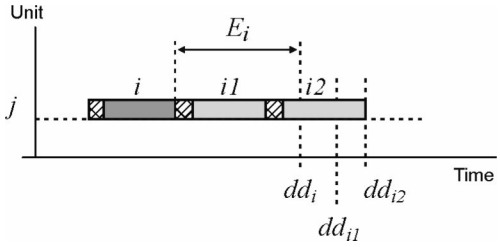

Figure A3. Necessary earliness of task i for Case II.

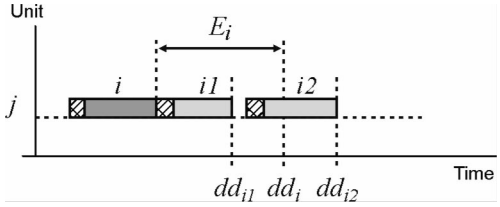

Figure A4. Necessary earliness of task i for Case III.1.

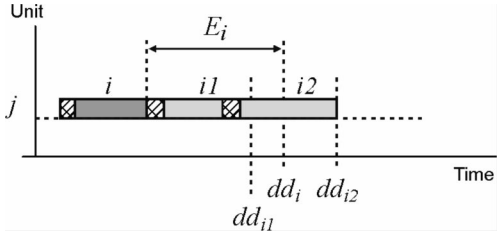

Figure A5. Necessary earliness of task i for Case III.2.

In Figure A3, tasks ($i1$, $i2$) have been arranged by increasing due dates so as to decrease the necessary earliness E_i. In this case, the last task $i2$ in the sequence T_i will control the necessary earliness E_i.

Since task $i1$ is scheduled before task $i2$ and because $\mathrm{pt}_{i2j} > (\mathrm{dd}_{i2} - \mathrm{dd}_i)$, then $i1$ must be completed before time dd_i, and its processing time pt_{i1j} is completely included on the exact earliness:

$$E_i = \mathrm{pt}_{i1j} + \mathrm{pt}_{i2j} - (\mathrm{dd}_{i2} - \mathrm{dd}_i)$$

On the other hand, according to eq (15),

$$\underline{E}_i = \mathrm{pt}_{i1j} + \mathrm{pt}_{i2j} - (\mathrm{dd}_{i2} - \mathrm{dd}_i) - (\mathrm{dd}_{i1} - \mathrm{dd}_i)$$

Since $\mathrm{dd}_{i1} - \mathrm{dd}_i > 0$, then $E_i > \underline{E}_i$.

Generalization for multiple succeeding tasks: Let us now assume that task i is processed in unit j and $T_i \subseteq \mathrm{IB}_i$ is the set of succeeding tasks satisfying the conditions

$$\text{(a)} \quad \mathrm{dd}_{i'} > \mathrm{dd}_i$$

$$\text{(b)} \quad \mathrm{pt}_{i'j} > (\mathrm{dd}_{i'} - \mathrm{dd}_i) \qquad (\text{for any } i' \in T_i)$$

Because the last task in the sequence T_i (task i_n) is also the controlling task, then the expressions of E_i and $\underline{E}_i$ for Case II will be given by:

$$E_i \geq \sum_{i' \in T_i} \mathrm{pt}_{i'j} X_{ii'j} - (\mathrm{dd}_n - \mathrm{dd}_i) X_{ii_nj}$$

$$\underline{E}_i = \sum_{i' \in T_i} [\mathrm{pt}_{i'j} - (\mathrm{dd}_{i'} - \mathrm{dd}_i)] X_{ii'j} = \sum_{i' \in T_i} \mathrm{pt}_{i'j} X_{ii'j} + \sum_{i' \in T_i} (\mathrm{dd}_{i'} - \mathrm{dd}_i) X_{ii'j}$$

Since $(\mathrm{dd}_{i'} - \mathrm{dd}_i) > 0$ for any $i' \in T_i$ and the last summation on the RHS of $\underline{E}_i$ also includes the term $(\mathrm{dd}_n - \mathrm{dd}_i)$, then $E_i \geq \underline{E}_i$. However, the processing unit performing task i is unknown before solving the mathematical model. Therefore, the general expressions of E_i and $\underline{E}_i$ for Case II are:

$$E_i \geq \sum_{i' \in \mathrm{IB}_i} \sum_{j \in \mathrm{JB}_{ii'}} \mathrm{pt}_{i'j} X_{ii'j} - \underset{i' \in \mathrm{IB}_i}{\mathrm{Max}} [(\mathrm{dd}_{i'} - \mathrm{dd}_i)(\sum_{j \in \mathrm{JB}_{ii'}} X_{ii'j})]$$

$$\underline{E}_i = \sum_{i' \in \mathrm{IB}_i} \sum_{j \in \mathrm{JB}_{ii'}} \mathrm{pt}_{i'j} X_{ii'j} + \sum_{i' \in \mathrm{IB}_i} \sum_{j \in \mathrm{JB}_{ii'}} (\mathrm{dd}_{i'} - \mathrm{dd}_i) X_{ii'j}$$

A3: Case III. Task i features an intermediate due date, relative to the succeeding tasks T_i. Therefore, $T_i = T_i^{\mathrm{A}} \cup T_i^{\mathrm{B}}$, where $T_i^{\mathrm{A}} \subseteq \mathrm{IA}_i$ and $T_i^{\mathrm{B}} \subseteq \mathrm{IB}_i$. Let us now assume that $T_i = \{i1, i2\}$, such that $i1 \in \mathrm{IA}_i$ and $i2 \in \mathrm{IB}_i$. Therefore, the following conditions are satisfied:

$$\text{(a)} \quad X_{i,i1,j} = X_{i,i2,j} = 1$$

$$\text{(b)} \quad \mathrm{dd}_{i1} < \mathrm{dd}_i < \mathrm{dd}_{i2}$$

$$\text{(c)} \quad \mathrm{pt}_{i2j} > (\mathrm{dd}_{i2} - \mathrm{dd}_i)$$

Moreover, tasks ($i1$, $i2$) have been arranged by increasing due dates to decrease the earliness E_i.

Case III.1: Task $i1 \in \mathrm{IA}_i$, featuring a due date not later than dd_i, controls the necessary earliness of task i to meet all the due-date constraints (see Figure A4).

Similar to Case I.1, the necessary earliness of batch i is

$$E_i = \mathrm{pt}_{i1j} + (\mathrm{dd}_i - \mathrm{dd}_{i1})$$

while the threshold value $\underline{E}_i$ given by eq (15) is

$$\underline{E}_i = \mathrm{pt}_{i1j} + \mathrm{pt}_{i2j} - (\mathrm{dd}_{i2} - \mathrm{dd}_i)$$

However,

$$\mathrm{dd}_i - \mathrm{dd}_{i1} \geq \mathrm{pt}_{i2j} - (\mathrm{dd}_{i2} - \mathrm{dd}_i)$$

Therefore:,

$$E_i \geq \mathrm{pt}_{i1j} + \mathrm{pt}_{i2j} - (\mathrm{dd}_{i2} - \mathrm{dd}_i) \quad \text{and} \quad E_i > \underline{E}_i$$

Case III.2: Task $i2 \in \mathrm{IB}_i$, featuring a due date later than dd_i, controls the necessary earliness of task i to meet all due-date constraints (see Figure A5).

Because $i2$ is the controlling task, it follows that:

$$E_i = \mathrm{pt}_{i1j} + \mathrm{pt}_{i2j} - (\mathrm{dd}_{i2} - \mathrm{dd}_i)$$

According to eq (15),

$$\underline{E}_i = \mathrm{pt}_{i1j} + \mathrm{pt}_{i2j} - (\mathrm{dd}_{i2} - \mathrm{dd}_i)$$

Therefore, $E_i = \underline{E}_i$.

A common expression of E_i for the two subcases III.1 and III.2 is

$$E_i \geq \mathrm{pt}_{i1j} + \mathrm{pt}_{i2j} - (\mathrm{dd}_{i2} - \mathrm{dd}_i)$$

Generalization for multiple succeeding tasks: Assuming that task i is processed in unit j and T_i is the sequence of tasks succeeding task i, such that $T_i = T_i^{\mathrm{A}} \cup T_i^{\mathrm{B}}$, where $T_i^{\mathrm{A}} \subseteq \mathrm{IA}_i$ and $T_i^{\mathrm{B}} \subseteq \mathrm{IB}_i$, it follows that the last element i_n of T_i will belong to either T_i^{A} or T_i^{B}. Because every task of T_i should be scheduled between C_i and the completion time of i_n, the possible expressions of E_i for Case III are given by:

$$\text{(a)} \quad E_i \geq \sum_{i' \in T_i} \mathrm{pt}_{i'j} X_{ii'j} + (\mathrm{dd}_i - \mathrm{dd}_n) X_{ii_nj} \qquad (\text{if } i_n \text{ belongs to } T_i^{\mathrm{A}})$$

(b) $E_i \geq \sum_{i' \in T_i} pt_{i'j} X_{ii'j} - (dd_n - dd_i) X_{ii_n j} \quad \left(\text{if } i_n \text{ belongs to } T_i^B\right)$

where dd_n is the due date of the last task i_n.

If the positive term $(dd_i - dd_n) X_{ii_n j}$ in expression (a) is omitted, the inequality is still valid. A general expression of E_i for Case III then can be written as follows:

$$E_i \geq \sum_{i' \in T_i} pt_{i'j} X_{ii'j} - \underset{i' \in T_i^B}{\text{Max}}[(dd_{i'} - dd_i) X_{ii'j}]$$

where j is the unit allocated to task i.

In turn, the threshold value for E_i is given by

$$\underline{E}_i \geq \sum_{i' \in T_i} pt_{i'j} X_{ii'j} - \sum_{i' \in T_i^B} (dd_{i'} - dd_i) X_{ii'j}$$

Since $dd_{i'} - dd_i \geq 0$ for any $i' \in T_i^B$ and the last summation on the RHS of $\underline{E}_i$ also includes the term $(dd_n^B - dd_i)$, then $E_i \geq \underline{E}_i$. More general expressions for E_i and $\underline{E}_i$ are given by:

$$E_i \geq \sum_{i' \in IA_i} \sum_{j \in J_{ii'}} pt_{i'j} X_{ii'j} + \sum_{i' \in IB_i} \sum_{j \in JB_{ii'}} pt_{i'j} X_{ii'j} - \underset{i' \in IB_i}{\text{Max}}[(dd_{i'} - dd_i)(\sum_{j \in JB_{ii'}} X_{ii'j})]$$

$$\underline{E}_i = \sum_{i' \in IA_i} \sum_{j \in J_{ii'}} pt_{i'j} X_{ii'j} + \sum_{i' \in IB_i} \sum_{j \in JB_{ii'}} pt_{i'j} X_{ii'j} - \sum_{i' \in IB_i} \sum_{j \in JB_{ii'}} (dd_{i'} - dd_i) X_{ii'j}$$

A4: Case IV. Let us assume that a particular task i' succeeding task i in the queue of unit j has a due date later than dd_i and belongs to the set IC_i (i.e., $pt_{i'j} < dd_{i'} - dd_i$). In addition, the processing time of i' in unit j satisfies the following condition: $pt_{i'j} > dd_{i'} - dd_k$ for some task $k \in IB_i$. Therefore, task i' may force to complete task k earlier than needed and thus indirectly contribute to the earliness of task i. In other words, task i' may become the controlling task ($i' = i_c$) and, therefore, the unit j will not remain idle from time C_i to time $dd_{i'}$. Let us assume that T_i represents the sequence of succeeding tasks up to task i_c. Then, i_c is the last task of T_i ($i_c = i_n$) and the expression for E_i will be

$$\underline{E}_i \geq \sum_{/ \in T_i} pt_{/j} X_{i/j} - (dd_n - dd_i) X_{ii_n j}$$

where T_i also includes some other tasks that belong to IC_i. In addition to the contributions from tasks belonging to IA_i and IB_i, the expression of E_i will also include positive terms related to tasks belonging to IC_i. Therefore, the value of E_i will experience a further increase, relative to the previous three cases. On the contrary, the threshold value for E_i is still given by eq (15):

$$\underline{E}_i = \sum_{i' \in T_i^\#} pt_{i'j} X_{ii'j} + \sum_{i \in T_i^B} (dd_i - dd_i) X_{ii'j}$$

where $T_i^\# \subset T_i$ does not include the contributions of succeeding tasks $i' \in IC_i$. It is straightforward to conclude that $E_i \geq \underline{E}_i$.

Nomenclature

(a) Sets

I = batches or tasks
J = equipment units
J_i = equipment units that can be allocated to task i
$J_{ii'}$ = equipment units that can be allocated to tasks i and i' ($J_{ii'} = J_i \cap J_{i'}$)
IA_i = tasks $i' \neq i$ featuring due dates $dd_{i'} \leq dd_i$ and common units with task i ($J_{ii'} \neq \varnothing$)
IB_i = tasks $i' \neq i$ featuring due dates $dd_{i'} > dd_i$, $J_{ii'} \neq \varnothing$ and $pt_{ij} > dd_{i'} - dd_i$ for some $j \in J_{ii'}$
IC_i = batches that cannot directly deteriorate the earliness of batch i
DD = due dates
DD_i^- = due dates earlier than dd_i ($t \leq dd_i$)
DD_i^+ = due dates later than dd_i ($t > dd_i$)

(b) Parameters

α_i = weighting coefficient for the earliness of batch i
$\beta_{ii'j}$ = threshold contribution of the succeeding task i' in unit j to the earliness of batch i
$\tau_{ii'j}$ = sequence-dependent setup time between batch i and batch i' in unit j
σ_{ij}^{Min} = lowest sequence-dependent changeover for task i in unit j
dd_i = due date for batch i
H = length of the scheduling horizon
pt_{ij} = processing time of batch i at unit j
PT_{ij} = total time allocation of unit j to batch i
$\underline{PT}_{ij}$ = conservative estimation of PT_{ij}
rt_i = release time of batch i
ru_j = ready time of unit j
su_{ij} = setup time of batch i in unit j

(c) Binary Variables

Y_{ij} = denotes that batch i is allocated to unit j
$X_{ii'j}$ = denotes that batch i is run before batch i' in unit j

(d) Continuous Variables

C_i = completion time of batch i
E_i = earliness of batch i
$\underline{E}_i$ = lower bound on the earliness of batch i
EST_i = earliest start time of batch i
LCT_i = latest completion time of batch i
MK = makespan
S_i = starting time of batch i
w_{it} = continuous variable denoting that batch i has been completed before the reference time t (when $w_{it} = 1$)

Acknowledgment

Financial support received from FONCYT-ANPCyT (under Grant PICT 11-14717), from CONICET (under Grant PIP-5729), and from Universidad Nacional del Litoral (under CAI+D 003-13) is fully appreciated.

Literature Cited

(1) Floudas, C. A.; Lin, X. Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. *Comput. Chem. Eng.* **2004**, *28*, 2109.

(2) Méndez, C. A.; Cerdá, J.; Grossmann, I. E.; Harjunkoski, I.; Fahl, M. State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Comput. Chem. Eng.* **2006**, *30*, 913.

(3) Kondili, E.; Pantelides, C. C.; Sargent, W. H. A general algorithm for short-term scheduling of batch operations—I. MILP formulation. *Comput. Chem. Eng.* **1993**, *17*, 211.

(4) Pantelides, C. C. Unified frameworks for optimal process planning and scheduling. In *Proceedings of the Second Conference on Foundations of Computer Aided Operations*; Cache Publications: New York, 1994; p 253.

(5) Shah, N.; Pantelides, C. C.; Sargent, W. H. A general algorithm for short-term scheduling of batch operations—II. Computational issues. *Comput. Chem. Eng.* **1993**, *17*, 229.

(6) Rodrigues, M. T. M.; Latre, L. G.; Rodrigues, L. C. A. Short-term planning and scheduling in multipurpose batch chemical plants: A multilevel approach. *Comput. Chem. Eng.* **2000**, *24*, 2247.

(7) Lee, K.; Park, H. I.; Lee, I. A novel nonuniform discrete time formulation for short-term scheduling of batch and continuous processes. *Ind. Eng. Chem. Res.* **2001**, *40*, 4902.

(8) Giannelos, N. F.; Georgiadis, M. C. A simple new continuous-time formulation for short-term scheduling of multipurpose batch processes. *Ind. Eng. Chem. Res.* **2002**, *41*, 2178.

(9) Maravelias, C. T.; Grossmann, I. E. New general continuous-time state-task network formulation for short-term scheduling of multipurpose batch plants. *Ind. Eng. Chem. Res.* **2003**, *42*, 3056.

(10) Schilling, G.; Pantelides, C. C. A simple continuous-time process scheduling formulation and a novel solution algorithm. *Comput. Chem. Eng.* **1996**, *20*, S1221.

(11) Zhang, X.; Sargent, W. H. The optimal operation of mixed production facilities—a general formulation and some approaches for the solution. *Comput. Chem. Eng.* **1996**, *20*, 897.

(12) Castro, P.; Barbosa-Póvoa, A. P. F. D.; Matos, H. An improved RTN continuous-time formulation for the short-term scheduling of multipurpose batch plants. *Ind. Eng. Chem. Res.* **2001**, *40*, 2059.

(13) Castro, P. M.; Barbosa-Póvoa, A. P.; Matos, H. A.; Novais, A. Q. Simple continuous-time formulation for short-term scheduling of batch and continuous processes. *Ind. Eng. Chem. Res.* **2004**, *43*, 105.

(14) Ierapetritou, M. G.; Floudas, C. A. Effective continuous-time formulation for short-term scheduling. 1. Multipurpose batch processes. *Ind. Eng. Chem. Res.* **1998**, *37*, 4341.

(15) Lin, X.; Floudas, C. A.; Modi, S.; Juhasz, N. M. Continuous-time optimization approach for medium-range production scheduling of a multiproduct batch plant. *Ind. Eng. Chem. Res.* **2002**, *41*, 3884.

(16) Vin, J. P.; Ierapetritou, M. G. A new approach for efficient rescheduling of multiproduct batch plants. *Ind. Eng. Chem. Res.* **2000**, *39*, 4228.

(17) Janak, S. L.; Lin, X.; Floudas, C. A. Enhanced continuous-time unit-specific event-based formulation for short-term scheduling of multipurpose batch processes: Resource constraints and mixed storage policies. *Ind. Eng. Chem. Res.* **2004**, *43*, 2516.

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

(19) Pinto, J. M.; Grossmann, I. E. An alternate MILP model for short-term scheduling of batch plants with preordering constraints. *Ind. Eng. Chem. Res.* **1996**, *35*, 338.

(20) Sundaramoorthy, A.; Karimi, I. A. A simpler better slot-based continuous-time formulation for short-term scheduling in multiproduct batch plants. *Chem. Eng. Sci.* **2005**, *60*, 2679.

(21) Cerdá, J.; Henning, G. P.; Grossmann, I. E. A mixed-integer linear programming model for short-term scheduling of single-stage multiproduct batch plants with parallel lines. *Ind. Eng. Chem. Res.* **1997**, *36*, 1695.

(22) Méndez, C. A.; Henning, G. P.; Cerdá, J. Optimal scheduling of batch plants satisfying multiple product orders with different due-dates. *Comput. Chem. Eng.* **2000**, *24*, 2223.

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

(24) Méndez, C. A.; Henning, G. P.; Cerdá, J. An MILP continuous-time approach to short-term scheduling of resource-constrained multistage flowshop batch facilities. *Comput. Chem. Eng.* **2001**, *25*, 701.

(25) Castro, P. M.; Grossmann, I. E. An efficient MILP model for the short-term scheduling of single stage batch plants. *Comput. Chem. Eng.* **2006**, *30*, 1003.

(26) Castro, P. M.; Grossmann, I. E. New continuous-time MILP model for the short-term scheduling of multistage batch plants. *Ind. Eng. Chem. Res.* **2005**, *44*, 9175.

(27) Ierapetritou, M. G.; Hené, T. S.; Floudas, C. A. Effective continuous-time formulation for short-term scheduling. 3. multiple intermediate due dates. *Ind. Eng. Chem. Res.* **1999**, *38*, 3446.

(28) Méndez, C. A.; Cerdá, J. Dynamic scheduling in multiproduct batch plants. *Comput. Chem. Eng.* **2003**, *27*, 1247.

(29) Harjunkoski, I.; Grossmann, I. E. Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. *Comput. Chem. Eng.* **2002**, *26*, 1533.

(30) Jain, V.; Grossmann, I. E. Algorithms for hybrid MILP/CP models for a class of optimization problems. *INFORMS J. Comput.* **2001**, *13*, 258.

Received for review December 27, 2007
Revised manuscript received July 25, 2008
Accepted August 12, 2008

IE701774W