Computers and Chemical Engineering 92 (2016) 101–117

Contents lists available at ScienceDirect

Computers and Chemical Engineering

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

A novel constraint programming model for large-scale scheduling problems in multiproduct multistage batch plants: Limited resources and campaign-based operation

Franco M. Novara[a], Juan M. Novas[b], Gabriela P. Henning[c,*]

[a] FIQ, Universidad Nacional del Litoral, Santiago del Estero 2829, Santa Fe 3000, Argentina
[b] CIEM (Universidad Nacional de Córdoba, CONICET), Medina Allende s/n, Córdoba 5000, Argentina
[c] INTEC (Universidad Nacional del Litoral, CONICET),Güemes 3450, Santa Fe 3000, Argentina

ARTICLE INFO

Article history:
Received 30 December 2015
Received in revised form 19 April 2016
Accepted 19 April 2016
Available online 20 May 2016

Keywords:
Scheduling
Constraint programming
Multiproduct multistage batch plants
Campaign operating mode

ABSTRACT

This contribution introduces an efficient constraint programming (CP) model that copes with large-scale scheduling problems in multiproduct multistage batch plants. It addresses several features found in industrial environments, such as topology constraints, forbidden product-equipment assignments, sequence-dependent changeover tasks, dissimilar parallel units at each stage, limiting renewable resources and multiple-batch orders, among other relevant plant characteristics. Moreover, the contribution deals with various inter-stage storage and operational policies. In addition, multiple-batch orders can be handled by defining a campaign operating mode, and lower and upper bounds on the number of batches per campaign can be fixed. The proposed model has been extensively tested by means of several case studies having various problem sizes and characteristics. The results have shown that the model can efficiently solve medium and large-scale problems with multiple constraining features. The approach has also rendered good quality solutions for problems that consider multiple-batch orders under a campaign-based operational policy.

1. Introduction

Multiproduct multistage batch plants are characterized by the manufacturing of multiple products having similar recipes. These plants generally operate on the basis of a set of orders, each having its due date. In these environments, the short-term scheduling has an important impact on the costs and benefits of the organization, as well as on the effective use of the limited resources. Therefore, it is crucial to develop efficient scheduling approaches capable of solving big size problems and finding good quality solutions in reduced computational times.

In multiproduct multistage environments, each order is fulfilled by means of a single or multiple batches. Most of the academic contributions, such as (Harjunkoski and Grossmann, 2002; Gupta and Karimi, 2003; Marchetti and Cerdá, 2009a,b); among others, have assumed that a single batch per product is demanded. When multiple batches per order are required, it is usual to operate under a campaign mode. Thus, a given number of batches of the same product are sequentially processed in order to reduce changeover times and costs. The number and size of the batches required to satisfy a given order can be specified before the scheduling activity takes place by decoupling the batching and the scheduling decisions (sequential approach). On the contrary, the scheduling problem can include the definition of the number and size of batches (monolithic approach). However, the number of batches that comprise each product campaign is a decision that must be taken in any case.

This contribution introduces a novel constraint programming (CP) approach to address the short-term scheduling of multiproduct multistage batch plants which operate under a campaign mode. The approach focuses on unit assignment, batch sequencing and timing decisions, while optimizing a time-based or cost-based objective function. The proposal assumes that the batching decisions have been taken beforehand. It accounts for several features that are present in industrial environments, such as dissimilar parallel units at each stage, topology constraints, forbidden product-equipment assignments, sequence-dependent changeover tasks, intermediate due-dates,

* Corresponding author.
E-mail address: ghenning@intec.unl.edu.ar (G.P. Henning).

http://dx.doi.org/10.1016/j.compchemeng.2016.04.030
0098-1354/© 2016 Elsevier Ltd. All rights reserved.

Nomenclature

Sets/indices

B/b	batches to be produced
Bp	batches of product p
C_p/c	set of all possible campaigns of product p. Its cardinality results from the expression $Card(B_p)/lB_p$
F_u	units belonging to stage $s+1$, which are unconnected to unit u, belonging to stage s
Fp	set of product couples that correspond to forbidden production sequences $f = <p,p'>$
$nbStages$	number of processing stages belonging to the production process
$P/p, p'$	demanded products
R/r	renewable resources (e.g. utilities, manpower, etc.)
S/s	processing stages
U/u	equipment units
Us	units belonging to stage s

Parameters

$avail_r$	maximum availability of resource r
$avail_{CIP}$	maximum number of available cleaning-in-place devices
$changeOverTime$	changeover time associated with the triplet $<p,p',u>$: the sequence dependent setup time between products p and p' on unit u
$cost_{p,u}$	processing cost of a batch of product p on unit u
dd_b	due-date of batch b
$fCost_u$	processing cost associated with the usage of unit u during the scheduling period
lB_p	minimum number of batches that a campaign of product p may comprise
$pt_{p,u}$	processing time required by a batch of product p on unit u
rd_u	ready time of unit u
$requir_{p,s,r}$	requirement of resource r in order to process a batch of product p at stage s
$cleanRequir_{b1,b2,u,r}$	requirement of resource r in order to perform the necessary cleaning between batches $b1$ and $b2$ in unit u
rt_b	release time of batch b
st_u	setup time of unit u
uB_p	maximum number of batches that a campaign of product p may comprise
$wt_{p,s}$	maximum waiting time for a batch of product p at stage s

Cumulative function

$UsageProfile_r$	accumulative usage of resource r as a function of time. It is employed to model the limited availability of resource r

Variables

$campaignTask_{c,p,u}$	campaign interval variable that spans over all the processing tasks that belong to a campaign c of product p carried out on unit u
$cleanTask_{b1,b2,u}$	optional interval variable that represents a cleaning activity between the processing tasks corresponding to batches $b1$ and $b2$ in unit u. When the campaign mode is considered, indexes $b1$ and $b2$ must be replaced by $c1$ and $c2$
$nTar_b$	binary variable that is equal to 1 when batch b finishes after its due-date, and 0 otherwise
$stageTask_{b,s}$	interval variable representing a processing task of batch b at stage s
tar_b	batch b tardiness
$task_{b,u,c}$	optional interval variable that represents a processing task of batch b in unit u, in the context of campaign c on unit u. If campaign mode is not considered, index c is ignored
$unitBatchSeq_u$	sequence variable defined for each unit u. It represents an ordering of task interval variables associated with u. Each interval variable (task) in this sequence is characterized by an attribute. When the changeovers between tasks are unit-independent, this attribute specifies the product p associated with each task variable. In case the changeovers are unit-dependent, the attribute represents both, the product and the unit where the task is assigned.
$unitCampaignSeq_u$	sequence variable defined for each unit u. It represents an ordering of campaign interval variables associated with u. Each interval variable (campaign) in this sequence is characterized by an attribute that specifies the product p associated with the campaign variable.
z_u	binary variable that is equal to 1 when unit u is used during the scheduling period, and 0 otherwise

different types of limiting renewable resources (e.g. utilities, manpower, cleaning-in-place devices, etc.), demandaded by both processing and cleaning activities, multiple-batch orders (addressed in a campaign mode with consideration of a lower and upper bound on the number of batches per campaign). In addition, the approach is able to solve large-scale problems like the ones found in actual settings. The CP model is based on the ILOG-IBM OPL language and the CP Optimizer, which are embedded within the CPLEX Optimization Studio (IBM ILOG, 2013). It has been tested by means of a variety of problem instances having various sizes and characteristics.

The rest of the work is organized as follows: Section 2 presents the state-of-the-art. Section 3 describes the problem under study and Section 4 introduces the CP formulation. Finally, Section 5 discusses the computational results, and Section 6 presents the conclusions of this work.

2. State-of-the-art

This contribution addresses the predictive scheduling problem of resource-constrained multiproduct multistage batch plants, which has received a lot of attention in the past 15 years. Different methods have been proposed to deal with this problem, prevailing various types of mathematical programming-based approaches: (i) direct precedence-based (Méndez et al., 2001; Gupta and Karimi, 2003) and generalized precedence-based MILP formulations (Méndez and Cerdá, 2003a,b; Marchetti and Cerdá, 2009a), (ii) slot-based (Pinto and Grossmann, 1995; Liu and Karimi, 2007a), and slot and sequence-based MILP approaches (Liu and Karimi, 2007b, 2008), (iii) continuous and discrete time-grid representations (Castro et al., 2006; Castro and Novais, 2008; Velez and Maravelias, 2013). Constraint programming combined with MILP formulations have been successfully applied (Harjunkoski and Grossmann, 2002; Maravelias and Grossmann, 2004). In addition, some pure CP-based approaches, such as the ones proposed by Zeballos et al. (2011) and Novara et al., (2013) have been recently reported. Other techniques, such as timed automata (Schoppmeyer et al., 2012), dispatching rules (Blömer and Günther, 2000), and artificial intelligence-based approaches (Yang et al., 2013) have also been employed to tackle this problem.

Despite the great number of contributions, there is still an important gap between the actual features of industrial problems that need to be represented and the ones that can be addressed by the academic proposals (Méndez et al., 2006; Henning, 2009; Harjunkoski et al., 2014). In order to reduce this gap, there are efforts that nowadays intend to capture more realistic aspects of the scheduling problem and that try to solve bigger size case-studies. Some of the challenges currently being addressed are related to: (i) limiting features found in practice, (ii) large number of product orders, (iii) the simultaneous solution of batching and scheduling problems, (iv) the integration of planning and scheduling, (v) the integration of scheduling and control activities. In order to have an overview of these advances, Table 1 summarizes some of the main features that are addressed by some recently published contributions, including this work. Among the aspects taken into account are the monolithic/decoupled solution of the batching and scheduling problems, single/multiple order/s per product, single/multiple batch/es per order, the possibility of operating in a campaign mode, as well as the problem size in terms of the number of batches being considered.

Marchetti and Cerdá (2009a) (referred as MCa) introduced an MILP approach based on the general precedence notion. The formulation includes a reduced number of binary variables and allows reaching good quality solutions in reduced computational times. With the aim of getting simpler models, Marchetti and Cerdá (2009b) (referred as MCb) proposed another approach that exploits a common feature of many multistage batch plants, which is the bottleneck stage. In order to reduce the model dimensionality and the computational requirements their proposal relies on the so-called Constant Batch Ordering Rule, by means of which a single sequencing variable is enough to establish the relative ordering of two batches at every stage in which both have been allocated to the same equipment unit. The formulation obtained optimal or near-optimal solutions for several examples without considering resource constraints, in reduced computational times. For problems with some limitations on manpower or steam, the results were favorably compared with those presented by Méndez and Cerdá (2003a). Both proposals, MCa and MCb, addressed medium size case-studies, in which batches were known beforehand. They considered a single batch per order and a single order per product.

Another work that allowed addressing medium size problems, considering a single batch per order and a single order per product, is the one of Zeballos et al. (2011) (ZNH). They proposed a CP-based approach that captures many features found in industrial settings. It comprises a model and a search strategy, which takes advantage of domain knowledge. Two search procedures were presented; both of them aiming at balancing the unit load at each stage. The first strategy orders the stages by their increasing number, and the second one, orders the critical stage first. It is shown that the performance of the CP approach is sensitive to the search strategy.

Recent contributions, like the ones of Castro et al. (2009, 2011) and Kopanos et al. (2010), were meant to address bigger size problems. Castro et al. (2009) (CHGa) presented an MILP-based decomposition approach that schedules a large number of orders in a sequential way, considering a few of them at a time, in order to keep complexity at a manageable level. The first constructive step consists of an iterative procedure in which orders keep their unit assignments, but are allowed changing their start time and/or their relative position in the sequence. Once a complete and feasible schedule is obtained for the whole set of orders, local improvements can be made in the second step. Both, the number of orders that are considered in the constructive step and the preordering heuristic, are control parameters that affect the algorithm's performance. To test the proposal, a real pharmaceutical plant scheduling problem was solved; the algorithm found good quality solutions en low CPU times. Castro et al. (2011) (CHGb) extended the decomposition algorithm presented in their previous work in order to consider sequence- and unit-dependent changeover times. The proposed approach was again tested by means of the pharmaceutical plant benchmark. The obtained solutions, found in low computational times, are of better quality than the ones found by Castro et al. (2009) and Kopanos et al. (2010).

Kopanos et al. (2010) (KMP) proposed an iterative hybrid approach based on an MILP model. The solution strategy consists of two major procedural steps: i) a constructive step, wherein a feasible schedule is obtained in low CPU time by an iterative insertion procedure, and ii) an improvement step, wherein the previous agenda is gradually improved by iteratively implementing various rescheduling techniques. In order to validate the proposal, a real pharmaceutical plant scheduling problem, already introduced by Castro et al. (2009), was addressed. Several large-scale problem instances were solved and the hybrid strategy found, on average, good quality solutions in reasonable computational times.

Baumann and Trautmann (2013) (BT) presented a rigorous comparison of network and precedence-based scheduling models available in the literature and developed a continuous-time precedence-based MILP formulation that copes with make-and-pack production processes. The MILP proposal addresses some relevant features, such as sequence-dependent changeover times, multipurpose storage vessels with limited capacity, batch splitting, and transfer times, among others. An illustrative example taken from the literature shows how this novel formulation accounts for transfer times between units belonging to the pre- and final-mixing stages, pump-out times between the final mixing stage and the storage units, as well as quarantine time of batches when it is required.

As it was pointed out, the previous contributions assume a single batch per order. However, in practice, it is usual to manage orders comprising multiple batches. This problem has been addressed by means of simultaneous batching and scheduling approaches that were developed in recent years (Prasad and Maravelias, 2008; Sundaramoorthy and Maravelias, 2008a; Sundaramoorthy et al., 2009; Marchetti et al., 2012).

Prasad and Maravelias (2008) (PM), proposed a novel MILP formulation that was the first to simultaneously tackle the selection and sizing of batches, as well as their assignment and sequencing. Previous to this contribution, the batching and scheduling problems were

Table 1
Main problem features addressed in recent contributions and in this work.

Features			MCa	MCb	ZNH	CHGa	CHGb	KMP	BT	PM	SMa	SMb	SMP	MMC	This work
Inter-stage storage & operational policy	UIS	UW	●	●	●	●	●	●		●	●	●	●	●	●
		FW										●			●
	NIS	ZW			●							●			●
		UW			●							●			●
		FW										●			●
	FIS	UW							●			●	●		
		FW										●			
Set up features	Sequence independent		●	●	●	●		●				●			
	Sequence dependent	Unit dependent		●			●	●	●	●	●	●		●	●
		Unit independent	●		●										
Resource availability	Unlimited				●	●	●	●	●	●	●	●		●	
	Limited		●	●									●		●
Additional constraints	Production sequence				●					●	●	●			●
	Topology constraints		●	●	●	●			●	●	●	●	●	●	●
	Unit assignments		●	●	●	●	●	●	●	●	●	●	●	●	●
	Transfer times							●	●						
Processing times	Fixed		●	●	●	●	●	●		●					●
	Variable								●		●	●	●	●	
Objective function	Makespan		●	●	●	●	●	●	●	●	●	●	●	●	●
	Tardiness		●	●	●	●		●		●	●	●	●	●	●
	Earliness				●		●	●		●	●	●	●		●
	Cost							●		●	●	●	●		●
Problem size	Medium		●	●					●	●	●	●	●	●	
	Large				●	●	●	●						●	●
Campaign mode	Not considered		●	●	●	●	●	●	●	●	●	●	●		
	Considered													●	●
Decoupled Batching & Scheduling/Single order per product/ Single batch per order			●	●	●	●	●	●	●						
Simultaneous Batching & Scheduling/Single order per product/ Multiple batches per order										●	●	●	●		
Decoupled Batching & Scheduling/Multiple orders per product/ Multiple batches per order															●
Simultaneous Batching & Scheduling/Multiple orders per product/ Multiple batches per order														●	

independently solved. It is assumed that various product orders need to be scheduled and each one may be associated with more than one batch. However, it is supposed that there is just one order per product. The proposal considers batches to be independent of one another and does not take into account a campaign operation mode. Processing times depend on the product and the unit. An example consisting of 2 stages, 6 units, and 8 orders was addressed. Several small-size problem instances, in which the average number of batches was 3 per order, have been solved. It was shown that the inclusion of batch selection and sizing decisions within the scheduling model led to better solutions than the ones obtained with the two-step decoupling approach (i.e. first, batching, then scheduling) for time- and cost-related objective functions.

Sundaramoorthy and Maravelias (2008a) (SMa) extended the approach of Prasad and Maravelias (2008) to account for variable processing times. Several examples that allow showing the advantages and appropriate performance of the proposal were presented. These contributions assumed unlimited storage. In order to tackle storage limitations, Sundaramoorthy and Maravelias (2008b) (SMb) developed a method that solves the batching and scheduling problem of multistage batch plants, while taking into account storage constraints. This work also presented a general classification of storage policies based on capacity and timing constraints. Later, Sundaramoorthy et al. (2009) (SMP) proposed a discrete-time MILP formulation that simultaneously addresses the batching and scheduling problems in multistage processes having storage and utility constraints. The method tackles batching decisions without resorting to explicit batch-selection variables. Nevertheless, the problem is simplified by assuming that the batch size is a continuous variable, something that rarely occurs in the industrial practice.

Previous contributions do not consider a campaign operation mode. In such a mode, which is pretty common in industry, batches associated with the same product are consecutively manufactured in order to reduce changeover times. Marchetti et al. (2012) (MMC) tackled medium-size and large-scale problems including the following features: multiple batches per order, multiple customer orders per product having different due-dates, as well as a campaign mode of plant operation. Two MILP formulations addressing the combined batching and scheduling problems were proposed. The first one performs selection, sizing, allocation, sequencing, and timing decisions for the individual batches. Despite its expressiveness, the computational performance of this detailed model deteriorates for medium and large-scale problems. The second formulation is an MILP cluster-based approach, aimed at addressing industrial-sized scheduling problems. The model simultaneously selects the campaigns (clusters), the number and size of their corresponding batches, and then schedules the campaigns. Large instances (around 90 batches) can be solved to optimality, when minimizing tardiness or makespan. The number of campaigns per order is defined by means of a parameter that considers the average number of batches per campaign. For higher values of it, the number of campaigns, the model size, and the solution time all decrease. However, the chances of reaching optimal solutions also diminish.

This work presents a CP formulation aimed at addressing several features found in real industrial problems. As it is summarized in Table 1, the proposed approach can tackle problems that include various storage and inter-stage operational policies, sequence/unit dependent changeovers, different limiting resources, forbidden paths and batch-unit assignments, topology constraints, as well as time-based and cost-based objective functions. The proposed CP model can handle multiple-batches per order and a campaign mode of operation. In addition, large-size problems, can be efficiently tackled.

3. Problem statement

A set of customer orders, which demands different products to be manufactured at the shop-floor, is known at the beginning of the scheduling horizon. An order is associated with a single product and can lead to multiple batches. In addition, for a given product there might be multiple orders in a given period. It is assumed that the batching problem is solved when the scheduling horizon begins, and a set of batches per order per product has been obtained then. The various batches required by a given order have the same due-date and release-time.

At each plant stage, there is a set of non-identical processing units operating in parallel. Each unit belongs to a single stage. Depending on the plant topology, a unit belonging to a given stage, may or may not be connected to all the units pertaining to the next one (i.e. topology constraints). Units may have different ready-times, and it is assumed that they remain available during the whole scheduling period. The processing time required by each batch at a given unit is considered to be deterministic and known beforehand. Every product change on units requires a cleaning task, thus leading to a changeover time – of sequence-dependent/independent type – that needs to be taken into account.

In addition, other features that need to be considered are: (i) forbidden product-unit assignments, (ii) prohibited production sequences, and (iii) limitations on certain discrete resources, such as manpower and utilities. Different intermediate storage and inter-stage waiting policies can be represented, such as unlimited intermediate storage (UIS), non-intermediate storage (NIS) with unlimited wait (NIS-UW), zero wait (NIS-ZW), and finite wait (NIS-FW).

Since different objectives may be pursued in industrial plants at different moments, the CP model presented in this contribution is able to handle several performance measures. A cost-based objective function and several efficiency time-based ones, such as makespan, total tardiness, number of tardy batches, are addressed.

The plant can operate under a campaign mode. This approach is adopted for several reasons, such as the reduction of changeover times and scrap material. A campaign c is defined as a collection of batches of the same product p that are produced in the same unit in a sequential way – one after the other – in order to eliminate or minimize changeover times, as well as to reduce the amount of scrap material. A campaign comprises a limited number of batches of p so as to avoid acting as a bottleneck. The minimum and maximum number of batches that a campaign may comprise, lB_p and uB_p, are parameters that the plant scheduler can set. The final number of batches of each campaign will be an outcome of the scheduling model, that will result from the trade-off of these two opposite effects: (i) a campaign as an efficient way to minimize changeover time as well as scrap material, (ii) a campaign as a bottleneck that postpones the manufacturing of other products.

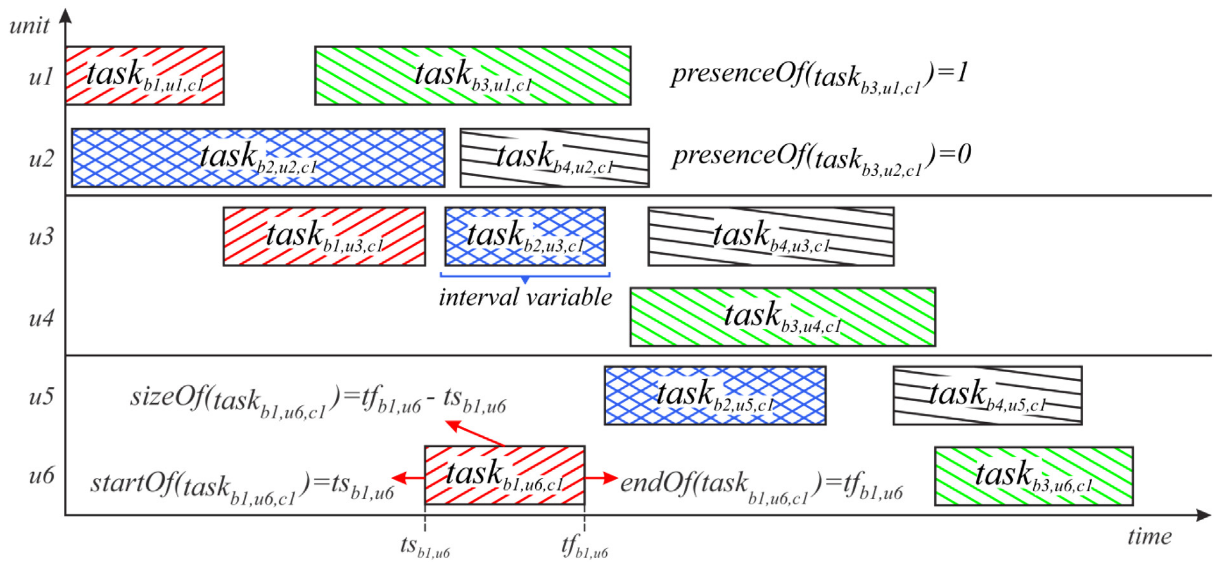

Fig. 1. Some functions associated with the presence of optional interval variables $task_{b,u,c}$.

4. CP formulation

In this section, the constraint programming formulation to tackle the scheduling problem specified in the previous section is presented. A brief description of the main CP constructs employed to address detailed scheduling activities is first presented. Then, the constraints and objective functions are described.

4.1. CP model characteristics

Constraint Programming (CP) techniques (Baptiste et al., 2005) have been successfully applied to scheduling problems by the Process Systems Engineering (PSE) community. The proposed model has been developed using the OPL programming language, supported by the IBM ILOG CPLEX Optimization Studio environment (IBM ILOG, 2013). The IBM ILOG OPL language, combined with the IBM ILOG CP Optimizer constraint programming engine, provides some specific scheduling constraints, functions, as well as different types of variables, aimed at describing scheduling problems in detail. Some of these OPL keywords are briefly introduced in this section (IBM ILOG, 2013).

- In OPL, the various tasks and activities are captured by the notion of *interval decision variables*. An interval variable represents a time interval during which an activity takes place and whose position in time is unknown. They are characterized by attributes such as start, end, and size. Several functions can be applied to interval variables, such as *startOf*, *endOf*, *sizeOf*, used to access the values of these attributes. In addition, interval variables can be declared as optional; i.e. the interval may or may not be present in the solution. For instance, whereas $stageTask_{b,s}$ cannot be an optional variable because it has to be present in the solution (since every batch needs to go through each stage), $task_{b,u,c}$ needs to be declared as optional because the execution of batch b can only take place in only one of the units u associated with stage s. The function *presenceOf* returns 1 if the optional interval exists in the solution, and 0 otherwise. Fig. 1 clarifies these concepts by showing some of the functions related to a set of optional interval variables. It can be seen, for instance, that $task_{b3,u1,c1}$ is part of the solution; thus, $presenceOf(task_{b3,u1,c1})=1$ and $presenceOf(task_{b3,u2,c1})=0$. This means that batch *b3* has been assigned to unit *u1* in the first stage. In addition, Fig. 1 shows that time points $ts_{b1,u6}$ and $tf_{b1,u6}$ define the start and completion times of $task_{b1,u6,c1}$, as well as its size.
- A *sequence decision variable* represents an ordering over a set of interval variables that belong to the solution of a problem. Each interval variable in a sequence is associated with an attribute that reflects one of its properties or a combination of properties that can be employed for sequencing decisions. Sequence decision variables are useful to represent the sequence of tasks/campaigns that are related to units. To avoid overlappings, different constraints can be applied to the interval variables belonging to the sequence. For instance, the *noOverlap* OPL construct is employed to constrain the intervals in a sequence such that their start and end times correspond to the relative ordering in the sequence. Thus, the activities do not overlap among themselves, and satisfy transition times.
- Given the set of interval variables associated with the $unitBatchSeq_u$ sequence variable, the OPL function *typeOfNext* ($unitBatchSeq_u$, $task_{b,u,c}$) returns the value of the attribute associated with the interval variable that is next to $task_{b,u,c}$ in such sequence.
- The OPL keyword *cumulFunction* allows the definition of a cumulative function that represents a quantity varying over time and whose value depends on other decision variables. In scheduling problems; cumulative function expressions are associated with renewable resources. They model the cumulated usage of resources by different activities as a function of time. An activity usually increases the cumulated resource usage at its start time and halts the use when it releases the resource at its end time. There are two types of cumulative functions: step and pulse functions; depending on the type of use/consumption of the resource that is made by the activities. In the proposed CP model; a pulse-based cumulative function has been adopted. However; other functions could have been chosen. For instance; the consumption of cooling water can be represented by the cumulative function: $UsageProfile_{water} = \Sigma_{b\epsilon GB}\, pulse(task_{b,u,c};\, requir_{p,s,w})$; where the $task_{b,u,c}$ interval variable consumes an amount $requir_{p,s,w}$ of resource w when a batch b of product p starts its

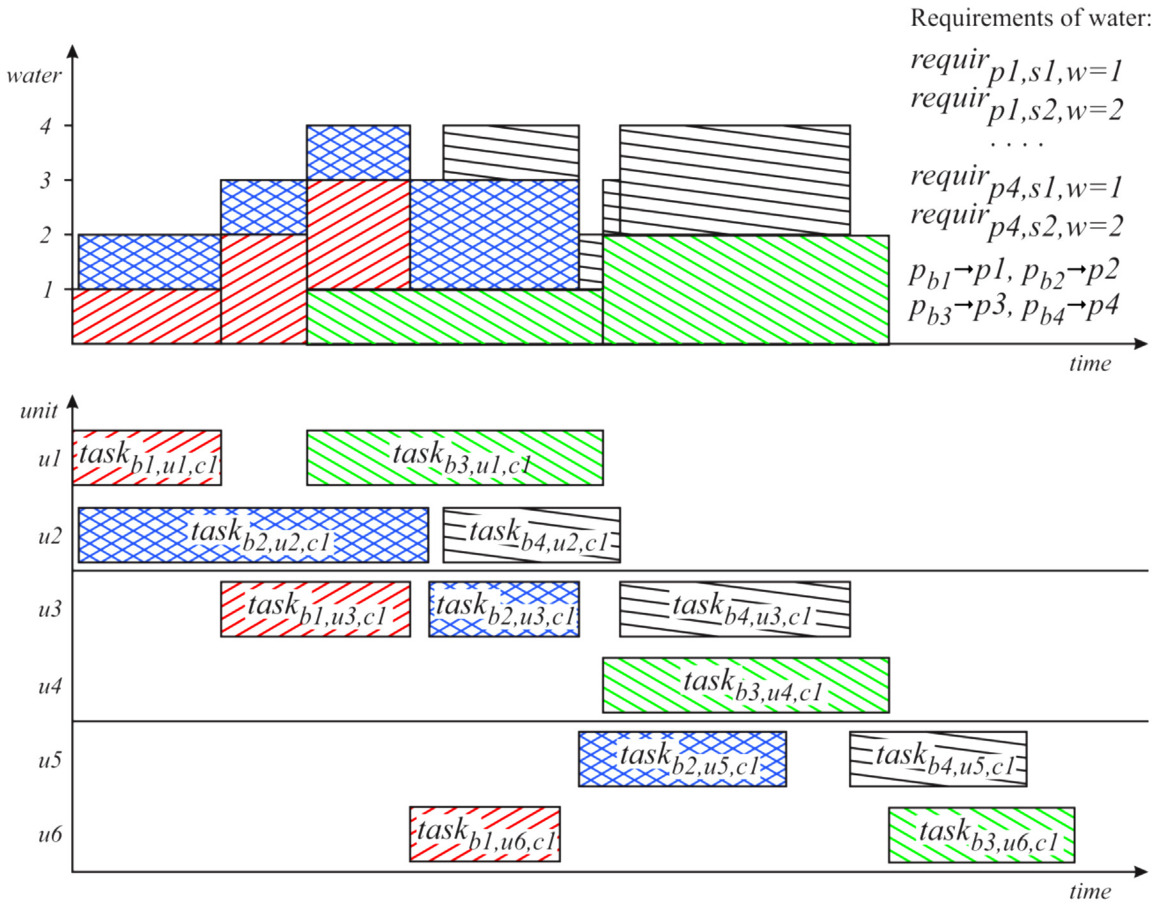

Fig. 2. Resource *water* usage profile represented by the cumulative function $UsageProfile_{water}$.

execution on unit u that belongs to stage s and releases the same quantity when it finishes. Fig. 2 illustrates the cumulative function $UsageProfile_{water}$; when different batches require distinct amounts of a resource w (water) in order to be executed in some of the units.

- The *span* function is employed to synchronize intervals that span over other ones. Thus, it is appropriate to model campaigns comprising a set of batches. For instance, the OPL constraint $span(campaignTask_{c1,p1,u1},\{task_{b1,u1,c1}, task_{b2,u1,c1}, task_{b3,u1,c1}, task_{b4,u1,c1}, task_{b5,u1,c1}\})$ states that the interval variable $campaignTask_{c1,p1,u1}$ will span over all the interval variables from the set of optional ones $\{task_{b1,u1,c1}, \ldots\ldots, task_{b5,u1,c1}\}$ that are present in the solution; i.e. interval $campaignTask_{c1,p1,u1}$ will start together with the first present interval variable ($task_{b1,u1,c1}$) and will end together with the last interval variable ($task_{b5,u1,c1}$) participating in the solution (see Fig. 3). On the contrary, if $campaignTask_{c1,p1,u1}$ is not part of the solution, then no interval from the set $\{task_{b1,u1,c1}, task_{b2,u1,c1}, task_{b3,u1,c1}, task_{b4,u1,c1}, task_{b5,u1,c1}\}$ will be part of it.
- The OPL *alternative* constraint provides an important way to control the execution and to synchronize different tasks. For instance, the constraint *alternative* $(stageTask_{b1,s1},\{task_{b1,u1,c1}, task_{b1,u2,c1}, task_{b1,u3,c1}\})$ represents an exclusive alternative among the elements of the set of *optional* interval variables $\{task_{b1,u1,c1}, task_{b1,u2,c1}, task_{b1,u3,c1}\}$ and synchronizes it with the non-optional variable $stageTask_{b1,s1}$, i.e. they will start and end together.
- The OPL constraints *endBeforeStart* and *endAtStart* are high-level constructs that represent precedence relationships between activity pairs. For instance, the constraint $endBeforeStart(stageTask_{b1,s1}, stageTask_{b1,s2})$ ensures that the end time of $stageTask_{b1,s1}$ occurs before the start time of $stageTask_{b1,s2}$. If the constraint *endAtStart* is used instead, $stageTask_{b1,s1}$ will end at the same time point that $stageTask_{b1,s2}$ starts.

4.2. CP model constraints and performance measures

4.2.1. Assignment, sequencing and task duration constraints

Expression (1) enforces each batch to be assigned to just one processing unit at each stage. The *alternative* construct is used to ensure that just one instance of the optional interval variable $task_{b,u,c}$ will be part of the resulting schedule. Those $task_{b,u,c}$ activities, which are actually present, are synchronized with the $stageTask_{b,s}$ ones.

$$alternative\left(stageTask_{b,s}, all\left(u \in U_s, c \in C_p\right) task_{b,u,c}\right) \quad \forall b \in B_p,\ \forall p \in P, \forall s \in S \tag{1}$$

Forbidden assignments of batches to units are addressed by simply not declaring the corresponding variables $task_{b,u,c}$. The duration (size) of each task depends on the unit assigned to it. Constraint (2) specifies the duration of tasks that are part of the solution under UIS or

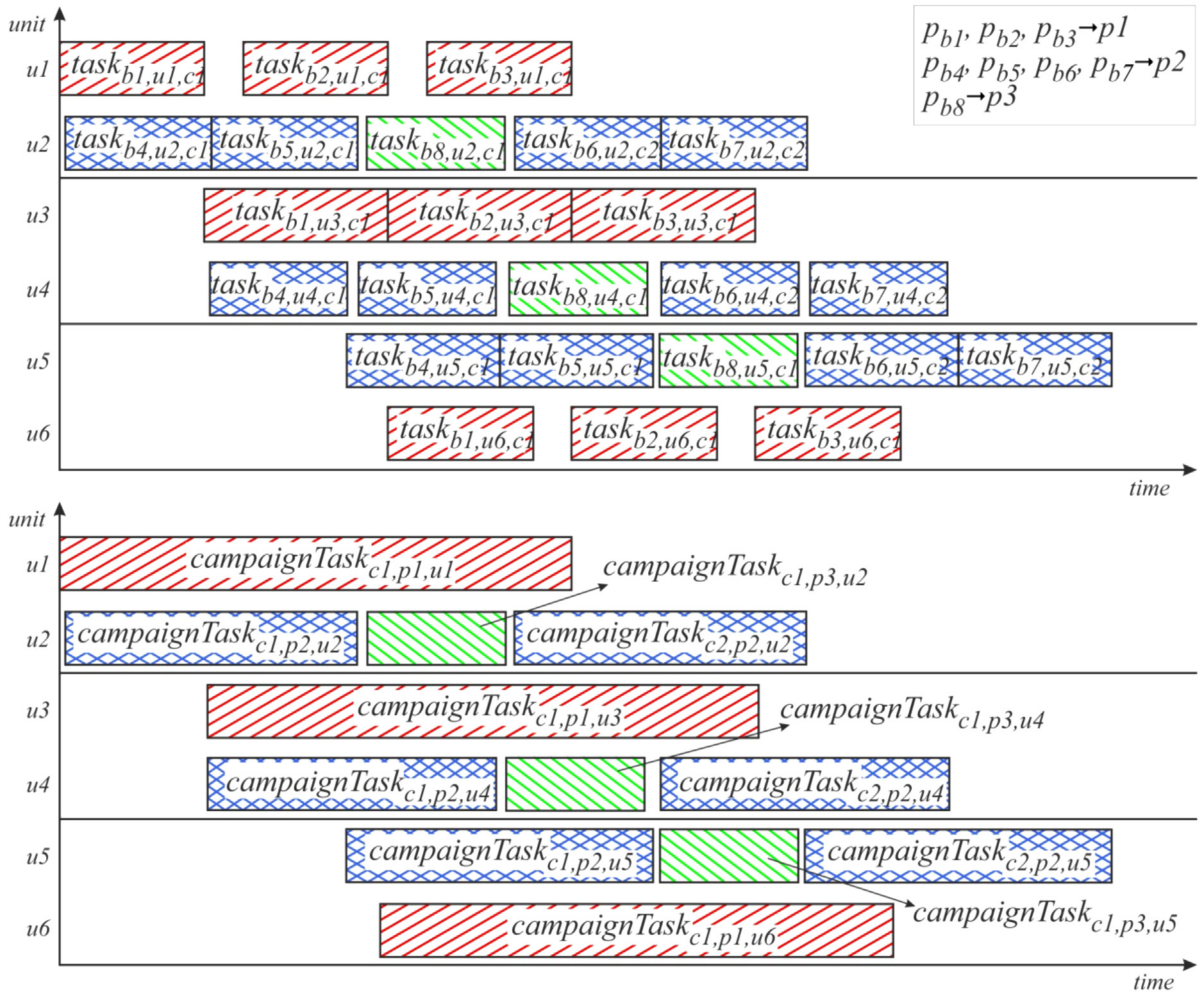

Fig. 3. *campaignTask* interval variables spanning over their corresponding processing tasks.

NIS/ZW policies, while constraint (2′) handles a NIS/UW policy. In case the adopted policy is NIS-FW, constraints (2′) and (3) are considered. The parameter $wt_{p,s}$ represents the maximum waiting time of product p at stage s.

$$sizeOf(task_{b,u,c}) = pt_{p,u}.presenceOf(task_{b,u,c}) \quad \forall b \in B_p, \forall u \in U, \forall c \in C_p, \forall p \in P \tag{2}$$

$$sizeOf(task_{b,u,c}) \geq pt_{p,u}.presenceOf(task_{b,u,c}) \quad \forall b \in B_p, \forall u \in U, \forall c \in C_p, \forall p \in P \tag{2′}$$

$$sizeOf(task_{b,u,c}) \leq (pt_{p,u} + \mathrm{wt}_{p,s}).presenceOf(\mathrm{task}_{b,u,c}) \quad \forall b \in B_p, \forall u \in U, \forall c \in C_p, \forall p \in P, \forall s \in S_u \tag{3}$$

Tasks belonging to a batch recipe must be executed without any overlapping, satisfying the prescribed inter-stage storage and waiting policies, as well as the precedence relationships. Constraint (4) models the precedence between two consecutive stage-tasks when the plant operates under an UIS policy. To handle NIS/ZW, NIS/UW and NIS/FW policies, constraint (4′) replaces (4).

$$endBeforeStart(stageTask_{b,s_1}, stageTask_{b,s_2}) \quad \forall b \in B_p, \forall p \in P, \forall s_1, s_2 \in S, s_1 + 1 = s_2 \tag{4}$$

$$endAtStart(stageTask_{b,s_1}, stageTask_{b,s_2}) \quad \forall b \in B_p, \forall p \in P, \forall s_1, s_2 \in S, s_1 + 1 = s_2 \tag{4′}$$

Each processing activity $task_{b,u,c}$ must start after the release-time corresponding to batch b and the ready-time of the assigned unit. Constraint (5) places a lower bound on the initial time for those tasks that are present on the solution.

$$\begin{aligned} &startOf(task_{b,u,c}) \geq max(rd_u, rt_b).presenceOf(task_{b,u,c}) \\ &\forall b \in B_p, \forall u \in U, \forall c \in C_p, \forall p \in P \end{aligned} \tag{5}$$

Constraint (6) enforces all the interval variables associated with a unit sequence variable $unitBatchSeq_u$ to not overlap each other (i.e. unit u is treated as a disjunctive resource). In addition, OPL allows handling transition times in an efficient and natural way. In a multiproduct batch plant, each batch to be processed can be naturally related to its product. Thus, when a unit consecutively executes two tasks associated with two different products, p and p', it will require a transition time that will be modelled by *changeOverTime*. However, when changeover times are not considered, constraint (6′) replaces constraint (6) to ensure that units are treated as disjunctive resources.

$$noOverlap(unitBatchSeq_u, changeOverTime) \quad \forall u \in U \tag{6}$$

$$noOverlap(unitBatchSeq_u) \quad \forall u \in U \tag{6′}$$

4.2.2. Forbidden production sequences and topology restrictions

Constraint (7) avoids forbidden batch production sequences. In order to accomplish that, the expression employs the *typeOfNext* construct.

$$typeOfNext(unitBatchSeq_u, task_{b,u,c}) \neq p' \qquad \forall b \in B_p, \forall u \in U, \forall c \in C_p, \forall (p, p') \in Fp \tag{7}$$

Constraint (8) enforces topology restrictions by ensuring that a batch is not going to be assigned to units belonging to consecutive stages that are not physically connected. Since u_2 belongs to the set of u_1 unconnected units, at most one task out of two consecutive ones of the same batch can be present in the solution. When a task is not present, its *endOf* value is 0.

$$\begin{gathered}\min(endOf(task_{b,u_1,c}), endOf(task_{b,u_2,c})) = 0 \\ \forall b \in B_p, \forall u_2 \in F_{u_1}, \forall u_1, u_2 \in U, \forall c \in C_p, \forall p \in P\end{gathered} \tag{8}$$

4.2.3. Campaign operating mode

Constraints (9)–(13) model different features concerning the campaign mode of operation. Constraints (9) and (10) allow fixing the lower and upper bounds, respectively, on the number of batches that belong to a particular product campaign.

$$\begin{gathered}\sum_{b \in B_p} presenceOf(task_{b,u,c}) \geq lB_p \cdot presenceOf(campaignTask_{c,p,u}) \\ \forall c \in C_p, \forall p \in P, \forall u \in U\end{gathered} \tag{9}$$

$$\begin{gathered}\sum_{b \in B_p} presenceOf(task_{b,u,c}) \leq uB_p \cdot presenceOf(campaignTask_{c,p,u}) \\ \forall c \in C_p, \forall p \in P, \forall u \in U\end{gathered} \tag{10}$$

By using the *span* CP construct, constraint (11) ensures that each processing task associated with a campaign takes place within the campaign spanning interval; i.e. all tasks related with a given campaign are encompassed by it.

$$span(campaignTask_{c,p,u}, all(b \in B_p) task_{b,u,c}) \qquad \forall c \in C_p, \forall p \in P, \forall u \in U \tag{11}$$

Constraint (12) enforces all the interval variables associated with a given campaign sequence variable, *unitCampaignSeq*$_u$, to not overlap each other, being separated by the corresponding transition times when the faced problem has sequence-dependent changeover times. This expression is similar to (6), but applied over campaign intervals instead of intervals related to batch tasks.

$$noOverlap(unitCampaignSeq_u, changeOverTime) \qquad \forall u \in U \tag{12}$$

Constraint (13) uses the *typeOfNext* construct to enforce those campaigns manufacturing the same product and belonging to *unitCampaignSeq*$_u$ not to be adjacent.

$$\begin{gathered}typeOfNext(unitCampaignSeq_u, campaignTask_{c,p,u}) \neq p \\ \forall c \in C_p, \forall p \in P, \forall u \in U\end{gathered} \tag{13}$$

When facing forbidden production sequences, an expression similar to (7) needs to be adapted and included in the model. In such constraint, the campaign sequence variable *unitCampaignSeq*$_u$ replaces the batch sequence one *unitBatchSeq*$_u$, and the campaign interval variable *campaignTask*$_{c,p,u}$ replaces the batch one *task*$_{b,u,c}$.

$$\begin{gathered}typeOfNext(unitCampaignSeq_u, campaignTask_{c,p,u}) \neq p' \\ \forall c \in C_p, \forall (p, p') \in Fp, \forall u \in U\end{gathered} \tag{14}$$

4.2.4. Limited resources

When limited renewable resources, such as cooling water, electricity, manpower, etc., are shared among the processing activities, the maximum capacity constraint (15) imposes an upper bound on the availability of each limited renewable resource, r. The maximum capacity, which depends on the resource, affects the usage profile represented by the *UsageProfile*$_r$ cumulative function that is defined by constraint (16). When *task*$_{b,u,c}$ is present in the solution, it demands a fixed amount of the resource that depends on the product and the stage, *requir*$_{p,s,r}$. The demand of resource r at each moment, originated from the execution of all product batches in all the units, defines the usage profile of r, and cannot exceed its availability.

$$UsageProfile_r \leq avail_r \qquad \forall r \in R \tag{15}$$

$$UsageProfile_r = \sum_{\forall c \in C_p} \sum_{\forall b \in B_p} \sum_{\forall p \in P} \sum_{\forall u \in U_s} \sum_{\forall s \in S} pulse(task_{b,u,c}, requir_{p,s,r}) \qquad \forall r \in R \tag{16}$$

An interesting situation appears when limited resources, such as cleaning-in-place devices, are required by changeover tasks. This case demands additional constraints that explicity model the cleaning activities and their associated resource demands. Constraint (17) relates the duration of the cleaning task required between the execution of batches of $p1$ and $p2$ in unit u with the corresponding changeover time associated with this batch sequence. The maximum capacity constraint (18) imposes an upper bound on the number of available cleaning devices. This upper bound affects the usage profile represented by the *UsageProfile*$_{CIP}$ cumulative function defined by constraint (19). In this expression the *cleanRequir*$_{b1,b2,u,CIP}$ parameter is equal to one since each cleaning task, when performed, requires just one device. Similar constraints can be defined for other limited resources demanded by the changeover activities, such as manpower or electricity, by simply adopting the appropriate values of the maximum availability and requirement parameters, $avail_r$ and $cleanRequir_{b1,b2,u,r}$, respectively.

$$sizeOf\left(cleanTask_{b1,b2,u}\right) = changeOverTime \cdot presenceOf\left(cleanTask_{b1,b2,u}\right) \quad \forall b1 \in B_{p1}, \forall b2 \in B_{p2}, \forall u \in U : (p1, p2, u) \in changeOverTime \tag{17}$$

$$UsageProfile_{CIP} \leq avail_{CIP} \tag{18}$$

$$UsageProfile_{CIP} \leq \sum_{\forall b1 \in B} \sum_{\forall b2 \in B} \sum_{\forall u \in U} pulse\left(cleanTask_{b1,b2,u}, cleanRequir_{b1,b2,u,CIP}\right) \tag{19}$$

The start and end of the cleaning tasks associated variables need to be synchronized with the finishing and begin times of the precedesor and successor processing tasks, respectively. This synchronization is achieved by means of expressions (20) and (21).

$$endOf\left(task_{b1,u,c}\right) \cdot presenceOf\left(cleanTask_{b1,b2,u}\right) \leq startOf\left(cleanTask_{b1,b2,u}\right) \quad \forall b1, b2 \in B, \forall c \in C, \forall u \in U \tag{20}$$

$$endOf\left(cleanTask_{b1,b2,u}\right) \leq startOf\left(task_{b2,u,c}\right) \cdot presenceOf\left(cleanTask_{b1,b2,u}\right) \quad \forall b1, b2 \in B, \forall c \in C, \forall u \in U \tag{21}$$

In addition, the number of cleaning tasks in a given unit u should be related to the number of processing tasks in the same unit.Constraint (22) imposes that the number of cleaning tasks is equal to the number of processing ones minus one. Similarly, constraint (23) prescribes that if a cleaning activity associated with the transition of batches $b1$ and $b2$ in unit u takes place, then, batches $b1$ and $b2$ should also be processed in the same unit.

$$max\left(0, \sum_{\forall b \in B} \sum_{\forall c \in C} presenceOf\left(task_{b,u,c}\right) - 1\right) = \sum_{\forall b1 \in B} \sum_{\forall b2 \in B} presenceOf\left(cleanTask_{b1,b2,u}\right), \forall u \in U \tag{22}$$

$$presenceOf\left(cleanTask_{b1,b2,u}\right) \Rightarrow \sum_{\forall c \in C} presenceOf\left(task_{b1,u,c1}\right) + presenceOf\left(task_{b2,u,c2}\right) = 2 \quad \forall b1, b2 \in B, \forall u \in U, \forall c1, c2 \in C \tag{23}$$

Constraints (24) and (25) state that if batch $b1$ ($b2$) is processed in unit u, then, there should be at most one cleaning task associated with a possible successor batch $b2$ (predecessor batch $b1$).

$$\sum_{\forall b2 \in B} presenceOf\left(cleanTask_{b1,b2,u}\right) \leq 1, \forall u \in U, \forall b1 \in B \tag{24}$$

$$\sum_{\forall b1 \in B} presenceOf\left(cleanTask_{b1,b2,u}\right) \leq 1, \forall u \in U, \forall b2 \in B \tag{25}$$

4.2.5. Accelerating constraints

Expressions (26)–(28) are a set of accelerating constraints introduced to improve the performance of the solution search process. Constraint (26) ensures that a campaign variable, related to a product p in unit u, is not present if the variable associated with the previous campaign of the same product on the same unit does not exist in the solution. Constraint (27) establishes the precedence between campaigns of the same product allocated to the same unit. Since $campaignTask_{c,p,u}$ variables are optional and defined over a set C_p of possible campaigns, expressions (26) and (27) are employed to ensure that the number of present campaigns j, with $j < card(C_p)$, are ordered from 1 to j.

$$presenceOf(campaignTask_{c_1,p,u}) \geq presenceOf(campaignTask_{c_2,p,u}) \quad \forall c_1, c_2 \in C_p, c_1 < c_2, \forall p \in P, \forall u \in U \tag{26}$$

$$endBeforeStart(campaignTask_{c_1,p,u}, campaignTask_{c_2,p,u}) \quad \forall c_1, c_2 \in C_p, c_1 < c_2, \forall p \in P, \forall u \in U \tag{27}$$

Constraint (28) enforces the presence of the campaign variable $campaignTask_{c,p,u}$ every time the interval $task_{b,u,c}$ is present in the solution.

$$presenceOf(task_{b,u,c}) \Rightarrow presenceOf(campaignTask_{c,p,u}) \quad \forall b \in B_p, \forall u \in U, \forall c \in C_p, \forall p \in P \tag{28}$$

4.2.6. Objective functions

The CP formulation can deal with several efficiency related objective functions, such as makespan, total tardiness, number of tardy batches, as well as cost related ones. When makespan is chosen, constraint (29) has to be included in the model as the performance measure to be optimized.

$$minimize\ max(endOf(stageTask_{b,s})) \quad \forall b \in B_p, \forall p \in P, s = nbStages \tag{29}$$

In case tardiness needs to be avoided, constraint (30) is the objective function to be optimized. Each batch b is associated with a tardiness variable tar_b that assumes a positive value when the last processing operation finishes after the due-date dd_b. This value is defined by expression (31).

$$minimize \sum_{b \in B} tar_b \tag{30}$$

$$tar_b = max(0, endOf(stageTask_{b,s}) - dd_b) \quad \forall b \in B_p, \forall p \in P, s = nbStages \tag{31}$$

If the total number of tardy batches is minimized, then expressions (32) and (33) must be included in the model. Expression (24) specifies whether the last operation of a batch b of product p is tardy or not. If it is, the variable $nTar_b$ takes the value 1, and 0 otherwise.

$$minimize \sum_{b \in B} nTar_b \tag{32}$$

$$nTar_b = min(max(0, endOf(stageTask_{b,s}) - dd_b), 1) \qquad \forall b \in B_p, \forall p \in P, s = nbStages \tag{33}$$

In addition to the traditional time based performance measures, the model can efficiently deal with cost related objective functions. In this proposal, a fixed cost is associated with each processing of a batch of product p on unit u, $cost_{p,u}$. Expression (34) minimizes the sum of those costs linked with the processing tasks actually scheduled.

$$minimize \sum_{\forall b \in B_p} \sum_{\forall u \in U} \sum_{\forall c \in C_p} \sum_{\forall p \in P} presenceOf(task_{b,u,c}) \cdot cost_{p,u} \tag{34}$$

Constraint (35), used by Harjunkoski and Grossmann (2002), presents an objective function that deals with processing costs, which depend on both, the assignment cost, $cost_{p,u}$, and a one-time cost such as initialization or startup for a specific unit, $fcost_u$. Constraint (36) captures the units that are actually used during the scheduling horizon.

$$minimize \sum_{\forall b \in B_p} \sum_{\forall u \in U} \sum_{\forall c \in C_p} \sum_{\forall p \in P} presenceOf\left(task_{b,u,c}\right) \cdot cost_{p,u} + \sum_{\forall u \in U} z_u \cdot fCost_u \tag{35}$$

$$z_u \geq presenceOf(task_{b,u,c}) \qquad \forall b \in B_p, \forall u \in U, \forall c \in C_p, \forall p \in P \tag{36}$$

5. Case studies and computational results

The computational performance of the proposed CP formulation has been tested through the solution of several sets of case studies. They vary in the number of considered production orders, plant topology, storage and waiting operation policies, different constraining features, as well as performance measures taken into account. In the first part of this section, the results obtained by the CP proposal are compared with the ones reported by various recent contributions. The second part shows the solutions obtained when addressing a set of new large-size problem examples based on the pharmaceutical facility presented by Castro et al. (2009). All the case studies have been solved using the CPLEX Optimization Studio 12.5.1 (IBM ILOG, 2013) in a Dell Precision T1650. Intel® Xen® CPU E3-1240 V2 @ 3.40 GHz., 8 GB RAM, 64 bits computer.

5.1. *Examples with single batch product orders*

All the case studies presented in this section assume that each product order comprises a single batch. Five examples were selected from literature in order to test the CP formulation. Data sets related with them are not shown here due to lack of space. However, all the data can be found in the original contributions where they were introduced. A brief description of each example is presented in the following paragraphs.

5.1.1. *Example 1*

This case study was introduced by Castro et al. (2009). The problem is taken from a real pharmaceutical production facility that has 6 processing stages with 17 dissimilar processing units. Originally named as P13, this example consists of 50 batches corresponding to different products that require to be scheduled under a UIS policy. There are some banned unit-batch assignments and the adopted performance measure is makespan. This example is labeled as P1.1 in Table 3.

5.1.2. *Example 2*

The second case study, which is based on the same production facility of example 1, was introduced by Kopanos et al. (2010). It considers the scheduling of 60 orders, includes stage-dependent changeover times, as well as UIS and NIS/ZW policies. Some forbidden unit-batch assignments need to be considered. The solutions for the examples identified as I.04, I.05, I.11 and I.12 in Kopanos et al. (2010), are reported in Table 3 as P2.1 and P2.2, respectively. The two cases minimize makespan as the objective function; P2.1 considers UIS and P2.2, NIS-ZW.

5.1.3. *Example 3*

This example is based on the 5-stages multiproduct batch plant having 25 non-identical units and topological constraints that was studied by Zeballos et al. (2011). Product orders and processing units are characterized by release and ready times, respectively. In addition, certain orders cannot be processed in some units and there are forbidden processing sequences, as well as sequence dependent changeover times. Scheduling problems were solved for different number of orders and storage/waiting policies, minimizing makespan. The results for problem instances considering 12 orders and UIS, NIS/ZW and NIS/UW policies, are presented in Table 3 as P3.1, P3.2, and P3.3, respectively. Those instances with 22 orders and UIS, NIS/ZW and NIS/UW policies are shown as P3.4, P3.5, and P3.6, respectively.

Table 2
Set of expressions used to solve the examples in Section 5.1..

Case study	Expressions*	Variables			Constraints
		Interval	Sequence	Integer	
1.1–1.2	(1), (2), (4), (6)[a], (29)	1054	17	0	3621
2.1–2.2	(1) – (4), (6), (29)	1100	17	0	3647
3.1–3.2–3.3	(1) – (8), (29	319	25	0	2228
3.4–3.5–3.6	(1) – (8), (29)	587	25	0	4207
4.1	(1) – (4), (5)[b], (6)[c], (7), (8), (15), (16), (30), (31)	176	12	0	1243
4.2	(1) – (4), (5)[b], (6)[c], (7), (8), (17) – (25), (30), (31)	1109	12	0	13,772
5.1–5.2	(1) – (6)[a], (8), (35), (36)	130	8	8	920

*Expressions (2) and (4) specified according to the addressed storage & waiting operational policy.
[a] Expression (6) represented without the changeover parameter.
[b] The rt_u parameter is replaced by st_u parameter.
[c] Changeover times also contemplate setup times.

Table 3
Computational results for test problems considering single batch product orders. Comparison with other approaches.

Problem	Results reported by other authors		Proposed CP approach					
	First solution	Best solution	First solution		Best solution (900 s)		Best solution (3600 s)	
	Objective function value	Objective function value	Objective function value	CPU Time (s)	Objective function value	CPU Time (s)[a]	Objective function value	CPU Time (s)[b]
P1.1	–	30.053	29.233	1.43	**28.533**	596.93	NI	–
P2.1	49.161	48.5480	55.6182	1.46	**43.9371**	763.32	NI	–
P2.2	58.104	56.0610	69.8031	0.92	48.72	872.8	**48.6045**	3103
P3.1	354.00	293.1	230.1	0.35	**199.0**	4.19	NI	–
P3.2	381.00	311.2	211.0	0.90	**199.0**	5.00	NI	–
P3.3	370.10	301.2	208.4	0.43	**199.0**[c]	7.06	NI	–
P3.4	534.40	509.4	318.2	1.29	263.9	630.24	**256.0**	1294
P3.5	–	–	293.0	0.59	260.0	20.34	**255.9**	2793
P3.6	592.40	550.4	288.6	0.63	261.0	371.7	**260.0**	3132
P4.1	–	**31.6**	131.1	0.39	**31.6**	12.43	NI	–
P4.2	–	–	456.7	1.66	102.8	656.1	101.2	1312.15
P5.1	–	111	**111**	0.50	NI	–	NI	–
P5.2	–	**704**	787	0.32	**704**[c]	0.95	NI	–

d. NI Noimprovements in the already found solution.
[a] Time required to obtain optimal solutions or to instantiate suboptimal ones within 900CPU seconds.
[b] Time required to obtain optimal solutions or to instantiate suboptimal ones within 3600CPU seconds.
[c] Optimal solution.

5.1.4. *Example 4*

The fourth case study corresponds to a facility having 5 processing stages, 12 units that feature setup times and topological constraints, presented by Marchetti and Cerdá (2009a). Twelve different orders having distinct due-dates are to be scheduled, but some of them cannot be processed in certain units. Sequence dependent changeover times are considered. Additionally, limited availabilities of resources different than equipment (e.g. steam or electricity), which are required by particular batches at some stages, are considered. The goal is to find a schedule that minimizes total tardiness, under a UIS policy. The problem originally named by Marchetti and Cerdá (2009a) as "4d.With all resources" is the one that is labeled as P4.1 in Table 3. It was chosen because it is one of the most complex case studies in Marchetti and Cerdá (2009a). P4.2 is another example that introduces additional complexity, since it also considers the limited cleaning resources that are employed during the changeover activities, stating that there is just one CIP (cleaning-in-place) device available.

5.1.5. *Example 5*

This example, introduced by Harjunkoski and Grossmann (2002), addresses a multistage plant with 8 machines and 3 stages, where 12 jobs require to be scheduled. The problem considers a UIS policy, some forbidden batch-unit assignments and paths. Two instances of this case study, named as P5.1 and P5.2, were solved using a cost-based performance measure (expression 35).

The model introduced in Section 4.2 has been simplified to deal with one batch per order. This can be easily achieved by fixing the *c* index equal to 1, representing the absent campaigns, and neglecting those constraints modeling specific campaign related issues (9–14 and 26–28).

The equations employed to solve the examples in this section undergo slight variations based on the addressed features. For instance, as it was explained in Section 4.2, constraint (4) or constraint (4′) are incorporated depending on whether UIS or NIS/ZW policies are considered, respectively. Table 2 specifies the constraints that are employed to solve the addressed problems, as well as the size and composition of each CP resulting model.

In Table 3, the solutions obtained by the proposed CP model and the ones reported by other authors are shown. The case studies are labeled as P*x*.*z*, where *x* is the number of the example considered, and *z* is an instance of the given example *x*. The objective function value and the CPU time are shown for each of the problem instances. Concerning the CP approach, the first solutions were captured, as well as the ones instantiated when the limits of 900 s and 3600 s of CPU were reached. The results that are reported have been obtained with the solver default parameters, with the exception of the search type, which was fixed to "Restart" (depth-first search that restarts according to predefined parameters).

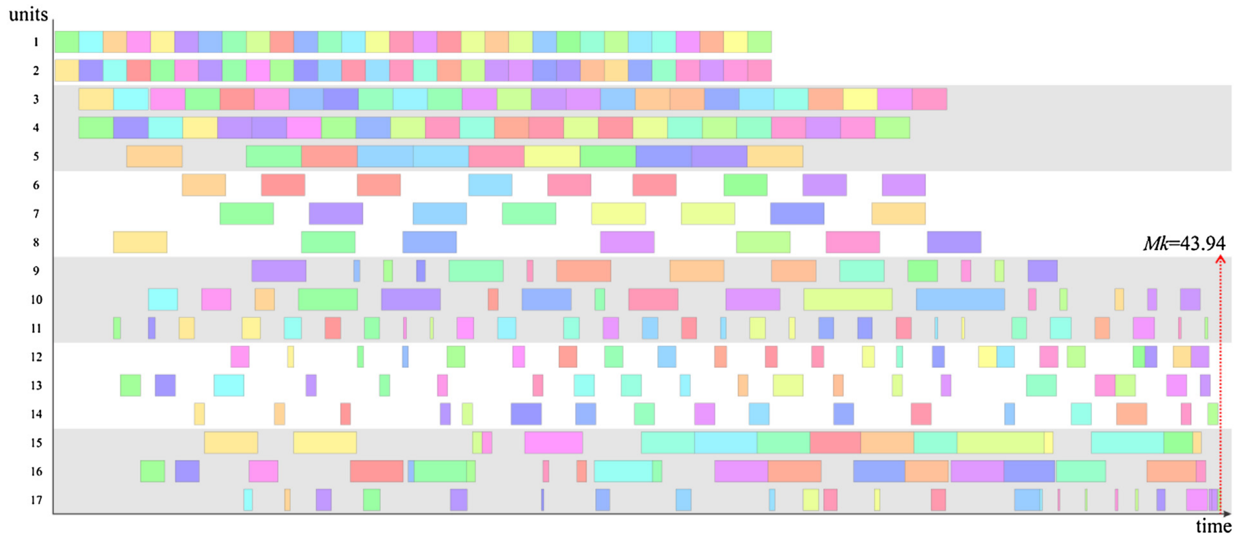

Fig. 4. Gantt chart representing the best production schedule for problem P2.1.

An analysis of the results reveals that all the solutions obtained with the proposed CP approach are of better quality than the ones previously reported, except for the examples P4.1 and P5.2, for which the proposal obtains the same solutions. Most of the results are suboptimal (optimality is only ensured for problems P3.3 and P5.2). Note that good quality solutions are reached for all the tackled problems in 900 s of computational time. Moreover, as it can be observed in Table 3, problems P2.2, P3.4, P3.5, P3.6 and P4.2 obtain better solutions when the solving time limit is extended to 3600 s of CPU time.

It can also be noted that the first solutions are instantiated in around one second of CPU time and, in most cases (8 out of 13), a very good first objective function value is found. The first solution of problems P1.1, P3.1, P3.2, P3.3, P3.5, P3.6, P5.1, and P5.2 is, for each case, at most 15% over the best solution found in 900 s of CPU.

Fig. 4 shows the Gantt chart corresponding to the best solution obtained for the P2.1 problem, which was found within 900 s of CPU time. In this case, the obtained makespan is 9.5% smaller than the one reported by Kopanos et al. (2010). For problems P3.1, P3.2, P3.3, P3.4, and P3.6, the makespan improvement is more than 30%. In addition, the CP approach found good quality solutions for the example P3.5, for which the proposal of Zeballos et al. (2011) did not instantiate any solution. For the rest of the case studies, the improvement was less than 15% in the examples P1.1, P2.1 and P2.2, and none for problems P4.1, P5.1 and P5.2.

An interesting case appears when the limited resources are demanded by the cleaning activities instead of the processing ones. The modeling of this situation is more complex, due to the need of explicitly representing cleaning tasks and to synchronize them with the processing ones; thus, resulting in a larger model. As it follows from Table 2, when comparing the models of problems P4.1 and P4.2, both the number of variables and constraints increase. In addition, some of the extra contraints (20–21) are nonlinear. The resulting larger and more complex model demands additional computational effort. In fact, while the optimality of model P4.1 solution can be proved in only 12.43 s, the solution of problem P4.2 is the suboptimal one that can be obtained in 3600 s. Nonetheless, to the best of our knowledge, other contributions have just considered limited resources that are demanded by processing activities, while problem P4.2 furnishes a much greater challenge. Fig. 5 depicts the Gantt charts of problems P4.1 (upper part) and P4.2 (bottom) solutions, along with their corresponding usage profile of CIP devices. An analysis of the figure shows that when no limitations are imposed, there are intervals during which three CIP devices are simultaneously demanded. Conversely, when just one is available, the solution seriously deteriorates, the batch sequence changes and both the makespan and total tardiness significantly increase. These results highlight the gross errors that can be introduced when models are oversimplified by omitting relevant constraints.

5.2. *Multiple batches per product order & campaign mode operation*

This section addresses large-scale examples in which each demanded product order comprises several batches (multi-batch orders). All the instances solved in this section are based on the pharmaceutical plant scheduling problem tackled by Castro et al. (2009) and presented in Section 5.1 as Example 1. However, the following case studies introduce some modifications to it, in order to tackle several features found at industrial settings, which have not been addressed by other contributions yet.

5.2.1. *Example 6*

This example is an extension of example 1 aimed at considering the following characteristics: topological constraints, forbidden processing sequences, as well as sequence- and unit-dependent changeover times. In this case study, 50 different product orders with their own release times require to be scheduled, and each order comprises 4 batches (a total of 200 batches is tackled). Depending on the adopted objective function, storage and waiting policy, four problem instances are defined. The P6.1 and P6.2 problems, which consider UIS and NIS-ZW policies, respectively, minimize makespan. The P6.3 and P6.4 problem instances employ the cost-based performance measure presented in Section 4.2 (expression 33), and adopt UIS and NIS-ZW policies, respectively.

5.2.2. *Example 7*

This case study considers a plant that operates under a campaign production policy and has unlimited intermediate storage. It is based on example 1, but avoiding forbidden processing sequences and topological constraints. Seven different products are to be manufactured

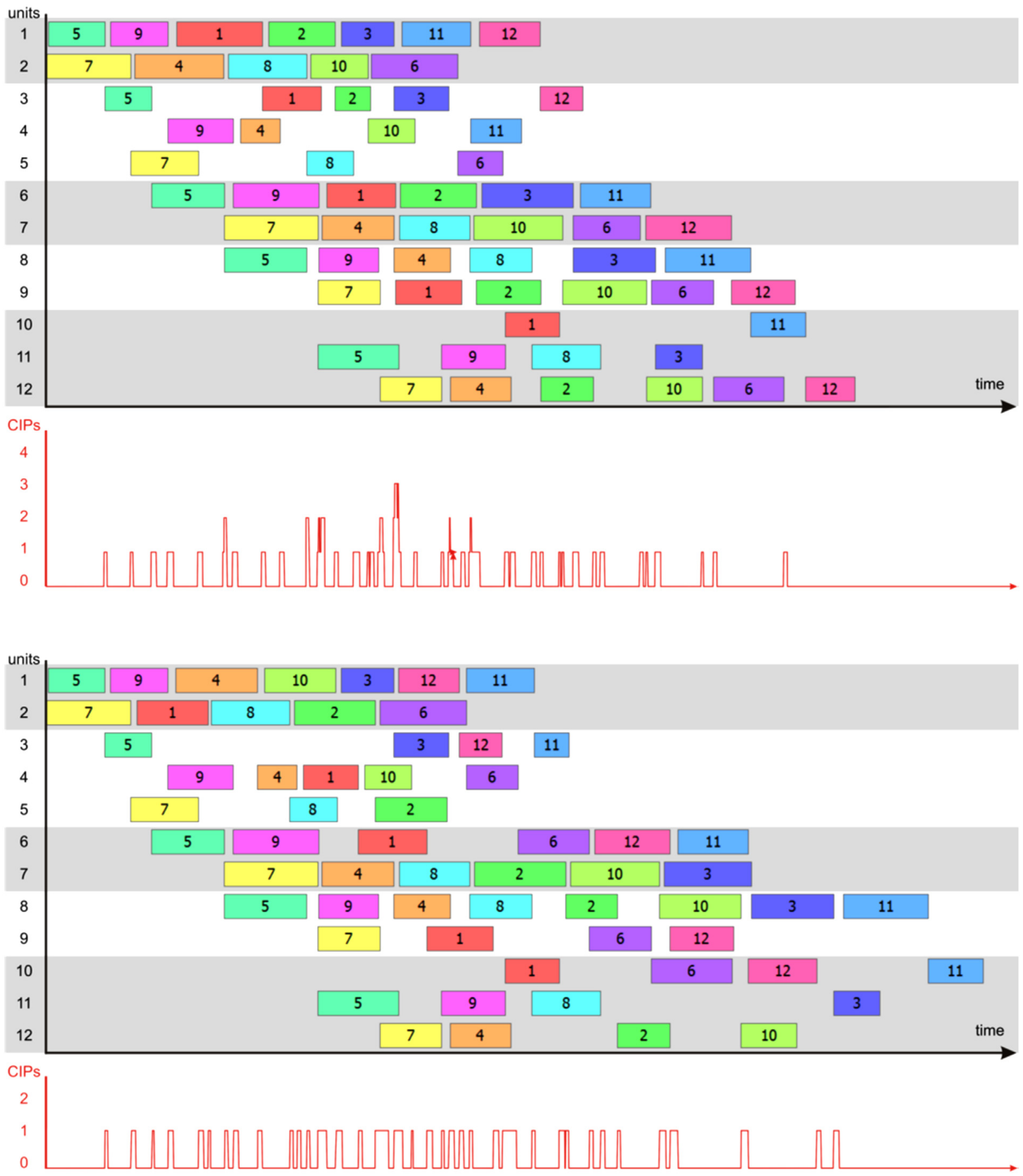

Fig. 5. Gantt chart comparing the best production schedules for problems P4.1 and P4.2.

in the plant. These products, identified as 15, 17, 26, 27, 29, 44 and 48, were selected randomly from the original problem. Nine product orders, having between 4 and 10 batches each, need to be scheduled (i.e. a total of 70 batches). Each order has a due-date and two of them correspond to the same product. Product campaigns have distinct lower and upper bounds on the number of batches that can comprise. Campaigns are separated by the corresponding sequence-dependent changeover time. Four problem instances, which vary the adopted performance measure, are solved: (i) Problem P7.1 corresponds to a makespan minimization example, in which due-dates are neglected; (ii) Case P7.2 consideres total tardiness as the objective function; (iii) Example P7.3, which minimizes the number of tardy batches; and (iv) Problem P7.4 that employs a cost-based objective function (expression 34).

Table 4 shows the expressions that need to be considered to solve the different problem instances, as well as the size and composition of each CP resulting model. The data associated with these examples can be found in the Supporting material.

Table 5 reports the first solutions and the best instantiated results after 900 s and 3600 s of CPU, for the various instances of examples 6 and 7. The results have been obtained with the solver default parameters, with the exception of the search type, which was fixed to “Restart” (depth-first search that restarts according to predefined parameters).

Table 4
Set of expressions employed in the solution of the examples in Section 5.2.

Case study	Expressions[a]	Variables			Constraints
		Interval	Sequence	Integer	
6.1, 6.2	(1) – (8), (29)	4216	17	0	29,257
6.3, 6.4	(1) – (8), (34)	4216	17	0	32,273
7.1	(1) – (6), (9) – (14), (26) – (29)	5453	34	0	93,554
7.2	(1) – (6), (9) – (14), (26) – (28), (30), (31)	5453	34	70	93,624
7.3	(1) – (6), (9) – (14), (26) – (28), (32), (33)	5453	34	70	93,624
7.4	(1) – (6), (9) – (14), (26) – (28), (34)	5453	34	0	98,094

[a] Expressions (2) and (4) are specified according to the addressed storage & waiting operational policy.

Table 5
Computational results for large-scale test problems.

Problem	Proposed CP approach					
	First solution		Best solution (900 s)		Best solution (3600 s)	
	Objective function value	CPU Time (s)	Objective function value	CPU Time (s)[a]	Objective function value	CPU Time (s)[b]
P6.1	154.086	41.97	129.889	889.85	128.244	2956.48
P6.2	203.557	27.08	185.977	877.34	162.417	3543.95
P6.3	211.275	23.23	210.482	782.68	**210.446**	1563.76
P6.4	211.328	60.65	211.074	289.75	**210.506**	3452.05
P7.1	NS	–	–	–	–	–
P7.2	26.514	893.79	NI	–	**0**[c]	2135.85
P7.3	5	299.13	3	593.61	**2**	3158.31
P7.4	90.453	**2748.45**	NI	–	NI	–

NS No solution was found in 3600 s.
NI No improvement in the already found solution.
[a] Time required to obtain optimal solutions or to instantiate suboptimal ones within 900CPU seconds.
[b] Time required to obtain optimal solutions or to instantiate suboptimal ones within 3600CPU seconds.
[c] Optimal solution.

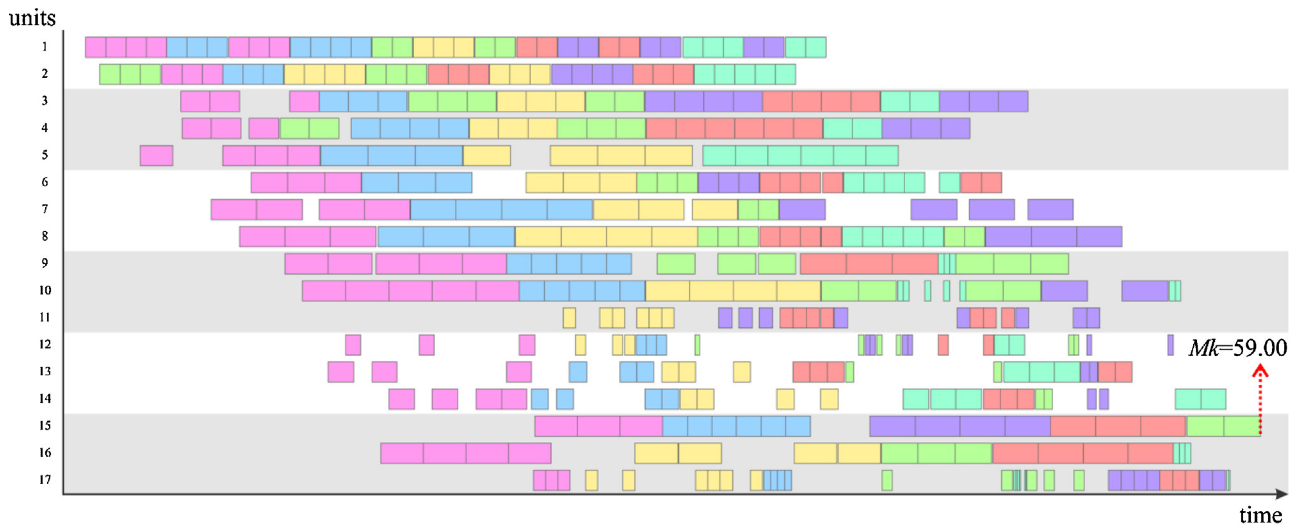

Fig. 6. Gantt chart representing the optimal schedule for problem P7.2 when minimizing tardiness.

The results reveal that big size case studies having 200 and 70 batches have been successfully solved. Moreover, problems P7.1, P7.2, P7.3, and P7.4 captured the campaign operational mode of many industrial batch plants, and the solutions found for most of these problem instances show that good quality results can be obtained for various objective functions. For instance, the P7.2 problem, which optimizes total tardiness, reaches its optimal solution in 2135 s of CPU.

First solutions are found in at most 60 s of CPU for all instances of example 6. A bit longer computational time is required to find the first solution for problems P7.2–P7.4, which has a cost-based objective function. No solution was found for problem P7.1.

The first solution values are improved in more than 15%, within the time limit of 900 s of CPU, for 2 out of the 8 case studies. When the time limit is extended to 3600 s, improvements over 15% are achieved for 4 out of the 8 examples.

Fig. 6 depicts the Gantt chart of the optimal solution that was reached for the P7.2 problem, which considers a campaign operating mode and minimizes total tardiness as the objective function. In this case, the obtained makespan value was 59.00.

The modification of solver parameters has been tried for the bigger size examples in order to assess their influence on the computational performance. Among the adjustable parameters, the instantiation ordering strategy (denoted as “Search phase” in the CPLEX Optimization

Table 6
Computational results for P7.1–P7.4 problems, considering an instantiation ordering strategy.

Problem	Proposed CP approach					
	First solution		Best solution (900 s)		Best solution (3600 s)	
	Objective function value	CPU Time (s)	Objective function value	CPU Time (s)[a]	Objective function value	CPU Time (s)[b]
P7.1	62.398	190.75	54.745	862.49	50.235	3005.78
P7.2	40.355	176.28	6.183	891.77	**0**[c]	1154.67
P7.3	7	175.71	**4**	602.08	NI	–
P7.4	91.216	296.19	88.028	836.04	**81.042**	3552.59

NI No improvement in the already found solution.
[a] Time required to obtain optimal solutions or to instantiate suboptimal ones within 900CPU seconds.
[b] Time required to obtain optimal solutions or to instantiate suboptimal ones within 3600CPU seconds.
[c] Optimal solution.

Studio 12.5.1), which allows defining an ordering of variables to be instantiated, appears to have a positive impact on the obtained solutions. Table 6 shows the results that were reached when the task interval variables are chosen to be instantiated first, for the P7.1–P7.4 problems.

When comparing these solutions with the corresponding ones on Table 5 (cases in which no instantiation ordering strategy was adopted), improvements in three out of the four problem instances are observed. When the proposed solution strategy is applied to P7.1, a first solution was found in less than 200 s and the final result was improved in almost 20% when the time limit was extended to 3600 s. In the case of problem P7.2, its optimum was found in almost half of the time that was required when no strategy was adopted. Finally, for problem P7.4 an improvement of 10% was reached. However, when addressing problem P7.3 the best solution presented in Table 5 could not be obtained.

Provided the task sequence variables, as well as the campaign task and campaign sequence ones, depend on the task interval variables, it makes sense to achieve an improvement when all the $Task_{b,u,c}$ variables are first instantiated.

6. Conclusions and future work

An efficient CP model that allows addressing large-scale scheduling problems associated with multiproduct multistage batch plants has been introduced. This novel approach accounts for several issues found in actual industrial environments, such as dissimilar parallel units at each stage, topology constraints, forbidden product-equipment assignments, sequence-dependent changeover tasks, intermediate due-dates, different types of constraining resources, as well as various interstage storage and waiting policies. With regards to resource limitations, it is important to remark that the proposed model lets to account for requests associated with both processing and cleaning activities. With respect to these last ones, many models assume that changeover activities do not demand resources, but one of the addressed problem instances shows that the obtained solutions can be completely unrealistic if restraining resources are ignored.

In addition, the formulation can deal with problems that involve multiple-batch orders, which are processed in plants that operate in a campaign mode and have lower and upper limits on the number of batches per campaign. Campaigns simplify the plant operation and allow reducing scrap and changeover times.

The proposed CP approach does not resort to any decomposition strategy and deals with unit assignment, batch and campaign sequencing, as well as timing decisions in an integrated way, while optimizing a time-based or cost-based objective function. Conversely, it is assumed that the number and size of the batches have been fixed beforehand and are taken as given data. Though this can be seen as a drawback, it tries to emulate to some extend how industry works. In fact, industrial scheduling takes the batch sizes – that are in the master recipes predefined – as given inputs, with the aim of transforming these recipes into control ones. Nevertheless, it would be more realistic to select the batch sizes from the predefined set of sizes that correspond to the feasible master recipes, which are associated with each product. This type of decision, along with the number of batches of each type, need to be included in the CP formulation. To the best of our knowledge, this issue has not been dealt with up to now, and we envision it as future work.

The computational performance of the proposal has been tested through several sets of case studies having different sizes and characteristics. The first set of problems corresponds to cases in which each product order comprises a single batch; for these instances the approach has generally found better quality solutions than the ones previously reported by other authors. In addition, the first solutions, which are associated with good quality values of the objetive function, are instantiated quite soon. Nevertheless, optimality could only be guaranteed for a reduced number of examples. When dealing with large-size problem instances, in which each product order comprises several batches, the approach also found good quality soluctions. The obtained results have shown that the examples could be successfully solved within the imposed time limit, even when addressing a campaign operating mode.

It was observed that the performance of the CP model is sensitive to the modification of one of the solver parameters, which is the instantiation ordering strategy. The specification of the order in which the variables are to be instantiated has a positive impact on the solutions.

In summary, a very expressive CP model, able to capture many industrial features, has been proposed. The examples that have been presented show that this novel formulation is computationally efficient. Nevertheless, its expressiveness and efficiency partly depend on its underying implementational language and solution engine, which are the ILOG-IBM OPL language and CP Optimizer, both embedded within the CPLEX Optimization Studio (IBM ILOG, 2013). The adopted modeling and solution platform provides simple but very expressive high level modeling concepts, such as intervals, functions and sequences that are specially well suited for modeling scheduling problems, as shown in Section 4.1.

Naturally, the use of another CP platform would demand additional modeling and assessment efforts, which could be seen as a drawback. The development of models under different platforms has drawn the attention of the CP community since more than a decade ago (Fernández and Hill, 2000; Shcherbina et al., 2003; Wallace et al., 2004) as both the number of CP engines and applications have increased. Lately, Mancini et al., (2008) evaluated the relative performance of three solvers for combinatorial problems belonging to the CSPLib library

(Jefferson, 2016) and concluded that there is not a single solver winning on all problems. They reached such conclusion after analyzing the effects of a popular reformulation technique, i.e. symmetry breaking, and the impact of other modelling aspects, like the use of global constraints and auxiliary predicates. More recently, Kjellerstrand (2013) presented a comparison of 25 different CP systems. The assessment was made from various viewpoints and did not established a clear winner. As it can be concluded from the previous paragraphs, the development and comparison of CP-based scheduling models on different platforms is an interesting research topic to be addressed in the future.

Acknowledgments

The authors wish to acknowledge the financial support received from CONICET (PIP 112 20110101145), ANPCyT (PAE-PICT-00051), UNL (CAI+D 2011 50120110100473LI).

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.compchemeng.2016.04.030.

References

Baptiste, P., Le Pape, C., Nuijten, W., 2005. Constrained-Based Scheduling: Applying Constraint Programming to Scheduling Problems. Springer, New York.

Blömer, F., Günther, H., 2000. LP-based heuristics for scheduling chemical batch processes. Int. J. Prod. Res. 38, 1029–1051.

Baumann, P., Trautmann, N., 2013. A continuous-time MILP model for short-term scheduling of make-and-pack production processes, Int. J. Prod. Res. 51, 1707–1727.

Castro, P.M., Novais, A.Q., 2008. Short-term scheduling of multistage batch plants with unlimited intermediate storage. Ind. Eng. Chem. Res. 47, 6126–6139.

Castro, P.M., Grossmann, I.E., Novais, A.Q., 2006. Two new continuous-time models for the scheduling of multistage batch plants with sequence dependent changeovers. Ind. Eng. Chem. Res. 45, 6210–6226.

Castro, P.M., Harjunkoski, I., Grossmann, I.E., 2009. Optimal short-term scheduling of large-scale multistage batch plants. Ind. Eng. Chem. Res. 48, 11002–11016.

Castro, P.M., Harjunkoski, I., Grossmann, I.E., 2011. Greedy algorithm for scheduling batch plants with sequence-dependent changeovers. AIChE J. 57, 373–387.

Fernández, A.J., Hill, P.M., 2000. A comparative study of eight constraint programming languages over the Boolean and Finite Domains. Constraints 5 (3), 275–301.

Gupta, S., Karimi, I.A., 2003. An improved MILP Formulation for scheduling multiproduct: multistage batch plants. Ind. Eng. Chem. Res. 42, 2365–2380.

Harjunkoski, I., Grossmann, I.E., 2002. Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. Comput. Chem. Eng. 26, 1533–1552.

Harjunkoski, I., Maravelias, C., Bongers, P., Castro, P., Engell, S., Grossmann, I., Hooker, J., Méndez, C., Sand, G., Wassick, J., 2014. Scope for industrial applications of production scheduling models and solution methods. Comput. Chem. Eng. 62, 161–193.

Henning, G.H., 2009. Production scheduling in the process industries: current trends, emerging challenges and opportunities. In: de Brito Alves, R., Oller do Nascimento, C., Biscaia, E. (Eds.), Computer-Aided Chemical Engineering-27. Elsevier Science Ltd.

IBM ILOG, (2013). IBM ILOG CPLEX Optimization Studio. http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/.

Jefferson, G., 2016, http://www.csplib.org/ (accessed 28.03.16.).

Kjellerstrand, H., 2013, CP Solvers/Learning Constraint Programming, Workshop CP Solvers: Modeling, Applications, Integration, and Standardization, CP 2013, Uppsala, Sweden.

Kopanos, G.M., Méndez, C.A., Puigjaner, L., 2010. MIP-based decomposition strategies for large-scale scheduling problems in multiproduct multistage batch plants: a benchmark scheduling problem of the pharmaceutical industry. Eur. J. Oper. Res. 207, 644–655.

Liu, Y., Karimi, I.A., 2007a. Scheduling multistage, multiproduct batch plants with non identical parallel units and unlimited intermediate storage. Chem. Eng. Sci. 62, 1549–1566.

Liu, Y., Karimi, I.A., 2007b. Novel continuous-time formulations for scheduling multi-stage batch plants with identical parallel units. Comput. Chem. Eng. 31, 1671–1693.

Liu, Y., Karimi, I.A., 2008. Scheduling multistage batch plants with parallel units and no interstage storage. Comput. Chem. Eng. 32, 671–693.

Méndez, C.A., Cerdá, J., 2003a. Short-term scheduling of multistage batch processes subject to limited finite resources. Comput.-Aided Chem. Eng. 15-B (26), 984–989.

Méndez, C.A., Cerdá, J., 2003b. An MILP continuous-time framework for short-term scheduling of multipurpose batch processes under different operation strategies. Optim. Eng. 4, 7–22.

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. Comput. Chem. Eng. 25, 701–711.

Méndez, C.A., Cerdá, J., Grossmann, I.E., Harjunkoski, I., Fahl, M., 2006. State-of-the-art review of optimization methods for short-term scheduling of batch processes. Comput. Chem. Eng. 30, 913–946.

Mancini, T., Micaletto, D., Patrizi, F., Cadoli, M., 2008. Evaluating ASP and commercial solvers on the CSPLib. Constraints 13, 407–436.

Maravelias, C.T., Grossmann, I.E., 2004. A hybrid MILP/CP decomposition approach for the continuous time scheduling of multipurpose batch plants. Comput. Chem. Eng. 28, 1921–1949.

Marchetti, P.A., Cerdá, J., 2009a. A general resource-constrained scheduling framework for multistage batch facilities with sequence-dependent changeovers. Comput. Chem. Eng. 33, 871–886.

Marchetti, P.A., Cerdá, J., 2009b. An approximate mathematical framework for resource-constrained multistage batch scheduling. Chem. Eng. Sci. 64, 2733–2748.

Marchetti, P.A., Méndez, C.A., Cerdá, J., 2012. Simultaneous lot sizing and scheduling of multistage batch processes handling multiple orders per product. Ind. Eng. Chem. Res. 51, 5762–5780.

Novara, F., Novas, M., Henning, G., (2013). A comprehensive CP approach for the scheduling of resource-constrained multiproduct multistage batch plants. In: Computer-Aided Chemical Engineering 32, Andrzej Kraslawski and Ilkka Turunen (Ed.), 589 −594. Elsevier.

Pinto, J.M., Grossmann, I.E., 1995. A continuous time mixed integer linear programming model for short-term scheduling of multistage batch plants. Ind. Eng. Chem. Res. 34, 3037–3051.

Prasad, P., Maravelias, C.T., 2008. Batch selection: assignment and sequencing in multi-stage multi-product processes. Comput. Chem. Eng. 32, 1106–1119.

Schoppmeyer, C., Subbiah, S., Engell, S., 2012. Modeling and solving batch scheduling problems with various storage policies and operational policies using timed automata. In: Karimi, I.A., Srinivasan, Rajagopalan (Eds.), Proceedings of the 11th International Symposium on Process Systems Engineering (PSE). 15-19 July, Singapore, pp. 635–645.

Shcherbina, O.A., Neumaier, A., Sam-Haroud, D., Vu, X.-H., Nguyen, T.-V., (2003) Benchmarking global optimization and constraint satisfaction codes, in Proc. of COCOS 2002, Lecture Notes in Computer Science Vol 2861, pp. 211–222.

Sundaramoorthy, A., Maravelias, C.T., 2008a. Simultaneous batching and scheduling in multistage multiproduct processes. Ind. Eng. Chem. Res. 47, 1546–1555.

Sundaramoorthy, A., Maravelias, C.T., 2008b. Modeling of storage in batching and scheduling of multistage processes. Ind. Eng. Chem. Res. 47, 6648–6660.

Sundaramoorthy, A., Maravelias, C.T., Prasad, P., 2009. Scheduling of multistage batch processes under utility constraints. Ind. Eng. Chem. Res. 48, 6050–6058.

Velez, S., Maravelias, C.T., 2013. Multiple and nonuniform time grids in discrete-Time MIP models for chemical production scheduling. Comput. Chem. Eng. 53, 70–85.

Wallace, M., Schimpf, J., Shen, K., Harvey, W., 2004. On benchmarking constraint logic programming platforms. Constraints 9 (1), 5–34.

Yang, F., Liang, T., Song, W., Li, Q., 2013. A supervised self-organizing approach for large-scale multistage batch scheduling problems. In: 10th IEEE International Conference on Control and Automation (ICCA), 170–175, IEEE.

Zeballos, L.J., Novas, J.M., Henning, G.P., 2011. A CP formulation for scheduling multiproduct multistage batch plants. Comput. Chem. Eng. 35, 2973–2989.