Computers and Chemical Engineering 33 (2009) 1511–1528

Contents lists available at ScienceDirect

Computers and Chemical Engineering

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

A novel network-based continuous-time representation for process scheduling: Part I. Main concepts and mathematical formulation

Diego M. Giménez [a], Gabriela P. Henning [a], Christos T. Maravelias [b,*]

[a] INTEC (Universidad Nacional del Litoral – CONICET), Güemes 3450, S3000GLN Santa Fe, Argentina
[b] Department of Chemical and Biological Engineering, University of Wisconsin – Madison, 1415 Engineering Drive, Madison, WI 53706-1691, USA

ARTICLE INFO

Article history:
Received 13 February 2008
Received in revised form 13 February 2009
Accepted 10 March 2009
Available online 27 March 2009

Keywords:
Process scheduling
Network-based continuous-time representation
Mixed-integer linear programming

ABSTRACT

A novel network-based framework for the short-term scheduling of multi-purpose batch processes is presented. The novelty of the proposed approach lies in five key concepts. First, it is based on a new continuous-time representation that does not require tasks to start (end) exactly at a time point; thus reducing the number of time points needed to represent a solution. Second, processing units are modeled as being in different activity states to allow storage of input/output materials. Third, time variables for "idle" and "storage" periods of a unit are introduced to enable the matching between tasks and time points without big-M constraints. Fourth, material transfer variables are introduced to explicitly account for unit connectivity. Fifth, inventory variables for storage in processing units are incorporated to model non-simultaneous and partial material transfers. The proposed representation leads to MILP formulations which address limitations of existing scheduling methods.

1. Introduction

The paper addresses the short-term scheduling of general (multi-purpose) batch facilities. Multi-purpose facilities are structurally complex because (a) a processing task can consume and/or produce multiple chemicals, called input and output materials, respectively; (b) a material produced by a single task can be consumed by more than one successor task (batch splitting), and materials produced by more than one predecessor task can be consumed by a single task (batch mixing); (c) processing units are shared among competing tasks (multi-purpose units); (d) storage vessels (or tanks) can contain different materials which compete for them; and (e) limited utilities are required by multiple tasks. An essential characteristic of such plants is that products do not have a unique processing route. Therefore, the development of effective methods for the scheduling of multi-purpose facilities is clearly a challenging task.

Most existing approaches rely on network-based representations. For example, under the state-task network (STN) representation (Kondili, Pantelides, & Sargent, 1993) a task-node is linked to a state-node via an arc if the state (material) is consumed or produced by the task. Alternatively, if units, states (materials) and utilities are viewed as resources consumed/produced by tasks, then task-nodes are linked to resource-nodes, and the representation is referred to as resource-task network (RTN) (Pantelides, 1994). Although rather general, existing network-based approaches are based on assumptions that can potentially exclude a number of feasible schedules, thus often leading to suboptimal solutions.

The goal of this paper is the development of a new network-based continuous-time representation that overcomes these shortcomings. The paper is organized as follows. In Section 2, we review concepts about multi-purpose facilities and existing approaches for process scheduling, pointing out their limitations. In Section 3, we introduce the main concepts of the proposed representation, and in Section 4 we present a novel MILP formulation for the short-term scheduling of multi-purpose batch plants. Finally, in Section 5 we illustrate the representation capabilities of our approach via three example problems.

2. Background

2.1. Multi-purpose facilities

Multi-purpose facilities consist of a set of interconnected processing units where different types of tasks can be carried out, and storage vessels (or tanks) in which materials can be stored. Tasks transform a set of input materials into a set of output ones, where the concepts of input and output materials are relative since a given product is considered as an output material in relation to tasks that produce it and as an input one with regard to tasks that consume it. Moreover, a material is defined by its composition, phase, pres-

* Corresponding author. Tel.: +1 608 265 9026; fax: +1 608 262 5434.
E-mail address: maravelias@wisc.edu (C.T. Maravelias).

0098-1354/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compchemeng.2009.03.007

Nomenclature

Sets/indices

$\mathbf{N}/n,n'$	global time points/intervals
$\mathbf{I}/i$	tasks
$\mathbf{J}/j, j'$	processing units
$\mathbf{M}/m$	materials
$\mathbf{K}/k, k'$	storage vessels (or tanks)
$\mathbf{R}/r$	utilities (i.e. non-unary resources)
$\boldsymbol{I}_m^{\mathrm{C}}$	tasks that consume material m
$\boldsymbol{I}_m^{\mathrm{P}}$	tasks that produce material m
$\mathbf{I}^{\mathrm{CZW}}$	tasks that consume an unstable material
$\mathbf{I}^{\mathrm{PZW}}$	tasks that produce an unstable material
$\mathbf{I}_j$	tasks that unit j can perform
$\mathbf{I}_r$	tasks that require utility r
$\mathbf{J}_i$	units that can perform task i
$\mathbf{J}_j$	units connected to unit j
$\mathbf{J}_k$	units connected to storage vessel k
$\mathbf{M}^{\mathrm{S}}$	sold materials
$\mathbf{M}^{\mathrm{V}}$	materials with commercial value
$\mathbf{M}^{\mathrm{P}}$	purchased materials
$\mathbf{M}^{\mathrm{ZW}}$	unstable materials (they must be handled under a ZW policy)
$\mathbf{M}^{\mathrm{NIS}}$	materials for which no storage vessel has been assigned (NIS policy)
$\mathbf{M}^{\mathrm{FIS}}$	materials for which storage vessels of limited capacity have been assigned (FIS policy)
$\mathbf{M}_k$	materials that can be stored in storage vessel k
$\mathbf{K}^{\mathrm{D}}$	dedicated storage vessels
$\mathbf{K}^{\mathrm{S}}$	shared storage vessels
$\mathbf{K}_m$	storage vessels that can store material m
$\mathbf{K}_k$	storage vessels connected to storage vessel k
$\mathbf{K}_j$	storage vessels connected to unit j

Parameters

H	time horizon
$a_{i,j}$	fixed duration of task i in unit j
$b_{i,j}$	variable duration of task i in unit j
$\beta_{i,j}^{\mathrm{MAX}}$	maximum batch size of task i in unit j
$\beta_{i,j}^{\mathrm{MIN}}$	minimum batch size of task i in unit j
$I_{m,k}^{0}$	initial amount of material m in storage vessel k
$\gamma_{i,m}$	mass balance coefficient for the consumption (−)/production (+) of material m by task i
$\varsigma_{m,k}^{\mathrm{MAX}}$	maximum storage capacity for material m in storage vessel k ($m \in \mathbf{M}^{\mathrm{FIS}}$)
$f_{i,j,r}$	fixed amount of utility r required by task i in unit j
$g_{i,j,r}$	variable amount of utility r required by task i in unit j
ρ_r^{MAX}	maximum availability of utility r
d_m	demand of material m
π_m	price (value) of material m

Variables (binary)

$X_{i,j,n}$	1 if task i formally starts in unit j at T_n (the actual beginning occurs within time interval n)
$Y_{i,j,n}$	1 if task i formally ends in unit j at T_n (the actual end occurs within time interval $n-1$)
$S_{m,k,n}^{\mathrm{S}}$	1 if material m is stored in shared storage vessel k ($k \in \mathbf{K}^{\mathrm{S}}$) during time interval n
$S_{k,n}^{\mathrm{D}}$	1 if dedicated storage vessel k ($k \in \mathbf{K}^{\mathrm{D}}$) stores the assigned material during time interval n
$S_{j,n}^{\mathrm{I}}$	1 if in unit j input materials are stored (before the formal task beginning) during time interval n
$S_{j,n}^{\mathrm{O}}$	1 if in unit j output materials are stored (after the formal task end) during time interval n

Continuous (non-negative)

$Z_{j,n}$	1 if unit j is executing at time point n a task started in a previous time point
$E_{j,n}$	1 if unit j is formally executing a task during time interval n
$W_{j,n}$	1 if unit j is idle during time interval n
T_n	time corresponding to global point n
$\bar{T}_{j,n}^{\mathrm{LB}}$	delay of the actual beginning of a task in unit j within interval n
$\bar{T}_{j,n}^{\mathrm{EE}}$	early end of a task in unit j within interval $n-1$
$\bar{T}_{j,n}^{\mathrm{S}}$	length of the storage interval n taking place in unit j from T_n till T_{n+1}
$\bar{T}_{j,n}^{\mathrm{W}}$	length of the idle interval n taking place in unit j from T_n till T_{n+1}
$B_{i,j,n}^{\mathrm{S}}$	batch size of task i formally starting at T_n in unit j
$B_{i,j,n}^{\mathrm{P}}$	batch size of task i being processed at T_n in unit j
$B_{i,j,n}^{\mathrm{E}}$	batch size of task i formally ending at T_n in unit j
$I_{m,k,n}^{\mathrm{V}}$	amount of material m stored in storage vessel k during time interval n
$I_{m,j,n}^{I}$	amount of input material m stored in unit j during time interval n
$I_{m,j,n}^{\mathrm{O}}$	amount of output material m stored in unit j during time interval n
$F_{m,k,j,n}^{\mathrm{VU}}$	amount of material m transferred from storage vessel k to unit j at T_n
$F_{m,j,k,n}^{\mathrm{UV}}$	amount of material m transferred from unit j to storage vessel k at T_n
$F_{m,j',j,n}^{\mathrm{UU}}$	amount of material m transferred from unit j' to unit j at T_n
$F_{m,k',k,n}^{\mathrm{VV}}$	amount of material m transferred from storage vessel k' to storage vessel k at T_n
$Q_{r,n}$	total amount of utility r required during time interval n

sure and temperature; therefore, the same chemical having two distinct temperatures is treated as two different materials. On the other hand, storage vessels can be dedicated (i.e. used to store only one material) or shared (i.e. used to store more than one material) and a given material can be stored in a single vessel or in multiple ones.

Furthermore, utilities are often required by tasks (e.g. cooling water and steam are required by cooling and heating tasks, respectively), and are generally available in limited amounts. Finally, the physical connectivity between processing units and vessels can actually be limited in two ways: (a) processing units are not connected to all downstream or upstream units (these concepts are also relative, since they refer to the processing units that produce the input materials and consume the output ones, respectively), and (b) processing units are not connected to all vessels used for the storage of input and output materials. Therefore, any method for the scheduling of multi-purpose facilities should include the aforementioned five main *elements* (tasks, processing units, storage vessels, materials and utilities) as well as the *connections* (logical and physical) among them. An example of a multi-purpose facility is shown in Fig. 1.

Elements	
Processing Units: R-101, R-102, R-103	
Tasks: T1, T2, T3, T4	
Storage Units: V-101, V-102, V-103, V-104, V-105, V-106	
Materials: RM1, RM2, INT1, INT2, INT3, P1, P2	
Utilities: Hot Steam (HS), Cooling Water (CW)	
Logical Connections	
Task/Processing Unit: T1/R-101, T1/R-102, T2/R-101, T2/R-102, T3/R-103, T4/R-103	
Task/Utility: T1/HS, T2/CW, T3/HS, T4/CW	
Material/Storage Unit: RM1/V-101, RM2/V-102, INT1/V-103, INT2/-, INT3/V-104, P1/V-105, P2/V-106	
Stoichiometric Relations	
T1	0.8 RM1 + 0.2 INT1 → INT3
T2	RM2 → 0.3 INT1 + 0.7 INT2
T3	INT3 → P1
T4	0.6 INT2 + 0.4 INT3 → P2
Plant Topology	

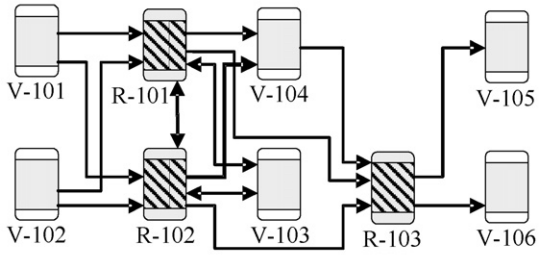

Fig. 1. Example of a simple multi-purpose facility: elements, connections and task information. Four tasks consuming two types of utilities are carried out in three processing units. Two products are obtained from two raw materials and three intermediates. Six dedicated vessels are available.

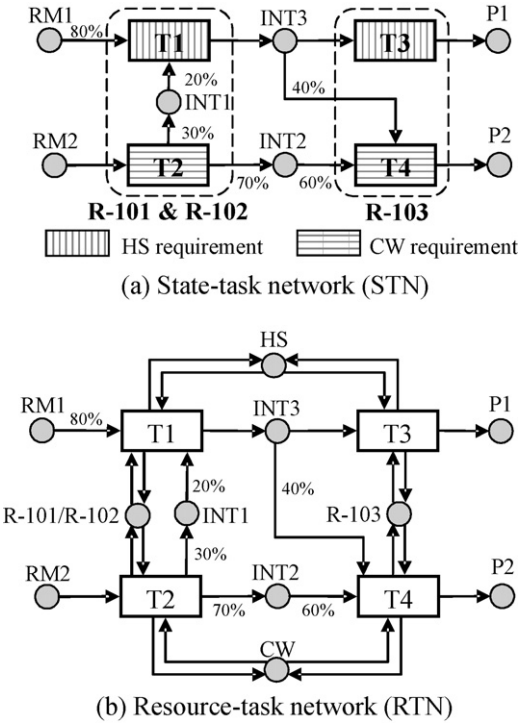

Fig. 2. STN and RTN representations of the facility in Fig. 1. (a) It includes only state- and task-nodes; the unit-task and resource-task mappings are represented with dashed lines and patterns. (b) Units and utilities are added as resource-nodes.

2.2. Network-based representations

In the STN representation (Kondili et al., 1993), the first general method for the scheduling of multi-purpose facilities, states (materials) and tasks are represented as nodes of a network, connected by arcs if a task consumes or produces a state. Processing units and utilities, however, are implicitly represented via *mappings*. The first mapping associates each processing unit with the subset of tasks that can be carried out in such a unit. The second one associates each utility with the subset of tasks consuming this utility. In turn, storage vessels are associated with states (materials): in the original formulation each state has either a dedicated tank or no tank, but the representation can be extended to model shared storage vessels. The production/consumption of states by tasks is modeled via material balance constraints that resemble flow balance constraints in networks. Finally, feasible unit-task assignments and utility consumption levels are enforced via assignment and utility constraints. The STN representation of the multi-purpose facility in Fig. 1 is shown in Fig. 2a. Note that only states and tasks are explicitly represented as circles and rectangles, respectively. Unit suitability and utility requirement information is informally represented by dashed lines and the pattern of the task nodes, respectively.

In an effort to develop a unified representation where not only task-state but also task-unit and task-utility connections are explicitly modeled, Pantelides (1994) proposed the resource-task network (RTN) representation, where states, processing units and utilities are all treated as resources consumed (produced) by a task at its beginning (end). Thus, the RTN representation involves only task- and resource-nodes and the corresponding arcs (connections) represent the consumption/production of states, as well as the task-unit and task-utility mappings. Therefore, material balances, assignment and utility consumption constraints are all expressed as flow balance constraints. The RTN representation of the facility in Fig. 1 is shown in Fig. 2b, where additional resource-nodes are added to model the task-unit and task-utility mappings.

It is important to highlight here that the concept of task is central to both formulations. This is because all changes in unit assignments, inventory levels, and utility consumption levels are triggered by (the beginning and end of) tasks. Thus, a solution is feasible if and only if all constraints are satisfied simultaneously at all the time points at which tasks start/end. The choice of these relevant time points and the corresponding time intervals defines the time grid and subsequently the time representation for MILP formulations, a topic discussed in the next subsection. Another implication of this task-centric approach, however, is that changes in the state of processing units and storage vessels, due to material transfers not directly related to a task beginning/end, cannot be modeled. Therefore, several solutions to some scheduling problems cannot be reached using existing approaches.

2.3. Time representations

In terms of computational performance, the most important *characteristic* of network-based formulations is arguably the time representation, a term that refers to: (a) the manner in which the scheduling horizon is divided into time intervals, and (b) the nature of the processing time of tasks. As explained in the previous subsection, to ensure that resource constraints are satisfied over the entire scheduling horizon, the corresponding constraints are enforced at some relevant time points that comprise the time-grid. If time points are fixed (i.e. specified prior to optimization), then, the resulting time intervals are of constant (i.e. fixed) duration. If the user defines the number but not the time point locations, the length of the corresponding time intervals becomes a continuous optimization decision. Processing times, in turn, can be constant (fixed) or variable.

The approaches of Kondili et al. (1993), Shah, Pantelides, and Sargent (1993) and Pantelides (1994) considered fixed processing

times and a time grid with intervals of uniform length equal to the greatest common divisor of all processing times. The advantage of these so-called discrete-time formulations is that the number of time intervals a task spans is known, thus no matching between time points and tasks is necessary, leading to tight MILP formulations. Their major drawback is the large size of the formulation due to the large number of time points. In an effort to develop smaller MILP models, several researchers proposed approaches having a variable time grid, leading to the so-called continuous-time formulations. Continuous-time formulations can be further classified into global and unit-specific ones. In the former, the time grid is common among all units (Castro, Barbosa-Póvoa, Matos, & Novais, 2004; Giménez & Henning, 2007; Lee, Park, & Lee, 2001; Maravelias & Grossmann, 2003; Mockus & Reklaitis, 1999; Schilling & Pantelides, 1996; Sundaramoorthy & Karimi, 2005; Zhang & Sargent, 1996), whereas in the latter different time grids can be employed for each unit (Castro, Grossmann, & Novais, 2006; Giannelos & Georgiadis, 2002; Ierapetritou & Floudas, 1998; Janak, Lin, & Floudas, 2004). Unit-specific formulations can potentially lead to smaller MILP models but are not as general as global formulations due to limitations in modeling interactions among tasks involving the same materials and utility constraints. In particular, material balances and intermediate storage constraints as well as resource constraints should be written carefully to avoid infeasible schedules. Continuous-time formulations can readily handle variable processing times. However, backlog and holding costs cannot be calculated linearly, and release times and due dates are expensive to model. Therefore, most approaches adopt profit maximization and/or makespan minimization as their objective functions. Furthermore, continuous-time formulations are not as tight as their discrete-time counterparts.

Finally, Maravelias (2005) developed a mixed-time representation where the time grid is fixed but the processing times may be variable. The proposed formulation combines the advantages of discrete- and continuous-time formulations (i.e. accounts for variable processing times and handles storage and backlog cost terms linearly) but it is computationally expensive. Existing time representations are schematically shown in Fig. 3. Note that the beginning (potentially the end) of a task should coincide with a time point in all cases. Reviews on batch scheduling can be found in Reklaitis, Sunol, Rippin, and Hortascu (1996), Schwindt and Trautmann (2000), Kallrath (2002), Burkard and Hatzl (2005), and Méndez, Cerda, Grossmann, Harjunkoski, and Fahl (2006).

2.4. Material transfer and storage

In most existing STN- and RTN-based formulations it is implicitly assumed that

(a) All processing units are connected to all the vessels that are used for the storage of the corresponding input and output materials, as well as linked to all upstream/downstream processing units. Thus, material transfer between units is always possible.
(b) All input (output) materials consumed (produced) by a task are transferred simultaneously to (from) the processing unit when the task starts (ends).
(c) Stable output materials can be temporarily stored in a processing unit after a task is completed, but stable input materials cannot be temporarily stored before a task actually starts, i.e. in continuous-time representations the beginning of a task must coincide with a time point; besides, the storage of stable output materials is always bounded by the time point representing the end of the task.
(d) Material transfers are viewed as instantaneous activities with no resource requirements.

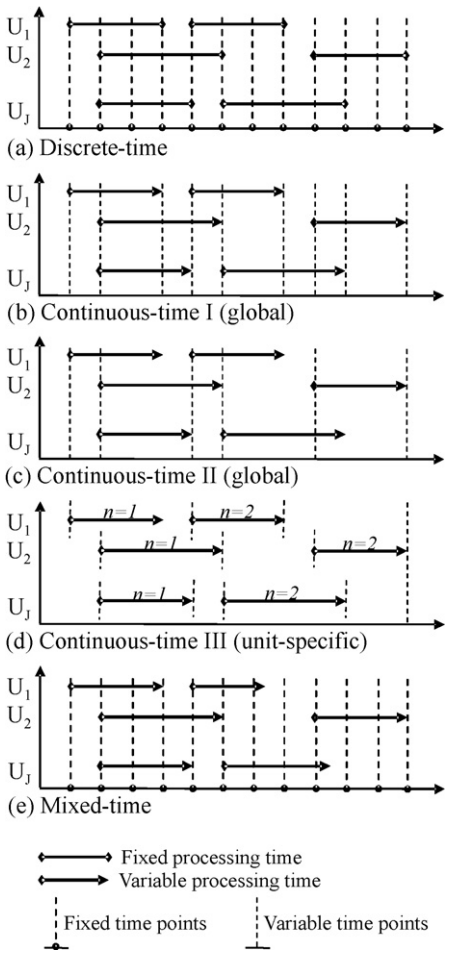

Fig. 3. Time representations for STN and RTN models. (a) Time intervals have a fixed and uniform length, tasks have constant processing times and start/end at a time point. (b) Time point locations are variable, tasks may have constant or variable processing times and must start/end at a time point. (c) Same as (b), but tasks are not forced to end at a time point. (d) Different time grids can be used for each unit. (e) Time intervals have a fixed and uniform length; tasks may have variable processing times and are not required to end at a time point.

However, these assumptions do not always hold in practice. For example, if an intermediate chemical is produced in multiple tasks, then several storage vessels may be used for its storage and each of these vessels may not be connected to all downstream processing units. Also, in many processes, input (output) materials associated with a task are not forced to be transferred simultaneously to (from) the corresponding processing unit. For instance, in recovery and purification processes the solvent can be drained earlier. Similarly, in certain chemical reactions reactants can be fed before the beginning of the task which actually occurs when the catalyst is added. In this way, the reactor can also be used as a temporary storage tank. On the other hand, a certain input (output) material may be fed (discharged) into (from) a processing unit by resorting to multiple transfers of the same material ("partial" transfer), instead of making a unique one.

It is interesting to note here that although the scheduling of multi-purpose facilities has received considerable attention, most approaches focus on the development of alternative MILP formulations for existing STN and RTN representations, but there are very few attempts to address the above limitations. The issue of unit connectivity and material transfer was addressed by Crooks (1992) and Barbosa-Póvoa and Macchietto (1994) in the context of discrete-time formulations. Specifically, Crooks (1992) proposed an extended network representation, the maximal STN (mSTN),

with explicit considerations of connections between units and transfer operations. Moreover, Crooks (1992) was the first one to bring in the notion of unit state to explicitly model the status of a processing unit. In turn, Barbosa-Póvoa and Macchietto (1994) extended this work to address the problem of multi-purpose process design, where transfer tasks can be used to model non-trivial transfer operations (e.g. operations that cannot be assumed to take place instantaneously) and multi-purpose storage vessels. Castro, Barbosa-Póvoa, and Novais (2005) tackled a simultaneous design and scheduling problem, using an RTN-based continuous-time formulation. They explicitly accounted for transfer tasks, which were assumed to be instantaneous, in order to address the synthesis of the plant pipeline network. They considered the case where several transfer tasks regarding different input materials occur at distinct time points (i.e. non-simultaneous material transfers); however, each material is associated with only one transfer. Finally, Castro and Grossmann (2005) discussed storage times in multi-stage processes. However, to our knowledge none of the existing approaches deals with the set of shortcomings due to assumptions (c) and (d). In this paper, we present a network-based global-continuous-time representation that overcomes the first three aforementioned limitations, while in Part II the proposal is generalized and the last shortcoming is also tackled.

3. Proposed approach

The proposed approach is based on the following five novel concepts. First, a time representation that does not require tasks to start (end) exactly at a time point is adopted. Second, the concept of processing unit *activity state* is introduced to enable the modeling of material storage and idling. Third, the matching between tasks and time points is achieved by means of novel time balances and without resorting to big-M constraints. Fourth, material transfers are explicitly modeled via "flow" variables; thus taking into account forbidden physical connections between processing units and storage vessels. Fifth, input (output) materials for a given task are allowed to be non-simultaneously and partially transferred to (from) a processing unit at different time points via the introduction of novel "storage" variables.

3.1. Time representation

In this paper we propose a new global-continuous-time representation for process scheduling. The time grid is defined by N global points $\{1, 2, \ldots, N\}$, with their corresponding timings $\{T_1, T_2, \ldots, T_N\}$, spanning the scheduling horizon from 0 to H and delimiting a set of $N-1$ time intervals of unknown length, as shown in Fig. 4. Time interval n begins at time point n and finishes at $n+1$. For simplicity reasons, index n is used to denote either a particular time point or interval.

The novelty of this time representation is that, unlike previous approaches, neither the actual task beginning nor the actual task end is forced to necessarily coincide with a time point, as it can be seen in Fig. 4. Therefore, a task can start after the time point that formally defines its beginning, and finish before the time point that indicates its formal end, i.e. it can *float* within a time period. The main motivation for this is the addition of an extra degree of freedom that can potentially lead to a reduction in the number of time points required to represent an optimal schedule.

The formal beginning (end) of task i is modeled via binary variable $X_{i,j,n}$ ($Y_{i,j,n}$). To accurately account for the *floating* of a task between two global points, we define two non-negative continuous "slack" variables: (a) the "late beginning" $\bar{T}^{\mathrm{LB}}_{j,n}$ that quantifies the possible delay in the actual beginning of a task executed in unit j with respect to time point n, related to its formal beginning

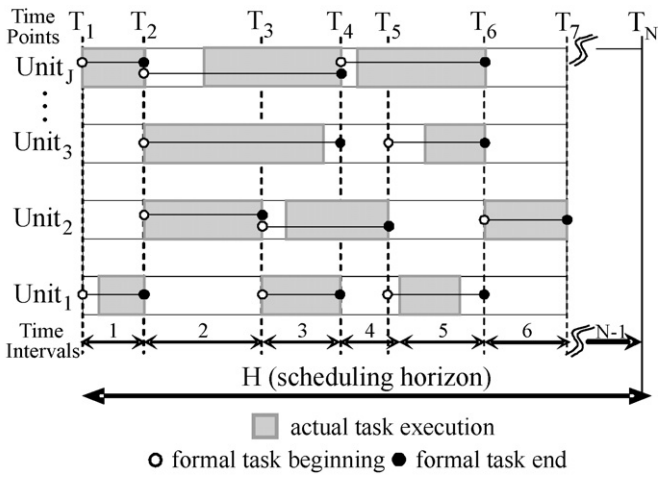

Fig. 4. Time representation: global time points and intervals.

($X_{i,j,n} = 1$); and (b) the "early end" $\bar{T}^{\mathrm{EE}}_{j,n}$ that quantifies the possible early end of a task taking place in unit j with respect to time point n, associated with its formal end ($Y_{i,j,n} = 1$). By definition, the time gaps related to slack variables $\bar{T}^{\mathrm{LB}}_{j,n}$ and $\bar{T}^{\mathrm{EE}}_{j,n}$ cannot be greater than the corresponding time interval. Specifically, if T_n is the timing of point n, then $\bar{T}^{\mathrm{LB}}_{j,n'} < T_{n'+1} - T_{n'}$ and $\bar{T}^{\mathrm{EE}}_{j,n} < T_n - T_{n-1}$ (see Fig. 5).

This new time representation allows us to tackle different situations depending on the physicochemical properties of the materials or on the kinetics of the reaction taking place. For example, stable input (output) materials could wait in the unit before (after) the task actually begins (ends). If at least one of the input (output) materials is unstable no late beginning (early end) will be allowed for the associated task.

3.2. Processing unit activity states

In the proposed approach, a processing unit can be used to carry out a task, to temporarily store input or output materials or be idle. Therefore, during each time interval a processing unit is in one of the following four states (see Fig. 6):

(a) *Execution* ($E_{j,n} = 1$): Unit j is in this state during interval n, if a task is formally being executed during interval n in unit j.
(b) *Input storage* ($S^{\mathrm{I}}_{j,n} = 1$): If input materials are being stored in unit j during interval n waiting for a task to start at time point $n' > n$ in unit j.
(c) *Output storage* ($S^{\mathrm{O}}_{j,n} = 1$): If output materials (formally produced at time point $n' \leq n$) are being stored in unit j during interval n waiting to be transferred to other processing units or storage vessels.

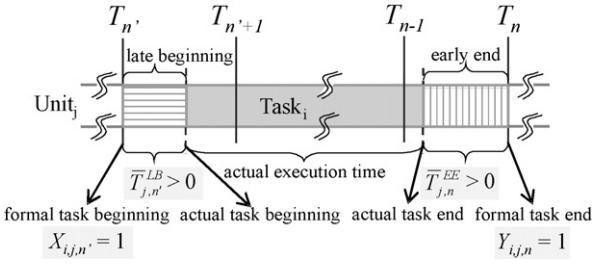

Fig. 5. Task location and associated slack variables. The formal task beginning corresponds to the time point immediately before or coinciding with the actual beginning of the task. In turn, the formal task end corresponds to the time point immediately after or coinciding with the actual end of the task.

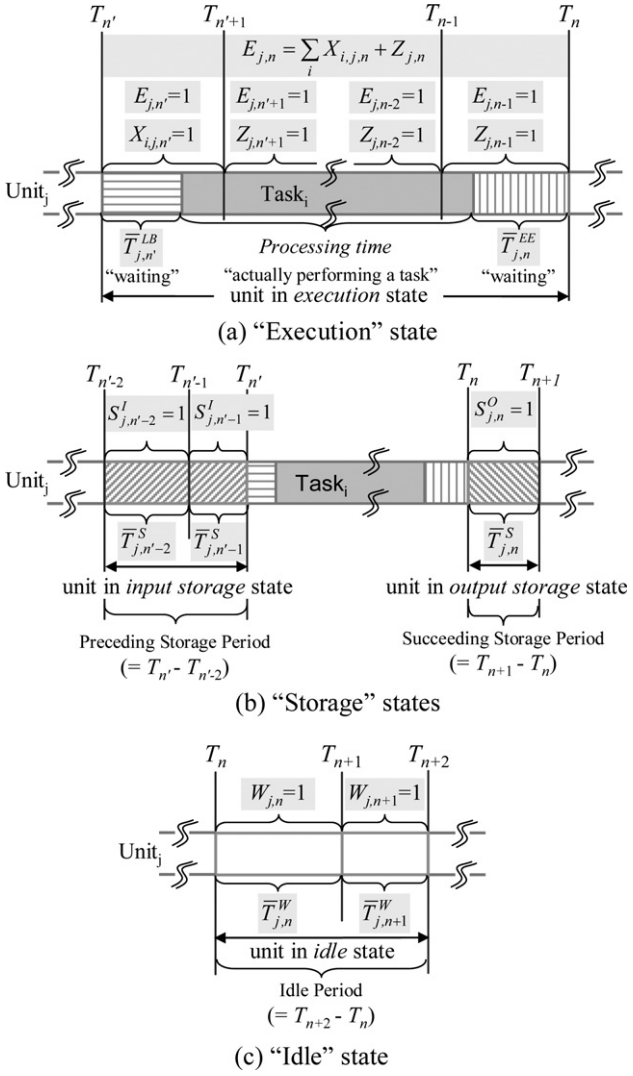

Fig. 6. Activity states of processing units. Associated binary variables are shown above the Gantt charts. Note that slack times and storage periods can exist independently of each other, i.e. a late beginning does not imply a preceding storage period (and vice versa), or an early end does not imply a succeeding storage period (or vice versa). Furthermore, there are many alternatives for the location or not of slack times and storage periods before and after the execution of a given task; e.g. if all the output materials are unstable, then a succeeding storage period and an early end are not considered.

(d) *Idle* ($W_{j,n} = 1$): If unit j during interval n is not used to carry out any processing task or to store input or output materials.

The modeling of equipment status is not new. Crooks (1992) introduced the notion of unit state to explicitly model the status of a processing unit (e.g. dirty, clean, etc.). More recently, Castro et al. (2004) employed the notion of equipment condition (the state of a unit after processing certain materials) to incorporate changeovers, which were treated as additional batch tasks.

Note that in this work the term *activity state* is employed to describe the status of a processing unit from a duty point of view, thus distinguishing whether the unit is inactive, carrying out a processing task or just acting as a temporary storage vessel. In this last case, we differentiate the storage of input materials from the storage of output ones. However, in none of the cases the state of the unit is linked to the material being held.

3.2.1. Execution state

In order to model the "execution" state of a processing unit, we use three binary variables: (a) variable $X_{i,j,n}$ to indicate the formal beginning of task i in unit j at time point n (or the actual beginning within interval n); (b) variable $Y_{i,j,n}$ to indicate the formal end of task i in unit j at time point n (or the actual end within interval $n-1$); and (c) variable $Z_{j,n}$ that indicates whether a task that formally started in a previous time point is still being executed in unit j at time point n. The combination of these binary variables allows the modeling of the "execution" state ($E_{j,n} = 1$) of the processing units. Specifically, the value of $E_{j,n}$ is obtained from the equation shown in Fig. 6a. It should be remarked that $E_{j,n} = 1$ whenever there is a task being formally executed in unit j during time interval n, in spite of the fact that in part of it the unit may be holding stable input materials (late beginning slack) and/or stable output materials (early end slack). Finally, the time a unit is in the "execution" state is equal to the processing time of the task being executed plus the time gaps that correspond to the "late beginning" and the "early end" of the task (see Fig. 6a), if they exist.

3.2.2. Input and output storage states

A unit is in the input (output) storage state if it holds stable input (output) materials before (after) a task formally begins (ends). Note that these states do not necessarily mean that all the input (output) materials are being held simultaneously, nor that the total amount consumed (produced) of a given input (output) material needs to be stored. In other words, non-simultaneous and partial transfers of materials are allowed.

Thus, unit j is in the "input storage" state during interval n ($S^{\mathrm{I}}_{j,n} = 1$) if at least one input material (of a task that will start in j later) is held in unit j during time interval n (i.e. from T_n to T_{n+1}). Note that this storage state can extend over multiple time intervals, but it should be followed by an "execution" state to avoid using processing units as pure storage vessels. The preceding storage period during which a unit is in the "input storage" state starts when the first transfer of input materials is performed (i.e. the "load" begins), and ends when the task formally begins, a moment at which all input materials are in the equipment at the correct proportions. Similarly, a unit j is in the "output storage" state ($S^{\mathrm{O}}_{j,n} = 1$) if some output materials (of a task already finished) are held in j during interval n. When an "output storage" state takes place, it starts when a task finishes (i.e. it follows an execution state), it can extend over multiple time intervals, and ends when the last transfer of an output material from the unit is performed (i.e. the "discharge" ends).

The time a unit is in either storage state is denoted by means of a unique continuous variable $\bar{T}^{\mathrm{S}}_{j,n}$. The ambiguity that may appear is resolved by identifying the specific storage state of the unit at the corresponding time interval. The variables associated with the two storage states are further illustrated in Fig. 6b. Note that when a preceding storage period concludes, it does not necessarily coincide with the actual beginning of a process task, because waiting time can be extended during the time gap of length $\bar{T}^{\mathrm{LB}}_{j,n'}$. Similar analysis can be done for the succeeding storage period.

3.2.3. Idle state

Finally, a unit is in the "idle" state ($W_{j,n} = 1$) if it is empty, i.e. it is not used to carry out a task nor to storage materials during time interval n (i.e. from T_n to T_{n+1}). The associated "idle" period $\bar{T}^{\mathrm{W}}_{j,n}$ is equal to the duration of time interval n in which the unit j is idle (see Fig. 6c).

3.3. Time balances

The explicit modeling of all possible activity states of a processing unit, and their corresponding durations, enables us to develop

novel time balances for each unit along the scheduling horizon. The underlying idea is to enforce the matching between tasks and time points without using big-M constraints. In addition to avoiding tasks overlapping, these balances lead to a tight MILP formulation. First, two time balances for each processing unit are posed at each time point; one that holds exactly before the time point and the other that must be sustained exactly after it. They are expressed as inequalities and are called *partial* balances because only a part of the scheduling horizon is considered. Second, a *global* balance is enforced for each unit. In this balance, the sum of all the time elements corresponding to the states of the processing unit along the schedule must be equal to the time horizon. In a generic form, the time balances corresponding to a certain unit j can be expressed as follows:

(a) Partial balance before time point n: $T_n \geq$ {sum of the processing times of all the already finished tasks carried out in unit j} + {sum of all the time gaps located before T_n in unit j} + {sum of all the idle and storage periods occurring before T_n in unit j}.
(b) Partial balance after time point n: $H - T_n \geq$ {sum of the processing times of all the tasks to be completely executed after T_n in unit j} + {sum of all the time gaps occurring after T_n in unit j} + {sum of all the idle and storage periods located after T_n in unit j}.
(c) Global balance: $H =$ {sum of all the processing times of tasks executed in unit j} + {sum of all the time gaps occurring in unit j} + {sum of all the idle and storage periods along the scheduling horizon in unit j}.

Although similar constraints have been proposed in the past to tighten the MILP formulation of continuous-time representations (Maravelias & Grossmann, 2003), in this work the time balance constraints are sufficient to correctly enforce the timing of the time points of the grid and the matching between time points and events. Therefore, unlike previous approaches, no additional big-M matching constraints or ordering inequalities between consecutive time points are needed. The ideas pertaining to time balances are exemplified in Fig. 7.

3.4. Material transfer

To be able to model all possible material transfers to/from a processing unit or storage vessel four "flow" variables are considered,

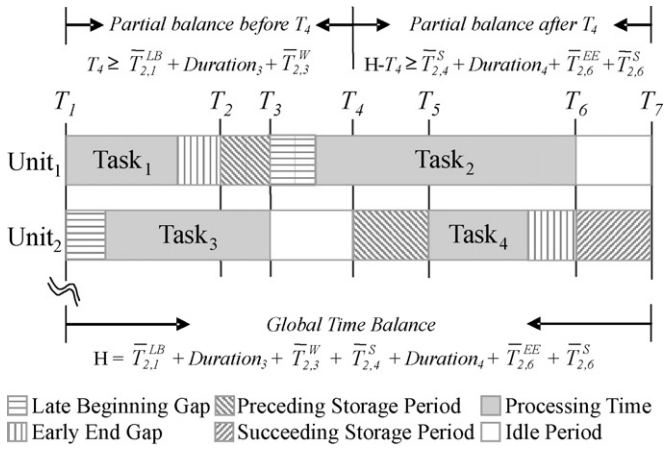

Fig. 7. Example of the time elements included in the different time balances. Unit 2 and time point 4 have been chosen to exemplify the time balances. In this case, the inequalities are satisfied as strict equalities since no task is being executed at the time point at which the partial balances are being expressed (e.g. such as the case of Task 2 in Unit 1 at T_4).

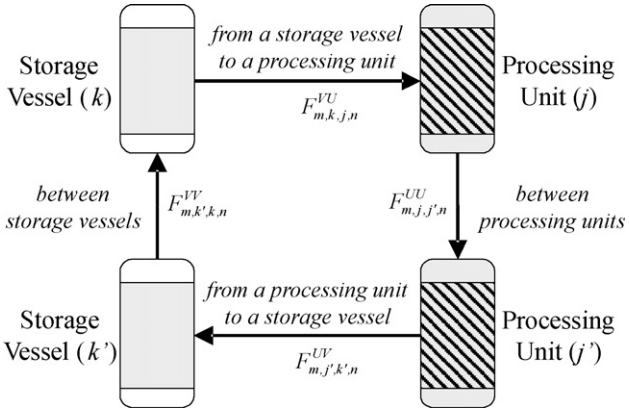

Fig. 8. Types of material transfers (flows).

where in the context of this contribution, the concept of "flow" refers to an instantaneous material transfer at a time point and not to a rate of transfer. Specifically, these non-negative continuous variables model four kinds of material transfers, as illustrated in Fig. 8.

(a) $F^{VU}_{m,k,j,n}$ = amount of material m transferred from storage vessel k to processing unit j at T_n;
(b) $F^{UV}_{m,j,k,n}$ = amount of material m transferred from processing unit$j$ to storage vessel k at T_n;
(c) $F^{UU}_{m,j',j,n}$ = amount of material m moved from processing unit j' to unit j at T_n;
(d) $F^{VV}_{m,k',k,n}$ = amount of material m moved from storage vessel k' to vessel k at T_n.

Note that the letter sequence in the superscript denotes the direction of the transfer between processing units (U) and storage vessels (V).

Using these variables, we can model non-simultaneous and partial material transfers, i.e. situations where not all the materials consumed/produced by a task are loaded/discharged in/from the processing unit at the same time or not all the consumed/produced amount of a given material is loaded/discharged by means of a unique transfer. An example of simultaneous transfer of both input and output materials is shown in Fig. 9a, while an example of non-simultaneous transfers is shown in Fig. 9b. However, there exist many possible combinations of simultaneous and non-simultaneous transfers of input and output materials. For example, a simultaneous transfer of input materials could be combined with a non-simultaneous transfer of output materials, and vice versa. The same is valid for partial transfers. Moreover, since material transfers are explicitly captured, connections between physical units can be taken into account, thus enabling a detailed representation of the process network. For example, in the simple multi-purpose facility illustrated in Fig. 1, the transfer of materials from vessel V-101 to reactor R-102 is allowed, contrary to the transfer from reactor R-102 to vessel V-103 which is not permitted. Regarding the time consumed by the transfers, it is considered insignificant; therefore, the material transfers are assumed to be instantaneous.

It is important to note that, unlike previous approaches, a transfer does not necessarily coincide with the task beginning/end. This is possible because the storage of materials in the unit decouples the material load/discharge from the task beginning/end. Moreover, input/output materials can be transferred from/to different storage vessels and processing units.

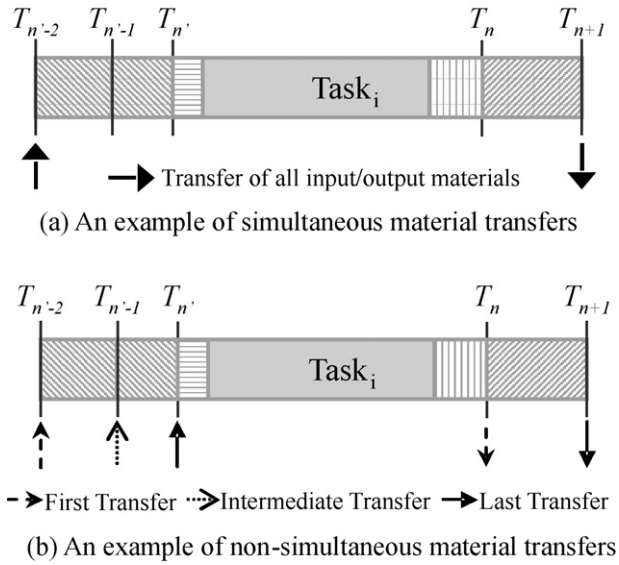

Fig. 9. Different types of material transfers to/from a processing unit. (a) All input (output) materials are simultaneously transferred to (from) the processing unit at T'_{n-2} (T_{n+1}). (b) The first transfer of input materials occurs at T'_{n-2}, an intermediate transfer takes place at T'_{n-1}, and the last one happens at T'_n; the first transfer of output materials occurs at T_n and the last one takes place at T_{n+1}.

3.5. Material storage

3.5.1. Processing units

To effectively model the storage of input and output materials in processing units two new storage variables are introduced. Variable $I^{I}_{m,j,n}$ represents the inventory of input material m in processing unit j (before a task beginning), during time interval n. Variable $I^{O}_{m,j,n}$ quantifies the inventory of output material m in processing unit j (after a task end), during interval n.

When posing the balance of input material m stored in processing unit j, the inventory level will increase any time material m is transferred to unit j and will decrease if a task taking place in unit j consumes m. Similarly, when developing the balance of an output material m' stored in processing unit j, the stored amount of m' will increase if a task taking place in unit j produces m' and will decrease any time material m' is transferred to another physical site (processing unit or storage vessel). Unlike previous approaches, tasks consume input materials which are stored in processing units, and not directly from storage vessels. Likewise, output materials produced by a task are added up to the inventory of each output material in the same unit at the time point in which the operation formally ends, requiring explicit material transfers from the unit to make them available in another physical site. These new storage concepts are illustrated in Fig. 10, where a hypothetical case with both input and output materials storage is shown.

3.5.2. Storage vessels

In this case, a storage variable $I^{S}_{m,k,n}$ denotes the inventory level of material m in storage vessel k during a time interval n. It increases any time an extra amount is transferred to such a unit and decreases any time the stored material is transferred to another physical site (processing unit or storage vessel). As in the case of processing units, binary variable $S^{D}_{k,n}$ can be used, if necessary, to define the "storage" state of dedicated vessel k during time interval n. In the case of shared storage vessels, the binary variable $S^{S}_{m,k,n}$ (representing the "material m storage" state) must be used to identify the material (m) kept in vessel k during interval n. In both cases, if a storage vessel is not in a "storage" state, then it is in the "empty" one. Moreover, "storage" and "empty" periods can be calculated by incorporating new time variables.

Continuing with the case exemplified in Fig. 10, Fig. 11 shows the inventory evolution for materials INT3 and P2 stored in vessels V-104 and V-106, respectively. Also the states of such dedicated storage vessels are illustrated.

3.6. Key issues summary

The proposed network representation is based upon the five key ideas already presented in this section:

(a) Tasks are not required to actually start (end) exactly at a time point.
(b) A processing unit can be in one of four activity states at any given time interval.
(c) Time matching between tasks and the global time grid is enforced via a set of three time balance constraints.
(d) Material transfers are explicitly modeled via "flow" variables.
(e) Input (output) materials associated with a given task are allowed to be temporally stored in a processing unit via the introduction of novel "storage" variables.

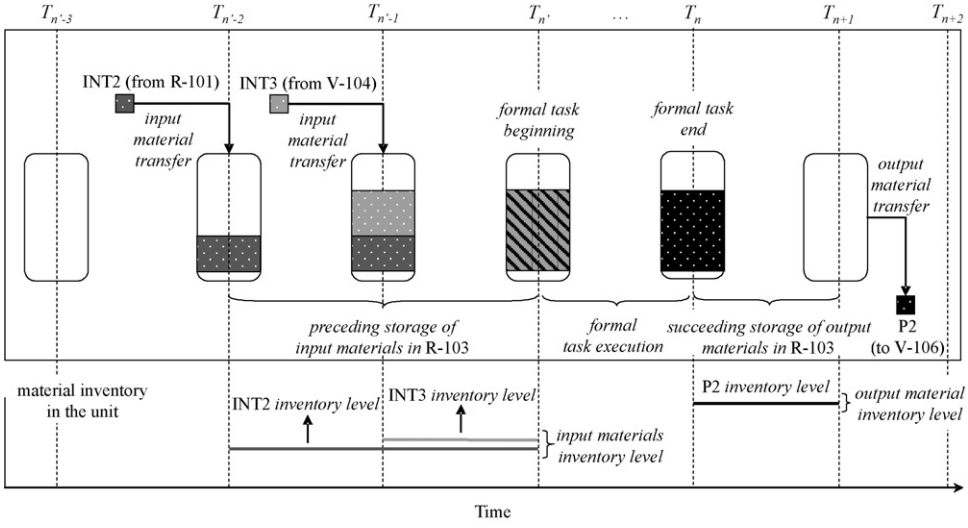

Fig. 10. Inventory evolution of each input (output) material temporarily stored in reactor R-103 of the motivating example in Fig. 1 before (after) reaction T_4 takes place.

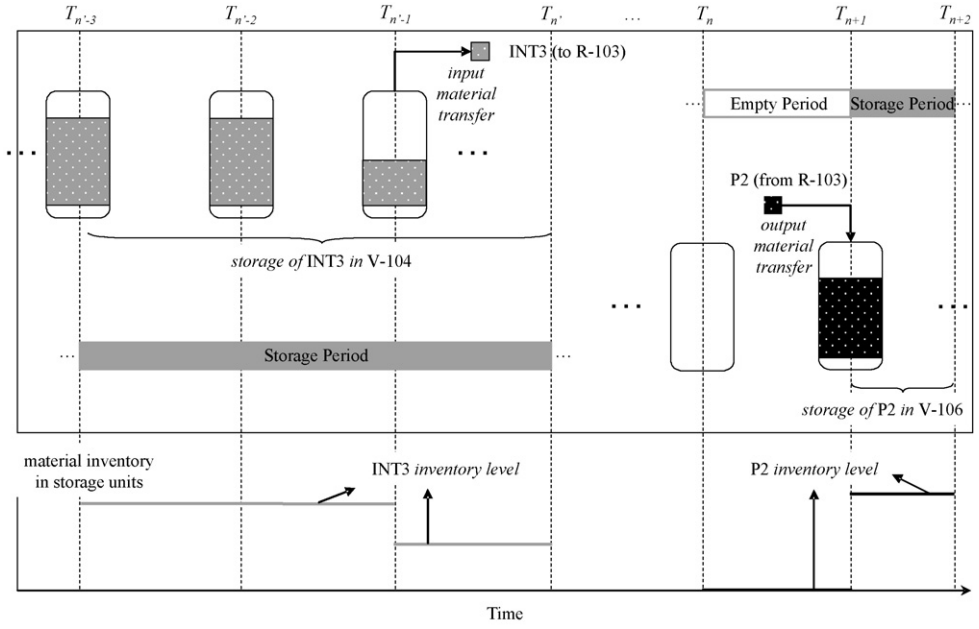

Fig. 11. Evolution of the material inventory in two storage vessels.

Undoubtedly, the flexibility associated with this novel representation is considerably higher than in previous approaches, and the actual behavior of certain processes can be more accurately represented. Finally, Fig. 12 shows a schematic view of the different elements that participate in the Gantt chart of the example under study. In the same figure, the three different concepts related to the time dimension are depicted: (i) time interval, which is delimited by two consecutive time points; (ii) time period, comprising a number of consecutive time intervals; and (iii) time gap, that is an extreme portion of a time interval.

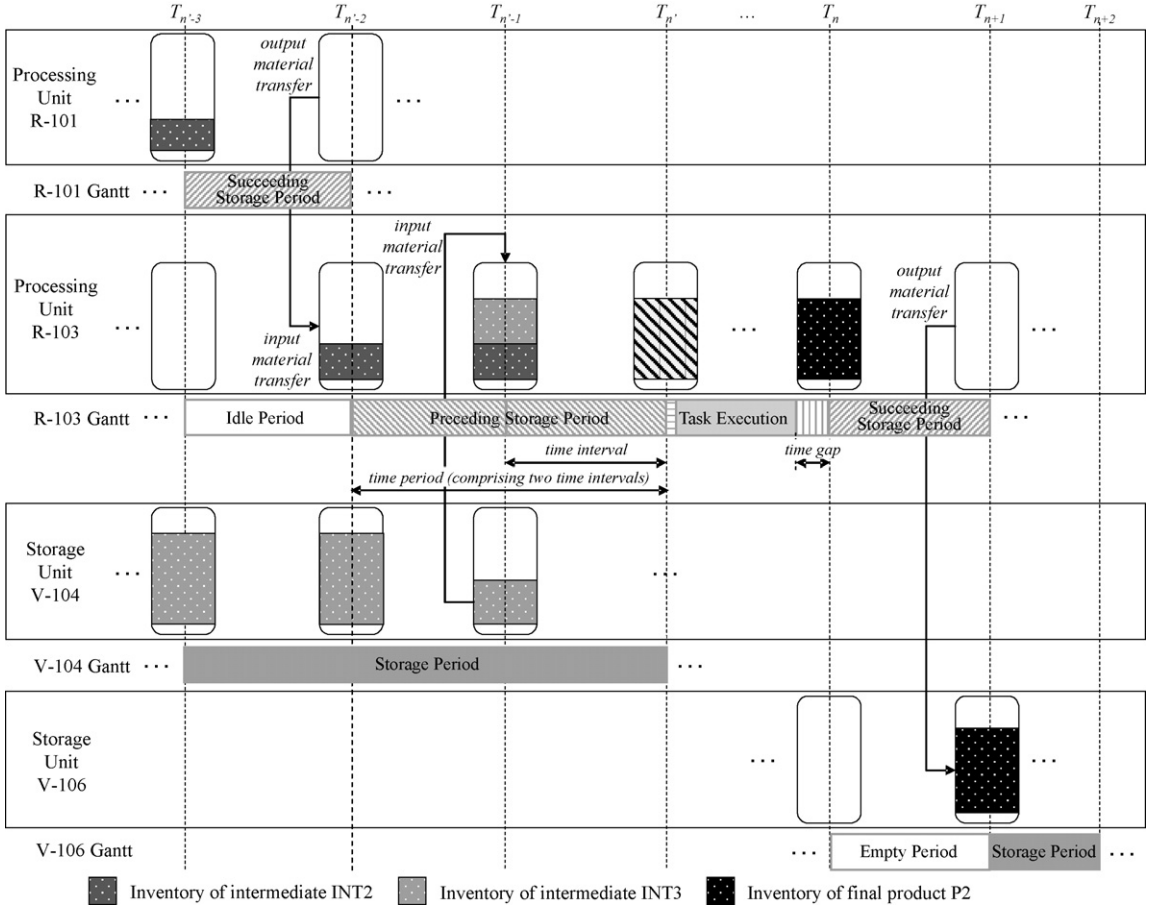

Fig. 12. Schematic view of the elements participating in the Gantt chart of the example under study.

4. Mathematical formulation

In this section, we present the MILP formulation for the scheduling of multi-purpose batch processes. Before proceeding with the constraints of the model, we summarize the different types of variables:

(a) *Execution variables (binary)*: Variable $X_{i,j,n}/Y_{i,j,n}$ is equal to one if task i formally begins/ends in unit j at time point n; also, variable $Z_{j,n}$ is activated if a task that started earlier is being executed in unit j at time point n.
(b) *State variables (binary)*: Execution $E_{j,n}$, input/output storage $S^{\mathrm{I}}_{j,n}/S^{\mathrm{O}}_{j,n}$, and idle $W_{j,n}$ variables are activated when processing unit j is in the corresponding activity state during interval n; storage $S^{\mathrm{D}}_{k,n}/S^{\mathrm{S}}_{m,k,n}$ variables are activated when storage unit k is in the corresponding state during interval n.
(c) *Time grid variable (non-negative continuous)*: The timing (i.e. location along the scheduling horizon) of time point n is denoted by T_n.
(d) *Time variables (non-negative continuous)*: Slack variables $\bar{T}^{\mathrm{LB}}_{j,n}$ and $\bar{T}^{\mathrm{EE}}_{j,n}$ are used to quantify the late beginning and early end of a task executed in unit j in relation to time point n, while the length of time interval n during which unit j is in the (input or output) storage or idle state is represented by variables $\bar{T}^{\mathrm{S}}_{j,n}$ and $\bar{T}^{\mathrm{W}}_{j,n}$, respectively.
(e) *Flow variables (non-negative continuous)*: Variables $F^{\mathrm{VU}}_{m,k,j,n}$, $F^{\mathrm{UV}}_{m,j,k,n}$, $F^{\mathrm{UU}}_{m,j',j,n}$, and $F^{\mathrm{VV}}_{m,k',k,n}$ are used to represent material transfers between the different sites of the facility (see Section 3.4).
(f) *Inventory variables (non-negative continuous)*: The amount of input/output material m stored in processing unit j during time interval n is denoted by $I^{\mathrm{UI}}_{m,j,n}/I^{\mathrm{UO}}_{m,j,n}$; the corresponding amount stored in a storage vessel is denoted by $I^{\mathrm{V}}_{m,k,n}$.
(g) *Batch-size variables (non-negative continuous)*: Variables $B^{\mathrm{S}}_{i,j,n}$, $B^{\mathrm{P}}_{i,j,n}$ and $B^{\mathrm{E}}_{i,j,n}$ are introduced to denote the batch size of task i that formally starts, is being executed and formally ends, respectively, in unit j at time point n.

4.1. Execution-state constraints

Eq. (1) forces each unit to be in only one activity state at each time interval. In turn, Eq. (2) defines the state variable $E_{j,n}$ in terms of variables $Z_{j,n}$ and $X_{i,j,n}$, as explained in Section 3.2.1.

$$E_{j,n} + W_{j,n} + S^{I}_{j,n} + S^{\mathrm{O}}_{j,n} = 1, \quad \forall j, n < N \tag{1}$$

$$E_{j,n} = Z_{j,n} + \sum_{i \in \mathbf{I}_j} X_{i,j,n}, \quad \forall j, n < N \tag{2}$$

where $\mathbf{I}_j$ is the subset of tasks that can be performed in unit j.

The expression in Eq. (3) relates task beginnings and ends with variable $Z_{j,n}$ in order to identify those time points in which unit j is executing a task which started in a previous time point. Since variable $Z_{j,n}$ assumes binary values, Eq. (3) implies that task i can start its execution in unit j at time point n only if there is no other task being performed or beginning in the same unit at the same time point. Besides, a task can finish its processing only if it began earlier:

$$Z_{j,n} = Z_{j,n-1} + \sum_{i \in \mathbf{I}_j} X_{i,j,n-1} - \sum_{i \in \mathbf{I}_j} Y_{i,j,n}, \quad \forall j, n > 1 \tag{3}$$

4.2. Slack time constraints

With respect to slack times, inequality (4) forces $\bar{T}^{\mathrm{LB}}_{j,n}$ to be equal to zero if (i) no task starts at T_n, or (ii) a task starts at T_n, but at least one of its input materials requires to operate under a zero-wait policy. Similarly, $\bar{T}^{\mathrm{EE}}_{j,n}$ must be equal to zero if (i) no task ends at T_n, or (ii) a task ends at such time point, but at least one of its output materials requires to operate under a zero-wait policy (Eq. (5)).

$$\bar{T}^{\mathrm{LB}}_{j,n} \le H \sum_{i \in (\mathbf{I}_j \setminus \mathbf{I}^{\mathrm{CZW}})} X_{i,j,n}, \quad \forall j, n < N \tag{4}$$

$$\bar{T}^{\mathrm{EE}}_{j,n} \le H \sum_{i \in (\mathbf{I}_j \setminus \mathbf{I}^{\mathrm{PZW}})} Y_{i,j,n}, \quad \forall j, n > 1 \tag{5}$$

where $\mathbf{I}^{\mathrm{CZW}}/\mathbf{I}^{\mathrm{PZW}}$ is the subset of tasks consuming/producing a zero-wait material.

4.3. Storage and idle periods constraints

Inequalities (6)–(8) relate variables representing storage and idle intervals to their corresponding state variables. Expression (6) makes the length of the storage interval $\bar{T}^{\mathrm{S}}_{j,n}$ equal to zero if unit j is not under any of the two possible storage states during time interval n. The same relation is established between $\bar{T}^{\mathrm{W}}_{j,n}$ and $W_{j,n}$ by means of constraint (7).

$$\bar{T}^{\mathrm{S}}_{j,n} \le H(S^{I}_{j,n} + S^{\mathrm{O}}_{j,n}), \quad \forall j, n < N \tag{6}$$

$$T^{\mathrm{W}}_{j,n} \le H(W_{j,n}), \quad \forall j, n < N \tag{7}$$

Finally, the inequalities introduced in expression (8) fix the lengths of both storage and idle intervals.

$$T_{n+1} - T_n - H(1 - S^{I}_{j,n} - S^{\mathrm{O}}_{j,n} - W_{j,n}) \le \bar{T}^{\mathrm{S}}_{j,n} + \bar{T}^{\mathrm{W}}_{j,n} \le T_{n+1} - T_n, \quad \forall j, n < N \tag{8}$$

Execution-state and timing constraints as well as their corresponding variables are clarified via a simple example in Appendix A.

4.4. Time balance constraints

For a given unit j, inequality (9) expresses the partial balance constraint for those time elements (i.e. processing times, time gaps as well as storage and idle periods) taking place before time point n:

$$T_n \ge \sum_{1<n' \le n} \sum_{i \in \mathbf{I}_j} (a_{i,j} Y_{i,j,n'} + b_{i,j} B^{\mathrm{E}}_{i,j,n'}) + \sum_{1<n' \le n} \bar{T}^{\mathrm{EE}}_{j,n'} + \sum_{n'<n} (\bar{T}^{\mathrm{LB}}_{j,n'} + \bar{T}^{\mathrm{S}}_{j,n'} + \bar{T}^{\mathrm{W}}_{j,n'}), \quad \forall j, n > 1 \tag{9}$$

where $a_{i,j}$ and $b_{i,j}$ are constants used to determine the duration of task i carried out in unit j. Similarly, expression (10) models the balance of those time elements that occur after T_n in unit j:

$$H - T_n \ge \sum_{n \le n' < N} \sum_{i \in \mathbf{I}_j} (a_{i,j} X_{i,j,n'} + b_{i,j} B^{\mathrm{S}}_{i,j,n'}) + \sum_{n'>n} \bar{T}^{\mathrm{EE}}_{j,n'} + \sum_{n \le n' < N} (\bar{T}^{\mathrm{LB}}_{j,n'} + \bar{T}^{\mathrm{S}}_{j,n'} + \bar{T}^{\mathrm{W}}_{j,n'}), \quad \forall j, n < N \tag{10}$$

Constraint (11) represents the global time balance for each unit:

$$\sum_{n>1}\bar{T}^{\mathrm{EE}}_{j,n}+\sum_{n<N}(\bar{T}^{\mathrm{LB}}_{j,n}+\bar{T}^{\mathrm{S}}_{j,n}+\bar{T}^{\mathrm{W}}_{j,n})+\sum_{n>1}\sum_{i\in\boldsymbol{I}_j}(a_{i,j}Y_{i,j,n}+b_{i,j}B^{\mathrm{E}}_{i,j,n})=H,\quad \forall j \tag{11}$$

Constraints (8)–(11) enforce timing and matching between tasks and time points without resorting to big-M constraints.

4.5. Batch size constraints

The expressions in constraints (12)–(14) are used to bound the size of a batch of task i executed in unit j. Eq. (15) guarantees batch size consistency along the different contiguous time intervals associated with the formal execution of the same batch.

$$\beta^{\mathrm{MIN}}_{i,j}Y_{i,j,n}\le B^{\mathrm{E}}_{i,j,n}\le\beta^{\mathrm{MAX}}_{i,j}Y_{i,j,n},\quad \forall i,j\in\boldsymbol{J}_i,\quad n>1 \tag{12}$$

$$B^{\mathrm{S}}_{i,j,n}\le\beta^{\mathrm{MAX}}_{i,j}X_{i,j,n},\quad \forall i,j\in\boldsymbol{J}_i,\quad n<N \tag{13}$$

$$\sum_{i\in\boldsymbol{I}_j}B^{\mathrm{P}}_{i,j,n}\le\max_{i\in\boldsymbol{I}_j}\{\beta^{\mathrm{MAX}}_{i,j}\}Z_{j,n},\quad \forall j,n \tag{14}$$

$$B^{\mathrm{S}}_{i,j,n}+B^{\mathrm{P}}_{i,j,n}=B^{\mathrm{P}}_{i,j,n+1}+B^{E}_{i,j,n+1},\quad \forall i,j\in\boldsymbol{J}_i,\quad n<N \tag{15}$$

4.6. Material storage in storage vessels

Eq. (16) represents the material balance for material m held in storage vessel k at time point n.

$$I^{\mathrm{S}}_{m,k,n}=I^{\mathrm{S}}_{m,k,n-1}-\sum_{j\in\boldsymbol{J}_k}F^{\mathrm{VU}}_{m,k,j,n}-\sum_{k'\in\boldsymbol{K}_k}F^{\mathrm{VV}}_{m,k,k',n}+\sum_{j\in\boldsymbol{J}_k}F^{\mathrm{UV}}_{m,j,k,n}+\sum_{k'\in\boldsymbol{K}_k}F^{\mathrm{VV}}_{m,k',k,n},\quad \forall m\notin(\boldsymbol{M}^{\mathrm{NIS}}\cup\boldsymbol{M}^{\mathrm{ZW}}),\quad k\in\boldsymbol{K}_m,n \tag{16}$$

where $\mathbf{M}^{\mathrm{NIS}}/\mathbf{M}^{\mathrm{ZW}}$ is the subset of materials with no-intermediate/zero-wait storage policy, $\mathbf{K}_m$ is the subset of storage vessels where material m can be stored in, and $I^{\mathrm{S}}_{m,k,0}=I^{0}_{m,k}$ is the given initial inventory of material m in storage vessel k.

Constraints represented by expression (17) restrain the amount of material to be stored in a dedicated storage vessel, where variable $S^{\mathrm{D}}_{k,n}$ can be omitted as discussed in Section 3.5.2. The ones in Eq. (18) account for shared storage vessels.

$$I^{\mathrm{S}}_{m,k,n}\le\varsigma^{\mathrm{MAX}}_{m,k}S^{\mathrm{D}}_{k,n},\quad \forall m\in(\boldsymbol{M}^{\mathrm{FIS}}\cap\boldsymbol{M}_k),\quad k\in\boldsymbol{K}^{\mathrm{D}},\quad n>1 \tag{17}$$

$$I^{\mathrm{S}}_{m,k,n}\le\varsigma^{\mathrm{MAX}}_{m,k}S^{\mathrm{S}}_{m,k,n},\quad \forall m\in(\boldsymbol{M}^{\mathrm{FIS}}\cap\boldsymbol{M}_k),\quad k\in\boldsymbol{K}^{\mathrm{S}},\quad n>1 \tag{18}$$

where $\mathbf{M}^{\mathrm{FIS}}$ is the subset of finite-intermediate-storage materials, $\mathbf{K}^{\mathrm{D}}/\mathbf{K}^{\mathrm{S}}$ is the subset of dedicated/shared storage vessels, $\mathbf{M}_k$ is the subset of materials that can be stored in storage vessel k, and $\varsigma^{\mathrm{MAX}}_{m,k}$ is the capacity of storage vessel k to store material m.

Inequality (19) ensures that at most one material is stored in a given shared vessel at any time interval:

$$\sum_{m\in\boldsymbol{M}_k}S^{\mathrm{S}}_{m,k,n}\le 1,\quad \forall k\in\boldsymbol{K}^{\mathrm{S}},\quad n>1 \tag{19}$$

4.7. Material storage in processing units

The amount of input material m stored in processing unit j during time interval n, prior to its consumption by a task, is given by Eq. (20). It increases by the transfer of m from physically connected storage vessels and processing units, respectively, and decreases by the amount consumed by a batch that formally starts at T_n:

$$I^{I}_{m,j,n}=I^{I}_{m,j,n-1}+\sum_{k\in(\boldsymbol{K}_j\cap\boldsymbol{K}_m)}F^{\mathrm{VU}}_{m,k,j,n}+\sum_{j'\in\boldsymbol{J}_j}F^{\mathrm{UU}}_{m,j',j,n}+\sum_{i\in(\boldsymbol{I}_j\cap\boldsymbol{I}^{\mathrm{C}}_m)}\gamma_{i,m}B^{\mathrm{S}}_{i,j,n},\quad \forall m,j,n \tag{20}$$

where $\mathbf{J}_j/\mathbf{K}_j$ is the subset of processing units/storage vessels physically connected to j, $\boldsymbol{I}^{\mathrm{C}}_m$ is the subset of tasks that consume material m, and $\gamma_{i,m}$ is the mass balance coefficient of material m in task i (negative if consumed).

Further, input materials can be maintained in unit j during time interval n only if the unit remains in the "input storage" state during such interval.

$$\sum_{m\in\boldsymbol{M}}I^{I}_{m,j,n}\le\max_i\{\beta^{\mathrm{MAX}}_{i,j}\}S^{I}_{j,n},\quad \forall j,n<N \tag{21}$$

On the other hand, the amount of an output material m stored in a processing unit j during time interval n, after being generated by an already finished task, is given by

$$I^{\mathrm{O}}_{m,j,n}=I^{\mathrm{O}}_{m,j,n-1}+\sum_{i\in(\boldsymbol{I}_j\cap\boldsymbol{I}^{\mathrm{P}}_m)}\gamma_{i,m}B^{\mathrm{E}}_{i,j,n}-\sum_{k\in(\boldsymbol{K}_j\cap\boldsymbol{K}_m)}F^{\mathrm{UV}}_{m,j,k,n}-\sum_{j'\in\boldsymbol{J}_j}F^{\mathrm{UU}}_{m,j,j',n},\quad \forall m,j,n>1 \tag{22}$$

where $\boldsymbol{I}^{\mathrm{P}}_m$ is the subset of tasks producing material m.

In turn, output materials are to be maintained in unit j during time interval n only if the unit remains in the "output storage" state during such interval:

$$\sum_{m\in\boldsymbol{M}}I^{\mathrm{O}}_{m,j,n}\le\max_i\{\beta^{\mathrm{MAX}}_{i,j}\}S^{\mathrm{O}}_{j,n},\quad \forall j,n<N \tag{23}$$

4.8. Utility constraints

The total amount of utility r consumed at each time interval is calculated and bounded by the maximum resource availability:

$$Q_{r,n}=Q_{r,n-1}+\sum_{i\in\boldsymbol{I}_r}\sum_{j\in\boldsymbol{J}_i}[f_{i,j,r}(X_{i,j,n}-Y_{i,j,n})+g_{i,j,r}(B^{\mathrm{S}}_{i,j,n}-B^{\mathrm{E}}_{i,j,n})],\quad \forall r,n \tag{24}$$

$$Q_{r,n}\le\rho^{\mathrm{MAX}}_r,\quad \forall r,n \tag{25}$$

where $\mathbf{I}_r$ is the subset of tasks requiring utility r, $f_{i,j,r}/g_{i,j,r}$ is the fixed/proportional constant for the consumption of utility r by task i when carried out in unit j, and ρ^{MAX}_r is the availability of utility r.

4.9. Additional constraints

Expression (26) ensures that the demand d_m for product $m\in\mathbf{M}^{\mathrm{S}}$ to be delivered at the end of the scheduling horizon is satisfied:

$$\sum_{k\in\boldsymbol{K}_m}I^{\mathrm{S}}_{m,k,n}\ge d_m,\quad \forall m\in\boldsymbol{M}^{\mathrm{S}},\quad n=N \tag{26}$$

If we assume that no tasks start and/or end outside the scheduling horizon, then variables $Z_{j,n}$ can be fixed to be zero for $n=1$ and $n=\mathrm{N}$. However, this assumption can be relaxed if necessary. Further, we do not allow the storage of output materials at the beginning of the horizon by setting $S^{\mathrm{O}}_{j,1}=0,\quad \forall j$. Also, we do not allow storage of unstable input and output materials by fixing $I^{I}_{m,j,n}=0$ and $I^{\mathrm{O}}_{m,j,n}=0,\quad \forall m\in M^{\mathrm{ZW}},j,n$. Finally, if necessary, it is ensured that

each unit will be left empty at the end of the scheduling horizon by adding:

$$\sum_{m\in\mathbf{M}} I^{I}_{m,j,n} + \sum_{m\in\mathbf{M}} I^{O}_{m,j,n} = 0, \quad \forall j, n = N \tag{27}$$

4.10. Objective function

The proposed model can be used to address problems with different objective functions. For the total profit (TP) maximization, Eq. (28) should be incorporated into the formulation:

$$TP = \sum_{m\in\mathbf{M}^{S}}\sum_{k\in\mathbf{K}_m} \pi_m I^{S}_{m,k,n} + \sum_{m\in\mathbf{M}^{V}}\sum_{k\in\mathbf{K}_m} \pi_m (I^{S}_{m,k,n} - I^{0}_{m,k}) - \sum_{m\in\mathbf{M}^{P}}\sum_{k\in\mathbf{K}_m} \pi_m (I^{0}_{m,k} - I^{S}_{m,k,n}), \quad n = N \tag{28}$$

where π_m is the price of material m, $\mathbf{M}^{S}$ is the subset of final products, $\mathbf{M}^{V}$ is the subset of intermediate materials with commercial value, and $\mathbf{M}^{P}$ is the subset of purchased materials. In this case, the objective function can be written as $z = \max TP$.

If the goal is to find the minimum makespan (MK) schedule that satisfies the required demand pattern, then parameter H (which is now an upper bound on time gaps and time intervals) should be replaced by variable MK in Eqs. (10) and (11), and the objective function can be written as $z = \min MK$.

4.11. Continuous relaxation of some execution and state variables

Despite execution and state variables being binary in nature, $Z_{j,n}$, $E_{j,n}$ and $W_{j,n}$ can be defined as non-negative continuous variables because they are forced to obtain binary values by Eqs. (1)–(3). Variable $Z_{j,n}$ is uniquely defined in Eq. (3), and since $X_{i,j,n}$ and $Y_{i,j,n}$ are strictly defined as binary variables, $Z_{j,n}$ can only be integral at every feasible solution. Similarly, variable $E_{j,n}$ is uniquely defined in Eq. (2), and since variable $X_{i,j,n}$ is binary and variable $Z_{j,n}$ can only assume integral values, $E_{j,n}$ will also assume integral values. Furthermore, since the left-hand side in Eq. (1) must always be equal to 1, $S^{I}_{j,n}$ and $S^{O}_{j,n}$ are strictly defined as binary variables, $E_{j,n}$ and $W_{j,n}$ are defined as non-negative continuous variables, and $E_{j,n}$ can only assume integral values; then, variables $E_{j,n}$ and $W_{j,n}$, and, therefore, $Z_{j,n}$, will always assume binary values at every feasible solution. Thus, the optimization variables in the proposed MILP formulation can be defined as follows:

$$X_{i,j,n}, Y_{i,j,n}, S^{I}_{j,n}, S^{O}_{j,n}, S^{S}_{m,k,n}, S^{D}_{k,n} \in \{0,1\}, \quad E_{j,n}, W_{j,n}, Z_{j,n} \geq 0$$
$$T_n, \bar{T}^{LB}_{j,n}, \bar{T}^{EE}_{j,n}, \bar{T}^{S}_{j,n}, \bar{T}^{W}_{j,n}, B^{S}_{i,j,n}, B^{P}_{i,j,n}, B^{E}_{i,j,n}, I^{S}_{m,k,n}, I^{I}_{m,j,n}, I^{O}_{m,j,n}, F^{VU}_{m,k,j,n}, F^{UV}_{m,j,k,n}, F^{UU}_{m,j',j,n}, F^{VV}_{m,k',k,n}, Q_{r,n} \geq 0$$

4.12. Number of time points

Continuous-time formulations for the short-term scheduling of multi-purpose batch plants are based on a set of time points (unit specific or global ones), which are non-uniformly distributed along the scheduling horizon. A limitation of these approaches is that the number of time points required to represent the optimal schedule is unknown a priori, so multiple MILP models need to be solved to reach the optimal solution. In this paper, we start by adopting a small number of time points (2 by default) to instantiate the model at the first iteration. Then, this number is gradually increased and the model is repeatedly updated and solved until the objective function does not exhibit an improvement. The development of methods for the reduction of the computational burden associated with this aspect will be explored further in Part III of this series, where we will adapt the procedure for estimating the minimum number of time points proposed by Giménez and Henning (2008).

4.13. Remarks

4.13.1. Modeling issues

Our goal is the development of a general framework which expands the scope of batch scheduling methods. The proposed representation introduces several innovative modeling concepts that turn it into a powerful tool. As discussed previously, the advantages are considerable in comparison with previous approaches. In this first part of the series we present the novel concepts of this representation and the mathematical formulation for short-term scheduling of multi-purpose batch plants. The proposed approach can be extended to consider changeover times and non-instantaneous material transfers, or even modified to account for continuous processes. Extensions will be presented in Part II of this series.

4.13.2. Computational performance

The generality of this framework leads inevitably to large MILP formulations that are perhaps computationally more demanding than previously proposed models (e.g. network-based discrete-time formulations). Nevertheless, our preliminary studies indicate that the computational requirements are comparable to those of other continuous-time formulations. Furthermore, the development of methods for the effective solution of the MILP formulations resulting from the proposed representation will be discussed in Part III of this series.

5. Representation capabilities of the proposed approach

To illustrate the advantages of the proposed representation we study three examples based upon the motivating multi-purpose facility in Fig. 1. Task information and material related data for all three examples are given in Tables 1 and 2, respectively. The

Table 1
Processing time coefficients, batch size limits, utility requirements and maximum utility availability for Examples 1–3.

Task i	Processing unit j	$a_{i,j}$ (h)	$b_{i,j}$ (h/kg)	$\beta^{MIN}_{i,j}$ (kg)	$\beta^{MAX}_{i,j}$ (kg)	Utility r	$f_{i,j,r}$ (kg/min)	$g_{i,j,r}$ (kg/min kg)	ρ^{MAX}_{r} (kg/min)
T1	R-101	0.5	0.025	40	80	HS	6	0.25	30[a] 40[b]
	R-102	0.5	0.04	25	50	HS	4	0.25	30[a] 40[b]
T2	R-101	0.75	0.0375	40	80	CW	4	0.3	30
	R-102	0.75	0.06	25	50	CW	3	0.3	30
T3	R-103	0.25	0.0125	40	80	HS	8	0.2	30[a] 40[b]
T4	R-103	0.5	0.025	40	80	CW[a] HS[b]	4	0.5	30[a] 40[b]

[a] Data corresponding to Examples 1–2
[b] Data corresponding to Example 3.

Table 2
Material prices, storage capacities, and initial inventories for Examples 1–3.

Material m	π_m ($/kg)	Storage vessel k	$\varsigma_{m,k}^{MAX}$ (kg)	$I_{m,k}^{0}$ (kg)
RM1	10	V-101	UIS	1000
RM2	15	V-102	UIS	1000
INT1	25	V-103	200	0
INT2	0	NIS	0	0
INT3	0	V-104[a]	500[a]	0
		NIS[b]	0[b]	
P1	30	V-105	UIS	0
P2	40	V-106	UIS	0

NIS = no-intermediate storage; UIS = unlimited intermediate storage.
[a] Data corresponding to Examples 1–2.
[b] Data corresponding to Example 3.

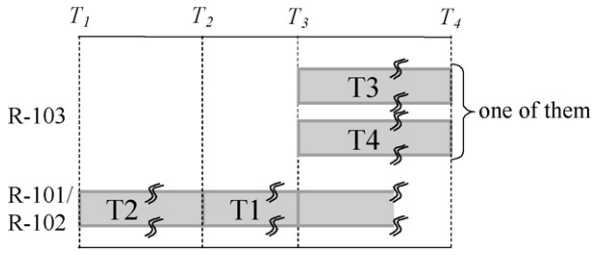

Fig. 13. Desired task ordering for Example 1.

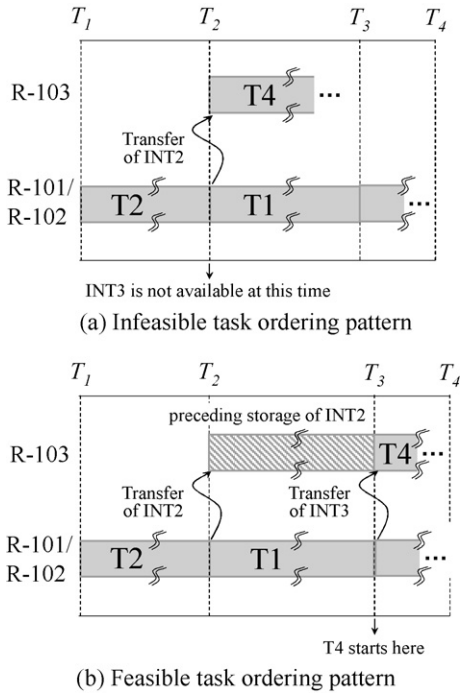

Fig. 14. Improvement in the treatment of the no-intermediate storage policy.

examples were solved using the formulation presented in Section 4 with the aim of getting optimal schedules in cases where existing approaches cannot obtain even a feasible solution. The MILP models were implemented in GAMS and solved using CPLEX 10.2 (using one thread) on a Pentium D (2.80 GHz) PC with 1 GB of RAM. In Part III of this series, a detailed computational analysis and comparisons with other approaches will be presented.

5.1. Example 1

The process network for this example is identical to the one shown in Fig. 1. The objective is the maximization of profit over a time horizon of 8 h ($H = 8$). Note that since no initial inventory of intermediates is held, the only way to obtain final products P1 and P2 is to perform task T2 first (so INT1 and INT2 become available), then execute T1 (so INT3 can be available), and finally carry out either T3 or T4. This situation is graphically illustrated in Fig. 13. Nevertheless, since a NIS policy is adopted for INT2, T4 should begin when task T2 finishes (at T_2 in the representation), but this is infeasible because INT3 is not yet available at this time (T_2). Therefore, current models cannot find a feasible solution for this example. On the contrary, the proposed formulation overcomes this limitation by allowing a temporal storage of INT2 in the unit R-103 until INT3 becomes available (preceding storage). Fig. 14 shows the two instances described above.

Fig. 15 presents the Gantt chart corresponding to the optimal solution and the associated utility consumption. An analysis of Fig. 15 reveals that the task ordering pattern suggested in Fig. 14b appears in the initial part of the schedule. In fact, it can be seen that a 40 kg batch associated with task T2 is processed in unit R-101 during time interval 1. Immediately afterwards, another batch of the same size, associated with task T1, is executed in the same unit during interval 2. While the first of these tasks produced INT2 that is sent to R-103 (28 kg), the second one produced INT3, which is also transferred to R-103 (18.67 kg). This allows starting task T4. Thus, a batch of 46.67 kg is manufactured during interval 3 to produce P2. The remaining tasks are allocated to different processing units.

Table 3 presents the computational results obtained from the implementation of the proposed MILP formulation adopting a zero optimality gap. It can be seen that six global time points (five time intervals) were required to achieve the optimal schedule. In this particular case the variable representing the material transfer between storage vessels ($F_{m,k',k,n}^{VV}$) was not considered since each material is allowed to be stored in at most one storage vessel. Also, variable $S_{m,k}^{D}$ was removed. Moreover, the variable $E_{j,n}$ was replaced by its equivalent expression (see Eq. (2)) in order to reduce the model size. Consequently, the model instance involved 87 binary variables, 655 continuous ones, and 646 constraints. An optimal solution having a profit value of $3592.2 was found in only 0.87 s, by exploring 282 nodes.

5.2. Example 2

In this subsection, we discuss the minimization of makespan. We consider three instances of the problem described in Example 1, corresponding to three demand patterns. We first considered minimum demands of 60 kg and 80 kg, for P1 and P2, respectively. In the second instance, we increased the demands to $d_{P1} = 80$ kg and $d_{P2} = 120$ kg. Finally, demands were modified to $d_{P1} = 150$ kg and $d_{P2} = 100$ kg. In all the cases, parameter H, which now only plays

Table 3
Model and solution statistics for Example 1.

N	CPU time (s)	Nodes	RMILP ($)	MILP ($)	Binary variables	Continuous variables	Constraints	Non-zeros
3	No profitable solution exists							
4	0.05	0	1475.3	1420.7	51	399	412	1720
5	0.38	21	3455.6	2730.8	69	527	529	2388
6	**0.87**	**282**	**4899.4**	**3592.2**	**87**	**655**	**646**	**3104**
7	9.86	2816	5748.7	3592.2	105	783	763	3868

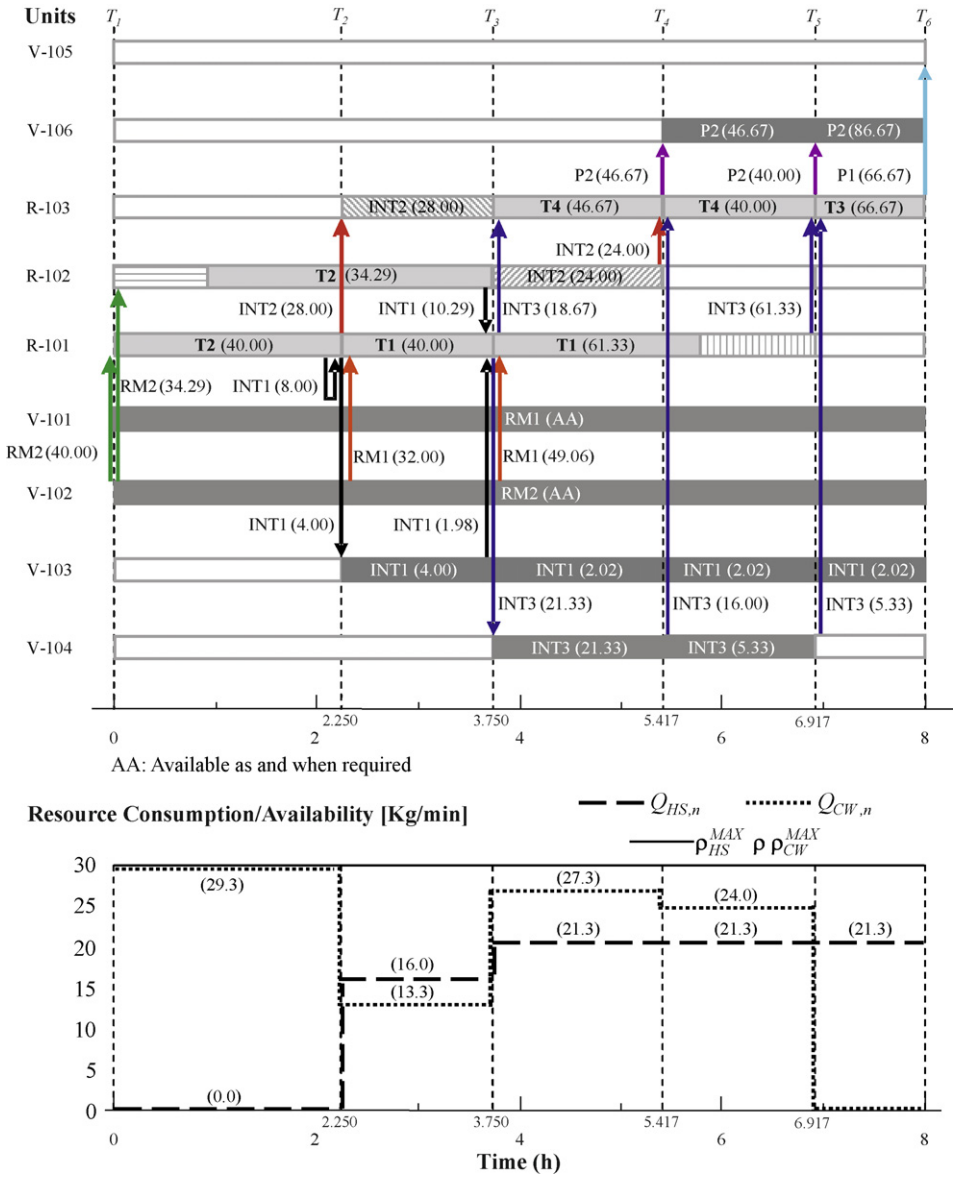

Fig. 15. Optimal schedule for Example 1.

Table 4
Model and solution statistics for Example 2.

N	CPU time (s)	Nodes	RMILP (h)	MILP (h)	Binary variables	Continuous variables	Constraints	Non-zeros
Example 2a (d_{P1} = 60 kg, d_{P2} = 80 kg)								
5	No feasible solution exists							
6	1.06	241	4.786	7.800	87	655	647	3118
7	**8.86**	**1317**	**4.439**	**7.781**	**105**	**783**	**764**	**3885**
8	84.5	14351	4.260	7.781	123	911	881	4700
Example 2b (d_{P1} = 80 kg, d_{P2} = 120 kg)								
6	No feasible solution exists							
7	3.08	510	6.528	11.488	105	783	764	3885
8	67.6	9698	6.191	11.417	123	911	881	4700
9	**1238**	**136,637**	**6.012**	**11.321**	**141**	**1039**	**998**	**5563**
10	10,000[a]	802,035	5.893	11.321	159	1167	1115	6474
Example 2c (d_{P1} = 150 kg, d_{P2} = 100 kg)								
8	No feasible solution exists							
9	58.2	4866	9.318	14.723	141	1039	998	5563
10	**1493**	**110,784**	**9.088**	**13.902**	**159**	**1167**	**1115**	**6474**
11	10,000[b]	647,576	8.921	13.902	177	1295	1232	7433

Relative gaps: [a] 10.99%, [b] 15.25%.

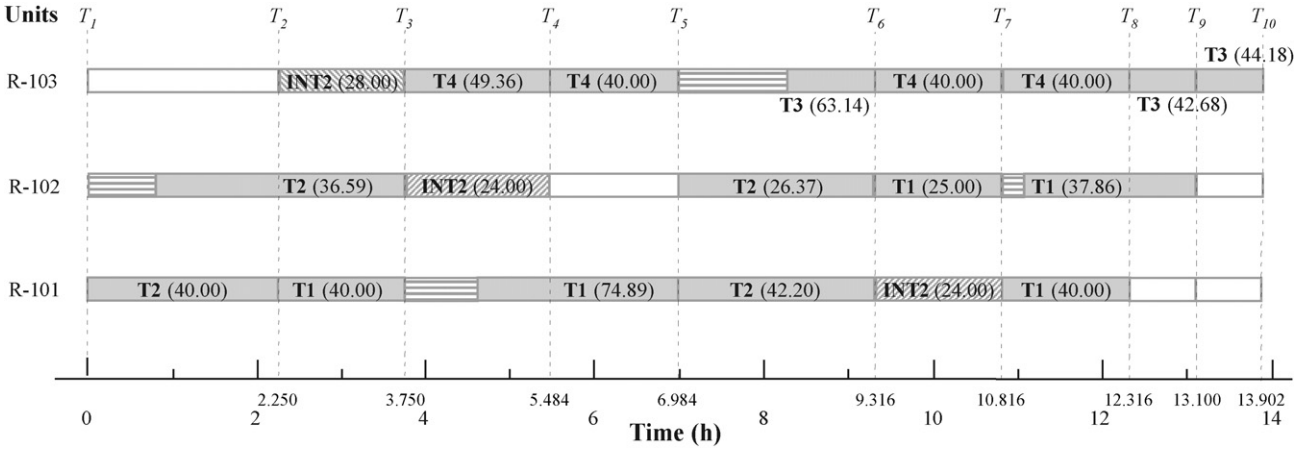

Fig. 16. Optimal schedule for Example 2c.

Elements	
Processing Units: R-101, R-102, R-103	
Tasks: T1, T2, T3, T4	
Storage Units: V-101, V-102, V-103, V-104, V-105, V-106	
Materials: RM1, RM2, INT1, INT2, INT3, P1, P2	
Utilities: Hot Steam (HS), Cooling Water (CW)	
Logical Connections	
Task/Processing Unit: T1/R-101, T1/R-102, T2/R-101, T2/R-102, T3/R-103, T4/R-103	
Task/Utility: T1/HS, T2/CW, T3/HS, T4/HS	
Material/Storage Unit: RM1/V-101, RM2/V-102, INT1/V-103, INT2/-, INT3/-, P1/V-105, P2/V-106	
Stoichiometric Relations	
T1	0.8 RM1 + 0.2 INT1 → INT3
T2	RM2 → 0.3 INT1 + 0.7 INT2
T3	INT3 → P1
T4	0.6 INT2 + 0.4 INT3 → P2
Plant Topology	

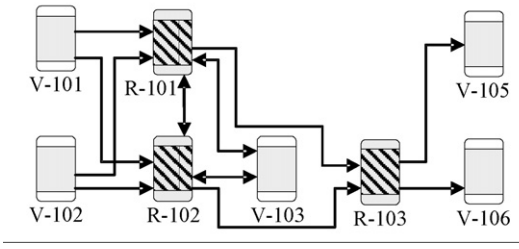

Fig. 17. Process network for Example 3.

the role of a reasonable upper bound on both slack variables (see Eqs. (4) and (5)) and time intervals (see Eqs. (6)–(8)), was fixed to 8 h. Table 4 summarizes the results of implementing the proposed formulation. Since, in general, the efficiency of continuous-time formulations deteriorates when minimizing makespan, we considered a time limit of 10,000 s for each model run. As Table 4 shows, this limit was exceeded in the last iteration for instances *b* and *c*.

The optimal solution for instance *a* has a makespan of 7.781 h and was found in only 8.86 s. Regarding instance *b*, the best solution with a makespan of 11.321 h was found in 1238 s. For instance *c* we obtained the best solution with a makespan of 13.90 h in 1493 s. Fig. 16 illustrates the optimal solution for problem instance *c*. For clarity reasons, a simplified Gantt chart with no material transfers is presented. Note that the solution is consistent with the production pattern identified in the previous subsection (Fig. 14).

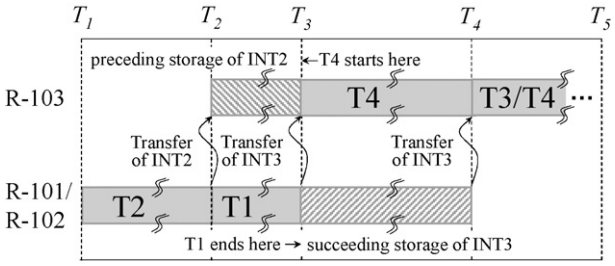

Fig. 18. Desired task ordering for Example 3.

Table 5
Computational analysis for Example 2.

N	PC A—1 thread		PC A—2 threads		PC B—4 threads	
	CPU time (s)	Nodes	CPU time (s)	Nodes	CPU time (s)	Nodes
Example 2a (d_{P1} = 60 kg, d_{P2} = 80 kg, H = 24 h)						
6	1.06	241	0.86	400	0.33	370
7	**8.86**	**1317**	**4.06**	**1043**	**2.50**	**1700**
8	84.5	14,351	29.8	9795	11.3	6139
Example 2b (d_{P1} = 80 kg, d_{P2} = 120 kg, H = 24 h)						
7	3.08	510	1.84	376	2.69	1080
8	67.6	9698	29.6	9278	11.2	6000
9	**1238**	**136,637**	**872**	**231,331**	**73.6**	**43,918**
10	>10,000	802,035	6815	1,485,592	521	326,309
Example 2c (d_{P1} = 150 kg, d_{P2} = 100 kg, H = 24 h)						
9	58.2	4866	27.9	4573	13.4	5900
10	**1493**	**110,784**	**524**	**87,830**	**107**	**51,926**
11	>10,000	647,576	6332	961,955	1356	437,686

PC A = Pentium Dual Core (2.80 GHz) PC with 1 GB of RAM. PC B = Pentium Quad Core (2.50 GHz) PC with 8 GB of RAM.

Table 6
Model and solution statistics for Example 3.

N	CPU time (s)	Nodes	RMILP ($)	MILP ($)	Binary variables	Continuous variables	Constraints	Non-zeros
3	No profitable solution exists							
4	0.07	0	2042.7	2042.7	51	386	404	1691
5	0.28	38	4130.5	3168.8	69	510	519	2350
6	**2.53**	**975**	**5250.5**	**3273.1**	**87**	**634**	**634**	**3057**
7	27.1	7195	5806.3	3273.1	105	758	749	3812

According to this solution, 150.00 kg of P1 and 169.35 kg of P2 are obtained. The overproduction of P2 is due to the production of intermediate INT2 (which cannot be stored) by task T1, which in turn produces the intermediate INT1, necessary to obtain INT3, which is in turn required by P1. Consequently, the productions of P1 and P2 are not independent from each other, and all the amount of INT2 produced has to be consumed to free up the corresponding processing unit. Hence, the following production pattern was identified: task T4 has to be executed each time INT2 is produced.

Finally, we discuss how the selection of hardware and software tools affects the computational performance of the proposed model. In particular, we solved the three instances described in this subsection running CPLEX using two and four threads on a dual- and a quad-core PC, respectively. Only computational statistics are presented in Table 5 since solutions are identical to the ones previously

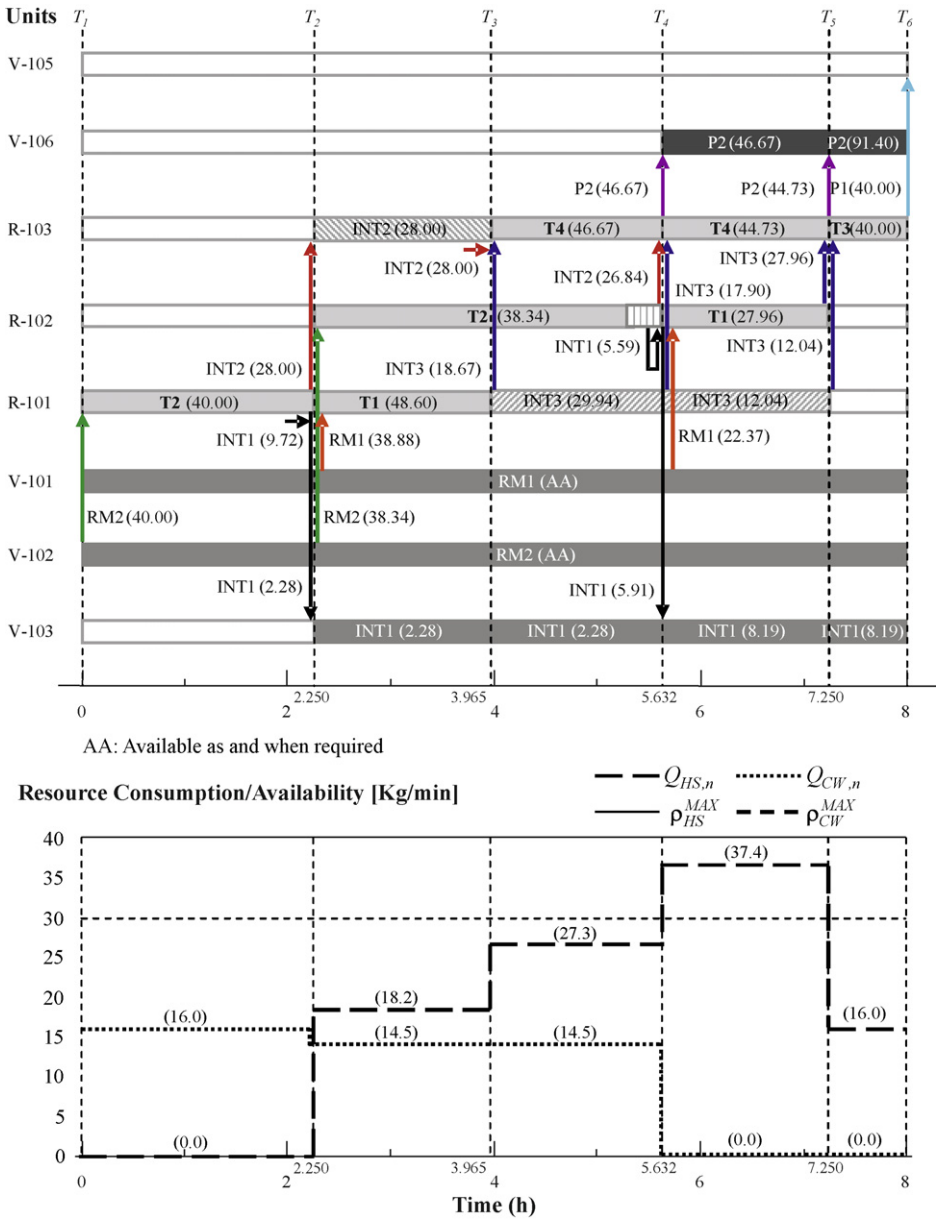

Fig. 19. Optimal schedule for Example 3.

found. As it can be seen, the increase in the computational power results in a meaningful reduction in the time required to find the optimal solution at each run.

5.3. Example 3

The process network for Example 3, which is a modification of the network of Examples 1 and 2, is depicted in Fig. 17. We consider the maximization of profit for a time horizon of 8 h ($H = 8$). Similarly to the preceding examples, an a priori analysis reveals that since two intermediates (INT2 and INT3) have no assigned storage vessel, all feasible solutions share a pattern where these two materials are temporarily stored in processing units at the beginning of the scheduling horizon, as shown in Fig. 18.

In existing approaches, any task beginning coincides with the start of the input materials load and task endings with the output materials discharge operation. Consequently, the previously described pattern might be infeasible due to a fictitious utility overconsumption (T1 and T4 require the same utility) even though material transfers can be explicitly modelled. Nevertheless, the proposed model decouples the material load/discharge from the task beginning/end, and therefore from the utility consumption (see Fig. 18).

Using the same variable simplifications proposed for example 1, we formulated a MILP model which involved 87 binary variables, 634 continuous ones, and 634 constraints. It was solved to optimality (zero optimality gap) in 2.53 s by exploring 975 nodes. Model and solution statistics are presented in Table 6. An optimal solution having an objective value of $3273.1 was reached when six global time points (five time intervals) were used. The schedule of this solution as well as the utility consumption profile along the scheduling horizon are shown in Fig. 19.

Once again, we observe that the initial task ordering follows the pattern identified by the previous analysis. A 40 kg batch of task T2 is processed in unit R-101 during time interval 1, immediately followed by another batch of task T1 of the same size processed in the same unit during time interval 2, while 28 kg of INT2 are temporally stored in unit R-103. Then, two batches of task T4 are successively processed in unit R-103, requiring a succeeding storage of INT3 in unit R-101 during time interval 3. Other tasks were allocated in the different processing units along the scheduling horizon. As it can be observed, the maximum resource availability is never exceeded.

6. Conclusions

In this paper, we presented a novel approach to the short-term scheduling of multi-purpose batch facilities. Its novelty lies in the explicit modeling of processing unit activity states, material transfers, and material storage in processing units. The resulting MILP formulation accounts for process features that are not considered in existing process scheduling approaches. Therefore, it can potentially obtain solutions to problems that are found infeasible by existing methods or obtain substantially better solutions. Therefore, the proposed representation and its associated MILP formulation can be considered a step forward in the solution of multi-purpose batch plant scheduling problems.

Acknowledgements

This work has been supported by CONICET (PIP 5915), Universidad Nacional del Litoral (CAI + D 3-14, 2005), and the National Science Foundation under Grant CTS-0547443.

Appendix A

Fig. A1 depicts the values of the so-called *execution* and *state* variables for a simple single-unit schedule. The values of the *time* variables are also shown.

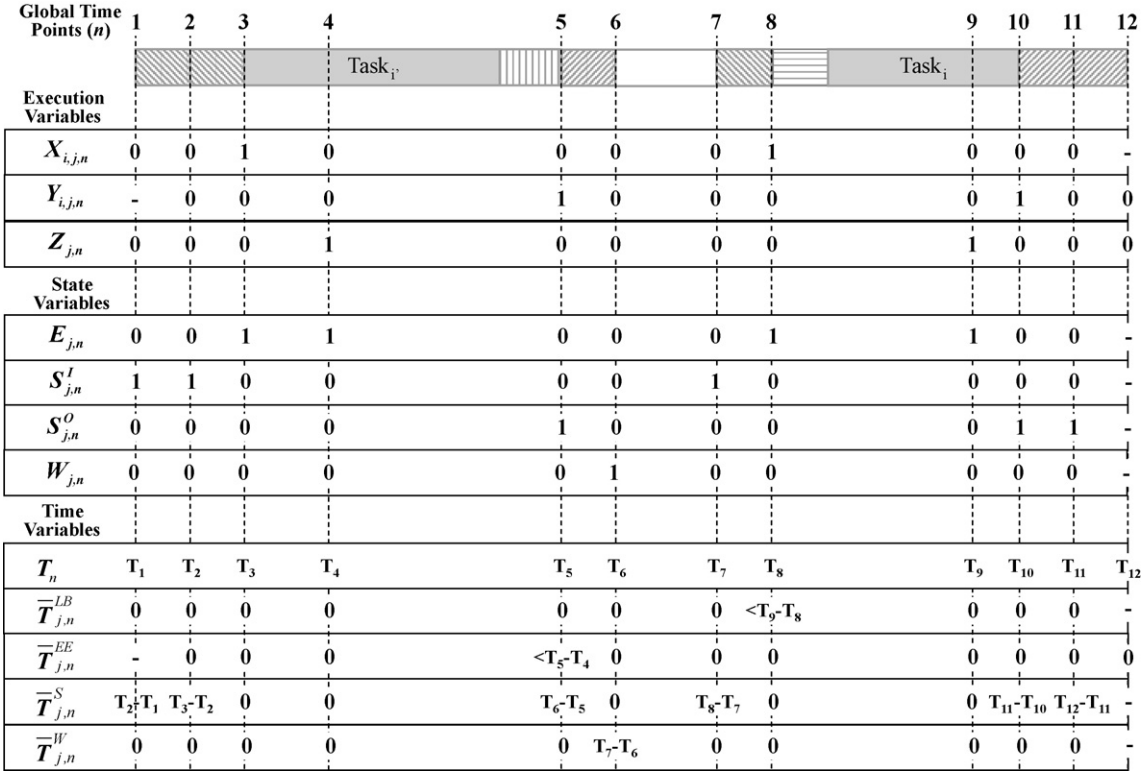

Fig. A1. Values of the execution, state and time variables for a simple schedule.

References

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

Burkard, R., & Hatzl, J. (2005). Review, extensions and computational comparison of MILP formulations for scheduling of batch processes. *Computers and Chemical Engineering*, *29*, 1752–1769.

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.

Castro, P. M., Barbosa-Póvoa, A. P., & Novais, A. Q. (2005). Simultaneous design and scheduling of multipurpose plants using resource task network based continuous-time formulations. *Industrial and Engineering Chemistry Research*, *44*, 343–357.

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

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. *Industrial and Engineering Chemistry Research*, *45*, 6210–6226.

Crooks, C. (1992). *Synthesis of operating procedures for chemical plants*. Ph.D. Thesis, University of London.

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.

Giménez, D. M., & Henning, G. P. (2007). An efficient global event-based continuous-time formulation for the short-term scheduling of multipurpose batch plants. *Computer-Aided Chemical Engineering*, *24*, 661–667.

Giménez, D. M., & Henning, G. P. (2008). The oSTN graph-set: an informed input that improves the efficiency of continuous-time scheduling formulations. In M. Ierapetritou, M. Bassett, & S. Pistikopoulos (Eds.), *Proceedings of the fifth international conference on foundations of computer-aided process operations* (pp. 409–412). CACHE Publications.

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.

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.

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

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*, *17*, 211–227.

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

Maravelias, C. T. (2005). Mixed-time representation for state-task network models. *Industrial and Engineering Chemistry Research*, *44*, 9129–9145.

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

Méndez, C. A., Cerda, J., Grossmann, I. E., Harjunkoski, I., & Fahl, M. (2006). State-of-the-art Review of optimization methods for short-term scheduling of batch processes. *Computers and Chemical Engineering*, *30*, 913–946.

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

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.

Reklaitis, G. V., Sunol, A., Rippin, D., & Hortascu, Ö. (Eds.). (1996). *Batch processing systems engineering*. Springer Verlag, NATO ASI Series.

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.

Schwindt, C., & Trautmann, N. (2000). Batch scheduling in process industries: an application of resource-constrained project scheduling. *OR Spectrum*, *22*, 501–524.

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*, *17*, 229–244.

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

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.