Chemical Engineering Science 64 (2009) 2733 – 2748

Contents lists available at ScienceDirect

Chemical Engineering Science

journal homepage: www.elsevier.com/locate/ces

An approximate mathematical framework for resource-constrained multistage batch scheduling

Pablo A. Marchetti, Jaime Cerdá*

INTEC (UNL-CONICET), Güemes 3450, 3000 Santa Fe, Argentina

ARTICLE INFO

Article history:
Received 2 September 2008
Received in revised form 24 February 2009
Accepted 3 March 2009
Available online 12 March 2009

Keywords:
Scheduling
Multistage batch plants
Chemical processes
Bottleneck
Mathematical modelling
Optimization

ABSTRACT

A rigorous representation of the multistage batch scheduling problem is often useless to even provide a good feasible schedule for many real-world industrial facilities. In order to derive a much simpler scheduling methodology, some usual features of multistage batch plants should be exploited. A common observation in industry is that multistage processing structures usually present a bottleneck stage (BS) controlling the plant output level. Therefore, the quality of the production schedule heavily depends on the proper allocation and sequencing of the tasks performed at the stage BS. Every other part of the processing sequence should be properly aligned with the selected timetable for the bottleneck tasks. A closely related concept with an empirical basis is the usual existence of a common batch sequencing pattern along the entire processing structure that leads to define the constant-batch-ordering rule (CBOR). According to this rule, a single sequencing variable is sufficient to establish the relative ordering of two batches at every processing stage in which both have been allocated to the same resource item. This work introduces a CBOR-based global precedence formulation for the scheduling of order-driven multistage batch facilities. The proposed MILP approximate problem representation is able to handle sequence-dependent changeovers, delivery due dates and limited manufacturing resources other than equipment units. Optimal or near-optimal solutions to several large-scale examples were found at very competitive CPU times.

1. Introduction

Batch processing plants producing a large variety of chemical compounds by performing a similar sequence of processing stages are very common in industry. Such multiproduct, multistage facilities usually comprise several non-identical equipment units running in parallel at every stage. Manufactured products are chemically similar and product processing times generally follow the same pattern along the stage sequence. The plant operation is usually order-driven and each customer order can involve one or several batches of a given product and has a promised delivery due date. This kind of industrial facilities can be referred to as order-driven multiproduct, multistage batch plants. Since the client service level is the major operational issue, the aim of the scheduling task is to dispatch customer orders on time. To this end, the short-term production schedule is routinely developed on a weekly basis to maximize customer satisfaction while making an efficient utilization of the available manufacturing resources. However, the scheduling of real-world multistage batch facilities is a combinatorial problem demanding a large computational cost that exponentially increases with the number of production orders, processing stages, equipment units, and the extent of the time horizon. The complexity of this scheduling problem rises even more when discrete/continuous resources other than units, like manpower, tools and utilities are available in finite amounts. Despite its industrial significance, limited work has been made on resource-constrained multistage batch scheduling.

Scheduling methodologies can be broadly classified into monolithic or sequential approaches. Monolithic methods tackle not only the scheduling problem but at the same time the batching problem in order to decide the number and size of product batches to be processed. In turn, sequential approaches assume that the set of batches is already known, every lot has the same linear recipe and batch mixing and splitting are forbidden operations. Accounting for their structural and operational features, sequential methodologies look more suitable for order-driven multistage batch plants. However, published results indicate that rigorous sequential approaches are usually unable to find a good feasible solution for real-world multistage batch scheduling problems at reasonable CPU times. Consequently, the development of highly efficient approximate methodologies for resource-constrained multistage batch scheduling appears to be a very promising research area.

A general review on batch scheduling can be found in Floudas and Lin (2004) and Méndez et al. (2006). Using different time

* Corresponding author.
E-mail address: jcerda@intec.unl.edu.ar (J. Cerdá).

0009-2509/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ces.2009.03.002

representations, several model-based sequential approaches for order-driven multistage batch facilities have been proposed. They include continuous-time formulations using the notions of time-slots (Pinto and Grossmann, 1995, 1996), direct precedence (Hui and Gupta, 2000; Hui et al., 2000; Gupta and Karimi, 2003), general precedence (Méndez et al., 2001), and multiple time-grids (Castro and Grossmann, 2005). The approach of Pinto and Grossmann (1995) assumed sequence-independent setup times and considered topology restraints. By using time matching constraints to equal time coordinates of both units and tasks, their formulation was able to solve small multistage batch scheduling problems with up to five single-batch orders and five processing stages. Additional efforts for reducing the computational cost included the use of pre-ordering constraints and a special decomposition scheme to solve large-scale problems. Later, continuous-time sequential approaches relying on the immediate precedence notion were presented by Hui and Gupta (2000) and Hui et al. (2000). They introduced a three-index sequencing variable to denote that two batches are consecutively executed on a given stage. In this way, the resulting MILP formulation was able to easily handle sequence-dependent changeovers and infeasible batch sequences. However, problems with more than five single-batch orders could not be solved to optimality. Méndez et al. (2001) introduced the general precedence concept to develop a continuous-time problem formulation that accounts for sequence-dependent changeovers and presents a much improved computational performance. At each processing stage, only a single 0–1 variable is needed to decide the relative processing order of any two batches allocated to the same equipment unit. As a result, a large saving in binary variables was achieved. Besides, more compact scheduling models can be derived from the original problem formulation by embedding preordering rules for batches allocated to the same unit in a very simple manner. Problems with five single-batch orders and five stages were solved and the results were favorably compared with those reported by Pinto and Grossmann (1995). Afterwards, Gupta and Karimi (2003) presented an extensive study of nine alternative immediate precedence continuous-time models, closely related to formulations previously proposed (Hui and Gupta, 2000; Hui et al., 2000) but requiring fewer binary variables and/or constraints. They solved several examples with up to 22 batches, and the results for the best-performance model were successfully compared with previous contributions. Recently, Castro and Grossmann (2005) introduced a multiple-time-grid continuous-time formulation that efficiently handles order release-times and due dates, and different types of objective functions. However, changeover times were neglected. To make an appropriate comparison with other approaches, the authors also presented reduced versions of a discrete-time and a uniform-time-grid continuous-time model. To this end, resource-task network (RTN) formulations for multipurpose plants (Pantelides, 1994; Castro et al., 2004) were adapted to the multistage case. In addition, results obtained with a general precedence model and a constraint programming (CP) approach (Harjunkoski and Grossmann, 2002) were also considered. By solving 30 examples with up to 40 processing tasks, the results led to conclude that there is not an ideal approach for all types of batch scheduling problems and objective functions. However, formulations with multiple time grids proved to be more convenient than their uniform time grid counterpart.

This work introduces an approximate formulation for the scheduling of resource-constrained multistage batch facilities that relies on the so-called Constant Batch Ordering Rule (CBOR). The resulting MILP mathematical model can be regarded as a compact version of previous global precedence-based formulations (Méndez et al., 2001; Méndez and Cerdá, 2002, 2003). Similar to these approaches, the approximate methodology still uses the notion of global precedence variables to sequence batches on the queue of any resource item. However, the CBO-rule embedded in the problem formulation produces a substantial saving in 0–1 sequencing variables and a significant reduction in CPU time. To compare its computational performance with previous work, our reduced MILP formulation was applied to a large set of order-driven multistage batch scheduling problems. In any case it usually discovers the true optimal solution or at least near-optimal schedules at much lower computational cost, especially for problem instances with limited resources other than equipment items.

This paper is organized as follows. Section 2 introduces the bottleneck resource notion and provides a thorough review of bottleneck-based batch sequencing procedures. Section 3 introduces the fundamentals of the Constant Batch Ordering Rule. Section 4 presents the formal statement of the multistage batch scheduling problem with resource constraints to be tackled, while Section 5 lists the model assumptions. In Section 6, we introduce a CBOR-based mathematical framework for non-constrained multistage batch scheduling problems. The proposed formulation is extended in Section 7 to account for discrete and continuous resource constraints. Section 8 includes the best solutions found for a significant number of benchmark examples and a thorough comparison with previous approaches. The final conclusions are presented in Section 9.

2. The bottleneck resource concept

In order to derive a simpler batch scheduling formulation, some usual features of multistage processing structures can be exploited. A common observation in industry is the fact that most plants have very few operations demanding the bottleneck resource. A bottleneck resource (BR) is one whose capacity Q_{BR} is often lower than the demand placed on it. In other words, the overall capacity requirement on BR is usually a high fraction of Q_{BR} and even exceeds its capacity during some time intervals. Since the BR determines the plant throughput, it should be working all the time and a buffer inventory is kept in front of it to make sure that it never remains idle. Therefore, the quality of the plant schedule strongly depends on the proper unit allocation and sequencing of bottleneck operations. On the contrary, a non-bottleneck resource (NBR) has a capacity greater than its workload and does not work constantly. Idle time is then a common feature of NBRs and the main reason for the existence of alternative schedules with a similar performance. Consequently, changes in the schedule of non-bottleneck tasks has a minor impact and merely produce slight variations on the production performance of batch facilities. On the other hand, a capacity-constrained resource (CCR) is one whose utilization is close to its maximum capacity and may become a production bottleneck if not carefully scheduled (Chase et al., 1998).

By definition, the aim of the scheduling function is to optimally allocate resources to processing tasks over time. As mentioned, not all resources but just those ones constraining the production flow through the processing sequence really deserve especial attention. The critical operations that should be carefully scheduled are those requiring the BR, i.e. the bottleneck operations. Every other part of the processing sequence should be properly aligned so that the right amount of material required by bottleneck tasks timely arrives. Consequently, a fundamental idea in multistage batch scheduling is that the equipment units at the bottleneck stage (BS), also referred to as the bottleneck work center, are the ones determining the plant performance. Such equipment items are the BRs and the critical tasks are those carried out at the BS.

2.1. A review of bottleneck-based sequencing procedures

This strategy of focusing the scheduling effort on the critical constraints was first explored by Goldratt and Cox (1986) through the

so-called Optimized Production Technology (OPT) and implemented in different OPT-software packages commercially available. They were convinced that some basic principles could be exploited to come up with a better scheduling system. An OPT-package performs the following four basic steps: (i) identify the bottleneck resource; (ii) schedule first the critical operations for the most effective usage of BRs; (iii) schedule the upstream stages from the bottleneck so as to meet the scheduled starting times of bottleneck operations; and (iv) schedule the downstream stages assuming that the end times of bottleneck operations are given (Chase et al., 1998). Therefore, the batch sequencing at the BS is supposed to control the batch ordering at the other upstream and downstream stages. As non-bottleneck resources have idle capacities, the OPT-algorithm assumes that feasible schedules for upstream and downstream stages satisfying the proposed BS timetable can always be found.

The notion of bottleneck resource was later applied to generate near-optimal schedules at lower cost in many other publications. Adams et al. (1988) developed the so-called shifting bottleneck procedure (SBP) based on a disjunctive graph representation to heuristically solve the standard job-shop scheduling problem. In this problem, there is a single machine for each operation, and the job-dependent task sequence is given. Then, only sequencing decisions should be made. The problem goal is the minimization of the makespan, i.e. the time needed for processing all jobs. Sequence-dependent setup times and limited resources other than machines were not considered. The SBP establishes the job-sequence on every machine not yet sequenced one-by-one to identify the bottleneck machine and the best sequence on that machine. After that, all previous established sequences are locally reoptimized taking into account the new partial schedule. The bottleneck identification and the local reoptimization procedures are both based on repeatedly solving one-machine scheduling (OMS) problems. The OMS problem is applied to each machine not yet sequenced to determine the makespan. The one yielding the maximum makespan is regarded as the bottleneck machine. To determine the makespan, the OMS formulation includes estimated values of the head/tail times for each job. The head time goes from the start of the schedule to the initial time of the job, while the tail time goes from the finishing time to the end of the schedule.

Dauzere-Peres and Lasserre (1993) proposed a modified version of SBP to not only considering some precedence constraints between jobs but also reducing the sensitivity of SBP to the number of local reoptimization cycles. Ivens and Lambrecht (1996) extended the disjunctive graph representation and the SBP to solve non-standard job-shop scheduling problems. The adapted SBP relies on an extended disjunctive graph (EDG) and has been applied to real-life problems involving product assemblies and splits, overlapping operations, parallel machines, sequence-independent setup times and transit times. However, it is required to solve parallel machine problems instead of one-machine problems for identifying the bottleneck processing stage. Scarce resources different from machines were not explicitly considered. Balas and Vazacopoulos (1998) developed a guided local search (GLS) procedure using interchange of operations and the neighborhood tree concept, and embedded it into a shifting bottleneck framework to generate a very efficient hybrid procedure for job-shop scheduling. In turn, Maravelias (2006) proposed an assignment-sequencing decomposition framework for the scheduling of multistage batch plants. The assignment subproblem is solved by applying mixed-integer methods, while the sequencing problem is tackled using the hybrid SBP (Balas and Vazacopoulos, 1998). In the first step of SBP, it should be checked whether a feasible schedule can be obtained using the task-unit assignments found through the allocation subproblem. If not, a strong integer cut forbidding the current assignments should be added and the procedure is repeated until a feasible set of assignments is generated.

2.2. Common features of bottleneck-based sequencing procedures

The major common features of the reviewed bottleneck procedures can be summarized as follows:

(a) They just deal with "pure" sequencing subproblems by assuming known, feasible order-unit assignments.
(b) They need to identify the BR because are based on one-resource sequencing (ORS) problems.
(c) They usually assume sequence-independent setup times.
(d) They do not account for scarce resources, like manpower or steam, shared by multiple processing stages whose partial schedules are therefore coupled. When using SBP procedures, the interdependency of ORS problems is likely to produce infeasible partial schedules.

3. The Constant Batch Ordering Rule (CBOR)

The supporting idea of bottleneck-based sequencing procedures is that the best ordering of processing tasks at the bottleneck stage mostly determines the whole production schedule, and therefore the associated sequencing decisions stand for the problem critical variables. Any pair of batches (i,i') allocated to the same equipment unit at some non-bottleneck stage will be arranged as required by the BS schedule. This explains why a characteristic sequencing pattern frequently observed in optimal short-term schedules of multiproduct batch facilities with multiple stages is a constant ordering of batches (i,i') in any processing stage where batches (i,i') have been allocated to the same equipment unit. In other words, if batches (i,i') share the same processing unit at stages s and s' and batch i is processed before batch i' in stage s, then the same sequencing pattern (i,i') usually holds at the later stage s'. In this way, the notion of BR leads to establishing a very simple principle, i.e. the so-called constant-batch-ordering pattern. In contrast to the bottleneck-based procedures, the constant-batch-ordering rule can be applied without knowing the location of *BS*. It simply assumes the existence of a bottleneck resource that controls the relative ordering of any pair of batches throughout the processing sequence. In Fig. 1, batches (i,i') have both been assigned to unit U2 in stage s and to unit U3 in the next stage s'. Moreover, batch i is processed before in unit U2. According to the CBO rule, the batch sequence (i,i') is repeated in unit U3 (see Fig. 1). In case tasks (i,BS) and (i',BS) are performed in two different units $j,j' \in J_{BS}$ at the bottleneck stage, the CBOR assumes that the role of the bottleneck stage *BS* is taken by the next capacity-constrained stage (CCS) where batches (i,i') share the same equipment unit. Moreover, the BR may change with the product mix but there is always at least one controlling the plant schedule.

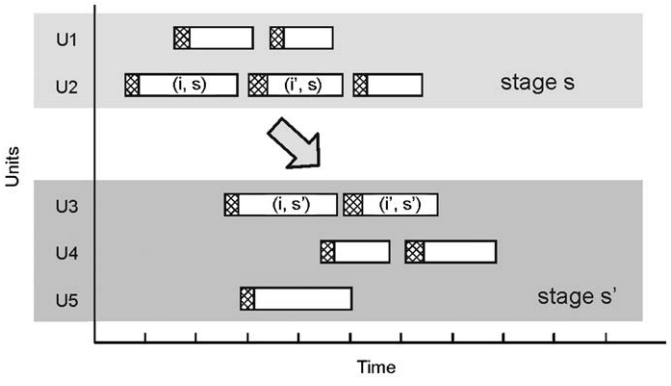

Fig. 1. Illustrating the constant-batch-ordering rule.

A detailed analysis of the optimal solutions found for a myriad of multistage batch scheduling problems reported in the literature (Pinto and Grossmann, 1995; Hui et al., 2000; Méndez et al., 2001; Gupta and Karimi, 2003; Castro and Grossmann, 2005) shows that the CBOR generally applies. Testing the CBO Rule can only be done when a detailed Gantt chart describing the best schedule has been published. The CBOR leads to a much simpler batch scheduling formulation with a unique sequencing variable $X_{ii'}$ establishing the relative ordering of any pair of batches (i,i') throughout the whole multistage processing system rather than a different one $X_{ii's}$ for each stage s. Since batches are sequenced only if allocated to the same unit, the value of variable $X_{ii'}$ is only relevant for stages where batches (i,i') actually share a processing unit, otherwise it is meaningless. In the proposed CBOR-based formulation,

$$X_{ii'} = \begin{cases} 1 & \text{If batch } i \text{ is executed before batch } i' \text{ at any stage } s \text{ where both share the same equipment unit} \\ 0 & \text{If batch } i \text{ is executed after batch } i' \text{ at any stage } s \text{ where both share the same equipment unit} \end{cases}$$

In case the batch process comprises five stages and every product can be processed in any equipment unit, the number of sequencing variables can be reduced five times through using the CBOR-based scheduling approach.

Since bottleneck resources other than equipment units may arise, the CBO-rule should be generalized. Let us assume that a discrete or continuous resource $r \in R$ different from equipment is required at some processing stages, and every stage s has its own set of resources R_s. Let us also suppose that a pair of batches (i,i') has been allocated to the same unit $j \in J_s$ in stage s and batch i is processed before. Then, the resource r will also be earlier allocated to batch i in stage s. In other words, there is a common ordering of batches (i,i') in the queue of any shared resource of stage s. Therefore, a unique sequencing variable $X_{ii'}$ is just needed to establish the relative ordering of a pair of batches (i,i') in the queue of any resource item through the entire processing structure. As a result, the number of sequencing variables is substantially reduced and the computational cost required to find near-optimal schedules is diminished by orders of magnitude. Though few detailed results on resource-constrained scheduling of linear batch processes have been published, it can be expected that the CBO-rule also yields good feasible solutions for the resource-constrained case. Usually, the requirement of a limited resource other than equipment has been confined to a single processing stage in the examples solved in the literature. However, a set of renewable resources could sometimes be shared by different processing stages. A further generalization of the CBOR-based multistage batch scheduling formulation to account for limiting resources shared by several stages is discussed in Section 7.

The constant-batch-ordering rule is proposed for the scheduling of multiproduct, multistage batch plants with multiple parallel units running at every stage. The CBOR-based scheduling formulation performs much better and usually provides the optimal schedule if: (a) the manufactured products are chemically similar and (b) the processing times mostly follow the same change pattern along the common task sequence, i.e. those difficult tasks requiring larger processing times are mostly the same for all products. Even if the processing time change pattern along the task sequence varies with the product, the method still finds good, feasible schedules. In contrast to shifting bottleneck procedures, the constant-batch-ordering rule-based scheduling approach can simultaneously handle allocation and sequencing decisions, sequence-dependent setup times and limited resources other than equipment required at one or several processing stages. Instead of sequencing machines one-by-one, all of them are sequenced at the same time. This is so because the proposed methodology does not need to explicitly identify the BS.

4. Problem statement

Given:

(i) a multiproduct multistage batch plant with several units $j \in J_s$ running in parallel at every stage $s \in S$,
(ii) the set of batches $i \in I$ to be processed following the same stage sequence, and their release times rt_i and delivery due-dates dd_i,
(iii) the available equipment units $j \in J_s$ at each stage $s \in S$ and their ready times ru_j,
(iv) the subset of units $j \in J_{is} \subseteq J_s$ that can be assigned to the stage s of batch i, i.e. the task (i,s),
(v) the plant topology structure specifying the interconnections between units belonging to consecutive stages,
(vi) the batch processing times pt_{ij} and the sequence-dependent setup times, given as the sum of two components su_{ij} and $\tau_{ii'j}$,
(vii) a set of renewable resource types $r \in R$ different from equipment comprising a subset of unary resources R^{α} and a subset of finite resources R^{β}, such that $R = R^{\alpha} \cup R^{\beta}$,
(viii) the set of resource items $z \in Z_r$ for each unary resource type $r \in R^{\alpha}$, and their capacities q_z,
(ix) the available capacity Q_r of each finite renewable resource $r \in R^{\beta}$,
(x) the subset of resource types $R_{is} \subset R$ required by task (i,s),
(xi) the amount ρ_{isr} of resource type $r \in R_{is}$ (or ρ_{isjr} if the resource requirement is unit-dependent) needed for the processing of task (i,s), and
(xii) the length of the time horizon H.

The problem goal is to determine: (a) the allocation of batches to equipment units and other resource items, (b) the sequence of tasks at each resource unit, and (c) the starting and completion times of every batch at each stage, so that all problem constraints are satisfied and, at the same time, a given performance criterion (e.g. overall tardiness, earliness or makespan) is optimized.

5. Model assumptions

To develop an approximate mathematical framework for the short-term scheduling of multistage multiproduct batch plants with resource constraints, the following assumptions have been made:

1. Model parameters are all deterministic data.
2. Once the processing of a task starts in a given unit, it should be carried out until completion without interruption.
3. Unlimited intermediate storage to receive in-process material from each stage is available.
4. Changeover times are sequence-dependent.
5. Batch transfer times are negligible.
6. Processed batches all meet quality specifications.
7. Batch mixing and splitting are not allowed.
8. The amount of resource needed by a processing task is used over the entire task length and released at the task end.
9. The relative ordering of batches throughout the processing structure always satisfies the CBOR.

6. The CBOR-based multistage batch scheduling formulation

6.1. Unit allocation constraints

Every batch i must be assigned to a single equipment unit $j \in J_{is}$ at every stage s. Since each unit belongs to a unique processing stage,

the stage index s can be deleted from the domain of variable Y_{ij}:

$$\sum_{j\in J_{is}} Y_{ij} = 1 \quad \forall i \in I,\ s \in S \tag{1}$$

6.2. Topological constraints

Often, multistage batch plants present topological constraints that limit the number of possible batch routes throughout the processing structure. Assuming that task (i,s) is performed in unit $j \in J_{is}$ (i.e. $Y_{ij} = 1$), the aim of constraint (2) is to guarantee that the unit allocated to task $(i,s{+}1)$ not only belongs to the set of eligible units $J_{i,s+1}$, but is also accessible from unit $j \in J_{is}$. In constraint (2), $J_{s+1}^{(j)}$ stands for the subset of equipment items available at stage $s{+}1$ that are physically connected to unit $j \in J_{is}$. Moreover, s^{ℓ} represents the last stage:

$$Y_{ij} \leqslant \sum_{j'\in(J_{i,s+1}\cap J_{s+1}^{(j)})} Y_{ij'} \quad \forall i \in I,\ s \in S - \{s^{\ell}\},\ j \in J_{is} \tag{2}$$

6.3. Batch timing constraints

Constraint (3) defines the relationship between the starting time (S_{is}) and the completion time (C_{is}) of task (i,s) in terms of the processing time required on the assigned unit:

$$S_{is} = C_{is} - \sum_{j\in J_{is}} pt_{ij} Y_{ij} \quad \forall i \in I,\ s \in S \tag{3}$$

Additional bounds on the starting and the completion time of every batch i are given by constraints (4)–(6). A lower bound on the starting time at the first stage s^f is set up not only by its release time rt_i but also by the sum of the ready time and the setup time of the allocated unit. Thus, constraint (4) assumes that all materials required to process batch i are not available until time rt_i. It is straightforward to account for the more general case where every task (i,s) has its own release time (i.e. rt_{is}).

$$S_{is} \geqslant \sum_{j\in J_{is}} \max[rt_i, ru_j + su_{ij}] Y_{ij} \quad \forall i \in I,\ s = s^f \tag{4}$$

The relationship between the completion time and the starting time of a batch at a pair of consecutive stages $(s, s{+}1)$ is defined by the technological constraint:

$$C_{is} \leqslant S_{i,s+1} \quad \forall i \in I,\ s \in S - \{s^{\ell}\} \tag{5}$$

Besides, constraint (6) states that the completion time of the last stage of batch i must not be later than its delivery due-date dd_i. The inclusion of this constraint in the problem formulation depends on the objective function being adopted:

$$C_{is} \leqslant dd_i \quad \forall i \in I,\ s = s^{\ell} \tag{6}$$

6.4. Sequencing constraints

Let us assume that batches (i,i') have been allocated to the same unit $j \in J_{ii's}$ at some processing stage s. Then, a pair of sequencing constraints are needed to ensure that tasks (i,s) and (i',s) are performed once at a time. By using the global precedence concept introduced by Méndez et al. (2001), a single 0–1 three-index sequencing variable $X_{ii's}$ would be enough to control the relative ordering of batches (i,i') at stage s. Moreover, $X_{ii's}$ will be equal to 0 or 1 depending on whether task (i,s) is queued after or before task (i',s) at unit $j \in J_{ii's}$, respectively. By assuming a constant batch ordering throughout the processing structure, however, the stage-index s can be withdrawn from the domain of X's and the sequencing variables will be simply given by $X_{ii'}$. Therefore, the batch sequencing constraints (7a,b) can be written as follows:

$$C_{is} + \tau_{ii'j} + su_{i'j} \leqslant S_{i's} + H(1 - X_{ii'}) + H(2 - Y_{ij} - Y_{i'j})$$
$$\forall i,i' \in I,\ s \in S,\ j \in (J_{is} \cap J_{i's}) : (i < i') \tag{7a}$$

$$C_{i's} + \tau_{i'ij} + su_{ij} \leqslant S_{is} + HX_{ii'} + H(2 - Y_{ij} - Y_{i'j})$$
$$\forall i,i' \in I, s \in S, j \in (J_{is} \cap J_{i's}) : (i < i') \tag{7b}$$

Let us call $\hat{S}_{ii'} \subseteq S$ the subset of processing stages at which batches (i,i') have been allocated to the same unit j, i.e. $Y_{ij} + Y_{i'j} = 2$. Since a single variable $X_{ii'}$ determines the relative ordering of batches (i,i'), then batch i will be permanently processed before ($X_{ii'} = 1$) or after batch i' ($X_{ii'} = 0$) at any stage $s \in \hat{S}_{ii'}$. In turn, the value of $X_{ii'}$ will be meaningless for any stage $s \notin \hat{S}_{ii'}$. Therefore, a single variable $X_{ii'}$ can be defined to determine the relative ordering of batches (i,i') at every stage $s \in S$. In this way, the number of sequencing variables is reduced by a factor equivalent to the number of processing stages. Constraint (7a) introduces the required changeover time ($\tau_{ii'j} + su_{i'j}$) after the completion of task (i,s) and before starting (i',s). Similarly, constraint (7b) considers the changeover time between the completion of batch i' and the start of batch i at stage s.

6.5. Objective functions

If the selected problem goal is to minimize the overall batch tardiness, the mathematical formulation should include the constraints (8) and (9). The tardiness of batch i (T_i) is obtained from its delivery due date dd_i and the completion time of the last processing stage. A batch-dependent weight coefficient ε_i can be used to penalize a tardy batch i:

$$C_{is} - dd_i \leqslant T_i \quad \forall i \in I,\ s = s^{\ell} \tag{8}$$

$$\text{Minimize} \quad \sum_{i\in I} \varepsilon_i T_i \tag{9}$$

When the minimization of the makespan MK is the problem goal, the largest batch completion time will be the lower bound for the objective function MK, as indicated by:

$$MK \geqslant C_{is} \quad \forall i \in I,\ s = s^{\ell} \tag{10}$$

$$\text{Minimize} \quad MK \tag{11}$$

7. Extending the approach to account for other types of resources

If manufacturing resources other than equipment units are also required at some stages, then two different cases can arise depending on whether: (1) each processing stage s has its own set of resource items $z \in Z_{rs}$ of type $r \in R$ or (2) several processing stages share a common set of resource items $z \in Z_r$ of type r.

7.1. Case 1: Each stage has its own set of resource items

As previously discussed in Section 3, the CBOR can be easily extended to Case 1 because the sequencing variable $X_{ii'}$ controls the relative ordering of tasks (i,s) and (i',s) in the queue of any shared resource item at stage s. For instance, let us assume that a discrete resource of type r is needed at stage s. If the pair of batches (i,i') has been allocated to both the unit $j \in J_{ii's}$ and the resource item $z \in Z_{rs}$ of type r at stage s, and, in addition, batch i is processed before, then batch i should also be first assigned to resource item z. If batches (i,i') only share a non-equipment resource item $z \in Z_{rs}$ of type r at stage s, then $X_{ii'}$ just determines their relative ordering in the queue of resource item z. In case the resource r is largely needed at stage

s, then such a stage could become the BS controlling the batch sequence throughout the processing structure. In this way, a particular stage can become a *BS* because of a BR different from equipment. Moreover, the CBOR still holds and a unique sequencing variable $X_{ii'}$ for the whole processing structure will be enough to establish the sequencing of batches (i,i') at any stage where they both share at least a resource item. Although resources different from equipment units have been considered, no additional sequencing variables are needed and the same set of variables $X_{ii'}$ defined in Section 6 remains valid for the resource-constrained case. As a result, an important saving in binary variables is achieved. However, further 0–1 allocation variables and constraints assigning tasks to non-equipment resource items must be included. They will be presented in Section 7.3 after introducing Case 2 because the same sets of assignment variables and constraints can be used for Cases 1 and 2.

7.2. Case 2: A common set of resource items is shared by several processing stages

On the other hand, Case 2 assumes that a resource of type $r \in R$ highly required at stage s is shared with other processing stages. Suppose that resource r makes a particular stage the process bottleneck *BS* and, consequently, the tasks performed at stage *BS* turn to be the critical ones. Therefore, the associated sequencing decisions become the problem critical variables $X_{ii',BS}$ (represented as $X_{ii'}$) mostly determining the whole production schedule. Similar to Case 1, such critical sequencing variables $X_{ii'}$ permit to also establish the relative ordering of tasks (i,i') in the queue of any dedicated resource shared by (i,i'). By definition, a dedicated resource item is one that can only be assigned to tasks performed at the stage to which it belongs. Therefore, the sequencing variables $X_{ii'}$ are able to synchronize the use of dedicated resource items by processing tasks at the corresponding stage. In contrast, bottleneck and even NBRs shared by several processing stages require a special treatment. Tasks performed at non-bottleneck stages requiring resource r are not synchronized with critical operations at the BS also demanding r through the set of variables $X_{ii'}$. Thus, additional sequencing variables and constraints are needed to control the ordering of tasks competing for the same BR/NBR but performed at different processing stages. Even if the CBOR would still hold, the approximate problem formulation should include further binary variables and constraints to coordinate the timetables of tasks demanding the same utility resource at different stages. As a result, the size of the MILP formulation grows but the increase in the model size remains much lower than in previous approaches (Méndez and Cerdá, 2003).

7.3. Allocation constraints for resources other than equipment units

Given the set of resource types $R = R^{\alpha} \cup R^{\beta}$, the proposed formulation should be able to handle every type $r \in R$. Unary discrete resources are included in the set R^{α} while finite resources will belong to the set R^{β}. To use a common treatment for discrete/continuous finite resources belonging to R^{β}, they will be divided into a discrete number of elementary resource items with unknown capacities to be optimized by the model. The maximum number of elements of type r is a model parameter usually estimated through the largest number of parallel jobs requiring resource r. Then, each resource element can be consecutively assigned to processing tasks but once at a time. If utilities like steam, electricity or cooling water are required, each resource item will represent a portion of the corresponding renewable resource. Similarly, finite discrete resources like a pool of manpower can be grouped into a relatively small number of crews and each one can be independently allocated to tasks one by one. After discretizing resources of type R^{β}, both sets R^{α} and R^{β} can be handled in a similar manner. Therefore, allocation constraints for every resource item $z \in Z_r$ of any type $r \in R$ can be written. The capacity of the resource element z can be either known beforehand (for unary resources) or a model variable (for finite discrete/continuous resources). The 0–1 assignment variable W_{isz} denotes that the resource item z has been assigned to task (i, s) whenever it is equal to one.

7.3.1. Allocation of unary resource items to processing tasks

In order to fulfill the resource requirement ρ_{isjr} of type $r \in R^{\alpha}$ demanded by task (i,s), several unary resource items $z \in Z_r$ of known capacities q_z $(=1)$ should be allocated to (i,s). Constraint (12) assumes that the resource requirement can vary with the equipment unit assigned to task (i,s). It can be regarded as a generalization of the allocation constraints proposed by Méndez and Cerdá (2003) to handle unary resource requirements in a multistage batch plant.

$$\sum_{j \in J_{is}} \rho_{isjr} Y_{ij} = \sum_{z \in Z_r} q_z W_{isz} \quad \forall i \in I,\ s \in S,\ r \in R^{\alpha}_{is} \tag{12}$$

Constraint (12) can be simplified if the requirement of resource r by task (i,s) is not unit-dependent. In such a case, the LHS of constraint (12) should be simply replaced by ρ_{isr}.

7.3.2. Allocation of finite discrete/continuous resources

To handle finite resources similarly to unary resources, each resource $r \in R^{\beta}$ is implicitly partitioned into a finite number of elements $z \in Z_r$ such that every element will be sequentially allocated to several demanding tasks. Let us assume that the total capacity of the finite resource $r \in R^{\beta}$ is Q_r. Therefore, the overall requirement of resource r by processing tasks simultaneously processed in different equipment units must never exceed Q_r at any moment. In contrast to unary resources, the capacity of each resource item is now a variable ϕ_z to be determined by the optimization process. According to Eq. (13), the sum of the resource element capacities ϕ_z should be equal to the total capacity Q_r. When finite discrete resources are considered in R^{β}, the elements included in the set Z_r represent clusters or groups of unary resources. For instance, the manpower pool may be divided into several operator crews of different sizes, with each one being handled as an individual resource item whose capacity is equal to the number of workers in the crew. Then,

$$\sum_{z \in Z_r} \phi_z = Q_r \quad \forall r \in R^{\beta} \tag{13}$$

Let us define the continuous positive variable β_{isz} to denote the amount of resource of type r provided by the resource element $z \in Z_r$ to task (i,s). Constraints (14) impose a pair of upper bounds on the value of β_{isz} by considering the element capacity ϕ_z and the allocation condition given by binary W_{isz}.

$$\beta_{isz} \leqslant \phi_z, \quad \beta_{isz} \leqslant Q_r W_{isz}$$
$$\forall i \in I,\ s \in S,\ r \in R^{\beta}_{is},\ z \in Z_r \tag{14}$$

Finally, constraint (15) indicates that the unit-dependent demand of a resource type $r \in R^{\beta}$ (ρ_{isjr}) should be fulfilled by one or several resource items $z \in Z_r$ allocated to task (i, s):

$$\sum_{j \in J_{is}} \rho_{isjr} Y_{ij} = \sum_{z \in Z_r} \beta_{isz} \quad \forall i \in I,\ s \in S,\ r \in R^{\beta}_{is} \tag{15}$$

7.4. General sequencing constraints for cases 1 and 2

At processing stage s, a pair of batches i and i' requiring the same resource of type $r \in R$ can eventually share one or several resource

Table 1
Ten instances of Example 1 grouped into three problem sets.

Problem features	Instances of Example 1		
	Problem set #1	Problem set #2	Problem set #3
Processing times	See Table 2	See Table 2	See Table 2
Changeover times	See Table 3 (sequence-dependent)	See Table 3 (sequence-dependent)	See Table 4 (unit-dependent)
Due dates	See Table 5	See Table 5	$dd_i = 500 \ \forall\, i \in I$
Objective function	Eq. (18)	Eq. (10) (tardiness)	Eq. (18)
Weighting coefficients	$N_i = 10\,000 \ \forall\, i \in I$	$\varepsilon_i = 1 \ \forall\, i \in I$	$N_i = 10\,000 \ \forall\, i \in I$
Problem sizes	5, 8, 10, 12 batches	10, 12, 17, 22 batches	5, 8 batches

Table 2
Batch processing times for every instance of Example 1.

Unit j	Batch i																					
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	18.1	23	18.1	20	17	15	31	12	13	12	8	10	18.1	23	18.1	20	17	15	31	12	13	12
2	18.1	23	18.1	20	17	14	30	12	7	4	9	7	18.1	23	18.1	20	17	14	30	12	7	4
3	18.1	23	18.1	20	17	13	34	14	8	23	13	14	18.1	23	18.1	20	17	13	34	14	8	23
4	18.1	23	18.1	20	17	12	32	15	9	12	–	–	18.1	23	18.1	20	17	12	32	15	9	12
5	18.1	23	18.1	20	17	18	31	–	–	–	13	–	18.1	23	18.1	20	17	18	31	–	–	–
6	18.1	23	18.1	20	17	15	–	16	16	14	–	13	18.1	23	18.1	20	17	15	–	16	16	14
7	14	14	14	11	14	15	31	–	15	13	13	13	14	14	14	11	14	15	31	–	15	13
8	5	5	5	5	5	7	31	16	15	13	35	23	5	5	5	5	5	7	31	16	15	13
9	5	5	5	5	5	7	31	15	11	13	23	13	5	5	5	5	5	7	31	15	11	13
10	12	12	24	12	12	13	14	12	13	12	8	10	12	12	24	12	12	13	14	12	13	12
11	12	12	24	12	12	12	15	13	7	4	9	7	12	12	24	12	12	12	15	13	7	4
12	12	12	24	12	12	15	16	14	8	23	13	14	12	12	24	12	12	15	16	14	8	23
13	12	12	24	12	12	17	41	14	9	12	–	–	12	12	24	12	12	17	41	14	9	12
14	12	12	24	12	12	17	15	14	–	–	13	–	12	12	24	12	12	17	15	14	–	–
15	12	12	24	12	12	18	81	14	16	14	–	13	12	12	24	12	12	18	81	14	16	14
16	12	12	24	12	12	19	–	14	15	13	13	13	12	12	24	12	12	19	–	14	15	13
17	12	12	24	12	12	–	16	14	15	13	35	23	12	12	24	12	12	–	16	14	15	13
18	–	12	–	–	–	16	16	14	11	13	23	13	–	12	–	–	–	16	16	14	11	13
19	–	12	–	–	–	13	21	10	12	6	8	10	–	12	–	–	–	13	21	10	12	6
20	9.5	–	9.3	7.9	12.5	13.5	12	10	15	–	5	–	9.5	–	9.3	7.9	12.5	13.5	12	10	15	–
21	9.5	–	9.3	7.9	12.5	14	13	9	17	12	12	12	9.5	–	9.3	7.9	12.5	14	13	9	17	12
22	–	100	–	–	–	14.5	11	8	17	23	23	23	–	10	–	–	–	14.5	11	8	17	23
23	24	–	24	24	24	12	11	–	22	12	12	12	24	–	24	24	24	12	11	–	22	12
24	24	–	24	24	24	12	11	7	21	22	22	21	24	–	24	24	24	12	11	7	21	22
25	–	48	–	–	–	23	11	7	–	12	12	16	–	10	–	–	–	23	11	7	–	12

elements $z \in Z_r$. Constraints (16) define the general sequencing constraints for every pair of tasks performed at the same stage and for every possible resource item that can be required by both ones. Allocation variables W_{isz} and $W_{i'sz}$ are used to detect if tasks (i,s) and (i',s) share the resource element z at stage s. Since the unique sequencing binary $X_{ii'}$ determines the relative order of tasks (i,s) and (i',s) in the queue of any resource item z, then the sequencing constraints can be derived from Eqs. (7a,b) by simply replacing Y_{isj} by W_{isz}.

$$C_{is} \leqslant S_{i's} + H(1 - X_{ii'}) + H(2 - W_{isz} - W_{i'sz})$$
$$\forall i, i' \in I, \ \ s \in S, \ \ r \in (R_{is} \cap R_{i's}), \ \ z \in Z_r : (i < i') \qquad (16a)$$

$$C_{i's} \leqslant S_{is} + HX_{ii'} + H(2 - W_{isz} - W_{i'sz})$$
$$\forall i, i' \in I, \ \ s \in S, \ \ r \in (R_{is} \cap R_{i's}), \ \ z \in Z_r : (i < i') \qquad (16b)$$

If tasks (i,s) and (i',s') share the same resource item but are performed at different processing stages, further sequencing variables and constraints must be added. Case 2 requires to define the additional binary variable $X_{is,i's'}$ $(i < i')$ to sequence tasks (i,s) and (i',s') carried out at the different stages s and s' (see Méndez and Cerdá, 2003). General expressions for the additional sequencing constraints are given below:

$$C_{is} \leqslant S_{i's'} + H(1 - X_{is,i's'}) + H(2 - W_{isz} - W_{i's'z})$$
$$\forall i, i' \in I, \ \ s, s' \in S, \ \ r \in (R_{is} \cap R_{i's'}),$$
$$z \in Z_r : (i < i') \wedge (s \neq s') \qquad (17a)$$

$$C_{i's'} \leqslant S_{is} + HX_{is,i's'} + H(2 - W_{isz} - W_{i's'z})$$
$$\forall i, i' \in I, \ \ s, s' \in S, \ \ r \in (R_{is} \cap R_{i's'}),$$
$$z \in Z_r : (i < i') \wedge (s \neq s') \qquad (17b)$$

8. Results and discussion

In order to compare the performance of the new approach with previous scheduling methodologies, four examples have been tackled. The comparison was made based on both the solution quality (in terms of overall earliness/tardiness or makespan) and the required CPU time. Examples 1 and 2 consider the short-term scheduling of a multistage batch facility without resource constraints while Examples 3 and 4 also include limitations in manpower or steam flow. The scarce resource is either confined to a single stage (Example 3) or shared by a pair of stages (Example 4). All examples have been solved using ILOG OPL Studio 3.7 (CPLEX 9.0 mixed-integer optimizer) on a Pentium IV PC (1.8 GHz) with 1 GB of memory.

8.1. Example 1

Example 1 was taken from Gupta and Karimi (2003). It involves the scheduling of at most 22 single-batch production orders to be processed in a multiproduct, multistage batch plant with 25 equipment units distributed among five consecutive stages as follows:

Table 3
Sequence-dependent changeovers for problem sets 1 and 2 of Example 1.

Batch i	Batch i'																					
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1		7	8	9	10	5	7	8	9	7	8	9	5	7	8	9	7	5	7	8	9	7
2	7		6	5	8	4	5	6	5	4	7	8	4	5	6	5	4	4	5	6	5	4
3	12	8		7	6	4	5	7	6	5	6	6	4	5	7	6	5	4	5	7	6	5
4	11	9	5		2	8	6	6	7	7	6	7	8	6	6	7	7	8	6	6	7	7
5	**12**[a]	4	4	6		2	1	6	5	8	7	8	2	1	6	5	8	2	1	6	5	8
6	7	2	5	6	7		9	7	8	7	7	8	7	2	5	6	7	7	9	7	8	7
7	**13**[b]	3	4	5	6	6		8	4	4	5	**15**[b]	**12**[b]	3	4	5	6	6	4	8	4	4
8	8	7	8	9	4	5	5		8	9	8	6	8	7	8	9	4	5	5	5	8	9
9	4	5	6	7	8	9	6	**16**[b]		7	7	8	4	5	6	7	8	9	6	**16**[b]	6	7
10	5	6	7	8	9	4	5	**17**[b]	8		8	9	5	6	7	8	9	4	5	**17**[b]	8	5
11	5	6	6	7	6	7	8	8	9	8		8	5	6	6	7	6	7	8	8	9	8
12	4	5	6	7	8	7	5	6	7	7	8		4	5	6	7	8	7	5	6	7	7
13	5	7	8	9	10	5	7	8	9	7	8	9		7	8	9	7	5	7	8	9	7
14	7	5	6	5	8	4	5	6	5	4	7	8	4		6	5	4	4	5	6	5	4
15	12	8	1	5	6	4	5	7	6	5	6	6	4	5		6	5	4	5	7	6	5
16	11	9	5	5	2	8	6	6	7	7	6	7	8	6	6		7	8	6	6	7	7
17	**12**[a]	4	4	6	3	2	1	6	5	8	7	8	2	1	6	5		2	1	6	5	8
18	7	2	5	6	7	5	9	7	8	7	7	8	7	2	5	6	7		9	7	8	7
19	**13**[b]	3	4	5	6	6	5	8	**13**[a]	4	4	5	**12**[b]	3	4	5	6	6		8	4	4
20	8	7	8	9	4	5	5	5	8	9	8	6	8	7	8	9	4	5	5		8	9
21	4	5	6	7	8	9	6	**16**[b]	5	7	7	8	4	5	6	7	8	9	6	**16**[b]		7
22	5	6	7	8	9	4	5	**17**[b]	8	5	8	9	5	6	7	8	9	4	5	**17**[b]	8	

[a]Modified value from original problem data.
[b]Infeasible batch sequence.

U_1–U_6 in stage 1, U_7–U_9 in stage 2, U_{10}–U_{19} in stage 3, U_{20}–U_{22} in stage 4, and U_{23}–U_{25} in the last stage 5. Reduced versions of Example 1 were first studied by Pinto and Grossmann (1995), Hui and Gupta (2000), and Hui et al. (2000). Pinto and Grossmann (1995) considered the schedule of five production orders assuming sequence independent changeovers, while Hui and Gupta (2000) and Hui et al. (2000) accounted for 12 single-batch orders and sequence-dependent changeovers. In the more recent version of Example 1 tackled in this paper (Gupta and Karimi, 2003), problems with up to 22 batches were considered.

To make a comparison of results with those previous approaches (Gupta and Karimi, 2003; Hui and Gupta, 2000; Hui et al., 2000; Pinto and Grossmann, 1995), ten instances of Example 1 were solved. They have been grouped into three different problem sets based on: (a) the number of production orders; (b) the type of objective function being adopted; (c) the handling of either sequence-dependent or non-sequence dependent changeovers; and (d) the handling of a specific due date for each production order or a common one for all of them. Similarly to Gupta and Karimi (2003), two different classes of problem objectives have been used: (1) minimum tardiness and (2) minimum weighted combination of order earliness and tardiness as given by

$$\text{Maximize} \quad \sum_{i\in I}\left(\left(\sum_{s\in S} w_{is}C_{is}\right) - N_i T_i\right) \tag{18}$$

where the coefficients w_{is} are

$$w_{is} = \frac{0.2 \times ord(s) \times \max_{i\in I}(dd_i)}{dd_i} \tag{19}$$

Instead of minimizing the batch earliness, an equivalent problem objective (18) simultaneously maximizing the order completion times (i.e. the lowest total earliness) and minimizing the overall tardiness has been used. Weighting coefficients w_{is} and N_i (a relatively large number) were defined similarly to Gupta and Karimi (2003). For each problem set, Table 1 indicates the selected objective function and the weighting coefficients used, the processing time and the changeover time data sets, the order due dates, and the problem sizes in terms of the number of production orders to be scheduled. Unit-dependent processing times for all problem instances are shown in Table 2, while sequence dependent changeovers for problem sets #1 and #2 are depicted in Table 3. Infeasible batch sequences (i,i') considered by Gupta and Karimi (2003) were handled by simply assigning to them relatively large changeover times $\tau_{ii'}$. Since the proposed formulation assumes that processing times dominate changeover times, a few originally large $\tau_{ii'}$ were slightly reduced but they still prevent from processing batch i' immediately after batch i (see Table 3). Specific delivery due dates for the production orders to be considered in problem sets # 1 and 2 are listed in Table 5. Problem set #3 assumes a fixed due date of 500 h for all batches and unit-dependent setup times whose values are given in Table 4. The largest problem considered in this paper belongs to problem set #2 and consists of scheduling 22 batches at minimum tardiness.

Table 4
Unit-dependent setup times for problem set # 3 of Example 1.

Unit j	1–7	8–9	10–19	20–21	22	23–24	25
su_{ij}	8	1	2.5	6	24	4	5

Table 5
Batch due-dates for problem sets 1 and 2 of Example 1.

Batch	dd_i
2, 13	500
1	510
3, 12, 15	520
4, 6, 16, 20	530
5, 7, 11	540
8, 9, 14, 19	550
21	560
17	570
10, 22	580
18	600

Table 6
Computational results for problem sets 1–3 of Example 1.

Problem set and number of orders	Solution approach	Binary vars, continuous vars, constraints	Objective function	CPU time (s)	Nodes	Iterations
Problem set 1						
5	CBOR Formulation	115, 30, 443	7498.29	0.36	170	683
	Gupta and Karimi (M8)	189, 136, 549	7498	0.5[a]	38	321
	Hui and Gupta/Hui et al.	189, 136, 709	7498	87.7	3357	8843
8	CBOR Formulation	202, 48, 1174	12,384.7	12.7	5694	18,847
	Gupta and Karimi (M8)	433, 223, 1294	12,386.3	38.6[a]	4400	33,990
	Hui and Gupta/Hui et al.	433, 223, 1790	12,329[b]	1268.2	14,568	100,000[c]
10	CBOR Formulation	263, 60, 1852	16,344.5	404.7	121,280	331,729
	Gupta and Karimi (M8)	635, 279, 1948	163,44.5	866[a]	68,550	680,495
	Hui and Gupta/Hui et al.	635, 279, 2729	16,120[b]	1489	8985	100,000[c]
12	CBOR Formulation	325, 72, 2618	19,526[b,d]	5000[c]	1,167,521	4,261,320
	Gupta and Karimi (M8)	881, 332, 2711	19,185[b,e]	5000[c]	–	–
	Hui and Gupta/Hui et al.	881, 332, 3879	16,004[b]	1762	6561	100,000[c]
Problem set 2						
10	CBOR Formulation	263, 60, 1852	0	0.41	0	85
	Gupta and Karimi (M6)	635, 279, 1948	0	1.1[f]	–	–
12	CBOR Formulation	325, 72, 2618	0	1.06	0	213
	Gupta and Karimi (M6)	881, 332, 2711	0	2.50[f]	–	–
17	CBOR Formulation	500, 102, 5321	0	4.34	70	526
	Gupta and Karimi (M6)	1631, 467, 5210	0	45.0[f]	–	–
22	CBOR Formulation	708, 132, 9098	0	36.97	150	2377
	Gupta and Karimi (M6)	2645, 610, 8658	0	342.0[f]	–	–
Problem set 3						
5	CBOR Formulation	115, 30, 443	6828.76	0.75	672	2207
	Gupta and Karimi (M8)	189, 136, 574	6828.76	0.63	–	–
	Hui et al.	189, 136, 709	6716.1	24.0	–	–
	Pinto and Grossmann	161, 167, 511	6151	92.09	1452	–
8	CBOR Formulation	202, 48, 1174	10,986.4	448.9	206,611	623,084
	Gupta and Karimi (M8)	433, 223, 1334	10,986.4	973	–	–

[a]Best CPU time for formulation (M8).
[b]Suboptimal solution.
[c]Resource limit exceeded.
[d]Relative gap = 0.98%.
[e]Relative gap = 3.15%.
[f]Best CPU time among nine formulations of Gupta and Karimi (2003).

Computational results for the three problem sets 1–3 are all presented in Table 6 together with those reported by Gupta and Karimi (2003), Hui et al. (2000), and Pinto and Grossmann (1995). A relative gap tolerance of 1e−06 has been adopted and the CPU time limit was set to 5000 s. For every problem instance a big-M parameter $H = 1000$ was chosen. From Table 6 it follows that the true optimal schedule for almost every instance of Example 1 has been discovered. In addition to discovering the optimal schedule or at least better schedules, a consistent saving in binary variables and what is more important, a systematic reduction of the CPU time have both been achieved by assuming the CBOR.

The best schedules found using the new approach for the 10- and 12-batch instances of problem set # 1, involving sequence-dependent changeovers and a weighted combination of earliness and tardiness as the problem objective, are shown in Figs. 2 and 3. Though the optimal value of the objective function (12,386.3) found by Gupta and Karimi (2003) for the 8-batch instance of problem set # 1 is slightly larger than the one found with the proposed formulation (12,384.7), there is not enough information available to confirm if such a difference really exists. Moreover, sizable savings in CPU time were achieved for almost every instance of problem set # 1. It is worth mentioning that: (a) Gupta and Karimi's models were all solved using CPLEX 7.0 on a Sun Enterprise 250 server with a single Ultra SPARC II 400-MHz processor having 2 GB of RAM, and (b) the CPU time for the G&K approach is the one associated to their best formulation (M8) among nine alternative models. Although differences in computer hardware and optimization software make the direct comparison of CPU times rather unfair, it should be remarked that the CBOR-based approach usually requires fewer iterations. Furthermore, a highly improved schedule has been found for the 12-batch instance of problem set # 1 involving a total of 60 tasks by using the approximate methodology. In less than 8 s the proposed approach discovers an integer solution with an objective value of 19,469.1 that is better than the best schedules found by previous approaches (see Table 6). After 5000 s of CPU time, an improved solution featuring an objective value of 19,526.0 and a relative gap of only 0.0098 was generated (see Fig. 3).

Regarding the model size, for the 5-batch problem instance of set # 1, the best model of Gupta and Karimi (2003) requires 189 binaries, while the proposed formulation includes only 115. This means a 40% saving in binary variables. Moreover, the number of continuous variables drops at least four times. Such a reduction in the model size rises even further with the number of batches to be scheduled. For the 12-batch instance of problem set # 1, savings in binary and continuous variables are substantially larger. On average, the number of binary variables for sets 1 and 2 decreases almost three times and the number of continuous variables drops by a factor of 4.5.

A larger improvement in the computational performance with regards to previous methods was obtained for problem set # 2 featuring sequence-dependent changeover times and the minimum tardiness as the problem objective. Comparing with the lowest CPU time reported for the best G&K model, the computational cost was reduced by a factor of 10 for problem instances with 17 and 22

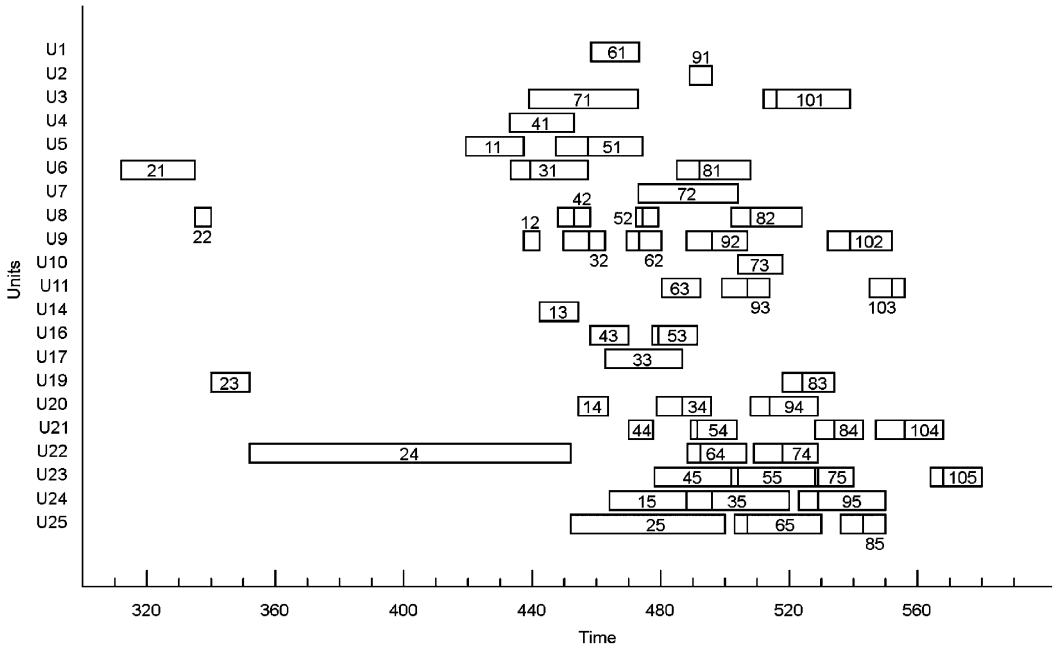

Fig. 2. Optimal schedule found for the 10-batch instance of set # 1 (Example 1).

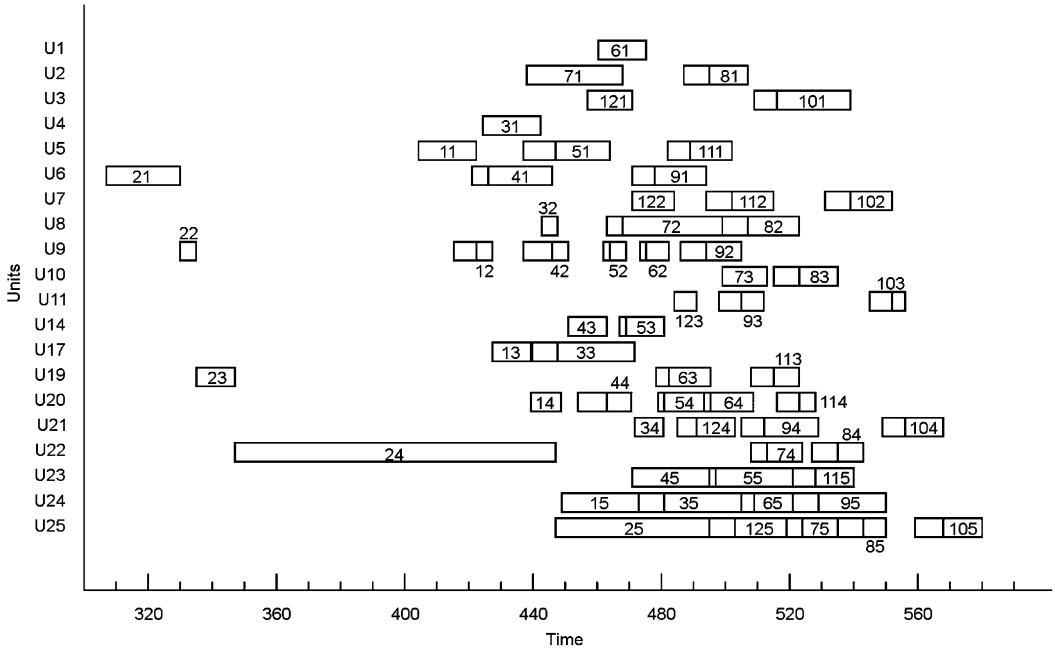

Fig. 3. Best schedule found for the 12-batch instance of set # 1 (Example 1).

batches, respectively. Fig. 4 shows the optimal schedule found for the 22-batch problem instance involving 110 individual tasks. Note that the scheduling horizon for problem set # 2 is large enough to complete all processing tasks before their due dates and the equipment units show sizable idle times. Gupta and Karimi (2003) remarked that problems with minimum tardiness as the scheduling objective are much easier to solve. This statement is actually true only if the total tardiness at the optimum is equal to zero and the problem integrality gap is null. In such cases, there are many ways of scheduling the batches to all meet their promised due-dates as happens in every instance of problem set # 2. Since the optimal values for the MIP and the RMIP formulations are both equal to zero, the branch-and-cut search will stop when the first integer solution with zero tardiness is discovered. This explains why the computational time for problem set # 2 is much lower. We expect that our approximate formulation leads to larger savings in computational requirements for problem instances with finite tardiness at the optimum (see Examples 2–4).

Our new approach also presents an improved computational performance for problem set # 3 assuming unit-dependent changeovers as the number of batches increases. For the 8-batch instance the CPU time is reduced by a half. The optimal objective value for the first instance of problem set # 3 reported by Hui et al. (2000) and Pinto

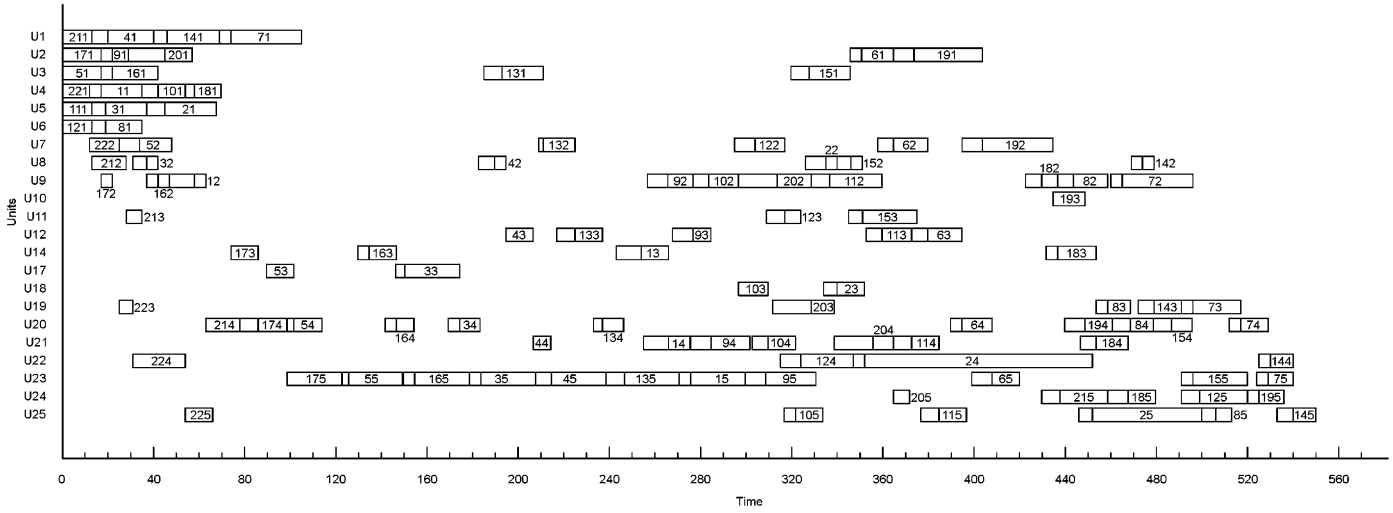

Fig. 4. Optimal schedule found for the 22-batches instance of set # 2 (Example 1).

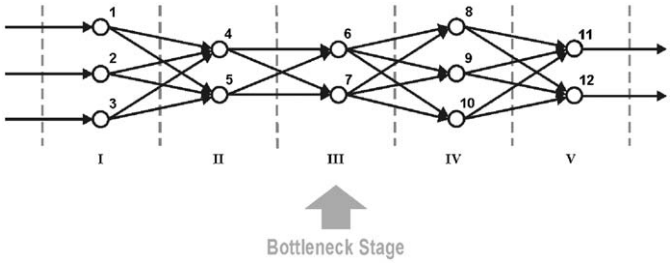

Fig. 5. Batch plant configuration for Examples 2–4.

and Grossmann (1995) differs from that found by Gupta and Karimi (2003) and also by our approximate formulation. As remarked by Gupta and Karimi (2003), such as difference arises because the two former approaches have considered unit-dependent setup times as part of the processing times. Doing that, it is equivalent to delaying the set up of the unit allocated to task (i,s) until the previous task $(i,s-1)$ on batch i is completed. Besides, Pinto and Grossmann (1995) defines an approximate number of time slots not enough to find out the optimal solution.

8.2. Example 2: a large-scale multiproduct batch plant with a strong BS

To further show the effectiveness of the proposed MILP approximate formulation, another multistage batch scheduling problem involving five processing stages and 12 equipment units has been tackled. No resource constraints are considered. Fig. 5 shows the plant configuration where the 12 units have been distributed as follows: U_1–U_3 are in stage 1, U_4–U_5 in stage 2, U_6–U_7 in stage 3, U_8–U_{10} in stage 4, and U_{11}–U_{12} in the last stage 5. Eight batches are to be processed over a time horizon of 100 h. Processing times, due-dates, and unit-dependent setup times data are given in Tables 7 and 8. Stage III with only two equipment units and performing the longest operation arises as the process BS with almost no idle time.

In order to compare the computational performance and the quality of the solution found, Example 2 has been solved using both the proposed approach and the formulation of Méndez et al. (2001). The overall tardiness and the makespan were alternatively selected as the problem goal. Computational results for Example 2 and the remaining Examples 3 and 4 were obtained on the same machine than Example 1 using the same solver (ILOG OPL Studio 3.7, CPLEX 9.0). The relative gap limit was set to 1e−04 and the resource limit for CPU time was set to 3600 s. Besides, a big-M parameter of $H = 200$ was selected for Examples 2–4.

Computational results and statistics are included in Table 9. For both problem objectives (overall tardiness and makespan) the CBOR-based model discovers the same optimal solution provided by the formulation of Méndez et al. (2001) but at much less computational effort. For instance, the number of binary variables drops to a half of the previous value and the CPU time is decreased 20 times. This is because the CBOR-approximate approach requires only 28 sequencing variables (of a total of 87 binaries) while the formulation of Méndez et al. (2001) requires 104 for all stages. Fig. 6 shows the optimal schedule found for Example 2 when the overall tardiness is minimized.

8.3. Example 3: limited manpower

Example 2 is revisited but this time a limited manpower is available to operate the set of units running either at stage I (Example 3a) or stage IV (Example 3b). Again, the overall tardiness and the makespan were alternatively used as problem objective functions. The available manpower consists of five workers and the batch manpower requirements are given at the bottom of Table 8. In order to handle manpower constraints, it is defined the set $Z_{manpower} = \{o_1, o_2, o_3, o_4, o_5\}$ comprising five individual unary resource items, each one featuring a capacity $q_z = 1$, $\forall z \in Z_{manpower}$. Computational results for Examples 3a and 3b using both the proposed approach and a multistage version of the model of Méndez and Cerdá (2002) are included in Table 9. The proposed CBOR formulation requires 40 additional binary variables to handle worker assignment decisions (8 batches×1 stage×5 resource items) in Examples 3a and 3b similarly to the model of Méndez and Cerdá (2002). However, the latter model should include an additional set of 14 sequencing variables on stage IV to tackle Example 3b. This is so because some pairs of batches (i,i') cannot be allocated to the same unit of stage S4 and the related sequencing variables $X_{i,i',s4}$ were previously omitted in Example 2. Since the tasks in stage IV now share manpower, such sequencing variables must be explicitly considered.

For all instances of Example 3 the proposed formulation finds the best solution satisfying the CBOR in very few CPU seconds. On the contrary, the CPU time required to solve the model of Méndez and Cerdá (2002) reaches the limit of 3600 s without proving optimality

Table 7
Processing times, due dates and unit-dependent setup times for Examples 2–4.

Batch	Stage I			Stage II		Stage III		Stage IV			Stage V		Due date
	U_1	U_2	U_3	U_4	U_5	U_6	U_7	U_8	U_9	U_{10}	U_{11}	U_{12}	
1		8.5	7.2	6.2		18.2	12.4	9.1		11.2	8.2		70
2	10.2		8.4		7.2	15.2			10.5			6.4	70
3	9.5		6.7		8.1	17.1	15.5		10.2		8.2	7.3	80
4	8.4	9.8		4.5	8.1		16.6	10.5				7.1	95
5	9.7	7.4		8.3		12.7		11.8		9.8	8.2	7.2	75
6		8.1	6.3		8.2	12.5	17.8		9.5		8.2		100
7	9.6		8.8		8.5		15.4		11.7	10.5		6.3	70
8	10.8	9.7		8.4		16.4	16.7	10.1			8.3	7.9	80
su_j	0.5	0.8	1.0	0.4	0.4	0.4	0.3	1.0	1.2	0.9	0.4	0.3	

Table 8
Sequence-dependent setup times (Examples 2–4) and manpower/steam needs (Example 3/4, respectively).

$\tau_{ii'}$	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8
B_1		1.4	0.7	1.1	1.4	0.7	1.1	0.6
B_2	1.5		1.9	0.9	1.0	2.0	1.6	1.6
B_3	1.6	0.7		1.5	1.4	1.0	1.9	2.0
B_4	1.2	0.7	1.9		0.8	1.1	1.1	0.6
B_5	1.1	1.6	0.7	0.6		0.8	1.0	1.7
B_6	2.0	1.4	1.8	2.0	0.8		1.5	1.4
B_7	1.4	1.5	1.6	0.8	0.7	1.4		1.3
B_8	0.8	1.7	1.1	0.7	1.1	0.8	1.2	
Manpower	2	3	2	1	3	2	3	2
Steam								
Stage I	9.0	7.0	9.0	15.0	12.0	9.0	15.0	8.0
Stage IV	7.0	9.0	12.0	9.0	6.0	18.0	12.0	18.0

Table 9
Computational results for Examples 2–4.

Objective function and solution approach	Binary vars, continuous vars, constraints	Objective function	CPU time (s)	Nodes	Iterations
Example 2					
Overall tardiness					
CBOR formulation	87, 88, 370	5.7	2.33	3923	11,035
Méndez et al. (2001)	163, 88, 370	5.7	52.64	95,224	200,264
Makespan					
CBOR Formulation	87, 81, 370	94.7	10.41	22,100	61,480
Méndez et al. (2001)	163, 81, 370	94.7	201.84	349,558	1,196,811
Example 3a					
Overall tardiness					
CBOR formulation	127, 88, 658	6.6	33.92	28,706	86,022
Méndez and Cerdá (2002)	203, 88, 658	25.3[a,b]	3600[c]	1,768,052	9,146,099
Makespan					
CBOR Formulation	127, 81, 658	94.7	15.14	14,485	68,837
Méndez and Cerdá (2002)	203, 81, 658	95.2[a,d]	3600[c]	2,457,606	11,429,900
Example 3b					
Overall tardiness					
CBOR Formulation	127, 88, 658	5.9	6.70	5339	25,502
Méndez and Cerdá (2002)	217, 88, 658	32.5[a,b]	3600[c]	1,619,341	10,342,110
Makespan					
CBOR Formulation	127, 81, 658	94.7	13.30	12,218	59,906
Méndez and Cerdá (2002)	217, 81, 658	95.6[a,e]	3600[c]	1,720,312	10,264,947
Example 4					
Overall tardiness					
CBOR Formulation	223, 173, 1667	5.7	47.45	12,657	113,183
Méndez and Cerdá (2003)	313, 173, 1667	45.4[a,b]	3600[c]	1,053,097	8,998,133
Makespan					
CBOR Formulation	223, 166, 1667	94.7	39.81	12,290	87,005
Méndez and Cerdá (2003)	313, 166, 1667	96.9[a,f]	3600[c]	1,081,428	8,605,495

[a]Suboptimal solution.
[b]Best possible solution = 0.0, relative gap = 1.0 (100%).
[c]Resource limit exceeded.
[d]Relative gap = 0.17.
[e]Relative gap = 0.25.
[f]Relative gap = 0.32.

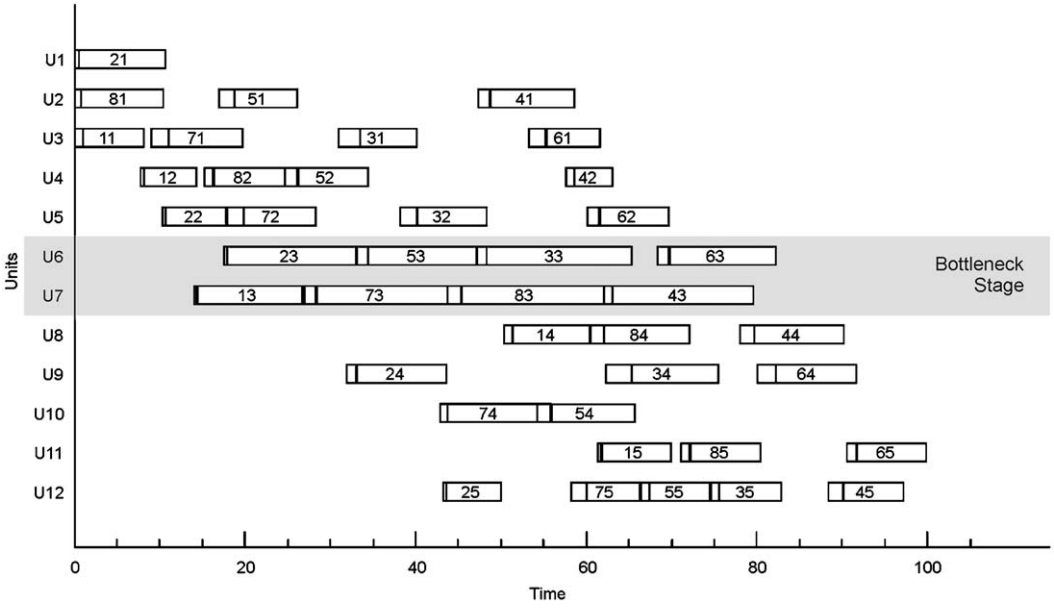

Fig. 6. Minimum-tardiness schedule found for Example 2.

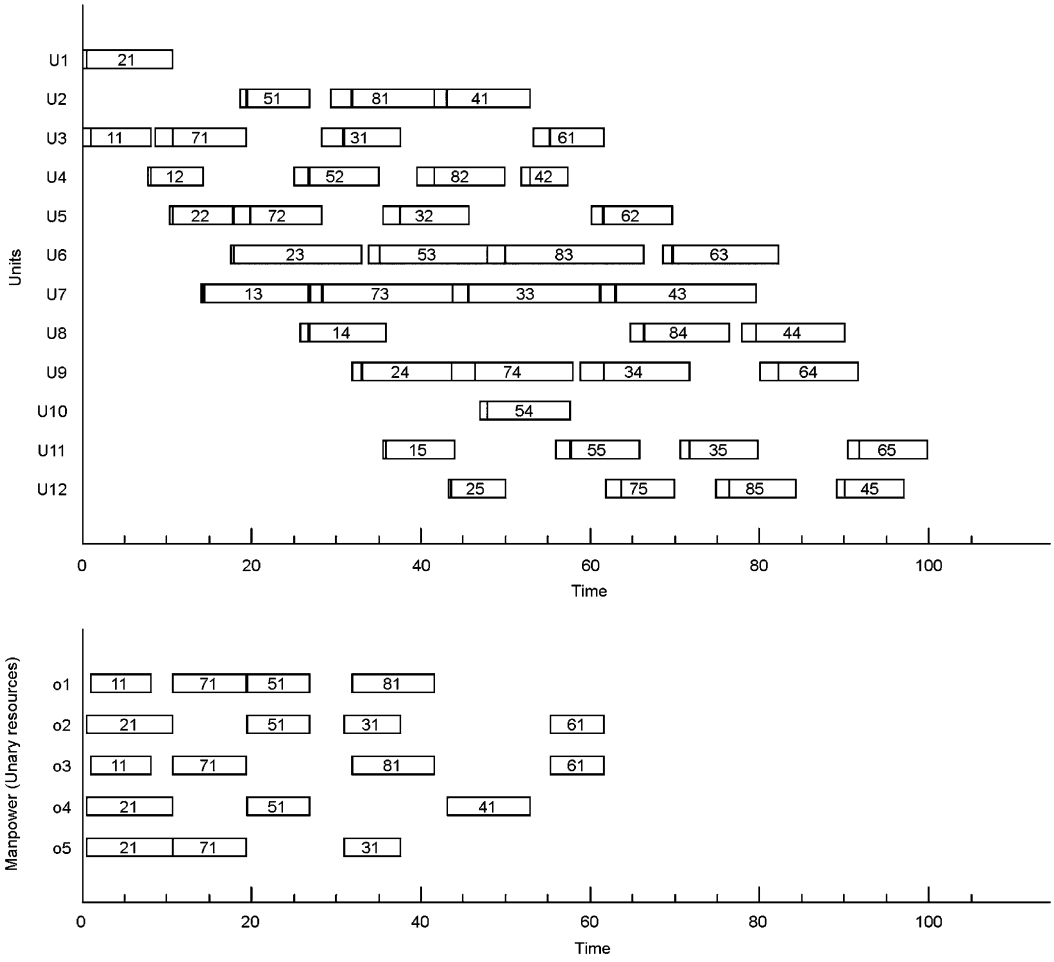

Fig. 7. Best minimum-tardiness schedule for Example 3a.

in all cases and for each problem goal being considered. When the minimum overall tardiness is the problem objective, the latter approach finds very poor feasible solutions with a relative gap of 100% at the time limit. Instead, the CBOR-optimal solution features a much lower objective value, i.e. almost four times lower for Example 3a and six times smaller for Example 3b. As the model of Méndez and

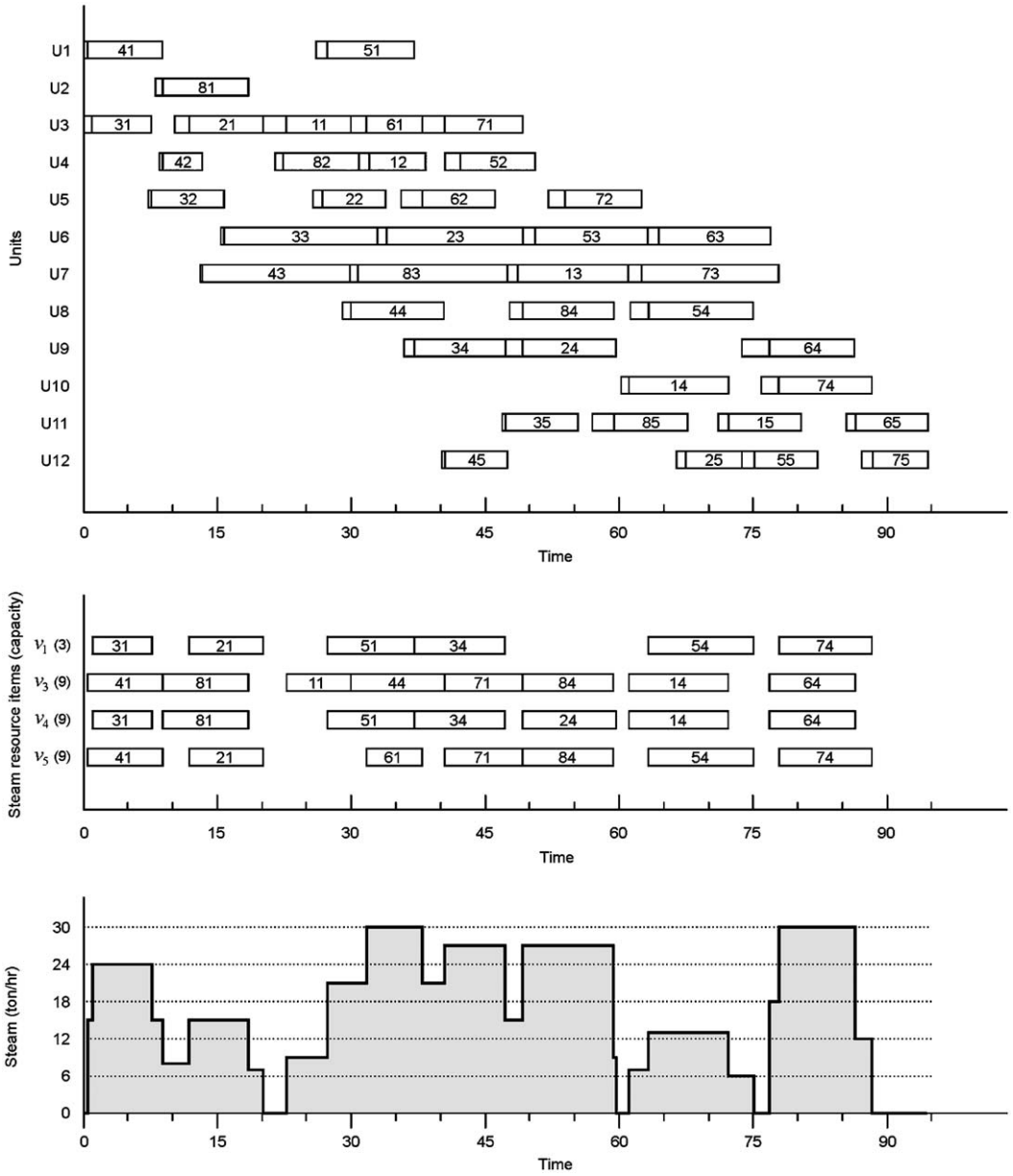

Fig. 8. Minimum-makespan schedule found for Example 4.

Cerdá (2002) does not end up with the optimal solution, it cannot be ensured that the schedules found through our approximate model are the optimal ones. For the minimum makespan problem, the best solution reported for Example 2 is still the optimal one and, therefore, the CBOR-based formulation was able to discover the optimal schedule for Examples 3a and 3b. In contrast, the model of Méndez and Cerdá (2002) fails to find the optimal solution within the CPU time limit. The best CBOR-based solution for Example 3a (with the minimum tardiness as the scheduling objective), including the assignment of individual workers to tasks at stage I, is shown in Fig. 7.

8.4. Example 4: limited steam flow for multistage steam requirements

Example 2 is revisited again but now the limiting manufacturing resource is the steam flow rather than the manpower. A steam pool is available to carry out the processing tasks at stages I and IV all requiring heating steam. The overall tardiness and the makespan were adopted as alternative objective functions. Table 8 shows the batch steam requirements in ton/h at stages I and IV. The maximum steam flow is 30 ton/h and it has been divided into five elements of variable size. Therefore, five steam resource items have been defined. Since the two stages share the same manufacturing resource, the sequencing constraints for Case 2 introduced in Section 7 are to be used.

Computational results found using both the proposed formulation and the previous approach of Méndez and Cerdá (2003) are included in Table 9. For both problem objectives, the same optimal value found in Example 1 has been encountered. Therefore, the steam resource is merely a capacity-constrained resource and the solutions provided by the proposed approach are consequently the optimal ones. Compared with the model size for Example 2, the proposed formulation requires 80 extra allocation variables and 56 additional sequencing variables to order each pair of tasks sharing the same resource and belonging to different stages. Besides, the resource element

capacities ϕ_z and the related amounts of resource assigned to tasks (β_{isz}) increase the number of continuous variables. Despite that, the best CBOR-solution for both objective functions has been found in a very low CPU time. In contrast, the approach of Méndez and Cerdá (2003) could not discover the optimal solution in either case within the time limit of 3600 s. Moreover, it provides a very poor feasible solution when the minimum total tardiness is the problem objective.

Fig. 8 shows the minimum-makespan schedule for Example 4. The optimal capacities found for the steam resource items and their assignment to processing tasks at stages I and IV as well as the steam usage profile along the time horizon are also depicted in Fig. 8.

9. Conclusions

An approximate continuous-time formulation for the short-term scheduling of multiproduct, multistage batch facilities with resource constraints that relies on the so-called constant-batch-ordering rule (CBOR) has been developed. The CBO Rule is a novel concept introduced in this paper from bottleneck resource considerations and some empirical evidence. It simply states that two batches should be processed in the same order at every stage whenever both have been allocated to the same resource item. By using the global precedence notion to formulate the problem, a unique sequencing variable will be enough to control the relative ordering of a pair of batches over the whole processing structure. This leads to a substantial saving in 0–1 variables and a significant reduction in CPU time. If the same batch ordering in certain stages is not attractive, the CBOR assumes that a pair of alternative machines or resource items to separately process them will always be available. The CBOR rule seems better suited for scheduling problems where the processing time change pattern along the task sequence is roughly similar for all production orders. Three different types of objective functions were applied: overall earliness, total tardiness and makespan. Despite the best production schedule not necessarily satisfies the CBO-Rule, optimal or near-optimal solutions to several instances of two large-scale multistage batch scheduling examples without resource constraints were discovered at low computational cost. Comparison with the results obtained using two of the most efficient sequential methodologies available in the literature (Méndez et al., 2001; Gupta and Karimi, 2003) show a substantial reduction in the CPU time, especially for problems with a large number of batches and the overall tardiness as the objective function. The CBOR-based problem representation has also been extended to account for resource constraints. Unary and finite renewable resources other than equipment units were considered. Additional constraints controlling the allocation of unary/finite resource items to batches and the ordering of batches at every resource item were presented. A decomposition scheme was embedded in the CBOR-formulation to convert finite resources into a set of unary resource items with capacities defined by the model. In this way, a unified treatment of unary and finite resources has been obtained. Different instances of two multistage batch scheduling problems with some limitations in the available manpower or steam flow were tackled. The results were favorably compared with those obtained using the general precedence approach of Méndez and Cerdá (2002, 2003). Therefore, the CBOR has proven to be an excellent assumption even for resource-constrained multistage batch scheduling.

Notation

Indices

i,i'	batch
j,j'	equipment unit
r	resource
s,s'	stage
s^f, s^ℓ	first and last processing stages
z	resource item

Sets

I	batches
J	units
J_{is}	units where task (i,s) can be allocated
J_s	units that belong to stage s
$J_{s+1}^{(j)}$	units at stage s+1 physically connected to unit $j \in J_s$
R	discrete/continuous renewable resources
R^α	subset of unary discrete resources
R^β	subset of finite discrete/continuous resources
R_{is}	resources required to run task (i,s)
S	stages
Z_r	available items of resource r

Parameters

dd_i	due date for batch i
H	length of the scheduling horizon
pt_{ij}	processing time of batch i at unit j
q_z	capacity of unary resource item z
Q_r	availability of finite resource $r \in R$
rt_i	release time of batch i
ru_j	ready time of unit j
su_{ij}	setup time of batch i in unit j
ε_i	tardiness weight for batch i
ρ_{isjr}	rth resource requirement for task (i,s) when allocated to unit j
ρ_{isr}	rth resource requirement for task (i,s)
$\tau_{ii'j}$	sequence-dependent setup time between batches i and i' in unit j

Binary variables

W_{isz}	binary variable denoting that resource item z has been allocated to task (i,s)
$X_{ii'}$	binary variable denoting that batch $i$is run before $(X_{ii'}=1)$ or after $(X_{ii'}=0)$ batch i'at any stage where both batches are allocated to the same resource (constant-batch-ordering principle)
$X_{is,i's'}$	binary variable denoting that stage $s$of batch $i$is run before $(X_{is,i's'}=1)$ or after $(X_{is,i's'}=0)$ stage $s'$of batch i'
Y_{ij}	binary variable denoting that batch i is allocated to unit j

Continuous variables

C_{is}	completion time of batch i at stage s
MK	makespan
S_{is}	starting time of batch i at stage s
T_i	tardiness of batch i
β_{isz}	amount of resource item z allocated to batch i at stage s
ϕ_z	capacity of discrete/continuous finite resource item z

References

Adams, J., Balas, E., Zawack, D., 1988. The shifting bottleneck procedure for job shop scheduling. Management Science 34 (3), 391–401.

Balas, E., Vazacopoulos, A., 1998. Guided local search with shifting bottleneck for job shop scheduling. Management Science 44 (2), 262–275.

Castro, P.M., Barbosa-Póvoa, A.P., Matos, H.A., Novais, A.Q., 2004. Simple continuous-time formulation for short-term scheduling of batch and continuous processes. Industrial & Engineering Chemistry Research 43, 105.

Castro, P.M., Grossmann, I.E., 2005. New continuous-time MILP model for the short-term scheduling of multistage batch plants. Industrial & Engineering Chemistry Research 44, 9175–9190.

Chase, R.B., Aquilano, N.J., Jacobs, F.R., 1998. Production and Operations Management. eighth ed. Irwin/McGraw-Hill, New York.

Dauzere-Peres, S., Lasserre, J.-B., 1993. A modified shifting bottleneck procedure for job-shop scheduling. International Journal of Production Research 31 (4), 923–932.

Floudas, C.A., Lin, X., 2004. Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. Computers and Chemical Engineering 28, 2109.

Goldratt, E.M., Cox, J., 1986. The Goal. North River Press.

Gupta, S., Karimi, I.A., 2003. An improved MILP formulation for scheduling multiproduct, multistage batch plants. Industrial & Engineering Chemistry Research 42, 2365.

Harjunkoski, I., Grossmann, I.E., 2002. Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. Computers and Chemical Engineering 26, 1533.

Hui, C.W., Gupta, A., 2000. A novel MILP formulation for short-term scheduling of multistage multi-product batch plants. Computers and Chemical Engineering 24, 1611–1617.

Hui, C.W., Gupta, A., Meulen, H.A.J., 2000. A novel MILP formulation for short-term scheduling of multi-stage multi-product batch plants with sequence-dependent constraints. Computers and Chemical Engineering 24, 2705–2717.

Ivens, P., Lambrecht, M., 1996. Extending the shifting bottleneck procedure to real-life applications. European Journal of Operational Research 90, 252–268.

Maravelias, C.T., 2006. A decomposition framework for the scheduling of single- and multi-stage processes. Computers and Chemical Engineering 30, 407–420.

Méndez, C.A., Henning, G.P., Cerdá, J., 2001. An MILP continuous-time approach to short-term scheduling of resource-constrained multistage flowshop batch facilities. Computers and Chemical Engineering 25, 701–711.

Méndez, C.A., Cerdá, J., 2002. An MILP framework for short-term scheduling of single-stage batch plants with limited discrete resources. Computer-Aided Chemical Engineering 10, 721–726.

Méndez, C.A., Cerdá, J., 2003. Short-term scheduling of multistage batch processes subject to limited finite resources. Computer-Aided Chemical Engineering 15, 984–989.

Méndez, C.A., Cerdá, J., Grossmann, I.E., Harjunkoski, I., Fahl, M., 2006. State-of-the-art review of optimization methods for short-term scheduling of batch processes. Computers and Chemical Engineering 30, 913.

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

Pinto, J.M., Grossmann, I.E., 1995. A continuous time mixed integer linear programming model for short term scheduling of multistage batch plants. Industrial and Engineering Chemistry Research 34, 3037–3051.

Pinto, J.M., Grossmann, I.E., 1996. An alternate MILP model for short-term scheduling of batch plants with preordering constraints. Industrial & Engineering Chemistry Research 35, 338–342.