Article

Material Transfer Operations in Batch Scheduling. A Critical Modeling Issue

Sergio Ferrer-Nadal, Elisabet Capo#n-Garci#a, Carlos A. Me#ndez, and Luis Puigjaner

Ind. Eng. Chem. Res., **2008**, 47 (20), 7721-7732• DOI: 10.1021/ie800075u • Publication Date (Web): 29 August 2008

Material Transfer Operations in Batch Scheduling. A Critical Modeling Issue

Sergio Ferrer-Nadal,[†] Elisabet Capón-García,[†] Carlos A. Méndez,[‡] and Luis Puigjaner*,[†]

Chemical Engineering Department—CEPIMA, Universitat Politècnica de Catalunya, ETSEIB, Av.Diagonal 647, E-08028, Barcelona, Spain, and INTEC (Universidad Nacional del Litoral—CONICET), Güemes 3450, 3000, Santa Fe, Argentina

An effective short-term scheduling formulation must simultaneously deal with several problem difficulties commonly arising in batch processes operations. One of the key features to be considered is the representation of the material transfer operations between process stages. A nonzero time as well as certain conditions and resources are always required to move the material from one processing stage to the next one according to the specified product recipe. The transfer task consumes a period of time during which a proper synchronization of the equipment units supplying and receiving the material is enforced. Synchronization implies that during the execution of the transfer task, one unit will be supplying the material whereas the other one will be receiving it and consequently, no other task can be simultaneously performed in both units. Most of the existing mixed-integer linear programming (MILP) optimization approaches have traditionally dealt with the batch scheduling problem assuming zero transfer times, and consequently no synchronization, between consecutive processing stages. Simplification relying on negligible transfer times may work properly for the scheduling of multiproduct batch plants with similar product recipes; however, it is demonstrated in this work that ignoring the important role of transfer times may seriously compromise the feasibility of the scheduling whenever shared units and storage tanks, material recycles, or bidirectional flows of products are to be considered. To overcome the serious limitations of current MILP-based scheduling approaches, a general precedence-based framework accounting for nonzero transfer times is introduced. Also, two alternative methods that avoid generating unfeasible schedules are proposed and tested in different case studies.

1. Introduction

Scheduling, not only for manufacturing operations, is a common requirement for the industry, management, and services. Scheduling problems can be posed in such assorted areas as personnel planning, computer design, and production management. Broadly speaking, scheduling may be defined as a decision-making process of allocation of scarce resources to a set of activities over time. Operations research has focused in a general way on the research of new solution techniques for this kind of problem since the late 1950s.[1–3] Nevertheless, operations research particularly lacks in the modeling of chemical industry operations, becoming an active field of research for process systems engineers.

An exact optimal solution of the scheduling problem can be obtained by formulating it as a mathematical programming model. However, because of the highly combinatorial complexity of the scheduling problems, models usually need to be simplified to reduce the exponential growth of solution times with problem size. A wide range of different assumptions can be formulated within the modeling according to the intrinsic characteristics of the problem. For example, if a given activity can only be performed in a specific resource, a nonalternative resource policy formulation is adopted. Moreover, model parameters, such as processing times and demands, are assumed deterministic, if uncertainty is not explicitly taken into account.

Another typical simplification is to consider an unlimited intermediate storage (UIS) policy. This approach assumes that, if necessary, intermediate products are immediately stored after processing until next stage starts; that is, unlimited storage is available between every pair of consecutive batch tasks. The aforementioned case entails an unrestrictive policy for intermediate storage management typical in mechanical industries. Instead, the chemical industry is commonly characterized by shared tanks as well as zero-wait, nonintermediate, or finite storage policies. Batch processes generally comprise multiple processing stages, complex process layouts, and topological implications which usually have a significant influence on the short-term scheduling problem complexity.

An additional example of model simplification is the symbolic workshop problem in the operations research, which ignores the transfer times of the pieces between different tasks. Chemical processes cannot stand this simplification. Fluids, contrarily to mechanical pieces, need proper containers to guarantee their handling and require specific units for transfer operations (pumps, piping, vessels, tanks, etc.).

However, in the scheduling of chemical plants, the assumption of negligible transfer times is generally accepted. Authors usually avoid the complexity of managing these transfer operations by arguing that the overall transfer time represents a very small percentage compared to process operation times. Transfer of material between consecutive stages usually needs to take into account a larger number of possible combinations that may lead to an intractable number of constraints. Thus, transfer time modeling has commonly been assumed as an irrelevant feature within the mathematical optimization frameworks, and consequently its importance has been scarcely addressed in the literature. Most of the works explicitly addressing transfer times are only focused on the multiproduct batch case. This is the case of the work of Kim et al.[4] who proposed a mixed-integer nonlinear programming formulation (MINLP) accounting for nonzero transfer times and various storage policies. Ha et al.[5] considered nonzero transfer times using a sequence-based MILP model for a multiproduct plant. Castro and Grossmann[6] proposed a multiple-time-grid resource task network (RTN)

* To whom correspondence should be addressed. E-mail: luis.puigjaner@upc.edu. Tel.: (+34) 934016678. Fax: (+34) 934010979.
[†] Universitat Politècnica de Catalunya.
[‡] INTEC.

10.1021/ie800075u CCC: $40.75
Published on Web 08/29/2008

formulation for multiproduct batch plants and dealt with transfer times using an additional continuous variable. For the more general and highly combinatorial multipurpose case, transfer time management has been an issue hardly treated in the literature. They are usually neglected or assumed to be lumped into the batch processing time.[7] Models relying on the concept of the batch precedence allow a straightforward treatment of the synchronization between consecutive stages and are able to easily deal with the transfer times. Several works have been reported in which the precedence-based scheduling model considers nonzero transfer times.[8–10]

This work focuses on the critical role of transfer times in the batch scheduling problem and points out that the assumptions introduced during the modeling process must be carefully analyzed to avoid generating schedules with unfeasible operational sequences. Accordingly, although simplifications are necessary to reduce problem complexity, they must be carefully considered in order to avoid unreal solutions.

This paper is organized as follows. First, an illustrative example is presented to clearly illustrate the generation of unfeasible scheduling solutions and conclude with a series of claims in the problem statement section. Next, some of the most well-known mathematical programming scheduling formulations are reviewed and analyzed to identify the sources of unfeasible schedules. Furthermore, two approaches to avoid these shortcomings are proposed. The first one is based on a preprocessing algorithm which identifies unfeasible combinations arising from the problem formulation, whereas the second one solves the problem in a two-stage approach. Both strategies are aimed to a MILP problem formulation. Then, three benchmark case studies are addressed, which are used to test the major limitations of existing scheduling formulations. In addition, these results are compared with the solutions provided by the two proposed approaches and their performance is discussed. Finally, some remarks derived from this work are presented in the conclusions section.

2. An Illustrative Example

Let us consider a multipurpose plant producing different products A and B under the assumption of nonintermediate storage policy. Regarding the production recipe, raw material for processing product A is first treated in unit U1 for 3 h and then transferred to unit U2 and processed for 3 additional hours. As commonly happens in multipurpose batch plants, product B shares the same equipment units with product A and it is manufactured first in unit U2 for 2 h and then in unit U1 for 4 h. Transfer times for discharging and loading intermediate materials between both stages are neglected since they are in the order of a few minutes. This assumption may be derived from a zero-wait policy in which intermediate products have to be consumed immediately after production and transfers are usually very fast. Optimal solution requires the minimization of makespan as a criterion to increase the utilization of the resources and the plant efficiency.

For this straightforward example, the solution depicted in the Gantt chart of Figure 1 will be obtained, if most of the MILP formulations available in the literature are applied. The value of the makespan is 7 h. However, this solution is unfeasible in practice because it is impossible to transfer the material from unit U1 to unit U2 and simultaneously also material from U2 to U1. When the first stage of product A is finished, unit U2 needs to be completely empty to receive material from unit U1. But in fact, unit U2 is trying to discharge product B to unit U1. This problem cannot be solved unless an intermediate storage tank is available for transferring one of the intermediates to this

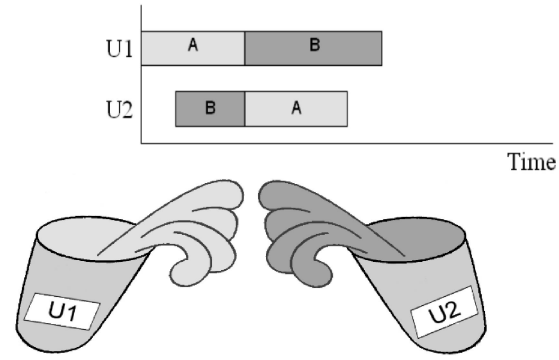

Figure 1. Unfeasible situation for the illustrative example.

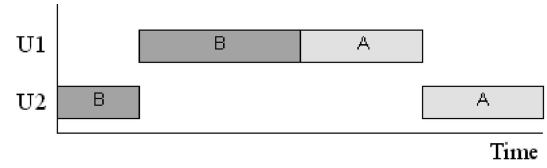

Figure 2. Feasible schedule for the illustrative example.

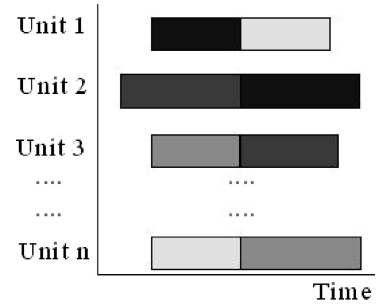

Figure 3. Generalization to *n* products.

storage while the other intermediate is transferred to its next processing unit. Figure 2 shows the feasible optimal solution producing a makespan of 12 h, almost 86% greater than the unfeasible one shown in Figure 1. Here it is worth remarking that the unfeasible schedule cannot be transformed in the feasible one by only performing left or right shifting. New sequencing decisions were also required.

However, an intermediate storage tank does not guarantee feasibility in all instances. For example, deriving from the previous case, assume there is a shared storage tank but three batch transfers are being carried out simultaneously. In this case, two storage facilities would be necessary to make this situation feasible. Therefore, in general, n units form an unfeasible sequence (Figure 3) if the materials are simultaneously transferred between them and there are less than $n-1$ additional intermediate storage units.

3. Problem Statement

Looking at the situation exposed in the illustrative example, the following claims are derived:

Claim 1. The Unfeasible Situations May Only Appear in Multipurpose Plant Configurations, Where Bidirectional Flow Can Be Generated. In multipurpose plants, routes of different products may undergo equipment units in reverse direction. In contrast, multiproduct batch plants always operate following unidirectional flow that cannot generate unfeasible sequences. In this particular case, nonzero transfer times can be easily incorporated to the scheduling by simply extending task durations and performing right shifting.

Claim 2. Unfeasible Solutions May Appear under the Condition of Non-UIS. Storage policy assumption is an important issue. Under the simple consideration of unlimited intermediate storage (UIS), unfeasible solutions do not occur since additional space is always available when needed in transfer operations. However, under more restrictive storage policies, such as common intermediate storage (CIS), finite intermediate storage (FIS), or even nonintermediate storage (NIS), unfeasible situations may arise.

Claim 3. Transfer Times Are an Important Matter for Tasks Synchronization. The main source of these unfeasible solutions has its root in the usual assumption of negligible transfer times. Since transfer times often represent only a small percentage of the whole task duration, in mathematical formulations they are often neglected or just summed up to the processing times in order to reduce the problem complexity. However, in scheduling problems, transfer time plays a key role in terms of task synchronization. Transfer entails that simultaneously to the emptying of a given product from a unit the receiving unit is being filled for a transfer time period. When transfer times are neglected, tasks synchronization among units is ignored, leading to unfeasible solutions in practice as shown in the previous examples. Hence, the modeling of multipurpose batch plants with limited storage policies must consider transfer times. Otherwise, synchronization among tasks regarding transfer times is completely omitted, and unfeasible solutions can be reached. This fact has not been taken into account in most of the existing mathematical formulations for short-term scheduling. Also, the explicit modeling of this feature can be awkward to address by using most of the existing optimization frameworks.

4. Mathematical Programming Formulations for the Batch Scheduling Problem

The scheduling problem has received a great attention over the last decades as a manner to improve the efficiency of batch chemical processes usually aimed at the production of high-added value products. Optimization models for batch scheduling are usually classified according to their time representation. Continuous time formulations allow scheduling events to occur at any time point along time horizon, whereas discrete time formulations have only a fixed number of time points. Discrete time models entail large size problems and even unfeasible schedules may be generated; however, they have proven to be very efficient, adaptable, and convenient for many industrial applications. Continuous time representation reduces the number of variables, and results in more flexible solutions, but at the same time it entails more complicated constraints, and so increases the model complexity.

Moreover, scheduling models can be distinguished according to the way they arrange the event points over the time horizon. Event points are used to guarantee that resource limitations are not exceeded. Discrete time formulations use global time intervals, whereas continuous formulations have a wider range of possible representations. On the one hand, the global time point representation corresponds to a generalization of global time intervals where timing of intervals is treated as a new model variable. This formulation employs a predefined time grid that is valid for all shared resources involved in the scheduling problem. On the other hand, unit specific time events use a different time grid for each resource. Other kinds of formulations are time slots and batch precedence based. Both of them are oriented toward sequential processes. The former defines a set of time intervals of unknown duration, whereas the latter enforces the sequential use of resources through model variables and constraints.

In addition, scheduling problems are also classified depending on the representation of the material balances. These methods are able to deal with arbitrary network processes involving complex product recipes. These models employ the state-task network (STN) or the resource task network (RTN) concept to represent the problem. The STN-based models represent the problem assuming that processing tasks produce and consume states. The RTN-based formulations employ a uniform treatment and representation framework for all available resources through the idea that processing and storage tasks consume and release resources at their beginning and ending times, respectively.

In this work, four of the most popular scheduling continuous-time formulations available in the literature have been selected and implemented for case studies of multipurpose batch plants with different storage policies in a minimization makespan problem. The aim is to analyze these formulations to identify the lack of consideration of transfer times which is leading to incorrect task synchronization. Next, the main features of MILP models based on the STN and RTN global time points, unit specific time events, and general precedence are briefly described. The general precedence model is completely developed in the next section because an extension to the previously published formulation dealing with intermediate storage is also included in this work. Further details related to the other alternative existing MILP optimization models can be found in Méndez et al.[12]

4.1. State-Task-Network-Based Continuous Formulation. There have been many efforts to develop a continuous-time formulation[13,14] based on the original STN representation proposed by Kondili et al.[15] For the testing purpose, we have selected the work by Maravelias and Grossmann[14] because it is able to handle general batch process concepts such as variable batch sizes and processing times, various storage policies, or sequence-dependent changeover times. This approach is based on the definition of a common time grid that is variable and valid for all shared resources. This definition involves time points occurring at unknown time. To guarantee the feasibility of the material balances at any time during the time horizon of interest, the model imposes that all tasks starting at a time point must occur at the same time. However, the ending time does not necessarily have to coincide with the occurrence of a time point, except for those tasks that need to transfer the material with a zero wait time policy. For other storage policies, it is assumed that the equipment can be used to store the material until the occurrence of next time point. The model adopts two binary variables, to denote at which time point a given task starts and finishes. A continuous variable represents the quantity of each resource available at each event point. The number of time intervals is a critical issue for all continuous-time models. The selected approach is to increase the number of time intervals from a relative small number until no improvement in the objective function is achieved.

None of the proposed STN-based continuous-time formulations available in the literature considers transfer times in their formulation. Therefore, it is predictable that this simplification will have a negative effect on the synchronization of tasks and unfeasible optimal results may appear. This fact is explicitly proven in the case studies section.

4.2. Resource-Task-Network-Based Continuous Formulation. The RTN-representation was first introduced by Pantelides.[16] Further improvements were achieved by Castro et al.[17] and again by Castro et al.[18] The improved model version[18] was

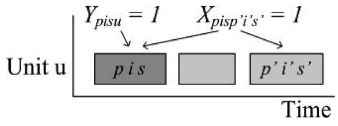

Figure 4. General precedence representation.

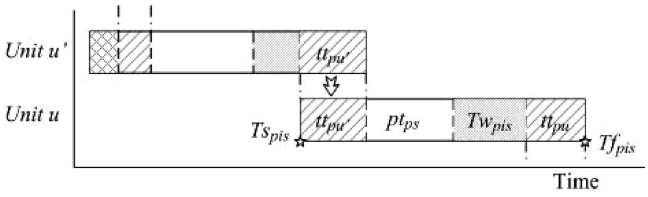

Figure 5. Task duration.

tested in this work. This approach adopts a common time grid for all resources. As other continuous time formulations the length of each time interval is unknown and is to be determined. In addition, a timing parameter is used to define the number of event points allowed between the beginning and ending of a batch task, in order to reduce the number of event points considered and so, the problem complexity. However, an exceedingly small value might prevent the formulation from reaching the global optimum or turn the model unfeasible. The use of a fixed value is a quite reasonable assumption in cases where task processing times are of the same order of magnitude, where it is expected that few events exist between the starting and ending of a given task.

The RTN representation considers two types of items: resources and tasks. A task defines an operation that transforms a certain set of resources into another set at the end of its duration. A resource includes all entities that are involved in process steps, such as materials (raw materials, intermediates, and products), processing and storage equipment (tanks, reactors, etc.) and utilities (operators, steam, etc.). All equipment resources, with the exception of storage tanks, are considered individually, moreover only one task can be executed in any given equipment resource at a certain time. The starting and finishing time points for a given task are defined through only one set of binary variables. It makes the model simpler and more compact, but on the other hand it increases the number of constraints and variables to be defined. The process resource variable represents the excess amount of a given resource at each time point.

RTN continuous models for multipurpose plants reported in the literature do not consider transfer times in their formulation. Therefore, in cases where neglecting transfer times influences the synchronization of tasks, unfeasible optimal results may appear, as proven in the test case studies.

4.3. Unit-Specific Time Event. The original idea of unit-specific events was first presented by Ierapetritou and Floudas[19] and then improved by Vin and Ierapetritou,[20] Lin et al.,[21] and Janak et al.[22] This is a flexible representation of the scheduling problem which is able to account for different intermediate storage policies and other resource constraints. The global time point representation is efficiently reformulated in these models: (a) by considering as an event just the starting of a task and (b) by allowing event points to take place at different times in each different unit. Then, the number of event points and associated binary variables are reduced compared to the global time point representation. Although this representation is mainly oriented to batch network processes, it can easily deal with sequential processes. This formulation requires the definition of the number of event points, especially critical when dealing with resource constraints and inventories. Probably the most functional strategy is starting with a small number of event points and to increase this number iteratively until there is no improvement in the objective function value. This formulation does not account for transfer times between tasks assuming them as negligible compared to the processing times.

The formulation proposed by Janak et al.[22] has been used for the testing purpose of this work. As it can be seen, this formulation also neglects transfer times and so, the entailed synchronization of tasks. Hence, the optimal solutions obtained

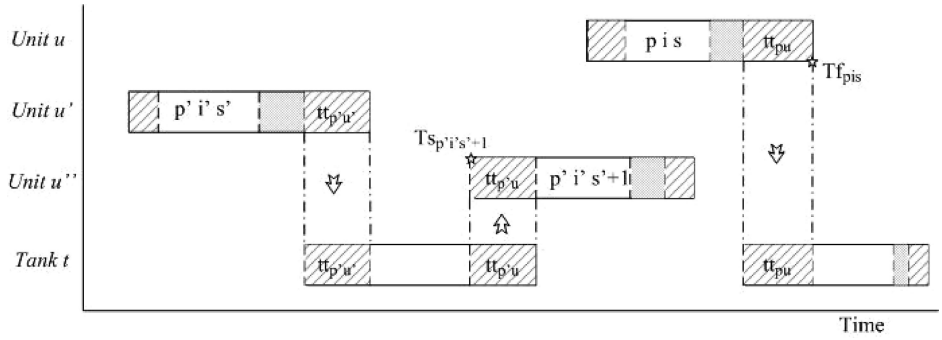

Figure 6. Illustrative representation of storage constraint 10.

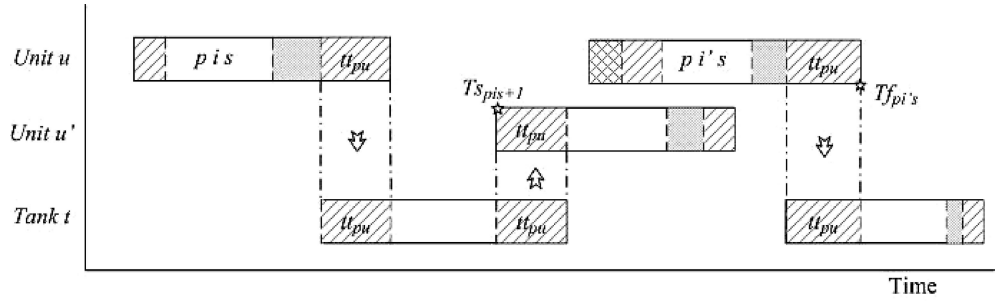

Figure 7. Illustrative representation of storage constraint 11.

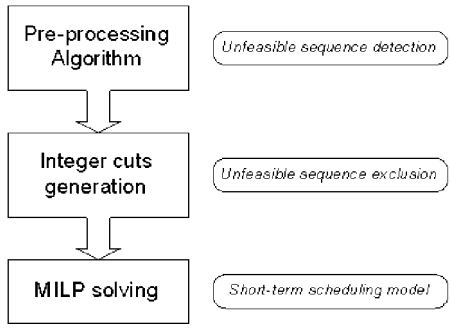

Figure 8. Preprocessing and integer cuts strategy.

with this model may be unfeasible, as it will be proved in the case studies addressed.

4.4. General Precedence. A very convenient approach for dealing with sequential processes is based on the concept of immediate batch precedence which was initially presented by Cerdá et al.[23] Subsequent works[8,11] developed a more efficient continuous-time MILP formulation that relies on the notion of general precedence. This generalized precedence notion extended the immediate predecessor concept to consider all batches belonging to the same processing sequence. As it can be seen in Figure 4, this model defines a couple of sets of binary variables in order to sequence ($X_{pisp'i's'}$) and allocate (Y_{pisu}) processing tasks.

Y_{pisu} is a binary variable equal to 1 whenever task (p,i,s), that is the stage s for manufacturing the batch i of product p, is allocated to equipment unit u. Regarding the sequencing decisions, $X_{pis,p'i's'}$ is a binary variable which establishes the general precedence relationship between a pair of tasks (p,i,s) and (p',i',s') executed at the same processing unit (otherwise $X_{pis,p'i's'}$ is meaningless). If $X_{pis,p'i's'}$ is equal to 1, task (p,i,s) is a direct or nondirect predecessor of task (p',i',s') on the waiting line for the allocated unit. Alternatively, in the case of task (p',i',s') being processed before task (p,i,s) in the same unit, $X_{pis,p'i's'}$ takes the value zero. It is worth noting that the six subindexes defined for sequencing variables are needed to deal with the general scheduling problem arising in multipurpose batch plants, where the same equipment unit can perform several operations related to the same or different products. Consequently, the sequencing variable can distinguish not only the batches and the products involved but also the stages that are being sequenced. Although the number of binary variables seems to be very large at first sight, it should be noted that sequencing variables are only defined for every pair of tasks (p,i,s) and (p',i',s') that can be performed in the same unit, which is an intrinsic characteristic of multipurpose equipment. If the general proposed scheduling method is applied to a multiproduct batch plant, the subindex related to the stages in the sequencing variables are no longer required.

Allocation Constraints. Unit allocation constraint 1 states that a single processing unit must be assigned to every required processing task.

$$\sum_{u} Y_{pisu} = 1 \qquad \forall\, p, i, s, u \tag{1}$$

Constraint 2 expresses the duration of a task and starting ($T_{s\text{-}pis}$) and finish times ($T_{f\text{-}pis}$), by considering the overall time required to perform the loading of material or transfer time from the previous stage ($t_{t\text{-}pu'}$), the batch processing operation itself ($p_{t\text{-}ps}$), a possible waiting time in the processing unit ($T_{w\text{-}pis}$) and unloading of the material to either next stage or to a suitable intermediate storage tank ($t_{t\text{-}pu}$). An illustrative representation of model variables is depicted in Figure 5.

$$T_{f\text{-}pis} = T_{s\text{-}pis} + \sum_{u' \in U_{p(s-1)}} t_{t\text{-}pu'} + \sum_{u \in U_{ps}} (p_{t\text{-}psu} + t_{t\text{-}pu}) Y_{pisu} + T_{w\text{-}pis}$$
$$\forall\, p, i, s \tag{2}$$

Constraints 3 and 4 sequence two batches of two different products processed in the same unit. Constraint 3 is active if task (p,i,s) precedes task (p',i',s') while constraint 4 is active in the opposite case. To reduce the number of binary variables $X_{pisp's'i'}$, constraints 3 and 4 are only used in the case of product p appearing before p' ($p < p'$) or if $p = p'$ for $s < s'$.

$$T_{s\text{-}p'i's'} \geq T_{f\text{-}pis} - M(1 - X_{pisp'i's'}) - M(2 - Y_{pisu} - Y_{p'i's'u}) \tag{3}$$

$$T_{s\text{-}pis} \geq T_{f\text{-}p'i's'} - MX_{pisp'i's'} - M(2 - Y_{pisu} - Y_{p'i's'u}) \tag{4}$$

$$\forall (p,i,s), (p',i',s'), u \in (U_{ps} \cap U_{p's'}) : (p < p') \text{ or } (p = p' \text{ and } s < s')$$

Constraint 5 sequences two batches i and i' of the same product p at the same stage s. Substantial savings in the number of these constraints are achieved considering that batch i is processed before batch i' ($i < i'$).

$$T_{s\text{-}pi's} \geq T_{f\text{-}pis} \;\forall\, p, i < i', s \tag{5}$$

The task precedence constraint 6 is defined for every pair of consecutive tasks that must be sequentially performed for a particular product. One task can never begin before the material from the preceding task starts being transferred to the unit assigned. Transfer times enforce that unloading and loading operations from/to units involving consecutive tasks must be synchronized, unless the material is previously stored in an intermediate storage tank.

$$Tf_{pis} - \sum_{u \in U_{ps}} tt_{pu} Y_{pisu} \leq Ts_{pi(s+1)} \qquad \forall\, p, i, s \tag{6}$$

Storage Constraints. One of the major advantages of the general precedence notion which strongly influences its efficiency is the fact that the same sequencing variables used for a pair of processing tasks can be utilized for their related storage tasks. However, the formulation presented by Méndez and Cerdá[11] should be further generalized by allowing selective interconnection between processing units and storage tank facilities. A new binary variable AT_{pist} denotes whether task (p,i,s) is sent to storage tank t. Then, constraint 7 expresses that material from task (p,i,s) may be stored in a tank t only if processing unit u is connected to storage tank t.

$$AT_{pist} \leq \sum_{u \in T_u} Y_{pisu} \qquad \forall\, p, i, s, t \tag{7}$$

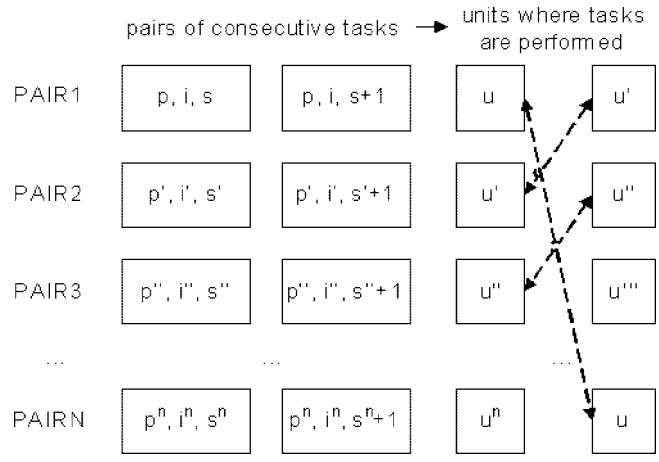

Figure 9. General scheme for a group of tasks that generate an unfeasible sequence.

Constraint 8 works together with 6 in order to sequence two stages of the processing of a batch. In case intermediate storage is not used, both constraints are the same with a different type of inequalities, becoming both in a single equality constraint. Otherwise, constraint 8 is relaxed.

$$T_{\text{f-}pis} - \sum_{u} t_{\text{t-}pu} Y_{pisu} \geq T_{\text{s-}pi(s+1)} - M \sum_{t} \text{AT}_{pist} \qquad \forall\, p, i, s, u \tag{8}$$

Storage task sequencing constraints 9 and 10 (Figure 6) define the order of storage tasks (p,i,s) and (p',i',s') assigned to the same tank. Constraint 9 is only active when task (p',i',s') precedes the task (p,i,s) while constraint 10 is active in the opposite case.

$$T_{\text{f-}p'i's'} - \sum_{u'} t_{\text{t-}p'u'} Y_{p'i's'u'} \geq T_{\text{s-}pi(s+1)} + \sum_{u} t_{\text{t-}pu} Y_{pisu} - M(1 - X_{pisp'i's'}) - M(2 - \text{AT}_{pist} - \text{AT}_{p'i's't}) \tag{9}$$

$$T_{\text{f-}pis} - \sum_{u'} t_{\text{t-}pu} Y_{pisu} \geq T_{\text{s-}p'i'(s+1)} + \sum_{u'} t_{\text{t-}p'u'} Y_{p'i's'u'} - MX_{pisp'i's'} - M(2 - \text{AT}_{pist} - \text{AT}_{p'i's't}) \tag{10}$$

$$\forall (p,i,s), (p,i,s), t, u \in U_{p,s}, u'' \in U_{p,s+1}, u' \in U_{p',s'}$$

In turn, constraint 11 sequences a pair of tasks of two different batches i and i' of the same product p sharing the same intermediate storage t (Figure 7).

$$T_{\text{f-}pi's} - \sum_{u} t_{\text{t-}pu} Y_{pi'su} \geq T_{\text{s-}pi(s+1)} + \sum_{u} t_{\text{t-}pu} Y_{pisu} - M(2 - \text{AT}_{pist} - \text{AT}_{pi's't}) \qquad \forall\, p, i, i', s, t \tag{11}$$

Equation 12 expresses the objective function in terms of makespan minimization.

$$\min \text{MK} \geq T_{\text{f-}pis} \qquad \forall\, p, i, s \tag{12}$$

The general precedence model allows including transfer times in tasks. But, if transfer times are assumed negligible by setting them to zero in the formulation, this model may fail to observe the synchronization of tasks, thus leading to unfeasible schedules. This will be proved in the testing case studies.

5. Solution Approaches

This kind of unfeasible schedule was first identified under NIS conditions by Sanmartí et al.[24] using a novel S-graph representation for the scheduling problem. The S-graph uses a graph representation and in contrast to the MILP-based methods embeds modeling aspects into the solution algorithm. A graph is used to represent recipes, where nodes represent the production tasks and arcs the precedence relationships among them. Then, a branch-and-bound procedure eventually generates the optimal schedule. In the bounding procedure the unfeasible solutions can be detected beforehand. They appear as directed cycles in the graph which are identified and excluded using the algorithm described by Comen.[25] Later on, Romero et al.[26] extended the use of the S-graph framework to include the common intermediate storage policy and then applied a similar algorithm to cycle detection, thus discarding unfeasible solutions. And finally, Ferrer-Nadal et al.[27] carried out a comparative study between the S-graph and a MILP formulation highlighting advantages and inconveniences for both representations. Although S-graph has proved to be a very efficient framework for solving the scheduling problem, it is still a limited framework under development which is not able to include either some modeling aspects in complex plant configurations or other kind of resource constraints than the processing units themselves.

Therefore, two different strategies are presented to deal with unfeasible sequences if the problem specifically requires zero transfer times in order to ensure the correct synchronization between tasks. The first solution method applies a pretreatment algorithm to generate a series of integer cuts that are introduced in the overall MILP formulation as new constraints. The second approach is based on a two stage algorithm which solves sequentially a MILP with very small transfer times and next a linear programming (LP) problem. Both of them are aimed to be embedded in the mathematical programming formulation, more specifically, they can be directly used in the general precedence formulation.

5.1. Pretreatment Algorithm and Integer Cuts. The first method proposed to avoid unfeasible sequences consists of first identifying those groups of tasks that can generate unfeasibilities with a search algorithm, then adding new constraints into the mathematical formulation to avoid them, and finally solving the resulting MILP problem (Figure 8).

For limited storage policies, a potentially unfeasible generated sequence is characterized by groups of n pairs of consecutive tasks such that the equipment unit of the second tasks of all given pairs in the group is also used as equipment unit for the first task of the pairs of tasks in the group (Figure 9).

The proposed algorithm (Figure 10) detects all those groups of tasks that may result in an unfeasible sequence. The algorithm considers a set of products p, their batches i, and that each product is processed in consecutive stages s; so, a task is defined by the product, its batch, and the stage: (p,i,s). CPAR is a set that contains a list of pairs PAIR (CPAR = {PAIR1,..., PAIRN}). Each pair PAIR in the list is defined by two consecutive tasks (p, i, s) and $(p, i, s + 1)$ as well as their corresponding units u and u', respectively. The set of pairs CPAR must contain as many different units as pairs of tasks. If a given group of pairs of tasks stored in CPAR constitutes an unfeasible sequence, then it is stored in the set SUNF.

An iterative search algorithm that looks through all the pairs of consecutive tasks (PAIR($n$pair), $n$pair = 1,..., lastpair) identifies the potential pairs that can create an unfeasible sequence, *C*PA*R*, and finally determines whether a given pair of tasks closes the unfeasible sequence. This algorithm can also be classified as recursive since it is repeated as many times as necessary to increase the group with an additional pair of tasks until an unfeasible sequence is found, and so the number of orders included in a given sequence is represented by $n$level2.

Given the information obtained from this algorithm, a number of integer cuts are added as additional constraints to the original formulation in order to exclude these potentially unfeasible configurations of assignments and sequences. Equation 13 expresses these integer cuts where ZERO and ONE are the sets of tasks (p,i,s) for which $X_{pisp'i's'} = 1$ and $X_{pisp'i's'} = 0$, respectively.

$$\sum_{(pisp'i's') \in \text{ONE}} X_{pisp'i's'} - \sum_{(pisp'i's') \in \text{ZERO}} X_{pisp'i's'} \leq |\text{ONE}| - 1 \qquad \forall\, \text{CPAR} \in \text{SUNF} \tag{13}$$

Computational effort for solving the mathematical problem is not increased since no new variables are created and even the search space is slightly reduced.

5.2. Two-Stage Formulation. The second alternative consists of a two-stage algorithm (Figure 11) for unfeasible schedule removal when zero transfer times are requested. In the first step, very small transfer times are specified to achieve a synchroniza-

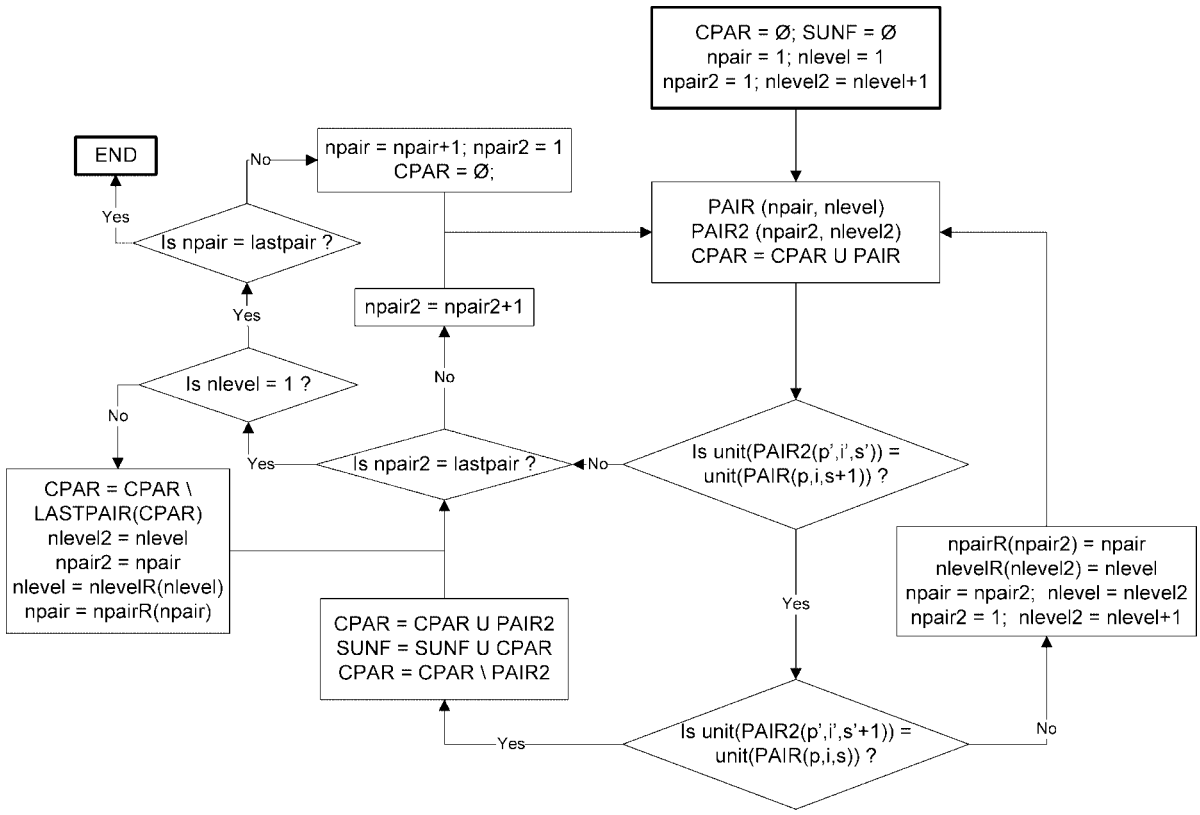

Figure 10. Preprocessing algorithm.

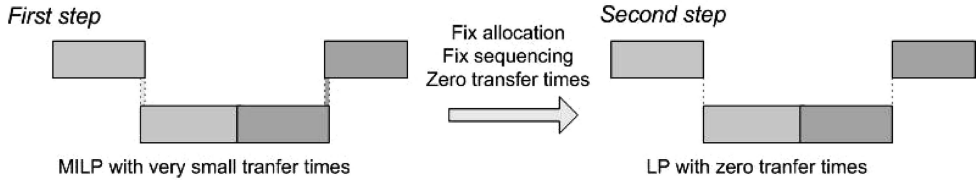

Figure 11. Two-stage algorithm.

Table 1. Production Data for Case Study 1

	stage 1		stage 2		stage 3		
products	unit	time (h)	unit	time (h)	unit	time (h)	number of batches
A	U1	6	U3	9	U4	7	2
B	U2	9	U3	15	U4	17	1
C	U4	8	U1	14	U2	16	1
D	U2	7	U3	11	U1	4	1

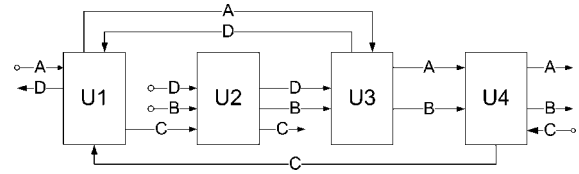

Figure 12. Product recipes for case study 1.

tion which automatically discards unfeasible configurations. Then, from the previous solution, allocation and sequencing variables are fixed, and as a second step, the problem is again solved specifying zero transfer times. In this second step, a LP problem, in which units can be synchronized by performing left or right shifting, is solved. Furthermore, computational effort is almost negligible because all the decision binary variables are already fixed. This makes this strategy very suitable for large problems without extra CPU time.

6. Case Studies

Three testing case studies have been posed to be solved with the four aforementioned mathematical models for batch plant scheduling. They consist of three multipurpose batch plants with different products under different storage policies and under the assumption of zero transfer times. Although the general precedence model explicitly includes transfer times in its formulation, they are set to zero to analyze the results obtained under this assumption. For the sake of simplicity, the objective function consists of the minimization of the makespan. Since optimal unfeasible solutions appear with the previous models, the two proposed solution approaches are applied, and the new results are compared to the previous ones. The mathematical models and algorithm have been implemented in GAMS and solved using the MILP solver CPLEX 9.0.

6.1. Case Study 1. This case study contemplates a multipurpose batch plant which manufactures four products (A, B, C, and D) in three units (U1, U2, and U3) through three processing stages. The production recipe and suitable equipment units are shown in Table 1 and Figure 12. The case study consists of producing two batches of product A, and one batch of each of the other three products. Four instances of the same problem have been solved using different storage policies, namely, unlimited intermediate storage (UIS), nonintermediate

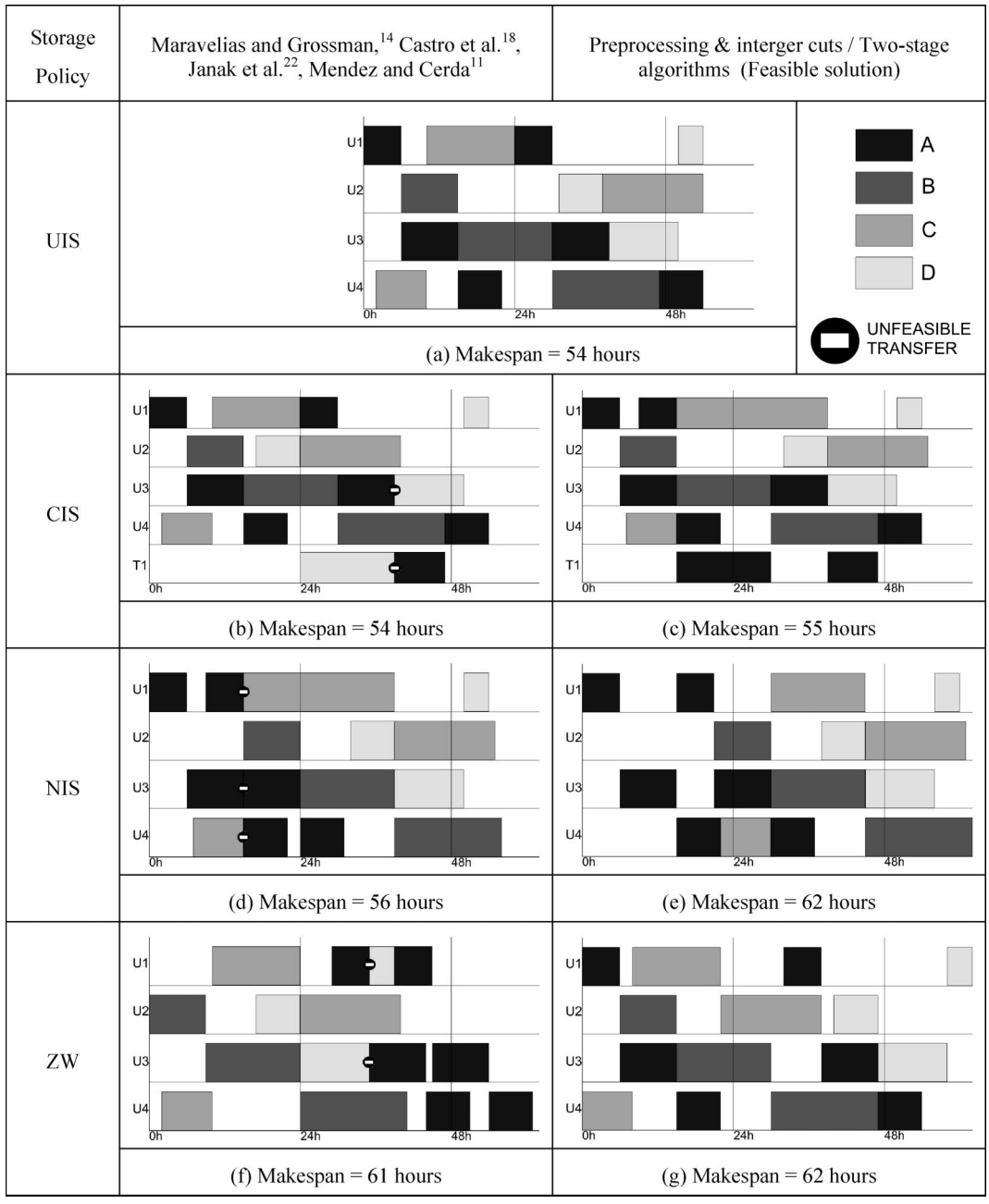

Figure 13. Optimal schedules for case study 1.

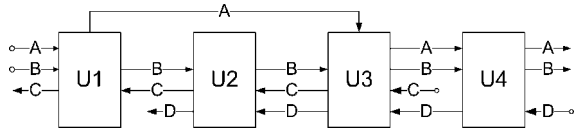

Figure 14. Product recipes for case study 2.

storage (NIS), common intermediate storage (CIS), and zero-wait time (ZW).

Optimal schedules for this case study are presented in Figure 13 along with their corresponding Gantt charts. The feasible solutions found using the two proposed solution approaches have a higher makespan value compared to the solutions given by the four MILP formulations. However, the latter solutions include unfeasible operation sequences, which are avoided in the solutions obtained by applying any of the two solution approaches in this work.

Table 2. Production Data for Case Study 2

	stage 1		stage 2		stage 3		stage 4		
products	unit	time (h)	unit	time (h)	unit	time (h)	unit	time (h)	number of batches
A	U1	15	U3	8	U4	12			1
B	U1	10	U2	20	U3	5	U4	13	1
C	U3	9	U2	7	U1	20			1
D	U4	5	U3	17	U2	7			1

For the UIS scenario, all the formulations produce feasible solutions because enough storage capacity is available to transfer the materials between different units. However, these formulations result in unfeasible situations for the more restrictive cases as NIS, CIS, or ZW. When considering a CIS policy with one available storage tank (Figure 13b), an unfeasible situation is identified when product A is transferred from unit U3 to the storage tank T1 at time 39 h and simultaneously product D is transferred from T1 to U3. Looking at the Gantt chart of the NIS solution in Figure 13d,

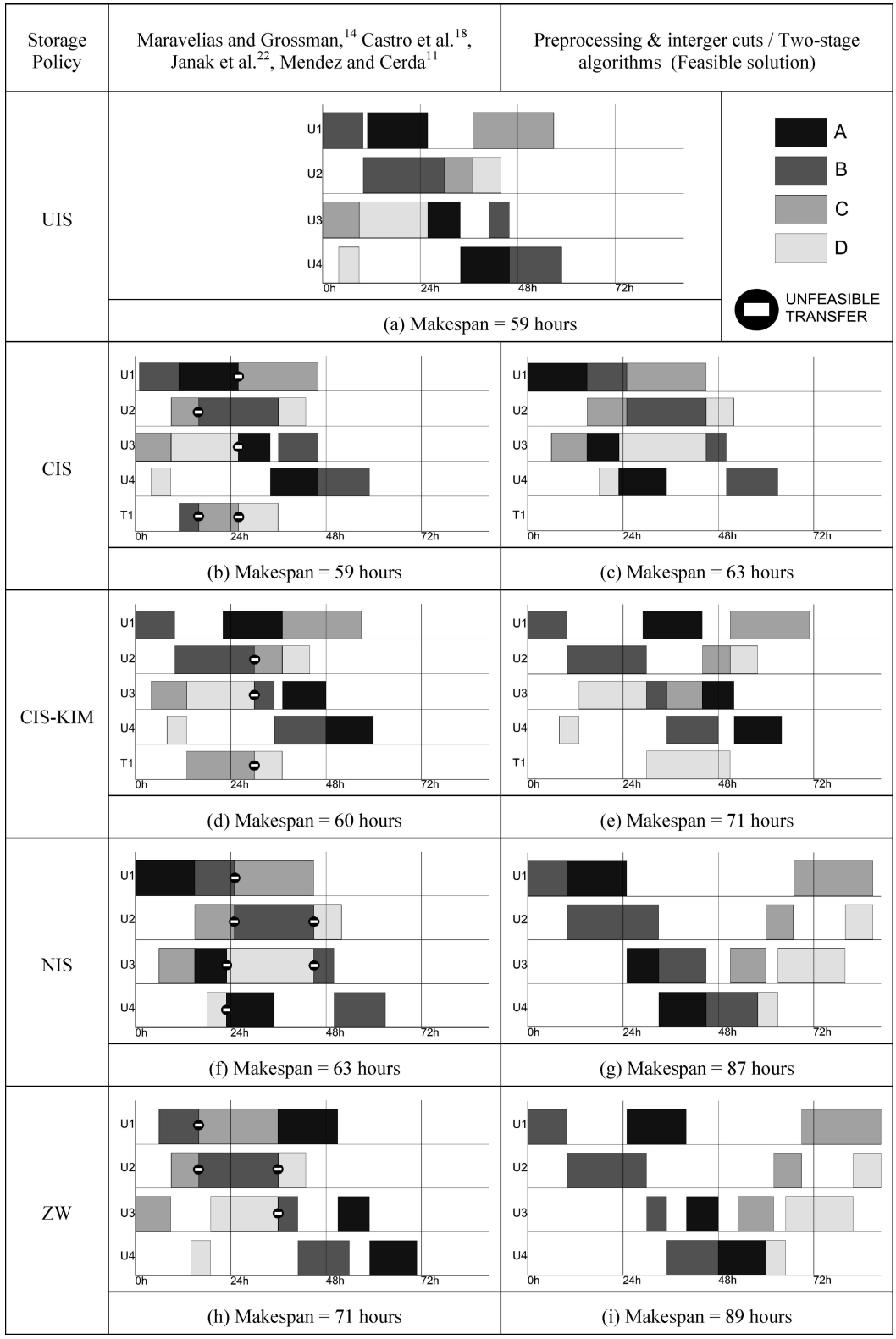

Figure 15. Optimal schedules for case study 2.

three simultaneous transfers take place at time 15 h. The second batch of product A must be transferred from unit U1 to unit U3, the first batch of A from unit U3 to unit U4, and product C from unit U4 to unit U1. These simultaneous transfers can not actually be performed unless two additional storage units would be available at this point. Similar reasoning can be applied to the solution shown in Figure 13f using the most restrictive ZW policy. In the solution given by the four mathematical formulations, product D and the first batch of product A must be transferred simultaneously at time 35 h, product D from unit U3 to unit U1, and the product A from unit U1 to U3.

These unfeasible situations are avoided applying either the proposed preprocessing algorithm plus integer cuts approach or the two-stage algorithm. In terms of makespan, this case study is giving a maximum of 10% difference between both solutions (6 h) for the case of NIS. Here, it is worth remarking that this difference is quite substantial in industrial practice since it would

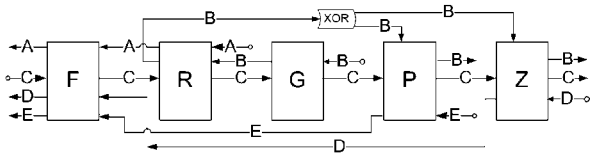

Figure 16. Product recipes for case study 3.

cause a bottleneck of 6 h or an immediate full-rescheduling task when unfeasible situations are observed. Significant differences in sequencing and timing decisions can easily be observed when comparing the Gantt charts corresponding to the feasible schedules and the unfeasible ones.

6.2. Case Study 2. The scheduling problem presented in case study 2 was originally proposed by Kim et al.,[28] and later solved by Méndez and Cerdá.[11] This multipurpose batch plant comprises four products which have to sequentially undergo several processing stages (Figure 14). One single batch of each product is assumed to be manufactured. In the original definition of the problem transfer times were neglected compared to the much longer processing times. The production sequences and processing times for the recipe of each product are stated in Table 2. Although in the problem solved by Kim et al.,[28] a single intermediate storage tank was available for receiving material only from unit U3, in this work we have included alternative scenarios to the same problem in order to evaluate the performance of the different models depending on the adopted intermediate storage policy. The alternatives contemplated are unlimited intermediate storage (UIS), nonintermediate storage (NIS), common intermediate storage tank (CIS), one common intermediate storage only available after unit U3 (CIS-Kim) and zero-wait time (ZW).

Figure 15 presents the comparison between the results obtained after applying the four MILP formulations and the results using any of the two approaches proposed in this work. MILP formulations obtain lower makespan value because, as expected, unfeasible transfer operations within these solutions are encountered.

Only in the case of the UIS policy do MILP formulations lead to an optimum feasible schedule (Figure 15a). For the case in which one tank can be shared by all units (CIS), two unfeasible sequences appear in the Gantt chart of the solution shown in Figure 15b. The first unfeasible transfer takes place at time 16 h with two products involved, products C and B, which are simultaneously transferred from unit U2 to the tank T1 and vice versa. At time 26 h, a second unfeasible transfer occurs with three products involved. Product A is simultaneously transferred (from unit U1 to unit U3) with product C (from tank T1 to unit U1) and product D (from unit U3 to tank T1). Figure 15c shows the feasible solution obtained by applying either of the two approaches proposed in this work. For the case described by Kim et al.[28] with one tank only available after unit U3, an unfeasible situation appears involving three transfers of products at time 30 h (Figure 15d). The difference between the values of the makespan of this unfeasible solution (60 h) and the feasible one shown in Figure 15e (71 h) is almost of 20%. A similar situation corresponds to the NIS policy, where unfeasible sequences take place at times 23, 25, and 45 h (Figure 15f). For this case study, this configuration presents the greatest discrepancy between the makespan values of the unfeasible and the feasible solutions, that is, 24 h. Finally, two unfeasible sequences arise when adopting a ZW policy. Products B and C are transferred simultaneously between units U1 and U2 at time 16 h while products B and D are transferred between units U3 and U2 at time 36 (Figure 15h).

Table 3. Production Data for Case Study 3

	stage 1		stage 2		stage 3		stage 4		stage 5		
products	unit	time (h)	unit	time (h)	unit	time (h)	unit	time (h)	unit	time (h)	number of batches
A	R1	3.5	F	2.5							2
	R2	7									
B	G	3.9	R1	4.1	P	2.9					2
			R2	8.2	Z	3.2					
C	F	4	R1	3.8	G	4.5	P	3	Z	2.9	2
			R2	7.6							
D	Z	5.7	F	3							2
E	P	2.5	F	3							2

6.3. Case Study 3. A multipurpose production plant proposed by Papageorgaki and Reklaitis[29] is revisited in this case study. The scheduling problem comprises the production of five products which have to sequentially undergo different processing stages with alternative units (Figure 16). Two batches of each product are assumed to be manufactured. The production sequences and processing times for the five products are stated in Table 3. The alternative storage policies contemplated are unlimited intermediate storage (UIS), nonintermediate storage (NIS), common intermediate storage tank (CIS), and zero-wait time (ZW).

Figure 17 presents the results obtained applying the four MILP formulations and using any of the two approaches proposed in this work. As in the previous cases, MILP formulations obtain lower makespan value because unfeasible transfer operations within these solutions are encountered.

In the case of UIS policy, MILP formulations lead to an optimum feasible schedule (Figure 17a). If one tank can be shared by all units (CIS), two unfeasible sequences appear in the Gantt chart of the solution shown in Figure 17b. The first unfeasible transfer takes place at time 11.6 h with two products involved, the first batch of product C and the second batch of B, which are simultaneously transferred from unit R2 to unit G and vice versa. At time 12.1 h, a second unfeasible transfer occurs with the first batch of A and the second batch of C, which are simultaneously transferred from unit R1 to unit F. Figure 17c shows the feasible solution obtained by applying either of the two approaches proposed in this work. If a NIS policy is considered, one unfeasible sequence takes place at time 11.6 h (Figure 17d). Finally, three unfeasible sequences arise adopting a ZW policy. The first batches of products C and B are transferred simultaneously between units R1 and G at time 7.8 h, the second batches of the same products are transferred between units R2 and G at time 16.4 h, and the first batch of product A and the second batch of product C are simultaneously transferred from unit R2 to F at time 8.8 h (Figure 17f).

7. Conclusions

A wide variety of MILP-based optimization methods for short-term scheduling of batch plants has been developed in the last years. Although they have showed a good computational performance in a wide variety of scheduling problems, most of them have only focused the attention on the modeling aspects of processing and changeover tasks, ignoring the important role of material transfer operations. Although transfer times may represent a very small percentage of time regarding the whole duration of processing tasks in the batch units, loading and unloading operations may play a crucial role in the synchronization of material movement tasks. Most of the mathematical formulations available in the literature neglect their importance, lumping transfer times into the processing times or just assuming them as zero. These formulations usually focus on ensuring that

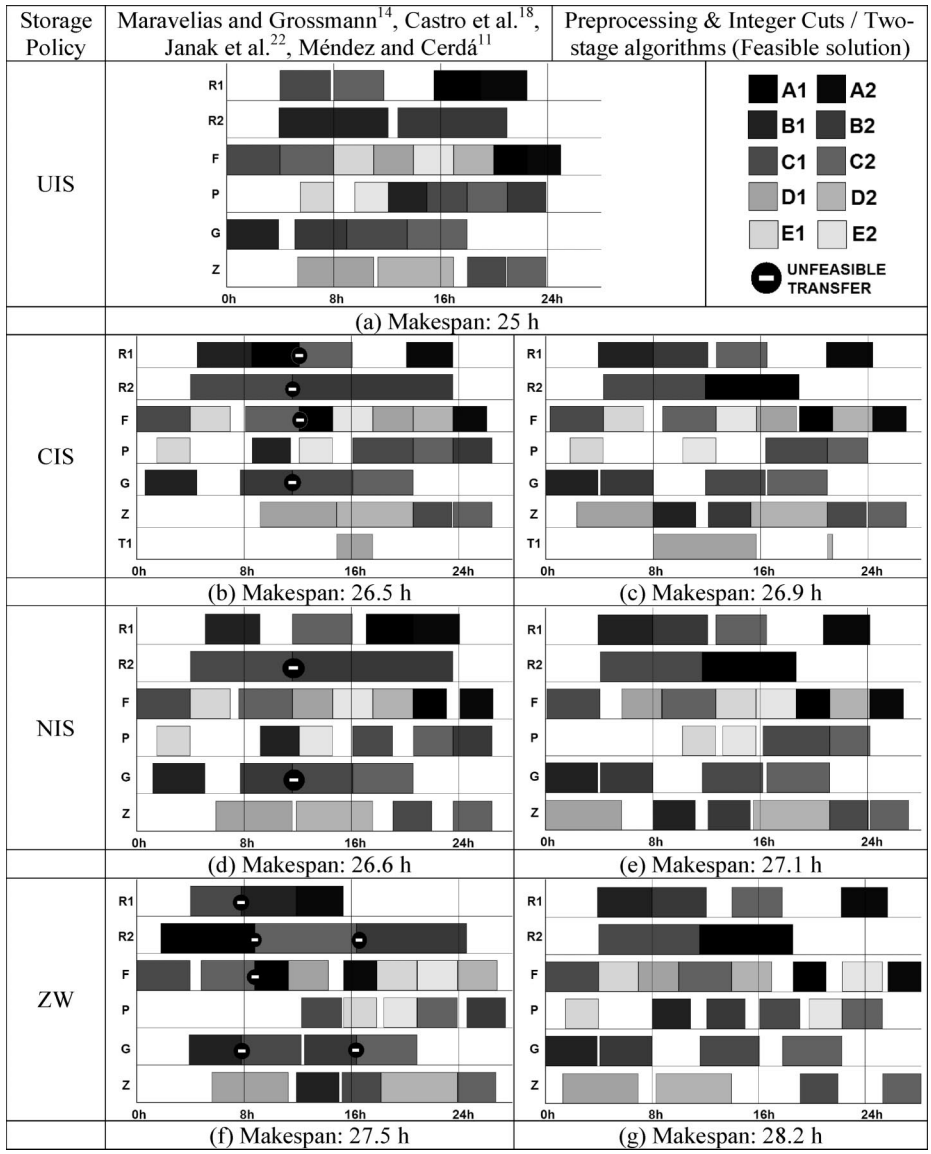

Figure 17. Optimal schedules for case study 3.

the material balances are feasible between consecutive stages. Therefore, by omitting the actual transfer times and their corresponding effect on the task synchronization, optimal but actually unfeasible solutions in practice may be reached, particularly in those cases involving shared units and storage tanks, material recycles, or bidirectional flows of products. To avoid this situation, a continuous time MILP framework based on the general precedence notion that can explicitly consider nonzero transfer times is introduced. Despite the direct representation of transfer activities, neither new variables nor additional constraints are required and consequently the computational effort remains almost the same. Also, two alternative solution approaches that avoid generating cyclic schedules are proposed for the general precedence formulation. The first approach consists of identifying those groups of sequenced tasks that can lead to unfeasible sequences, and then adding the corresponding integer-cuts to the mathematical model, so that the unfeasible sequences are avoided. A second approach consists of a two-stage algorithm, which first assigns a small value to the transfer times, in order to force a proper synchronization, and second solves the problem fixing the previous schedule, but assigning zero transfer-times, and assessing the new starting and finishing times of tasks. Finally, the appearance of unfeasible solutions has been proved along a series of case studies accounting for different intermediate storage policies. These unfeasible solution schedules have been compared to the feasible ones obtained by using either of the two solution approaches proposed in this work, highlighting the significant effect of transfer operation restrictions on scheduling decisions. Finally, it is worth remarking that further work is still needed to explicitly represent transfer operations in general network processes treated by STN or RTN-based formulations. It is also suggested that specific restrictions and resources related to transfer tasks should be systematically incorporated in any optimization scheduling approach.

Acknowledgment

Financial support received from the European Community projects (MRTN-CT-2004-512233; INCO-CT-2005-013359), the Departament d'Educació i Universitats de la Generalitat de Catalunya, the European Social Fund, the FPU program from Ministerio de Educación y Ciencia, from FONCYT-ANPCyT under Grant PICT 2006-01837 and CONICET is fully appreciated. We would also like to thank Prof. Ferenc Friedler from University of Pannonia for his fruitful discussions.

Appendix

Nomenclature

Subscripts

p, p' = product
i, i' = batch
s, s' = processing stage
p,i,s = batch operation (p,i,s)
u, u', u'' = processing unit
t, t' = storage tank

Sets

U_{ps} = set of available units for processing product p at stage s
T_u = set of storage tanks available after unit u

Parameters

$p_{t\text{-}psu}$ = processing time for stage s of product p in unit u
$t_{t\text{-}pu}$ = transfer time of product p from unit u
M = a very large number

Continuous Variables

$T_{f\text{-}pis}$ = completion time for the task (p,i,s)
$T_{s\text{-}pis}$ = starting time for task (p,i,s)
$T_{w\text{-}pis}$ = waiting time for task (p,i,s)
MK = makespan

Binary Variables

$X_{pis,p'i's'}$ = 1 if task (p,i,s) is processed before another task (p',i',s'), or 0 otherwise
Y_{pisu} = 1 if task (p,i,s) is allocated to equipment unit u, or 0 otherwise
AT_{pis} = 1 if material from task (p,i,s) is transferred to storage tank t, or 0 otherwise

Pretreatment Algorithm

PAIR = a pair of consecutive tasks
CPAR = set of PAIR that may give rise to an unfeasible transfer
SUNF = set that contains all the unfeasible CPAR
*n*pair = ordinal associated to a pair of consecutive tasks
lastpair = total number of pairs of consecutive tasks
*n*level = number of PAIR in CPAR
*n*pair*R* = *n*pair of the last PAIR in CPAR
*n*level*R* = *n*level of the last PAIR in CPAR

Literature Cited

(1) Baker, K. R. *Introduction to Sequencing and Scheduling*; Wiley and Sons: New York, 1974.

(2) French, S. *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*; Ellis Horwood Limited: England, 1982.

(3) Baudin, M. *Manufacturing Systems Analysis with Applications to Production Scheduling*; Yourdon Press: Englewood Cliffs, NJ, 1990.

(4) Kim, M. S.; Jung, J. H.; Lee, I. B. Optimal Scheduling of Multiproduct Batch Processes for Various Inter-Stage Storage Policies. *Ind. Eng. Chem. Res.* **1996**, *35*, 4058.

(5) Ha, J. H.; Chang, H. K.; Lee, E. S.; Lee, I. B.; Lee, B. S.; Yi, G. Inter-Stage Storage Tank Operation Strategies in the Production Scheduling of Multi-Product Batch Processes. *Comput. Chem. Eng.* **2000**, *24*, 1633.

(6) 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.

(7) Sundaramoorthy, A.; Karimi, I. A. A Simpler Better Slot-Based Continuous-Time Formulation for Short-Term Scheduling in Multipurpose Batch Plants. *Chem. Eng. Sci.* **2005**, *60*, 2679.

(8) 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.

(9) Heo, S. K.; Lee, K. H.; Lee, H. K.; Lee, I. B.; Park, J. H. A New Algorithm for Cyclic Scheduling and Design of Multipurpose Batch Plants. *Ind. Eng. Chem. Res.* **2003**, *42*, 836.

(10) Ferrer-Nadal, S.; Méndez, C. A.; Graells, M.; Puigjaner, L. Optimal Reactive Scheduling of Manufacturing Plants with Flexible Batch Recipes. *Ind. Eng. Chem. Res.* **2007**, *46*, 6273.

(11) Méndez, C. A.; Cerdá, J. An MILP continuous-Time Framework for Short-Term Scheduling of Multipurpose Batch Processes under Different Operation Strategies. *Optim. Eng.* **2003**, *4*, 7.

(12) Méndez, C. A.; Cerdá, J.; Harjunkoski, I.; Grossmann, I. E.; Fahl, M. State-of-the-Art Review of Optimization Methods for Short-Term Scheduling of Batch Processes. *Comput. Chem. Eng.* **2006**, *30*, 913.

(13) Giannelos, N. F.; Georgiadis, M. C. A Novel Event-Driven Formulation for Short-Term Scheduling of Multipurpose Continuous Processes. *Ind. Eng. Chem. Res.* **2002**, *41*, 2431.

(14) 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.

(15) Kondili, E.; Pantelides, C. C.; Sargent, R.; W, H. A General Algorithm for Short-Term Scheduling of Batch Operations. I. MILP Formulation. *Comput. Chem. Eng.* **1993**, *17*, 211.

(16) Pantelides, C. C. *Unified Frameworks for the Optimal Process Planning and Scheduling;* Second International Conference Foundations of Computer-Aided Process Operations; Cache Publications: New York, 1994; Vol. 253.

(17) Castro, P.; Barbosa-Povoa, 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.

(18) Castro, P.; Barbosa-Povoa, A. P. F. D.; 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.

(19) 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.

(20) Vin, J. P.; Ierapetritou, M. G. A New Approach for Efficient Rescheduling of Multiproduct Batch Plants. *Ind. Eng. Chem. Res.* **2000**, *39*, 4228.

(21) 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.

(22) 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.

(23) Cerdá, J.; Henning, 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.

(24) Sanmarti, E.; Holczinger, T.; Puigjaner, L.; Friedler, F. Combinatorial Framework for Effective Scheduling of Multipurpose Batch Plants. *AIChE J.* **2002**, *48*, 2557.

(25) Cormen, T. H.; Leiserson, C. E.; Rivest, R. L. *Introduction to Algorithms*; MIT Press: Cambridge, MA, 1997.

(26) Romero, J.; Puigjaner, L.; Holczinger, T.; Friedler, F. Scheduling Intermediate Storage Multipurpose Batch Plants Using the S-Graph. *AIChE J.* **2004**, *50*, 403.

(27) Ferrer-Nadal, S.; Holczinger, T.; Méndez, C. A.; Friedler, F.; Puigjaner, L. Rigorous Scheduling Resolution of Complex Multipurpose Batch Plants: S-Graph vs. MILP. In *16th European Symposium on Computer Aided Process Engineering*; Marquardt, W., Pantelides, C. Ed.; Elsevier: Amsterdam, The Netherlands, 2006, *21*, 2033.

(28) Kim, S. B.; Lee, H.; Lee, I.; Lee, E. S.; Lee, B. Scheduling of Non-Sequential Multipurpose Batch Process Under Finite Intermediate Storage Policy. *Comput. Chem. Eng.* **2000**, *24*, 1703.

(29) Papageorgaki, S.; Reklaitis, G. V. Optimal Design of Multipurpose Batch Plants. 2. A Decomposition Solution Strategy. *Ind. Eng. Chem. Res.* **1990**, *29*, 2062.

Received for review January 17, 2008
Revised manuscript received July 6, 2008
Accepted July 16, 2008

IE800075U