Computers and Chemical Engineering 33 (2009) 871–886

Contents lists available at ScienceDirect

Computers and Chemical Engineering

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

# A general resource-constrained scheduling framework for multistage batch facilities with sequence-dependent changeovers

Pablo A. Marchetti, Jaime Cerdá*

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

ARTICLE INFO

*Article history:*
Received 11 October 2006
Received in revised form 27 June 2008
Accepted 16 December 2008
Available online 31 December 2008

*Keywords:*
Resource-constrained scheduling
Multiproduct batch plant
MILP optimization model

ABSTRACT

This work introduces a new MILP sequential approach to the short-term scheduling of multistage batch plants that accounts for sequence-dependent changeover times, intermediate due dates and limited availability of renewable resources. It relies on a continuous-time formulation based on the general precedence notion that uses different sets of binary variables to handle allocation and sequencing decisions. To avoid resource overloading, additional constraints in terms of sequencing variables and a new set of 0-1 overlapping variables are presented. They allow tracking the set of tasks requiring the same resource and running in parallel at the start of another process operation. In this way, the proposed formulation involves a reasonable number of binary variables and constraints and features a very good computational behavior, even in the presence of hard bottleneck resources. Four illustrative examples, one of them including multiple bottleneck resources shared by several processing stages, have been efficiently solved.

© 2008 Elsevier Ltd. All rights reserved.

## 1. Introduction

Batch processing plants generally provide the necessary flexibility to produce a wide range of high-value added products by running a set of process operations using the same manufacturing resources. Besides equipment units and storage tanks, processing tasks may require heating steam, electricity, tools or manpower. Batch scheduling methodologies often assume unlimited availability of production resources other than processing and storage units. In this way, such resources never become production bottlenecks and can be ignored in the problem formulation. In real-world manufacturing environments, however, it is quite usual to prevent certain processing tasks from running in parallel simply because their overall requirement of some production resources different from equipment (like utilities, tools or manpower) exceeds their total availabilities. To guarantee the development of a feasible schedule, resource constraints should be added to the problem representation in order to monitor the overall resource consumption over the scheduling horizon. In this way, the mathematical formulation will be able to avoid the overloading of some resource capacity at any time. Manufacturing resources are generally grouped into two types: renewable and non-renewable resources. A renewable resource is one that is recovered when the task to which it was allocated has concluded. Renewable resources can be discrete (tools, manpower) or continuous (heating, refrigeration, electricity). In contrast, non-renewable resources, like intermediates or raw materials, are consumed by tasks and every resource capacity allocated to them is no longer recovered at their completion.

Depending on the way the set of batches and their sizes are defined, model-based batch scheduling methodologies can be grouped into two types: monolithic and sequential approaches (Méndez, Cerdá, Grossmann, Harjunkoski, & Fahl, 2006). Monolithic approaches are those which simultaneously determine the set of tasks to be scheduled, the allocation of manufacturing resources to tasks and the sequence of tasks at any equipment unit. A task refers to a processing/storage operation performed on a batch of some intermediate or final product. Finding the set of tasks to be scheduled implies to establish the set of batches and batch sizes to be processed at every manufacturing stage over the scheduling horizon. Although monolithic scheduling models are quite general, they are more oriented towards the treatment of arbitrary network processes involving complex product recipes. They are based on either the state-task network (STN) or the resource-task network (RTN) concepts to describe production recipes. Since large model sizes are usually generated, their application is restricted to schedule processes with a small number of tasks over rather short time horizons. The selected time representation divides monolithic approaches into two groups: discrete-time and continuous-time representations. Discrete-time monolithic formulations use a fixed time grid that is valid for all shared resources and the tasks are constrained to start and finish at predefined grid points. Different

* Corresponding author. Tel.: +54 342 4559175; fax: +54 342 4550944.
*E-mail address:* jcerda@intec.unl.edu.ar (J. Cerdá).

0098-1354/$ – see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compchemeng.2008.12.007

**Nomenclature**

*Subscripts*

| | |
|---|---|
| $i, i'$ | batch |
| $j, j'$ | equipment unit |
| $s, s'$ | stage |
| $r$ | resource |

*Sets*

| | |
|---|---|
| $I$ | set of batches |
| $S$ | set of processing stages |
| $S_i$ | set of stages for batch $i$ |
| $S_{ii'}$ | set of common stages for batches $i$ and $i'$ |
| $J$ | set of equipment units |
| $J_s$ | set of equipment units at stage $s$ |
| $J_{is}$ | set of equipments units that can be allocated to batch $i$ at stage s |
| $J_{ii's}$ | set of units that can be allocated to batches $i$ and $i'$ at stage $s$ ($J_{ii's} = J_{is} \cap J_{i's}$) |
| $J^{(j)}_{s+1}$ | set of units at stage $s+1$ physically connected to unit $j \in J_s$ at stage $s$ |
| $J^{P}_{isr}$ | set of units where a task sharing resource $r$ and eventually overlapping task $(i, s)$ can be allocated |
| $R$ | set of discrete/continuous renewable resources |
| $R_{is}$ | set of resources required to process batch $i$ at stage $s$ |

*Parameters*

| | |
|---|---|
| $pt_{ij}$ | processing time of batch $i$ at unit $j$ |
| $dd_i$ | due date for batch $i$ |
| $rt_i$ | release time of batch $i$ |
| $ru_j$ | ready time of unit $j$ |
| $s^f_i, s^\ell_i$ | first and last processing stages for batch $i$ |
| $su_{ij}$ | setup time of batch $i$ in unit $j$ |
| $\tau_{ii'j}$ | sequence-dependent setup time between batches $i$ and $i'$ in unit $j$ |
| $\alpha_i$ | weight of earliness for batch $i$ |
| $\beta_i$ | tardiness weight for batch $i$ |
| $D_r$ | maximum capacity of resource $r$ |
| $\rho_{isr}$ | $r$th resource requirement for batch $i$ at stage $s$ |
| $\rho_{isjr}$ | $r$th resource requirement for task $(i, s)$ when performed at unit $j$ |
| $\delta$ | a very small positive number |
| $H$ | length of the scheduling horizon |
| $pt^{\mathrm{Min}}_{is}$ | minimum processing time of batch $i$ at stage $s$ |

*Binary variables*

| | |
|---|---|
| $Y_{ij}$ | binary variable denoting that batch $i$ is allocated to unit $j$ |
| $X_{i's',is}$ | binary variable denoting that stage $s'$ of batch $i'$ is run before ($X_{i's',is} = 1$) or after ($X_{i's',is} = 0$) stage $s$ of batch $i$ |
| $W_{i's',is}$ | binary variable denoting that stage $s'$ of batch $i'$ is completed after starting batch $i$ at stage $s$ |

*Continuous variables*

| | |
|---|---|
| $S_{is}$ | starting time of batch $i$ at stage $s$ |
| $C_{is}$ | completion time of batch $i$ at stage $s$ |
| $T_i$ | tardiness of order $i$ |
| $MK$ | makespan |
| $Q_{isjr}$ | amount of resource $r$ required by the task running at unit $j$ and overlapping the starting task $(i, s)$ |

STN discrete models (Kondili, Pantelides, & Sargent, 1993; Shah, Pantelides, & Sargent, 1993; Rodrigues, Latre, & Rodrigues, 2000) and RTN-based discrete formulations (Pantelides, 1994) have been proposed in the literature. By including a material balance for each state at every fixed time slot, batch mixing/splitting and material recycles can be handled. The fact that resource constraints have to be satisfied at predefined time points reduces the problem complexity and makes discrete models very attractive in the case of considering resource and inventory limitations. A RTN representation is preferred for batch scheduling problems involving identical equipment since it leads to a significant saving in binary variables. Another interesting feature of discrete approaches is the easy handling of resource requirements varying along the task execution. However, the treatment of sequence-dependent changeovers requires a finer time discretization to account for much smaller setup times, that significantly increases the model size and the computational requirement. In addition, RTN discrete models need to define further tasks for each processing unit.

On the other hand, continuous-time monolithic models involve a variable time grid that can be either common for all shared resources (Castro, Barbosa-Póvoa, & Matos, 2001; Castro, Barbosa-Póvoa, Matos, & Novais, 2004; Giannelos & Georgiadis, 2002; Lee, Park, & Lee, 2001; Maravelias & Grossmann, 2003; Mockus & Reklaitis, 1997) or unit-specific (Ierapetritou & Floudas, 1998; Janak, Lin, & Floudas, 2004). By using a variable time grid, not only an exact problem representation can be developed but also a significant reduction on the number of time points, 0-1 variables and constraints is simultaneously achieved. Moreover, variable batch sizes and batch-size dependent resource requirements can be handled. When a common variable time grid is considered, rather simple resource constraints monitor the total resource requirement to avoid resource overloading at any time point. Since at least one time point is needed for each different task start time, the number of global time points mostly determines the problem size. Usually, problems requiring more than 15 time points need a considerable CPU time to guarantee optimality. Unit-specific continuous-time models use a different variable time grid for each shared resource, thus allowing tasks to start at different times, despite they have been allocated to the same time event. Because of such a model flexibility, unit-specific time grids generally require less event points than continuous-time monolithic methods using global time points. Despite they were developed for STN-based processes, unit-specific continuous-time monolithic models can be easily adapted to tackle sequential processes. However, the lack of reference points for checking the limited availability of shared resources further increases the model complexity by forcing to define additional problem variables and new big-M special constraints that deteriorate its computational performance.

In monolithic approaches, both the batching problem and the batch assignment and sequencing problems are solved at once. In turn, sequential scheduling approaches assume that the batching problem has already been solved. Based on the production orders placed by customers and the available plant resources, the batching problem provides the exact set of batches for each product/state to be processed, batch sizes, due dates and any other related information. As a result, sequential methods deal with a predefined set of batches and linear product recipes where splitting/mixing operations never arise. Since the batch integrity is preserved throughout the processing sequence, allocation and sequencing constraints can be written in terms of processing tasks on predefined batches instead of using global or unit-specific time-events. Moreover, no mass balances need to be included in the problem formulation because every intermediate is just allocated to a single product. These assumptions reduce the complexity of the mathematical

model since decision variables are defined for a given set of tasks (each stage of every batch) with known resource requirements. Different sequential methods have been published in the last decade (Cerdá, Henning, & Grossmann, 1997; Gupta & Karimi, 2003; Hui & Gupta, 2000; Hui, Gupta, & Meulen, 2000; Méndez, Henning, & Cerdá, 2001; Pinto & Grossmann, 1995, 1996). Some of them were generalized to also handle renewable resources different from equipment. Pinto and Grossmann (1997) presented a MILP sequential approach based on a slot-based continuous-time representation that extends a former mathematical formulation for unconstrained multistage batch plants (Pinto & Grossmann, 1995, 1996). As the number of binary variables and big-M constraints substantially increased, the general MILP resource-constrained model became almost computationally unsolvable. Consequently, the authors developed a problem solution methodology that combines a branch-and-bound MILP algorithm with disjunctive programming. Slot-based representations were also presented by Lamba and Karimi (2002) and Lim and Karimi (2003) to tackle semicontinuous scheduling problems of single-stage parallel production lines with resource constraints. Lamba and Karimi (2002) used identical slots across all processors while Lim and Karimi (2003) employed asynchronous slots. Since the underlying idea of an asynchronous slot is similar to the unit-specific time event, checkpoints for resource utilization are placed at the start of each slot and additional variables and constraints should be included to establish the slot relative positions. Méndez and Cerdá (2002) developed an MILP continuous-time representation that independently handles unit allocation and task sequencing decisions through different sets of binary variables. Sequencing variables allow to order the tasks allocated either to the same equipment unit or to another discrete resource. In this way, an important saving in binary variables was achieved. Afterwards, Méndez and Cerdá (2003) reported a more general MILP formulation to deal with both continuous and discrete finite renewable resources. Each continuous resource is divided into a discrete number of sub-sources or pieces that are assigned to tasks through new allocation variables. Then, sequencing variables are still used to ordering tasks allocated to the same discrete or continuous resource item. The maximum number of pieces into which a continuous renewable can be divided is a model parameter, while each piece capacity is a non-negative variable selected by the model. However, the proposed resource representation may sometimes exclude the problem optimum from the feasible space and, consequently, optimality is not guaranteed.

This work introduces a new MILP formulation for the short-term scheduling of sequential batch processes that accounts for limited renewable resources aside from equipment. Different sets of binary variables allocating equipment units to tasks and sequencing processing tasks on the same queue are defined. The mathematical model is based on the general precedence notion proposed by Méndez et al. (2001). To handle general continuous/discrete renewable resource constraints that guarantee optimality, a new set of 0-1 overlapping variables is defined for each pair of tasks sharing a common resource. To reduce the number of 0-1 variables, sequencing variables related to pairs of tasks not allocated to the same unit, in combination with the overlapping variables, permit to detect task overlappings and resource overloading. The resulting MILP formulation includes a reasonable amount of binary variables and constraints and shows a good computational behavior even for rather hard resource limitations. Resource capacity constraints have been extended to account for unit-dependent resource requirements at reasonable cost in terms of additional variables and CPU time. Four illustrative examples involving the scheduling of single/multistage batch plants with several units running in parallel at each stage and multiple limiting resources were successfully solved.

## 2. Problem formulation

Given:

(i) a multiproduct multistage batch facility with several units $j \in J$ running in parallel at each processing stage $s \in S$,
(ii) the set of batches $i \in I$ to be processed and their release times $rt_i$ and promised due-dates $dd_i$,
(iii) the sequence of processing stages $s \in S_i$ required for each batch $i \in I$,
(iv) the available equipment units $j \in J_s$ at each stage $s \in S$ and their ready times $ru_j$,
(v) the subset of units $j \in J_{is} \subseteq J_s$ available for task $(i, s)$,
(vi) the plant topology structure specifying the interconnections between units of consecutive processing stages,
(vii) the batch processing times $pt_{ij}$ and the sequence-dependent setup times given as the sum of two components $su_{ij}$ and $\tau_{i'ij}$,
(viii) a set of renewable resources different from equipment, $r \in R$, and their available capacities $D_r$,
(ix) the subset of resources $R_{is} \subset R$ required to run stage $s$ of batch $i$,
(x) the amount $\rho_{isr}$ of resource $r \in R_{is}$ (or $\rho_{isjr}$ if unit-dependent) needed per unit amount of task $(i, s)$, and
(xi) the length of the time horizon $H$.

The problem goal is to find a feasible production schedule that satisfies all problem constraints while completing all batches within the specified time horizon, and simultaneously optimizes the selected plant performance criterion, like the overall weighted earliness, the overall weighted tardiness or the makespan.

## 3. Model assumptions

(1) Model parameters are all deterministic.
(2) The set of batches is already known, i.e. a batching procedure converting product demands into batches has already been performed.
(3) Equipment units operate in non-preemptive mode.
(4) Batch mixing and splitting are not allowed.
(5) Setup times are sequence dependent.
(6) Renewable resources feature constant capacities over the scheduling horizon.
(7) The resource requirement rate remains constant during the execution of the processing task.

## 4. Non-resource constrained mathematical formulation

The mathematical formulation proposed by Méndez et al. (2001) is extended to rigorously deal with limiting resources other than equipment units also constraining the problem feasible domain. Model constraints for the unit assignment, sequencing and timing of processing tasks are first analyzed. Additional constraints to deal with discrete/continuous resource requirements are presented on the next section.

### 4.1. Allocation constraints

Each stage $s$ of batch $i$ must be assigned to an available unit $j \in J_{is}$. Since a multistage multiproduct facility is being considered, each unit $j$ belongs to a single stage and the stage index $s$ is not needed in the definition of the allocation variable $Y_{ij}$.

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

### 4.2. Topological constraints

The plant layout may limit the number of possible routes a batch can follow throughout the processing structure. Since an equipment item is often physically connected to a subset of the units available at the next downstream stage, the Constraint (2) aims to guarantee that the next stage $s+1$ of batch $i$ will be scheduled on an equipment $j'$ eligible for task $(i, s+1)$ and accessible from unit $j$, assuming that task $(i, s)$ is performed in unit $j \in J_{is}$. If $J_{s+1}^{(j)}$ stands for the set of units available at stage $s+1$ that are physically connected to unit $j \in J_{is}$, then unit $j'$ should belong to $J_{s+1}^{(j)} \cap J_{i,s+1}$. In Eq. (2), $s_i^\ell$ stands for the last stage of batch $i$.

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

### 4.3. Task sequencing constraints

Given a pair of batches $(i, i') \in I$ running at the same stage $s \in S_{ii'}$ (where $S_{ii'} = S_i \cap S_{i'}$), and assuming $J_{ii's} = (J_{is} \cap J_{i's}) \neq \emptyset$, it may be possible that the corresponding tasks $(i, s)$ and $(i', s)$ were allocated to the same processing unit $j \in J_{ii's}$. Then, a pair of sequencing constraints for tasks $(i, s)$ and $(i', s)$ to ensure that the tasks are performed one by one at some unit $j \in J_{ii's}$ and, in addition, a single binary variable $X_{i's,is}$ to indicate their relative ordering must both be defined. By using the general precedence concept introduced by Méndez et al. (2001), which allows to independently handle assignment and sequencing decisions through different sets of binary variables, the sequencing constraints for tasks $(i', s)$ and $(i, s)$ verifying the condition $i' < i$ can be expressed through Eqs. (3) and (4). Since the subscript $j$ is not needed on the domain of the sequencing variables $X_{i's,is}$, a remarkable reduction in the number of binary variables is achieved. Whenever batches $(i, i')$ are allocated to the same unit $j \in J_{ii's}$ at some processing stage $s$ ($Y_{ij} = Y_{i'j} = 1$), then either the Constraint (3) or (4) becomes active depending on whether $X_{i's,is}$ is one or zero. In case $X_{i's,is} = 1$, then task $(i', s)$ must finish before task $(i, s)$ begins. Otherwise, $X_{i's,is} = 0$ and task $(i, s)$ is executed earlier. Notice that the value of $X_{i's,is}$ will be meaningful only if both tasks have been assigned to the same processing queue. Otherwise, it becomes meaningless and can be used for another purpose.

$$C_{i's} + \tau_{i'ij} + su_{ij} \leq S_{is} + H(1 - X_{i's,is}) + H(2 - Y_{ij} - Y_{i'j})$$
$$\forall i, i' \in I, s \in S_{ii'}, j \in J_{ii's} : (i' < i) \tag{3}$$

$$C_{is} + \tau_{ii'j} + su_{i'j} \leq S_{i's} + H X_{i's,is} + H(2 - Y_{ij} - Y_{i'j})$$
$$\forall i, i' \in I, s \in S_{ii'}, j \in J_{ii's} : (i' < i) \tag{4}$$

In the next section, the definition of variables $X_{i's,is}$ is extended to make its value still meaningful even if tasks $(i, s)$ and $(i', s)$ have been assigned to different equipment units. In this way, such variables will also help monitor the overlapping of parallel tasks sharing the same renewable resource (different from equipment) so as to prevent temporal overloading of the resource capacity at any time. As a result, a significant saving in additional 0-1 variables is achieved.

### 4.4. Task timing

The relationship between the starting time and the completion time of each task is defined based on the processing time required on the allocated unit.

$$S_{is} = C_{is} - \sum_{j \in J_{is}} pt_{ij} Y_{ij} \qquad \forall i \in I, \quad s \in S_i \tag{5}$$

### 4.5. Technological constraints

Constraint (6) defines the relationship between the completion and the starting times for every pair of consecutive processing stages on the same batch $i$.

$$C_{is} \leq S_{i,s+1} \qquad \forall i \in I, \quad s \in S_i - \{s_i^\ell\} \tag{6}$$

### 4.6. Bounds on batch starting and completion times

Constraints (7)–(9) set up bounds on batch starting and completion times. In particular, Constraint (7a) establishes a lower bound on the time at which the first stage of any batch can start taking into account not only its release time but also both the ready time and the setup time for the allocated unit. Constraint (7b) considers the more general case where every stage of each batch has its own release time $rt_{is}$. This is a common situation in multistage batch scheduling problems where some production resources will be available after the time horizon has begun.

$$S_{is} \geq \sum_{j \in J_{is}} \mathrm{Max}[rt_i, ru_j + su_{ij}] Y_{ij} \qquad \forall i \in I, \quad s = s_i^f \tag{7a}$$

$$S_{is} \geq \sum_{j \in J_{is}} \mathrm{Max}[rt_{is}, ru_j + su_{ij}] Y_{ij} \qquad \forall i \in I, \quad s \in S_i \tag{7b}$$

In turn, Constraint (8) defines the tardiness of a batch based on its own due date and the completion time of the last processing stage.

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

If batch tardiness is not allowed ($T_i = 0$), then Constraint (8) reduces to Eq. (9) stating that each batch must finish before its promised due-date.

$$C_{is} \leq dd_i \qquad \forall i \in I, \quad s = s_i^\ell \tag{9}$$

### 4.7. Makespan definition

The time required to run all batches on the production facility is limited by the largest batch completion time. Constraint (10) defining the schedule *makespan MK* will be required when the minimum *MK* is the selected problem goal.

$$MK \geq C_{is} \qquad \forall i \in I, \; s = s_i^\ell \tag{10}$$

## 5. Additional continuous and discrete renewable resource constraints

Further sets of binary variables and constraints are required to deal with the limited capacity of discrete and continuous renewable resources. They are needed in order to avoid exceeding any resource capacity over the scheduling horizon. Let $RT_t = \{(i, s) | i \in I, s \in S_i : S_{is} \leq t < C_{is}\}$ be the set of running tasks $(i, s)$ at some time $t \in [0, H]$. If the available capacity $D_r$ of resource $r$ is not exceeded at any time $t$, then the following condition $\sum_{(i,s) \in RT_t} \rho_{isr} \leq D_r$ will be satisfied over the entire time interval $[0, H]$, where $\rho_{isr}$ is the $r$th-resource requirement of the running task $(i, s)$. Then, the LHS of the previous condition stands for the overall demand of resource $r$ by the set of tasks $(i, s)$ requiring $r$ and running in parallel at time $t$. Since the worst condition for overcapacity occurs just at the start of a processing task, it is better to control if enough free capacity of resource $r$ for running a new task $(i, s)$ requiring $r$ is available at its initial time $S_{is}$. In other words, the fulfilment of the resource constraints can be guaranteed by only monitoring the overall $r$th-resource demand at the starting time of any task $(i, s)$ with $\rho_{isr} > 0$.

### 5.1. Unit-independent task resource requirements

Resource capacity constraints based on the assumption that every task resource requirement is not unit-dependent will be first analyzed. In Section 5.3, such resource constraints are generalized to also consider the unit-dependent case. To determine the overall $r$th-resource demand at the start of a processing task $(i, s)$, it is necessary to identify the set of tasks requiring resource $r$ and running at time $S_{is}$ through additional 0-1 variables. Such required binary variables are grouped into two classes: *sequencing variables* and *overlapping variables*. To prevent from defining new sequencing variables, Constraints (3) and (4) are reformulated in such a way that the variables $X_{i's',is}$ can also control the relative ordering of parallel tasks $(i, s)$ and $(i', s')$ allocated to different equipment items.

As already mentioned, the idle capacity of a required resource $r$ must be checked before deciding to start a new task $(i, s)$ with $\rho_{isr} > 0$. To this end, it is important to know which other tasks $(i', s')$ consuming resource $r$ are running at time $S_{is}$. They will be called the set of overlapping tasks $OT_{is}$ for the task $(i, s)$ at its start time $S_{is}$. Moreover, $OT_{is} \subset RT_{t=Sis}$ since some running tasks at time $S_{is}$ may not require resource $r$. By definition, a task $(i', s') \in OT_{is}$ overlapping the starting task $(i, s)$ should satisfy the following four conditions (see Fig. 1):

(A) It has been assigned to a processing unit different from the one allocated to task $(i, s)$.
(B) It starts before ($S_{i's'} < S_{is}$) or exactly at time $S_{is}$ ($S_{i's'} = S_{is}$).
(C) It should end after time $S_{is}$, i.e. $C_{i's'} > S_{is}$.
(D) It should demand some resource $r \in R_{is}$ also required by task $(i, s)$.

The set $OT_{is}$ is just defined to detect the occurrence of resource overloads. If the set $OT_{is}$ has been determined, the following constraint will prevent the overall $r$th-resource requirement from exceeding the $r$th-resource capacity at the start of any task $(i, s)$ demanding resource $r$.

$$\rho_{isr} + \sum_{(i',s') \in OT_{is}} \rho_{i's'r} \leq D_r \qquad \forall i \in I, \quad s \in S_i, \quad r \in R_{is}$$

There will be as many $r$th-resource capacity constraints as the number of tasks $(i, s)$ requiring resource $r$.

In Fig. 1, the starting task $(i, s)$ is just overlapped by task $(i', s')$ beginning before time $S_{is}$ and ending after $S_{is}$. Then, if both demand resource $r$ the set $OT_{is}$ will include the task $(i', s')$ while $OT_{i's'}$ is an empty set ($S_{is} > S_{i's'}$). Let us assume that $\rho_{isr} < D_r$, $\rho_{i's'r} < D_r$ but $(\rho_{isr} + \rho_{i's'r}) > D_r$. Consequently, just the $r$th-capacity constraint for task $(i, s)$ will be violated, i.e. the one written for the task starting later. In the general case, the $r$th-capacity constraint for at least one of the overlapping tasks must be violated when the $r$th-resource capacity is overloaded. Let us now assume that the overlapping tasks $(i, s)$ and $(i', s')$ both demanding resource $r$ start at the same time and, in addition, $\rho_{isr} < D_r, \rho_{i's'r} < D_r$ but $(\rho_{isr} + \rho_{i's'r}) > D_r$. Since the purpose of the $r$th-capacity constraint set is to detect resource overloads and, by so doing, reject infeasible schedules, it is important that at least one of the $r$th-resource constraint for either task $(i, s)$ or task $(i', s')$ will be violated. Assuming arbitrarily that task $(i', s')$ began earlier whenever $i' < i$, the $r$th-capacity constraint for the "later" task $(i, s)$ is violated and, consequently, the proposed schedule will be infeasible. The reverse assumption will also work but this time the $r$th-capacity constraint for task $(i', s')$ cannot be satisfied. For either choice, we can conclude that the schedule is infeasible. The first choice $i' < i$ for handling overlapping tasks with the same start time will be used in this paper. When the overlapping tasks $(i, s)$ and $(i', s')$ demanding resource $r$ have the same starting times and $i' < i$, it will be assumed that task $(i', s')$ starts before so that the resource overload, if any, will be detected by the $r$th-capacity constraint for the overlapping task with the largest index $i$. This scheme for equal starting times can be applied whatever the number of overlapping tasks.

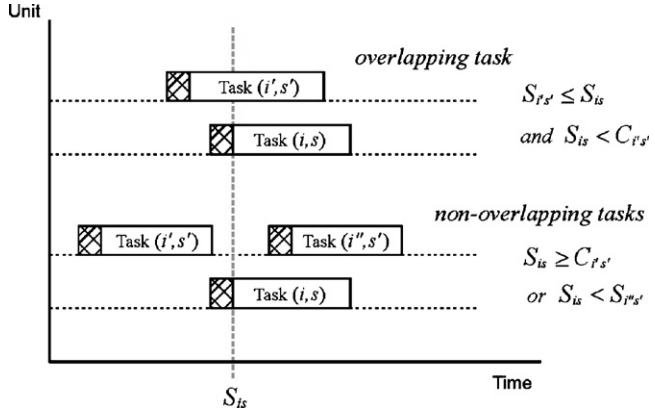



**Fig. 1.** Illustrating the task conditions for overlapping a starting task $(i, s)$.

In order to identify the elements of the set $OT_{is}$ fulfilling conditions (A)–(D), a pair of binary variables associated to task $(i, s)$ and each potential overlapping task $(i', s')$ sharing some resource $r \in R_{is}$ is to be defined. Since only a pair of tasks allocated to different units can overlap each other, conditions (B) and (C) imply the fulfilment of condition (A). If $J_{is} \cap J_{i's'} \neq \emptyset$ but the tasks $(i, s)$ and $(i', s')$ are not performed in the same unit, then the value of variable $X_{i's',is}$ becomes meaningless. To make the value of $X_{i's',is}$ still meaningful for parallel tasks to help monitor if task $(i', s')$ overlaps the starting task $(i, s)$, a more general definition of the variable $X_{i's',is}$ is introduced. In this way, the number of additional binary variables needed to identify the set of overlapping tasks for the starting task $(i, s)$ will be significantly decreased. Constraints (3) and (4) and the sequencing variable $X_{i's',is}$ are not required for tasks $(i, s)$ and $(i', s')$ if $s \neq s'$ or $J_{is} \cap J_{i's'} = \emptyset$. Since we should now also deal with the relative ordering of parallel tasks, the variable $X_{i's',is}$ will be incorporated in the mathematical model even if such a condition arises.

In case $i' < i$ and both tasks $(i, s)$ and $(i', s)$ are performed in the same unit, the condition $X_{i's,is} = 1$ means that $(i', s)$ precedes $(i, s)$ and consequently $C_{i's} \leq S_{is}$ and $S_{i's} \leq S_{is}$ are both true. Therefore, $X_{i's,is} = 1$ implies that task $(i', s)$ satisfies condition (B). If $i < i'$, $X_{is,i's} = 0$ also means that $(i', s)$ precedes $(i, s)$ and thus task $(i', s)$ satisfies condition (B). To extend the definition of $X_{i's',is}$ so that it also controls the relative ordering of the starting times of parallel tasks $(i, s)$ and $(i', s')$ and detect task overlapping, more general relaxed versions of Eqs. (3) and (4) in addition to Eqs. (3) and (4) themselves are included in the problem formulation. They are:

$$S_{i's'} - S_{is} \leq H(1 - X_{i's',is}) \qquad \forall i, i' \in I, \quad s \in S_i,$$
$$s' \in S_{i'} : (R_{is} \cap R_{i's'} \neq \emptyset) \wedge (i' < i) \tag{11}$$

$$S_{is} - S_{i's'} + \delta \leq H X_{i's',is} \qquad \forall i, i' \in I, \quad s \in S_i,$$
$$s' \in S_{i'} : (R_{is} \cap R_{i's'} \neq \emptyset) \wedge (i' < i) \tag{12}$$

Eqs. (11) and (12) are written for every pair of tasks sharing a resource $r \in R$ different from equipment. If tasks $(i, s)$ and $(i', s')$ have been allocated to the same equipment unit, Constraints (11) and (12) are automatically satisfied since they result from relaxing Eqs. (3) and (4) through the elimination of positive terms on the LHS. If not, Constraints (3) and (4) become redundant but the variable $X_{i's',is}$ still controls the relative ordering of the starting times of tasks $(i, s)$ and $(i', s')$ through Eqs. (11) and (12). In this way, it monitors the

fulfilment of condition (B) by task $(i', s')$ with regards to task $(i, s)$ and vice versa. If $S_{i's'} < S_{is}$ and $i' < i$, then task $(i', s')$ meets condition (B) with respect to task $(i, s)$ and $X_{i's',is} = 1$ to satisfy Eq. (11) and make redundant Constraint (12). In turn, if $S_{i's'} < S_{is}$ but $i < i'$, task $(i', s')$ still fulfils condition (B) but this time $X_{is,i's'} = 0$ to satisfy Eq. (12) and make redundant Constraint (11).

Constraint (12) includes a small positive scalar $\delta$ to cope with the case where tasks $(i, s)$ and $(i', s')$ start at the same time. A value much smaller than the lowest setup time is chosen for $\delta$. By including the model parameter $\delta$ in Constraint (12), $X_{i's',is}$ must equal 1 whenever $S_{is} = S_{i's'}$ and $i' < i$. In this way, Constraint (11) is satisfied while Eq. (12) becomes redundant. The other option $X_{i's',is} = 0$ is not a feasible choice since Constraint (12) leads to the inequality: $S_{is} \le S_{i's'} - \delta$. As previously discussed, the task with the largest index $i$ is assumed to start last and the corresponding set $OT_{is}$ will comprise all the other parallel tasks starting at the same time and sharing the same resource item $r$. If $r$ is the bottleneck resource, the $r$th-capacity constraint for the task $(i, s)$ with the largest index $i$ will verify if the overall requirement of resource $r$ by the whole set of overlapping tasks exceeds the available capacity.

In order to write the mathematical expression for the overlapping condition (C), let us define a new binary variable $W_{i's',is}$ that must be equal to one whenever the stage $s'$ of batch $i'$ is completed after starting stage $s$ of batch $i$. Variables $W_{i's',is}$ are called *overlapping variables*. Therefore,

$$C_{i's'} - S_{is} \le H\,W_{i's',is} \qquad \forall\, i,\ i' \in I, \quad s \in S_i,$$
$$s' \in S_{i'} : (R_{is} \cap R_{i's'} \neq \emptyset) \wedge (i \neq i') \tag{13}$$

When $C_{i's'} > S_{is}$, condition (C) is satisfied and the LHS of Constraint (13) becomes positive. As a result, the variable $W_{i's',is}$ is forced to equal 1. Therefore, the pair of values $X_{i's',is} = 1$ and $W_{i's',is} = 1$ (assuming $i' < i$) indicates the fulfilment of conditions (B) and (C). In other words, such values mean that the task $(i', s')$ starts before $(i, s)$ and is still running at time $S_{is}$. If so, it is said that task $(i', s')$ overlaps the starting task $(i, s)$. It is important to note, however, that task $(i, s)$ does not overlap $(i', s')$ since it starts after time $S_{i's'}$. Despite they run in parallel during some period of time, the task $(i, s)$ does not fulfill condition (B) to be an overlapping task of $(i', s')$. Therefore, an overlapping task is always a parallel task but the reverse statement is not always true. Similarly, tasks running at the start of task $(i', s')$ are identified by also considering the variable $W_{is,i's'}$.

In contrast to variables $X_{i's',is}$ defined only for $i' < i$, both variables $W_{is,i's'}$ and $W_{i's',is}$ must be defined for any pair of processing tasks $(i, s)$ and $(i', s')$ that can be allocated to a common renewable resource $r$. Notice that two tasks involving the same batch $i$ can never be run in parallel. In other words, the task $(i, s)$ can never overlap another stage $s'$ of batch $i$. If two tasks $(i, s)$ and $(i', s')$ are running in parallel the Boolean condition $(S_{is} < C_{i's'}) \wedge (S_{i's'} < C_{is})$ is satisfied and therefore it is true that $W_{i's',is} + W_{is,i's'} = 2$. However, it is important to keep in mind that not every parallel task $(i', s')$ satisfying the necessary condition $W_{i's',is} + W_{is,i's'} = 2$ is an overlapping task for $(i, s)$, but just those running at the starting time $S_{is}$. Given the condition $W_{i's',is} + W_{is,i's'} = 2$, the variable $X_{i's',is}$ is required to decide which task $(i, s)$ or $(i', s')$ overlaps the other one. In summary, assuming that $i' < i$ the task $(i', s')$ is an overlapping task for $(i, s)$ only if $X_{i's',is} = 1$ and $W_{i's',is} = 1$. If $i < i'$ then the task $(i', s')$ is an overlapping task for $(i, s)$ only if $X_{is,i's'} = 0$ and $W_{i's',is} = 1$. As a result, the set of overlapping tasks for the starting task $(i, s)$ can be defined as:

$$OT_{is} = \{(i', s') | i' \in I, s' \in S_{i'} : [(i' < i \wedge X_{i's',is} = 1) \vee (i' > i \wedge X_{is,i's'} = 0)]$$
$$\wedge W_{i's',is} = 1 \wedge R_{is} \cap R_{i's'} \neq \emptyset\}, \quad i \in I, \quad s \in S_i$$

It can be easily proven that $W_{i's',is} + X_{i's',is} \ge 1$ (if $i' < i$) and $W_{i's',is} - X_{is,i's'} \ge 0$ (if $i < i'$) for any pair of tasks $(i, s)$ and $(i', s')$. Then, the constraint preventing from exceeding the $r$th-resource capacity at the start time of any task $(i, s)$ is given by:

$$\rho_{isr} + \sum_{i' \in I : i' \neq i} \sum_{s' \in S_{i'} : r \in R_{i's'}} \left[ \sum_{i' < i} \rho_{i's'r} (W_{i's',is} + X_{i's',is} - 1) + \sum_{i < i'} \rho_{i's'r} (W_{i's',is} - X_{is,i's'}) \right] \le D_r \quad \forall\, i \in I, \quad s \in S_i, \quad r \in R_{is} \tag{14}$$

According to condition (B), the task $(i', s')$ belongs to the set $OT_{is}$ only if $S_{i's'} \le S_{is}$ with the equality just in case $i' < i$. As previously discussed, if both tasks start at the same time the one related to the batch arising first in the set $I$ (i.e. the one with the lower $i$) is assumed to begin earlier.

### 5.2. Improving the convergence rate of the solution algorithm

To speed up the solution algorithm, assuming $i' < i$, some additional relationships between variables $X_{i's',is}$, $W_{i's',is}$ and $W_{is,i's'}$ can be derived. If $X_{i's',is} = 1$, then $S_{i's'} \le S_{is} < C_{is}$, and consequently $C_{is} - S_{i's'} > 0$ and $W_{is,i's'} = 1$. Similarly, if $X_{i's',is} = 0$ then $S_{is} < S_{i's'} < C_{i's'}$ (or equivalently $C_{i's'} - S_{is} > 0$), and consequently $W_{i's',is} = 1$. Therefore,

$$X_{i's',is} \le W_{is,i's'} \quad \forall\, i, i' \in I, s \in S_i, s' \in S_{i'} : (R_{is} \cap R_{i's'} \neq \emptyset) \wedge (i' < i) \tag{15}$$

$$(1 - X_{i's',is}) \le W_{i's',is} \quad \forall\, i, i' \in I, s \in S_i, s' \in S_{i'} : (R_{is} \cap R_{i's'} \neq \emptyset) \wedge (i' < i) \tag{16}$$

In order to further improve the convergence rate to the optimal solution, the sequencing Constraints (11) and (12) that are relevant for processing tasks executed in different units can be re-written in the following way:

$$S_{i's'} + pt^{Min}_{i's'}(1 - W_{i's',is}) \le S_{is} + H(1 - X_{i's',is}) \qquad \forall\, i,\ i' \in I, \quad s \in S_i,$$
$$s' \in S_{i'} : (R_{is} \cap R_{i's'} \neq \emptyset) \wedge (i' < i) \tag{17}$$

$$S_{is} + \delta + (pt^{Min}_{is} - \delta)(1 - W_{is,i's'}) \le S_{i's'} + H\,X_{i's',is} \qquad \forall\, i,\ i' \in I,$$
$$s \in S_i, \quad s' \in S_{i'} : (R_{is} \cap R_{i's'} \neq \emptyset) \wedge (i' < i) \tag{18}$$

so as to get more tightening restrictions by incorporating the non-negative term $pt^{Min}_{i's'}(1 - W_{i's',is})$ on the LHS, where: $pt^{Min}_{i's'} = \underset{j \in J_{i's'}}{\text{Min}}[pt_{i'j}]$. If task $(i', s')$ overlaps the starting task $(i, s)$ and $i' < i$, then $X_{i's',is} = 1$ ($S_{i's'} < S_{is}$), $W_{i's',is} = 1$ ($C_{i's'} > S_{is}$) and the additional term in Constraint (17) is driven to zero. As a result, Eq. (17) reduces to Eq. (11). Similarly to Eq. (12), Constraint (18) becomes redundant. On the other hand, if task $(i, s)$ overlaps task $(i', s')$ and $i' < i$, then Constraints (12) and (18) both become equivalent and Inequalities (11) and (17) turn to be redundant. Therefore, the Constraints (11) and (12) can be replaced by Eqs. (17) and (18) since they are equivalent. To see the tightening effect of the additional non-negative term, let us assume that the task $(i', s')$ overlaps task $(i, s)$ and $X_{i's',is} = 1$ ($i' < i$) but the overlapping has not yet been detected by the solution algorithm because it does not yet branch on the variable $W_{i's',is}$. As a result, $W_{i's',is}$ takes a fractional value and the additional term on the LHS of Eq. (17) forces such a value to be sufficiently close to one so that Constraint (17) is satisfied. The closer the starting times of tasks $(i, s)$ and $(i', s')$ the higher the effect of the additional term on the level of $W_{i's',is}$. In this way, the overlapping is earlier detected.

### 5.3. Unit-dependent task resource requirements

The resource capacity constraints introduced in Section 5.1 should be generalized to deal with the more complex case involving

unit-dependent task resource requirements. A new set of resource capacity constraints that still includes the overlapping variables and constraints defined in Section 5.1 is introduced. For each task $(i, s)$ demanding resource $r$, additional continuous variables $Q_{isjr}$ are defined to monitor the $r$th-resource utilization on each unit $j$ where a potential overlapping task also requiring resource $r$ can be allocated. In spite of its higher model size, it has been found that the upgraded unit-dependent problem formulation shows a computational performance similar to the unit-independent case.

Let $Q_{isjr}$ be a non-negative continuous variable that represents the amount of resource $r \in R_{is}$ required by a parallel task $(i', s')$ running at the start time of $(i, s)$ and being performed in unit $j$. Since every equipment unit can process tasks one by one, it is expected that at most a single task is running in unit $j$ at time $S_{is}$. If task $(i', s')$ is the one being performed, then no other task can be allocated to unit $j$ at time $S_{is}$. Therefore, $Q_{isjr}$ will be equal to the amount of resource $r$ required by a task overlapping the task $(i, s)$ and running in unit $j$. If no overlapping task is performed in unit $j$ at time $S_{is}$, then $Q_{isjr} = 0$. Constraints (19) and (20) are introduced to define the new variables $Q_{isjr}$ in terms of sequencing $(X_{i's',is})$, allocation $(Y_{ij})$ and overlapping variables $(W_{i's',is})$.

$$[W_{i's',is} + X_{i's',is} + Y_{i'j} - 2]\rho_{i's'jr} \le Q_{isjr} \quad \forall i,\ i' \in I,\quad s \in S_i,\quad s' \in S_{i'},\quad j \in J_{i's'},\quad r \in (R_{is} \cap R_{i's'}) : (i' < i) \tag{19}$$

$$[W_{i's',is} - X_{is,i's'} + Y_{i'j} - 1]\rho_{i's'jr} \le Q_{isjr} \quad \forall i,\ i' \in I,\quad s \in S_i,\quad s' \in S_{i'},\quad j \in J_{i's'},\quad r \in (R_{is} \cap R_{i's'}) : (i < i') \tag{20}$$

Notice that the terms between square brackets involve binary variables, and can take any value from the set $\{-1, 0, 1\}$. If the bracket is equal one, the variable $Q_{isjr}$ is constrained from below by the resource requirement $\rho_{i's'jr}$ of task $(i', s')$ running in unit $j$ and overlapping task $(i, s)$. Otherwise, Constraints (19) or (20) becomes redundant since the LHS is lower or equal to zero. In short, two possible scenarios arise depending on whether or not an overlapping task $(i', s')$ is performed in unit $j$. On one hand, $Q_{isjr}$ is made equal to $\rho_{i's'jr}$ by the model if task $(i', s')$ belongs to $OT_{is}$ and is allocated to unit $j$. If not, $Q_{isjr} = 0$. When the task resource consumption depends on the allocated unit, the resource capacity constraint preventing overloading of resource $r$ at the start time of any task $(i, s)$ can be formulated in terms of the new variables $Q_{isjr}$ as follows:

$$\left(\sum_{j \in J_{is}} \rho_{isjr} Y_{ij}\right) + \sum_{j \in J^P_{isr}} Q_{isjr} \le D_r \quad \forall i \in I,\quad s \in S_i,\quad r \in R_{is} \tag{21}$$

where $J^P_{isr} = \{j | i' \in I, s' \in S_{i'}, j \in J_{i's'} : i \ne i' \wedge r \in R_{i's'}\}$, $\forall i \in I, s \in S_i, r \in R_{is}$, is the set of units where a potential parallel task $(i', s')$ requiring resource $r$ and eventually overlapping task $(i, s)$ can be performed. The first term on the LHS of Constraint (21) provides the unit-dependent $r$th-resource requirement of task $(i, s)$, while the second term is the total consumption of resource $r$ by the set of parallel tasks different from $(i, s)$ running at the start time $S_{is}$ and requiring $r$. In turn, the right side of Constraint (21) is the total $r$th-resource capacity that is assumed to remain constant over time. Notice that the sets $J_{is}$ and $J^P_{isr}$ are generally different since the latter one may include units belonging to other stages $s' \ne s$. Moreover, $Q_{isjr}$ is automatically set to zero if unit $j \in (J_{is} \cap J^P_{isr})$ performs the task $(i, s)$.

In this way, the approach just needs a small number of additional continuous variables but no further binary variables to handle unit-dependent resource requirements. Though a larger set of constraints is to be considered, it is still smaller than the one included in other resource constrained formulations (Janak et al., 2004; Pinto & Grossmann, 1997).

To handle discrete/continuous renewable resources aside from equipment, Constraints (13) and (15)–(18) are always incorporated in the mathematical model. However, the expression of the resource capacity constraint depends on whether or not the task resource requirement is a function of the allocated unit. When it does not change with the unit, Constraint (14) is to be further considered. Otherwise, Constraints (19)–(21) are to be included to prevent overlapping tasks from exceeding the $r$th-resource capacity. In summary, the problem constraint set always includes Eqs. (1)–(6), (7a), (13)–(18) if the task resource requirement $(\rho_{isr})$ does not depend on the assigned equipment unit. Moreover, one of the Constraints (8)–(10) should also be considered depending on the selected objective function. Otherwise, when $\rho_{isjr}$ varies with the allocated unit $j$, then the constraint set includes Eqs. (1)–(6), (7a), (13), (15)–(21) plus one of the Constraints (8)–(10) related to the problem objective.

### 5.4. Extending the mathematical formulation for multipurpose batch plants

The proposed model can be generalized to also account for multipurpose batch plant configurations in a straightforward manner. To this end, a pair of changes are required: (i) it should be used the allocation variable $Y_{isj}$ rather than $Y_{ij}$ since unit $j$ may perform more than a single stage on any batch $i$. In this way, the number of allocation variables may increase; (ii) it is necessary to incorporate the sequencing variables $(X_{i's',is})$ and formulate the sequencing constraints using the concept of general precedence for any pair of tasks $(i, s)$ and $(i', s')$ involving different batches and stages that can be allocated to the same unit. As a result, Constraints (3) and (4) should be written for any $(i, i') \in I\ (i' < i)$, $s \in S_i$, $s' \in S_{i'}$, $j \in J_{is,i's'}$.

## 6. Alternative objective functions

The proposed MILP mathematical model for the resource-constrained multistage batch scheduling problem allows to easily handle different types of objective functions. The Problem Objective (22) is used to minimize the weighted overall earliness for a multistage batch facility, assuming that enough production capacity is available to timely meet all batch due dates. A positive weight parameter $\alpha_{is}$ for every task $(i, s)$ should be chosen. Usually, it represents the inventory cost of either intermediate products between consecutive stages or final products coming from the last stage.

$$\text{Minimize} \sum_{i \in I} \left( \alpha_{is^\ell_i}(dd_i - C_{i,s^\ell_i}) + \sum_{s \in S_i - \{s^\ell_i\}} \alpha_{is}(S_{i,s+1} - C_{is}) \right) \tag{22}$$

In particular, the problem of minimizing the overall earliness is equivalent to the problem of maximizing the summation of the task completion times for a single-stage batch plant.

$$\text{Minimize} \sum_{i \in I} \alpha_i(dd_i - C_i) \equiv \text{Maximize} \sum_{i \in I} \alpha_i C_i \tag{23}$$

Together with Problem Objective (22) or (23), Constraint (9) must be used in order to properly meet the specified due dates. If the problem goal is to minimize the overall tardiness, the mathematical formulation should include the constraint (8) and the objective function will be given by,

$$\text{Minimize} \sum_{i \in I} \beta_i T_i \tag{24}$$

where $\beta_i$ is the weight coefficient for batch $i$. When the minimization of the makespan is the problem goal, the Constraint (10)

defining the makespan *MK* should be considered and the problem objective will be given by the following equation:

$$\text{Minimize } MK \tag{25}$$

## 7. Results and discussion

The proposed resource-constrained batch scheduling formulation has been applied to four examples all involving limited renewable resources. Examples 1 and 2 were previously studied by other authors while Examples 3 and 4 are new case studies with a pair of additional features: processing tasks with unit-dependent resource requirements and multiple bottleneck resources. All the examples were solved to optimality using ILOG OPL Studio 3.6 (ILOG OPL Studio, 2002) with the solver CPLEX v. 8.0 on a Pentium IV 1.8 GHz with 1 Gb of memory. Default solver options (1E−6 relative gap tolerance) have been used on every example, and a CPU time limit of 1 h was adopted.

### 7.1. Example 1

This example introduced by Pinto and Grossmann (1997) and subsequently studied by other authors (Janak et al., 2004; Méndez & Cerdá, 2002) deals with a plastic compounding facility with a single processing stage and 4 extruders running in parallel. The extruders have different capacities and their processing rates depend on the product being manufactured. A total of 12 single-batch orders should be scheduled over a time horizon of 30 days. Specific order due dates are to be satisfied but the plant capacity is large enough to meet all of them on time. Therefore, the problem goal is to minimize the overall order earliness. Processing times, due dates, and unit-dependent setup times are all presented in Table 1. Assuming that a single worker is required to run each extruder, three different versions of Example 1 have been solved. While Example 1a assumes unlimited available manpower, the other two instances of Example 1 involve a strong constraint on the number of extruders running in parallel by considering a limited manpower capacity. Example 1b assumes that at most 3 units can be run simultaneously, whereas a maximum of 2 extruders can work in parallel in Example 1c. The non-constrained Example 1a has recently been solved by Castro and Grossmann (2006) using five conceptually different formulations, one of which is relying on global precedence sequencing variables like the one proposed here. The following values for the model parameters were selected: $H = \underset{i \in I}{\text{Max}}[dd_i] = 30$ and $\delta = 0.001$.

Fig. 2 shows the optimal schedules found for the three instances of Example 1. Tasks not being completed just-in-time are depicted darker in order to appreciate the effect of manpower limitations. Alternative optimal solutions have already been reported by Méndez and Cerdá (2002) and Janak et al. (2004).

**Table 1**
Data for Example 1.

| Order | Due date (days) | Processing time (days) | | | |
|---|---|---|---|---|---|
| | | $U_1$ | $U_2$ | $U_3$ | $U_4$ |
| 1 | 15 | 1.538 | | | 1.194 |
| 2 | 30 | 1.500 | | | 0.789 |
| 3 | 22 | 1.607 | | | 0.818 |
| 4 | 25 | | | 1.564 | 2.143 |
| 5 | 20 | | | 0.736 | 1.017 |
| 6 | 30 | 5.263 | | | 3.200 |
| 7 | 21 | 4.865 | | 3.025 | 3.214 |
| 8 | 26 | | | 1.500 | 1.440 |
| 9 | 30 | | | 1.869 | 2.459 |
| 10 | 29 | | 1.282 | | |
| 11 | 30 | | 3.750 | | 3.000 |
| 12 | 21 | | 6.796 | 7.000 | 5.600 |
| Unit setup time | | 0.180 | 0.175 | | 0.237 |

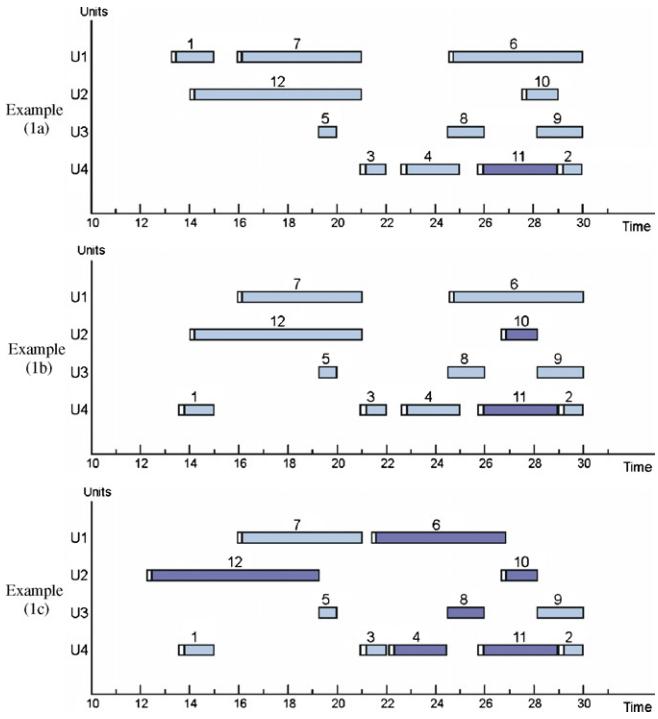



**Fig. 2.** Optimal solutions found for Example (1a) without resource constraints, (1b) at most 3 parallel units and (1c) at most 2 parallel units.

Model sizes, optimal objective values, CPU times and number of explored nodes for the three instances of Example 1 are all included in Table 2. Such results are not only given for the proposed approach but also for other resource-constrained scheduling methodologies (Janak et al., 2004; Méndez & Cerdá, 2002; Pinto & Grossmann, 1997). Results found with the mathematical model of Méndez and Cerdá (2002) and the proposed approach were both obtained using the same computer and solver described at the beginning of this Section.

As shown in Table 2, both methodologies of Pinto and Grossmann (1997) found a near-optimal solution for Examples 1b and 1c by considering preordering constraints and using the summation of the order starting times as the problem objective to be maximized. Their formulation presents an important number of binary variables and constraints and a large computational requirement even for the non-constrained instance of Example 1. Though the computer platform used by Pinto and Grossmann (1997) is not comparable with the one applied in this work, it can be concluded from: (i) the extremely high number of explored nodes, (ii) the CPU time of 21 h for the most constrained instance 1c and (iii) the need of preordering conditions to reduce the model size, that such a scheduling approach is computationally less efficient than the proposed formulation. In turn, the MILP model of Méndez and Cerdá (2002) shows a good computational performance. When Example 1a is considered, this model and the mathematical representation introduced in this work both become equivalent to the one proposed by Méndez et al. (2001), since no renewable resources different from equipment are considered. Interestingly, the formulation of Méndez and Cerdá (2002) presents a remarkable reduction in binary variables and constraints for Example 1c with regards to Example 1b. Since the approach is based on the allocation of resource items to tasks, such a decrease in the model size arises simply because Example 1b has three resource items (workers) available to allocate to tasks while case 1c has only two. Despite that, a higher CPU time for the more restricted case 1c that

**Table 2**
Model sizes and computational results for Example 1.

| Model | Case | Binary variables, cont. variables, constraints | Objective function | CPU time (s) | Nodes |
|---|---|---|---|---|---|
| Pinto and Grossmann (1997) (MILP formulation) | 1a | 100, 117, 153 | 269.10 # | 71.53 a | 1072 |
| | 1b † | 289, 329, 1156 | 268.24 # | 2224 a | 1941 |
| | 1c † | 289, 329, 1156 | 264.98 # | 76390 a | 99148 |
| Pinto and Grossmann (1997) (MILP + DP) | 1a † | – | 269.10 # | 63.56 a | 283 |
| | 1b † | – | 268.24 # | 125.42 a | 673 |
| | 1c † | – | 264.98 # | 927.16 a | 7341 |
| Méndez and Cerdá (2002) | 1a | 82, 24, 214 | 1.026 | 0.03 | 12 |
| | 1b | 127, 24, 622 | 1.895 | 2.93 | 4372 |
| | 1c | 115, 24, 490 | 7.334 | 13.28 | 21583 |
| Janak et al., 2004 | 1a | 150, 513, 1389 | 1.026 | 0.07 b | 7 |
| | 1b | 458, 2137, 10382 | 1.895 | 6.53 b | 1374 |
| | 1c ‡ | 444, 2137, 10382 | 7.334 | 236.87 b | 38621 |
| Proposed Formulation | 1a | 82, 24, 214 | 1.026 | 0.03 | 12 |
| | 1b | 223, 24, 622 | 1.895 | 0.66 | 121 |
| | 1c | 223, 24, 622 | 7.334 | 7.39 | 3943 |

† With preordering constraints.
# Problem goal: max $\Sigma_{i\in I} S_i$.
‡ Reported in Janak, Lin, and Floudas (2005).
a IBM 6000-530 w/GAMS/OSL.
b 3.00 GHz Linux workstation w/GAMS 2.50/CPLEX 8.1.

almost doubles the one required by our formulation is needed (see Table 2).

On the other hand, the model of Janak et al. (2004) uses utility-related tasks and unit-specific time events to detect the starting time of every resource usage. Since 12 event points for cases 1b and 1c should be adopted to guarantee optimality, the resulting problem formulation includes a relatively large number of binary/continuous variables and constraints and consequently the computational time significantly rises. Despite using a faster computer, the model of Janak et al. (2004) requires a CPU time substantially larger than those reported for general precedence based models. Therefore, the proposed MILP formulation presents a reasonable size in terms of binary variables and constraints and a good computational performance for every instance of Example 1. It is by far the best one when the manpower capacity drops from 3 to 2 workers. Since just the manpower capacity $D_r$ (a model parameter) is diminished, the number of binary/continuous variables and constraints for cases 1b and 1c remain the same.

### 7.2. Example 2

Example 2 originally introduced by Pinto and Grossmann (1995) involves the processing of five single-batch orders in a multistage multiproduct batch plant producing dyes. The production facility consists of 25 equipment units distributed among five processing stages. The presence of limited interconnections between consecutive stages (topology constraints) reduces the number of feasible unit-task assignments. Topology constraints arise between consecutive stages I–II and IV–V, respectively. The plant layout is shown in Fig. 3.

An unlimited intermediate storage (UIS) capacity between consecutive stages is assumed to be available. Unit-dependent processing and setup times for each production order at each stage are listed in Table 3. Furthermore, Table 4 shows the steam requirements for stages I (reaction) and IV (drying) taken from the work of Méndez and Cerdá (2003). Such steam requirements do not depend on the allocated unit. Three different instances of Example 2 have been defined by changing the maximum available steam flow rate. Example 2a assumes a maximum steam flow of 25 ton/h, that is reduced to 23 ton/h in Example 2b. In turn, the steam flow is further constrained to a maximum of 22 ton/h in Example 2c. Examples 2a and 2b have been previously tackled by Méndez and Cerdá (2003). The selected problem goal is the minimum makespan.

The following values for the parameters were adopted: $H = 400$ and $\delta = 0.1$. Model sizes and computational results obtained through both the proposed approach and the formulation of Méndez and

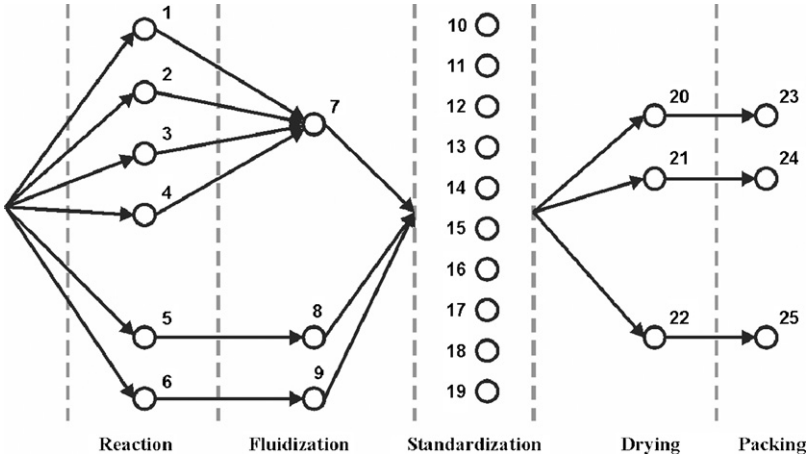



**Fig. 3.** Multistage plant structure for Example 2.

**Table 3**
Processing and transition times for Example 2.

| Order | Equipment | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Reaction | Fluidization | | Standardization | | Drying | | Packing | |
| | $U_1 - U_6$ | $U_7$ | $U_8 - U_9$ | $U_{10} - U_{17}$ | $U_{18} - U_{19}$ | $U_{20} - U_{21}$ | $U_{22}$ | $U_{23} - U_{24}$ | $U_{25}$ |
| 1 | 18.1 | 14.0 | 5.0 | 12.0 | | 9.5 | | 24.0 | |
| 2 | 23.0 | 14.0 | 5.0 | 12.0 | 12.0 | | 100.0 | | 48.0 |
| 3 | 18.1 | 14.0 | 5.0 | 24.0 | | 9.3 | | 24.0 | |
| 4 | 20.0 | 11.0 | 5.0 | 12.0 | | 7.9 | | 24.0 | |
| 5 | 17.0 | 14.0 | 5.0 | 12.0 | | 12.5 | | 24.0 | |
| Setup time | 8.0 | 8.0 | 1.0 | 2.5 | 2.5 | 6.0 | 24.0 | 4.0 | 5.0 |

**Table 4**
Order steam requirements (in ton/h) for Example 2.

| Stage | Order | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| Reaction ($U_1 - U_6$) | 8.0 | 9.0 | 9.5 | 8.5 | 7.0 |
| Drying ($U_{20} - U_{22}$) | 11.0 | 14.0 | 12.0 | 11.0 | 9.0 |

Cerdá (2003) are presented in Table 5. In order to make a fair comparison, both methodologies were applied to the three instances of Example 2 using the same solver and computer platform described at the beginning of this Section. The approach of Méndez and Cerdá (2003) requires an additional parameter denoting the maximum number of pieces into which the steam source can be divided. For all instances of Example 2 three steam subsources were defined.

Overall, the proposed model performs very well for all instances of Example 2. In particular, it is clearly the best one for the most restrictive instance 2c where the steam flow rate has been limited to 22 ton/h. It converges to the optimal solution in 424.28 s. In contrast, the model of Méndez and Cerdá (2003) was unable to solve Example 2c to optimality after the time limit of 1 h. Besides, the CPU time reported by Méndez and Cerdá (2003) for Example 2b and shown in Table 5 could not be replicated using the current solver/machine configuration, although the same model definition file was used. Nonetheless, the CPU time included in Table 5 is the one reported in their original work. Though both approaches provide the same best solution for Examples 2a and 2b, a different problem representation has been adopted. The proposed methodology does not need to partition a continuous renewable resource into several pieces (or sub-sources) in order to handle it as a set of elementary discrete resources. Instead, it uses a rigorous problem representation that guarantees in any case the discovery of the optimal resource-constrained schedule. In the other approach (Méndez & Cerdá, 2003), the number of pieces should be large enough to guarantee solution optimality. However, the higher the number of pieces the larger the model size and the computational requirement.

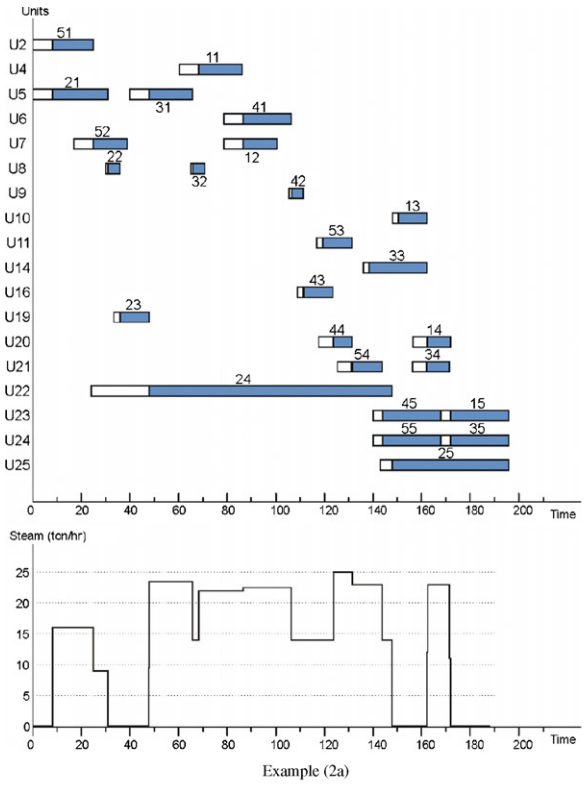
As shown in Table 5, the proposed mathematical formulation includes a total of 251 binary variables: 105 assignment variables ($Y_{ij}$), 80 overlapping variables ($W_{i's',is}$), and 66 sequencing variables ($X_{i's',is}$). The sequencing variables are distributed as follows: (i) 26 were just used to control the sequencing of tasks on stages $S_2$, $S_3$ and $S_5$; (ii) 16 were defined to account for tasks that can be allocated to the same equipment unit on stages $S_1$ and $S_4$ but reused by the model to control task overlapping, in combination with variables $W_{i's',is}$; (iii) the remaining 24 were needed to detect the overlapping of parallel tasks that cannot share the same equipment unit. These specific binaries $X_{i's',is}$ are to be defined for: (a) the pairs of tasks $(O_2, S_4) - (O_i, S_4)$, $\forall i \neq 2$, all carried out in the drying stage; (b) the pairs of processing tasks performed in stages I and IV, $(O_i, S_1) - (O_{i'}, S_4)$, $\forall\ i, i' \in I\ (i \neq i')$. Such pairs of processing tasks all require steam (a common resource) but do not have any equipment unit in common.

The optimal schedules found for all instances of Example 2 are depicted in Fig. 4 together with the steam usage profiles. When the less constrained case (Example 2a) is solved, the optimal value still remains at the same level (196 h) found for the non-constrained case and, therefore, the steam flow does not arise as a bottleneck resource. It becomes a limiting resource when it is reduced to a maximum of 23 ton/h (Example 2b). As a result, the overlapping of task $(O_3, S_1)$ at the start time of $(O_2, S_4)$ and the running of $(O_2, S_4)$ at the beginning of $(O_4, S_4)$ are both no longer permitted since their overall consumption of 23.5 and 25 ton/h, respectively, exceeds the available steam flow. Then, the starting time of $(O_2, S_4)$ is delayed and the remaining tasks are properly reaccommodated to meet the resource constraints. Consequently, the optimal makespan increases from 196 to 200.6 (see Fig. 4). A more restrictive steam flow capacity of 22 ton/h makes the optimal schedule of Example 2b an infeasible solution. Both tasks $(O_5, S_4)$ and $(O_4, S_1)$ can no longer start during the execution of the overlapping task $(O_2, S_4)$ because the overall steam requirement reaches 23 and 22.5 ton/h respectively. Moreover, task $(O_3, S_4)$ can no longer overlap $(O_4, S_4)$. As a result, unit reassignments and changes in the timing of several tasks with regards to the best solution of Example 2b should be made. In particular, task $(O_4, S_1)$ is scheduled to start at the beginning of the time horizon while task $(O_5, S_5)$ is completed at the end of the new optimal schedule (see Fig. 4). Besides, tasks $(O_4, S_4)$ and $(O_3, S_4)$ are now sequentially performed in unit U20.

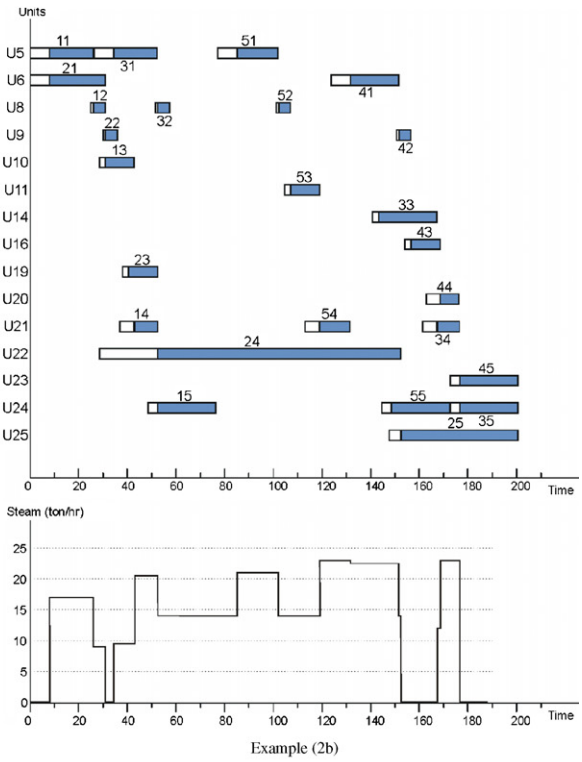
**Table 5**
Model size and computational results for Example 2.

| Model | Case | Binary variables, cont. variables, constraints | Objective function | CPU time (s) | Nodes |
|---|---|---|---|---|---|
| Méndez and Cerdá (2003) | 2a | 201, 84, 817 | 196 | 0.88 | 258 |
| | 2b | 201, 84, 817 | 200.6 | 13.07[#] | – |
| | 2c | 201, 84, 817 | 209.5 [a] | 3600[†] | 2,024,341 |
| Proposed formulation | 2a | 251, 51, 756 | 196 | 0.80 | 130 |
| | 2b | 251, 51, 756 | 200.6 | 72.72 | 28,160 |
| | 2c | 251, 51, 756 | 209.5 | 424.28 | 196,837 |

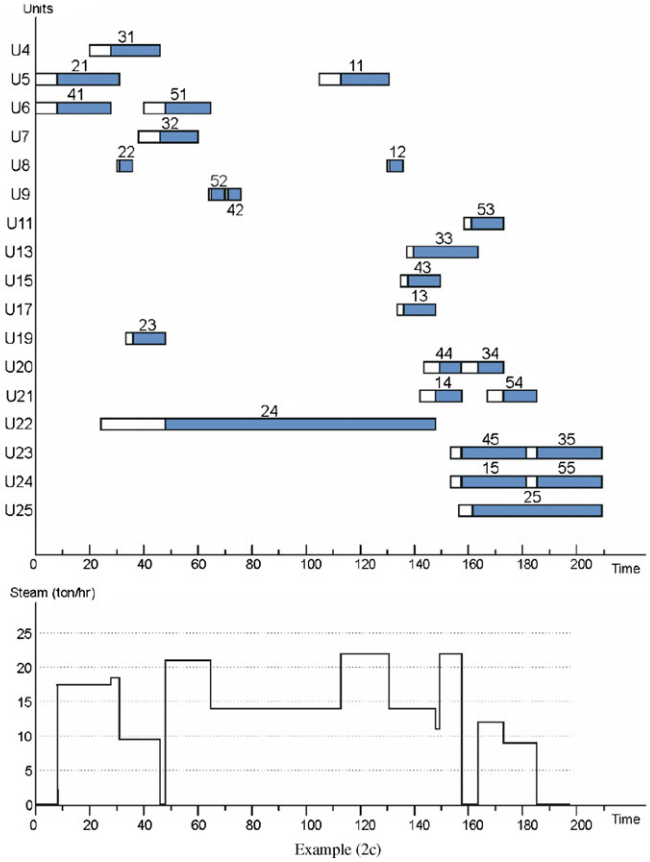
[#] Seconds on Pentium III PC (933 MHz) with ILOG/CPLEX.
[†] Time limit exceeded.
[a] Lower bound = 196, relative gap tolerance = 6.4%.




Example (2a)




Example (2b)




Example (2c)

**Fig. 4.** Optimal solutions and steam demand profiles for Examples 2a–2c.

### 7.3. Example 3

Example 3 is a modified version of Example 1 that results by considering unit-dependent manpower requirements (see Table 6). The other problem data are already given in Table 1. Two instances of Example 3 were defined by changing the level of the available manpower. A total of 5 workers is assumed to be available in Example 3a while only 4 workers are on hand in Example 3b. The problem goal is to minimize the total earliness. Values for model parameters $\delta$ and $H$ are those chosen for Example 1.

In order to compare their computational performance, the proposed model and the formulation of Méndez and Cerdá (2002) were both applied to the two instances of Example 3. Computational results are presented in Table 7. The approach of Méndez and Cerdá (2002) needs to define 5 and 4 unary resources in order to handle Examples 3a and 3b, respectively. Though it requires less binary variables and constraints, the model of Méndez and Cerdá (2002) was unable to discover the optimal schedule for Example 3a and prove the optimality of the best solution found for both Examples 3a and 3b after the time limit of 1 h. In contrast, the proposed approach solves both instances of Example 3 in few CPU seconds. The corresponding Gantt charts and manpower profiles are presented in Fig. 5. Tasks not being completed just-in-time are depicted darker. Despite the production schedule should be developed under more stringent conditions, the CPU time for Example 3b does not present a substantial increase compared with Example 3a. Moreover, the fact that manpower requirements are now unit-dependent has a little effect on the CPU time (see Tables 2 and 7). To cope with unit-dependent resource requirements, 48 continuous variables and 275 constraints with regards to the unit-independent MILP formulation are further needed. If the optimal manpower level is to be established, the problem goal will aim to minimize the sum of manpower and product storage costs.

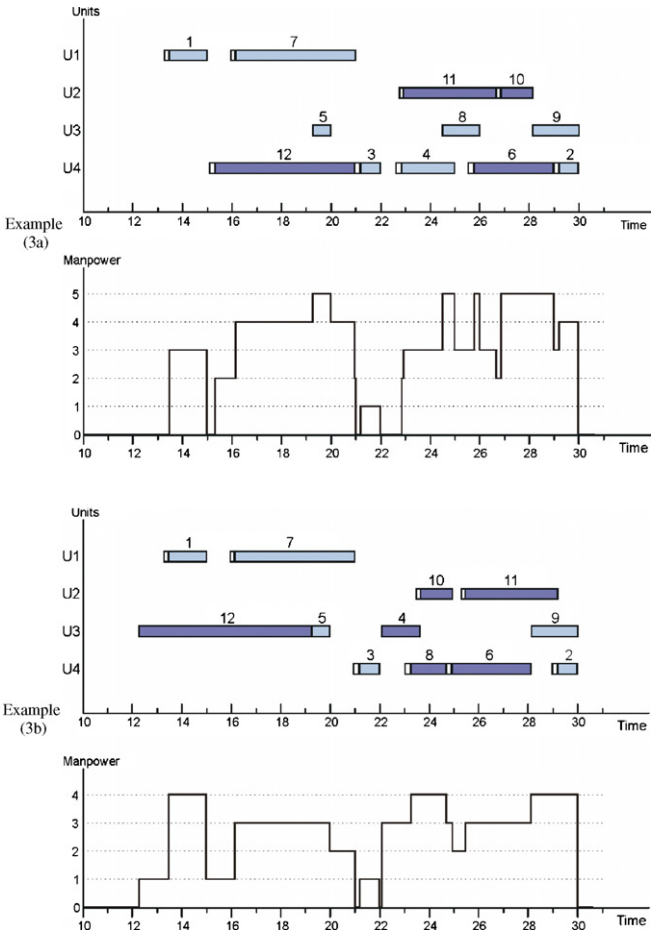



**Fig. 5.** Optimal schedules and manpower consumption profiles for Examples 3a–3b.

### 7.4. Example 4

Example 4 introduces a new case study involving a multistage batch facility with five processing stages and 12 processing units. Twelve single-batch orders featuring different due dates are to be scheduled. Multiple discrete/continuous bottleneck resources are simultaneously considered. The plant layout is shown in Fig. 6 and unit-dependent processing and setup times are given in Table 8. Additionally, sequence-dependent changeover times for every pair of batches are included in Table 9. The problem goal is to find the production schedule that minimizes the total tardiness.

Limited availabilities for production resources different from equipment like manpower, electricity and steam flow are considered. Resource capacities and order-dependent resource requirements at some stages are given in Table 10. Manpower must be allocated to six orders at stage V, while electricity is required by four orders at stage III. In turn, steam flow is needed to heat up 4 orders at stages III and V.

Several instances of Example 4 have been studied in order to evaluate the computational performance of the proposed approach as the number of bottleneck resources increases. Example 4a stands for the non-constrained case (limitations on resource capacities are ignored), Example 4b just considers the limitation on manpower capacity, Example 4c accounts for manpower and electricity as additional bottleneck resources, and finally Example 4d handles all resource requirements shown in Table 10. In order to reduce the number of binary variables, tasks potentially performed at the same equipment unit are preordered by increasing due dates before solving the problem. Otherwise, the problem cannot be solved on a CPU time below the time limit of 1 h. Preordering conditions will only apply to tasks featuring different due-dates and allocated to the same equipment item. Such conditions are handled by replacing Constraints (3)–(4) by Restrictions (26)–(28).

**Table 6**
Unit-dependent manpower requirements for Example 3.

| Order | Unit-dependent manpower requirement | | | |
|---|---|---|---|---|
| | $U_1$ | $U_2$ | $U_3$ | $U_4$ |
| 1 | 3 | – | – | 2 |
| 2 | 2 | – | – | 1 |
| 3 | 2 | – | – | 1 |
| 4 | – | – | 3 | 2 |
| 5 | – | – | 1 | 3 |
| 6 | 3 | – | – | 2 |
| 7 | 2 | – | 3 | 3 |
| 8 | – | – | 2 | 1 |
| 9 | – | – | 3 | 2 |
| 10 | – | 3 | – | – |
| 11 | – | 1 | – | 3 |
| 12 | – | 3 | 1 | 2 |

$$C_{i's} + \tau_{i'ij} + su_{ij} \le S_{is} + H(1 - X_{i's,is}) + H(2 - Y_{ij} - Y_{i'j}) \qquad \forall\, i,\ i' \in I,\ s \in S_{ii'},\quad j \in J_{ii's} : (i' < i) \wedge (dd_i = dd_{i'}) \tag{26}$$

$$C_{is} + \tau_{ii'j} + su_{i'j} \le S_{i's} + H X_{i's,is} + H(2 - Y_{ij} - Y_{i'j}) \qquad \forall\, i,\ i' \in I,\ s \in S_{ii'},\quad j \in J_{ii's} : (i' < i) \wedge (dd_i = dd_{i'}) \tag{27}$$

$$C_{i's} + \tau_{i'ij} + su_{ij} \le S_{is} + H(2 - Y_{ij} - Y_{i'j}) \qquad \forall\, i,\ i' \in I,\ s \in S_{ii'},\quad j \in J_{ii's} : (dd_{i'} < dd_i) \tag{28}$$

**Table 7**
Model size and computational results for Example 3.

| Model | Case | Binary variables, cont. variables, constraints | Objective function | CPU time (s) | Nodes | Iterations |
|---|---|---|---|---|---|---|
| Méndez and Cerdá (2002) | 3a | 151, 24, 886 | 6.234 [a] | 3600 † | 2,125,831 | 6,543,180 |
| | 3b | 139, 24, 754 | 11.12 [b] | 3600 † | 2,490,249 | 5,410,129 |
| Proposed formulation | 3a | 223, 72, 897 | 5.276 | 3.63 | 1,416 | 9,956 |
| | 3b | 223, 72, 897 | 11.12 | 8.59 | 4,216 | 27,090 |

† Time limit exceeded.
[a] Lower bound = 2.038, relative gap tolerance = 67.3%.
[b] Lower bound = 8.151, relative gap tolerance = 26.6%.

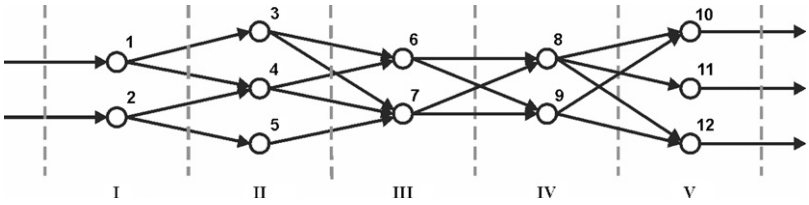



**Fig. 6.** Multistage batch facility at Example 4.

**Table 8**
Data for Example 4.

| Order | 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 | 13.2 | 11.0 | 9.2 | 8.5 | | 10.6 | 13.9 | | 10.3 | 8.4 | 10.4 | | 70 |
| 2 | 10.3 | 12.5 | 5.5 | | 11.1 | 11.8 | 14.1 | 8.0 | 9.9 | 11.5 | 9.9 | 8.2 | 85 |
| 3 | 8.1 | | 8.5 | 9.7 | 4.9 | 14.2 | 13.5 | 12.7 | | 12.8 | 7.2 | 8.3 | 100 |
| 4 | 12.7 | 13.7 | 9.9 | 6.1 | | 15.1 | 11.1 | 8.7 | 8.4 | 9.6 | | 9.4 | 70 |
| 5 | 8.7 | | 7.3 | 10.4 | | 11.3 | | 12.7 | 12.5 | 8.5 | 12.5 | 10.3 | 56 |
| 6 | 12.8 | 13.4 | 6.5 | 7.9 | 7.0 | 11.1 | 10.3 | | 9.6 | 12.5 | 9.4 | 10.9 | 110 |
| 7 | 10.3 | 13.0 | | 9.9 | 10.4 | 11.6 | 14.4 | 12.0 | 9.7 | | | 9.5 | 56 |
| 8 | | 12.1 | 7.9 | | 7.3 | | 10.8 | 9.7 | | 11.8 | 10.6 | 11.0 | 85 |
| 9 | 8.9 | | | 10.1 | 9.9 | 13.3 | 14.9 | 9.2 | 13.9 | | 9.8 | 9.3 | 70 |
| 10 | 11.0 | 8.9 | | 7.2 | 6.5 | | 13.7 | 12.7 | 13.0 | 11.5 | 10.8 | 8.6 | 100 |
| 11 | 10.6 | 8.5 | 5.3 | 7.9 | | 10.9 | 9.9 | 13.2 | | 8.5 | 10.4 | | 120 |
| 12 | 9.4 | 8.0 | 6.7 | 4.8 | 9.6 | 11.6 | 13.2 | 10.3 | 9.9 | 10.6 | | 7.7 | 120 |
| $su_{ij}$ | 0.4 | 0.2 | 0.5 | 0.4 | 0.4 | 0.2 | 0.3 | 0.6 | 0.5 | 0.3 | 0.2 | 0.6 | |

**Table 9**
Sequence-dependent setup times for Example 4.

| $\tau_{ii'}$ | $O_1$ | $O_2$ | $O_3$ | $O_4$ | $O_5$ | $O_6$ | $O_7$ | $O_8$ | $O_9$ | $O_{10}$ | $O_{11}$ | $O_{12}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $O_1$ | – | 0.6 | 0.6 | 1.0 | 0.3 | 0.9 | 0.9 | 0.4 | 0.4 | 0.6 | 0.3 | 0.5 |
| $O_2$ | 0.7 | – | 0.6 | 0.6 | 0.4 | 0.6 | 0.8 | 0.5 | 0.6 | 0.5 | 0.8 | 0.9 |
| $O_3$ | 0.7 | 0.8 | – | 0.5 | 0.8 | 0.6 | 0.4 | 0.3 | 0.9 | 0.6 | 0.9 | 0.3 |
| $O_4$ | 0.6 | 0.8 | 0.6 | – | 0.6 | 0.7 | 0.7 | 0.6 | 0.7 | 0.7 | 0.4 | 0.8 |
| $O_5$ | 0.8 | 0.8 | 0.4 | 0.8 | – | 0.5 | 0.4 | 0.6 | 0.5 | 0.6 | 0.4 | 0.8 |
| $O_6$ | 0.9 | 0.4 | 0.5 | 0.5 | 0.4 | – | 0.5 | 0.8 | 0.9 | 0.8 | 1.0 | 0.6 |
| $O_7$ | 0.7 | 0.7 | 0.6 | 0.4 | 0.7 | 0.3 | – | 0.8 | 0.3 | 0.7 | 0.6 | 0.4 |
| $O_8$ | 0.5 | 0.5 | 0.7 | 0.5 | 1.0 | 1.0 | 0.9 | – | 0.6 | 0.4 | 0.4 | 1.0 |
| $O_9$ | 1.0 | 0.6 | 0.8 | 0.8 | 0.3 | 0.5 | 0.5 | 0.7 | – | 0.6 | 0.3 | 0.8 |
| $O_{10}$ | 0.3 | 0.7 | 0.5 | 0.9 | 0.9 | 0.3 | 0.7 | 0.4 | 0.4 | – | 0.5 | 0.8 |
| $O_{11}$ | 0.3 | 0.7 | 0.7 | 0.4 | 0.6 | 0.4 | 0.7 | 1.0 | 1.0 | 0.5 | – | 0.9 |
| $O_{12}$ | 0.5 | 0.6 | 0.5 | 0.8 | 0.8 | 0.4 | 0.8 | 0.9 | 0.5 | 0.6 | 0.7 | – |

**Table 10**
Resource requirement data for Example 4.

| Resource | Order | | | | | | | | | | | | Capacity |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | |
| Manpower (Stage V) | 1 | | 2 | 1 | | | 2 | 1 | 2 | | | | 3 |
| Electricity (Stage III) | | | 9 | 10 | | 7 | | | 8 | | | | 15 |
| Steam (Stage III) | | | | | 8 | | 7 | | | | 7 | 5 | 12 |
| Steam (Stage V) | | | | | 6 | | 6 | | | | 5 | 8 | |

**Table 11**
Computational results for Examples 4a–d.

| Problem | Binary variables, cont. variables, constraints | MILP objective value | CPU time (s) | Nodes | Iterations |
|---|---|---|---|---|---|
| 4a. Without resource constraints | 151, 132, 777 | 9.3 | 69.73 | 52,667 | 258,814 |
| 4b. With manpower | 193, 132, 873 | 11.9 | 70.41 | 44,666 | 221,883 |
| 4c. With manpower + electricity | 210, 132, 913 | 18.3 | 119.36 | 62,510 | 355,351 |
| 4d. With all resources | 278, 132, 1065 | 31.6 | 114.05 | 43,945 | 303,092 |

**Table 12**
Number of binary variables for Examples 4a–d.

| Binary variables | Problem | | | |
|---|---|---|---|---|
| | 4a | 4b | 4c | 4d |
| Allocation $Y_{ij}$ | 116 | 116 | 116 | 116 |
| Sequencing $X_{i's',is}$ | 35 | 47 | 52 | 72 |
| Overlapping $W_{i's',is}$ | 0 | 30 | 42 | 90 |
| Total | 151 | 193 | 210 | 278 |

**Table 13**
Resource overloads at Example 4a.

| Resource | Capacity | Interval | Overlapping tasks | Overall Req. | Overload |
|---|---|---|---|---|---|
| Manpower | 3 | (A) | $(O_7, S_5), (O_9, S_5)$ | 4 | +1 |
| Electricity | 15 | (B) | $(O_9, S_3), (O_4, S_3)$ | 18 | +3 |
| | | (C) | $(O_3, S_3), (O_6, S_3)$ | 16 | +1 |
| Steam | 12 | (D) | $(O_5, S_3), (O_7, S_3)$ | 15 | +3 |
| | | (E) | $(O_{11}, S_5), (O_{12}, S_5)$ | 13 | +1 |

Values adopted for the model parameters are $H = 240$ and $\delta = 0.1$. Model sizes, optimal objective values, CPU times, and number of explored nodes and iterations for each instance of Example 4 are all given in Table 11. The rising number of resource constraints by moving from Example 4a to Example 4d almost doubles the number of binary variables and produces a significant increase in the overall tardiness at the optimum. Nonetheless, the CPU time just grows by a factor less than 2 and the number of explored nodes and iterations remains almost the same. Table 12 shows the required number of binary variables of each type (assignment, sequencing, overlapping) for every instance of Example 4. By including preordering conditions, the proposed MILP model requires a total of 278 binary variables for the most constrained case (Example 4d), from which 116 are assignment variables, 72 are sequencing variables, and 90 stand for overlapping decisions. Therefore, 127 additional variables are to be incorporated in the non-resource constrained problem formulation to deal with the most constrained instance of Example 4.

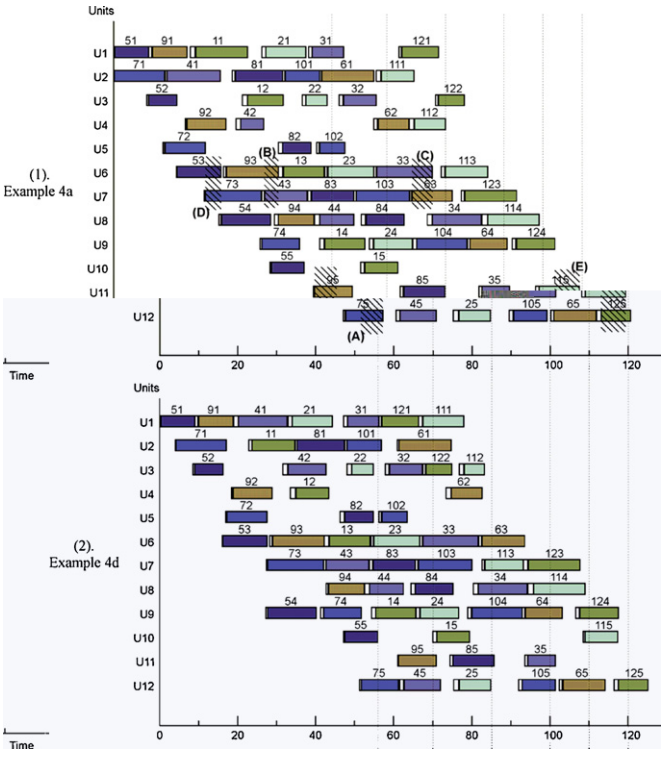



**Fig. 7.** (1) Optimal schedule for Example 4a with resource capacity overloads (A)–(E); (2) optimal schedule for Example 4d.

Although it represents a significant growth in the number of binary variables, the increase in CPU time is relatively low.

Fig. 7 presents the optimal schedules found for Examples 4a and 4d where processing tasks are identified by the concatenation of the batch number and the batch stage, i.e. 121 refers to the stage 1 of batch 12. Since limitations on resource availabilities are ignored, the optimal solution for Example 4a depicted in Fig. 7.1 includes five

**Table 14**
Starting and completion times of the orders at Stages $S_3$ and $S_5$ for Examples 4a and 4d.

| Order | $dd_i$ | Example 4a - Without Resources | | | | | Example 4d - With resource constraints | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Stage III | | Stage V | | | Stage III | | Stage V | | |
| | | $S_{i,S3}$ | $C_{i,S3}$ | $S_{i,S5}$ | $C_{i,S5}$ | $T_i$ | $S_{i,S3}$ | $C_{i,S3}$ | $S_{i,S5}$ | $C_{i,S5}$ | $T_i$ |
| $O_5$ | 56 | (D) 16.4 | 27.7 | 40.4 | 48.9 | | 16.4 | 27.7 | 47.5 | 56.0 | |
| $O_7$ | 56 | **23.6** | 38.0 | (A) 47.7 | 57.2 | 1.2 | 27.7 | 42.1 | 51.8 | 61.3 | 5.3 |
| $O_9$ | 70 | (B) 29.0 | 42.3 | **51.5** | 61.3 | | 29.0 | 42.3 | 61.3 | 71.1 | 1.1 |
| $O_4$ | 70 | **38.7** | 49.8 | 61.6 | 71.0 | 1.0 | 42.8 | 53.9 | 62.6 | 72.0 | 2.0 |
| $O_1$ | 70 | 43.5 | 54.1 | 64.4 | 72.8 | 2.8 | 43.5 | 54.1 | 71.1 | 79.5 | 9.5 |
| $O_8$ | 85 | 50.7 | 61.5 | 74.4 | 85.0 | | 54.8 | 65.6 | 75.3 | 85.9 | 0.9 |
| $O_2$ | 85 | 54.9 | 66.7 | 76.6 | 84.8 | | 54.9 | 66.7 | 76.8 | 85.0 | |
| $O_{10}$ | 100 | 62.2 | 75.9 | 90.6 | 99.2 | | 66.3 | 80.0 | 93.0 | 101.6 | 1.6 |
| $O_3$ | 100 | (C) 67.5 | 81.7 | 94.4 | 101.6 | 1.6 | 67.5 | 81.7 | 94.4 | 101.6 | 1.6 |
| $O_6$ | 110 | **76.5** | 86.8 | 101.0 | 111.9 | 1.9 | 82.7 | 93.8 | 103.4 | 114.3 | 4.3 |
| $O_{11}$ | 120 | 85.0 | 95.9 | (E) 109.1 | 119.5 | | 83.4 | 93.3 | 109.1 | 117.6 | |
| $O_{12}$ | 120 | 90.0 | 103.2 | **113.1** | 120.8 | 0.8 | 94.5 | 107.7 | 117.6 | 125.3 | 5.3 |