Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computers and Chemical Engineering 30 (2006) 913–946

Computers & Chemical Engineering

www.elsevier.com/locate/compchemeng

Review

State-of-the-art review of optimization methods for short-term scheduling of batch processes

Carlos A. Méndez[a], Jaime Cerdá[b], Ignacio E. Grossmann[a,*], Iiro Harjunkoski[c], Marco Fahl[c]

[a] *Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, USA*
[b] *INTEC (UNL-CONICET), Güemes 3450, 3000 Santa Fe, Argentina*
[c] *ABB Corporate Research Center, Ladenburg, Germany*

Received 30 June 2005; received in revised form 24 January 2006; accepted 9 February 2006
Available online 17 April 2006

Abstract

There has been significant progress in the area of short-term scheduling of batch processes, including the solution of industrial-sized problems, in the last 20 years. The main goal of this paper is to provide an up-to-date review of the state-of-the-art in this challenging area. Main features, strengths and limitations of existing modeling and optimization techniques as well as other available major solution methods are examined through this paper. We first present a general classification for scheduling problems of batch processes as well as for the corresponding optimization models. Subsequently, the modeling of representative optimization approaches for the different problem types are introduced in detail, focusing on both discrete and continuous time models. A comparison of effectiveness and efficiency of these models is given for two benchmarking examples from the literature. We also discuss two real-world applications of scheduling problems that cannot be readily accommodated using existing methods. For the sake of completeness, other alternative solution methods applied in the field of scheduling are also reviewed, followed by a discussion related to solving large-scale problems through rigorous optimization approaches. Finally, we list available academic and commercial software, and briefly address the issue of rescheduling capabilities of the various optimization approaches as well as important extensions that go beyond short-term batch scheduling.

Keywords: Short-term scheduling; Optimization models; Batch processes; MILP

Contents

* Corresponding author. Tel.: +1 412 268 3642; fax: +1 412 268 7139.
E-mail address: grossmann@cmu.edu (I.E. Grossmann).

0098-1354/$ – see front matter
doi:10.1016/j.compchemeng.2006.02.008

Introduction

Scheduling is a critical issue in process operations and is crucial for improving production performance. For batch processes, short-term scheduling deals with the allocation of a set of limited resources over time to manufacture one or more products following a batch recipe. There have been significant research efforts over the last decade in this area in the development of optimization approaches, and several excellent reviews can be found in Floudas and Lin (2004), Kallrath (2002), Pekny and Reklaitis (1998), Pinto and Grossmann (1998), Reklaitis (1992), Shah (1998). Despite significant advances there are still a number of major challenges and questions that remain unresolved. For instance, it is not clear the extent to which general methods aimed at complex network structures (see Fig. 1), can also be effectively applied to commonly encountered structures such as the multistage structure shown in Fig. 2. There are also many detailed questions related on the specific capabilities of the methods for handling a large number of operational issues (e.g. variable or fixed batch size, storage and transfer policies, changeovers), as well as different objectives (e.g. makespan, earliness, or cost minimization). Finally, there are also questions on the limitations

Nomenclature

Indices

f, f'	product family
i, i'	batch task
i^{ST}	storage task
j, j'	batch processing unit
k	time slot (continuous time)
l	stage
n, n'	time or event point (continuous time)
r, r'	resource type
s	stage
t, t'	time intervals (discrete time)
z, z'	resource item

Sets

I	batch tasks
I_j	tasks that can be processed in unit j
I_r	tasks that require resource r
I_j^f	tasks belonging to family f that can be processed in unit j

I^{ST} storage tasks
I^{ZW} tasks that produce at least one zero wait state
I_s^{ST} storage tasks for state s
I_r^{zw} tasks that produce the resource r which requires zero wait policy
I_s^{c} tasks that consume state s
I_s^{p} tasks that produce state s
J processing units
J_i processing units that can perform task i
$J_{ii'}$ processing units that can perform both task i and task i'
J^T storage units
J_s^T storage units that can store state s
K_j time slots for processing unit j
L_i stages of batch i
N time or event points (continuous time)
R resources
R_{il} resources required in stage l of task i
R^J resources corresponding to processing equipment
R_i^J resources corresponding to processing equipment that can be allocated to task i
R_i^S resources corresponding to storage equipment that can be allocated to task i
S states
S_j states that can be stored in a shared storage tank j
S^T states that can be stored in tanks
S^{ZW} states that require a zero wait policy
T time intervals (discrete time)
Z resource items
Z_r resource items of type r

Parameters
α_i fixed processing time of task i
β_i variable processing time of task i
μ_{irt} coefficient for the fixed production/consumption of resource r at time t relative to the start of the task i
μ_{ir}^{c} coefficient for the fixed consumption of resource r at the beginning of task i
μ_{ir}^{p} coefficient for the fixed production of resource r at the end of task i
ν_{irt} coefficient for the variable production/ consumption of resource r at time t relative to the start of the task i
ν_{ir}^{c} coefficient for the variable consumption of resource r at the beginning of task i
ν_{ir}^{p} coefficient for the variable production of resource r at the end of task i
ν_{ilrj} amount of resource r required when task (i, l) is allocated to unit j
ρ_{is}^{c} proportion of state s consumed by task i
ρ_{is}^{p} proportion of state s produced by task i
$\prod_{st}$ amount of state s received at time t
$\mathrm{cl}_{ff'}$ changeover time required between a task belonging family f and a task belonging to family f'
$\mathrm{cl}_{ii'}$ changeover time required between task i and a task i'
$\mathrm{cl}_{il,i'l'}$ changeover time required between stage l of task i and stage l' of task i'
$cl_{ii'j}$ changeover time required between task i and a task i' in unit j
C_j maximum capacity of storage tank j
C_s^{min} minimum storage capacity for state s
C_s^{max} maximum storage capacity for state s
D_{st} amount of state s delivered at time t
H time horizon of interest
pt_{ij} processing time of task i in unit j
q_{rz} amount of resource r available at the resource item z of type r
R_r^{min} minimum availability of resource r
R_r^{max} maximum availability of resource r
R_{rt}^{max} maximum availability of resource r at the beginning of time interval t
V_i^{min} minimum batch size of task i
V_i^{max} maximum batch size of task i
V_{ij}^{min} minimum capacity of unit j for task i
V_{ij}^{max} maximum capacity of unit j for task i
V_{ir}^{min} minimum capacity of resource r for task i
V_{ir}^{max} maximum capacity of resource r for task i
Su_{ij} setup time for processing task i in unit j
Tp_{ij} processing time of batch i in unit j
Tp_{ilj} processing time of batch task (i, l) in unit j

Binary variables
V_{jsn} define if state s is being stored in tank j at time point n
W_{it} define if task i starts at the beginning of time interval t
W_{in} define if task i is being performed at event point n
$W_{inn'}$ define if task i starts at time point n and ends at time point n'
W_{ijt} define if task i starts in unit j at the beginning of time interval t
W_{ijkl} define if the stage l of task i is allocated to the time slot k of unit j
W_{ij} define if task i is allocated to unit j
Ws_{in} define if task i starts at time or event point n
Wf_{in} define if task i finishes at time or event point n
WF_{ij} define if task i starts the processing sequence of unit j
$X_{ii'j}$ define if task i is processed right before task i' in unit j (immediate precedence)
$X_{ii'}$ define if task i is processed right before task i' in some unit (immediate precedence)
$X'_{il,i'l'}$ define if stage l of task i is processed before/ after stage l' of task i' in some unit (general precedence)
Y_{ilz} define if resource item z is allocated to stage l of task i

Continuous variables

B_{ijt} batch size of the task i started at the beginning of time interval t in unit j
B_{it} batch size of the task i started at the beginning of time interval t
B_{in} batch size of the task i activated at time or event point n
$B_{inn'}$ batch size of the task i started at time n and ended at time n'
Bs_{in} batch size of the task i started at time or event point n
Bf_{in} batch size of the task i finished at or before time or event point n
Bp_{in} batch size of the task i that is being processed at time point n
PT_{in} processing time of task i that starts at time point n
R_{rn} amount of resource r that is being consumed at time point n
R_{irn} amount of resource r that is being consumed by task i at time point n
R_{rt} amount of state r that is being consumed at the beginning of time interval t
S_{sn} amount of state s at time point n
S_{sjn} amount of state s stored in shared tank j at time point n
S_{st} amount of state s at the beginning of time interval t
T_n time that corresponds to time point n
Ts_{in} start time of task i that starts at time point n
Ts_{rn} start time of usage of resource r at event point n
Ts_{jk} start time of slot k in unit j
Ts_{i} start time of task i
Ts_{il} start time of stage l of task i
Tf_{i} finish time of task i
Tf_{in} finish time of task i that starts at time point n
Tf_{jk} finish time of slot k in unit j
Tf_{il} finish time of stage l of task i

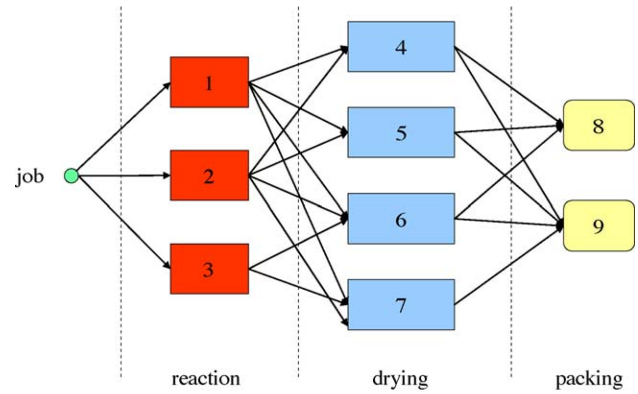
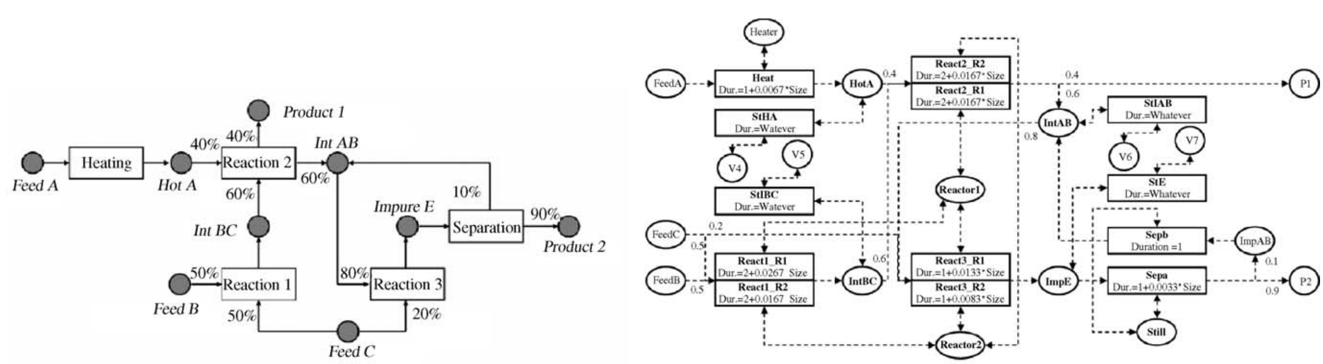


Fig. 2. Batch process with sequential structure.

and strengths of the various optimization models that have been reported in the literature and the size of problems that one can realistically solve with these models.

It is the objective of this paper to provide a comprehensive review of the state-of-the art of short-term batch scheduling. Our aim is to try to provide answers to the questions posed in the above paragraph. The paper is organized as follows. We first present a classification for scheduling problems of batch processes, as well as of the features that characterize the optimization models for scheduling. We then present the major equations for representative optimization approaches for general network and sequential batch plants, focusing on the discrete and continuous time models. Computational results on two specific case studies (general network and sequential plants) are presented in order to compare the performance of several of the methods, particularly discrete and continuous models. We also discuss two examples of real world industrial scheduling problems to demonstrate difficulties that are faced by existing methods in accommodating complex process requirements. Other alternative solution approaches are briefly discussed, followed by a discussion on the solution of large-scale problems with exact methods. We briefly describe academic and commercial software available in the batch scheduling area, and address the issue of rescheduling capabilities of the various optimization approaches as well as various important extensions that go beyond short-term batch scheduling.



(a) *STN representation* (b) *RTN representation*

Fig. 1. Batch process with complex network structure.

1. Classification of batch scheduling problems

There are a great variety of aspects that need to be considered when developing scheduling models for batch processes. In order to provide a systematic characterization we present first a general roadmap for classifying most relevant problem features. This roadmap is summarized in Fig. 3 and considers not only equipment and material issues, but also time and demand-related constraints. As can be seen, the main features involve 13 major categories, each of which are linked to central problem characteristics. These significantly complicate the task of providing a unified treatment that can address all the cases covered in Fig. 3.

First, the process layout and its topological implications have a significant influence on problem complexity. In practice many batch processes are sequential, single or multiple stage processes, where one or several units may be working in parallel in each stage. Each batch needs to be processed following a sequence of stages defined through the product/batch recipe.

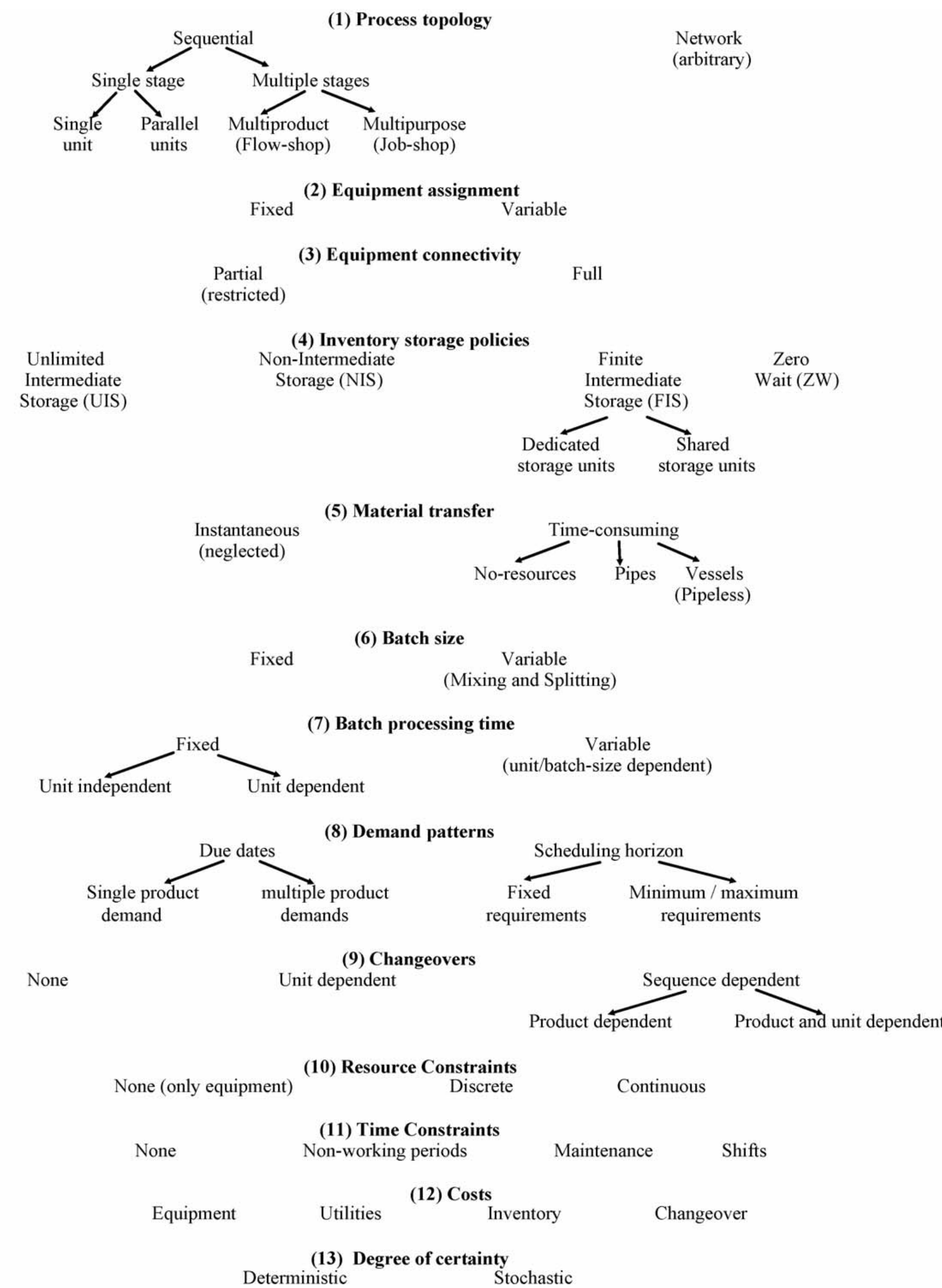


Fig. 3. Roadmap for scheduling problems of batch plants.

However, increasingly as applications become more complex, networks with arbitrary topology must be handled. Complex product recipes involving mixing and splitting operations and material recycles need to be considered in these cases. Closely related to topology considerations are requirements/constraints on equipment in terms of its assignment and connectivity, ranging from fixed to flexible arrangements. Limited interconnections between equipment impose hard constraints on unit allocation decisions.

Another important aspect of process flow requirements is reflected in inventory policies. These often involve finite and dedicated storage, although frequent cases include shared tanks as well as zero-wait, non-intermediate and unlimited storage policies. Material transfer is often assumed instantaneous, but in some cases like in pipe-less plants it is significant and must be accounted for in corresponding modeling approaches. Another major factor is the handling of batch size requirements. For instance, pharmaceutical plants usually handle fixed sizes for which integrity must be maintained (no mixing/splitting of batches), while solvent or polymer plants handle variable sizes that can be split and mixed. Similarly, different requirements on processing times can be found in different industries depending on process characteristics. For example pharmaceutical applications might involve fixed times due to FDA regulations, while solvents or polymers have times that can be adjusted and optimized with process models.

Demand patterns also can vary significantly ranging from cases where due dates must be obeyed to cases where production targets must be met over a time horizon (fixed or minimum amounts). Changeovers are also a very important factor that is particularly critical in cases of transitions that are sequence dependent on the products, as opposed to simple set-ups that are only unit dependent.

Resource constraints, aside from equipment, e.g. labor, utilities, are also often of great importance and can range from pure discrete to continuous. Practical operating considerations often give rise to time constraints such as non-working periods on the weekend or maintenance periods. Also, while scheduling is often regarded as a feasibility problem, costs associated with the use of equipment, inventories, changeovers and utilities can have a significant impact in defining an optimal schedule. Finally, there is the issue of the degree to which uncertainty in the data must be accounted for, which is particularly critical for demands as longer time horizons are used.

The classification in Fig. 3 shows that there is a tremendous diversity of factors that must be accounted for in short-term batch scheduling, which makes the task of developing unified general methods quite difficult. At the same time, there might be the trade-off of having a number of specialized methods that can address specific cases of this classification in a more efficient way.

2. Classification of optimization models for batch scheduling

Having presented the general features of typical batch scheduling problems we introduce a roadmap that describes the main features of current optimization approaches. This section is of particular importance because alternative ways of addressing/formulating the same problem are described. These usually have a direct impact on the computational performance, capabilities and limitations of the resulting optimization model. Each modeling option that is presented is able to cope with a subset of the features described in Fig. 3.

The roadmap for optimization model classification (Fig. 4) focuses on four main aspects that are described in more detail in the remainder of this section.

- *Time representation:* the first and most important issue is the time representation. Depending on whether the events of the schedule can only take place at some predefined time points, or can occur at any moment during the time horizon of interest, optimization approaches can be classified into discrete and continuous time formulations. Discrete time models are based on: (i) dividing the scheduling horizon into a finite number of time intervals with predefined duration and, (ii) allowing the events such as the beginning or ending of tasks to happen only at the boundaries of these time periods. Therefore, scheduling constraints have only to be monitored at specific and known time points, which reduces the problem complexity and makes the model structure simpler and easier to solve, particularly when resource and inventory limitations are taken into account. On the other hand, this type of problem simplification has two major disadvantages. First, the

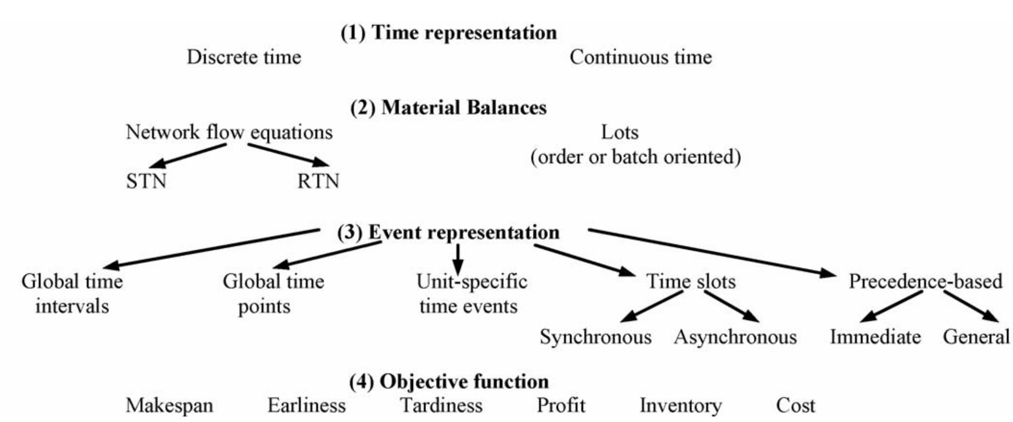


Fig. 4. Roadmap for optimization models for short-term scheduling of batch plants.

size of the mathematical model as well as its computational efficiency strongly depend on the number of time intervals postulated, which is defined as a function of the problem data and the desired accuracy of the solution. Second, sub-optimal or even infeasible schedules may be generated because of the reduction of the domain of timing decisions. Despite being a simplified version of the original scheduling problem, discrete formulations have proven to be very efficient, adaptable and convenient for a wide variety of industrial applications, especially in those cases where a reasonable number of intervals is sufficient to obtain the desired problem representation.

In order to overcome the previous limitations and generate data-independent models, a wide variety of optimization approaches employ a continuous time representation. In these formulations, timing decisions are explicitly represented as a set of continuous variables defining the exact times at which the events take place. In the general case, a variable time handling allows obtaining a significant reduction of the number of variables of the model and at the same time, more flexible solutions in terms of time can be generated. However, because of the modeling of variable processing times, resource and inventory limitations usually needs the definition of more complicated constraints involving many big-M terms, which tends to increase the model complexity and the integrality gap and may negatively impact on the capabilities of the method.

- *Material balances:* the handling of batches and batch sizes gives rise to two types of optimization model categories. The first category refers to monolithic approaches, which simultaneously deal with the optimal set of batches (number and size), the allocation and sequencing of manufacturing resources and the timing of processing tasks. These methods are able to deal with arbitrary network processes involving complex product recipes. Their generality usually implies large model sizes and consequently their application is currently restricted to processes involving a small number of processing tasks and rather short scheduling horizons. These models employ the state-task network (STN), or the resource-task network (RTN) concept to represent the problem. As shown in Fig. 1a, the STN-based models represent the problem assuming that processing tasks produce and consume states (materials). A special treatment is given to manufacturing resources aside from equipment. The STN is a directed graph that consists of three key elements: (i) *state nodes* representing feeds, intermediates and final products; (ii) *task nodes* representing the process operations which transform material from one or more input states into one or more output states and; (iii) *arcs* that link states and tasks indicating the flow of materials. State and task nodes are denoted by circles and rectangles, respectively. In contrast, the RTN-based formulations employ a uniform treatment and representation framework for all available resources through the idea that processing and storage tasks consume and release resources at their beginning and ending times, respectively (see Fig. 1b). In this particular case, circles represent not only states but also other resources required in the batch process such as processing units and vessels.

  The second category comprises models that assume that the number of batches of each size is known in advance. These solution algorithms can indeed be regarded as one of the modules of a solution approach for detailed production scheduling, widely used in industry, which decomposes the whole problem into two stages, batching and batch scheduling. The batching problem converts the primary requirements of products into individual batches aiming at optimizing some criterion like the plant workload. Afterwards, the available manufacturing resources are allocated to the batches over time. This approximate two stage approach can address much larger practical problems than monolithic methods, especially those involving a quite large number of batch tasks related to different intermediates or final products. However, this approach is still restricted to processes comprising sequential product recipes.
- *Event representation:* in addition to the time representation and material balances, scheduling models are based on different concepts or basic ideas that arrange the events of the schedule over time with the main purpose of guaranteeing that the maximum capacity of the shared resources is never exceeded. As can be seen in Figs. 4 and 5, we classified these concepts into five different types of event representations, which have been broadly utilized to develop a variety of mathematical formulations for batch scheduling problems. Particularly, Fig. 5 depicts a schematic representation of the same schedule obtained by using the alternative concepts. The small example given involves five batches (a, b, c, d, e) allocated to two units (J1, J2). To represent this solution, the different alternatives require: (a) 10 fixed time intervals, (b) five variable global time points, (c) three unit-specific time events, (d) three asynchronous time slots for each unit, (e) three immediate precedence relationships or four general precedence relationships. Although some event representations are more general than others, they are usually oriented towards the solution of either arbitrary network processes requiring network flow equations or sequential batch processes assuming a batch-oriented approach. Table 1 summarizes the most relevant modeling characteristics and problem features related to the alternative event representations. Critical modeling issues refer to those aspects that may seriously impact the model size and hence the computational effort. In turn, critical problem features indicate certain problem aspects that may be awkward to consider through specific basic concepts.

  For discrete time formulations, the definition of global time intervals is the only option for general network and sequential processes. In this case, a common and fixed time grid valid for all shared resources is predefined and batch tasks are enforced to begin and finish exactly at a point of the grid. Consequently, the original scheduling problem is reduced to an allocation problem where the main model decisions define the assignment of the time interval at which every batch task begins, which is modeled through the discrete variable W_{ijt} as shown in Table 1. A significant advantage of using a fixed time grid is that time-dependent problem aspects can be modeled in a relatively simple way without compromising the linear-

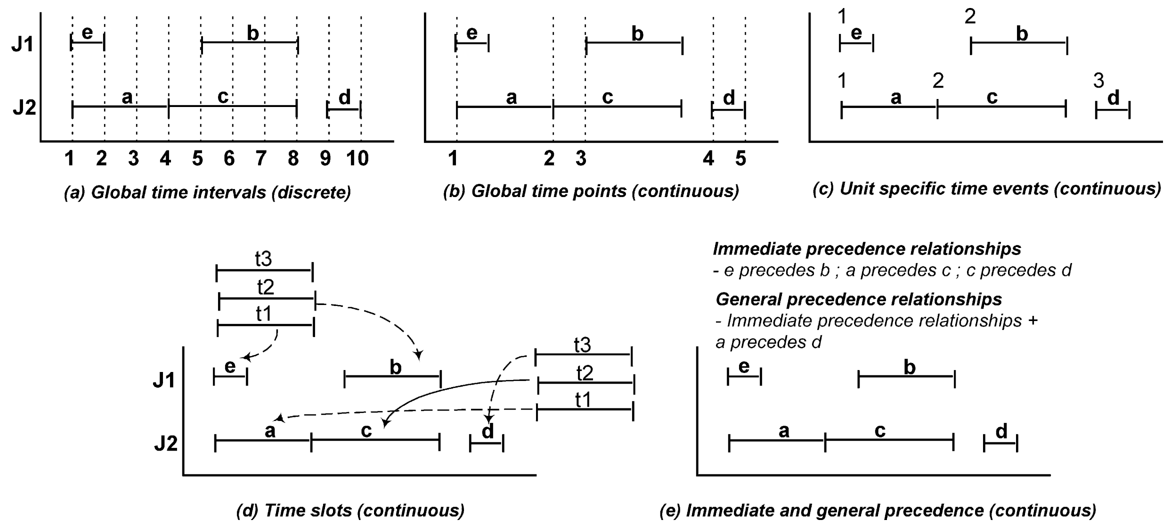


Fig. 5. Different concepts for representing scheduling problems.

ity of the model. Some of these aspects comprise hard time constraints, time-dependent utilities cost, inventory cost and multiple product demands and/or raw material supplies taking place during the scheduling horizon.

In contrast to the discrete time representation, continuous time formulations involve extensive alternative event representations, which are focused on different types of batch processes. For instance, for general network processes global time points and unit-specific time events can be used, whereas in the case of sequential processes the alternatives involve the use of time slots and different batch precedence-based approaches. The global time point representation corresponds to a generalization of global time intervals where the timing of time intervals is treated as new model variable. In this case, a common and variable time grid is defined for all shared resources. The beginning and the finishing times of the set

Table 1
General characteristics of current optimization models

Characteristic	Discrete time models	Continuous time Models					
Event representation	Global time intervals	Global time points	Unit-specific time events	Time slots*	Unit-specific immediate precedence*	Immediate precedence*	General precedence*
Main decisions	Lot-sizing, allocation, sequencing, timing				Allocation, sequencing, timing		
Key discrete variables	W_{ijt} defines if task *I* starts in unit *j* at the beginning of time interval *t*.	Ws_{in} / Wf_{in} define if task *i* starts/ends at time point *n*. $W_{inn'}$ defines if task *i* starts at time point *n* and ends at time point *n'*.	Ws_{in} / W_{in} / Wf_{in} define if task *i* starts/is active/ends at event point *n*.	W_{ijk} define if unit *j* starts task *i* at the beginning of time slot *k*.	$X_{ii'j}$ defines if batch *i* is processed right before of batch *i'* in unit *j*. XF_{ij} defines if batch *i* starts the processing sequence of unit *j*.	$X_{ii'}$ defines if batch *i* is processed right before of batch *i'*. XF_{ij} / W_{ij} defines if batch *i* starts/is assigned to unit *j*.	$X'_{ii'}$ define if batch *i* is processed before or after of batch *i'*. W_{ij} defines if batch *i* is assigned to unit *j*
Type of process	General network				Sequential		
Material balances	Network flow equations (STN or RTN)	Network flow equations (STN or RTN)	Network flow equations (STN)		Batch-oriented		
Critical modeling issues	Time interval duration, scheduling period (data dependent)	Number of time points (iteratively estimated)	Number of time events (iteratively estimated)	Number of time slots (estimated)	Number of batch tasks sharing units (lot-sizing) and units	Number of batch tasks sharing units (lot-sizing)	Number of batch tasks sharing resources (lot-sizing)
Critical problem features	Variable processing times, sequence-dependent changeovers	Intermediate due dates and raw-material supplies	Intermediate due dates and raw-material supplies	Resource limitations	Inventory, resource limitations	Inventory, resource limitations	Inventory

* Batch-oriented formulations assume that the overall problem is decomposed into the lot-sizing and the short-term scheduling issues. The lot-sizing or "batching" problem is solved first in order to determine the number and size of "batches" to be scheduled.

of batch tasks are linked to specific time points through the key discrete variables reported in Table 1. Both for continuous STN as RTN based models, limited capacities of resources need to be monitored at a small number of variable time points in order to guarantee the feasibility of the solution. Models following this direction are relatively simple to implement even for general scheduling problems. In contrast to global time points, the idea of unit-specific time events defines a different variable time grid for each shared resource, allowing different tasks to start at different moments for the same event point. These models make use of the STN representation. Because of the heterogeneous locations of the event points, the number of events required is usually smaller than in the case of global time points. However, the lack of reference points for checking the limited availability of shared resources makes the formulation much more complicated. Special constraints and additional variables need to be defined for dealing with resource-constrained problems.

The usefulness and computational efficiency of the formulations based on global time points or unit-dependent time events strongly depend on the number of time or events points predefined. For instance, if the global optimal solution of the problem requires the definition of at least n time or event points, fewer points will lead to suboptimal or infeasible schedules whereas a larger number will result in significant and unnecessary computational effort. Since this number is unknown a priori, a practical criterion is to determine it through an iterative procedure where the number of variable points or events is increased by 1 until there is no improvement in the objective function. This means that a significant number of instances of the model need to be solved for each scheduling problem, which may lead to a high total CPU time. It is worth mentioning that this stopping criterion cannot guarantee the optimality of the solution and in some cases may also stop with a poor feasible schedule.

The previous continuous-time representations are mostly oriented towards arbitrary network processes. On the other hand, different continuous-time formulations were initially focused on a wide variety of sequential processes, although some of them have been recently extended to also consider general batch processes. One of the first developments following this direction was based on the concept of time slots, which stands for a set of predefined time intervals with unknown durations. The main idea is to postulate an appropriate number of time slots for each processing unit in order to allocate them to the batch tasks to be performed. The selection of the number of time slots required is not a trivial decision and represents an important trade-off between optimality and computational performance. Slot-based representations can be classified into two types: synchronous and asynchronous. The synchronous representation, which is similar to the idea of global time points, defines identical or common slots across all units in such a way that the shared resources involved in network batch processes are more natural and easier to handle. Alternatively, the asynchronous representation allows the postulated slots to differ from one unit to another, which for a given number of slots provides more flexibility in terms of timing decisions than its synchronous counterpart. This representation is similar to the idea of unit-specific time events and is more appropriate when for dealing with sequential batch processes.

Other alternative approaches for sequential processes are based on the concept of batch precedence. Model variables and constraints enforcing the sequential use of shared resources are explicitly employed in these formulations. As a result, sequence dependent changeover times can be treated in a straightforward manner. In order to determine the optimal processing sequence in each unit, the concept of batch precedence can be applied to either the immediate or any batch predecessor. The immediate predecessor of particular batch i is the batch i' that is processed right before in the same processing unit whereas the general precedence notion extends the immediate precedence concept to not only consider the immediate predecessor but also all batches processed before in the same processing sequence. Three different types of precedence-based mathematical formulations are reported in Table 1. When the immediate precedence concept is applied, sequencing decisions in each processing unit can be easily determined through a unique set of model variables $X_{ii'j}$. However, in order to reduce the model size and consequently, the computational effort, allocation and sequencing decisions are frequently decoupled in two different sets of model variables W_{ij} and $X_{ii'}$, as described in Table 1. In contrast to the immediate precedence-based models, the general precedence concept needs the definition of a single sequencing variable for each pair of batch tasks that can be allocated to the same shared resource. In this way, the formulation is simpler and smaller than those based on the immediate predecessor. In addition, this approach can handle the use of different types of renewable shared resources such as processing units, storage tanks, utilities and manpower through a single set of sequencing variables without compromising the optimality of the solution. A common weakness of precedence-based formulations is that the number of sequencing variables scales in the number batches to be scheduled, which may result in significant model sizes for real-world applications.

- *Objective function:* different measures of the quality of the solution can be used for scheduling problems (Fig. 4). However, the criteria selected for the optimization usually has a direct effect on the model computational performance. In addition, some objective functions can be very hard to implement for some event representations, requiring additional variables and complex constraints.

3. Modeling aspects of alternative approaches

Having introduced a general road map for classifying problems and models for batch scheduling, we present in this section the specific model equations and variables that are involved in the most relevant work developed for the different types of event representations shown in Table 1. Some formulations were slightly modified from their original version in order to use similar nomenclature and model structure.

3.1. Global time intervals (discrete time)

The event representation based on the definition of global time intervals employs a predefined time grid T that is valid for all shared resources involved in the scheduling problem, such as processing units J (see Fig. 5a). Relevant modeling features of discrete models based on the STN and RTN process representation are described below.

3.1.1. STN-based discrete formulation

The most relevant contribution for discrete time models is the state task network representation proposed by Kondili, Pantelides, and Sargent (1993) and Shah, Pantelides, and Sargent (1993) (see also Rodrigues, Latre, & Rodrigues, 2000). The STN model covers all the features that are included at the column on discrete time in Table 1. The general constraints and variables included in these models are introduced below.

3.1.1.1. Allocation constraints. Constraint (1), which is expressed in terms of the binary variables W_{ijt} to denote the start of task i in equipment j at time t, states that at most one task i can be processed in unit j during time interval t. To do that, this constraint makes use of a full backward aggregation that takes into account the implications for previous allocations. Fig. 6 illustrates the application of that constraints at $t=4$, for the case of two tasks of duration 2 and 3 time units. The dots represent that not more than one of them can be started at those fixed time points. In comparison with the original STN MILP formulation by Kondili et al. (1993), constraint (1) requires much fewer equations and reduces the integrality gap by eliminating any type of big-M constraint, which significantly enhances the computational performance of the solution procedure. Thus, a longer scheduling horizon can be addressed. It should be noted that this constraint requires that fixed and known processing times are predefined for all tasks to be scheduled. In addition, it is implicitly assumed that all tasks must release the allocated processing equipment when they finish, i.e. processing units are not allowed to be used as temporary storage devices

$$\sum_{i \in I_j} \sum_{t'=t-\mathrm{pt}_{ij}+1}^{t} W_{ijt'} \leq 1 \quad \forall j, t \tag{1}$$

3.1.1.2. Capacity limitations. Constraints (2) and (3) account for variable batch size B_{ijt} for each task i at unit j and limited storage capacities S_{st} for each state s. The amount of material that starts to be processed by task i in unit j at time t is bounded by the minimum and maximum capacities of that unit. In addition, constraint (2) forces the batch size variable B_{ijt} to be zero if $W_{ijt}=0$

$$V_{ij}^{\min} W_{ijt} \leq B_{ijt} \leq V_{ij}^{\max} W_{ijt} \quad \forall i, j \in J_i, t \tag{2}$$

Constraint (3) denotes that the amount of state s at time t must always satisfy minimum and maximum inventory requirements. It should be noted that dedicated storage units are assumed to be available for each state s

$$C_s^{\min} \leq S_{st} \leq C_s^{\max} \quad \forall s, t \tag{3}$$

3.1.1.3. Material balances. Constraint (4) computes the amount of state s stored at time t by considering the amount of state s (i) stored at time $t-1$, (ii) produced at time t, (iii) consumed at time t, (iv) received as raw material at time t and, (v) delivered at time t. Parameters ρ_{is}^{p} and ρ_{is}^{c} define the proportion of state s produced/consumed by task i

$$S_{st} = S_{s(t-1)} + \sum_{i \in I_s^{\mathrm{p}}} \rho_{is}^{\mathrm{p}} \sum_{j \in J_i} B_{ij(t-\mathrm{pt}_{ij})} - \sum_{i \in I_s^{\mathrm{c}}} \rho_{is}^{\mathrm{c}} \sum_{j \in J_i} B_{ijt} + \prod\nolimits_{st} - D_{st} \quad \forall s, t \tag{4}$$

3.1.1.4. Resource balances. Limited availability of resources R other than processing units can be explicitly modeled by constraint (5) and (6). Constraint (5) computes the total requirement of resource r at every time interval t. Taking advantage of the predefined time grid as well as the fixed processing times, this constraint is able to deal with variable resource requirements along the task execution. Whenever the binary variable $W_{ij(t-t')}$ takes the value one, it means that task i is being performed in unit j at time t and has been started t' time intervals earlier than t. Additionally, the value of continuous variable $B_{ij(t-t')}$ define the corresponding batch size of the task. Coefficients $\mu_{irt'}$ and $\nu_{irt'}$ are used to specify the fixed and variable requirement of resource r of task i

$$R_{rt} = \sum_{i \in I_r} \sum_{j \in J_i} \sum_{t'=0}^{\mathrm{pt}_{ij}-1} (\mu_{irt'} W_{ij(t-t')} + \nu_{irt'} B_{ij(t-t')}) \quad \forall r, t \tag{5}$$

Besides, the maximum availability of resource r cannot be exceeded at any time during the time horizon, as expressed by constraint (6)

$$0 \leq R_{rt} \leq R_{rt}^{\max} \quad \forall r, t \tag{6}$$

3.1.1.5. Sequence-dependent changeovers. Changeover requirements can be modeled by ensuring that adequate time is left for a unit to be cleaned between uses. In this way, constraint (7) guarantees that if the unit j starts processing any task of family f at time t, i.e. $W_{ijt}=1$, no task i' of family f' can start at least $\mathrm{cl}_{f'f} + \mathrm{pt}_{i'j}$ units of time before time t

$$\sum_{i \in I_j^f} W_{ijt} + \sum_{i' \in I_j^{f'}} \sum_{t'=t-\mathrm{cl}_{f'f}-\mathrm{pt}_{i'j}+1}^{t} W_{i'jt'} \leq 1 \quad \forall j, f, f', t \tag{7}$$

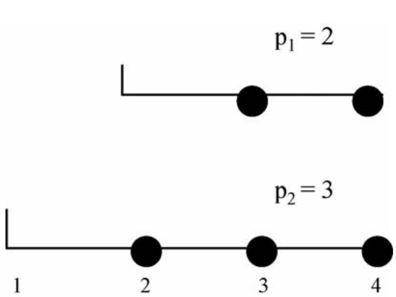


Fig. 6. Illustration of the inequality in (1) for two tasks at time $t=4$.

We should note however that implementing constraint (7) is rather awkward because it requires a finer discretization of time in order to accommodate smaller changeover times. Moreover, the number of constraints (7) quickly becomes extremely large when problems involving a significant number of changeovers are solved.

It is interesting to note how the capability for solving MILPs, such as the discrete time STN, has evolved over time. For instance, consider the classic problem shown in Fig. 1a (Kondili et al., 1993) with the STN MILP over a horizon of ten time units. In 1987, Kondili solved this problem using a weaker form of the constraints in (1) in 908 s and 1466 nodes on a VAX-8600 using her own LP-based branch and bound code with MINOS. In 1992 Shah solved this problem in 119 s and 149 nodes on a SUN-SPARC using the strong form of the inequality in (1) and also with his own branch and bound method. In 2003, one of the authors (Grossmann) solved the same model in his laptop IBM-T40 using CPLEX 7.5, which required only 0.45 s and 22 nodes! Thus it is clear that a combination of better models, faster computers and faster MILP solvers is greatly increasing the capability for solving optimization models for scheduling.

3.1.2. RTN-based discrete formulation

A simpler and general discrete time scheduling formulation can also be derived by means of the resource task network concept proposed by Pantelides (1994). The major advantage of the RTN formulation over the STN counterpart arises in problems involving identical equipment. Here, the RTN formulation introduces a single binary variable instead of the multiple variables used by the STN model. The RTN-based model also covers all the features at the column on discrete time in Table 1. In order to deal with different types of resources in a uniform way, this approach requires only three different classes of constraints in terms of three types of variables defining the task allocation W_{it}, the batch size B_{it}, and the resource availability R_{rt}. In few words, this model reduces the batch scheduling problem to a simple resource balance problem carried out in each predefined time period. It is worth mentioning that the elimination of the unit sub-index from the allocation variable W_{it} relies on the assumption that each task can be performed in a single processing unit. Task duplication is always required to handle alternative equipment and unit-dependent processing times.

3.1.2.1. Resource balances. Constraint (8) expresses in terms of the variables R_{rt}, the fact that the availability of resource r changes from one time interval to the next one due to the interactions of this resource both with active tasks i and with the environment. The new binary variable $W_{i(t-t')}$ takes the value 1 if task i starts t units of time earlier than time t. In this way, the model is able to easily deal with variable resource requirement during the task execution. The parameter $\prod_{rt}$ defines the amount of resource r provided (positive number) or removed (negative number) from external sources at time t. As expressed by constraint (8), the amount of resource r consumed or released by task i is defined as a combination of a constant and a variable term depending on the task activation and the batch size, respectively. Parameters $\mu_{irt'}$ and $\nu_{irt'}$ indicate fixed and variable proportion of production (positive value) or consumption (negative value) of resource r for a task i at interval t' relative to start of processing of the task. For instance, if r corresponds to a processing unit in which task i requires pt_i units of time, μ_{ir0} is equal to -1 and $\mu_{ir(\mathrm{pt}_i)}$ is equal to 1, which means that the task consumes the processing unit at its starting time and releases the unit at the end of its processing. All other parameters for this task and resource will be zero. Moreover, the maximum availability of resource r has to be limited by constraint (6). In the case of unary resources such as processing units the maximum capacity is always equal to 1

$$R_{rt} = R_{r(t-1)} + \sum_{i \in I_r} \sum_{t'=0}^{\mathrm{pt}_i} (\mu_{irt'} W_{i(t-t')} + \nu_{irt'} B_{i(t-t')}) + \prod\nolimits_{rt} \quad \forall r, t \tag{8}$$

3.1.2.2. Operational constraints. Different types of constraints can be imposed on the operation of a task. For instance, a typical constraint is the minimum and maximum batch size with respect to the capacity of the processing equipment $r \in R_i^J$, which can simply be written as

$$V_{ir}^{\min} W_{it} \le B_{it} \le V_{ir}^{\max} W_{it} \quad \forall i, r \in R_i^J, t \tag{9}$$

3.1.2.3. Sequence-dependent changeovers. Although resource-task network formulations are able to deal with sequence-dependent changeovers, they need to explicitly define additional tasks associated to each type of cleaning requirement as well as different states of cleanliness for each processing unit. Since changeover tasks must be performed in a specific unit, the definition of many identical processing equipment as the same resource can no longer be used. The available processing resources must be defined individually. In this way, different equipment states allow the model to guarantee that the corresponding cleaning task has been performed before starting a particular processing task. The definition of cleaning tasks significantly increases the model size and the computational requirements, making the problem intractable even if a modest number of changeovers need to be considered.

We can then conclude that while the discrete time STN and RTN models are quite general and effective in monitoring the level of limited resources at the fixed times, their major weakness is the handling of long time horizons and relatively small processing and changeover times. Regarding the objective function, these models can easily handle profit maximization (cost minimization) for a fixed time horizon. Other objectives such as makespan minimization are more complex to implement since the time horizon and, in consequence, the number of time intervals required, are unknown a priori (see Maravelias & Grossmann, 2003).

3.2. Global time points (continuous time)

3.2.1. STN-based continuous formulation

A wide variety of continuous-time formulations based both on the STN-representation and the definition of global time points have been developed in the last years (see Fig. 5b). Some of the work falling into this category is represented by the approaches proposed by Giannelos and Georgiadis (2002), Lee, Park, and Lee (2001), Maravelias and Grossmann (2003), Mockus and Reklaitis (1999a,b), Schilling and Pantelides (1996), Zhang and Sargent (1996).

In this section, we describe the formulation by Maravelias and Grossmann (2003), which is able to handle most of the aspects found in standard batch processes (see first column for continuous models in Table 1). This approach is based on the definition of a common time grid that is variable and valid for all shared resources. This definition involves time points n occurring at unknown time T_n, $n = 1, 2, \ldots, |N|$, where N is the set of time points. To guarantee the feasibility of the material balances at any time during the time horizon of interest, the model imposes that all tasks starting at a time point n must occur at the same time T_n. However, in order to have more flexibility in terms of timing decisions, the ending time of tasks does not necessarily have to coincide with the occurrence of a time point n, except for those tasks that need to transfer the material with a zero wait policy (ZW). For other storage policies it is assumed that the equipment can be used to store the material until the occurrence of next time point. Given that the model assumes that each task can be performed in just one processing unit, task duplication is required to handle alternative equipment and unit-dependent processing times. General constraints for this model are introduced below.

3.2.1.1. Assignment constraints. Constraints (10) and (11) define that at most one task i can start ($\mathrm{Ws}_{in} = 1$) or finish ($\mathrm{Wf}_{in} = 1$) at the corresponding unit j at any time n whereas constraint (12) enforces the condition that all tasks that start must finish. In addition, constraint (13) forces that at most one task can be performed at unit j at any time n. This constraint makes use of a full backward aggregation that takes into account the number of tasks that have been started and finished before or at time point n

$$\sum_{i \in I_j} \mathrm{Ws}_{in} \leq 1 \quad \forall j, n \tag{10}$$

$$\sum_{i \in I_j} \mathrm{Wf}_{in} \leq 1 \quad \forall j, n \tag{11}$$

$$\sum_{n} \mathrm{Ws}_{in} = \sum_{n} \mathrm{Wf}_{in} \quad \forall i \tag{12}$$

$$\sum_{i \in Ij} \sum_{n' \leq n} (\mathrm{Ws}_{in'} - \mathrm{Wf}_{in'}) \leq 1 \quad \forall j, n \tag{13}$$

3.2.1.2. Batch size constraints. Minimum and maximum batch sizes are imposed at the beginning as well as at the end of each task through constraints (14) and (15). Additionally, the batch size of each task is also defined for each event where the task is active, as expressed by constraint (16). To guarantee that the batch size does not change during the processing of a task, constraint (17) is also required

$$V_i^{\min} \mathrm{Ws}_{in} \leq \mathrm{Bs}_{in} \leq V_i^{\max} \mathrm{Ws}_{in} \quad \forall i, n \tag{14}$$

$$V_i^{\min} \mathrm{Wf}_{in} \leq \mathrm{Bf}_{in} \leq V_i^{\max} \mathrm{Wf}_{in} \quad \forall i, n \tag{15}$$

$$V_i^{\min} \left(\sum_{n' < n} \mathrm{Ws}_{in'} - \sum_{n' \leq n} \mathrm{Wf}_{in'} \right) \leq \mathrm{Bp}_{in} \leq V_i^{\max} \left(\sum_{n' < n} \mathrm{Ws}_{in'} - \sum_{n' \leq n} \mathrm{Wf}_{in'} \right) \quad \forall i, n \tag{16}$$

$$\mathrm{Bs}_{i(n-1)} + \mathrm{Bp}_{i(n-1)} = \mathrm{Bp}_{in} + \mathrm{Bf}_{in} \quad \forall i, n > 1 \tag{17}$$

3.2.1.3. Material balances. For each state s and time point n, the mass balance and the maximum storage capacity are considered by constraints (18) and (19). In this way, the amount of state s stored at time n will depend on the amount of state s that is (i) stored at time point $n - 1$; (ii) consumed at time n and; (iii) produced at time n. The amount of state s consumed/produced at the start/end of a task i at time point n depends on the batch size and the mass balance coefficients ρ_{is}^{p} and ρ_{is}^{c}. It is worth noting that constraint (19) assumes that a dedicated storage capacity is available for each state. The issue of shared storage tanks is addressed below

$$S_{sn} = S_{s(n-1)} - \sum_{i \in I_s^{\mathrm{c}}} \rho_{is}^{\mathrm{c}} \mathrm{Bs}_{in} + \sum_{i \in I_s^{\mathrm{p}}} \rho_{is}^{\mathrm{p}} \mathrm{Bf}_{in} \quad \forall s, n > 1 \tag{18}$$

$$S_{sn} \leq C_s^{\max} \quad \forall s, n \tag{19}$$

3.2.1.4. Utility constraints. By using this formulation, it is also possible to easily take into account limited resources other than processing units. To do that, constraint (20) carries out a resource balance in each time point n considering the amount of resource r available at time point $n - 1$ as well as the amount of resource r consumed/produced by those tasks starting/ending at time point n. Moreover, the model is able to deal with resource requirements that depend not only on the task activation but also on the batch size. The maximum availability of resource r is enforced by constraint (21)

$$R_{rn} = R_{r(n-1)} - \sum_{i} \mu_{ir}^{\mathrm{c}} \mathrm{Ws}_{in} + \nu_{ir}^{\mathrm{c}} \mathrm{Bs}_{in} + \sum_{i} \mu_{ir}^{\mathrm{p}} \mathrm{Wf}_{in} + \nu_{ir}^{\mathrm{p}} \mathrm{Bf}_{in} \quad \forall r, n \tag{20}$$

$$R_{rn} \leq R_r^{\max} \quad \forall r, n \tag{21}$$

3.2.1.5. Timing and sequencing constraints. The first time point corresponds to the start $T_1 = 0$ and the last to the end $T_n = H$ of the time horizon whereas the ascending ordering of times points is enforced by constraint (22). Also, the ending time of a task i started at time point n is calculated through constraints (23) and (24) by considering the task activation $\mathrm{Ws}_{in} = 1$, the batch size Bs_{in} and the starting time of the task T_n. Thus, the ending time is computed through big-M constraints which are active only if task i starts at time point n. Since a common time grid is used for all shared processing units, the continuous variable T_n defines the time at which all tasks i starting at time point n will begin

$$T_{n+1} \geq T_n \quad \forall n \tag{22}$$

$$\mathrm{Tf}_{in} \leq T_n + \alpha_i \mathrm{Ws}_{in} + \beta_i \mathrm{Bs}_{in} + H(1 - \mathrm{Ws}_{in}) \quad \forall i, n \tag{23}$$

$$\mathrm{Tf}_{in} \geq T_n + \alpha_i \mathrm{Ws}_{in} + \beta_i \mathrm{Bs}_{in} - H(1 - \mathrm{Ws}_{in}) \quad \forall i, n \tag{24}$$

Once the ending of a task i is defined at the time point n where the task is started, constraint (25) defines that the ending time of a task i remains unchanged from its starting time until the next occurrence of the task ($\mathrm{Ws}_{in} = 1$). To guarantee that constraint (25) works properly, we must enforce the condition that Tf_{in} is always greater or equal to $\mathrm{Tf}_{i(n-1)}$. In this way, it is possible to know the ending time of a task i not only at the time point where the task starts, but also at any time point n where the task is activated. This information is used in constraint (26) to express that the ending time of a task i finishing at time point n must be lower or equal than the time at which time point n takes place, i.e. T_n. On the other hand, if task i produces a material for which a zero wait (ZW) storage policy applies, the finish time must coincide with the time point n, which is forced by constraint (27)

$$\mathrm{Tf}_{in} - \mathrm{Tf}_{i(n-1)} \leq H \mathrm{Ws}_{in} \quad \forall i, n > 1 \tag{25}$$

$$\mathrm{Tf}_{i(n-1)} \leq T_n + H(1 - \mathrm{Wf}_{in}) \quad \forall i, n > 1 \tag{26}$$

$$\mathrm{Tf}_{i(n-1)} \geq T_n - H(1 - \mathrm{Wf}_{in}) \quad \forall i \in I^{\mathrm{ZW}}, \quad n > 1 \tag{27}$$

3.2.1.6. Sequence-dependent changeover times. Assuming that changeover times are shorter than processing times, which is a common but not general situation, constraint (28) can be added to account for sequence-dependent changeovers between task i and task i'. Since this model assumes that tasks can only start at time points, the new continuous variable $\mathrm{Ts}_{i'n}$ must be enforced to be equal to T_n. Although this constraint does not require additional variables to handle changeover times, the use of a common grid for all shared resources requires that a larger number of time points be defined in order to consider exact sequence-dependent transition times. Otherwise, most of the changeovers required may be overestimated. It should be noted that due to the definition of additional time points the model may become intractable even for small or medium size problems. In addition, the number of constraints (28) will quickly grow when a large number of tasks can be performed in the same unit j

$$\mathrm{Ts}_{i'n} \geq \mathrm{Tf}_{i(n-1)} + \mathrm{cl}_{ii'} \quad \forall j, i \in I_j, i' \in I_j, n > 1 \tag{28}$$

3.2.1.7. Shared storage tanks. In order to consider the fact that a storage tank can be shared among many states, constraints (29)–(31) have to be added to the model together with a new binary variable V_{jsn} that is 1 if state s is stored in tank j during period n. In this way, allocation constraint (29) allows that at most one state s can be stored in tank j at time n, whereas inequality (30) forces the amount of state s not to exceed the maximum capacity of the tank j. Finally, the total amount of state s available at time n is computed through constraint (31). It is worth mentioning that this set of constraints can only guarantee that (i) the maximum storage capacity is never exceeded and (ii) different states are never simultaneously stored in the same tank. However, the lack of explicit decisions to allocate states to tanks in each time point makes it impossible to enforce the condition that the material stored in a particular tank must remain in the same device until being consumed. Consequently, the schedule generated may be too flexible, allowing a specific amount of material to be stored in different tanks for consecutive time points, which may result infeasible for real batch plants

$$\sum_{s \in S_j} V_{jsn} \leq 1 \quad \forall j \in J^{\mathrm{T}}, n \tag{29}$$

$$S_{sjn} \leq C_j V_{jsn} \quad \forall j \in J^{\mathrm{T}}, s \in S_j, n \tag{30}$$

$$S_{sn} = \sum_{j \in J_s^{\mathrm{T}}} S_{sjn} \quad \forall s \in S^{\mathrm{T}}, n \tag{31}$$

3.2.2. RTN-based continuous formulation

In this section we focus our attention to the most recent continuous-time formulations based on the RTN concept initially proposed by Pantelides (1994). The work developed by Castro, Barbosa-Póvoa, and Matos (2001) which was then improved in Castro, Barbosa-Póvoa, Matos, and Novais (2004) falls into this category and is described below. Major assumptions of this approach are: (i) processing units are considered individually, i.e. one resource is defined for each available unit, and (ii) only one task can be performed in any given equipment resource at any time (unary resource). These assumptions increase the number of tasks and resources to be defined, but at the same time allow reducing the model complexity. This model also covers all the features given at the column on continuous time and global time points in Table 1.

3.2.2.1. Timing constraints. In the same way as in the previous STN-based continuous-time formulation, a set of global time points N is predefined where the first time point takes place at the beginning $T_1 = 0$ whereas the last one at the end of the time horizon of interest $T_n = H$. However, the main difference in comparison to the previous model arise in the definition of the allocation variable $W_{inn'}$ which is equal to 1 whenever task i starts at time point n and finishes at or before time point $n' > n$. In this way, the starting and finishing time points for a given task i are defined through only one set of binary variables. It should be noted that this definition on the one hand makes the model simpler and more compact, but on the other hand it significantly increases the number of constraints and variables to be defined. Constraints (32) and (33) impose that the difference between the absolute times of any two time points (n and n') must be either greater than or equal to (for zero wait tasks) than the processing time of all tasks starting and finishing at those same time points. As can be seen in the equations, the processing time of a task will depend on the task activation as well as on the batch size

$$T_{n'} - T_n \geq \sum_{i \in I_r} (\alpha_i W_{inn'} + \beta_i B_{inn'}) \quad \forall r \in R^{\mathrm{J}}, n, n', (n < n') \tag{32}$$

$$T_{n'} - T_n \leq H \left(1 - \sum_{i \in I_r^{\mathrm{ZW}}} W_{inn'}\right) + \sum_{i \in I_r^{\mathrm{ZW}}} (\alpha_i W_{inn'} + \beta_i B_{inn'}) \quad \forall r \in R^J, n, n', (n < n') \tag{33}$$

3.2.2.2. Batch size constraints. Assuming that each task can only be performed in a single processing unit, limited capacity of equipment is taken into account through constraint (34)

$$V_i^{\min} W_{inn'} \leq B_{inn'} \leq V_i^{\max} W_{inn'} \quad \forall i, n, n', (n < n') \tag{34}$$

3.2.2.3. Resource balances. The resource availability is a typical multiperiod balance expression, in which the excess of a resource at time point n is equal to the excess amount at the previous event point $(n-1)$ adjusted by the amount of resource consumed/produced by all the tasks starting/ending at time point n, as expressed by constraint (35). A special term taking into account the consumption/releasing of storage resources is included for any storage task I^{ST}. Here, negative values are used to represent consumption whereas a positive number defines the production of a resource. Also, the amount of resource available is bounded by constraint (36)

$$R_{rn} = R_{r(n-1)} + \sum_{i \in I_r} \left[\sum_{n' < n} (\mu_{ir}^{\mathrm{p}} W_{in'n} + \nu_{ir}^{\mathrm{p}} B_{in'n}) + \sum_{n' > n} (\mu_{ir}^{\mathrm{c}} W_{inn'} + \nu_{ir}^{\mathrm{c}} B_{inn'}) \right] + \sum_{i \in I^{\mathrm{ST}}} (\mu_{ir}^{\mathrm{p}} W_{i(n-1)n} + \mu_{ir}^{\mathrm{c}} W_{in(n+1)}) \quad \forall r, n > 1 \tag{35}$$

$$R_r^{\min} \leq R_{rn} \leq R_r^{\max} \quad \forall r, n \tag{36}$$

3.2.2.4. Storage constraints. Assuming a single storage task i per material resource r, the definition of constraint (36) in combination with Eqs. (37) and (38) guarantee that if there is an excess amount of the resource r at time point n, then the corresponding storage task i will be activated for both intervals $n-1$ and n. As can be observed in these constraints, dedicated storage tanks with constant minimum and maximum capacities can only be defined for states that are amenable to storage. The use of shared storage tanks is not considered in this MILP formulation

$$V_i^{\min} W_{in(n+1)} \leq \sum_{r \in R_i^{\mathrm{ST}}} R_{rn} \leq V_i^{\max} W_{in(n+1)} \quad \forall i \in I^{\mathrm{ST}}, n, (n \neq |N|) \tag{37}$$

$$V_i^{\min} W_{i(n-1)n} \leq \sum_{r \in R_i^{\mathrm{ST}}} R_{rn} \leq V_i^{\max} W_{i(n-1)n} \quad \forall i \in I^{\mathrm{ST}}, n, (n \neq 1) \tag{38}$$

We can conclude that the continuous time STN and RTN models based on the definition of global time points are quite general. They are capable of easily accommodating a variety of objective functions such as profit maximization or makespan minimization. However, events taking place during the time horizon such as multiple due dates and raw material receptions are more complex to implement given that the exact position of the time points is unknown. Also, the continuous time domain representation makes that inventory cost cannot be estimated without compromising the linearity of the model.

3.3. Unit-specific time event

In order to gain more flexibility in timing decisions without increasing the number of time points to be defined, an original concept of event points was introduced by Ierapetritou and Floudas (1998), which relaxes the global time point representation by allowing different tasks to start at different moments in different units for the same event point (see Fig. 5c). Subsequently, the original idea was implemented in the work presented by Lin, Floudas, Modi, and Juhasz (2002) and Vin and Ierapetritou (2000) and recently extended by Janak, Lin, and Floudas (2004). In this section we describe the work presented in Janak et al. (2004), which represents the most general STN-based formulation that makes use of this type of event representation and covers all the features reported at the corresponding column in Table 1. Due to the fact that the entire formulation involves a very significant number of constraints, only central ones will be reported in this review whereas the remainder can be found in the original work.

3.3.1. Assignment constraints

In order to determine at which event points each task i starts (Ws_{in}), is active (W_{in}) and finishes (Wf_{in}), constraints (39)–(43) enforce the following conditions over the model allocation variables: (i) at most one task i can be being performed in unit j at event time n, (ii) task i will be active at event time n whenever this task has been started before or at event n and has not been finished before that event, (iii) all tasks that start must finish, (iv) one task i can only be started at event point n if all tasks i beginning earlier have finished before event point n and, (v) one task i can only finish at event point n if it has been started at a previous event point n' and has not ended before event point n. It should be noted that equipment index is not used in model variables because this formulation assumes that each task can only be performed in one unit. Task duplication is required to deal with multiple pieces of equipment working in parallel

$$\sum_{i \in I_j} W_{in} \leq 1 \quad \forall j, n \tag{39}$$

$$\sum_{n' \le n} \mathrm{Ws}_{in'} - \sum_{n' < n} \mathrm{Wf}_{in'} = W_{in} \quad \forall i, n \tag{40}$$

$$\sum_{n} \mathrm{Ws}_{in} = \sum_{n} \mathrm{Wf}_{in} \quad \forall i \tag{41}$$

$$\mathrm{Ws}_{in} \le 1 - \sum_{n' < n} \mathrm{Ws}_{in'} + \sum_{n' < n} \mathrm{Wf}_{in'} \quad \forall i, n \tag{42}$$

$$\mathrm{Wf}_{in} \le \sum_{n' < n} \mathrm{Ws}_{in'} - \sum_{n' < n} \mathrm{Wf}_{in'} \quad \forall i, n \tag{43}$$

3.3.2. Batch size constraints

Minimum and maximum batch sizes on all active tasks are imposed through constraint (44). Also, since the formulation allows tasks to extend over several event points, constraints (45) and (46) force batch sizes at these consecutive event points to be consistent. In this way, if a task is active and does not finish at event $n-1$, then the same amount of material will be processed at both event points

$$V_i^{\min} W_{in} \le B_{in} \le V_i^{\max} W_{in} \quad \forall i, n \tag{44}$$

$$B_{in} \le B_{i(n-1)} - V_i^{\max}(1 - W_{i(n-1)} + \mathrm{Wf}_{i(n-1)}) \quad \forall i, n > 1 \tag{45}$$

$$B_{in} \ge B_{i(n-1)} - V_i^{\max}(1 - W_{i(n-1)} + \mathrm{Wf}_{i(n-1)}) \quad \forall i, n > 1 \tag{46}$$

Constraints (47)–(49) determine the batch size at the beginning of a task Bs_{in}, which will be equal to the batch size B_{in} whenever task i starts at event point n. Otherwise, these constraints become redundant. In a similar way, the batch size at the end of task Bf_{in} is defined through constraints (50)–(52)

$$\mathrm{Bs}_{in} \le B_{in} \quad \forall i, n \tag{47}$$

$$\mathrm{Bs}_{in} \le B_{in} + V_i^{\max} \mathrm{Ws}_{in} \quad \forall i, n \tag{48}$$

$$\mathrm{Bs}_{in} \ge B_{in} - V_i^{\max}(1 - \mathrm{Ws}_{in}) \quad \forall i, n \tag{49}$$

$$\mathrm{Bf}_{in} \le B_{in} \quad \forall i, n \tag{50}$$

$$\mathrm{Bf}_{in} \le B_{in} + V_i^{\max} \mathrm{Wf}_{in} \quad \forall i, n \tag{51}$$

$$\mathrm{Bf}_{in} \ge B_{in} - V_i^{\max}(1 - \mathrm{Wf}_{in}) \quad \forall i, n \tag{52}$$

To deal with scheduling problems involving finite intermediate storage capacity, constraint (53) simply represents the maximum amount of material s that can be stored through storage task i^{ST} at any event point n.

$$B_{i^{\mathrm{ST}}n} \le C_s^{\max} \quad \forall s, i^{\mathrm{ST}} \in I_s^{\mathrm{ST}}, n \tag{53}$$

3.3.3. Material balances

The amount of material of state s available at event n is equal to that at event $n-1$ increased by any amounts produced or stored at event $n-1$ and decreased by any amounts consumed or stored at event n

$$S_{sn} = S_{s(n-1)} + \sum_{i \in I_s^{\mathrm{p}}} \rho_{is}^{\mathrm{p}} Bf_{i(n-1)} + \sum_{i^{\mathrm{st}} \in I_s^{\mathrm{ST}}} B_{i^{\mathrm{ST}}(n-1)} - \sum_{i \in I_s^{\mathrm{c}}} \rho_{is}^{\mathrm{c}} Bs_{in} - \sum_{i^{\mathrm{st}} \in I_s^{\mathrm{ST}}} B_{i^{\mathrm{ST}}n} \quad \forall s, n \tag{54}$$

3.3.4. Timing and sequencing constraints (processing tasks)

These constraints represent the relationship between the starting and finishing times of task i at event point n. Then, if task i is not active at event point n, constraint (55) along with (56) makes the processing time equal to zero by setting the finishing time equal to the starting time. In addition, if task i is active and must extend to the following event n, i.e. it does not finish at event $n-1$, constraint (57) along with the sequencing constraint (60) forces the ending time at $n-1$ to be equal to the starting time at n. Otherwise, these constraints are relaxed

$$\mathrm{Tf}_{in} \ge \mathrm{Ts}_{in} \quad \forall i, n \tag{55}$$

$$\mathrm{Tf}_{in} \le \mathrm{Ts}_{in} + HW_{in} \quad \forall i, n \tag{56}$$

$$\mathrm{Ts}_{in} \le \mathrm{Tf}_{i(n-1)} + H(1 - W_{i(n-1)} + \mathrm{Wf}_{i(n-1)}) \quad \forall i, n > 1 \tag{57}$$

Constraints (58) and (59) define the processing time of a task i starting at event n ($\mathrm{Ws}_{in} = 1$) and ending at a later event point $n'(\mathrm{Wf}_{in'} = 1)$. In this way, the two constraints force the ending time at n' to be equal to the starting time at n plus the batch-size dependent processing time. This hard condition is only imposed for those tasks requiring a zero wait storage policy, as expressed in constraint (59). To account for other storage policies, constraint (58) relaxes the processing time in order to consider not only the processing time itself but also the storage time of the material in the processing unit. Constraint (60) defines that the starting time of a task i at event n must be greater than the finishing time of a task i ending at the previous event point

$$\mathrm{Tf}_{in'} - \mathrm{Ts}_{in} \ge \alpha_i \mathrm{Ws}_{in} + \beta_i B_{in} + H(1 - \mathrm{Ws}_{in}) + H(1 - \mathrm{Wf}_{in'}) + H\left(\sum_{n \le n'' \le n'} \mathrm{Wf}_{in''}\right) \quad \forall i, n, n', (n \le n') \tag{58}$$

$$\mathrm{Tf}_{in'} - \mathrm{Ts}_{in} \le \alpha_i \mathrm{Ws}_{in} + \beta_i B_{in} + H(1 - \mathrm{Ws}_{in}) + H(1 - \mathrm{Wf}_{in'}) + H\left(\sum_{n \le n'' \le n'} \mathrm{Wf}_{in''}\right) \quad \forall i \in I^{\mathrm{ZW}}, n, n', (n \le n') \tag{59}$$

$$\mathrm{Ts}_{in} \ge \mathrm{Tf}_{i(n-1)} \quad \forall i, n > 1 \tag{60}$$

Different types of sequencing constraints are proposed for tasks that are performed in the same unit j or in different units j and j'. Then, constraint (61) defines that if task i' ends at event $n-1$ and task i starts at event n in the same unit j, i.e. they are consecutive, task i must start after both task i' and the required cleaning operation have finished. On the other hand, constraints (62) and (63) impose certain sequencing conditions on those tasks that are performed in different units but take place consecutively according to the process recipe. In this way, if a task i' producing a state s finishes at event $n-1$, then any task i consuming that state at event n must start after the ending of task i' at the previous event point. This condition is enforced as equality for those tasks involving a material s that requires a zero wait storage policy, as expressed by constraint (63)

$$\mathrm{Ts}_{in} \geq \mathrm{Tf}_{i'(n-1)} + \mathrm{cl}_{i'i} + H(1 - \mathrm{Wf}_{i'(n-1)} - \mathrm{Ws}_{in}) \quad \forall i, i', i \neq i', j \in J_{ii'}, n > 1 \tag{61}$$

$$\mathrm{Ts}_{in} \geq \mathrm{Tf}_{i'(n-1)} + H(1 - \mathrm{Wf}_{i'(n-1)}) \quad \forall s, i \in I_s^{\mathrm{c}}, i' \in I_s^{\mathrm{p}}, j \in J_i, j' \in J_{i'}, j \neq j', n > 1 \tag{62}$$

$$\mathrm{Ts}_{in} \leq \mathrm{Tf}_{i'(n-1)} + H(2 - \mathrm{Wf}_{i'(n-1)} - \mathrm{Ws}_{in}) \quad \forall s \in S^{\mathrm{ZW}}, i \in I_s^{\mathrm{c}}, i' \in I_s^{\mathrm{p}}, j \in J_i, j' \in J_{i'}, j \neq j', n > 1 \tag{63}$$

3.3.5. Storage constraints

In contrast to global time interval based models, the unit specific time event representation needs to explicitly define a set of storage tasks i^{ST} for dealing with those materials that can be stored in a tank, i.e. where a FIS policy is required. Therefore, a new set of constraints is included into the model to manage related processing and storage tasks. The corresponding starting and ending times of storage tasks at consecutive event points are modeled through additional model variables.

3.3.6. Resource constraints

In order to account for resource limitations other than processing units, the unit-specific-time-event based formulation requires a new set of constraints and variables which monitor the level of resources at every time event. Due to the fact that the same time event can take place at different times for different units, these constraints are significantly more complex and numerous than in the case of global time points. A larger number of event points as well as additional continuous variables for timing of resources are also needed.

We can conclude that continuous time STN formulations based on the definition of unit-specific time events are quite general. They are capable of modeling different scheduling aspects and objective functions. This particular idea proves to be very powerful for those scheduling problems where a few or no shared resources are taken into account, i.e. those cases where reference points for checking resource limitations are barely used. For problems where resources are strongly shared and limited or hard inventory constraints must be satisfied, the use of time events may result less attractive because much more complex models are required. Also, a larger number of event points, similar to the idea of global time points, are usually needed for generating feasible schedules. In this way, the main advantage of this particular idea is lost and larger computational effort may be needed because of the complex structure of the model. As in the case of global time points, events taking place during the time horizon such as multiple due dates and raw material receptions are awkward to consider. Because of the variable event points, inventory cost can only be estimated if additional bilinear constraints are included in the model.

3.4. Time slots

One of the first contributions focused on batch-oriented processes is based on the concept of time slots, which stands for a set of predefined time intervals with unknown durations (Pinto & Grossmann, 1995). A set of time slots is postulated for each processing unit in order to allocate them to the batches to be processed. Relevant work on this area is represented by the formulations developed by Chen, Liu, Feng, and Shao (2002), Lim and Karimi (2003), Pinto and Grossmann (1995, 1996). More recently, a new STN-based formulation that relies on the definition of synchronous time slots and a novel idea of several balances was developed to also deal with network batch processes (Sundaramoorthy & Karimi, 2005). In order to describe the main model constraints and variables, let us consider the original slot-based model proposed in Pinto and Grossmann (1995), assuming a multistage sequential scheduling problem with multiple equipment working in parallel in each stage.

3.4.1. Allocation constraints

Constraint (64) defines that every processing stage l of batch i must be allocated to exactly one time slot k of a unit j belonging to the set of units that can perform the batch task, i.e. J_{il}. In turn, each time slot k of unit j can at most be assigned to one batch processing task corresponding to the stage l of batch i, which is defined through constraint (65)

$$\sum_{j \in J_{il}} \sum_{k \in K_j} W_{ijkl} = 1 \quad \forall i, l \in L_i \tag{64}$$

$$\sum_{i} \sum_{l \in L_i} W_{ijkl} \leq 1 \quad \forall j, k \in K_j \tag{65}$$

3.4.2. Time matching constraints

Slot-based formulations employ two different time coordinates for processing units and batch tasks. The binary variable W_{ijkl} which defines the assignment of stage l of batch i to time slot k of unit j is used to enforce both coordinates to coincide. In this way, when a batch i is allocated to unit j ($W_{ijkl} = 1$), the big-M constraints (66) and (67) become active and the starting times of the unit and the batch are forced to be the same. Otherwise, these constraints are relaxed

$$-M(1 - W_{ijkl}) \leq \mathrm{Ts}_{il} - \mathrm{Ts}_{jk} \quad \forall i, j, k \in K_j, l \in L_i \tag{66}$$

$$M(1 - W_{ijkl}) \geq \mathrm{Ts}_{il} - \mathrm{Ts}_{jk} \quad \forall i, j, k \in K_j, l \in L_i \tag{67}$$

It should be noted that (66) and (67) can be replaced by a set of fewer constraints that involve disaggregated variables and that are tighter as discussed in Pinto and Grossmann (1995). Constraints (68) and (69) force the ending times of the unit and the batch to coincide whenever the allocation variable W_{ijkl} is equal to one. To do that, the starting time, the batch processing time p_{ij} and the setup time Su_{ij} associated to the batch task are taken into consideration in both constraints

$$\mathrm{Tf}_{jk} = \mathrm{Ts}_{jk} + \sum_{i}\sum_{l \in L_i} W_{ijkl}(p_{ij} + \mathrm{Su}_{ij}) \quad \forall j, k \in K_j \tag{68}$$

$$\mathrm{Tf}_{il} = \mathrm{Ts}_{il} + \sum_{j}\sum_{k \in K_j} W_{ijkl}(p_{ij} + \mathrm{Su}_{ij}) \quad \forall j, k \in K_j \tag{69}$$

Since the postulated time slots are sequentially arranged over time, the starting time of slot $k+1$ at every unit j requires that the processing of slot k be finished, which is expressed through constraint (70). In this way, no overlap of time slots is allowed. Additionally, a time relation for every pair of successive processing stages is considered in constraint (71). For instance, in the case of an unlimited intermediate storage policy, the stage $l+1$ of batch i can be performed any time after the completion time of stage l. For a zero wait intermediate storage policy, constraint (71) must be transformed into equality

$$\mathrm{Tf}_{jk} \le \mathrm{Ts}_{j(k+1)} \quad \forall j, k \in K_j \tag{70}$$

$$\mathrm{Tf}_{il} \le \mathrm{Ts}_{i(l+1)} \quad \forall j, k \in K_j \tag{71}$$

3.5. Unit-specific immediate precedence

The concept of batch precedence can be applied to the immediate or the general batch predecessor, which generates three different types of basic mathematical formulations. In this section we present the general constraints and variables for the concept of immediate precedence in each unit. For this particular case, the binary variable $X_{ii'j}$ becomes equal to 1 whenever batch i is processed immediately before batch i' in the processing sequence of unit j. It should be noted that allocation and sequencing decisions are modeled through this variable. To illustrate the use of this concept, let us consider the formulation of Cerdá, Henning, and Grossmann (1997) where a single-stage batch plant with multiple equipment working in parallel is assumed.

3.5.1. Allocation and sequencing constraints

The set of constraints (72)–(75) aims at generating a feasible processing sequence of batches in each available unit. Constraint (72) enforces the condition that at most one batch i can start the processing sequence of unit j. Subsequently, constraint (73) defines that a batch i can be processed either in the first place ($\mathrm{WF}_{ij} = 1$) or right after another batch i' ($X_{i'ij} = 1$), here called its immediate predecessor. This implies that every batch i must be processed in some unit j and have a single predecessor i' at most. Moreover, every batch i can be either allocated to the last position of the processing sequence, or right before another batch i', here called its immediate successor. This condition is enforced through constraint (74). Finally, constraint (75) is employed to guarantee that the immediate predecessor and successor of a given batch i are always assigned to the same processing unit j

$$\sum_{i \in I_j} \mathrm{WF}_{ij} \le 1 \quad \forall j \tag{72}$$

$$\sum_{j \in J_i} \mathrm{WF}_{ij} + \sum_{j \in J_i}\sum_{i' \in I_j} X_{i'ij} = 1 \quad \forall i \tag{73}$$

$$\sum_{i' \in I_j} X_{ii'j} \le 1 \quad \forall i \tag{74}$$

$$\mathrm{WF}_{ij} + \sum_{i' \in I_j} X_{i'ij} + \sum_{\substack{j' \in J_i \\ j \ne j'}} \sum_{i' \in (I_j \cup I_{j'})} X_{ii'j'} \le 1 \quad \forall i, j \in J_i \tag{75}$$

It is worth mentioning that constraints (72)–(75) are not sufficient to prevent the generation of subcycles and, in principle, a large number of subtour elimination constraints should be also included in the model. However, the temporal aspect considered in constraints (76) and (77) contributes to eliminate any possible subcycle from the feasible region and in consequence, subtour elimination constraints are no longer required.

3.5.2. Timing constraints

The timing decisions of batches are modeled through constraints (76) and (77). The first one derives the ending time Tf_i of a batch i from its starting time Ts_i and its processing time Tp_{ij} in the allocated unit j. Then, whenever batch i' is the immediate predecessor of batch i in unit j, i.e. $X_{i'ij} = 1$, constraint (77) imposes that the starting time of batch i must be greater than the ending time of batch i' plus the changeover time $cl_{i'ij}$ in unit j. In this way, it is possible to guarantee that no overlap will occur over time

$$\mathrm{Tf}_i = \mathrm{Ts}_i + \sum_{j \in J_i} \mathrm{Tp}_{ij}\left(WF_{ij} + \sum_{i' \in I_j} X_{i'ij}\right) \quad \forall i \tag{76}$$

$$\mathrm{Ts}_i \ge \mathrm{Tf}_{i'} + \sum_{j \in J_{ii'}} \mathrm{cl}_{i'ij} X_{i'ij} - M\left(1 - \sum_{j \in J_{ii'}} X_{i'ij}\right) \quad \forall i, i' \tag{77}$$

3.6. Immediate precedence

In this section we introduce the general constraints and variables of an alternative formulation based on the concept of immediate batch precedence. In contrast to the previous model, allocation and sequencing decisions are divided into two different sets of binary variables. To illustrate the use of this idea let us consider the work presented by Méndez, Henning, and Cerdá (2000), where a single-stage batch plant with multiple equipment in parallel is assumed. Relevant work following this direction can also be found in Gupta and Karimi (2003). Key variables are defined as follows: WF_{ij} denotes that batch i is the first processed in unit j; W_{ij} denotes that batch i is allocated to unit j but not in the first place and; $X_{ii'}$ denotes that batch i is processed right before batch i'.

3.6.1. Allocation constraints

Constraint (78) defines that at most one batch i can be the first processed in unit j whereas constraint (79) enforces every batch i to be allocated to the processing sequencing of an available unit j

$$\sum_{i \in I_j} \mathrm{WF}_{ij} \leq 1 \quad \forall j \tag{78}$$

$$\sum_{j \in J_i} \mathrm{WF}_{ij} + \sum_{j \in J_i} W_{ij} = 1 \quad \forall i \tag{79}$$

3.6.2. Sequencing-allocation matching constraints

Whenever a pair of batches i, i' are related through the immediate precedence relationship, i.e. $X_{ii'} = 1$, both batches must be allocated to the same unit j. This condition is imposed through inequalities (80) and (81) that relate allocation and sequencing decisions among themselves. The former imposes the condition upon the set the units that can perform both batches whereas the latter is applied to those units that can only process the batch i

$$\mathrm{WF}_{ij} + W_{ij} \leq W_{i'j} - X_{ii'} + 1 \quad \forall i, i', j \in J_{ii'} \tag{80}$$

$$\mathrm{WF}_{ij} + W_{ij} \leq 1 - X_{ii'} \quad \forall i, i', j \in (J_i \backslash J_{ii'}) \tag{81}$$

3.6.3. Sequencing constraints

Every batch i should either be the first processed or directly preceded by another batch i', as expressed in constraint (82). In addition, constraint (83) defines that every batch i can at most be directly succeeded by another batch i', here called its immediate successor. In this way, a feasible processing sequence for every unit is always generated

$$\sum_{j \in J_i} \mathrm{WF}_{ij} + \sum_{i'} X_{ii'} = 1 \quad \forall i \tag{82}$$

$$\sum_{i'} X_{ii'} \leq 1 \quad \forall i \tag{83}$$

3.6.4. Timing constraints

Constraint (84) computes the ending time Tf_i of batch i from its starting time Ts_i and its processing time Tp_{ij} in the allocated unit j. To prevent batch overlapping, constraint (85) states that batch i directly succeeding batch i ($X_{ii'} = 1$) in the jth-unit processing sequence must start after both the ending time of batch i and the corresponding unit and sequence-dependent changeover tasks have taken place

$$\mathrm{Tf}_i = \mathrm{Ts}_i + \sum_{j \in J_i} \mathrm{Tp}_{ij}(\mathrm{WF}_{ij} + W_{ij}) \quad \forall i \tag{84}$$

$$\mathrm{Ts}_{i'} \geq \mathrm{Tf}_i + \sum_{j \in J_{ii'}} (\mathrm{cl}_{ii'} + \mathrm{Su}_{i'j}) W_{i'j} - M(1 - X_{ii'}) \quad \forall i, i' \tag{85}$$

3.7. General precedence

The generalized precedence notion extends the immediate precedence concept to not only consider the immediate predecessor, but also all batches processed before in the same processing sequence. In this way, the precedence concept is completely generalized which simplifies the model and reduces by half the number of sequencing variables when compared to the immediate precedence model. This reduction is obtained by defining just one sequencing variable for each pair of batch tasks that can be allocated to the same resource. Additionally, a major strength of this approach is that sequencing decisions can be easily extrapolated to different types of renewable shared resources. In this way, the use of processing units, storage tanks, utilities and manpower can be efficiently handled through the same set of sequencing variables without compromising the optimality of the solution. Part of the work falling into this category is represented by the approaches developed by Méndez, Henning, and Cerdá (2001) and Méndez and Cerdá (2003a, 2004a,b). Here we assume a multistage sequential scheduling problem with multiple equipment working in parallel in each stage.

3.7.1. Allocation constraints

A single processing unit j must be assigned to every required stage l for manufacturing batch i, here called the task (i, l)

$$\sum_{j \in J_{il}} W_{ilj} = 1 \quad \forall i, l \in L_i \tag{86}$$

3.7.2. Timing constraints

In order to define the exact timing for every batch task (i, l), constraint (87) determines the ending time of the task from the starting and processing time in the assigned unit j. Precedence constraints between consecutive stages $l-1$ and l of batch i are imposed through constraint (88)

$$\mathrm{Tf}_{il} = \mathrm{Ts}_{il} + \sum_{j \in J_{il}} \mathrm{Tp}_{ilj} W_{ilj} \quad \forall i, l \in L_i \tag{87}$$

$$\mathrm{Ts}_{il} \geq \mathrm{Tf}_{i(l-1)} \quad \forall i, l \in L_i, l > 1 \tag{88}$$

3.7.3. Sequencing constraints

Sequencing constraints (89) and (90), which are expressed in terms of big-M constraints, are defined for every pair of tasks (i, l) and (i', l') that can be allocated to the same unit j. If both are allocated to unit j, i.e. $W_{ilj} = W_{i'l'j} = 1$, either constraint (90) or (91) will be active. If task (i, l) is processed earlier than (i', l'), then $X'_{il,i'l'}$ is equal to one and constraint (90) is enforced to guarantee that task (i', l') will begin after completing both the task (i, l) and the subsequent changeover operation at unit j. Moreover, constraint (90) becomes redundant. In case that task (i', l') is run earlier in the same unit, constraint (90) is applied and constraint (89) is relaxed. Otherwise, such a pair of tasks is not carried out at the same unit and, consequently, constraints (89) and (90) become both redundant and the value of the sequencing variable is meaningless for unit j. It should be noted that the precedence concept used in the sequence variable involves not only the immediate predecessor but also all batches processed before in the $X'_{il,i'l'}$ same shared equipment. However, only the immediate predecessor will enforce the minimum starting time on its immediate successor

$$\mathrm{Ts}_{i'l'} \geq \mathrm{Tf}_{il} + \mathrm{cl}_{il,i'l'} + \mathrm{Su}_{i'l'} - M(1 - X'_{il,i'l'}) - M(2 - W_{ilj} - W_{i'l'j}) \quad \forall i, i', l \in L_i, l' \in L_{i'}, j \in J_{il,i'l'} : (i, l) < (i', l') \tag{89}$$

$$\mathrm{Ts}_{il} \geq \mathrm{Tf}_{i'l'} + \mathrm{cl}_{i'l',il} + \mathrm{Su}_{il} - MX'_{il,i'l'} - M(2 - W_{ilj} - W_{i'l'j}) \quad \forall i, i', l \in L_i, l' \in L_{i'}, j \in J_{il,i'l'} : (i, l) < (i', l') \tag{90}$$

3.7.4. Resource limitations

Taking advantage of the concept of general precedence, this formulation is able to deal with resource limitations aside from processing units without predefining reference points for checking resource availabilities. The general idea is to utilize a uniform treatment of resource limitations, where the use of processing units and other resources such as manpower, tools and services is handled through common allocation and sequencing decisions. To do that, the different types of resources r (manpower, tools, steam, energy, etc.) involved in the scheduling problem as well as the individual items or pieces of resources z available for each type r need to be defined. For instance, three operator crews z_1, z_2 and z_3 can be defined for the resource type manpower, here called r_1. Therefore, constraint (91) ensures that sufficient resource r will be allocated to meet the requirement of batch i, where q_{rz} is the amount of resource r available at the resource item z of type r and ν_{ilrj} defines the amount of resource r required when task (i, l) is allocated to unit j, i.e. unit-dependent resource demands can be easily accounted for. In addition, the pair of constraints (91) and (92) enforces the sequential usage of each resource item by using the same idea introduced above for sequencing processing units. It should be noted that the same sequencing variable $X'_{il,i'l'}$ is utilized for processing units and other resources, constraints (89)–(93), which allows generating a simple problem formulation comprising a smaller number of binary variables. New allocation variables Y_{iz} are defined for every item z of each type of resource r.

$$\sum_{z \in R_{ir}} q_{rz} Y_{iz} = \sum_{j \in J_{il}} \nu_{ilrj} W_{ilj} \quad \forall r \in R_i, i, l \in L_i \tag{91}$$

$$\mathrm{Ts}_{i'l'} \geq \mathrm{Tf}_{il} - M(1 - X'_{il,i'l'}) - M(2 - Y_{ilz} - Y_{i'l'z}) \quad \forall i, i', l \in L_i, l' \in L_{i'}, r \in R_{il,i'l'}, z \in Z_r : (i, l) < (i', l') \tag{92}$$

$$\mathrm{Ts}_{il} \geq \mathrm{Tf}_{i'l'} - MX'_{il,i'l'} - M(2 - Y_{ilz} - Y_{i'l'z}) \quad \forall i, i', l \in L_i, l' \in L_{i'}, r \in R_{il,i'l'}, z \in Z_r : (i, l) < (i', l') \tag{93}$$

We can conclude that existing slot and precedence-based formulations are able to efficiently deal with a broad variety of intrinsic characteristics of batch sequential processes. On the one hand, slot-based models are highly efficient if the workload of processing units is well-balanced, i.e. a minimum number of slots needs to be postulated for each processing unit. On the other hand, precedence-based models are usually more effective when problem aspects such as sequence-dependent changeovers or forbidden processing sequences need to be considered. The performance of these models can be significantly improved if a partial or total preordering of batches can be performed a priori by using certain problem data, for instance due dates, colors, flavors, etc. So far, the major limitation of these approaches relies on the treatment of inventory constraints.

4. Comparison of optimization approaches

In the following, the MILP models that have been introduced in the previous sections will be used to solve benchmarking examples taken from literature.

Two case studies for batch scheduling problems arising in process industries are presented. Based on the roadmap introduced in Section 2 (see Fig. 3), a summary of the problem characteristics is given in Table 2. The computational results for the case studies allow comparing the efficiency and limitations of specific modeling approaches.

4.1. Case study I

In order to test the effectiveness and current limitations of discrete and continuous time representations, we performed a computational comparison using MILP models that rely on the definition of global time intervals (Shah et al., 1993) or global time points (Maravelias & Grossmann, 2003). The generality, efficiency and easy implementation of these formulations were the main reasons to choose them within a variety of alternatives. The case study selected is based on the benchmark problem proposed by Westenberger and Kallrath (1995) and later published in Kallrath (2002). This case covers most of the features that contribute to the high complexity of batch scheduling (network structure, variable batch size, storage constraints and, different transfer policies). It has, however, the important simplification that neither changeover times nor non-zero transfer times are

Table 2
Case study features

Feature	Case I	Case II
Process topology	Network	Sequential, single stage
Equipment assignment	Variable	Variable
Equipment connectivity	Full	–
Inventory storage policies	FIS (dedicated), ZW and UIS	UIS
Material transfer	Instantaneous	–
Batch size	Variable	Fixed
Batch processing time	Fixed, unit dependent	Fixed, unit dependent
Demand patterns	Scheduling horizon	Due dates
Changeovers	None	Unit dependent
Resource constraints	None	Discrete (manpower)
Time constraints	None	None
Costs	None	None
Degree of certainty	Deterministic	Deterministic

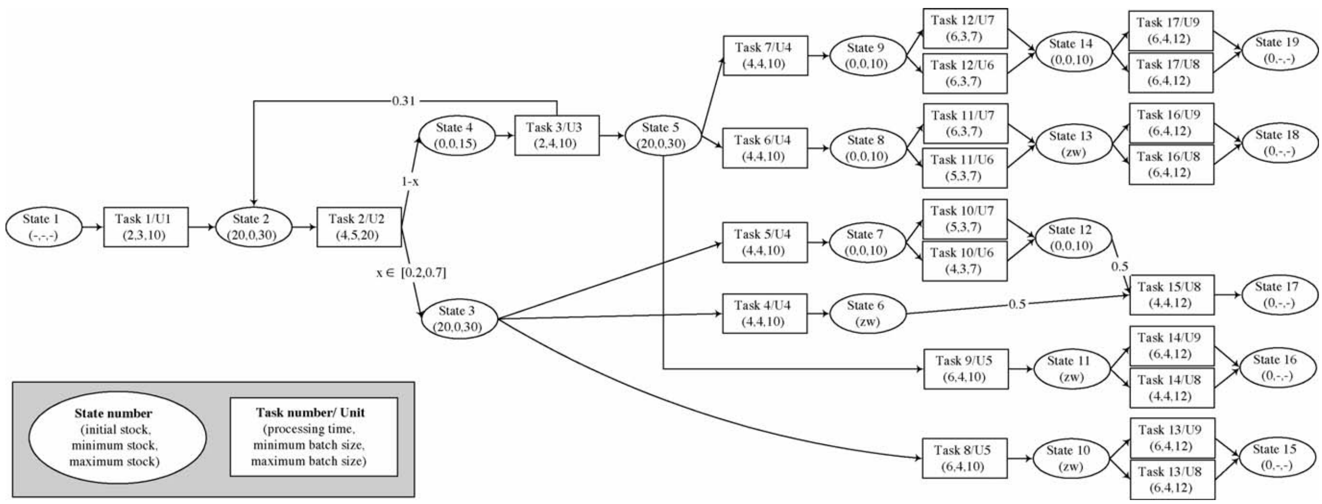


Fig. 7. STN representation for the batch process of case study I.

considered. Fig. 7 provides a graphical representation of this chemical batch process that relies on the state task network (STN) concept introduced by Kondili et al. (1993). Problem data related to states and processing tasks are also described. The STN consists of: state nodes representing the feeds (state 1), intermediates (states 2–14) and final products (states 15–19); task nodes representing the process operations (tasks 1–17) and; arcs that link states and tasks indicating the flow of materials. The available units for performing each batch task are shown within the corresponding rectangle. As shown in Fig. 7, this process comprises 17 processing tasks, 19 states and 9 production units. Fractions of input and output goods are marked on the arcs indicating the particular flow of material. In general, these proportions are fixed. However, the output fractions of task 2 are variable, which means that $100x\%$ of the total output is allotted to state 2 and the remaining quantity to state 3, where the fraction x is allowed to vary between 0.2 and 0.7. Moreover, it is assumed that there is sufficient initial stock of raw material (state 1) and unlimited capacity to store the required raw material (state 1) and the final products (states 15–19). Different intermediate storage polices are taken into account for different states. For instance, a zero-wait transfer policy (ZW) is assumed for states 6, 10, 11 and 13 whereas a finite dedicated intermediate storage capacity (FIS) is considered for the remaining intermediate states.

It is worthwhile to mention that the problem data involves only discrete processing times, which represents a fortunate situation for discrete time models since no special provisions for rounding are needed. In order to evaluate the influence of the objective function on the computational performance, we solved two different instances: minimizing makespan (case 1.a) and maximizing profit (case 1.b). For the makespan, product demands of 20 tons for states 15, 16 and 17 have to be satisfied. Instances comprising a larger number of demands were not possible to be solved in a reasonable time by using the selected pure optimization approaches, which suggests limitations that may be faced when addressing real-world problems. When the profit was maximized, minimum product demands of 10, 10, 10, 5 and 10 tons for states 15, 16, 17, 18 and 19 were considered. Also, original discrete processing times were slightly modified in order to use more realistic data that enforces a finer discretization. Therefore, processing times of 2, 4, 5 and 6 h were changed to 1.3, 3.7, 4.2 and 5.6 h, respectively. Raw material cost, inventory cost, unit operating cost and product values were considered to estimate the total profit of the schedule.

Gantt charts for the optimal solutions for the two instances are shown in Figs. 8 and 9. Model sizes, computational times and objective values are summarized in Table 3. The number of time

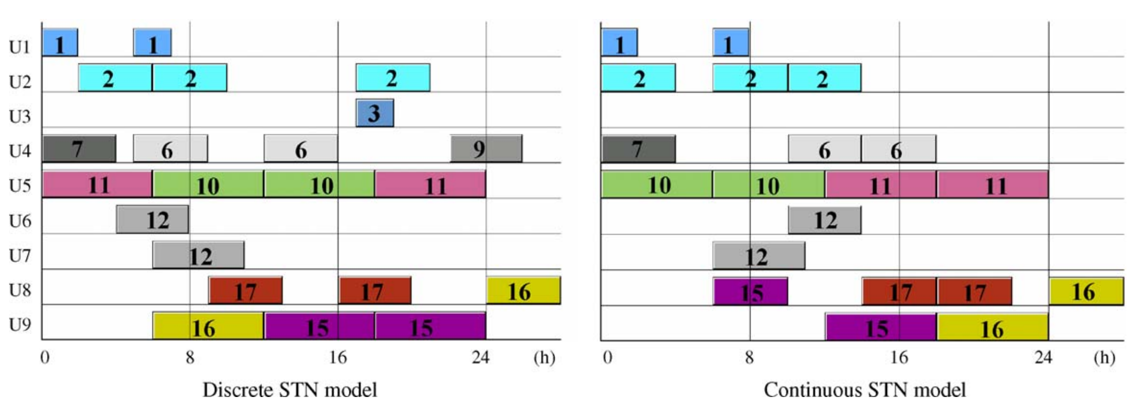


Fig. 8. Gantt charts for case 1.a (makespan minimization).

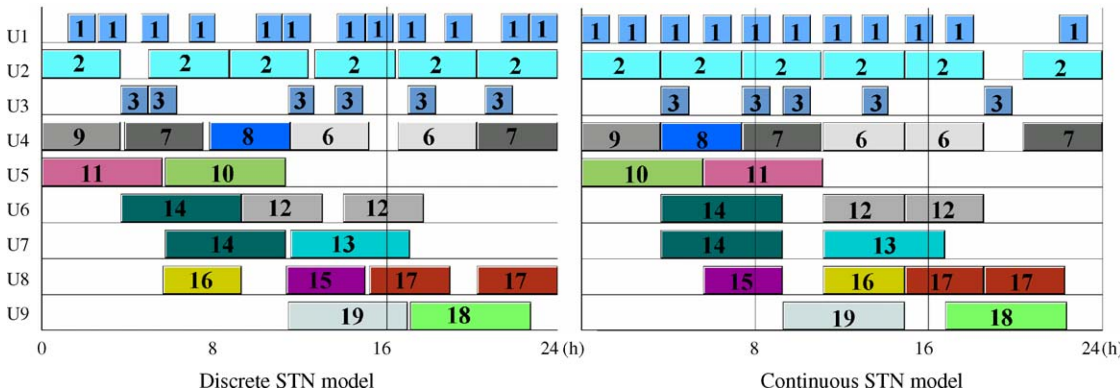


Fig. 9. Gantt charts for case 1.b (profit maximization).

intervals or points that was required in each case is also reported in brackets. For case 1.a, it can be observed that both formulations are able to reach the same objective value of 28 h. Thirty time intervals of 1 h duration were defined for the discrete time, whereas eight variable time points were required for the continuous representation. Only 1.34 s were required by the discrete time model, while 108 s were needed by the continuous model. An iterative procedure that gradually increases the time horizon until a feasible solution is generated was implemented for the discrete time model. In turn, the iterative procedure described in Section 3 was utilized to define the minimum number of variable time points required. The computational effort corresponding to the last iteration for each case is reported in Table 3. However, we would like to remark that the total computational cost for both cases is significantly higher and depends not only on the starting point of the iterative procedure but also on the stopping criterion selected in each iteration.

For the case of profit maximization, a fixed time period of 24 h was assumed. The scheduling horizon was represented through 240 fixed time intervals and 14 variable time points in the discrete and continuous time models, respectively. Longer periods were not possible to be solved in a reasonable time for both time representations. In this case, the solution found through the discrete time model was slightly better than the continuous one, probably because the number of variable time points required for generating a better solution exceeds the current continuous model capabilities. In fact, continuous time models comprising more than 14 time points only generated poor solutions with significant computational effort. With 14 time points the continuous time model was considerably faster than the discrete time model (258 s versus 7202 s).

Although the usefulness and performance of continuous and discrete time models strongly depends on the particular problem and solution characteristics, our experience in the area and the results obtained from the case study performed allow us to draw the following interesting conclusions for general scheduling problems: (i) despite the fact that discrete time models are usually larger than its continuous counterpart, their simpler model structure tends to significantly reduce the CPU time requirements when a reasonable number of time intervals is postulated (around 400 intervals usually appears as a tractable number); (ii) the complex structure of continuous time models makes them useful only for problems that can be solved with reduced number of time points (15 points may be a current upper bound); (iii) discrete time models may generate better and faster solutions than continuous ones whenever the time discretization is a good approximation to the real data; (iv) the model objective function selected may have a notable influence on the computational cost and the model efficiency. Computational costs ranging from 1 to 7202 s were obtained for this case study and; (v) some difficulties for generating near-optimal or even feasible schedules were encountered when solving large-scale problem instances requiring a large number of fixed time intervals or variable time points.

4.2. Case study II

The second case study to be presented here was initially addressed by Pinto and Grossmann (1997) and later studied by Méndez and Cerdá (2002) and Janak et al. (2004). The problem comprises a single stage process with four parallel extruders (U1–U4) with different capacities where a total of 12 batch

Table 3
Computational results for discrete and continuous STN models

Case study	Event representation (time intervals or points)	Binary vars, cont. vars, constraints	LP relaxation	Objective function	CPU time[a]	Relative gap
1.a	Global time intervals (30)	720, 3542, 6713	9.9	28	1.34	0.0
	Global time points (8)	384, 2258, 4962	24.2	28	108.39	0.0
1.b	Global time intervals (240)	5760, 28322, 47851	1769.9	1425.8	7202	0.122
	Global time points (14)	672, 3950, 8476	1647	1407.4	258.54	0.042

[a] Seconds on Pentium IV PC with CPLEX 8.1 in GAMS 21.

Table 4
Comparison of model sizes and computational requirements

Case study	Event representation	Binary vars, cont. vars, constraints	Objective function	CPU time	Nodes
2.a	Time slots and preordering	100, 220, 478	1.581	67.74[a] (113.35)[b]	456
	General precedence	82, 12, 202	1.026	0.11[c]	64
	Unit-based time events (4)	150, 513, 1389	1.026	0.07[d]	7
2.b	Time slots and preordering	289, 329, 1156	2.424	2224[a] (210.7)[b]	1941
	General precedence	127, 12, 610	1.895	7.91[c]	3071
	Unit-based time events (12)	458, 2137, 10382	1.895	6.53[d]	1374
2.c	Time slots and preordering	289, 329, 1156	8.323	76390[a] (927.16)[b]	99148
	General precedence	115, 12, 478	7.334	35.87[b]	19853
	Unit-based time events (12)	446, 2137, 10381	7.909	178.85[d]	42193

[a] Seconds on IBM 6000-530 with GAMS/OSL.
[b] Seconds for disjunctive branch and bound.
[c] Seconds on Pentium III PC with ILOG/CPLEX.
[d] Seconds on 3.0 GHz Linux workstation with GAMS 2.5/CPLEX 8.1.

orders need to be accomplished by minimizing earliness given specific due dates over a 30-day time horizon. The corresponding unit-dependent processing rates and setups as well as the specific due dates are reported in Pinto and Grossmann (1997). In contrast to the previous case study, this problem involves continuous temporal data, which makes the use of pure discrete time models very complex and inefficient. Because of that, our comparison is based on three different optimization approaches that rely on alternative continuous time representations such as time slots, general precedence and unit-specific time events. This comparison attempts to show alternative ways of addressing the same problem through a variety of optimization approaches, highlighting main differences, advantages and limitations in each case. The problem objective is to minimize the total earliness, assuming that due dates are imposed as hard constraints on the completion times. General problem features are again summarized in Table 2.

The scheduling problem is solved considering that only limited manpower (operator crews) is available to operate simultaneously (a) all extruders; (b) three extruders at most; (c) two extruders at most. Model sizes and computational requirements for the alternative approaches to the corresponding cases 2.a, 2.b, 2.c are shown in Table 4. Therefore, Gantt charts describing the optimal solutions obtained for the cases are shown in Fig. 10. Allocation, sequencing and timing decisions related to processing units (U1–U4) and to operator crews (R1–R3) are represented in these diagrams. Also, for the sake of clarity in the presentation of results, tasks that contribute to non-zero earliness are colored darker than those that ended just-in-time. Although this case study can be considered as a relatively small and simple scheduling problem, it still represents a significant challenge for pure optimization techniques. Interestingly, the computational effort in terms of CPU time and nodes clearly reflects the higher complexity of resource constrained scheduling problems, as reported in Table 4. Significant differences in model sizes and in results can be observed for the different approaches evaluated. A major reason is that the MILP models are not the same, and the CPU times and number of nodes were obtained with different computers and with different MILP solvers (OSL and CPLEX) over a 7 year period. This illustrates the combined impact of modeling, speed of computers and sophistication of MILP solvers on the computational effort in solving these problems.

For the time slot model, a mixed integer representation for resource constraints is utilized together with the core of the formulation reported in Section 3.4. Since the pure MILP approach proved to be ineffective, a more efficient logic-based approach was also presented in Pinto and Grossmann (1997). Since the set of new variables and constraints for modeling resource limitations notably increases the model complexity, preordering rules were embedded into the formulation to expedite the search. These rules along with a likely underestimation of the number

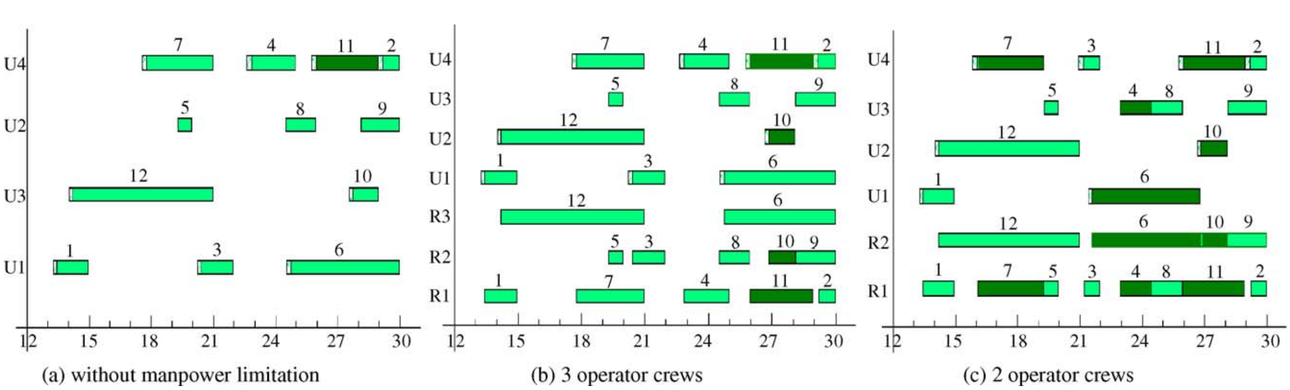


Fig. 10. Optimal schedules for case study 2.

of time slots required can generate suboptimal solutions, which can be observed in the solutions reported for this model.

Subsequently, Méndez and Cerdá (2002) revisited this resource constrained scheduling problem and proposed an optimization approach based on the general precedence concept and a uniform treatment of resource limitations, as described in Section 3.7. Instead of using the standard approach that monitors the level of resources at specific time points, this method employs allocating and sequencing decisions over time to guarantee that resource availabilities are never exceeded. In this case, specific allocating variables were used for processing units and operators crews, whereas a common sequencing variable was used for both shared resources. This was possible because of the use of the general precedence concept. Given that a limited number of checking points was not used to monitor the limited resources, the optimality of the solution can be guaranteed.

Finally, the same scheduling problem was recently addressed by Janak et al. (2004) through the extended version of the formulation based on the definition of unit-specific time events. As can be observed in Table 4, both the resource unconstrained and constrained problems were efficiently solved with a modest computational effort although, curiously, the model sizes reported were significantly larger than the other approaches. For the cases with manpower limitations, the number of event points required was increased from 4 to 12 event points which gave rise to more complicated models involving more variables and constraints. It should be noted that Table 4 only reports the computational statistics for a given number of event points, i.e. 4 and 12 event points, respectively. However, considering the fact that these numbers are unknown a priori, the iterative procedure previously described in Section 3 must be used in each case, which may represent a much higher total CPU time. For the case 2.c, this formulation was not able to reach the actual optimal solution (see Table 4). This situation typically arises when a smaller number of time events than needed is predefined. However, in this particular case the problem was attributed to a special constraint restricting the starting time of a given batch and, consequently, eliminating the optimal solution from the feasible region (Janak, Lin, & Floudas, 2005). The use of special constraints is not mentioned in their original paper but we assume that they were used to speed up the search.

5. Real-world scheduling examples involving complex process considerations

From a mathematical perspective, most scheduling problems found in industrial environments can be regarded as very large-scale combinatorial and complex optimization problems, which rarely can be solved to optimality within a reasonable amount of computational time. Such a combinatorial explosiveness has to do with the increased number of products to be processed, the long sequence of processing stages, the multiple units available for each task and the length of the scheduling horizon to be considered. The complexity arises from a wide range of operational constraints that often need to be taken into account in real world problems. Based on this fact, this section attempts to illustrate the main motivation for developing more realistic and efficient optimization models. A concise description highlighting the major characteristics and difficulties of two challenging industrial problems, both extensively studied by different authors, is presented. They deal with the scheduling of a polymer plant and a steel-making plant, respectively. The section is concluded with some discussion on the sizes of the required mathematical formulations in terms of variables and constraints.

5.1. Scheduling of a polymer batch plant

A real-world scheduling problem from the polymer industries was studied by Schulz, Engell, and Rudolf (1998) and Wang, Löhl, Stobbe, and Engell (2000). It deals with a multiproduct batch plant where two types of expandable polystyrene (EPS) are produced in several grain fractions. Within the considered scheduling horizon a number of orders has to be fulfilled. Each order specification includes information on due date and a given amount of some grain size fraction. The main objective is to satisfy the customer orders with minimum delay. A schematic representation of the plant is shown in Fig. 11.

The considered polymerization process includes three stages: preparation of raw material, polymerization and finishing. The first two stages are operated in batch mode and each stage involves several units running in parallel. The finishing step splits the polystyrene suspension into different grain fractions in a pair of continuous production lines. In the preparation step, batches of input material are mixed in vessels, and then the mixture is pumped into one of several storage tanks and subsequently fed into the polymerization reactor. Every tank has a capacity equal to the batch size in the next polymerization step and is devoted to just one type of polystyrene during the entire horizon. On the other hand, polymerization and finishing steps are connected by mixers in which batches of polystyrene of the same type coming from the reactors are mixed and continuously supplied to the finishing lines. The feed flowrate can change with time but its value must remain within certain bounds. Each finishing line is assigned to just one type of polystyrene and must be shutdown whenever the minimal feed rate condition cannot be satisfied.

Since the polymerizations require the same basic structure with minor variations in some parameters, the plant layout is of the flowshop type. The major input to the polymerization step is a mixture of styrene and some additives coming from the preparation stage. The choice of the additives (i.e. the recipe) determines the grain size distribution and the type of expandable polystyrene

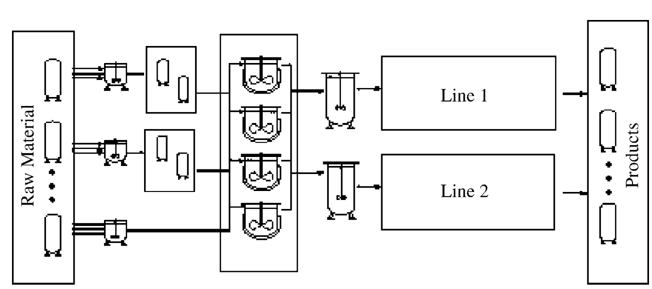


Fig. 11. Schematic representation of the polymer batch process.

being made. For each EPS-type, there are five recipes, each one yielding a different grain distribution. However, the choice of the recipe has a limited influence on the particle size distribution and all grain fractions are produced in significant amounts in all batch runs. Therefore, none of the different products can be produced separately and batches processed in the reactors using different recipes are mixed together before going to the finishing stage.

Overproduction is another major issue. Since the batch size at the polymerization step is constant and the number of recipes is limited, every grain fraction demand can hardly be satisfied exactly. This results in overproduction of certain grain fractions that must be stored. Moreover, unwanted grain fractions to be sold at low prices are always produced. Therefore, the real scheduling goal is not only to produce the required grain size fractions with minimum delay, but also to make them in the right amounts and with least production of unwanted fractions.

Several process constraints are to be satisfied: (1) the batch size in the polymerization stage is constant due to technological restrictions. The filling level of the reactors affects the grain size distribution. (2) For safety reasons, the simultaneous start of two or more polymerization runs in different reactors is prohibited. A minimum delay of 4 h between the starts of two subsequent polymerizations in different reactors is required. (3) Each mixer connecting reactors to finishing lines is allocated to just one type of polymerization and the assignment must remain fixed along the scheduling horizon. (4) At the end of a polymerization, enough volume must be available in the corresponding mixer so that the reactor can be emptied. (5) If the mixers run empty, the finishing line connected to them has to be temporarily shut down. In some cases, the shutdown of the finishing lines is not permitted and, therefore, the feed flowrate should be properly adjusted with respect to time to always have enough material in the mixers. (6) Tracking of each grain fraction concentration in every mixer with respect to time is required to establish the fraction feed rate to the corresponding finishing line. To do so, non-linear mass balances around the mixer must be included in the problem formulation.

The major decisions of discrete and continuous nature include the following:

(a) The choice of the recipes and the number of batches (polymerizations) to be processed. There are five choices for each type of polymerization. In addition, a maximum number of 36–70 polymerizations can be performed over a time horizon ranging from 8 to 14 days, respectively (Wang et al., 2000).
(b) The assignment of a reactor unit and the timing of every polymerization.
(c) The mixer assignment to each polymerization, as well as, both the concentration of each grain size fraction and the total mass in the mixer after completing the loading of a polymerization batch.
(d) The feed rates of the separation stages and the total output of each grain fraction at any time.
(e) The timing of the start-up and shut-down of the finishing lines.

Assuming that 36 polymerizations (batches) are made using one of the ten available recipes, the total number of possible tasks adds up to 360. Moreover, the whole set of batches processed in the polymerization stage must be ordered to account for the minimum delay of 4 h between subsequent polymerizations in different reactors. Therefore, there are 36! possible sequences of batches in the polymerization stage. Preordering of batches by due dates cannot be applied since every polymerization produces significant amounts of each grain size fraction; i.e. the batches are coupled. In addition, a huge number of continuous variables are also required. For instance, the actual concentration of every fraction in each mixer and the feed flowrates to the finishing lines must be handled. Considering the production of ten different fractions and the running of four mixers, a total of 4 mixers $\times$ 36 polymerizations $\times$ 10 different fractions = 1440 concentration variables must be defined. Since the problem is intrinsically non-linear, it must be represented through a mixed-integer non-linear mathematical problem formulation (MINLP). Adopting a scheduling horizon of 8 days, Schulz et al. (1998) developed a problem formulation involving 2656 variables of which 1009 are binary variables. Given that the size of the problem, especially the large number of binary variables, makes it impossible to use general purpose algorithms, they presented a special scheduling algorithm, which takes particular problem features into account, leading to a good suboptimal solution with a reasonable CPU time.

5.2. *Scheduling of a steel-making casting plant*

The production scheduling of a steel-making continuous casting plant producing a wide variety of steel ingots in a production line has been recognized as one of the most difficult industrial scheduling problems (Harjunkoski & Grossmann, 2001; Pacciarelli & Pranzo, 2004). Products are characterized by their width, thickness and chemical composition or grade. Each grade has a given production recipe with strict specifications of temperature, chemistry and processing times at the different production stages. Grades are further subdivided into sub-grades with minor differences in, for instance, the carbon content.

The production is organized by orders or lots, each one composed of a given number of ladles with similar product grades to be cast consecutively. The size of a lot may typically vary from one to eight ladles. The scheduling horizon is often 1 week, during which typically an average of 30 orders and 120 ladles are made. Given the customer orders, the equipment items and the quality constraints, the scheduling problem consists of completing all the production requirements at minimum makespan, thus maximizing the throughput of the plant.

The processing of stainless steel consists of a sequence of high temperature operations starting with the loading of scrap iron into an electric arc furnace (EAF) and ending with the continuous casting (CC). The molten steel from the EAF is poured into ladles that a crane transports to a subsequent equipment called argon oxygen decarburization unit (AOD), where mainly the carbon is removed by argon and oxygen injection in order to meet the steel quality requirements. After the AOD, the ladles are transported to a ladle furnace (LF) for secondary metallurgy operations, such as chemical adjustments (e.g. nickel, oxygen, nitrogen, hydrogen contents), degassing and temperature homogenization. In practice, LF also acts as a buffer to maintain the ladles at the proper temperature before the last operation in the continuous caster (CC). Between the LF and the CC there is a buffer that can hold at most one ladle. A ladle can stay in the buffer at most for 10 min, otherwise the liquid steel may cool down and must be reheated to the correct temperature. In the CC, the liquid steel is cast and cooled to form slabs. The time required for casting one ladle ranges from 60 to 70 min.

In the CC operation, the melt steel is solidified into slabs of a pre-specified width and thickness. In order to achieve the desired properties of the final products, the slab formation process has strict requirements of material continuity and casting speed to fulfill. When a continuous steel flow is broken, the caster needs maintenance and the caster mold needs to be replaced, which involves high costs and a delay in production. A new setup of the caster means several hours of interruption in the casting. This happens, for instance, when either the slab thickness or the grade is changed. If two subsequent products have a similar thickness and grade, it may be possible to proceed without stopping. Otherwise, the caster needs to be stopped for service. Moreover, the caster can only be run continuously for a limited number of compatible ladles or products due to the extreme operating conditions. Therefore, the continuous casting process can be considered as one of the major challenges in steel production planning, and even obtaining feasible solutions is not trivial. It has been addressed separately in several studies, see for instance Tang, Liu, Rong, and Yang (2000). The steel production process is illustrated in Fig. 12.

Scheduling 80–100 orders on a production system involving a sequence of four processing stages with some parallel units and subject to many operational restrictions is a highly complex combinatorial problem requiring a huge number of 0–1 sequencing variables and constraints. For simplification of the problem representation, a common approach by practitioners and reflected in research is grouping of customer orders, often named as heats, into a smaller number of sequences. Heats featuring the same grade and/or related subgrades and similar slab thickness can be cast in the same sequence. Members of a sequence are then arranged by monotonically decreasing or increasing slab width and increasing carbon content. In this way, the grouping strategy aims to minimize the number of sequences and, consequently, the overall casting setup time. The resulting sequences are to be ordered such that the schedule makespan and the average order tardiness/earliness are both minimized.

Harjunkoski and Grossmann (2001) studied the scheduling of a steel-making continuous casting plant producing up to 82 different ladles or products within a 1-week time horizon. If this industrial example were formulated as a single scheduling problem, the mathematical model would include as many as 74,000 equations and 34,000 variables of which more than 33,000 are discrete. Most likely such a large MILP problem is not solvable, at least in the near future. Instead, Harjunkoski and Grossmann (2001) applied a three-stage decomposition strategy in order to: (1) optimally group ladles into sequences so as to minimize the total CC setup time, (2) find a detailed schedule of each sequence at every production stage that reduces the makespan and the buffer hold-time violations and (3) determine the proper ordering of sequences to decrease the number of caster mold thickness changes while accounting for the order due dates. Since the products are clustered into 20–25 groups, the resulting MILP mathematical models for the steps (1) and (3) remain still quite large and the formulation for step (3) may involve as many as 16,000 variables (620 binary) and 15,643 constraints for the last step of the solution strategy. Although the CPU time required may exceed 10,000 CPU-s, the predicted schedules lie within 3% of the theoretically optimal makespan.

6. Alternative solution approaches

While this paper has been focused on optimization approaches and related modeling aspects, it is important to note that there are other solution methods for dealing with short-term scheduling of batch processes. These methods can be used either as alternative methods, or as methods that can be combined with MILP. As seen in Fig. 13 there is a great variety of solution methods for solving scheduling problems.

This paper has dealt with MILP methods where the most common solution algorithms are LP-based branch and bound methods (Wolsey, 1998), which are enumeration methods that solve LP subproblems at each node of the search tree (Dakin, 1965). Cutting plane techniques, which were initially proposed by Gomory (1958), and which consist of successively generating valid inequalities for the relaxed MILP problem, have received renewed interest through the work of Crowder, Johnson, and Padberg (1983), Van Roy and Wolsey (1986), and especially the lift and project method of Balas, Ceria, and Cornuejols (1993).

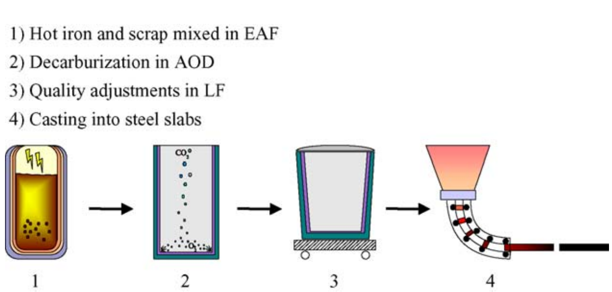


Fig. 12. Steel making process.

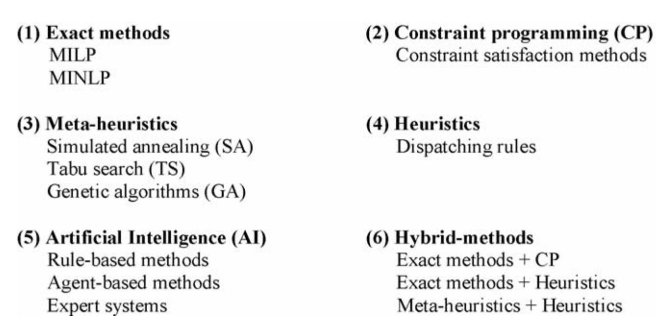


Fig. 13. Solution methods used in batch scheduling problems.

Currently most MILP methods correspond to branch-and-cut techniques in which cutting planes are generated at the various nodes of the branch and bound tree in order to tighten the LP relaxation. A recent review of branch and cut methods can be found in Johnson, Nemhauser, and Savelsbergh (2000). Finally, Benders decomposition (Benders, 1962) is another technique for solving MILPs in which the problem is successively decomposed into LP subproblems for fixed 0–1 and a master problem for updating the binary variables.

The major software packages for MILP are CPLEX (ILOG, 1999) and XPRESS (Dash Optimization, 2003), which use the LP-based branch and bound algorithm combined with cutting plane techniques. These codes have seen tremendous progress over the last decade in terms of capabilities for solving much larger problem sizes and achieving several order of magnitude reductions in the speed of computation, as discussed in Bixby, Fenelon, Gu, Rothberg, and Wunderling (2002) and Bixby (2002). MILP models and solution algorithms have been developed and successfully applied to many industrial problems (e.g. see Kallrath, 2000). In addition, some work has been done in the area of batch scheduling to try to accelerate the branch and bound method by using some problem specific structure, as reported in Shah et al. (1993) and Burkard and Hatzl (2005). It should also be noted that MINLP models may arise in batch scheduling problems. Particularly, modeling the effect of inventories may give rise to non-linearities in the objective function. For a review on MINLP methods see Grossmann (2002).

Constraint Programming (CP) (Van Hentenryck, 1989; Van Hentenryck, 2002; Hooker, 1999) is a relatively new modeling and solution paradigm that was originally developed to solve feasibility problems, but it has been extended to solve optimization problems, particularly scheduling problems. Constraint Programming is very expressive since continuous, integer, as well as Boolean variables are permitted and moreover, variables can be indexed by other variables. Furthermore, a number of constructs and global constraints have also been developed to efficiently model and solve specific problems, and constraints need neither be linear nor convex. The solution of CP models is based on performing constraint propagation at each node by reducing the domains of the variables. If an empty domain is found the node is pruned. Branching is performed whenever a domain of an integer, binary or boolean variable has more than one element, or when the bounds of the domain of a continuous variable do not lie within a tolerance. Whenever a solution is found, or a domain of a variable is reduced, new constraints are added. The search terminates when no further nodes must be examined. The effectiveness of CP depends on the propagation mechanism behind constraints. Thus, even though many constructs and constraints are available, not all of them have efficient propagation mechanisms. For some problems, such as scheduling, propagation mechanisms have been proven to be very effective. Some of the most common propagation rules for scheduling are the “time-table” constraint (Le Pape, 1998), the “disjunctive-constraint” propagation (Baptiste, Le Pape, & Nuijten, 2001), the “edge-finding” (Nuijten, 1994) and the “not-first, not-last” (Baptiste et al., 2001). Software for constraint programming includes OPL and ILOG Solver from ILOG, CHIP (Dincbas et al., 1988), and ECLiPSe (Wallace, Novello, & Schimpf, 1997).

CP methods have proved to be quite effective in solving certain types of scheduling problems, particularly those that involve sequencing and resource constraints. However, they are not always effective for solving more general optimal scheduling problems that involve assignments. Therefore the use of constraint programming in combination with MILP techniques, known as hybrid methods (see Fig. 13), has recently received attention since the two techniques are complementary to each other. Significant computational savings have been reported by Harjunkoski and Grossmann (2002), Jain and Grossmann (2001) and Maravelias and Grossmann (2004) using hybrid methods in which assignment decisions are handled by an MILP subproblem and sequencing decisions by a CP subproblem.

It should be noted that methods based on meta-heuristics, or also known as local search methods, do not make any assumptions on the functions as they are often inspired by moves arising in natural phenomena. For larger scheduling problems the use of a local search algorithms such as Simulated Annealing (Aarts & Korst, 1989; Kirkpatrick, Gelatt, & Vechi, 1983), Genetic Algorithms (Goldberg, 1989), or Tabu Search (Glover, 1990) may be preferable, since these algorithms can obtain good quality solutions within reasonable time. Therefore, these techniques have become popular for optimizing certain types of scheduling problems. However, these algorithms also have significant drawbacks - they do not provide any guarantee on the quality of the solution obtained, and it is often impossible to tell how far the current solution is from optimality. Furthermore, these methods do not formulate the problem as a mathematical program since they involve procedural search techniques that in turn require some type of discretization or graph representation, and the violation of constraints is handled through ad hoc penalty functions. For that reason, the use of meta-heuristics based on local search methods might be problematic for problems involving complex constraints and continuous variables. In this case, the set of feasible solutions might lack nice properties and it might even be difficult to find a feasible solution (see Burkard, Hujter, Klinz, Rudolf, & Wennink, 1998). Tabu Search is the more deterministic of the three techniques and also has fewer tunable parameters. The variant of Tabu Search called Reactive Tabu Search (RTS) (Battiti & Tecchiolli, 1994) has proved to be the more successful implementation for scheduling problems. Examples of application of these techniques in batch scheduling include the work by Graells, Cantón, Peschaud, and Puigjaner (1998), Lee and Malone (2000) and Ryu, Lee, and Lee (2001) for simulated annealing, Löhl, Schulz, and Engell (1988) for genetic algorithms, and Cavin, Fischer, Glover, and Hungerbühler (2004) for tabu search.

Meta-heuristics are also known as improvement heuristics, given that they employ an iterative procedure that starts with an initial schedule that is gradually improved. On the other hand, there are several heuristics called dispatching rules which are considered as construction heuristics. These rules use certain empirical criteria to prioritize all the batches that are waiting for processing on a unit. For simple scheduling problems, they have demonstrated to have very good performance, although their efficiency is usually evaluated empirically. The usefulness of dispatching rules is still limited to quite a narrow variety of scheduling problems and optimality can be proved only in some special cases. Some relevant dispatching rules are: FCFS (first come first served), EDD (earliest due date), SPT (shortest processing time), LPT (longest processing time), ERD (earliest release date), WSPT (weighted shortest processing time). Often, composite dispatching rules involving a combination of basic rules can perform significantly better. Besides, dispatching rules can be easily embedded in exact models to generate more efficient hybrid approaches for large-scale scheduling problems. An extensive review and a classification of various dispatching rules can be found in Blackstone, Phillips, and Hogg (1982) and Panwalkar and Iskander (1977).With the main goal of making a more efficient use of the process information as well as the essential knowledge provided by human schedulers, artificial intelligence (AI) techniques have also been widely applied to scheduling problems. AI is the mimicking of human thought and cognitive processes to solve complex problems automatically. It uses techniques for writing computer code to represent and manipulate knowledge. Different techniques mimic the different ways that people think and reason. For instance, case-based reasoning (CBR) solves a current problem by retrieving the solution to previous similar problems and altering those solutions to meet the current needs. It is based upon previous experiences and patterns of previous experiences. On the other hand, model-based reasoning (MBR) concentrates on reasoning about a system's behavior from an explicit model of the mechanisms underlying that behavior. Within the AI field, agent-based approaches are software programs that are capable of autonomous, flexible, purposeful and reasoning action in pursuit of one or more goals. They are designed to take timely action in response to external stimulus from their environment on behalf of a human. Scheduling problems have been solved by a set of individual agents (see Rabelo & Camarinha-Matos, 1994), which can work parallel and their coordination may bring a more effective way to find an optimal solution. When multiple agents are being used together in a system, individual agents are expected to interact together to achieve the goals of the overall system. A survey by Shen and Norrie (1999) reports 30 projects using agent technology for manufacturing planning, scheduling and execution control where agents represent physical entities, processes, operations, parts, etc.The development of expert systems, also known as knowledge-based approaches, is also an important field of the AI area. They encapsulate the specialist knowledge gained from a human expert (such as an experienced scheduler) and apply that knowledge automatically to make decisions. The process of acquiring the knowledge from the experts and their documentation and successfully incorporating it in the software is called knowledge engineering, and requires considerable skills to perform successfully. Some interesting applications based on AI technologies for addressing real-world scheduling problems have been reported in Henning and Cerdá (2000), Sauer and Bruns (1997) and Zweben and Fox (1994).

7. Use of exact methods in industrial problems

The vast literature in the scheduling area highlights the successful application of different optimization approaches to an extensive variety of challenging problems. Nowadays, more difficult and larger problems than those studied years ago can be solved, sometimes even to optimality, in a reasonable time by using more efficient integrated mathematical frameworks. This important achievement comes mainly from the remarkable advances in modeling techniques, algorithmic solutions and computational technologies that have been made in the last few years.

Although a promising near future in the area can be predicted from this optimistic current situation, it is also well-known that the actual gap between practice and theory is still evident. New academic developments are mostly tested on complex but relatively small problems whereas current real-world applications consist of hundreds of batches, dozens of pieces of equipment and long scheduling periods, usually ranging from one to several weeks. Industrial problems are also very hard-constrained, which means that optimization solvers have to find the optimal or near-optimal solutions in a huge search space with a relatively small feasible region. This difficulty can be easily observed in the reported results for case study II. This may result in unstable and unpredictable computational performance of optimization models, which is definitely not suitable for industrial environments.

In order to make the use of exact methods more attractive in the real-world, increasing effort has been oriented towards the development of systematic techniques that allow maintaining the number of decisions at a reasonable level, even for large-scale problems. A reduced search space can guarantee a more stable and predictable behavior of the optimization model. Manageable model sizes may be obtained by applying heuristic model reduction methods, decomposition or aggregation techniques. Additionally, once the best possible solution has been generated in the specified time, optimization-based techniques could be employed to gradually enhance a non-optimal solution with very modest computational effort. A clear disadvantage of these techniques is that the optimality of the solution can no longer be guaranteed. However, requiring optimality may not be relevant in practice due to the following: (1) a very short time is available to generate a solution, (2) optimality is easily lost because of the dynamic nature of industrial environments, (3) implementing the schedule as such is limited by the real process and (4) only a subset of the actual scheduling goals are taken into account.

Some available techniques widely used to deal with large-scale problems are described below.

1. Heuristic model reduction methods: taking advantage of an empirical and well-known solution method or a particular feature of the problem addressed, it is frequently convenient to incorporate this essential knowledge into the mathematical problem representation. Performing this task usually permits not only to obtain reduced models that describe only the critical decisions to be made but also to generate good solutions in a reasonable time. A clear example of this strategy is given by multiple sequential process-oriented models that make use of

simple or combined dispatching rules, also called preordering rules, to generate better solutions in a given short time. (See Blömer & Günther, 2000; Cerdá et al., 1997; Méndez et al., 2001; Pinto & Grossmann, 1995).

2. Decomposition and aggregation techniques: two major approaches are to either consider aggregation techniques, or else to use decomposition either in spatial or in temporal forms. Examples of strategies based on aggregation are works by Bassett et al. (1997), Birewar and Grossmann (1990) and Wilkinson (1996). These include aggregating later time periods within the specified time horizon in order to reduce the dimensionality of the problem, or to aggregate the scheduling problem so that it can be considered as part of a planning problem. Approaches based on spatial or temporal decomposition, usually rely on Lagrangean decomposition (Graves, 1982; Gupta & Maranas, 1999). In the case of spatial decomposition the idea is to use the links between subsystems (e.g. manufacturing, distribution and retail) by dualizing the corresponding interconnection constraints, which then requires the multiperiod optimization of each system. In the case of temporal decomposition the idea is to dualize the inventory constraints in order to decouple the problem by time periods. The advantage of this decomposition scheme is that consistency is maintained over every time period (Jackson & Grossmann, 2003).
3. Improvement optimization-based techniques: the gradual improvement of a non-optimal solution can be interpreted as a special case of rescheduling where the available solution is partially adjusted with the only goal of enhancing a particular scheduling criterion. These techniques use the entire current schedule as the starting point of a procedure that, based on the problem representation, iteratively enhances the existing solution in a systematic manner. The model size remains usually under user control by allowing that only a small number of potential changes be performed in each iteration. The work that has followed this direction has shown promising results with modest computational effort (see Méndez & Cerdá, 2003b; Roslöf, Harjunkoski, Björkqvist, Karlsson, & Westerlund, 2001).

8. Academic and commercial software for scheduling of batch plants

With few exceptions, academic software for batch scheduling is normally available as part of a modeling system such as GAMS and AMPL. Therefore, there are relatively few academic software packages that can be used as commercial packages and which involve sophisticated graphical user interfaces. For the sake of brevity, we only present a table of a number of academic groups who are actively working in the area of batch scheduling and are known to have unique computational tools for batch scheduling. As can be seen in Table 5, there is a growing number of active researchers in batch scheduling, although the size of this community is still rather modest.

Commercial software for batch scheduling, on the other hand, has only begun to emerge over the last few years. Table 6 lists few representative software packages that are currently available in the market, and for which the users only need to specify data on the problem at hand. Establishing the precise capabilities and methodologies behind these packages is not an easy task since vendors do not normally disclose the full technical information behind these packages. We should also clarify that there are several software packages available that require that the user model the scheduling problem as a mixed-integer program or a constrained programming problem. Examples of the former type of software includes systems like GAMS, AMPL, AIMMS, while examples of the former include systems like OPL, CHIP and ECLIPSE, although among these OPL can handle both MILP as well as CP models. Furthermore, OPL has access to the special purpose software ILOG scheduler, which is especially suitable for batch scheduling problems.

8.1. Aspen Plant Scheduler

Aspen Plant Scheduler from AspenTech (http://www.aspentech.com) is a member of the MIMI family of supply chain solutions (Jones & Baker, 1996). Its objective is to create an optimal or near-optimal short-term schedule for unit production, consistent with the longer-term group production plan, to address the inevitable variability in actual versus planned customer orders. The solution is developed at the end item level (i.e. a shippable, billable item) scheduled by production work center by start and stop times, shift, day, or other finite time period. Decision rules and heuristics are used to speed creation of an executable schedule for large numbers of items sharing limited capacity. Users typically generate a schedule for a time horizon spanning a few days to a few weeks. The solution is integrated with the Aspen Available-to-Promise/Capable-to-Promise solution, to enable rapid response to customer requests for new orders, make-to-order items, and new product formulations. The solution is also linked to Aspen Collaborative Forecasting and Collaborative Replenishment solutions to link the Aspen client to both suppliers and customers. Finally, integration from Aspen Supply Planner allows for direct conversion of an annual plan to a more granular schedule used to organize final staging, testing, and product distribution.

8.2. Model Enterprise Optimal Single-site Scheduler (OSS scheduler)

The OSS scheduler from Process Systems Enterprise Ltd. (http://www.psenterprise.com) determines optimal production schedules for given availabilities of plant resources, recipe information and known product demands, using as a basis the STN and RTN MILP models by Kondili et al. (1993) and Pantelides (1994), respectively. The OSS scheduler determines an economically optimal schedule for a process plant producing multiple products. It is especially suited to multi-purpose plants where products can be processed on a selection of alternative equipment, via different routes and in any batch size. The objective of the schedule can be configured according to the economic requirements of the operation, for example, to deliver maximum profit, maximum output or on-time in full. The schedules produced satisfy all operating constraints such as hard

Table 5
Academic groups with software for batch scheduling

School	Researcher(s) (weblink)
Åbo Akademi University	*T. Westerlund* (http://www.abo.fi/~twesterl/)
Carnegie Mellon University	*I.E. Grossmann* (http://egon.cheme.cmu.edu)
Imperial College	*C. Pantelides*, *N. Shah* (http://www.ps.ic.ac.uk)
INETI	*P. Castro* (http://www.ineti.pt/colaborador/colaborador.aspx?id=7540&UO=&subUO=)
Instituto Superior Lisbon	*A. Barbosa Povoa* (http://alfa.ist.utl.pt/~d3662/)
INTEC – CONICET	*J. Cerdá and C. Méndez* (http://intecwww.arcride.edu.ar/~jcerda/)
National University of Singapore	*I.A. Karimi* (http://www.chee.nus.edu.sg/staff/000731karimi.html)
Polytechnic University	*J. Pinto* (http://www.poly.edu/faculty/josempinto/)
Princeton University	*C.A. Floudas* (http://titan.princeton.edu/home.html)
Purdue University	*J. Pekny and G.V. Reklaitis* (http://engineering.purdue.edu/ChE/Research/Systems/index.html)
Rutgers University	*M. Ierapetritou* (http://sol.rutgers.edu/staff/marianth/)
Technical University Graz	*R. E. Burkard* (http://www.opt.math.tu-graz.ac.at/burkard/)
Universitat Politécnica de Catalunya	*L. Puigjaner* (http://deq.upc.es/wwwdeq/cat/infogral/curriculs/Lluis%20Puigjaner.htm)
University College London	*L. Papageorgiou* (http://www.chemeng.ucl.ac.uk/staff/papageorgiou.html)
University Karlsruhe (TH)	*K. Neumann* (http://www.wior.uni-karlsruhe.de/LS_Neumann/)
University of Dortmund	*S. Engell* (http://www.bci.uni-dortmund.de/ast/en/content/mitarbeiter/elehrstuhlinhaber/engell.html)
University of Sao Paulo	*J. Pinto* (http://www.lscp.pqi.ep.usp.br/pro_zeca.html)
University of Tessaloniki	*M. Georgiadis* (http://www.cperi.certh.gr/en/compro.shtml#SECT2)
University of Wisconsin	*C. Maravelias* (http://www.engr.wisc.edu/che/faculty/maravelias_christos.html)

and soft delivery deadlines. The OSS scheduler can be applied to both continuous and batch processing. Intermediate products can be stored in vessels, in individual tanks or a tank farm. The application accepts complex recipes with blending, separation and re-cycles. Changeovers and downtime can be included and cleaning can be added as downtime or even as a process. The OSS scheduler can also be used to design the economically optimum process plant for a given production requirement.

Table 6
Batch scheduling Software

Software	Vendor
Aspen Plant Scheduler	AspenTech
Model Enterprise Optimal Single-site Scheduler	Process systems Enterprise
VirtECS Schedule	Advanced Process Combinatorics
SAP Advanced Planner and Optimizer (APO)	SAP

8.3. VirtECS Schedule

VirtECS Schedule from Advanced Process Combinatorics (http://www.combination.com) builds an optimized schedule that satisfies all constraints and levels load on parallel equipment. The package is based on an MILP model similar to the STN model, but incorporates a special MILP solver developed at Purdue University that exploits more effectively the structure of the scheduling models (e.g. see Bassett, Pekny, & Reklaitis, 1997). VirtECS Schedule includes an Interactive Scheduling Tool (IST) to facilitate the ability to modify production schedules through direct control of key inputs. VirtECS Schedule has also the capability of rescheduling to respond to changing operating conditions on the plant floor, for instance when mechanical failures or rush orders make the current schedule obsolete.

8.4. SAP advanced planner and optimizer (SAP APO)

SAP (http://www.sap.com) offers the system mySAP, a comprehensive framework for supply chain optimization. mySAP includes the module SAP Advanced Planner and Optimizer (SAP APO), in which the planning and scheduling tool helps to support real-time and network optimization across the extended supply chain (Braun & Kasper, 2004). This module is used within a hierarchical decomposition scheme for planning and scheduling. While the higher level planning tools are based on MILP models, it appears that the detailed scheduling module (PP/DS) is largely based on constraint programming and genetic algorithms. It also appears to be restricted to multistage plant configurations.

9. Current reactive scheduling capabilities

The scheduling techniques examined in the previous sections are aimed at generating a priori production schedules assuming that plant parameters and production requirements will remain unchanged throughout the entire time horizon. However, industrial environments are highly dynamic and although the proposed initial schedule may be the best option under the predicted circumstances, it can quickly become inefficient or even infeasible after the occurrence of unforeseen events, which are not only related to external market factors (late order arrivals, orders cancellations, delayed raw material shipments, modifications in order due dates and/or customer priorities) but also to the operational level (changes in batch processing/setup times, unit breakdown/startup, reprocessing of batches, changes in resource availabilities). In such a case, the ability to handle unpredictable circumstances and periodically or driven by events re-optimize the schedule on a daily or hourly basis becomes a key issue in batch plant operation.

Despite the great importance of rescheduling functionality for batch processes, only a few number of optimization approaches have been reported in the last decade. Hasebe, Hashimoto, and Ishikawa (1991) proposed a reordering algorithm for the scheduling of multiproduct batch plants consisting of parallel production lines with a shared unit. The algorithm involved two reordering operations, the insertion of a job and the exchange of two jobs. More recently, Vin and Ierapetritou (2000) developed a solution approach that addresses the problem of reactive scheduling in multiproduct batch plants. The approach was based on a two-stage solution procedure where the optimal reschedule is obtained from the solution of a MILP formulation that systematically incorporates all different rescheduling alternatives. Two kinds of disturbances involving machine breakdown and rush order arrivals were only considered. Roslöf et al. (2001) presented an MILP reordering algorithm to improve a non-optimal schedule or update the schedule in progress because of unforeseen events. Test runs were performed by releasing, i.e. re-allocating and/or re-sequencing, either one or two jobs at a time. Méndez and Cerdá (2003b) developed a MILP formulation for the reactive scheduling problem in multiproduct batch plants. The proposed approach allowed performing multiple rescheduling operations at the same time such as the insertion of new order arrivals, the reassignment of existing batches to alternative units due to equipment failures and the reordering and time-shifting of old batches at the current processing sequences. To prevent rescheduling actions from disrupting smooth plant operation, limited changes in batch sequencing and unit assignment were permitted. Subsequently, the original model was extended in Méndez and Cerdá (2004a) to consider resource-constrained multistage batch facilities, where manpower limitations aside from processing units must be taken into account in the rescheduling framework.

So far the reported work clearly reveals that current capabilities of optimization methods to reactive scheduling problems are still very restricted and mostly focused on sequential batch processes. More general, efficient and systematic rescheduling tools are required for recovering feasibility and/or efficiency with short reaction time and minimum additional cost. The main effort should be oriented towards avoiding a time-expensive full-scale rescheduling, allowing during the rescheduling process only limited changes to the scheduling decisions already made at the beginning of the time horizon. In addition to the data required for predictive scheduling models, a generic rescheduling tool also needs to provide an explicit representation of the current situation by incorporating the information related to: (a) the schedule in progress, (b) the present plant state, (c) current inventory levels, (d) present resource availabilities, (e) the current time data, (f) unexpected events, (g) rescheduling actions that can be taken and (h) the criterion to be optimized. Rescheduling actions may range from a simple time shifting to a full-scale re-optimization, depending on the type of events that occurred, the current situation of the plant and the available time to adjust the schedule. Given that more flexible rescheduling operations usually involve higher computational cost, the role of the human expert or scheduler should be oriented at the definition of the scope of the possible rescheduling actions. Therefore, a reactive scheduling framework for general batch processes should provide the basic rescheduling operations to optimally:

(1) Fix critical scheduling decisions already made at the beginning of the time horizon (lot-sizing, allocation, sequencing and timing).
(2) Modify or adjust some scheduling decisions (resource re-allocation, batch re-sequencing and time-shifting).
(3) Eliminate batches (order cancellation).
(4) Mix or split batches already scheduled.
(5) Modify size of batches already scheduled (re-sizing).
(6) Transform new demands into a set of new batches to be processed (lot-sizing).
(7) Insert new batches into the schedule in progress.

Furthermore, these rescheduling actions should be performed simultaneously and with a modest computational effort, aiming at satisfying all process constraints while optimizing a specific rescheduling goal. The estimated cost of updating the on-going schedule should be incorporated in the problem representation.

10. Beyond short-term scheduling

We would like to conclude this paper by noting that short-term scheduling is an important problem in its own right for which a number of challenges must still be overcome (e.g. effective handling of changeovers in discrete time models or efficient solutions for continuous time models as discussed in Section 4).

However, we should note that scheduling is also a basic building block in the more general area of Enterprise-wide Optimization (see Grossmann, 2005) where significant extensions that give rise to interesting research challenges are the following. First, simultaneous planning and scheduling is concerned with coupling the longer term planning decisions for establishing production levels with the shorter term scheduling decisions. The main research issue is how to guarantee consistency and optimality between the two levels (e.g. see Erdirik Dogan & Grossmann, 2006; Subrahmanyam, Bassett, Pekny, & Reklaitis, 1995; Wilkinson, Shah, & Pantelides, 1996; Zhu & Majozi, 2001). Second, simultaneous design and scheduling considers a further level of integration where decisions on the configuration and design of units must be coupled with scheduling decisions. The main research issue here is how to develop superstructures for batch plants that are rich enough in terms of alternatives and that in a realistic way be coupled to scheduling models (e.g. see Barbosa-Povoa and Macchietto, 1994; Lin and Floudas, 2001; Papageorgaki and Reklaitis, 1990). Third, at a lower level an important question is how to couple scheduling models with process models (see Section 5), and particularly with dynamic models that can rigorously predict the optimal control for the transitions. Here a major research issue is how to solve complex mixed-integer dynamic optimization problems (see Flores-Tlacuahuac & Grossmann, 2006; Mishra, Mayer, Raisch, & Kienle, 2005; Nyström, Franke, Harjunkoski, & Kroll, 2005). Fourth, another important aspect is to account for financial cash flow as part of the planning and scheduling problem (Badell and Puigjaner, 2001; Romero, Badell, Bagajewicz, & Puigjaner, 2003). Finally, another major issue is the handling of uncertainty both in terms of processing times and availability of equipment. A major challenge here is how to best formulate a stochastic optimization model that is meaningful and whose results are easy to interpret (e.g. see, Balasubramanian and Grossmann, 2002; Engell, Märkert, Sand, & Schultz, 2004; Jia and Ierapetritou, 2004).

11. Conclusions

This paper has presented a comprehensive review of the state-of-the art of batch scheduling. An extensive classification of problem types has shown the great diversity involved in short-term batch scheduling problems. A general classification of optimization models was used as framework for describing the major optimization approaches that have emerged over the last decade in this area. Modeling aspects of representative optimization models were presented emphasizing the main ideas and highlighting their strengths and limitations. Two benchmark problems were solved by using the different approaches to illustrate the performance of methods discussed in the review. Two real-world industrial problems were also discussed to highlight some of the limitations of current methods. Finally, other alternative solution methods were briefly discussed, followed by approaches for solving large-scale problems. From the academic and commercial software that was discussed it is clear that the general scheduling software that can address all cases is still elusive. The important issue of rescheduling capabilities was also briefly discussed showing that substantial work remains to be done in this area.

It is hoped that this paper will stimulate further research as it is clear that even though very significant progress has been made in short-term batch scheduling, the direct and systematic solution of large-scale industrial problems through mathematical programming is still an unresolved issue.

Acknowledgments

The authors gratefully acknowledge financial support from ABB Corporate Research. We would also like to thank Dr. Pousga Kaboré and the PhD students Anna Bonfill and Gonzalo Guillén from UPC for helping to produce the numerical result for the case studies.

References

Aarts, E. H. L., & Korst, J. (1989). *Simulated annealing and boltzmann machines: A stochastic approach to combinatorial optimization and neural computing*. New York: John Wiley.

Badell, M., & Puigjaner, L. A. (2001). Advanced enterprise resource management systems for the batch industry. The TicTacToe algorithm. *Computers and Chemical Engineering*, *25*, 517–538.

Balas, E., Ceria, S., & Cornuejols, G. (1993). A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming*, *58*, 295–324.

Balasubramanian, J., & Grossmann, I. E. (2002). A novel branch and bound algorithm for scheduling flowshop plants with uncertain processing times. *Computers and Chemical Engineering*, *26*, 41–57.

Baptiste, P., Le Pape, C., & Nuijten, W. (2001). *Constrained-based scheduling: Applying constraint programming to scheduling problems*. Kluwer Academic Publishers.

Barbosa-Povoa, A., & Macchietto, S. (1994). Detailed design of multipurpose batch plants. *Computers and Chemical Engineering*, *18*, 1013–1042.

Bassett, M. H., Pekny, J. F., & Reklaitis, G. V. (1997). Using detailed scheduling to obtain realistic operating policies for a batch processing facility. *Industrial and Engineering Chemistry Research*, *36*, 1717–1726.

Battiti, R., & Tecchiolli, G. (1994). The reactive tabu search. *ORSA Journal on Computing*, *6*, 126.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, *4*, 238–252.

Birewar, D., & Grossmann, I. E. (1990). Simultaneous planning and scheduling of multiproduct batch plants. *Industrial and Engineering Chemistry Research*, *29*, 570.

Bixby, R. E., Fenelon, M., Gu Z., Rothberg E., & Wunderling R. (2002). MIP: Theory and practice: Closing the gap. http://www.ilog.com/products/optimization/tech/research/mip.pdf.

Bixby, R. E. (2002). Solving real-world linear programs: A decade and more of progress. *Operations Research*, *50*, 3–15.

Blackstone, J. H., Phillips, D. T., & Hogg, G. L. (1982). A state-of-the-art-survey of dispatching rules for manufacturing job shop operations. *International Journal of Production Research*, *20*, 27–45.

Blömer, F., & Günther, H. (2000). LP-based heuristics for scheduling chemical batch processes. *International Journal of Production Research*, *38*, 1029–1051.

Braun, H., & Kasper, T. (2004). Optimization in SAP supply chain management. Presented at CPAIOR04, Nice. http://www-sop.inria.fr/coprin/cpaior04/files/CPAIOR2004SAP.PDF.

Burkard, R., & Hatzl, J. (2005). Review, extensions and computational comparison of MILP formulations for scheduling of batch processes. *Computers and Chemical Engineering*, *29*, 2823–2835.

Burkard, R., Hujter, M., Klinz, R., Rudolf, R., & Wennink, M. (1998). A process scheduling problem arising from chemical production planning. *Optimization Methods and Software*, *10*, 175–196.

Castro, P., Barbosa-Póvoa, A. P. F. D., & Matos, H. (2001). An improved RTN continuous-time formulation for the short-term scheduling of multipurpose batch plants. *Industrial and Engineering Chemistry Research*, *40*, 2059–2068.

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 and Engineering Chemistry Research*, *43*, 105–118.

Cavin, L., Fischer, U., Glover, F., & Hungerbühler, K. (2004). Multi-objective process design in multi-purpose batch plants using a tabu search optimization algorithm. *Computers and Chemical Engineering*, *28*, 459–478.

Cerdá, J., Henning, G. P., & Grossmann, I. E. (1997). A mixed-integer linear programming model for short-term scheduling of single-stage multiproduct batch plants with parallel lines. *Industrial and Engineering Chemistry Research*, *36*, 1695–1707.

Chen, C., Liu, C., Feng, X., & Shao, H. (2002). Optimal short-term scheduling of multiproduct single-stage batch plants with parallel lines. *Industrial and Engineering Chemistry Research*, *41*, 1249–1260.

Crowder, H. P., Johnson, E. L., & Padberg, M. W. (1983). Solving large-scale zero-one linear programming problems. *Operations Research*, *31*, 803–834.

Dakin, R. J. (1965). A tree search algorithm for mixed integer programming problems. *Computer Journal*, *8*, 250–255.

Dash Optimization (2003). http://www.dashoptimization.com/products_version2003.html.

Dincbas, M., Van Hentenryck, P., Simonis, H., Aggoun, A., Graf, T., & Berthier, F. (1988). The constraint logic programming language CHIP. In *FGCS-88: Proceedings of International Conference on Fifth Generation Computer Systems* (pp. 693–702).

Engell, S., Märkert, A., Sand, G., & Schultz, R. (2004). Aggregated scheduling of a multiproduct batch plant by two-stage stochastic integer programming. *Optimization and Engineering*, *5*, 335–359.

Erdirik Dogan, M., & Grossmann, I. E. (2006). Simultaneous Planning and Scheduling for Multiproduct Continuous Plants. *Industrial and Engineering Chemistry Research*, *45*, 299–315.

Flores-Tlacuahuac, A., & Grossmann, I. E. (2006). Simultaneous cyclic scheduling and control of a multiproduct CSTR reactor. *Industrial and Engineering Chemistry Research*, in press.

Floudas, C. A., & Lin, X. (2004). Continuous-time versus discrete-time approaches for scheduling of chemical processes: A review. *Computers and Chemical Engineering*, *28*, 2109–2129.

Giannelos, N. F., & Georgiadis, M. C. (2002). A simple new continuous-time formulation for short-term scheduling of multipurpose batch processes. *Industrial and Engineering Chemistry Research*, *41*, 2178–2184.

Glover, F. (1990). Tabu search: A tutorial. *Interfaces*, *20*, 74.

Goldberg, D. E. (1989). Genetic algorithms in search. In *Optimisation and machine learning*. Reading Mass: Addison-Wesley.

Gomory, R. E. (1958). Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematics Society*, *64*, 275–278.

Graells, M., Cantón, J., Peschaud, B. T., & Puigjaner, L. (1998). General approach and tool for the scheduling of complex production systems. *Computers and Chemical Engineering*, *22*(1), S395–S402.

Graves, S. C. (1982). Using lagrangean techniques to solve hierarchical production planning problems. *Management Science*, *28*, 260–275.

Grossmann, I. E. (2002). Review of non-linear mixed-integer and disjunctive programming techniques for process systems engineering. *Journal of Optimization and Engineering*, *3*, 227–252.

Grossmann, I. E. (2005). Enterprise-wide Optimization: A New Frontier in Process Systems Engineering. *AIChE Journal*, *51*, 1846–1857.

Gupta, S., & Karimi, I. A. (2003). An improved MILP formulation for scheduling multiproduct, multistage batch plants. *Industrial and Engineering Chemistry Research*, *42*, 2365–2380.

Gupta, A., & Maranas, C. D. (1999). A hierarchical lagrangean relaxation procedure for solving midterm planning problems. *Industrial and Engineering Chemistry Research*, *38*, 1937–1947.

Harjunkoski, I., & Grossmann, I. E. (2001). A decomposition approach for the scheduling of a steel plant production. *Computers and Chemical Engineering*, *55*(11), 1647–1660.

Harjunkoski, I., & Grossmann, I. E. (2002). Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. *Computers and Chemical Engineering*, *26*, 1533–1552.

Hasebe, S., Hashimoto, I., & Ishikawa, A. (1991). General reordering algorithm for scheduling of batch process. *Journal of Chemical Engineering of Japan*, *24*(4), 483–489.

Henning, G. P., & Cerdá, J. (2000). Knowledge-based predictive and reactive scheduling in industrial environments. *Computers and Chemical Engineering*, *24*(9), 2315–2338.

Hooker, J. N. (1999). *Logic-based methods for optimization*. John Wiley and Sons.

Ierapetritou, M. G., & Floudas, C. A. (1998). Effective continuous-time formulation for short-term scheduling. 1. Multipurpose batch processes. *Industrial and Engineering Chemistry Research*, *37*, 4341–4359.

ILOG (1999). ILOG OPL Studio 2.1., User's Manual. ILOG Inc.

Jackson, J., & Grossmann, I. E. (2003). A temporal decomposition scheme for non-linear multisite production planning and distribution models. *Industrial and Engineering Chemistry Research*, *42*, 3045–3055.

Jain, V., & Grossmann, I. E. (2001). Algorithms for hybrid MILP/CP model for a class of optimization problems. *INFORMS Journal in Computing*, *13*, 258–276.

Janak, S. L., Lin, X., & Floudas, C. A. (2004). Enhanced continuous-time unit-specific event-based formulation for short-term scheduling of multipurpose batch processes: Resource constraints and mixed storage policies. *Industrial and Engineering Chemistry Research*, *43*, 2516–2533.

Janak, S. L., Lin, X., & Floudas, C. A. (2005). Enhanced continuous-time unit-specific event-based formulation for short-term scheduling of multipurpose batch processes: Resource constraints and mixed storage policies. *Addition/Correction. Industrial and Engineering Chemistry Research*, *44*, 426.

Jia, Z., & Ierapetritou, M. G. (2004). Short-term scheduling under uncertainty using MILP sensitivity analysis. *Industrial and Engineering Chemistry Research*, *43*, 3782.

Johnson, E. L., Nemhauser, G. L., & Savelsbergh, M. W. P. (2000). Progress in linear programming based branch-and-bound algorithms: An exposition. *INFORMS Journal on Computing*, 12.

Jones, C. V., & Baker, T. E. (1996). MIMI/G: A graphical environment for mathematical programming and modeling. *Interfaces*, *26*(3), 90ff.

Kallrath, J. (2000). Mixed integer optimization in the chemical process industry: Experience, potential and future. *Transactions I. Chemical E.*, *78*(Part A), 809–822.

Kallrath, J. (2002). Planning and scheduling in the process industry. *OR Spectrum*, *24*, 219–250.

Kirkpatrick, S., Gelatt, C. D., & Vechi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*, 671.

Kondili, E., Pantelides, C. C., & Sargent, W. H. (1993). A general algorithm for short-term scheduling of batch operations-I. MILP formulation. *Computers and Chemical Engineering*, *2*, 211–227.

Le Pape, C. (1998). Implementation of resource constraints in ILOG schedule: A library for the development of constrained-based scheduling systems. *Intelligent Systems Engineering*, *3*(2), 55–66.

Lee, K., Park, H. I1, & Lee, I. (2001). A novel non-uniform discrete time formulation for short-term scheduling of batch and continuous processes. *Industrial and Engineering Chemistry Research*, *40*, 4902–4911.

Lee, Y. G., & Malone, M. F. (2000). Flexible batch processing. *Industrial and Engineering Chemistry Research*, *39*(6), 2045–2055.

Lim, M., & Karimi, I. A. (2003). Resource-constrained scheduling of parallel production lines using asynchronous slots. *Industrial and Engineering Chemistry Research*, *42*, 6832–6842.

Lin, X., & Floudas, C. F. (2001). Design, synthesis and scheduling of multipurpose batch plants via an effective continuous-time formulation. *Computers and Chemical Engineering*, *25*, 665–674.

Lin, X., Floudas, C. A., Modi, S., & Juhasz, N. M. (2002). Continuous-time optimization approach for medium-range production scheduling of a multiproduct batch plant. *Industrial and Engineering Chemistry Research*, *41*, 3884–3906.

Löhl, T., Schulz, C., & Engell, S. (1988). Sequencing of batch operations for a highly coupled production process: Genetic algorithms versus mathematical programming. *Computers and Chemical Engineering*, *22*(1), S579–S585.

Maravelias, C. T., & Grossmann, I. E. (2003). New general continuous-time state-task network formulation for short-term scheduling of multipurpose batch plants. *Industrial and Engineering Chemistry Research*, *42*, 3056–3074.

Maravelias, C. T., & Grossmann, I. E. (2004). A hybrid MILP/CP decomposition approach for the continuous time scheduling of multipurpose batch plants. *Computers and Chemical Engineering*, *28*, 1921–1949.

Méndez, C. A., Henning, G. P., & Cerdá, J. (2000). Optimal scheduling of batch plants satisfying multiple product orders with different due-dates. *Computers and Chemical Engineering*, *24*, 2223–2245.

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*, *12*, 721–726, Elsevier Science Ltd., ISBN 0-444-51109-1.

Méndez, C. A., & Cerdá, J. (2003a). An MILP continuous-time framework for short-term scheduling of multipurpose batch processes under different operation strategies. *Optimization and Engineering*, *4*, 7–22.

Méndez, C. A., & Cerdá, J. (2003b). Dynamic scheduling in multiproduct batch plants. *Computers and Chemical Engineering*, *27*, 1247–1259.

Méndez, C. A., & Cerdá, J. (2004a). An MILP framework for batch reactive scheduling with limited discrete resources. *Computers and Chemical Engineering*, *28*, 1059–1068.

Méndez, C. A., & Cerdá, J. (2004b). Short-term scheduling of multistage batch processes subject to limited finite resources. *Computer-aided Chemical Engineering*, *15B*, 984–989.

Mishra, B. V., Mayer, E., Raisch, J., & Kienle, A. (2005). Short-term scheduling of batch processes. A comparative study of different approaches. *Industrial and Engineering Chemistry Research*, *44*, pp. 4022, 4034.

Mockus, L., & Reklaitis, G. V. (1999a). Continuous time representation approach to batch and continuous process scheduling. 1. MINLP formulation. *Industrial and Engineering Chemistry Research*, *38*, 197–203.

Mockus, L., & Reklaitis, G. V. (1999b). Continuous time representation approach to batch and continuous process scheduling. 2. Computational issues. *Industrial and Engineering Chemistry Research*, *38*, 204–210.

Nuijten, W. P. M. (1994). Time and resource constrained scheduling: A constraint satisfaction approach. *PhD Thesis*, Eindhoven University of Technology.

Nyström, R. H., Franke, R., Harjunkoski, I., & Kroll, A. (2005). Production campaign planning including grade transition sequencing and dynamic optimization. *Computers and Chemical Engineering*, *29*, 2163–2179.

Pacciarelli, D., & Pranzo, M. (2004). Production scheduling in a steelmaking-continuous casting plant. *Computers and Chemical Engineering*, *28*, 2823–2835.

Pantelides, C. C. (1994). Unified frameworks for optimal process planning and scheduling. In *Foundations of computer-aided process operations*. New York: Cache publications, pp. 253–274.

Panwalkar, S. S., & Iskander, W. A. (1977). Survey of scheduling rules. *Operations Research*, *25*, 45–61.

Papageorgaki, S., & Reklaitis, G. (1990). Optimal design of multipurpose batch processes. 1: Problem formulation. *Industrial Engineering and Chemistry Research*, *29*, 2054–2062.

Pekny, J. F., & Reklaitis, G. V. (1998). Towards the convergence of theory and practice: A technology guide for scheduling/planning methodology. In *Proceedings of the third international conference on foundations of computer-aided process operations* (pp. 91–111).

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 and Engineering Chemistry Research*, *35*, 338–342.

Pinto, J. M., & Grossmann, I. E. (1997). A logic-based approach to scheduling problems with resource constraints. *Computers and Chemical Engineering*, *21*, 801–818.

Pinto, J. M., & Grossmann, I. E. (1998). Assignments and sequencing models of the scheduling of process systems. *Annals of Operations Research*, *81*, 433–466.

Rabelo, R. J., & Camarinha-Matos, L. M. (1994). Negotiation in multi-agent based dynamic scheduling. *Journal on Robotics and Computer Integrated Manufacturing*, *11*(4), 303–310.

Reklaitis, G. V. (1992). Overview of scheduling and planning of batch process operations. In *NATO advanced study institute—batch process systems engineering*. Turkey: Antalya.

Rodrigues, M. T. M., Latre, L. G., & Rodrigues, L. C. A. (2000). Short-term planning and scheduling in multipurpose batch chemical plants: A multi-level approach. *Computers and Chemical Engineering*, *24*, 2247–2258.

Romero, J., Badell, M., Bagajewicz, M., & Puigjaner, L. (2003). Integrating budgeting models into scheduling and planning models for chemical batch industry. *Industrial and Engineering Chemistry Research*, *42*, 6125–6134.

Roslöf, J., Harjunkoski, I., Björkqvist, J., Karlsson, S., & Westerlund, T. (2001). An MILP-based reordering algorithm for complex industrial scheduling and rescheduling. *Computers and Chemical Engineering*, *25*, 821–828.

Ryu, J. H., Lee, H. K., & Lee, I. B. (2001). Optimal scheduling for a multiproduct batch process with minimization of penalty on due date period. *Industrial and Engineering Chemistry Research*, *40*(1), 228–233.

Sauer, J., & Bruns, R. (1997). Knowledge-based scheduling in industry and medicine. *IEEE Expert*, *12*, 24–31.

Schilling, G., & Pantelides, C. C. (1996). A simple continuous-time process scheduling formulation and a novel solution algorithm. *Computers and Chemical Engineering*, *20*, S1221–S1226.

Schulz, C., Engell, S., & Rudolf, R. (1998). Scheduling of a multiproduct polymer batch plant. In J. F. Pekny, G. E. Blau, & B. Carnahan (Eds.), *Proceeding for foundations of computer-aided process operations (FOCAPO' 98)*. Snowbird: CACHE Publications.

Shah, N., Pantelides, C. C., & Sargent, W. H. (1993). A general algorithm for short-term scheduling of batch operations-II. Computational issues. *Computers and Chemical Engineering*, *2*, 229–244.

Shah, N. (1998). Single and multisite planning and scheduling: Current status and future challenges. In *Proceedings of the third international conference on foundations of computer-aided process operations* (pp. 75–90).

Shen, W., & Norrie, D. H. (1999). Agent-based systems for intelligent manufacturing: A state of the art survey. *Knowledge and Information Systems*, *1*(2), 129–156.

Subrahmanyam, S., Bassett, M. H., Pekny, J. F., & Reklaitis, G. V. (1995). Issues in solving large scale planning, design and scheduling problems in batch chemical plants. *Computers and Chemical Engineering*, *19*, 577–582.

Sundaramoorthy, A., & Karimi, I. A. (2005). A simpler better slot-based continuous-time formulation for short-term scheduling in multiproduct batch plants. *Chemical and Engineering Science*, *60*, 2679–2702.

Tang, L., Liu, J., Rong, A., & Yang, Z. (2000). A mathematical programming model for scheduling steelmaking continuous casting production. *European Journal of Operational Research*, *120*(2), 423–435.

Van Hentenryck, P. (1989). *Constraint satisfaction in logic programming*. Cambridge, MA: MIT PRESS.

Van Hentenryck, P. (2002). Constraint and integer programming in OPL. *INFORMS Journal on Computing*, *14*(4), 345–372.

Van Roy, T. J., & Wolsey, L. A. (1986). Valid inequalities for mixed 0-1 programs. *Discrete Applied Mathematics*, *14*, 199–213.

Vin, J. P., & Ierapetritou, M. G. (2000). A new approach for efficient rescheduling of multiproduct batch plants. *Industrial and Engineering Chemistry Research*, *39*, 4228–4238.

Wallace, M., Novello, S., & Schimpf, J. (1997). ECLiPSe: A platform for constraint logic programming. *ICL Systems Journal*, *12*(1), 159–200.

Wang, K., Löhl, T., Stobbe, M., & Engell, S. (2000). A genetic algorithm for online-scheduling of a multiproduct polymer batch plant. *Computers and Chemical Engineering*, *24*, 393–400.

Westenberger, H. & Kallrath, J. (1995). Formulation of a job shop problem in process industry. *Internal report*, Bayer AG, Leverkusen, and BASF AG, Ludwigshafen.

Wilkinson, S. J. (1996) Aggregate formulations for large-scale process scheduling problems, *Thesis dissertation*, Department of Chemical Engineering and Chemical Technology, Imperial College, London.

Wilkinson, S. J., Shah, N., & Pantelides, C. C. (1996). Aggregate modeling of multipurpose plant operation. *Computers and Chemical Engineering*, *19*(Supplement 1, 11), 583–588.

Wolsey, L. (1998). *Integer programming*. John Wiley and Sons.

Zhang, X., & Sargent, W. H. (1996). The optimal operation of mixed production facilities—a general formulation and some approaches for the solution. *Computers and Chemical Engineering*, *20*, 897–904.

Zhu, X. X., & Majozi, T. (2001). Novel continuous time MILP formulation for multipurpose batch plants. 2. Integrated planning and scheduling. *Industrial and Engineering Chemistry Research*, *40*, 5621–5634.

Zweben, M., & Fox, M. S. (1994). *Intelligent scheduling*. San Francisco: Morgan Kaufmann.