I&EC research
Industrial & Engineering Chemistry Research

Article

pubs.acs.org/IECR

Simultaneous Lot Sizing and Scheduling of Multistage Batch Processes Handling Multiple Orders per Product

Pablo A. Marchetti, Carlos A. Méndez, and Jaime Cerdá*

Instituto de Desarrollo Tecnológico para la Industria Química (INTEC), Universidad Nacional del Litoral−CONICET, Santa Fe, Argentina

ABSTRACT: A pair of precedence-based continuous-time formulations addressing the combined lot sizing and scheduling of order-driven multistage batch facilities is presented. The proposed mixed-integer linear programming (MILP) models can handle multiple orders per product with different delivery dates, variable processing times, and sequence-dependent changeovers. As each order may be filled by one or more batches, enough batches for each order ensuring optimality are initially defined. The two monolithic formulations are intended for sequential batch processes where batch integrity is preserved throughout the entire production system. However, lots of final products can be split to satisfy two or more orders. One of the approaches is based on a detailed MILP formulation allocating individual batches to units and ordering them in every unit. In contrast, the second methodology is specially designed for large scheduling problems. It first gathers batches for the same order into clusters, and then assigns clusters to units and sequences groups of batches in every unit. The larger the number of groups, the more rigorous is the cluster-based formulation. Alternative sequencing constraints based on reliable assumptions were also tested. Several examples involving up to 92 batches have been successfully solved using one or both formulations.

1. INTRODUCTION

Chemical batch processes are very attractive for producing low volumes of a wide variety of products in a highly flexible manner. In addition to the flexibility advantage, they require lower investment costs in facilities and equipment and reduced time for introducing new products to the market. Besides, they can readily accommodate the manufacture of seasonal items and variations in production volume, feedstock, and product specifications. Batch processes are very common in the production of foods, pharmaceuticals, cement, paints, adhesives, and specialty chemicals. Production is mostly driven by customer orders with specific delivery dates. Every week several dozens of batches are processed in industrial facilities. The major challenge of batch processes is to cope with their inherent inefficiencies. Once the processing of a batch is completed, the equipment must be stopped and reconfigured and its output should be tested before starting the production of the next batch. Moreover, setup times are important because of frequent changes of product in the production system. Maximizing the productivity of manufacturing resources and the customer satisfaction are then the major operational goals for order-driven batch facilities. Scheduling is the process of allocating resources to processing tasks over time in such a way that some performance measure is optimized. Batch scheduling methods usually help to reduce unproductive time and to deliver customer orders on time. At present, there is a lack of efficient computational tools for the scheduling of industrial-sized multistage batch processes.

In multiproduct batch plants, each batch follows a manufacturing route defined by the related product recipe that usually includes several processing stages to convert the input material into the desired final product. Many batch facilities additionally have a sequential structure with several nonidentical units working in parallel at each stage. After completing the batch processing in stage s, the whole batch is transferred to another unit in the next stage $(s + 1)$. Then, the identity of every batch is preserved throughout the whole processing system and material balance equations can be omitted. In other words, batch mixing and splitting as well as material recycles never arise. Such production systems are termed multiproduct, multistage sequential batch processes. Depending on whether the set of batches to be processed and their sizes are or are not problem data, batch scheduling methodologies can be broadly classified into two types: sequential and monolithic approaches. Sequential methodologies mostly assume that the number and size of batches are known beforehand; i.e., the lot-sizing problem has already been solved. Moreover, task processing times are fixed data. In contrast, monolithic approaches not only assign batches to equipment units at every stage and sequence the batches in each unit but also simultaneously solve the lot-sizing problem.

In this work, a pair of mixed-integer linear programming (MILP) formulations addressing the combined lot sizing and scheduling of sequential, multistage batch processes are presented. One of them has been specially designed for solving very large scheduling problems. Both approaches can schedule multiple customer orders for the same product due at different delivery dates, with each order requiring the production of several batches. Moreover, they allow splitting a batch of final product to satisfy more than one customer order.

Literature Review. Thorough reviews on batch scheduling methods can be found in Méndez et al.[1] and Floudas and Lin.[2] Most approaches can be regarded as rigorous solution methods

Received: October 4, 2011
Revised: March 19, 2012
Accepted: March 29, 2012
Published: March 29, 2012

relying on mathematical programming models. According to the process topology and the use of single or multifunctional equipment, batch processes can be classified into two major categories: network type and sequential multistage batch facilities. In network-type processes, product recipes are rather complex and include splitting and mixing operations and recycle streams. In addition, multifunctional units can carry out different kinds of processing tasks. Discrete and continuous-time monolithic approaches for the scheduling of network-type batch processes based on state–task–network (STN) and resource–task–network (RTN) representations have been proposed.[3−8] Though quite general and applicable to any type of batch scheduling problem, their computational efficiency for sequential batch processes cannot compete with those of methodologies explicitly exploiting the series structure. Most of the approaches specially designed for sequential batch processes just perform the scheduling of a known set of batches with fixed sizes. Sequential methods are classified into three categories: slot-based,[9] unit-specific time grids,[10] and precedence-based formulations in terms of global[11,12] or immediate precedence variables.[13,14] Using the so-called constant batch ordering rule (CBOR), Marchetti and Cerdá[15] developed an approximate MILP formulation for larger scheduling problems. The CBOR rule states that any pair of batches allocated to the same equipment item at different stages present the same relative ordering on the batch queues of the shared units.

Only in recent years has attention been paid to the development of monolithic methods for sequential batch processes. Lim and Karimi[16] developed an MILP model to decide simultaneously the product batches to be processed and their schedule in single-stage batch plants equipped with nonidentical parallel units. It is a slot-based formulation that explicitly considers multiple orders per product with different due dates, batch size-dependent processing times, and finite changeover times between consecutive batches processed in the same unit. The model assumes that a single batch may fill multiple orders of a product, but the allocation of batches to orders with earlier due dates is favored. Moreover, an order may be filled by one or more batches that may be produced in different units. The plant objective is to maximize some measure of customer satisfaction (e.g., tardiness) or plant performance (e.g., makespan). Prasad and Maravelias[17] first introduced an MILP precedence-based formulation for the simultaneous batching and scheduling of sequential, multistage batch processes. The proposed model assumes that every product goes through all the stages following the same route, the intermediate storage capacity between consecutive stages is unlimited, and enough amounts of resources other than equipment units are available to prevent them from becoming production bottlenecks. Several customer orders with different due dates are to be satisfied, but each one involves a different product. Multiple orders for the same product are not considered. By modeling the problem through a global precedence-based formulation, sequence-dependent changeover times are readily handled. To enhance the computational efficiency of the approach, symmetry-breaking constraints based on batch-selection variables have also been considered. In addition, the model size is reduced through using time window information for fixing some sequencing variables, identifying forbidden paths, and developing a class of valid knapsack inequalities to exclude subsets of infeasible assignments. Sundaramoorthy and Maravelias[18] extended the formulation of Prasad and Maravelias[17] to account for variable processing times. In this new model, processing times are functions of both batch selection and sizing decisions. Marchetti et al.[19] presented a pair of MILP continuous-time formulations for the simultaneous lot sizing and scheduling of single-stage multiproduct batch facilities. Both global precedence-based models account for variable processing times and consider several customer orders per product with different delivery dates. Moreover, several batches may be necessary to meet a single order and each lot of final product can be used to satisfy more than one order. One of the MILP models rigorously arranges individual batches in each unit, while the other sequences clusters of batches sharing the same product and due date and processed in the same equipment item. Grouping batches into clusters aims to reduce the number of product changeovers and at the same time significantly decrease the number of sequencing variables. Contents of clusters are model decisions. Powerful symmetry-breaking constraints based on allocation variables were also developed.

On the other hand, Sundaramoorthy and Maravelias[20] further generalized the precedence-based formulation for the batching and scheduling of multistage batch processes to also consider storage constraints. Storage vessels are modeled as additional processing units for which assignment and sequencing constraints are also written. However, the identity of every lot is preserved by storing just one batch at a time, and therefore batches are never mixed. A general classification of storage policies in multistage batch processes based upon possible combinations of storage capacity and timing constraints is presented. The proposed MILP representation can easily accommodate all the combinations of such constraints. Later, Sundaramoorthy et al.[21] presented a discrete-time framework for the batching and scheduling of multistage batch processes that accounts for storage and utility constraints. They consider several types of utilities such as cooling water, steam, and electricity. In contrast to previous formulations, the approach handles batching decisions without the usage of explicit batch selection variables. A common time grid is defined by dividing the time horizon into a number of fixed-length periods using a set of time points. In this way, utility constraints can be accurately represented and the sequencing of batches can be enforced on each unit/vessel. Tasks can start and finish at some time points of the grid, and changes in the status of units and in both inventory and utility consumption levels can only occur at those times. As before, the identity of batches is still preserved and the formulation is able to handle different types of storage policies.

This paper presents two new monolithic approaches addressing the combined lot sizing and scheduling of multistage sequential batch facilities. They generalize the global precedence-based continuous-time formulations introduced by Marchetti et al.[19] for single-stage batch plants. A predefined set of generic batches for each customer order containing enough elements to guarantee optimality is initially proposed. Again, the two formulations handle batch allocation and sequencing decisions in a different manner. The first one arranges individual batches processed in the same equipment item, while the second defines clusters of batches allocated to the same order and following the same route and sequences them in every unit. Contents of clusters in the best schedule are model decisions. The cluster-based approach has been proposed to tackle very large multistage batch scheduling problems in a very efficient manner.

2. PROBLEM STATEMENT AND MAIN ASSUMPTIONS

The unified lot-sizing and scheduling problem for multistage batch facilities processing multiple production orders per product can be stated as follows. Given

(i) a multiproduct batch plant with several processing stages $s \in S$, each one equipped with nonidentical parallel units $j \in J_s$
(ii) the set of products $i \in I$ to be manufactured
(iii) the set of customer orders for each product i and their delivery dates $d \in D_i$
(iv) the size r_{id} of every order (i, d) involving product $i \in I$ and due at time $d \in D_i$
(v) the set of units $J_{is} \subseteq J_s$ available at stage $s \in S$ for processing product $i \in I$
(vi) the minimum ($q_{ij}^{\min}$) and maximum ($q_{ij}^{\max}$) batch sizes for product i at every unit $j \in J_{is}$ of any stage $s \in S$
(vii) the parameters (ft_{ij}, vt_{ij}) for the expression used to evaluate the processing time of a batch of product i in unit $j \in J_{is}$ as the sum of a fixed-term and a size-dependent time contribution
(viii) sequence-dependent changeover times $\tau_{ii'j}$ for each ordered pair of products (ii') that can be processed in unit $j \in J_{is}$
(ix) the length of the time horizon H

The problem goal is to determine (a) the number and size of batches to be processed for fulfilling every customer order (i, d), (b) the allocation of units to batches at every stage, (c) the batch processing sequence at each equipment unit, and (d) the starting and completion times of batch processing tasks, such that production orders and operational constraints are all satisfied at minimum total tardiness or makespan.

To model the problem, the following assumptions have been made:

(1) Model parameters are all deterministic.
(2) A nonpreemptive operation mode is used.
(3) All products require the same sequence of processing stages.
(4) Batch mixing and splitting operations are not allowed at intermediate stages; i.e., batch integrity is preserved throughout the entire processing structure.
(5) Production orders all feature a single product. In other words, customer orders involving multiple products and a common due date are decomposed into single-product orders with the same promised date.
(6) Production orders for the same product are fulfilled in chronological order; i.e., the one featuring the earlier due date is first completed.
(7) A single batch can be used to satisfy multiple orders of the same product.
(8) There is an unlimited intermediate storage between consecutive processing stages (UIS mode), and enough capacity of resources aside from equipment (i.e., raw materials, manpower, or utilities) is always available at every stage.
(9) Changeover time between batches containing the same product is negligible.

By assumption 7, batches can be split when the final product is ready, but not at intermediate stages (see assumption 4). Assumptions 3 and 4 are specially suited for sequential batch processes and usually lead to a simpler problem formulation and a lower number of alternative solutions. Because of assumption 4, the entire content of a batch leaving a processing stage is always transferred to the next one, and material balance constraints are not needed. Moreover, assumption 6 allows postulating beforehand a predefined set of batches for each production order (i, d), thus avoiding the need of binary decisions to allocate batches to orders.

3. DETAILED BATCHING AND SCHEDULING MODEL

A detailed mixed-integer linear programming (MILP) formulation for batching and scheduling of multistage sequential batch facilities is first presented. The proposed model is based on a continuous-time general-precedence representation, where decision variables (either continuous or binary) are explicitly related to individual batches. In order to decide how many lots for each order should be manufactured, an initial set of generic batches B_{id} for each production order (i, d) is to be proposed. The cardinality of every set B_{id} ($\forall\, d \in D_i$, $i \in I$) should be large enough to account for all feasible schedules. Figure 1 presents a schematic representation of the hierarchical decisions simultaneously performed by the proposed detailed MILP model.

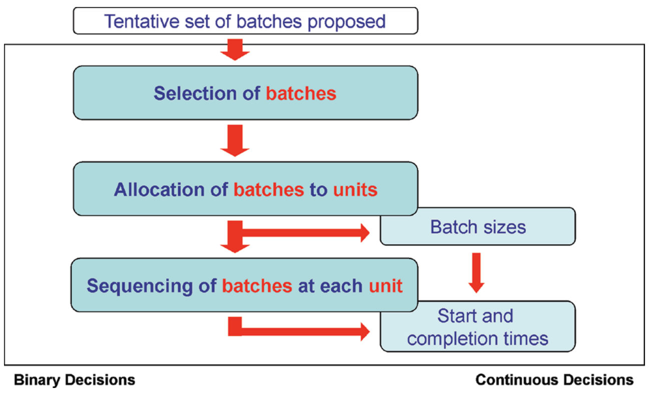

Tentative set of batches proposed
Selection of batches
Allocation of batches to units
Batch sizes
Sequencing of batches at each unit
Start and completion times
Binary Decisions
Continuous Decisions

Figure 1. Hierarchical decisions simultaneously taken by the detailed formulation.

They are as follows: (a) the selection of batches $b \in B_{id}$ ($\forall\, d \in D_i$, $i \in I$) to be processed, (b) the allocation of processing units to selected batches, and (c) the relative ordering of any pair of batches (b, b') at each equipment item of every stage $s \in S$. Such discrete decisions are handled through the binary variables W_b, Y_{bj}, and $X_{b,b',s}$, respectively. Besides, continuous decision variables are defined to properly choose the batch size BS_b and the processing starting/completion times for any selected batch b at every stage s (ST_{bs}, CT_{bs}). Although discrete decisions are typically handled through binary variables, in some occasions this is not necessary. Sometimes continuous variables in the range [0, 1] can be used because some model constraints force them to take integer values.

In the next sections, the methodology used to define proper sets of batches B_{id} ($d \in D_i$, $i \in I$) ensuring the discovery of the optimal schedule, the model constraints, and the objective functions of the proposed MILP formulation are all presented.

3.1. Proposed Number of Batches for Each Product. A simple procedure to find a conservative estimation of the number of batches required to meet all customer orders for every product i is first presented. It aims to improve the methods of Marchetti et al.[19] by generating a more convenient number of batches. Equation 1 gives the minimum ($q_i^{\min}$) and maximum ($q_i^{\max}$) batch sizes for product i in a multistage processing structure with several units at every stage. In turn, eq

2 provides a rigorous upper bound on the integer number of batches of product i (nb_i) needed to satisfy all production orders.

$$q_i^{\min} = \max_{s\in S}[\min_{j\in J_{is}}(q_{ij}^{\min})]; \qquad q_i^{\max} = \min_{s\in S}[\max_{j\in J_{is}}(q_{ij}^{\max})] \quad \forall\, i \in I \tag{1}$$

$$\mathrm{nb}_i = \left\lceil \frac{\sum_{d\in D_i} r_{id}}{q_i^{\min}} \right\rceil \qquad \forall\, i \in I \tag{2}$$

3.2. Proposed Number of Batches for Each Production Order. In the proposed model, batches of product i are associated beforehand with production orders (i, d) and, consequently, binary variables linking batches to orders are not necessary. Certainly, a number of batches $\mathrm{nb}_{id} = |B_{id}|$ large enough to meet every production order (i, d) has to be defined. Although eq 2 gives a conservative value for $|B_i|$, a few additional batches will be needed to generate an adequate partition of the set B_i such that

$$B_i = \cup_{d\in D_i} B_{id} \tag{3}$$

Every generic batch $b \in B_{id}$ will have a specific due date $d \in D_i$. If b is selected by the model, it will be processed and assigned to order (i, d). Because of assumption 6 (section 2), production orders must be fulfilled in chronological order. Then, order (i, d) should be satisfied before order (i, d') whenever $d' > d$. If the size of batch b is large enough ($\mathrm{BS}_b > r_{id}$), it can be used to fulfill not only order (i, d) but also some other orders (i, d') with $d' > d$. Then, a selected batch may satisfy more than one order and, consequently, it cannot be known in advance which batches will be allocated to order (i, d) at the best solution. This fact further complicates the estimation of nb_{id}. Nonetheless, a reference batch size is needed for computing nb_{id}. By adopting a batch reference size bs_i larger than $q_i^{\min}$, a lower number of batches for each order (i, d) will be required and the model size consequently decreases. However, the optimality of the solution found is no longer guaranteed. Alternative choices for the reference size bs_i are given by eqs 4, 5a, or 5b.

$$\mathrm{bs}_i = q_i^{\min} \qquad \forall\, i \in I \tag{4}$$

$$\mathrm{bs}_i = \max_{s\in S}\left[\frac{1}{|J_{is}|}\sum_{j\in J_{is}} q_{ij}^{\min}\right] \qquad \forall\, i \in I \tag{5a}$$

$$\mathrm{bs}_i = \max_{s\in S}\left[\frac{1}{|J_{is}|}\sum_{j\in J_{is}}\left(\frac{q_{ij}^{\min} + q_{ij}^{\max}}{2}\right)\right] \qquad \forall\, i \in I \tag{5b}$$

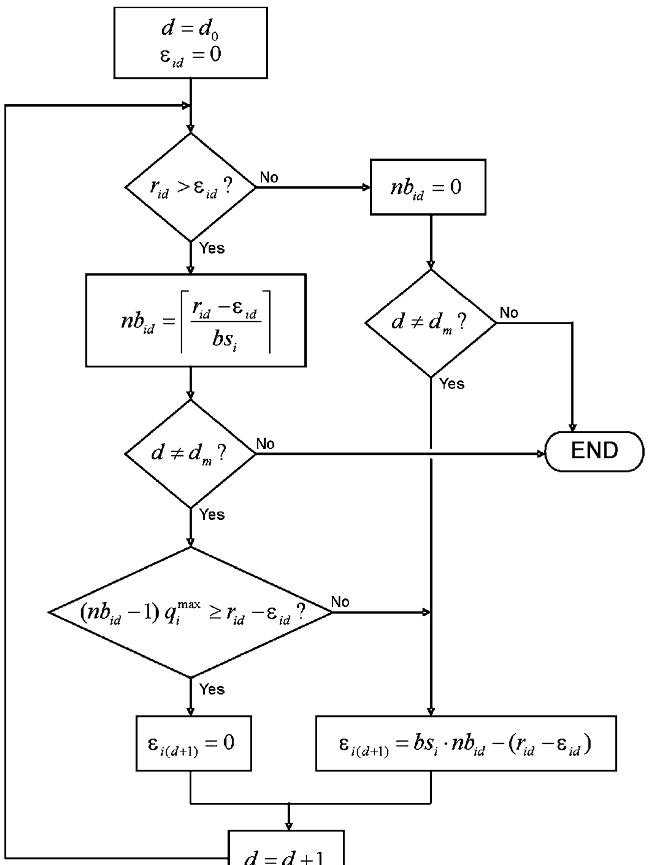
The procedure used to define $|B_{id}|$ is now presented. Let $D_i = \{d_0, ..., d_m\}$ be the set of delivery dates for the production orders of product i, such that $d_k < d_{k+1}$ for $k = 0, 1, 2, ..., m - 1$. In other words, the elements of D_i are arranged by increasing due dates. The integer parameter nb_{id} defined by eq 6 represents the cardinality of B_{id}, i.e., the number of generic batches associated beforehand with order (i, d). In turn, the parameter ε_{id} is the inventory of product i that is available for order (i, d) from a batch $b \in B_{id'}$ with an earlier due date $d' < d$. For the first due date d_0 of product i, there is no inventory available and eq 7 sets the value of ε_{id_0} at zero. The available inventory for each successive due date $d > d_0$ of product i is given by eq 8.

$$\mathrm{nb}_{id} = \max\left[0, \left\lceil \frac{r_{id} - \varepsilon_{id}}{\mathrm{bs}_i} \right\rceil\right] \qquad \forall\, i \in I,\, d \in D_i \tag{6}$$

$$\varepsilon_{i,d_0} = 0 \qquad \forall\, i \in I \tag{7}$$

$$\varepsilon_{i,d+1} = \max\{0, (\varepsilon_{id} + \mathrm{bs}_i \cdot \mathrm{nb}_{id} - r_{id})\} \tag{8}$$

Figure 2 shows a flow diagram describing the procedure for the calculation of ε_{id} and nb_{id}. Due dates for each product i are



$d = d_0$
$\varepsilon_{id} = 0$
$r_{id} > \varepsilon_{id}$?
No
$nb_{id} = 0$
Yes
$nb_{id} = \left\lceil \frac{r_{id} - \varepsilon_{id}}{bs_i} \right\rceil$
$d \neq d_m$?
No
Yes
END
$d \neq d_m$?
No
Yes
$(nb_{id} - 1)\, q_i^{\max} \geq r_{id} - \varepsilon_{id}$?
No
Yes
$\varepsilon_{i(d+1)} = 0$
$\varepsilon_{i(d+1)} = bs_i \cdot nb_{id} - (r_{id} - \varepsilon_{id})$
$d = d + 1$

Figure 2. Procedure for estimating the number of batches allocated to each order.

considered in consecutive order, starting from $d = d_0$ and an available inventory $\varepsilon_{i,d_0} = 0$, $\forall\, i$. If $\varepsilon_{id} \geq r_{id}$ for some $d \geq d_0$, then $\mathrm{nb}_{id} = 0$ and B_{id} is an empty set. If so, an excess portion of batch $b \in B_{id'}$ with $d' < d$ is enough to meet order (i, d). Regarding this case, some precautions are taken by the estimation procedure of nb_{id} to protect against nonoptimality or problem infeasibility due to an insufficient number of batches for some production order. Such corrective steps arise at the bottom part of Figure 2.

It frequently happens that the size of some batches $b \in B_{id}$ at the best solution is greater than the reference size bs_i. Under these conditions, it is possible that the number of batches selected by the model to meet the demand $r_{i,d}$ at the best solution will be lower than the value of $\mathrm{nb}_{i,d}$ obtained using the reference size together with eqs 6–8; for instance, $(\mathrm{nb}_{i,d} - 1)$ or less. Moreover, it may occur that no inventory of product i

will remain from batches $b \in B_{id}$ to meet the order $(i, d+1)$ by a proper choice of the batch sizes, i.e., $\varepsilon_{i,d+1} = 0$. In such a case, the use of eqs 6–8 may lead to defining $B_{i,d+1}$ with at least one batch less than needed. As a result, a nonoptimal solution may be found or even the problem could become infeasible.

To prevent the occurrence of this kind of situation, the proposed procedure given in Figure 2 sets $\varepsilon_{i,d+1} = 0$ when the condition $(\mathrm{nb}_{id} - 1)q_i^{\max} \geq r_{id} - \varepsilon_{id}$ holds. Such a condition means that $(r_{id} - \varepsilon_{id})$ units of product i are satisfied by processing a number of $(\mathrm{nb}_{id} - 1)$ batches or less featuring the maximum size $q_i^{\max}$. Under these circumstances, an appropriate selection of batch sizes can produce exactly $(r_{id} - \varepsilon_{id})$ units of product i, and therefore $\varepsilon_{i(d+1)} = 0$. To anticipate this situation, the procedure described in Figure 2 directly sets $\varepsilon_{i(d+1)} = 0$ to guarantee that $B_{i,d+1}$ includes a sufficient number of batches for order $(i, d+1)$. In other words, eq 8 is replaced by eq 8′.

$$\varepsilon_{i,d+1} = \begin{cases} 0, & \text{if } (\mathrm{nb}_{id} - 1)q_i^{\max} \\ & \geq r_{id} - \varepsilon_{id} \\ & \geq 0 \\ \max\{0, (\varepsilon_{id} - r_{id} & \text{otherwise} \\ \quad + \mathrm{bs}_i \cdot \mathrm{nb}_{id})\}, & \end{cases} \tag{8′}$$

Though it may lead to a higher nb_i, this modification ensures the availability of enough batches for every order (i, d). Once the value of nb_{id} for each order (i, d) is determined, the proposed number of batches for product i is given by

$$\mathrm{nb}_i = \sum_{d \in D_i} \mathrm{nb}_{id} \qquad \forall\, i \in I \tag{9}$$

The total number of batches (nb_i) given by eq 9 is expected to slightly overestimate the one really needed at the optimal schedule. If bs_i is defined by eq 4, the set B_i will include at most $|D_i| - 1$ more batches if eqs 6, 7, 8′, and 9 instead of eq 2 are used to compute nb_i.

3.3. Model Constraints.

The detailed mathematical formulation to be presented is closely related to the sequence-based batch scheduling models of Prasad and Maravelias[17] and Sundaramoorthy and Maravelias.[18] However, some differences on modeling both the batch selection and the symmetry-breaking constraints arise. The proposed MILP model includes the selection variable W_b denoting the existence of batch b, but unlike previous work, it is a continuous variable. Besides, another main difference can be found on the symmetry-breaking constraints used to avoid redundant solutions. By formulating them in terms of batch selection and allocation decision variables, such constraints are stronger than those used in previous methods based on batch sizes. Moreover, the proposed approach considers several alternatives to reduce the number of sequencing variables by restraining the ordering of lots in the batch queue of any equipment unit.

3.3.1. Allocation Constraints.

The batch existence variable $W_b \in [0, 1]$ is a continuous variable determining if batch $b \in B_{id}$ will be processed. If $W_b = 1$, batch b does exist and must be assigned to an equipment unit at every stage as prescribed by eq 10. Binary allocation variables Y_{bj} for batches containing product i are defined for each unit $j \in J_{is}$ of every stage $s \in S$ where product i can be processed. They all are driven to zero when $W_b = 0$.

$$\sum_{j \in J_{is}} Y_{bj} = W_b \qquad \forall\, b \in B_{id},\, d \in D_i,\, i \in I,\, s \in S \tag{10}$$

3.3.2. Symmetry-Breaking Constraints.

Symmetry-breaking constraints based on batch allocation decisions have been defined to avoid equivalent solutions. As already pointed out by Marchetti et al.,[19] different approaches can be used to reduce the search space by implicitly eliminating duplicate solutions. One of them consists of restraining the sizes of consecutive batches.[17,18] However, such constraints only affect the values of continuous variables, and therefore have a limited impact on the model computational performance. In contrast, the symmetry-breaking constraints presented by Marchetti et al.[19] for single-stage multiproduct facilities are based on the idea of restraining the allocation of consecutive batches in the set B_{id} to a particular unit $j \in J_i$. To this end, it exploits the fact that batches $b \in B_{id}$ are generic batches with no predefined sizes that may or may not be processed. The basic idea is as follows. If batch $b \in B_{id}$ is allocated to unit j $(Y_{bj} = 1)$, then the preceding batch $(b - 1)$ in the set B_{id}, if any, must be assigned to the same unit j or to a unit $j' < j$, with $j, j' \in J_i$. In other words, consecutive batches of the set B_{id} that were selected must be allocated either to the same unit or to consecutive units.

To apply this approach to a multistage scenario, the symmetry-breaking constraint (11) given in terms of allocation variables should be applied to only one stage $s^* \in S$ arbitrarily chosen. It is suggested to choose as stage s^* the one with the lowest capacity, i.e., the bottleneck stage. Equation 11 cannot be applied to every stage $s \in S$ because it will impose some further restrictions on the processing routes of consecutive batches, thus cutting some portion of the problem feasible space.

$$Y_{bj} \leq \sum_{\substack{j' \in J_{is^*} \\ j' \leq j}} Y_{(b-1)j'}$$
$$\forall\ (b-1),\, b \in B_{id},\, j \in J_{is^*},\, d \in D_i,\, i \in I \tag{11}$$

3.3.3. Batch Size Constraints.

Equation 12 defines the value of the continuous variable BS_b representing the size of batch $b \in B_{id}$ (in terms of the related final product) at all stages. It involves two components. The first one, given by $q_{ij}^{\min} Y_{bj}$, denotes the constant part of the batch size for any selected lot of product i allotted to unit j. The second component, given by the variable $Q_{bj} \in [0, \Delta_{ij}]$, represents the variable portion of the batch size when b is allocated to unit j. An upper bound on the value of Q_{bj} is given by $\Delta_{ij} = (q_{ij}^{\max} - q_{ij}^{\min})$, i.e., the difference between the maximum and minimum batch sizes for product i in unit j. If batch b is not selected, then $Y_{bj} = Q_{bj} = 0$, $\forall\, j \in J_i$, and $\mathrm{BS}_b = 0$, according to eqs 12 and 13. Otherwise, the value of Q_{bj} cannot exceed Δ_{ij} and the size BS_b will belong to the range $(q_{ij}^{\min}, q_{ij}^{\max})$.

$$\mathrm{BS}_b = \sum_{j \in J_{is}} (q_{ij}^{\min} Y_{bj} + Q_{bj})$$
$$\forall\, b \in B_{id},\, d \in D_i,\, i \in I,\, s \in S \tag{12}$$

$$Q_{bj} \leq \Delta_{ij} Y_{bj} \qquad \forall\, b \in B_{id},\, d \in D_i,\, i \in I,\, j \in J_{is},\, s \in S \tag{13}$$

3.3.4. Production Demands.

Equation 14 guarantees that all production orders (i, d) are accomplished by stating that the accumulated requirement of product i up to due date $d \in D_i$ must be fulfilled using batches associated with product i with due dates $d' \leq d$.

$$\sum_{\substack{d' \in D_i \\ d' \leq d}} \sum_{b \in B_{id'}} \mathrm{BS}_b \geq \sum_{\substack{d' \in D_i \\ d' \leq d}} r_{id'} \qquad \forall\, d \in D_i,\, i \in I \tag{14}$$

3.3.5. Task Processing Times. Equation 15 gives the relationship between the starting and completion times of batch $b \in B_{id}$ at each stage s. The processing time PT_{bj} for batch $b \in B_{id}$ in unit $j \in J_{is}$, defined by eq 16, takes a nonzero value only if batch b does exist and has been allocated to unit j in stage s, i.e., $Y_{bj} = 1$. Otherwise, $Y_{bj} = 0$ and $\mathrm{PT}_{bj} = 0$. Similar to eq 12 for BS_b, the processing time PT_{bj} is the sum of two components: a fixed component and a size-dependent component. When $q_{ij}^{\min} = q_{ij}^{\max}$ and $Y_{bj} = 1$, then $\Delta_{ij} = Q_{bj} = 0$ and the processing time PT_{bj} becomes equal to $(\mathrm{ft}_{ij} + \mathrm{vt}_{ij} q_{ij}^{\min})$.

$$\mathrm{CT}_{bs} = \mathrm{ST}_{bs} + \sum_{j \in J_{is}} \mathrm{PT}_{bj}$$
$$\forall\, b \in B_{id},\, d \in D_i,\, i \in I,\, s \in S \tag{15}$$

$$\mathrm{PT}_{bj} = \mathrm{ft}_{ij} Y_{bj} + \mathrm{vt}_{ij}(q_{ij}^{\min} Y_{bj} + Q_{bj})$$
$$\forall\, b \in B_{id},\, d \in D_i,\, j \in J_{is},\, i \in I,\, s \in S \tag{16}$$

3.3.6. Technological Constraints. Technological constraints (17) define the relationship between the completion and starting times for every pair of consecutive stages of the same batch. Stage s^{l} stands for the last processing stage.

$$\mathrm{CT}_{bs} \leq \mathrm{ST}_{b(s+1)}$$
$$\forall\, b \in B_{id},\, d \in D_i,\, i \in I,\, s \in S - \{s^{l}\} \tag{17}$$

3.3.7. Topological Constraints. An equipment unit running at stage s may not be physically connected to every unit of the next stage $s + 1$. If batch $b \in B_{id}$ is assigned to unit j in stage s, the topological constraint (18) ensures that batch b can only be allocated to an available unit $j' \in J_{i,s+1}$ that is physically connected to $j \in J_{i,s}$. The set $J_{s+1}^{(j)}$ stands for the set of units in stage $s + 1$ connected to unit $j \in J_{i,s}$.

$$Y_{bj} \leq \sum_{j' \in (J_{i,s+1} \cap J_{s+1}^{(j)})} Y_{bj'}$$
$$\forall\, b \in B_{id},\, d \in D_i,\, j \in J_{is},\, i \in I,\, s \in S - \{s^{l}\} \tag{18}$$

3.3.8. Sequencing Constraints. Three alternative sets of sequencing constraints, called S1, S2, and S3, are proposed to handle the ordering of every pair of batches allocated to the same equipment unit at any stage. In any case, the sequencing constraints are based on the general precedence concept introduced by Méndez et al.[11] The first sequencing constraint set S1 given by eqs 19, 20a, and 20b ensures the optimality of the solution found. Note that a different treatment is followed for stage s^* with regard to the other stages $s \neq s^*$. Stage s^* is the one already discussed in section 3.3.2 to which eq 11 applies. Generic batches $b \in B_{id}$ for the same production order (i, d) are preordered just at stage s^*. Equation 19 states that batch b should be processed before batch b' at stage s^* if batches $(b, b') \in B_{id}$ $(b < b')$ are associated with order (i, d) and share the same unit at stage s^*. In this way, the number of redundant solutions is decreased. Every other pair of batches not considered by eq 19 is sequenced using constraints 20a and 20b. Such constraints are written for every stage $s \in S$ if $i \neq i'$ or $i = i'$ and $d \neq d'$, and for any stage $s \neq s^*$ if $i = i'$ and $d = d'$. By applying the global precedence concept, they include a single sequencing variable $X_{b,b',s}$ for each pair of batches $b \in B_{id}$ and $b' \in B_{i'd'}$ $(b < b')$ potentially sharing an equipment unit. Three cases should be considered to explain the criterion used to establish which one of the two batches (b, b') plays the role of batch b. If both batches are associated with the same order (i, d), then batch b is the one first arising in the set B_{id}. If both are associated with different production orders (i, d) and (i, d') with $d < d'$ but contain the same product i, batch b is the one featuring the earlier due date d. If both contain different products i and i' with $i < i'$ in the set I, batch b is the one containing product i. If $X_{b,b',s} = 1$, then eq 20a is enforced and batch b is to be completed before starting batch b' at stage s. Otherwise, $X_{b,b',s} = 0$ and eq 20b indicates that batch b' should be processed earlier. Equations 19, 20a, and 20b define the rigorous batch sequencing scheme S1. In such equations, τ_{iij} is assumed to be 0.

$$\mathrm{CT}_{bs^*} \leq \mathrm{ST}_{b's^*} + H(2 - Y_{bj} - Y_{b'j})$$
$$\forall\, b, b' \in B_{id}\ (b < b'),\, d \in D_i,\, j \in J_{is^*},\, i \in I \tag{19}$$

$$\mathrm{CT}_{bs} + \tau_{ii'j} \leq \mathrm{ST}_{b's} + H(1 - X_{b,b',s}) + H(2 - Y_{bj} - Y_{b'j})$$
$$\forall\, b \in B_{id},\, b' \in B_{i'd'}\ (b < b'),\, d \in D_i,$$
$$d' \in D_{i'},\, j \in J_{i,i',s},\, (i, i') \in I,$$
$$s \in S\ (\text{if } i < i' \text{ or } (i = i' \wedge d < d')) \text{ or}$$
$$s \in S - \{s^*\}\ (\text{if } (i = i' \wedge d = d')) \tag{20a}$$

$$\mathrm{CT}_{b's} + \tau_{i'ij} \leq \mathrm{ST}_{bs} + H X_{b,b',s} + H(2 - Y_{bj} - Y_{b'j})$$
$$\forall\, b \in B_{id},\, b' \in B_{i'd'}\ (b < b'),\, d \in D_i,$$
$$d' \in D_{i'},\, j \in J_{i,i',s},\, (i, i') \in I,$$
$$s \in S\ (\text{if } i < i' \text{ or } (i = i' \wedge d < d')) \text{ or}$$
$$s \in S - \{s^*\}\ (\text{if } (i = i' \wedge d = d')) \tag{20b}$$

Two less conservative sets of sequencing constraints, called S2 and S3, are also proposed. The sequencing scheme S2 is more in line with assumption 6 (section 2) because it supposes that a pair of batches containing the same product and allocated to the same unit is always queued according to their relative order in the set $B_i = \cup_{d \in D_i} B_{id}$ at any stage s. If both lots (b, b') with $b < b'$ are associated with the same order (i, d), then the batch b arising first in the set B_{id} is processed before. On the other hand, if the two batches are related to different orders (i, d) and (i, d') with $d < d'$, the one featuring the earlier due date d is first processed. Sequencing constraints for batches involving the same product are given by eq 21. Moreover, a single binary variable $X_{b,b',s}$ (with $b < b'$) for each pair of batches containing different products is to be defined and the sequencing constraints for ordering them on the queue of any unit are given by eqs 22a and 22b. No special consideration is given to stage s^* when the set of sequencing constraints S2 is used.

$$\mathrm{CT}_{bs} \leq \mathrm{ST}_{b's} + H(2 - Y_{bj} - Y_{b'j})$$
$$\forall\, b, b' \in B_i,\, j \in J_{is},\, i \in I,\, s \in S\colon (b < b') \tag{21}$$

$$CT_{bs} + \tau_{ii'j} \le ST_{b's} + H(1 - X_{b,b',s}) + H(2 - Y_{bj} - Y_{b'j}) \quad \forall\ b \in B_i,\ b' \in B_{i'}\ (b < b'),\ j \in J_{i,i',s},\ (i, i') \in I\ (i < i'),\ s \in S \tag{22a}$$

$$CT_{b's} + \tau_{i'ij} \le ST_{bs} + HX_{b,b',s} + H(2 - Y_{bj} - Y_{b'j}) \quad \forall\ b \in B_i,\ b' \in B_{i'}\ (b < b'),\ j \in J_{i,i',s},\ (i, i') \in I\ (i < i'),\ s \in S \tag{22b}$$

The remaining sequencing scheme S3 introduces an additional assumption to reduce the size of the solution space and the problem complexity. The sequencing constraint S3 relies on the constant batch ordering rule introduced by Marchetti and Cerdá.[15] It assumes that any pair of batches $b \in B_i$ and $b' \in B_{i'}$ containing different products ($i \ne i'$) presents the same relative ordering at all stages where both share the same equipment unit. In the sequencing constraints 23a and 23b, just a single sequencing binary variable $\hat{X}_{b,b'}$ is defined for each pair of batches $b \in B_i$ and $b' \in B_{i'}$ (with $i \ne i'$) potentially sharing a processing unit at some stage s. Then, the variables $X_{b,b',s}$ in eqs 23a and 23b are replaced by a unique global sequencing variable $\hat{X}_{b,b'}$ that controls the relative ordering of batches (b, b') throughout the whole processing structure. Assuming that $b < b'$, then batch b precedes b' if both were allocated to the same unit at any stage s and $\hat{X}_{b,b'} = 1$. Otherwise, $\hat{X}_{b,b'} = 0$ and batch b' is queued before. Because eq 21 is still applied for ordering the batches containing the same product, and the set of sequencing constraints S3 is given by eqs 21, 23a, and 23b.

$$CT_{bs} + \tau_{ii'j} \le ST_{b's} + H(1 - \hat{X}_{b,b'}) + H(2 - Y_{bj} - Y_{b'j}) \quad \forall\ b \in B_i,\ b' \in B_{i'}\ (b < b'),\ j \in J_{i,i',s},\ (i, i') \in I\ (i < i'),\ s \in S \tag{23a}$$

$$CT_{b's} + \tau_{i'ij} \le ST_{bs} + H\hat{X}_{b,b'} + H(2 - Y_{bj} - Y_{b'j}) \quad \forall\ b \in B_i,\ b' \in B_{i'}\ (b < b'),\ j \in J_{i,i',s},\ (i, i') \in I\ (i < i'),\ s \in S \tag{23b}$$

Sequencing schemes based on the idea of global precedence have the advantage of reducing the number of sequencing binary variables while properly handling sequence-dependent changeovers. Besides, additional preordering rules based on the due dates associated with batches can be easily introduced.

3.3.9. Tardiness and Makespan Definitions. Equations 24 and 25 define the schedule makespan and the delivery tardiness of order (i, d), respectively. Equation 26 replaces eq 25 when every order must be delivered on time.

$$CT_{bs^1} \le MK \qquad \forall\ b \in B_i,\ i \in I \tag{24}$$

$$CT_{bs^1} - d \le T_{id} \qquad \forall\ b \in B_{id},\ d \in D_i,\ i \in I \tag{25}$$

$$CT_{bs^1} \le d \qquad \forall\ b \in B_{id},\ d \in D_i,\ i \in I \quad (s^1 = \text{last stage}) \tag{26}$$

Because of assumption 6 (section 2), order (i, d') must be completed before order (i, d) if $d, d' \in D_i$ and $d' < d$. Since a batch $b' \in B_{id'}$ can also be used to fulfill the order (i, d), it may potentially increase the tardiness T_{id} of order (i, d) if it is completed after time d. In this situation, the condition $d' + T_{i,d'} > d$ holds and eq 27 is required to set the appropriate value for T_{id}. This constraint (27) is written for each pair of consecutive due dates $(d', d) \in D_i$, and it becomes redundant if $T_{i,d'} = 0$. As stated by eq 25, the lower bound of $T_{i,d'}$ is defined by the finishing time of the batch $b' \in B_{id'}$ assigned to the order (i, d') that is completed last.

$$d' + T_{id'} \le d + T_{id} \qquad \forall\ d, d' \in D_i,\ i \in I\colon d' = d - 1 \tag{27}$$

3.4. Alternative Objective Functions. The alternative problem goals given by eqs 28 and 29 aim to minimize either the overall weighted tardiness or the schedule makespan. Additional tightening constraints A1 and A2 helping to enhance the model's computational performance for those objective functions are presented in the Appendix.

$$\text{minimize} \sum_{i \in I} \sum_{d \in D_i} \alpha_{id} T_{id} \tag{28}$$

$$\text{minimize MK} \tag{29}$$

When the objective function 28 is adopted, the proposed detailed MILP formulation using the sequencing scheme S1 comprises the set of constraints 10–19, 20a, 20b, 25, 27, and A1. If instead the minimum makespan 29 is the problem goal, the MILP model includes the constraints 10–19, 20a, 20b, 24, and A2. In case some production orders have specific delivery dates and the makespan is minimized, it will be assumed that the delivery dates are strictly satisfied by also considering constraint 26. When the sequencing scheme S2 is applied, eqs 19, 20a, and 20b are replaced by eqs 21, 22a, and 22b in the problem constraint set. The third alternative is to use the sequencing scheme S3 based on the constant batch ordering rule. In such a case, eqs 19, 20a, and 20b are replaced by eqs 21, 23a, and 23b. Whatever is the selected objective function, the tentative number of batches for each production order (i, d) is computed through the procedure depicted in Figure 2 based on eqs 4, 6, 7, and 8′. The reference size bs_i is usually computed using eqs 1 and 4.

4. THE GROUP-BASED MONOLITHIC FORMULATION

Real-life problems usually require scheduling dozens or hundreds of batches every week in order to satisfy multiple customer orders at different promised due dates. As a result, the number of 0−1 allocation and sequencing variables to be included in a detailed problem formulation shows an exponential increase and the computational cost becomes extremely large. To reduce the size and complexity of the mathematical model for huge problems, an approximate group-based formulation has been developed. It relies on a realistic methodology commonly used by industry practitioners. Its main purpose is to substantially reduce the computational effort for large multistage batch scheduling problems and still find good solutions reasonably quickly.

In real production environments, the scheduler often assumes that batches destined for the same production order follow the same manufacturing route and are consecutively processed. In other words, batches of the same product due at the same date are packed together and treated as a single entity for allocation and sequencing purposes. In this way, a significant decrease in the number of 0−1 allocation and sequencing variables is achieved.

To improve this approximate methodology used in industry, more than one group of batches $k \in K_{id}$ that may be processed in parallel at different units of any stage $s \in S$ will be defined for each customer order (i, d). Moreover, each generic batch $b \in B_{id}$ for order (i, d) should at most belong to one of the groups $k \in K_{id}$. Figure 3 shows a schematic representation of the hierarchical decisions simultaneously taken by the approximate cluster-based formulation. Tentative sets of generic batches B_{id} and clusters K_{id} for each production order (i, d) are both defined by the user. As shown in Figure 3, the group-based formulation selects the elements of B_{id} to actually process and simultaneously assigns them to one of the selected clusters $k \in K_{id}$. A cluster with no assigned batch is a dummy group ignored by the formulation. In addition, a manufacturing route is assigned to each existing cluster by allocating units to existent groups, and the sizes of allotted batches are also selected to establish the content of every group. Also, the clusters allocated to the same unit at each stage are sequenced and the starting/finish processing times for each group are determined. At the same time, the individual batches in each cluster are also scheduled. In contrast to the detailed approach, where allocation and sequencing decisions are associated with individual lots, such variables in the cluster-based approach are related to groups of batches sharing the same manufacturing route and processed one after another in common equipment units. As the number of groups is well below the number of batches, a significant saving in binary variables is thus obtained.

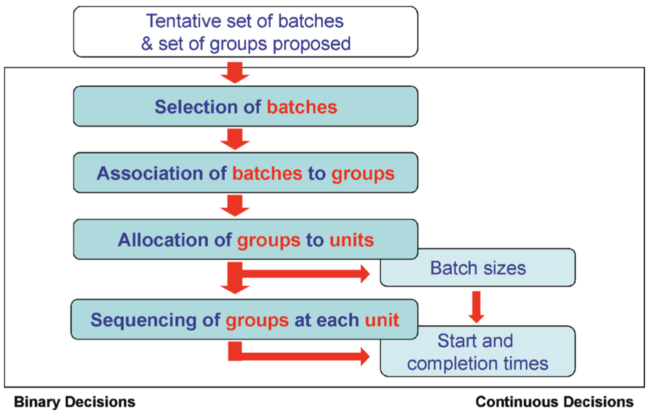

Tentative set of batches & set of groups proposed
Selection of batches
Association of batches to groups
Allocation of groups to units
Batch sizes
Sequencing of groups at each unit
Start and completion times
Binary Decisions
Continuous Decisions

Figure 3. Hierarchical decisions taken by the cluster-based approach.

4.1. Defining Generic Groups for Each Order (*i*, *d*). In section 3.2, a simple methodology for defining a tentative set of batches B_{id} for each order (i, d) that guarantees the discovery of the optimal schedule was presented. In a similar way, the cluster-based approach needs to adopt an adequate number of clusters for each order (i, d). When $|B_{id}| = |K_{id}|$, the cluster-based model becomes equivalent to the detailed batch scheduling formulation but it includes a significant number of redundant solutions. On the other hand, $|K_{id}| = 1$ stands for the practical scheme used in industry. Then, a good option is to start with $|K_{id}| = 1$ and iteratively rise the cardinality of K_{id} by 1, especially for large production orders, until no further improvement in the objective function is achieved.

In order to define an adequate number of clusters for each order (i, d), a new parameter ks_i for each product i is introduced. The parameter ks_i represents an estimation of the average number of batches in each cluster of product i. Given the value of ks_i, the cardinality of the proposed set of clusters $k \in K_{id}$ for the order (i, d) can be computed through eq 30.

$$|K_{id}| = \left\lceil \frac{|B_{id}|}{ks_i} \right\rceil \qquad \forall\ i \in I,\ d \in D_i \tag{30}$$

4.2. Model Constraints. *4.2.1. Selecting Groups and Allocating Batches to Existing Groups.* The continuous variable $Z_k \in [0, 1]$ is used to indicate that cluster $k \in K_{id}$ does exist and a number of batches have been assigned to it. Besides, the binary variable W_{bk} indicates that batch $b \in B_{id}$ has been allocated to cluster $k \in K_{id}$ whenever $W_{bk} = 1$. If $W_{bk} = 1$ for some batch $b \in B_{id}$, then eq 31 makes $Z_k = 1$ and cluster k is included in the solution. If none of the batches $b \in B_{id}$ is assigned to cluster k, then eq 32 drives Z_k to 0 and k becomes a dummy cluster. Moreover, eq 33 states that each batch can at most take part of a single cluster.

$$W_{bk} \le Z_k \qquad \forall\ b \in B_{id},\ k \in K_{id},\ d \in D_i,\ i \in I \tag{31}$$

$$Z_k \le \sum_{b \in B_{id}} W_{bk} \qquad \forall\ k \in K_{id},\ d \in D_i,\ i \in I \tag{32}$$

$$\sum_{k \in K_{id}} W_{bk} \le 1 \qquad \forall\ b \in B_{id},\ d \in D_i,\ i \in I \tag{33}$$

4.2.2. Allocating Clusters to Equipment Units. A selected cluster $k \in K_{id}$ featuring $Z_k = 1$ must be allocated to an available equipment item $j \in J_{is}$ at every processing stage $s \in S$. Such a condition forcing one of the variables Y_{kj}, $j \in J_{is}$, to be equal to 1 at any stage $s \in S$ is imposed through eq 34.

$$Z_k = \sum_{j \in J_{is}} Y_{kj} \qquad \forall\ k \in K_{id},\ d \in D_i,\ i \in I,\ s \in S \tag{34}$$

4.2.3. Symmetry-Breaking Constraints. Two sources of redundant solutions must be eliminated to speed up the search for the optimal group-based schedule through the addition of the so-called symmetry-breaking constraints. Such redundancy sources enlarge the solution space by generating a substantial number of equivalent schedules. They come from (a) the process of allocating batches to clusters and (b) the process of assigning clusters to equipment units. In order to eliminate the first source (a), batches $b \in B_{id}$ associated with order (i, d) are orderly allocated to clusters $k \in K_{id}$ in such a way that (1) a batch b cannot be assigned to cluster k if the prior cluster $(k - 1)$ has not been activated (i.e., $Z_{k-1} = 0$) and (2) a batch b can be assigned to cluster k only if the previous batch $(b - 1)$ in the set B_{id} has been allocated either to cluster k or to the previous cluster $(k - 1)$. These two rules are enforced through eqs 35 and 36, respectively.

$$Z_k \le Z_{(k-1)} \qquad \forall\ (k-1),\ k \in K_{id},\ d \in D_i,\ i \in I \tag{35}$$

$$W_{bk} \le W_{(b-1)k} + W_{(b-1)(k-1)} \qquad \forall\ (b-1),\ b \in B_{id},\ (k-1),\ k \in K_{id},\ d \in D_i,\ i \in I \tag{36}$$

Constraint 35 specifies that group k cannot be activated if group $(k - 1)$ is a dummy cluster. By eq 36, batches for order (i, d) effectively assigned to the active cluster $k \in K_{id}$ are always consecutive elements of the set B_{id}. As a result, nonselected batches and clusters will always arise at the end of sets B_{id} and K_{id}, respectively.

Equivalent solutions can also be generated during the allocation of groups to units, i.e., source b. To partially eliminate source b, the active clusters $k \in K_{id}$ should be orderly assigned to equipment units at some selected stage s^*. In other words, a cluster k can be allocated to unit $j \in J_{is^*}$ only if the previous group $(k-1)$ is processed in unit $j' \leq j$ with $j' \in J_{is^*}$. Then, constraints similar to those presented in section 3.3.2 to avoid symmetric solutions when assigning individual batches to equipment items should be applied, but in this case batches are replaced by groups. As previously discussed, the additional constraint 37 restrains the process of orderly allocating groups to units in the selected stage $s^* \in S$. As already discussed in section 3.3.2, eq 37 cannot be applied to all processing stages because it will cut a portion of the problem feasible region by creating some artificial links among the cluster processing routes.

$$Y_{kj} \leq \sum_{\substack{j' \in J_{is^*} \\ j' \leq j}} Y_{(k-1)j} \qquad \forall\ (k-1), k \in K_{id}, d \in D_i, j \in J_{i,s^*}, i \in I \tag{37}$$

Constraint 37 does not completely prevent the generation of symmetric solutions during the unit allocation process because two groups related to the same order (i, d) may still have a common processing route. Fortunately, the sequencing constraints presented in section 4.2.9 further reduce the impact of the redundancy source b.

4.2.4. Allocating Individual Batches to Equipment Units.

The continuous variable $V_{bkj} \in [0, 1]$ is used to indicate that batch $b \in B_{id}$ associated with an existent cluster $k \in K_{id}$ should be assigned to exactly one unit $j \in J_{is}$ at every stage s. Such a condition is enforced by eq 38. Besides, eq 39 establishes that batch b cannot be allocated to unit j if the associated cluster k is not processed in unit j. Then, the condition $V_{bkj} = 1$ will hold only if $W_{bk} = 1$ and $Y_{kj} = 1$.

$$W_{bk} = \sum_{j \in J_{is}} V_{bkj} \qquad \forall\ k \in K_{id}, b \in B_{id}, d \in D_i, i \in I, s \in S \tag{38}$$

$$\sum_{b \in B_{id}} V_{bkj} \leq |B_{id}|\, Y_{kj} \qquad \forall\ k \in K_{id}, d \in D_i, j \in J_{is}, i \in I, s \in S \tag{39}$$

4.2.5. Choosing Batch Sizes To Meet Customer Orders.

If batch $b \in B_{id}$ is allocated to unit j, its size must be limited to the allowable range $[q_{ij}^{\min}, q_{ij}^{\max}]$. Then, the size of an existent batch b processed in unit j can be regarded as composed by two parts: a fixed part equal to $q_{ij}^{\min}$ and a variable portion Q_{bkj} that ranges between 0 and Δ_{ij}. The continuous variable $Q_{bkj} \in [0, \Delta_{ij}]$ may take a positive value only if b is a selected batch associated with cluster k (i.e., $W_{bk} = 1$) and cluster k is processed in unit j (i.e., $Y_{kj} = 1$). In other words, Q_{bkj} may be positive only if $V_{bkj} = 1$.

Compact constraints 40 and 41 work together to guarantee an appropriate value for Q_{bkj}. Constraint 40 drives the value of the variable portion Q_{bkj} for all batches $b \in B_{id}$ belonging to cluster $k \in K_{id}$ to 0 if group k has not been allotted to unit j. In turn, eq 41 either drives Q_{bkj} to 0 when $V_{bkj} = 0$ or imposes the upper bound Δ_{ij} on the value of Q_{bkj} if $V_{bkj} = 1$. Variables Q_{bkj} are not defined if $\Delta_{ij} = 0$.

$$\sum_{b \in B_{id}} Q_{bkj} \leq |B_{id}| \Delta_{ij} Y_{kj} \qquad \forall\ k \in K_{id}, d \in D_i, j \in J_{is}, i \in I, s \in S\colon \Delta_{ij} > 0 \tag{40}$$

$$\sum_{\substack{j \in J_{is} \\ \Delta_{ij} > 0}} Q_{bkj} \leq \sum_{\substack{j \in J_{is} \\ \Delta_{ij} > 0}} \Delta_{ij} V_{bkj} \qquad \forall\ b \in B_{id}, k \in K_{id}, d \in D_i, i \in I, s \in S \tag{41}$$

Based on the definition of Q_{bkj} given by constraints 40 and 41 for every stage $s \in S$, the size of batch $b \in B_{id}$ can be defined through eq 42 as follows:

$$\mathrm{BS}_b = \sum_{k \in K_{id}} \sum_{j \in J_{is}} (q_{ij}^{\min} V_{bkj} + Q_{bkj}) \qquad \forall\ b \in B_{id}, d \in D_i, i \in I, s \in S \tag{42}$$

Constraint 42 replaces eq 12 at the group-based approximate formulation. Instead, eq 14 ensuring the fulfillment of all customer orders can be still included without any change in the group-based model.

4.2.6. Batch Processing Times at Every Stage.

After choosing the batch size, the processing time of batch $b \in B_{id}$ given by PT_{bkj} is defined by eq 43 as the sum of two contributions: (a) a fixed component ft_{ij} that is common to all batches associated with cluster $k \in K_{id}$ processed in unit $j \in J_s$ (i.e., $V_{bkj} = 1$) and (b) a variable contribution that rises with the total size of batch b at a rate vt_{ij}.

$$\mathrm{PT}_{bkj} = \mathrm{ft}_{ij} V_{bkj} + \mathrm{vt}_{ij}(q_{ij}^{\min} V_{bkj} + Q_{bkj}) \qquad \forall\ b \in B_{id}, k \in K_{id}, d \in D_i, j \in J_{is}, i \in I, s \in S \tag{43}$$

4.2.7. Starting/Finishing Times for Selected Batches and Clusters at Every Stage.

Each selected cluster or group $k \in K_{id}$ for order (i, d) includes a number of *consecutive* elements of the set B_{id} with a common processing route. Let us suppose that batches $b, (b+1) \in B_{id}$ were selected and assigned to group k. To avoid equivalent solutions, it will be assumed that the generic batch b is processed right before $(b+1)$ in the unit allotted to group k at every stage $s \in S$. Let ST_{bks} and CT_{bks} denote the starting and completion times for the stage s of batch $b \in B_{id}$ belonging to cluster $k \in K_{id}$. Equations 44 and 45 provide lower bounds on the value of ST_{bks} for any batch b in the existing cluster $k \in K_{id}$. Equation 44 indicates that nonfictitious batches assigned to cluster $k \in K_{id}$ are consecutively processed one by one in the same unit $j \in J_{is}$ at every stage $s \in S$. On the other hand, eq 45 accounts for the fact that stage $(s+1)$ of batch b can be started only if the previous stage s has ended. The new continuous variables $\mathrm{ST}_{ks}^{\mathrm{G}}$ and $\mathrm{CT}_{ks}^{\mathrm{G}}$ stand for the starting and completion time of cluster k at stage s, respectively. Every batch $b \in B_{id}$ of cluster $k \in K_{id}$ must start and finish within the time slot $[\mathrm{ST}_{ks}^{\mathrm{G}}, \mathrm{CT}_{ks}^{\mathrm{G}}]$ at any stage $s \in S$. Such conditions hold if the processing of cluster k at stage s begins before the starting time of its first batch b_{id}^{f} $(\mathrm{ST}_{b^{\mathrm{f}}ks})$ and after the completion time of its last batch b_{id}^{l} $(\mathrm{CT}_{b^{\mathrm{l}}ks})$. These bounds on the values of $\mathrm{ST}_{ks}^{\mathrm{G}}$ and $\mathrm{CT}_{ks}^{\mathrm{G}}$ for every existing group k are given by eqs 46 and 47.

$$\mathrm{ST}_{bks} + \sum_{j \in J_{is}} \mathrm{PT}_{bkj} \le \mathrm{ST}_{(b+1),ks}$$
$$\forall\ b \in B_{id}\ (b < b^{l}_{id}),\ k \in K_{id},\ d \in D_i,\ i \in I,\ s \in S \tag{44}$$

$$\mathrm{ST}_{bks} + \sum_{j \in J_{is}} \mathrm{PT}_{bkj} \le \mathrm{ST}_{bk(s+1)}$$
$$\forall\ b \in B_{id},\ k \in K_{id},\ d \in D_i,\ i \in I,\ s \in S\ (s < s^{l}) \tag{45}$$

$$\mathrm{ST}^{G}_{ks} \le \mathrm{ST}_{bks}$$
$$\forall\ b \in B_{id},\ k \in K_{id},\ d \in D_i,\ i \in I,\ s \in S\text{: } b = b^{f}_{id} \tag{46}$$

$$\mathrm{ST}_{bks} + \sum_{j \in J_{is}} \mathrm{PT}_{bkj} \le \mathrm{CT}^{G}_{ks}$$
$$\forall\ b \in B_{id},\ k \in K_{id},\ d \in D_i,\ i \in I,\ s \in S\text{: } b = b^{l}_{id} \tag{47}$$

As every existent cluster $k \in K_{id}$ always contains consecutive elements of B_{id}, batches $b \in B_{id}$ not allocated to group k (i.e., featuring $V_{bkj} = 0$ and $Q_{bkj} = 0$ for every unit $j \in J_{is}$ at any stage s) will arise in the set B_{id} either before the first lot ($b < b^{f}_{k}$) or after the last one ($b > b^{l}_{k}$) belonging to group k. Such lots may be dummy or have been assigned to other groups $k' \in K_{id}$ ($k' \neq k$). According to eq 43, their associated processing times ($\sum_{j \in J_{is}} \mathrm{PT}_{bkj}$) at any stage s will be equal to 0, and consequently eqs 44 and 45 reduce to the following: (i) $\mathrm{ST}_{bkj} \le \mathrm{ST}_{b^{f}ks}$ for any $b < b^{f}_{k}$, and (ii) $\mathrm{ST}_{b^{l}ks} + \sum_{j \in J_{is}} \mathrm{PT}_{b^{l}kj} \le \mathrm{ST}_{bks}$ for any $b > b^{l}_{k}$, respectively. If the makespan or the total tardiness is to be minimized, the values of ST_{bks} for such batches, if necessary, will move toward the related bounds. In this way, ST^{G}_{ks} will tend to coincide with $\mathrm{ST}_{b^{f}ks}$ and CT^{G}_{ks} will approximate $\mathrm{CT}_{b^{l}ks}$.

4.2.8. Topological Constraints. Topology constraints are needed to restrict the number of alternative routes a group can follow throughout the plant. If group $k \in K_{id}$ is processed in unit $j \in J_{is}$ (i.e., $Y_{kj} = 1$) at stage s, constraint 48 guarantees that the unit allocated to group k at the next stage ($s + 1$) not only belongs to the set of eligible units $J_{i,s+1}$, but also is reachable from unit j.

$$Y_{kj} \le \sum_{j' \in (J_{i,s+1} \cap J^{(j)}_{s+1})} Y_{kj'}$$
$$\forall\ k \in K_i,\ j \in J_{is},\ i \in I,\ s \in S - \{s^{l}\} \tag{48}$$

4.2.9. Sequencing Clusters Processed in the Same Equipment Item. Groups of batches of the same or different products allocated to the same equipment item must be processed one by one. To determine the queue of clusters at every unit, 0−1 sequencing variables $X^{G}_{k,k',s}$ ordering pairs of existent clusters $k, k' \in K$ at every stage s are to be defined. If $X^{G}_{k,k',s} = 1$, cluster k is processed before k' at stage s, and $X^{G}_{k,k',s} = 0$ denotes that cluster k' is first completed. The value of the sequencing variable $X^{G}_{k,k',s}$ will be meaningful only if both clusters $k, k' \in K$ exist and have been assigned to the same unit at stage s. As said before, sequencing variables can also be used to further prevent the generation of symmetric solutions. Constraints 49, 50a, and 50b are the clustered versions of the sequencing constraint set S2 that replace eqs 21, 22a, and 22b of the proposed formulation at the level of individual batches. Equation 49 is used to queue clusters associated with the same product and processed in the same unit, whether they belong or do not belong to the same production order. The definition of set K_i is such that k appears before k' if either (k, k') both belong to the same order (i, d), or $k \in K_{id}$, $k' \in K_{id'}$ and the group k features the earlier due date ($d < d'$). For clusters associated with different products, the sequencing constraints are given by eqs 50a and 50b and the value of $X^{G}_{k,k',s}$ is found by solving the group-based approximate formulation.

$$\mathrm{CT}^{G}_{ks} \le \mathrm{ST}^{G}_{k's} + H(2 - Y_{kj} - Y_{k'j})$$
$$\forall\ k, k' \in K_i,\ j \in J_{is},\ i \in I,\ s \in S\ (k < k') \tag{49}$$

$$\mathrm{CT}^{G}_{ks} + \tau_{ii'j} \le \mathrm{ST}^{G}_{k's} + H(1 - X^{G}_{k,k',s}) + H(2 - Y_{kj} - Y_{k'j})$$
$$\forall\ k \in K_i,\ k' \in K_{i'},\ j \in J_{ii's},\ i, i' \in I,\ s \in S,\ \text{: } (i < i') \tag{50a}$$

$$\mathrm{CT}^{G}_{k's} + \tau_{i'ij} \le \mathrm{ST}^{G}_{ks} + HX^{G}_{k,k',s} + H(2 - Y_{kj} - Y_{k'j})$$
$$\forall\ k \in K_i,\ k' \in K_{i'},\ j \in J_{ii's},\ i, i' \in I,\ s \in S,\ \text{: } (i < i') \tag{50b}$$

The number of sequencing variables can be further decreased by using a sequencing pattern often found at good feasible solutions. It is the so-called *constant batch ordering rule* prescribing that clusters sharing an equipment item in two or more stages present the same relative ordering on the queues of such units. An identical rule was already applied in section 3.3.8 but at the level of individual batches. If the constant ordering rule is applied, constraints 50a and 50b should be replaced by eqs 51a and 51b, and the new set of sequencing constraints S3 therefore includes eqs 49 , 51a, and 51b. It is observed that a unique sequencing binary $\hat{X}^{G}_{k,k'}$ for each pair of clusters k and k' (with $i \neq i'$) potentially sharing the same unit at some processing stage is to be defined. Clusters k and k' contain batches of products (i, i') with $i \neq i'$.

$$\mathrm{CT}^{G}_{ks} + \tau_{ii'j} \le \mathrm{ST}^{G}_{k's} + H(1 - \hat{X}^{G}_{k,k'}) + H(2 - Y_{kj} - Y_{k'j})$$
$$\forall\ k \in K_{id},\ k' \in K_{i'd'},\ d \in D_i,\ d' \in D_{i'},\ j \in J_{ii's},$$
$$i, i' \in I,\ s \in S,\ \text{: } (i < i') \tag{51a}$$

$$\mathrm{CT}^{G}_{k's} + \tau_{i'ij} \le \mathrm{ST}^{G}_{ks} + H\hat{X}^{G}_{k,k'} + H(2 - Y_{kj} - Y_{k'j})$$
$$\forall\ k \in K_{id},\ k' \in K_{i'd'},\ d \in D_i,\ d' \in D_{i'},\ j \in J_{ii's},$$
$$i, i' \in I,\ s \in S,\ \text{: } (i < i') \tag{51b}$$

4.2.10. Tardiness and Makespan Definitions. As stated by constraint 52, the makespan MK is defined as the lowest upper bound for the completion time of any existent cluster. Besides, the cluster $k \in K_{id}$ associated with the production order (i, d) last completed determines the tardiness of order (i, d) through eq 53. If every production order must be delivered on time, eq 53 should be replaced by eq 54.

$$\mathrm{CT}^{G}_{ks^{l}} \le \mathrm{MK} \qquad \forall\ i \in I,\ k \in K_i \tag{52}$$

$$\mathrm{CT}^{G}_{ks^{l}} - d \le T_{id} \qquad \forall\ i \in I,\ d \in D_i,\ k \in K_{id} \tag{53}$$

$$\mathrm{CT}^{G}_{ks^{l}} \le d \qquad \forall\ i \in I,\ d \in D_i,\ k \in K_{id} \tag{54}$$

4.3. Objective Functions. The objective functions 28 and 29 defined in section 3.4 are also applied when the cluster-based formulation is used. If the total weighted tardiness is to be minimized, the MILP group-based formulation assuming the sequencing scheme S2 includes the set of constraints 14,

Table 1. Model Statistics and Computational Results for Examples 1 and 2

	example 1 (makespan)				example 2 (tardiness)			
		detailed model: section 3				detailed model: section 3		
	ref 17[a]	S1	S2	S3	ref 17[a]	S1	S2	S3
binary variables	287	171	168	116	578	325	316	205
continuous variables	34	131	131	131	65	199	199	199
constraints	712	819	810	810	1510	1664	1637	1637
MILP solution	30.8	30.8	30.8	30.8	0.0	0.0	0.0	0.0
CPU time (s)	2.8	0.56	0.53	0.49	126.3	1.99	2.84	2.63
nodes	1667	528	1059	822	48710	1953	1859	2417
iterations		4827	7432	5778		14508	16468	20089

[a]Pentium 4 2.8 GHz computer running GAMS 22.0/CPLEX 10.0.

31−49, 50a, 50b, 53, and A1. If instead the makespan is minimized and some production orders have strict due dates, the group-based model comprises the constraints 14, 31−49, 50a, 50b, 52, 54, and A2. In case the sequencing scheme S3 based on the constant batch ordering rule is applied, eqs 50a and 50b are replaced by eqs 51a and 51b. Tightening constraints A1 and A2 are given in the Appendix. Equation 30 is usually applied to determine the proposed clusters for each order.

5. COMPUTATIONAL RESULTS AND DISCUSSION

In this section, five case studies of increasing size and complexity are presented to test the computational efficiency of the two proposed scheduling approaches for multistage batch processes. The first two examples taken from the work of Prasad and Maravelias[17] consider a single customer order per product. Since each order involves a few batches, the problem size is rather small and only the exact formulation (at the level of individual batches) is applied to solve examples 1 and 2. In addition, three new large examples (examples 3−5) were introduced to illustrate the significant savings in CPU time that can be attained by using the group-based scheduling method.

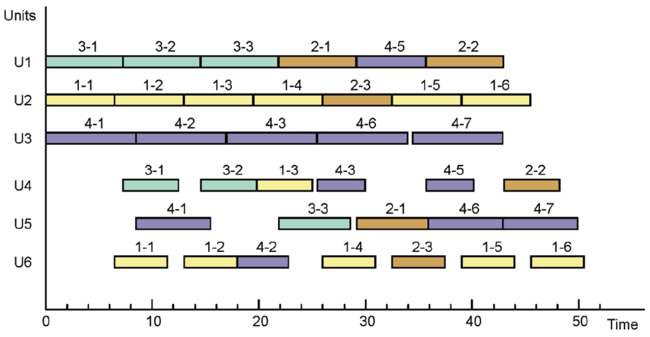


Figure 4. Optimal schedule for example 3 using the detailed formulation.

Table 2. Product Requirements at Every Due Date (in kg) for Example 3

	due dates	
product	30 h	50 h
P_1	90	60
P_2		75
P_3	60	
P_4	80	85

Table 3. Proposed Batches for Each Customer Order at Example 3

	due dates	
product	d_1	d_2
P_1	b_1−b_4	b_5−b_7
P_2		b_1−b_3
P_3	b_1−b_3	
P_4	b_1−b_4	b_5−b_8

Table 5. Number of Batches and Groups for Example 3

	no. of batches per order		no. of groups proposed/used per value of ks_i		
product	d_1	d_2	ks_i = 4	ks_i = 3	ks_i = 2
P_1	4	3	2	3	4/3
P_2		3	1	1	2
P_3	3		1	1	2
P_4	4	4	2	4	4
total			6	9	12/11

Table 4. Computational Results for Example 3

model	sequencing scheme	batches/group param (ks_i)	binary variables, cont variables, constraints	obj function	CPU time (s)	nodes
detailed formulation	S1	–	512, 252, 2593	1.5	104.01	243 175
	S2	–	429, 252, 2356	1.5	84.38	250 482
	S3	–	274, 252, 2356	1.5	27.77	87 950
group-based formulation	S2	4	81, 338, 532	2.0	0.59	65
		3	138, 513, 890	2.0	2.64	6 004
		2	214, 648, 1332	1.5	9.66	52 647
	S3	4	68, 338, 532	2.0	0.34	298
		3	111, 513, 890	2.0	2.62	8 105
		2	162, 648, 1332	1.5	7.44	24 501

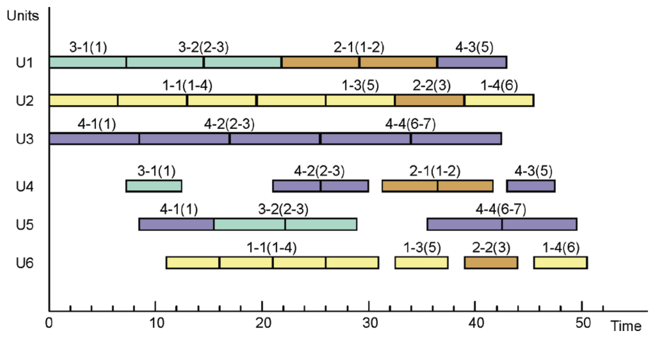


Figure 5. Best schedule for example 3 using the group-based formulation.

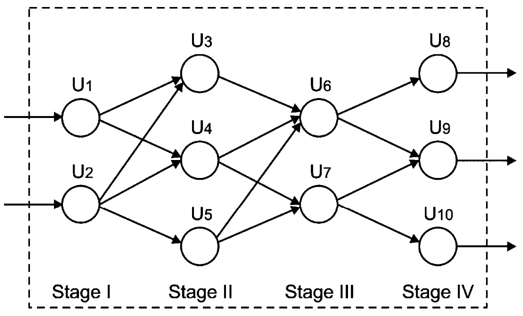


Figure 6. Multistage batch facility for examples 4 and 5.

Example 3 is based on the same two-stage production facility of examples 1 and 2, but considers multiple customer orders per product with different delivery dates. Example 4 deals with a more complex four-stage production facility that takes into account (a) multiple orders per product with different due dates, (b) sequence-dependent changeovers, (c) variable batch sizes, and (d) processing times depending on both the unit and the batch size. No longer can this example be solved to optimality using the formulation at the level of individual batches. The only alternative tool to find good feasible solutions is the proposed cluster-based approach. Three instances of example 4 with a growing number of production orders and the minimum total tardiness as the problem goal have been tackled. In each case, the group-based model has been evaluated using different values for the average number of batches per cluster (ks_i) to study its impact on the model size, the CPU time, and the solution quality.

Example 5 involves the same four-stage batch facility of example 4, but it now includes large make-to-stock orders for six products and the minimum makespan as the problem target. This example permits showing the computational advantage of the group-based approach even if order due dates are omitted and the minimization of the makespan is the problem objective. The two proposed monolithic approaches have been implemented using the GAMS modeling system. Computational results were obtained using GUROBI 4.0 64 bits with an Intel Core i7 960 at 3.2 GHz clone PC having 16 GB of RAM and allowing up to eight parallel threads. The examples were solved to optimality within a CPU time limit of 1 h. An optimality gap of 10^{-3} has been adopted as the stopping criterion.

5.1. Examples 1 and 2.

Examples 1 and 2 both consider a batch facility with two processing stages and three units per stage that produces eight different products (P_1–P_8). The parallel units in each stage have different capacities, and processing times are batch-size independent. Changeover times between consecutive tasks are assumed to be negligible. Data for examples 1 and 2 can be found as problems P1.M and P2.L in Prasad and Maravelias.[17] Products A–H have been renamed as P_1–P_8 in this work. For comparison purpose, eq 55 was applied to estimate the batch size used to calculate the proposed number of batches for each product i.

$$\mathrm{bs}_i^* = \min_{s\in S}[\min_{j\in J_{is}}(q_{ij}^{\max})] \qquad \forall\ i \in I \tag{55}$$

In this way, $\mathrm{bs}_i^* = 25$ is the reference batch size for every product in both examples.

In example 1, eight customer orders, each one involving a different product, are to be manufactured. Each order has an associated delivery date to be strictly satisfied, and the minimum makespan is the problem goal. On the other hand, example 2 also considers eight orders of different products with different delivery dates, but now the problem goal is the minimization of the total tardiness. Batches are never used to fulfill more than one order in examples 1 and 2. Because $\mathrm{bs}_i^* = 25$ for all i, a maximum of two batches is needed for products P_1, P_2, and P_3 and just one batch is needed for products P_4–P_8 at example 1. The total number of batches is equal to 11. Similarly, two batches are defined for products P_1–P_4, P_6, and P_7, three are defined for P_5, and just one is defined for P_8 at example 2. Therefore, a total of 16 batches have been proposed for example 2. In both examples, symmetry-breaking constraints were just applied to the initial stage $s^* = s_1$.

Table 6. Detailed Schedule of the Activated Batch Clusters for Example 3

			selected groups								
product	due date	req	group	batches	total size	processing route	start	end	order completion	tardiness	remaining inventory
P_1	30.0	90.0	k_1	b_1–b_4	100.0	U_2–U_6	0.0	31.0	31.0	1.0	10.0
	50.0	60.0	k_3	b_5	25.0	U_2–U_6	26.0	37.5			
			k_4	b_6	25.0	U_2–U_6	39.0	50.5	50.5	0.5	0.0
P_2	50.0	75.0	k_1	b_1–b_2	50.0	U_1–U_4	21.9	41.7			
			k_2	b_3	25.0	U_2–U_6	32.5	44.0	44.0	0.0	0.0
P_3	30.0	60.0	k_1	b_1	25.0	U_1–U_4	0.0	12.5			
			k_2	b_2–b_3	50.0	U_1–U_5	7.3	28.9	28.9	0.0	15.0
P_4	30.0	80.0	k_1	b_1	30.0	U_3–U_5	0.0	15.5			
			k_2	b_2–b_3	50.0	U_3–U_4	8.5	30.0	30.0	0.0	0.0
	50.0	85.0	k_3	b_5	25.0	U_1–U_4	36.5	47.5			
			k_4	b_6–b_7	60.0	U_3–U_5	25.5	49.5	49.5	0.0	0.0

Table 7. Batch Size Limits and Processing Time Coefficients for Examples 4 and 5

	batch size (kg)		processing time coefficients															
			fixed (h)								variable (h/kg)							
unit	q_j^{min}	q_j^{max}	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
U_1	75	120	6		6	5	4		7	5	0.15		0.06	0.12	0.1		0.08	0.1
U_2	50	90	4	6	7		6	4	5		0.1	0.08	0.12		0.12	0.1	0.15	
U_3	75	100		5		5	7	5	4	7		0.12	–	0.1	0.15	0.12	0.1	0.05
U_4	70	95	6		8	7	4	8	9	5	0.08		0.08	0.12	0.1	0.15	0.06	0.1
U_5	50	85	5	4	6		4	7			0.12	0.12	0.06		0.1	0.15		
U_6	50	80	4	5		5	6	4	3	8	0.1	0.1	–	0.1	0.12	0.18	0.05	0.1
U_7	75	105		4	7			5	6	5		0.1	0.12			0.12	0.18	0.08
U_8	70	110	6			4	4	8		7	0.08		–	0.08	0.1	0.15		0.1
U_9	60	90	6	5	8	6	4		5	4	0.08	0.12	0.08	0.12	0.1		0.06	0.2
U_{10}	80	115		4	8			8	6	5		0.1	0.08			0.15	0.1	0.04

Table 8. Sequence-Dependent Changeovers (h) for Examples 4 and 5

	$\tau_{ii'}$							
	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
P_1	0.0	3.1	3.4	1.6	5.1	3.3	4.1	2.1
P_2	6.1	0.0	2.5	3.4	1.3	1.7	2.3	1.8
P_3	4.3	6.9	0.0	1.8	3.1	1.7	3.3	1.5
P_4	3.3	3.3	1.2	0.0	1.6	4.4	2.0	3.7
P_5	2.7	3.0	1.4	4.7	0.0	1.4	3.1	2.7
P_6	1.8	1.7	3.1	2.0	5.5	0.0	2.5	1.3
P_7	2.3	3.5	2.7	1.5	1.7	4.0	0.0	3.2
P_8	3.1	1.8	3.5	2.9	2.2	1.1	1.3	0.0

Table 1 presents the computational results for examples 1 and 2 reported by Prasad and Maravelias,[17] and those obtained with the detailed approach presented in section 3. Our detailed formulation has been solved using the three alternative sequencing schemes S1, S2, and S3. The best schedule for example 1 features a makespan of 30.8 h, while the minimum tardiness for example 2 is equal to 0. In any case, the detailed approach always provides the optimal solution for both examples. The agreement with the results reported by Prasad and Maravelias[17] holds for the three sequencing schemes (see Table 1). However, the proposed problem model includes a lower number of binary variables and requires a shorter CPU time compared with the model of Prasad and Maravelias,[17] even if the exact sequencing scheme S1 is considered. This saving in binary variables can be easily explained because the model of Prasad and Maravelias[17] requires defining 0−1 batch selection variables and two sequencing binary variables for each pair of batches that can be processed in the same unit. Despite the large difference in CPU times (at least 5 times faster in example 1 and 44 times faster in example 2), it cannot be claimed that the proposed detailed method outperforms the model of Prasad and Maravelias,[17] because their results were obtained with a machine/solver having a lower performance than the one used in this contribution.

Table 10. Number of Batches and Groups for Example 4a

	no. of batches per order			no. of groups proposed/used per value of ks_i			
product	d_1 = 120	d_2 = 192	d_3 = 240	6	5	4	3
P_1		6		1	2	2	2
P_2	5		5	2	2	4	4
P_3	6	3		2	3	3	3
P_4		4	2	2	2	2	3/2
Total				7	9	11	12/11

Examples 1 and 2 both involve a rather small number of batches for each order, and their optimal schedules can be found in a few CPU seconds by the proposed detailed approach. As a result, there is no sense in solving them again using the group-based method. To deal with adequate examples for the group-based formulation, a higher number of production orders with larger sizes r_{id} must be considered. By doing so, the size of the problem will increase very quickly, hence making the problem unsolvable using the detailed formulation within reasonable CPU times. A medium-size case study based on the same production facility of examples 1 and 2, called example 3, is presented in section 5.2 and solved using the two proposed monolithic approaches to compare their computational efficiencies.

Table 9. Customer Orders for the Three Instances of Example 4 at Various Due Dates

	example 4a			example 4b			example 4c			
product	120 h	192 h	240 h	168 h	336 h	480 h	144 h	288 h	480 h	600 h
P_1		340		450	560			310	475	
P_2	270		290		330	250	240		250	360
P_3	390	210		500	200			350	335	
P_4		275	120			380	255		240	235
P_5				270		470	310	175	165	
P_6					400	500	180	270		155
P_7								315	205	
P_8							160		250	390

Table 11. Computational Results Using the Detailed Model for Example 4

example	sequencing scheme	binary variables, cont variables, constraints	obj function	CPU time (s)	nodes	iterations/nodes
4a	S1	1843, 529, 5198	2.8[a]	3600[b]	2 297 783	20.25
	S2	1459, 529, 4556	1.5904	160.41	117 591	17.75
	S3	565, 529, 4556	1.5904	50.59	24 377	17.06
4b	S1	9730, 1240, 29 032	1408.08[a]	3600[b]	159 055	34.87
	S3	2576, 1240, 26 398	22.78[a]	3600[b]	147 959	53.72
4c	S1	16 310, 1619, 48 196	3339.8[a]	3600[b]	58 757	44.86
	S3	4360, 1619, 44 888	663.95[a]	3600[b]	51 378	51.46

[a]Best possible solution =0.0, relative gap =100%. [b]Time limit exceeded.

Table 12. Computational Results Using the Group-Based Method for Example 4

	batches/group param (ks_i)	binary variables, cont variables, constraints	obj function	CPU time (s)	nodes
example 4a					
sequencing scheme S3	6	96, 648, 1050	12.8	0.29	245
	5	134, 882, 1435	4.2904	1.78	4995
	4	172, 1080, 1810	1.5904	5.31	9657
	3	191, 1153, 1975	1.5904	11.97	21389
sequencing scheme S2	6	140, 648, 1050	12.8	1.08	3066
	5	208, 882, 1435	4.2904	3.23	11175
	4	284, 1080, 1810	1.5904	11.03	20378
	3	323, 1153, 1975	1.5904	8.94	17040
example 4b					
sequencing scheme S3	10	201, 1502, 2429	8.5	8.63	9239
	8	227, 1691, 2733	8.3	13.62	12289
	7	306, 2182, 3634	0	15.90	6311
sequencing scheme S2	10	337, 1502, 2429	8.5	9.86	10691
	8	388, 1691, 2733	8.3	25.34	30810
	7	558, 2182, 3634	0	44.50	21566
example 4c					
sequencing scheme S3	8	439, 2025, 4796	36.6	63.20	60356
	6	473, 2178, 5168	36.6	306.93	311372
	5	609, 2670, 6660	12.4	110.14	48662
	4	757, 3076, 8199	10.8	427.98	120456
	3	958, 3561, 10319	10.8	280.09	46187
sequencing scheme S2	8	963, 2025, 4796	37.85[a]	3600[b]	6839940
	6	1052, 2178, 5168	37.85[a]	3600[b]	4139105
	5	1417, 2670, 6660	14.45[c]	3600[b]	3280813

[a]Best possible solution = 35.7, relative gap = 5.68%. [b]Time limit exceeded. [c]Best possible solution = 10.8, relative gap = 25.2%.

5.2. Example 3.

Example 3 is a new case study introduced in this work to evaluate the computational effectiveness of the cluster-based scheduling approach in comparison with the proposed detailed formulation. Example 3 considers the same production facility of the two previous examples, but in this case only four products (P_1−P_4) are manufactured. Moreover, production orders for products P_1−P_4 are to be fulfilled at two different due dates (30 h, 50 h) as shown Table 2. The problem goal is to find the feasible schedule with the least total tardiness.

Table 3 displays the proposed batches for each customer order that were obtained using both the reference batch size given by eq 55 and the procedure presented in section 3.2. In order to calculate the number of batches for each set B_{id}, bs_i =

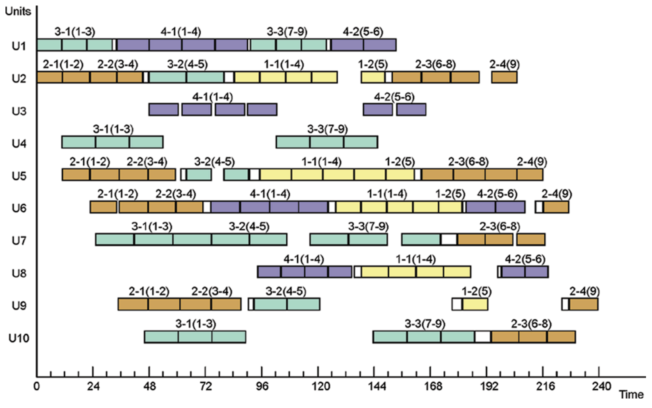


Figure 7. Best schedule found for example 4a.

Table 13. Number of Batches and Groups for Example 4b

	no. of batches per order			no. of groups proposed/used per value of ks_i		
product	d_1 = 168	d_2 = 336	d_3 = 480	10	8	7
P_1	8	10		2	3	4/3
P_2		6	4	2	2	2
P_3	7	3		2	2	2
P_4			6	1	1	1
P_5	5		8	2	2	3
P_6		6	8	2	2	3/2
total				11	12	15/13

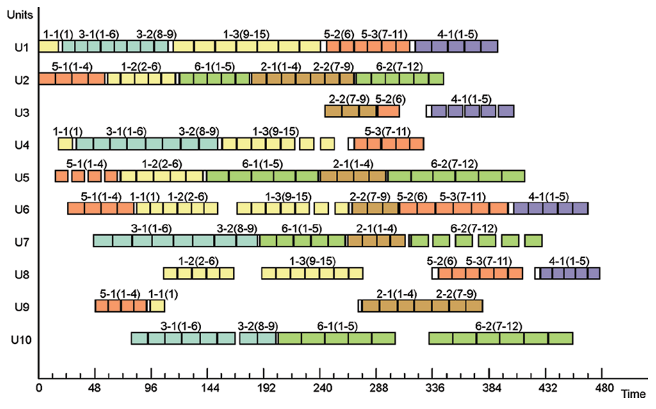


Figure 8. Optimal schedule for example 4b.

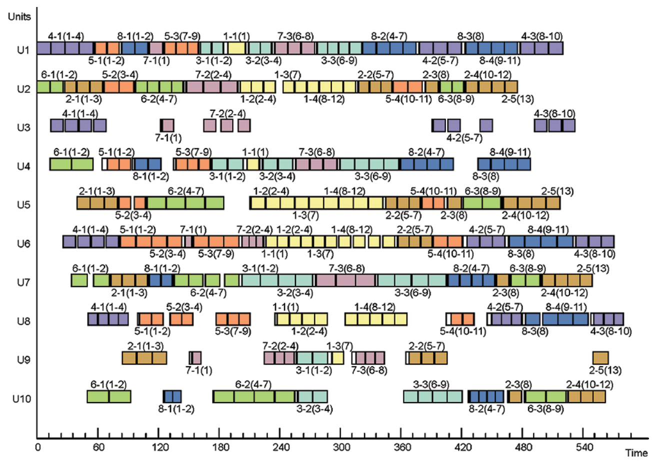


Figure 9. Best solution found for example 4c.

25 and $q_i^{\max} = 30$ have been adopted for every product i. As a result, product P_1 requires $nb_{P_1,d_1} = 4$ and $nb_{P_1,d_2} = 3$ batches to meet its demand at due dates $d_1 = 30$ h and $d_2 = 50$ h, respectively. Thus, batches of P_1 are numbered b_1−b_4 for order (P_1, d_1) and b_5−b_7 for order (P_1, d_2). Similar results for the remaining products are shown in Table 3. Overall, 21 batches should be considered by the problem formulation. Additional parameters for the example are $H = 120$ h and $s^* = s_1$.

Example 3 has first been solved to optimality by using the detailed formulation introduced in section 3. Computational results obtained with the alternative sequencing constraint sets S1, S2, and S3 are presented in Table 4. Using the sequencing scheme S1, a CPU time of 104 s was needed to find the best schedule featuring a total tardiness of 1.5 h. The S1 detailed model includes 512 binary variables and 2593 constraints. When the less conservative sequencing constraint sets S2 and S3 are used, the true optimal solution is still found and the number of binary decisions and the CPU time both significantly decrease. The application of the constant batch ordering rule at every processing stage (scheme S3) reduces the number of binary variables by almost half to 274, and the CPU time decreases more than 3.7 times to 27 CPU s.

Figure 4 presents the Gantt chart for the optimal schedule. Each selected batch has been identified through two indices, with the first one standing for the product and the second standing for the batch number shown in Table 3. The bottleneck stage of the plant is stage s_1 (units U_1−U_3). Inspecting the solution, batch (P_1, b_4) associated with due date d_1 ends at time = 31 h, thus generating a tardiness of 1 h. Besides, batch (P_1, b_6) associated with d_2 is completed at 50.5 h with a tardiness of 0.5 h. Of the original 21 batches proposed, only 18 were selected by the model at the optimum. The constant ordering of batches at every processing stage and the symmetry-breaking condition allocating batches of the same production order to consecutive units at stage $s^* = s_1$ can both be verified with the Gantt chart shown in Figure 4.

Next, the group-based approximate model has also been applied to example 3. The average number of batches per group (ks_i) for each product i is an important parameter of the group-based method that determines the model size and the difficulty

Table 15. Make-to-Stock Orders for Example 5

product	make-to-stock orders
P_1	620
P_2	520
P_3	610
P_4	490
P_5	550
P_6	480

Table 16. Number of Batches and Groups for Example 5

		no. of groups proposed/used per value of ks_i			
	no. of batches per product	11	10	9	7
P_1	11	1	2	2	2
P_2	9	1	1	1	2
P_3	9	1	1	1	2
P_4	7	1	1	1	1
P_5	10	1	1	2	2
P_6	7	1	1	1	1
total	53	6	7	8	10

Table 14. Number of Batches and Groups for Example 4c

	no. of batches per due date				no. of groups proposed/used per value of ks_i				
product	d_1	d_2	d_3	d_4	8	6	5	4	3
P_1		6	8		2	3	4/3	4	5/4
P_2	4		5	6	3	3	4	5	6/5
P_3		5	5		2	2	2	4/3	4/3
P_4	4		3	3	3	3	3	3	4/3
P_5	6	3	3		3	3	4	4	4
P_6	3	4		3	3	3	3	3	4/3
P_7		5	3		2	2	2	3	3/2
P_8	3		4	6	3	3	4	4	5
total					21	22	26/25	30/29	35/29

Table 17. Results for Example 5 Using the Group-Based Approach

sequencing scheme	batches/group param (ks_i)	binary variables, cont variables, constraints	obj function	CPU time (s)	nodes	iterations/nodes
S3	11	111, 1082, 1478	361.2	1.38	2 048	18.22
	10	134, 1289, 1750	361.2	5.23	3 738	34.77
	9	158, 1498, 2029	351.4	41.96	48 149	43.60
	7	205, 1840, 2542	341.1[a]	3070.13	2 869 944	54.25
S2	11	151, 1082, 1478	361.2	1.91	2 038	23.84
	10	188, 1289, 1750	361.2	4.79	5 963	21.92
	9	229, 1498, 2029	351.4	49.69	109 709	25.99
	7	315, 1840, 2542	344.02[b]	3600[c]	4 110 208	40.60

[a]Best possible solution = 340.77, relative gap = 0.001. [b]Best possible solution = 336.3, relative gap = 0.022. [c]Time limit exceeded.

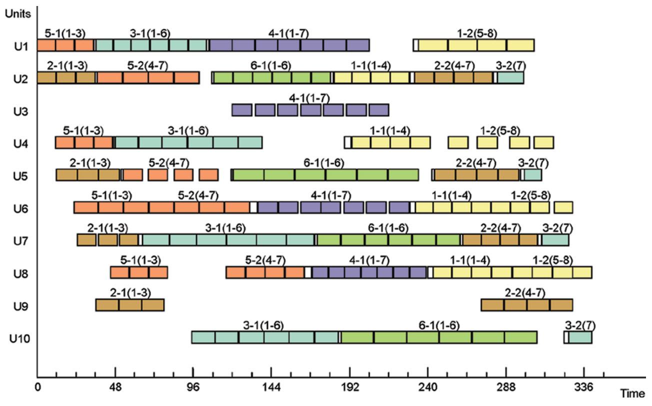


Figure 10. Best solution found for example 5.

to solve it. If a small number of groups is proposed (higher ks_i), poor quality solutions may be found. As the value of ks_i decreases, the number of groups for each order (i, d) and the chance to find a near-optimal solution both increase. On the left side of Table 5, the proposed number of batches for each production order already included in Table 3 is reported. On the right side, the total number of groups proposed/selected for each product i when ks_i is set at 4, 3, and 2, respectively, is shown. The number of selected groups used at the optimal schedule is just indicated when it is lower than the one proposed. For $ks_i = 2$, the total number of groups amounts to 12 while the total number of batches rises to 21. This fact explains the computational advantage of the cluster-based formulation. Computational results obtained with the group-based model using the sequencing schemes S2 and S3 for different values of the parameter ks_i are included at the bottom of Table 4. Notice that the true optimal solution featuring a total tardiness of 1.5 h is found using $ks_i = 2$ for any $i \in I$. The S2-group based formulation needs a CPU time of 9.66 s to find the best solution against 84.3 s required by the exact formulation, i.e., a reduction factor of 8 (see Table 4). Moreover, the number of binary variables decreases by a half by using the group-based problem model. Near-optimal solutions found with $ks_i = 4$ and $ks_i = 3$ both present a total tardiness of 2 h by obviously activating all the proposed groups (6 and 9, respectively). Although never chosen in this work, fractional values of ks_i are permitted.

The Gantt chart for the optimal schedule found by the group-based approach using $ks_i = 2$ is depicted in Figure 5. Each group is identified through two indices, with the first one standing for the associated product and the second indicating the cluster number. In parentheses, the individual batches that each selected group comprises are additionally indicated. Detailed data about the clusters activated by the model to fulfill every customer order are included in Table 6. When a cluster satisfies, partially or totally, two production orders, the remaining portion after fulfilling the first one is reported. This is the case for cluster k_1 of product P1 featuring a total size of 100 kg: 90 kg has been assigned to order (P_1, d_1) and 10 kg has been assigned to order (P_1, d_2). From Table 6, it is observed that 11 clusters are used instead of the 12 proposed, because group k_2 associated with order (P_1, d_1) is a dummy cluster at the optimum.

This case study shows that the cluster-based approach is able to find a solution proven optimal by the detailed formulation, but at much lower computational cost. Moreover, the size of the cluster-based model can be controlled by adjusting the value of the parameter ks_i. Decreasing model sizes are obtained for higher values of ks_i. Even for rather large ks_i, the group-based approach provides good feasible solutions. This model feature is quite important for the solution of large batch scheduling problems.

5.3. Example 4.

In order to cope with industrial-sized scheduling problems featuring larger number of batches, additional case studies have been tackled. Example 4 includes three instances of increasing size and complexity, all involving a batch plant with four processing stages and a total of 10 equipment units. Eight different products (P_1–P_8) are to be manufactured. The plant topology is depicted in Figure 6. The allowable batch size range together with unit-dependent fixed/variable processing time coefficients are presented in Table 7. Besides, Table 8 includes the sequence-dependent changeover time data. Customer orders for each product and their promised delivery dates for the three instances of example 4 are presented in Table 9. The selected problem goal is the minimization of the overall tardiness.

5.3.1. Example 4a.

In example 4a, seven customer orders for products P_1–P_4 with three different delivery dates are to be satisfied (see Table 9). In Table 10, the proposed number of batches for each customer order using the reference batch size given by eqs 1 and 4 and the procedure presented in section 3.2 are reported. Then, bs_i = 60 kg for products P_1 and P_2 and 75 kg for P_3 and P_4. Moreover, q_j^{max} = 80 kg for P_1 and P_4, 90 kg for P_2, and 95 kg for P_3. A total of 31 batches are to be handled. For this example, symmetry-breaking constraints were just applied to the last stage $s^* = s_2$. The computational results obtained with the detailed formulation using sequencing constraint sets S1–S3 are presented in Table 11. When the exact sequencing scheme S1 is applied, the detailed approach fails to find the optimal schedule within the CPU time limit of 1 h. The best schedule discovered presents a total tardiness of 2.8 h. If the other sequencing schemes are adopted, the solution algorithm converges to an improved solution featuring a total tardiness of 1.59 h. As reported in Table 11, it was found in

160.4 s with the suitable scheme S2, and in 50.6 s with S3. These results show the advantage of using less conservative sequencing schemes to discover near-optimal batch schedules at relatively low CPU time. Of the proposed 31 batches, the best schedule just includes 27.

The group-based approach has also been applied to example 4a. The proposed clusters for each product using different values of ks_i are shown on the right-hand side of Table 10. Even for $ks_i = 3$, the approximate formulation considers a number of groups almost 3 times lower than the proposed number of batches. Computational results for example 4a using the group-based approach and the sequencing scheme S2 or S3 are shown in Table 12. For either scheme, the solution algorithm always reaches optimality in a few CPU seconds and the best schedule featuring a total tardiness of 1.59 h was discovered by setting $ks_i = 4$ or lower. The use of the group-based approach instead of the detailed formulation produces a significant reduction in both the model size and the CPU time. The number of binary variables and the CPU time both decrease from 1459 and 160.41 s (for the S2-detailed formulation) to 284 and 11.03 s (for the S2-group based approach), respectively. For the approximate scheme S3, a similar pattern is observed. In this case, the number of binaries drops from 565 to 172, and the CPU time decreases 10 times from 50.59 to 5.31 s.

The best schedule found is depicted in Figure 7. The two tardy clusters are related to orders (P_1, d_2) and (P_3, d_1). In Figure 7, they are labeled as group 1−2 (i.e., cluster k_2 of product P_1) containing batch b_5 and group 3−2 (i.e., cluster k_2 of product P_3) comprising two batches b_4 and b_5. Group 1−2 is completed at time 192.59 h beyond $d_2 = 192$ h, while the processing of cluster 3−2 ends at 121 h one hour after the promised date $d_1 = 120$ h.

5.3.2. Example 4b. Eleven customer orders for six products (P_1−P_6) with three different delivery dates must be fulfilled at example 4b (see Table 9). Since the order sizes are larger compared with the previous instance, a much higher number of batches should be processed. The proposed number of batches for each order is shown in Table 13. A total of 71 batches should be considered to guarantee the discovery of the optimal schedule.

When the detailed approach was applied using either the exact sequencing scheme S1 or even the less conservative S3, the optimality condition cannot be achieved within the CPU time limit of 3600 s (see Table 11). This is so because the required number of binary variables rises to 2576 for scheme S3. The best solution found using the S3-detailed approach is rather poor and features a total tardiness of 22.78 h (see Table 11). In contrast, the group-based formulation with either S2 or S3 discovers the true optimal solution featuring zero tardiness. The number of groups was determined for three different values of the parameter ks_i (see Table 13). In any case, the group-based model was solved to optimality, but the true optimum was found just for $ks_i = 7$ using S2 or S3. Moreover, the search takes 44.5 s of CPU time using S2 and 15.9 s with S3 (see Table 12). Compared with S3-detailed approach, the number of binary variables in the S3 group based approach drops from 2576 to 306. The optimal schedule for example 4b is shown in Figure 8. Of the proposed 71 batches, just 54 lots are processed. Besides, 13 out of the 15 proposed groups have been selected.

5.3.3. Example 4c. Example 4c is the largest instance of example 4. It involves the fulfilment of 21 customer orders for eight products (P_1−P_8) due at four different promised dates (see Table 9). The batches to be considered for each customer order are given in Table 14. Overall, 92 batches and 368 processing tasks should be handled by the proposed formulations. As before, the detailed formulation cannot be solved to optimality within the time limit of 1 h, even if scheme S3 is adopted. The best solution found presents a total tardiness of 663.95 h, i.e., a very poor schedule (see Table 11). On the other hand, the number of clusters to be considered by the group-based approach is shown on the right-hand side of Table 14 for five different values of ks_i. For $ks_i = 4$, the total number of clusters amounts to 30, i.e., one third of the total number of batches. The S3 group based approach has been solved to optimality for each value of ks_i. The best schedule featuring a total tardiness of 10.8 h was discovered for $ks_i = 4$ in 428 s of CPU time (see Table 12). Lowering ks_i to 3 does not produce any change in the optimal tardiness, but the CPU time drops to 280 s. In contrast, when scheme S2 was adopted, the group-based model cannot be solved to optimality. The best schedule found for example 4c is shown in Figure 9. Just 73 out of 92 batches are processed, and 29 out of 30 groups were activated at the optimum for $ks_i = 4$.

Results for the three instances of example 4 all clearly confirm that the group-based approach together with the effective sequencing schemes S2 and S3 is able to find good (or even optimal) solutions for real-world batch scheduling problems in very short computational times.

5.4. Example 5. Data for example 5 are given in Figure 6 and Tables 7 and 8. In addition, Table 15 lists the six make-to-stock production orders to be processed with the minimum makespan selected as the problem goal. Table 16 includes the number of batches proposed for each order and the number of groups per product for different values of ks_i. A total of 53 batches are to be handled, and 6−10 groups should be considered depending on the value of ks_i. Computational results for example 5 using the group-based model with the sequencing constraint set S2 and the less conservative scheme S3 are shown in Table 17. When scheme S3 is applied, example 5 is solved to optimality for the four proposed values of ks_i. The best solution was found for $ks_i = 7$ in a CPU time of 3070 s and features a makespan of 341.1 h (see Figure 10). However, good feasible solutions were obtained in a much lower CPU time for smaller ks_i. The more suitable sequencing scheme S2 provides results similar to those of S3 at comparable computational costs except for $ks_i = 7$.

6. CONCLUSIONS

Two new MILP monolithic approaches for the scheduling of multistage batch facilities have been developed. The proposed methods handle multiple customer orders for each product with different due dates, variable processing times, and sequence-dependent changeover times. Both models include effective symmetry-breaking constraints in terms of allocation variables. Production orders are first converted into a set of batches all featuring the same due date through a novel procedure that ensures the discovery of the optimal schedule. One of the formulations performs selection, sizing, allocation, sequencing, and timing decisions for the individual batches. However, its computational efficiently rapidly deteriorates if around 25 batches or more are to be scheduled. To allow its use for larger problems, two other sequencing constraint sets (called S2 and S3) were tested. If two batches of the same product are allocated to the same unit, the set S2, in agreement with the model assumptions, supposes that the batch with the earlier due date is processed before. Conversely, the less conservative scheme S3 based on the constant batch ordering rule of Marchetti and Cerdá[15] states that two batches sharing a processing unit at different stages are queued in the same order at the common units. By using the sequencing constraint set S2 or S3, the detailed formulation is able to solve problems involving 30−35 batches at convenient CPU times. To deal with industrial-sized scheduling problems, this work additionally introduces an MILP cluster-based formulation. In this approach, batches destined for the same customer order are first grouped into clusters. Afterward, the proposed cluster-based model simultaneously selects the clusters to be processed and their contents (number and size of batches), and schedule the clusters so as to minimize either the total tardiness or the makespan. The group-based formulation in combination with the sequencing constraint set S2 or S3 was applied to the solution of very large examples involving up to 92 batches. In every case, it was solved to optimality at reasonable computational cost. The proposed number of clusters for each order is defined by choosing the value of the parameter ks_i representing the average number of batches per cluster. For higher ks_i, the number of clusters, the model size, and the solution time all significantly decrease. However, the chance of discovering the true optimal schedule also diminishes. Nonetheless, very good solutions for large case studies are still found at low CPU times with rather high ks_i values. Besides, the cluster-based approach can efficiently handle problems involving make-to-stock orders with no associated due dates and the minimum makespan as the problem objective.

APPENDIX: TIGHTENING CONSTRAINTS FOR THE PROPOSED FORMULATIONS

Additional constraints A1 and A2 are proposed to further tighten the feasible region of the MILP detailed model. The proposed cuts are conservative in the sense that they do not remove integer solutions from the feasible space. They are extensions of the tightening constraints proposed by Marchetti et al.[19] for single-stage batch plants. Equation A1 is applied when the problem goal is the least overall tardiness, and eq A2 is used when the minimum makespan is sought. Constant parameters $\mathrm{est}_{j,s}^{\min}$ and $\pi_{j,s}^{\min}$ are the lowest possible earliest start time for a task in unit j and the lowest time needed to complete any batch in unit j, respectively.

$$\sum_{\substack{i\in I_j\\ r_{id}>0}} \sum_{\substack{d'\in D_i\\ d'\le d}} \sum_{b\in B_{id'}} \mathrm{PT}_{bj} \le d - (\mathrm{est}_{j,s}^{\min} + \pi_{j,s}^{\min}) + \sum_{\substack{i\in I_j\\ r_{id}>0}} T_{id} \quad \forall\, d \in D, j \in J_d \tag{A1}$$

$$\sum_{i\in I_j} \sum_{b\in B_i} \mathrm{PT}_{bj} \le \mathrm{MK} - (\mathrm{est}_{j,s}^{\min} + \pi_{j,s}^{\min}) \qquad \forall\, j \in J \tag{A2}$$

where

$$\mathrm{est}_j^{\min} = \min_{i\in I_j}\Big[\sum_{s'<s} \min_{j'\in J_{is'}}[\mathrm{ft}_{ij'} + q_{ij'}^{\min}\,\mathrm{vt}_{ij'}]\Big] \quad \forall\, j \in J_s,\ s \in S \tag{A3}$$

$$\pi_j^{\min} = \min_{i\in I_j}\Big[\sum_{s'>s} \min_{j'\in J_{is'}}[\mathrm{ft}_{ij'} + q_{ij'}^{\min}\,\mathrm{vt}_{ij'}]\Big] \qquad \forall\, j \in J_s,\ s \in S \tag{A4}$$

$$J_d = \{j \in J\colon \exists\ \ i \in I\colon j \in J_i \wedge r_{id} > 0\} \tag{A5}$$

Tightening constraints A1 and A2 can also be applied in the context of the group-based formulation by using $\mathrm{PT}_{bj} = \sum_{k\in K_{id}} \mathrm{PT}_{bkj}$, $\forall\ b \in B_{id}$.

AUTHOR INFORMATION

Corresponding Author

*E-mail: jcerda@intec.unl.edu.ar.

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

Financial support received from FONCYT-ANPCyT under Grant PICT 01837, from CONICET under Grant PIP-2221, and from Universidad Nacional del Litoral under CAI+D 66-335 is fully appreciated.

NOMENCLATURE

Subscripts

b = batch
d = due date
i = product
j = unit
k = cluster or group of batches
s = stage

Sets

B_i = tentative batches of product i
B_{id} = tentative batches associated with order (i, d)
D_i = due dates associated with product i
I = products
J = units
J_i = units available for product i
J_s = units at stage s
J_{is} = units available for product i at stage s
J_{s+1}^j = units in stage $s + 1$ physically connected to unit $j \in J_s$
K_i = clusters proposed for product i
K_{id} = clusters proposed for order (i, d)
S = stages

Parameters

Δ_{ij} = difference between $q_{ij}^{\max}$ and $q_{ij}^{\min}$
α_{id} = weighting penalty for the tardiness of order (i, d)
ε_{id} = lowest inventory of product i available for order (i, d)
$\tau_{ii'j}$ = sequence-dependent changeover time between products i and i' in unit j
bs_i = reference batch size for product i
ft_{ij} = fixed processing time for product i at unit j
ks_i = average number of batches of product i per cluster
H = length of the scheduling horizon
nb_i = maximum number of batches to meet the total demand of product i
nb_{id} = estimation of the number of batches needed to satisfy order (i, d)
r_{id} = requirement of product i at due date d
s^l = last processing stage
$q_{ij}^{\min}$, $q_{ij}^{\max}$ = minimum/maximum batch size for product i at unit j
vt_{ij} = variable processing time rate for product i at unit j

Binary Variables

$X_{b,b',s}$ = denotes that batch b is run before b' $(b' > b)$ at stage s if $X_{b,b',s} = 1$
$\hat{X}_{b,b'}$ = denotes that batch b is run before b' $(b' > b)$ at any stage if $\hat{X}_{b,b'} = 1$

$X^{G}_{k,k',s}$ = denotes that cluster k is processed before k' ($k' > k$) at stage s if $X^{G}_{k,k',s} = 1$
Y_{bj} = denotes that batch b is allocated to unit j
Y_{kj} = denotes that cluster k is allocated to unit j
W_{bk} = allocation of batch b to cluster k

Continuous Variables

BS_{b} = size of batch b
CT_{bs} = completion time of batch b at stage s
CT_{bks} = completion time of batch b in cluster k at stage s
CT^{G}_{ks} = completion time of cluster k at stage s
MK = makespan
PT_{bj} = processing time of batch b allocated to unit j
PT_{bkj} = processing time of batch b in cluster k allocated to unit j
Q_{bj} = size of batch b above minimum when allocated to unit j
Q_{bkj} = size of batch b in cluster k above minimum when allocated to unit j
ST_{bs} = starting time of batch b at stage s
ST_{bks} = starting time of batch b in cluster k at stage s
ST^{G}_{ks} = starting time of cluster k at stage s
T_{id} = tardiness of order (i, d)
V_{bkj} = denotes that batch b in cluster k is processed in unit j
W_{b} = denotes the selection of batch b
Z_{k} = denotes the selection of cluster k

REFERENCES

(1) 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*, 913−946.
(2) Floudas, C. A.; Lin, X. Continuous-Time Versus Discrete-Time Approaches for Scheduling of Chemical Processes: A Review. *Comput. Chem. Eng.* **2004**, *28* (11), 2109−2129.
(3) 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*, 211−227.
(4) Pantelides, C. C. Unified Frameworks for the Optimal Process Planning and Scheduling. In *Proceedings on the Second Conference on Foundations of Computer Aided Operations*; Cache Publications: New York, 1994; pp 253−274.
(5) Ierapetritou, M. G.; Floudas, C. A. Effective Continuous-Time Formulation for Short-Term Scheduling. 1. Multipurpose Batch Processes. *Ind. Eng. Chem. Res.* **1998**, *37*, 4341−4359.
(6) 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*, 3056−3074.
(7) 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*, 105−118.
(8) 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*, 2516−2533.
(9) 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*, 3037−3051.
(10) 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*, 9175−9190.
(11) 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*, 701−711.
(12) Marchetti, P. A.; Cerdá, J. A General Resource-constrained Scheduling Framework for Multistage Batch Facilities with Sequence-dependent Changeovers. *Comput. Chem. Eng.* **2009**, *33* (4), 871−886.
(13) 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*, 1695−1707.
(14) Gupta, S.; Karimi, I. A. An Improved MILP Formulation for Scheduling Multiproduct, Multistage Batch Plants. *Ind. Eng. Chem. Res.* **2003**, *42*, 2365−2380.
(15) Marchetti, P. A.; Cerdá, J. An Approximate Mathematical Framework for Resource-Constrained Multistage Batch Scheduling. *Chem. Eng. Sci.* **2009**, *64* (11), 2733−2748.
(16) Lim, M.; Karimi, I. A. A Slot-Based Formulation for Single-Stage Multi-Product Batch Plants with Multiple Orders per Product. *Ind. Eng. Chem. Res.* **2003**, *42*, 1914−1924.
(17) Prasad, P.; Maravelias, C. T. Batch Selection, Assignment and Sequencing in Multi-Stage Multi-Product Processes. *Comput. Chem. Eng.* **2008**, *32*, 1106−1119.
(18) Sundaramoorthy, A.; Maravelias, C. T. Simultaneous Batching and Scheduling in Multistage Multiproduct Processes. *Ind. Eng. Chem. Res.* **2008**, *47*, 1546−1555.
(19) Marchetti, P. A.; Méndez, C. A.; Cerdá, J. Mixed-Integer Linear Programming Monolithic Formulations for Lot-Sizing and Scheduling of Single-Stage Batch Facilities. *Ind. Eng. Chem. Res.* **2010**, *49*, 6482−6498.
(20) Sundaramoorthy, A.; Maravelias, C. T. Modeling of Storage in Batching and Scheduling of Multistage Processes. *Ind. Eng. Chem. Res.* **2008**, *47*, 6648−6660.
(21) Sundaramoorthy, A.; Maravelias, C. T.; Prasad, P. Scheduling of Multistage Batch Processes Under Utility Constraints. *Ind. Eng. Chem. Res.* **2009**, *48*, 6050−6058.