An Integrated CP-Based Approach for Scheduling of Processing and Transport Units in Pipeless Plants

Luis J. Zeballos

Universidad Tecnológica Nacional—FRP, Almafuerte 1033 (3100) Entre Ríos—Argentina

Carlos A. Méndez*

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

This work addresses the simultaneous production and material handling scheduling problem typically arising in pipeless plants with alternative layouts. An integrated constraint programming (CP) methodology, comprising both a detailed CP model and a suitable search strategy, is developed to address manufacturing environments in which multiple products with different recipes are to be produced. The general pipeless plant topology involves fixed processing stations and a limited number of moveable vessels used to transfer the material between consecutive stations, according to the predefined product recipes. Because of the high combinatorial complexity of the problem, an efficient search methodology is also presented to significantly accelerate the search and reduce the computational effort. The applicability of the proposed integrated CP methodology (model + search strategy) is successfully tested with several challenging examples taken from literature.

1. Introduction

Most of the work in the field of production scheduling has been focused on traditional recipe-driven multipurpose plants with different storage policies and fixed piping between batch processing units. A comprehensive review paper of the state of the art of optimization methods to traditional batch scheduling problems can be found in Méndez et al.[1] Nevertheless, several publications have also addressed scheduling problems of pipeless batch plants, which provide a significant increase of the plant flexibility to efficiently handle fast changes in market requirements. Pipeless plants represent a special type of multipurpose batch plant in which the batches of material are moved across the plant using limited moveable vessels. Thus, processing stations are linked by a transportation system or by automated guided vehicles (AGVs) that pick up the vessels to execute the material movements. Moveable vessels go from one processing unit to another, following a specific route, which consists of several consecutive subpaths (hereafter named tracks). Since, in general, several moveable vessels work simultaneously and they cannot pass each other, apart from buffer areas, their movements must be properly scheduled, to avoid collisions between them. Figure 1 shows a typical layout of a pipeless plant involving eight fixed processing units, a unique moveable vessel, 11 tracks, and 4 buffer areas.

As was noted by Huang and Chung,[2] the scheduling problem of pipeless plants is more challenging than in traditional batch plants. The higher complexity arises from the combinatorial nature of the assignment of machines, storage areas and moveable vessels, as well as the sequencing of tasks (processing, storage, and transport) to be faced. In addition, the generation of efficient schedules in pipeless plants is strongly dependent on the plant layout and the number of moveable vessels that are available. In this direction, Huang and Chung[2] also remarked that a wrong selection of moveable vessel routes may result in unproductive waiting times when more than one vessel requires the same tracks. Thus, to avoid collisions, the routing problem of moveable vessels in a given plant layout must be explicitly taken into consideration in the development of an effective scheduling approach.

An exhaustive analysis of the most relevant contributions related to operation of pipeless plants reveals that most of the approaches reported in the literature do not consider the scheduling and routing problems in an integrated manner. Pantelides et al.[3] presented a mixed-integer linear programming (MILP) model of the scheduling problem of pipeless plants. Their formulation showed significant computational limitations, because of the representation of the flexibility of units, such as moveable vessels. In turn, Realff et al.[4] presented a mixed-integer programming model to simultaneously integrate multiple decisions about design, layout, and scheduling of pipeless plants. Later, Gonzalez and Realff[5] developed a simulation model that was able to evaluate the results of the MILP model introduced by Realff et al.[4] The simulation approach allowed determination of the detailed movements of vessels, which were not specified in the MILP solution. Bok and Park[6] presented a new MILP model to address the scheduling problem of pipeless plants. These authors took into account jobshop features, such as re-entrance production flows and different processing directions. However, one of the main limitations of this model is that it ignored the issues related to the detailed routing of moveable vessels. More recently, Lee et al.[7] presented a MILP approach that was composed of three formulations: the production scheduling, the moveable vessel scheduling, and the buffer area scheduling. Their proposal included an iterative solution procedure, which was used when the number of products increased. Finally, Huang and Chang[2] developed a constraint programming (CP)-based model for the production scheduling problem, together with a procedure that generates routes for moveable vessels. The route planner is utilized to determine feasible routes after production activities related to every batch have been allocated to suitable stations. New constraints are added to the CP model when a feasible route is found. This approach can be applied to alternative plant layouts.

In this work, we present a detailed constraint programming formulation[8−10] that consists of both a scheduling model and a

* To whom correspondence should be addressed. Tel.: +54-342-4559175. Fax: +54-342-4550944. E-mail: cmendez@intec.unl.edu.ar.

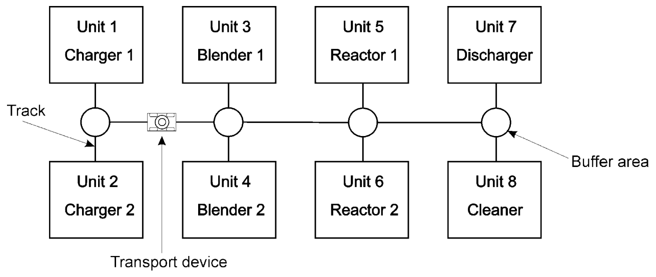

Figure 1. Schematic representation of a typical pipeless plant.

search strategy, which addresses the production scheduling and moveable vessel routing problems of pipeless plants in an integrated way. It is important to note that, since one problem is strongly dependent on the other one, all their related decisions must be made simultaneously to obtain good-quality schedules. The contribution takes into account transport related issues, because it is assumed that moveable vessels represent critical resources in the plant. In addition, storage areas are explicitly considered in the formulation, because of the fact that their finite capacities can enforce certain restrictions on the routing decisions. The CP formulation has been implemented in the OPL constraint programming language,[8] embedded in the OPL Studio Package.[11] The proposed approach uses the makespan as the objective function to be minimized. Other alternative criterion can be easily implemented.

The rest of the paper is structured as follows. Section 2 describes the integrated problem under consideration. Section 3 presents an overview of CP techniques, and sections 4 and 5 introduce the CP formulation: CP model and search strategy. The solution of different examples is presented and discussed in section 6. Finally, conclusions are given in section 7.

2. Problem Definition

The scheduling problem of pipeless batch plants with fixed processing stages and limited moveable vessels to be tackled in this paper has the following features:

(a) A set of batches of different products must be manufactured.

(b) Batches require that several processing stages be performed to achieve the condition of final products. Distinct product batches may require different numbers and types of stages.

(c) Processing operations required by a given product are performed in different processing units. Consequently, product batches must visit multiple stations.

(d) Each product batch is characterized by a known release time and a due date.

(e) The processing time for a product at given stage is dependent on the product, the stage, and the allocated unit.

(f) Processing stations are linked by a material handling system that transports several moveable vessels storing different product batches. The system has a finite number of vessels with limited storage capacity to perform the movement of material throughout the pipeless plant. Therefore, vessels become critical resources when generating efficient schedules.

(g) Moveable vessels move at a constant speed.

(h) Each processing unit and moveable vessel can execute just one operation on a product batch at a time (unary resources).

(i) The plant has several storage areas with a limited capacity.

(j) There are no breakdowns associated with units or moveable vessels.

(k) The same vessel is used throughout all processing stages for a given batch of product.

(l) Processing stations cannot be used to store materials.

Given all the above features, the scheduling problem consists of determining (i) the allocation of product batches to units, (ii) the product batch sequence at each unit, (iii) the allocating and sequencing of transport activities to be performed by the moveable vessels, (iv) the assignment and sequencing of storage activities that occur on buffer areas, as well as (v) the timings of the processing, storage, and transport activities. The goal of this problem is the makespan minimization.

3. The Constraint Programming Paradigm

Constraint Programming (CP) is a paradigm for modeling and solving optimization problems. This technique has been successfully applied in many domains, such as planning, vehicle routing, network configurations, and short-term scheduling.

One of the major strengths of CP is its inherent flexibility and expressiveness to define continuous, integer, and symbolic (as well as Boolean) decision variables. In addition, constraints can be easily described in terms of mathematical, symbolic, and logic notation and special constructs and global constraints are available to efficiently address specific problems, such as the short-term batch scheduling problem.

The CP models are solved using tree search algorithms combined with domain reduction and constraint propagation algorithms. While search algorithms systematically explore the possible assignments of values to variables, domain reduction and constraint propagation algorithms are applied to restrict the domains of the other variables whose values are not fixed. Domain reduction algorithms modify the range of possible values of each variable belonging to a given constraint when the domain of one of them has been modified. Constraint propagation algorithms transmit the effects of a variable domain change to any constraint that interacts with such variable. It is important to highlight that special propagation mechanisms have been developed for scheduling problems.[12-14]

The constraint programming paradigm has been extensively used by the process systems engineering community. In most of the cases, this technique has been combined with alternative optimization methodologies (Harjunkoski et al.,[15] Jain and Grossmann,[16] Harjunkoski and Grossmann,[17] Maravelias and Grossmann,[18] and Roe et al.[19]). However, in a few cases, it has been tested in its pure form (Zeballos and Henning[20]). In addition, the development and application of effective search strategies embedded in the core of the CP model have not been addressed for complex batch scheduling problems.

4. The Proposed CP-Based Approach

In this section, the CP approach (model + search strategy) for the scheduling problem of a pipeless plant is introduced.

4.1. Nomenclature. The following subsection give the nomenclature used in this study.

Subscripts

i = product

b = batch

l, l' = processing stage

j, j' = processing unit

f = storage unit

k, k' = moveable vessel

r = track

Sets

I = set of products

B_i = batches per each product i

L = set of stages

L_i = set of stages required by product i
J = set of alternative processing units
JIL_{il} = set of processing stations that can perform the processing of product i at stage l
MV = set of alternative moveable vessels
MVI_i = set of moveable vessels that can execute the transport of product i
TR = set of tracks
$TRJJ_{jj'}$ = ordered track set that corresponds to the route between units j and j'
Bf = set of alternative buffers
$BfJJ_{jj'}$ = ordered buffer set that corresponds to the buffers found on the route between units j and j'

Parameters
d_{ib} = due date of batch b of product i
rt_{ib} = release time of batch b of product i
t_{ilj} = processing time of stage l of product i in equipment unit j
tt_r = time taken by a moveable vessel to travel across track r
$ttr_{jj'}$ = time taken by a moveable vessel to go from unit j to station j'
cb_f = the total number of moveable vessels allowed to stay in buffer f
MNT = maximum number of consecutive tracks that make up the largest route in a given layout
MNB = maximum number of consecutive storage areas that are traversed in the largest route in a given layout

4.1.1. Special Parameters and Variables. To address any scheduling problem, tasks and resources must be explicitly represented. These elements are building blocks in the modeling language ILOG OPL Studio.[11] The model handles the following parameters related to the definition of resources:

- "*unit*" represents any processing station where the material loaded in a vessel is converted from one state to another. Stations are unary resources, which are resources that can perform, at most, one activity at a time.
- "*buffer*" refers to a storage area in which moveable vessels can wait, as well as pass each other. Buffers are discrete resources, which are another resource type that has a capacity greater than or equal to one. Thus, two and more tasks requiring the same storage space at a given time point can overlap in time, as long as their total requirement for the resource does not exceed its capacity (cb_f).
- "*mv*" represents any moveable vessel. This device is another unary resource that can transport only one product batch at a time.
- "*track*" denotes a subpath of a route between two different stations. Since tracks cannot be shared by two moveable vessels at the same time, these are declared as unary resources.

4.1.2. Model Variables. In this contribution, four types of activities are defined: "*PrTask*", "*CprTask*", "*MvtTask*", and "*StTask*". Each activity is described by means of its duration, start and end time variables (i.e., "*Task.duration*", "*Task.start*", and "*Task.end*"), which are related among themselves. The first type represents processing tasks executed on product batches. $PrTask_{ibl}$ activity corresponds to the processing of batch b of product i at stage l by a given station. The second one describes the complete processing routes of product batches. Thus, task $CprTask_{ib}$ models the manufacturing path of batch b of product i to achieve the condition of final product.

A route is formed by consecutive tracks that vessels should take to go from one station to another. Therefore, the third type of activities represents movements executed by moveable vessels on tracks. $MvtTask_{ibll's}$ task models the sth travel of a moveable vessel transporting batch b of product i between two different processing stations at stages l and l'. The last type of activities describes material storage taking place in buffer areas. $StTask_{ibll'v}$ task represents the vth wait of batch b of product i between stages l and l'.

Figure 2 shows a Gantt chart illustrating these activities and the relationships among them. The diagram conceptualizes the four types of activities that occur when batch 1 of product 1 is moved from unit 1 belonging to stage 1 to station 4 at stage 2. In this case, the moveable vessel takes tracks 1, 3, and 5, as well as buffers 1 and 2, to go from unit 1 to station 4. As can be seen, transportation activities $MvtTask_{1112s}$ with s = 1, 2, and 3 are associated with tracks 1, 3, and 5, respectively. In addition, storage tasks $StTask_{1112v}$ with v = 1 and 2 are connected with buffers 1 and 2. Finally, Figure 2 illustrates activity $CprTask_{11}$, which models the processing path of batch 1 of product 1 to achieve the condition of final product.

In addition, the model incorporates the following variable that defines the objective value to be optimized:

- Mk = maximum completion time or makespan.

4.2. Model Constraints. The CP approach employs some specific scheduling constructs available in the modeling language ILOG OPL Studio.[11] One of them is the construct "*requires*", which prescribes the assignment of renewable resources demanded by activities. Another construct is "*precedes*", which imposes a proper sequence of nonoverlapping activities. One of the most important constructs is "*activityHasSelectedResource*", which acts like a predicate that assumes a value of 1 when an activity has been assigned to a specific resource belonging to a given set of alternative resources.

4.2.1. Processing Station Assignment Constraints.

$$PrTask_{ibl}\ requires\ J \qquad \forall i \in I,\ \forall b \in B_i,\ \forall l \in L_i \tag{1}$$

$$\text{not } activityHasSelectedResource(PrTask_{ibl}, J, unit_j) \qquad \forall i \in I,\ \forall b \in B_i,\ \forall l \in L_i,\ \forall j \notin JIL_{il} \tag{2}$$

$$PrTask_{ibl}\ precedes\ PrTask_{ibl'} \qquad \forall i \in I,\ \forall b \in B_i,\ \forall l, l' \in L_i,\ l \neq last(L_i),\ ord(l') = ord(l) + 1 \tag{3}$$

Constraint 1 is an assignment relation denoting that batch b of product i at stage l must be assigned to just one processing unit belonging to the set of alternative stations J. Since machines have been declared as unary resources, all the activities that are assigned to them will be automatically sequenced, without requiring additional constraints. Constraint 1 is complemented with constraint 2, which negates the *activityHasSelectedResource* predicate to forbid the allocation of any $unit_j$, not belonging to the set of permitted stations JIL_{il} to the $PrTask_{ibl}$ activity. In turn, constraint 3 enforces a proper sequencing of tasks corresponding to any pair of consecutive processing operations at stages l and l' to be executed on batch b of product i by resorting the special construct *precedes*. Therefore, the activity located at the right-hand side cannot be started until the activity on the left-hand side is finished.

4.2.2. Timing Constraints of Processing Tasks.

$$PrTask_{ibl}.start \geq rt_{ib} \qquad \forall i \in I,\ \forall b \in B_i,\ l = first(L_i) \tag{4}$$

$$activityHasSelectedResource(PrTask_{ibl}, J, unit_j) \Rightarrow PrTask_{ibl}.duration = t_{ilj} \qquad i \in I,\ \forall b \in B_i,\ l = L_i,\ \forall j \in JIL_{il} \tag{5}$$

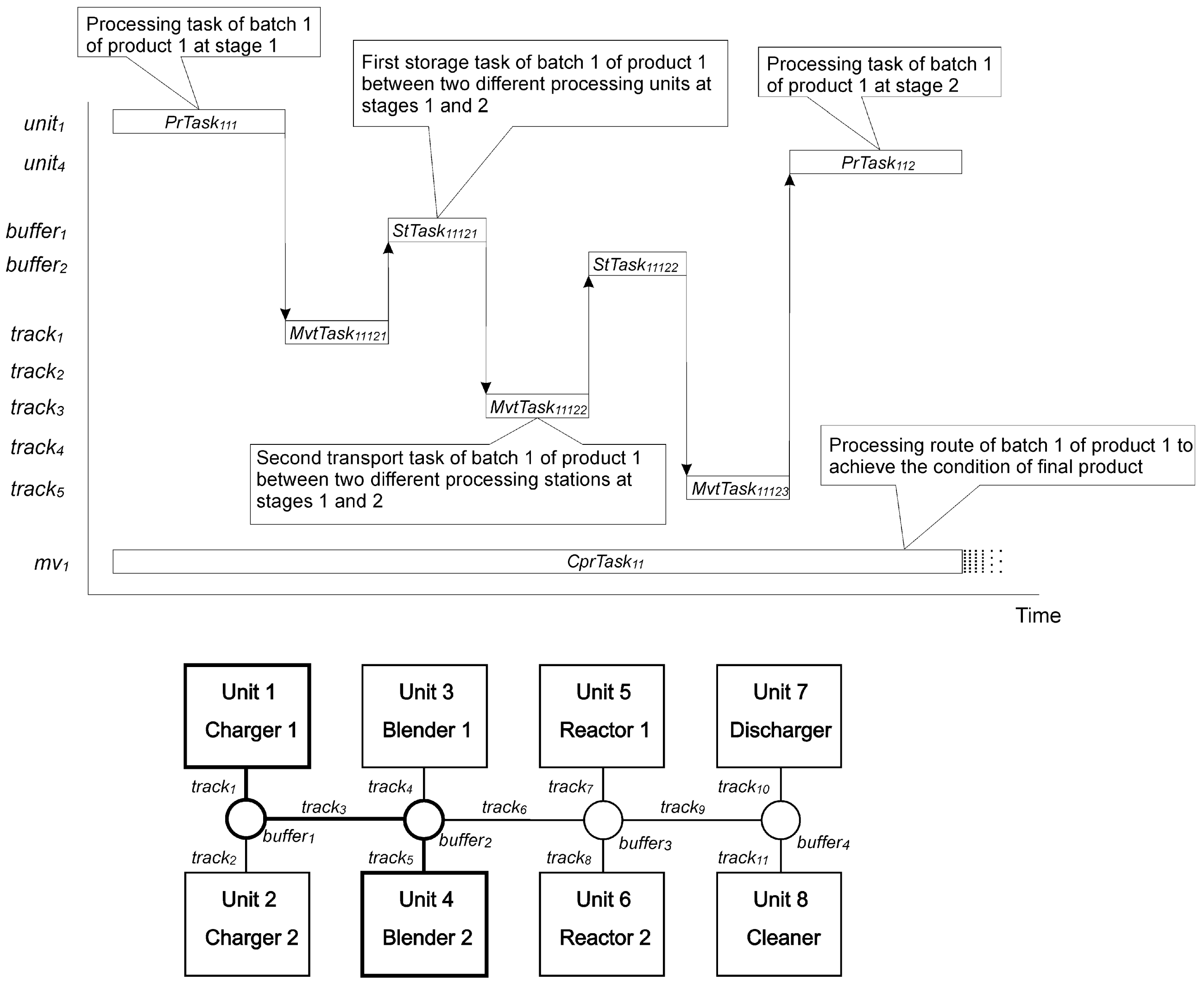
Processing task of batch 1 of product 1 at stage 1
First storage task of batch 1 of product 1 between two different processing units at stages 1 and 2
Processing task of batch 1 of product 1 at stage 2
unit1
unit4
buffer1
buffer2
track1
track2
track3
track4
track5
mv1
PrTask111
PrTask112
StTask11121
StTask11122
MvtTask11121
MvtTask11122
MvtTask11123
CprTask11
Second transport task of batch 1 of product 1 between two different processing stations at stages 1 and 2
Processing route of batch 1 of product 1 to achieve the condition of final product
Time
Unit 1 Charger 1
Unit 3 Blender 1
Unit 5 Reactor 1
Unit 7 Discharger
Unit 2 Charger 2
Unit 4 Blender 2
Unit 6 Reactor 2
Unit 8 Cleaner
track1
track2
track3
track4
track5
track6
track7
track8
track9
track10
track11
buffer1
buffer2
buffer3
buffer4

Figure 2. Conceptual representation of activities related to the execution of two consecutive processing operations on different stations.

$$PrTask_{ibl}.end \leq d_{ib} \qquad \forall i \in I,\ \forall b \in B_i,\ l = last(L_i) \tag{6}$$

Constraint 4 places a lower bound on the start time of the first processing stage on batch *b* of product *i*, because of the product release time (rt_{ib}). In turn, constraint 5 fixes the duration of the $PrTask_{ibl}$ activity according to the assigned station. Finally, constraint 6 forces an upper bound on the end time of the last processing stage of batch *b* of product *i*, because of its due date (d_{ib}).

4.2.3. Moveable Vessel Allocation Constraints.

$$CprTask_{ib}\ requires\ MV \qquad \forall i \in I,\ \forall b \in B_i \tag{7}$$

$$\text{not } activityHasSelectedResource(CprTask_{ib}, MV, mv_k) \qquad \forall i \in I,\ \forall b \in B_i,\ \forall k \notin MVI_i \tag{8}$$

Constraint 7 conveys that the transportation task $CprTask_{ib}$ must be assigned to just one moveable vessel belonging to the MV set. Constraint 8 forbids the allocation of any mv_k, not belonging to the set of permitted moveable vessels MVI_i to the $CprTask_{ib}$ activity.

4.2.4. Timing Constraints of Activities Describing the Complete Processing Routes of Product Batches.

$$CprTask_{ib}.start = PrTask_{ibl}.start \qquad \forall i \in I,\ \forall b \in B_i,\ l = first(L_i) \tag{9}$$

$$CprTask_{ib}.end = PrTask_{ibl}.end \qquad \forall i \in I,\ \forall b \in B_i,\ l = last(L_i) \tag{10}$$

Constraints 9 and 10 specify the start and end times of activities that occur in the moveable vessels. Expression 9 enforces the start of task $CprTask_{ib}$ of batch *b* of product *i* to coincide with the start of the first processing activity on batch *b* of product *i*. Constraint 10 sets the end of activity $CprTask_{ib}$ to be equal to the end of the last processing task on batch *b* of product *i*.

4.2.5. Tracks Assignment Constraints.

$$MvtTask_{ibll's}\ requires\ TR \qquad \forall i \in I, \forall b \in B_i,\ \forall l, l' \in L_i,\ \forall s = 1,...,MNT,\ ord(l') = \mathrm{ord}(l) + 1,\ ord(l) \neq last(L_i) \tag{11}$$

$$activityHasSelectedResource(PrTask_{ibl}, J, unit_j) \text{ and } activityHasSelectedResource(PrTask_{ibl'}, J, unit_{j'}) \Rightarrow activityHasSelectedResource(MvtTask_{ibll's}, TR, track_r) \quad \forall i \in I,\ \forall b \in B_i,\ \forall l, l' \in L_i,\ \forall s = 1,...,MNT,\ \forall j \in JIL_{il},\ \forall j' \in JIL_{il'},\ ord(l') = ord(l) + 1,\ ord(l) \neq last(L_i),\ \forall r \in TRJJ_{jj'}, ord(r) = s \tag{12}$$

```
search {
forall(i in I)
  forall(b in B[i])
    tryall(n in 1..card(MV))
      activityHasSelectedResource(CprTask[i,b],MV,mv[nextc(first(MV),n-1+ ord(b)+sum(i' in I: i'< i)  card(B[i'])]);
once {
  forall(i in I)
    forall(b in B[i])
      forall(l in L[i])
        tryall(j in JIL[i,l])
          activityHasSelectedResource(PrTask[i,b,l],J,mach[ j]);
};
setTimes(CprTask);
setTimes(PrTask);
setTimes(MvtTask);
setTimes(StTask);
};
```

Figure 3. Search strategy that balances the load of moveable vessels (GLEU-VDR, which stands for Guided Load of *E*quipment Units—Variable Domain Reduction).

$$MvtTask_{ibll's} \textit{ precedes } MvtTask_{ibll's'} \quad \forall i \in I,\ \forall b \in B_i,\ \forall l, l' \in L_i,\ \forall s, s' = 1,...,MNT,$$
$$ord(l') = ord(l) + 1, l \neq last(L_i), s' = s + 1, s \neq MNT \quad (13)$$

Constraint 11 imposes the assignment of activity $MvtTask_{ibll's}$ to just one track. Constraint 12 allocates the *s*th transportation task $MvtTask_{ibll's}$ of batch *b* of product *i* between different processing stations *j* and *j′* (belonging to stages *l* and *l′*) to track *r*. Thus, this constraint associates the *s*th transport activity with a specific track *r* on the plant layout. It is important to note that the allocations are made in order. The first track that belongs to the route between *j* and *j′* is connected with activity $MvtTask_{ibll's}$ with $s = 1$. It is also worth remarking that the proposed constraints can be easily used to represent any type of plant layout. The condition $ord(r) = s$ in constraint 12 is defined according to the set $TTJJ_{jj'}$, which describes an ordered track set between units *j* and *j′*. For instance, based on the layout shown in Figure 2, the set $TTJJ_{unit1,unit4}$ will be /track1,track3,track5/, where $ord(track1) = 1$, $ord(track3) = 2$, and $ord(track5) = 3$.

Constraint 13 enforces a proper sequencing of two consecutive transportation tasks ($MvtTask_{ibll's}$ and $MvtTask_{ibll's'}$ with $s' = s + 1$), corresponding to the movement of batch *b* of product *i* between stages *l* and *l′*.

4.2.6. Timing Constraints of Transportation Activities.

$$activityHasSelectedResource(PrTask_{ibl}, J, unit_j) \text{ and } activityHasSelectedResource(PrTask_{ibl'}, J, unit_{j'}) \Rightarrow$$
$$MvtTask_{ibll's}.duration = tt_r \forall i \in I, \forall b \in B_i,\ \forall l, l' \in L_i,$$
$$\forall s = 1,...,MNT, \forall j \in JIL_{il},\ \forall j' \in JIL_{il'}, ord(l') = ord(l) + 1,$$
$$ord(l) \neq last(L_i), \forall r \in TRJJ_{jj'},\ ord(r) = s \quad (14)$$

$$MvtTask_{ibll's}.start = PrTask_{ibl}.end \qquad \forall i \in I,\ \forall b \in B_i,$$
$$\forall l, l' \in L_i,\ ord(l') = ord(l) + 1,\ l \neq last(L_i),\ s = 1 \quad (15)$$

$$activityHasSelectedResource(PrTask_{ibl}, J, unit_j) \text{ and } activityHasSelectedResource(PrTask_{ibl'}, J, unit_{j'}) \Rightarrow$$
$$MvtTask_{ibll's}.end = PrTask_{ibl'}.start \forall i \in I,$$
$$\forall b \in B_i,\ \forall l, l' \in L_i,\ \forall s = 1,...,MNT,\ \forall j \in JIL_{il},$$
$$\forall j' \in JIL_{il'},\ ord(l') = ord(l) + 1, ord(l) \neq last(L_i),$$
$$\forall r \in TRJJ_{jj'},\ r = last(TRJJ_{jj'}),\ ord(r) = s \quad (16)$$

Constraint 14 fixes the length of the *s*th transportation activity $MvtTask_{ibll's}$ of batch *b* of product *i* between stations *j* and *j′*. The duration of this task is equal to the time required by a moveable vessel to travel across the *s*th track of set $TRJJ_{jj'}$. Constraints 15 and 16 specify the start and end times of the first transportation activity ($s = 1$) and the one that corresponds to the order of the last element of $TRJJ_{jj'}$ ($s = ord(r)$ with $r = last(TRJJ_{jj'})$), respectively, when batch *b* of product *i* is moved from stage *l* to stage *l′*. Expression 15 enforces the beginning of activity $MvtTask_{ibll's}$ with $s = 1$ to coincide with the end of processing task $PrTask_{ibl}$ at stage *l*. Constraint 16 specifies that the end of activity $MvtTask_{ibll's}$ with $s = ord(r)$ matches with the start of task $PrTask_{ibl'}$ performed by unit *j′* at stage *l′*.

4.2.7. Assignment of Storage Activities.

$$StTask_{ibll'v} \textit{ requires } Bf \qquad \forall i \in I,\ \forall b \in B_i,\ \forall l, l' \in L_i,$$
$$\forall v = 1,...,MNB,\ ord(l') = ord(l) + 1,\ ord(l) \neq last(L_i) \quad (17)$$

$$activityHasSelectedResource(PrTask_{ibl}, J, unit_j) \text{ and } activityHasSelectedResource(PrTask_{ibl'}, J, unit_{j'}) \Rightarrow$$
$$activityHasSelectedResource(StTask_{ibll'v}, \mathrm{Bf}, \mathrm{buffer}_f)$$
$$\forall i \in I,\ \forall b \in B_i,\ \forall l, l' \in L_i,\ \forall v = 1,...,MNB,$$
$$\forall j \in JIL_{il},\ \forall j' \in JIL_{il'}, ord(l') = ord(l) + 1,$$
$$ord(l) \neq last(L_i), \forall f \in BfJJ_{jj'}, ord(f) = v \quad (18)$$

Table 1. Major Characteristics of Examples 1, 2, and 3

example	products	batches per product	available moveable vessels	processing units
1	1−3	2 batches of products 1 and 2, 1 batch of product 3	3	8
2	1− 3	1 batch per product	2	8
3	1− 6	1 batch per product	5	8

Expressions 17 and 18 enforce the assignment of storage tasks to buffers. Constraint 17 prescribes that the υth task $StTask_{ibll'\upsilon}$ must be allocated to just one buffer, belonging to the Bf set. Expression 18 assigns the υth storage activity $StTask_{ibll'\upsilon}$ of batch b of product i between two different processing stations j and j' (belonging to stages l and l'), to the υth buffer of set $BfJJ_{jj'}$. Because the allocations are made in order, the first buffer area found in the route between j and j' is connected with activity $StTask_{ibll'\upsilon}$ with $\upsilon = 1$.

4.2.8. Timing Constraints of Storage Activities.

$$MvtTask_{ibll's}.end = StTask_{ibll'\upsilon}.start \text{ and } MvtTask_{ibll's'}.start = StTask_{ibll'\upsilon}.end \qquad \forall i \in I,\ \forall b \in B_i,\ \forall l, l' \in L_i,\ \forall s, s' = 1,...,MNT,\ \upsilon = 1,...,MNB,\ ord(l') = ord(l) + 1,\ ord(l) \neq last(L_i), s' = s + 1,\ \upsilon = s \quad (19)$$

Constraint 19 specifies the start and end times of a storage activity that occurs between two consecutive transportation activities of batch b of product i between stages l and l'. This expression establishes that the υth storage activity begins when the sth transportation task finishes, and ends when the s'th movement task starts.

4.2.9. Constraints That Accelerate the Search Process. The computational performance of the CP approach is highly dependent on variable domains. Thus, several constraints are introduced in the model to quickly reduce the domains of some variables. Constraints 20 and 22 permit it by setting bounds on the duration of model activities, $PrTask_{ibl}$, $CprTask_{ib}$, and $MvtTask_{ibll's}$. For

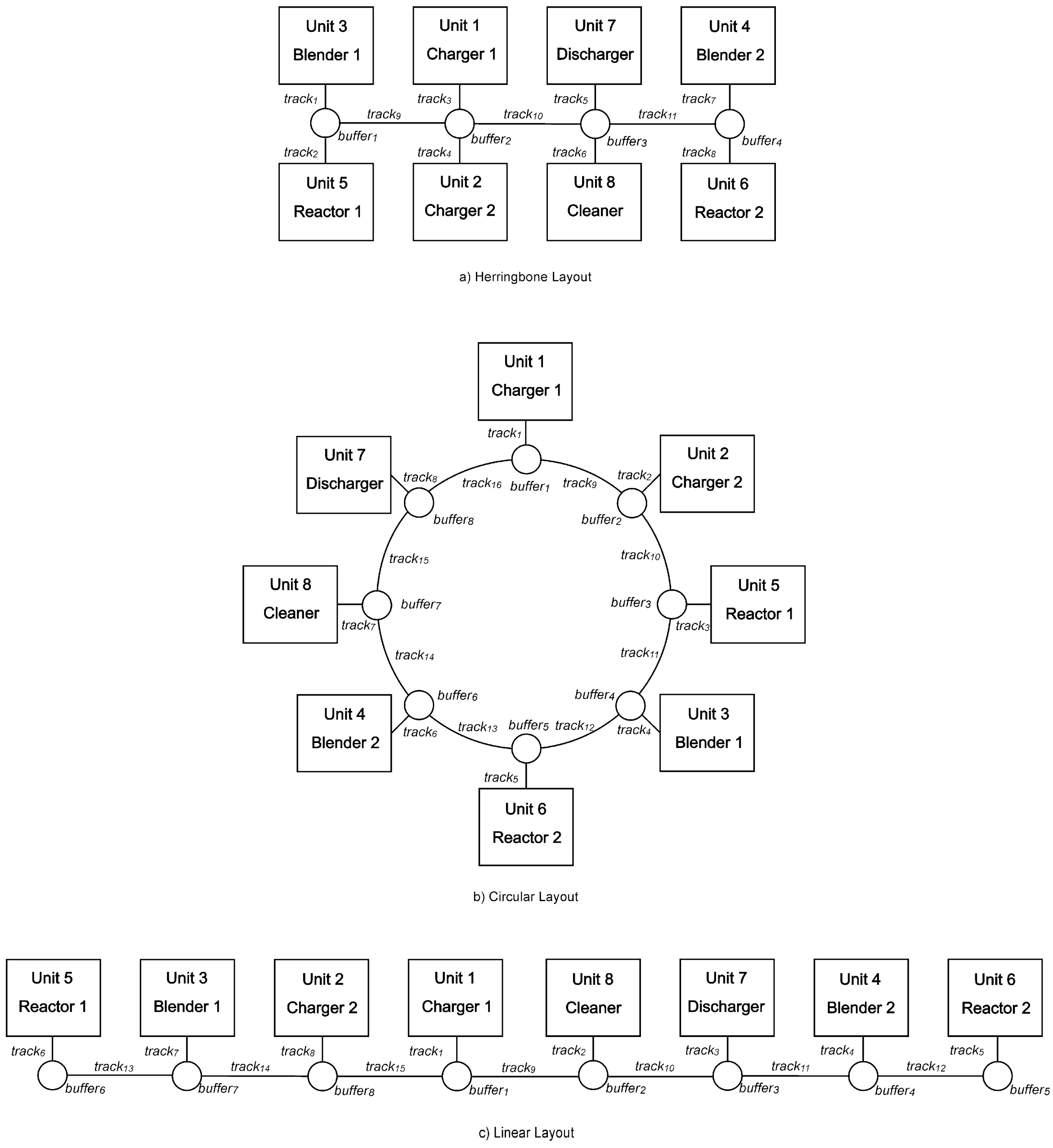

Figure 4. Schematic representation of typical pipeless plant layouts: (a) herringbone, (b) circular, and (c) linear.

Table 2. Computational Results for Example 1, Considering a Herringbone Layout

	Huang and Chung[2]			CP Model + DFS			CP Model + GLEU-VDR		
	first solution	last solution	system termination	first solution	last solution	system termination	first solution	last solution	system termination
number of choice points	53	137383	241586				348	2239	6320
CPU time (s)	0.1[a]	722.599[a]	945.609[a]				1.41	11.94	49.16
makespan (unit of 0.1 h)	177	123	123				177	123	123

[a] Obtained using a P3 500-MHz device.

Table 3. Computational Results for Example 1, Considering a Circular Layout

	Huang and Chung[2]			CP model + DFS			CP model + GLEU-VDR		
	first solution	last solution	system termination	first solution	last solution	system termination	first solution	last solution	system termination
number of choice points	53	269655	354221				348	7762	7762
CPU time (s)	0.26[a]	8091.15[a]	10800[a]				5.39	173.28	173.28
makespan (unit of 0.1 h)	261	248	248				201	139	139

[a] Obtained using a P3 500-MHz device.

Table 4. Computational Results for Example 1, Considering a Linear Layout

	Huang and Chung[2]			CP model + DFS			CP model + GLEU-VDR		
	first solution	last solution	system termination	first solution	last solution	system termination	first solution	last solution	system termination
number of choice points	53	136606	226281	1009	1711		569	3224	
CPU time (s)	0.12[a]	876.44[a]	1099.27[a]	7.12	12.05		6.83	27.95	
makespan (unit of 0.1 h)	195	135	135	197	195	195[b]	195	135	135[b]

[a] Obtained using a P3 500-MHz device. [b] Reaches termination criterion.

instance, using the accumulated minimum processing times of batch b of product i in every stage l, constraint 21 enforces a lower bound on the entire task duration. Transportation times related to common tracks through the product recipe may be also included in this constraint to make it even tighter. In turn, constraint 22 defines an upper bound on the total transportation time of a vessel moving a batch b of product i between stages l and l'. Although, in some cases, the proposed inequality constraint can be replaced by a strict equality, a better computational performance was observed when the proposed equation was used to solve the examples included in this paper. Frequently, the use of very tight constraints are not always the best option for generating more-efficient CP models.

$$\min_{j\in \mathrm{JIL}_{il}}(t_{ilj}) \le PrTask_{ibl}.duration \le \max_{j\in \mathrm{JIL}_{il}}(t_{ilj}) \qquad \forall i \in I,\ \forall b \in B_i,\ \forall l \in L_i \quad (20)$$

$$CprTask_{ib}.duration \ge \sum_{l\in L_i}(\min_{j\in \mathrm{JIL}_{il}}(t_{ilj})) \qquad \forall i \in I,\ \forall b \in B_i \quad (21)$$

$$activityHasSelectedResource(PrTask_{ibl}, J, unit_j) \text{ and} activityHasSelectedResource(PrTask_{ibl'}, J, unit_{j'}) \Rightarrow \sum_{s=1,\dots,\mathrm{MNT}}(MvtTask_{ibll's}.\mathrm{duration}) \le \mathrm{ttr}_{jj'}$$
$$\forall i \in I,\ \forall b \in B_i,\ \forall l, l' \in L_i,\ \forall j \in JIL_{il},\ \forall j' \in JIL_{il'},\ ord(l') = ord(l) + 1, ord(l) \ne last(L_i) \quad (22)$$

4.2.10. Objective Function. The makespan is chosen as the problem objective function and expressions 23 and 24 allow its explicit definition.

$$PrTask_{ibl}.end \le Mk \qquad \forall i \in I,\ \forall b \in B_i,\ ord(l) = last(L_i) \quad (23)$$

$$\text{Minimize } Mk \quad (24)$$

5. Search Procedure

CP systems allow users to choose one search strategy from a set of default ones or define one that can be tailored to a given problem type. One of the most common default strategies is Depth-First Search (DFS), which is an effective strategy from the viewpoint of memory management. In this section, a domain-specific search procedure is introduced, which attempts to avoid wasting time because of early bad choices in the search and tries to produce good-quality feasible solutions with low computational effort. The proposed strategy is referred as GLEU-VDR (where this acronym stands for Guided Load of *E*quipment Units—Variable Domain Reduction).

The GLEU-VDR strategy is depicted in Figure 3. Statements in lines 2 and 3 arrange the assignment of activities that describe the complete processing routes of product batches. Next, statements in lines 4 and 5 will attempt to allocate activities corresponding to the batches of the first product to moveable vessels, then it will attempt with tasks related to the batches of the second product and so on, until reaching the batches of the last product. Thus, it proceeds chronologically, to build an initial allocation of activities to moveable vessels in a consistent fashion. The assignment step will start with tasks corresponding to the first product, $CprTask_{1b}$, in turn, trying to allocate them to one of the units that belongs to the set of alternative moveable vessels MV. If an assignment is found to be infeasible, the system will backtrack and try another one. When all moveable vessels belonging to set MV have been tried and have also failed, the backtracking step will be deeper and will affect the previously allocated task. Therefore, if necessary, all the assignments can be revised and the backtracking process can achieve the allocation of the activity corresponding to the first batch of the first product. Note that the third argument of the *activityHasSelectedResource* predicate (line 5 in Figure 3) is a function that will attempt each task assignment by iterating over the corresponding set of alternative moveable vessels in a

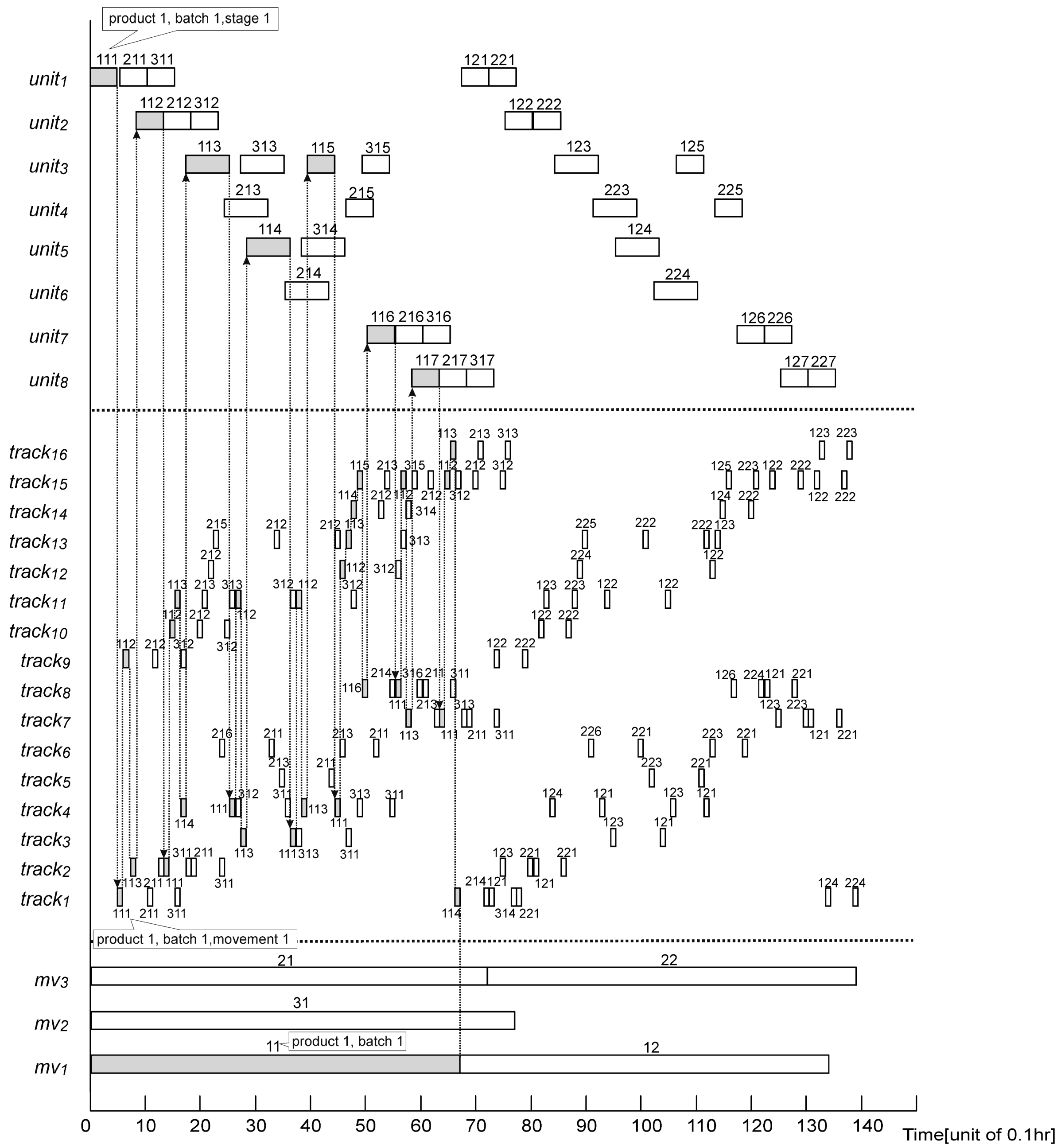

Figure 5. Gantt diagram depicting the best solution for the scheduling problem of a pipeless plant in a circular layout.

Table 5. Computational Results for Example 2, Considering Herringbone, Circular, and Linear Layouts

	Herringbone Layout			Circular Layout			Linear Layout		
	first solution	last solution	system termination	first solution	last solution	system termination	first solution	last solution	system termination
number of choice points	202	649	649	198	1747	1747	343	4380	4380
CPU time (s)	0.81	30.13	30.13	1.13	41.64	41.64	2.19	52.84	52.84
makespan (unit of 0.1 h)	113	113	113	129	125.5	125.5	125	125	125

circular way (*nextc*). Consequently, if an activity $CprTask_{ib}$ has been initially assigned to moveable vessel *k*, the system will try to allocate the next activity $CprTask_{ib+1}$ to unit *k*′, which is the next one in set MV. If the number of product batches is greater than the number of moveable vessels, once the last unit of such set has been assigned an activity, the next task will be allocated to the first unit of set MV. Therefore, the procedure begins a second step of assignments.

Assuming that the allocation step of activities describing the complete processing routes of product batches to moveable vessels has been successful, the assignment of tasks to processing units is tackled. This part of the procedure will only try to find a feasible solution (line 6). Statements in lines 7, 8, and 9 in Figure 3 arrange processing activities. The "*tryall*" instruction in line 10, shown in Figure 3, will attempt to allocate processing stage tasks corresponding to the first batch of the first product to units. After that, the system will continue with processing stage activities associated with the second batch of the first product and so on, until achieving tasks of the last batch of the last product. Therefore, it proceeds chronologically to develop

Table 6. Computational Results for Example 3, Considering Herringbone, Circular, and Linear Layouts

	Herringbone Layout			Circular Layout			Linear Layout		
	first solution	last solution	system termination	first solution	last solution	system termination	first solution	last solution	system termination
number of choice points	316	850589		414	252992		485	19917	
CPU time (s)	3.83	5303.31		7.8	5078.88		10.27	695.61	
makespan (unit of 0.1 h)	795	720	720[a]	980	780	780[a]	900	800	800[a]

[a] Reaches termination criterion.

a solution. The station allocation only considers the equipment units that can execute the activity being taken into account (JIL_{il}). The system will start with activities corresponding to the first batch of the first product, $PrTask_{11L}$, in turn, trying to assign them to one of the units that belongs to set JIL_{il}. In the case that an infeasibility is detected, the system will backtrack and try another unit. If all processing units belonging to set JIL_{il} have been tried and have also failed, the backtracking step will affect the earlier allocated task. When the assignment procedure cannot find a set of feasible allocations, the backtracking will be even deeper and will reach the assignment of activities describing the complete processing routes of product batches to moveable vessels.

If all processing tasks have successfully been allocated, the variable domain reduction procedure begins. It will attempt to find the timing to the complete set of model activities operating on the domains of their start time variables. The domains of such variables are characterized by two extreme points: the earliest start time (EST) and the latest start time (LST). The procedure will execute the domain pruning by adopting the lowest domain values for the start time variables. The domain reduction method (the script written in lines 13−16 in Figure 3, using the "*setTimes*" instruction) begins by selecting the earliest start time between all domains of activities describing the complete processing routes of product batches and, then, it chooses one task that can be scheduled at that time. When one activity is selected, two alternatives arise: while the first option fixes the start time of the task at the earliest starting time, the second one postpones the activity. The procedure continues in a similar way, taking into account all activities that are not yet scheduled or not postponed. A delayed activity is reanalyzed when its start time is updated. When all of the start times of the $CprTask_{ib}$ activities have been fixed, the domain reduction procedure continues in the same fashion with the remaining model activities. If the domain reduction method cannot find a set of feasible start times for one activity type, the backtracking will reach the assignment of tasks describing the complete processing routes of product batches to moveable vessels. If the domain reduction of storage tasks, which are the last tackled ones, succeeds, a feasible solution is found. To find new solutions, the system backtracks and, then, it will attempt a new allocation of activities describing the complete processing routes of product batches to moveable vessels.

6. Computational Results

In this section, three different examples of scheduling problems arising in pipeless plants are solved to illustrate the applicability and computational performance of the proposed CP-based approach (model + search strategy). In all of the examples, three different sets of plant layouts were considered (herringbone, circular, and linear). The computational performance was obtained using the makespan as the objective function to be minimized. Table 1 summarizes the major features of the three challenging examples addressed. Example 1 corresponds to the problem first introduced by Huang and Chung[21] and later studied by Huang and Chung,[2] whereas Examples 2 and 3 respectively refer to the scheduling problems 1 and 2 presented in the work of Bok and Park.[6]

Data for the three examples are given in Tables A1−A5 of the Appendix. While Table A1 illustrates the relationship between tasks and processing units for all case studies, Tables A2−A4 describe the processing times. The suitability for allocating moveable vessels to products in each example is shown in Table A5. In addition, the process recipes for products of Examples 1, 2, and 3 are explicitly represented through the state task network (STN) concept in Figures A1 and A2.

Because of the fact that plant layout may have direct impact on the results of scheduling problems in pipeless plants, the three most common layouts,[2] which are "herringbone", "circular", and "linear", were taken into account for Examples 1, 2, and 3. Figure 4 illustrates the connectivity of units for the three types of layouts. As can be seen, this figure shows where the units are located and what tracks and storage areas are included for herringbone, circular, and linear layouts. Note that, for all case studies, it is assumed that the time required by moveable vessels to cross any track is 0.1 h, and the capacity of storage areas is limited to one moveable vessel. Also, note that, because CP only involves integer values for the time representation, a time grid of 0.1 h is defined for Example 1. In turn, Examples 2 and 3 require a finer interval of 0.01 h.

Case studies were solved considering two search strategies. Therefore, the computational performance of the DFS and GLEU-VDR strategies was analyzed. Problems were solved to optimality, enforcing a maximum time limit of 5400 s of CPU. The commercial software ILOG OPL Studio 3.7, the ILOG Solver,[22] and the Scheduler[23] packages were used to solve all the examples in a Pentium 4 3.2 GHz PC with 1 GB of RAM.

6.1. Example 1. Computational results for Example 1, considering three different plant layouts, are presented in Tables 2, 3, and 4 together with those reported by Huang and Chung.[2] From the analysis of the results, one can easily observe the poor performance of the CP model when a standard DFS strategy is employed. The pure DFS strategy was not able to find even a feasible solution for the problem instances considering herringbone and circular layouts. In addition, this strategy was not able to demonstrate the optimality of the solution generated for the linear layout. On the other hand, it can be also seen that the proposed GLEU-VDR strategy clearly outperforms the DFS method, allowing one to obtain better solutions for all of the instances of Example 1 with modest computational effort.

A comparison between the results of the CP model using GLEU-VDR strategy and the solution reported by Huang and Chung[2] shows that the approach presented in this paper has a significantly better computational performance. However, it is important to remark that Huang and Chung's[2] approach was solved using a P3 500-MHz processor. Although differences in computer hardware make the direct comparison of CPU times rather unfair, it should be noted that the proposed CP approach requires a significantly fewer number of choice points in all the cases, which highlights its higher efficiency. For the

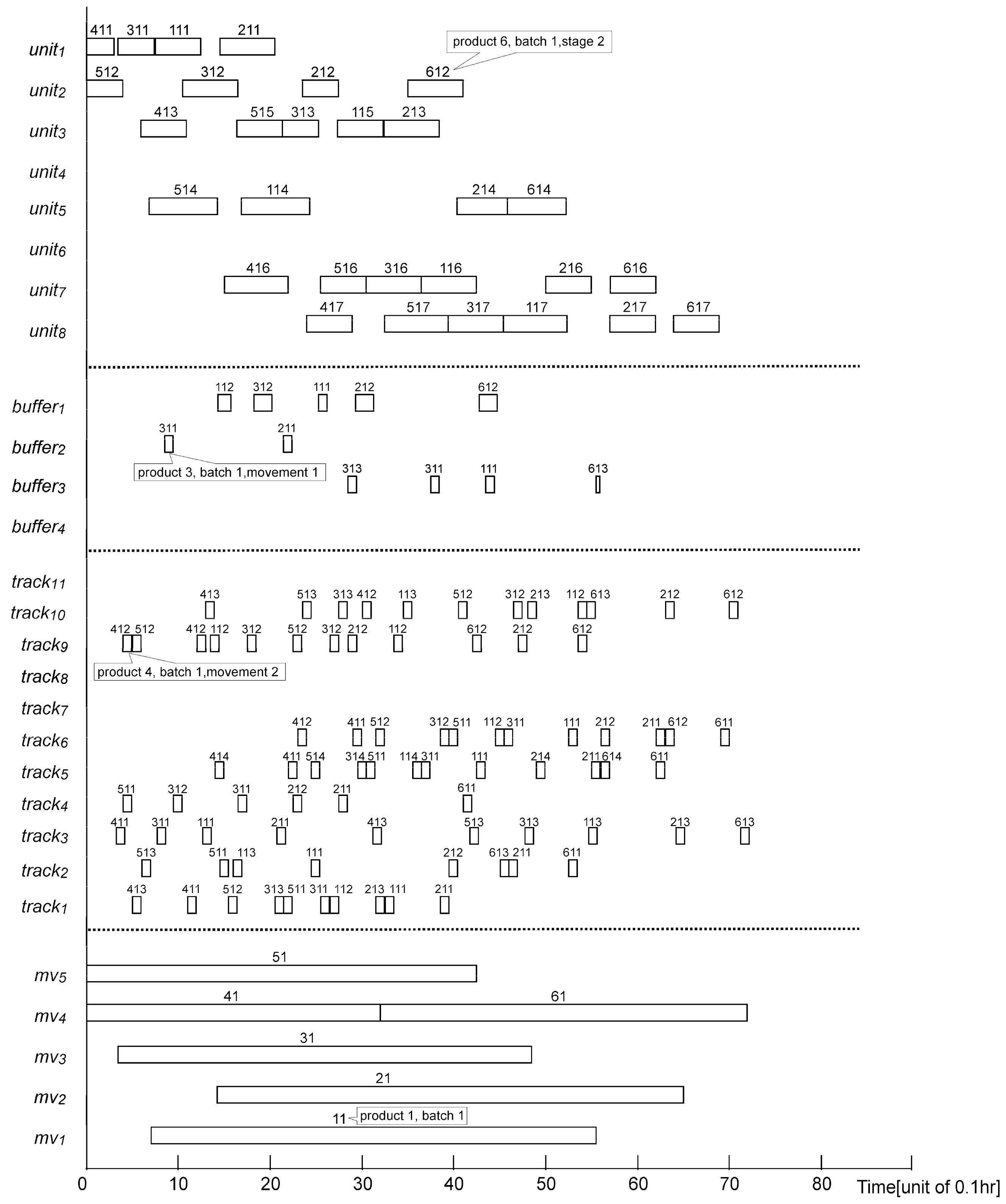

Figure 6. Gantt diagram depicting the best solution for Example 3, considering a herringbone layout.

herringbone layout, this contribution achieved the optimal solution within <50 s, in comparison to almost 16 min required by the method reported by Huang and Chung.[2] Furthermore, this integrated approach is able to generate the optimal schedule for the circular layout, which cannot be achieved using the iterative procedure reported by Huang and Chung.[2] In addition, the proposed method was able to determine the best reported solution for Example 1, considering the linear layout, with very modest computational effort. Nevertheless, in this case, the approach was not able to guarantee the global optimality of the best solution generated within the maximum CPU time enforced for this example.

The best schedule generated for Example 1 with a circular layout is shown in Figure 5. This figure depicts the station's workload and the requirements of moveable vessels. In addition, it illustrates the use of tracks by moveable vessels. Buffer areas are not shown in the diagram, because they are not used in the best solution generated. To clarify the temporal relationships between the different activities used in the model, the entire routing of batch 1 of product 1 is explicitly marked (dotted lines) in the figure. As can be seen, *track1*, *track9*, and *track2*, which correspond to the subpaths between *unit1* and *unit2* in Figure 4b, are used by the mv_1 moveable vessel to execute the movement of batch 1 of product 1 from stage 1 to stage 2.

6.2. Examples 2 and 3. Tables 5 and 6 present the results for Examples 2 and 3, respectively, which consider different numbers of products. As can be seen in Table 1, Example 3 is bigger than Example 2, in terms of the number of products. In addition, products of Example 3 have diverse production flows, depending on the recipe of each one. Note, however, that, because the DFS search mechanism tested in this contribution rendered a very poor computational performance in all instances of Example 1, Tables 5 and 6 only show computational results corresponding to the CP model with the GLEU-VDR strategy. In addition, no comparisons were made with the work of Bok and Park,[6] because their formulation did not take into account moveable vessel collisions when they are operated simultaneously.

As can be observed in Table 5, the CP formulation rendered optimal solutions for all instances of Example 2 within <60 CPU s. Moreover, good initial solutions were obtained with very low computational effort. The analysis of Table 6 indicates that, because of the high combinatorial complexity of the problem addressed, the global optimality of the solutions could not be proven for Example 3. Nevertheless, highly efficient solutions in terms of resource utilization, given by a low makespan, were achieved within the 5400-s time limit enforced for all case studies.

Figure 6 shows the best solution for Example 3 considering a herringbone layout when adopting the GLEU-VDR strategy during the solution process. As it can be seen from this figure, buffers 1, 2, and 3 were employed. Several storage tasks, which occur between two successive transport activities, are depicted. The storage activity denoted as "311" in Figure 6 refers to the wait of batch 1 of product 3 after the first transport activity over this batch.

6.3. Computational Performance Discussion. The proposed CP approach was tested with three different examples concerning a seven-stage pipeless plant. The test problems consider different numbers of batches per product and three types of alternative plant layouts. An analysis of the results of the nine case studies addressed in this section reveals that optimal schedules were generated in six of the instances that were solved. In the remaining three, optimality could not be proved, but efficient solutions were generated within a time limit of 5400 s. Consequently, the proposed approach showed the advantage of achieving very good solutions in reasonable CPU times for all the cases addressed.

Results presented in Tables 2−6 can also be analyzed from the point of view of the different layouts and their associated computational difficulty. Solutions of the herringbone layout related Examples 1 and 2 were obtained in shorter CPU times.

Finally, it should be remarked that the performance of the proposed integrated CP approach surpasses that of the one of Huang and Chung.[2] In particular, the new approach introduced in this paper was able to reach the same optimal solutions in shorter CPU times for two of the problem instances and it achieved a much better schedule for the remaining case study.

7. Conclusions

An integrated Constraint Programming (CP)-based approach for the scheduling of pipeless plants has been introduced. The method is composed of two parts: a rigorous model and a proper search strategy. The model is able to simultaneously address the scheduling and routing problems that arise in pipeless plants. In addition, it can easily represent different plant layouts in a rigorous manner. Therefore, the major contributions of this work are (a) the development of an integrated model, considering, at the same time, the production and material handling scheduling problems for pipeless plants, and (b) the incorporation of an efficient search methodology, which directly impacts on the computational performance of the CP approach.

It is important to highlight that, in most of the case studies, the proposed integrated approach showed better computational performance, in comparison to the results reported by previous contributions. Despite the fact that optimal solutions could not been proved for some problem instances within the maximum time limit enforced, highly efficient solutions were achieved for all of them with reasonable computational effort, which represents very attractive conditions for addressing real-world industrial problems.

Acknowledgment

Financial support received from FONCYT-ANPCyT (under Grant PICT 2006-01837), from CONICET (under Grant PIP-2221), and from Universidad Nacional del Litoral (under Grant PI-66-337) are fully appreciated.

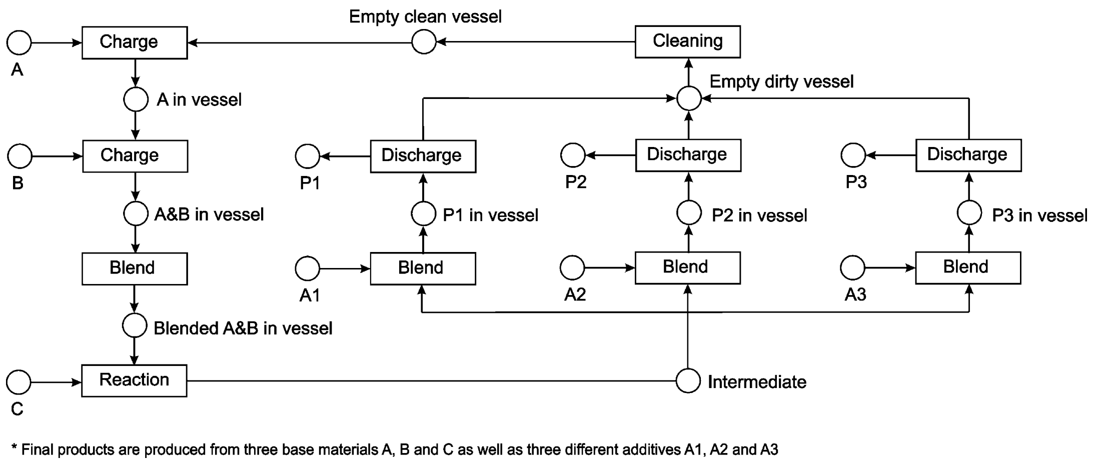

Figure A1. State task network for Examples 1 and 2.

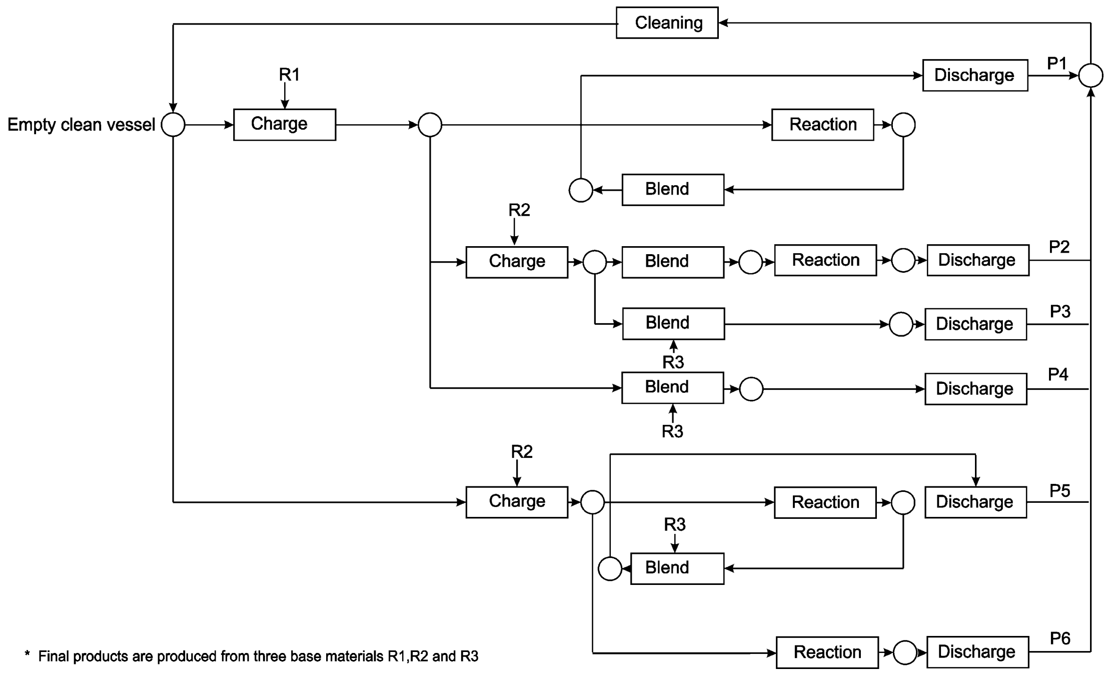

Figure A2. State task network for Example 3.

Table A1. Relations between Tasks and Processing Units for Examples 1, 2, and 3

production task	available units
charge A	charger 1 (*unit1*)
charge B	charger 2 (*unit2*)
blend A + B	blender 1 (*unit3*) and blender 2 (*unit4*)
reaction	reactor 1 (*unit5*) and reactor 2 (*unit6*)
blend additives	blender 1 (*unit3*) and blender 2 (*unit4*)
discharge	discharger (*unit5*)
cleaning	cleaner (*unit6*)

Table A2. Processing Time for Example 1

	Processing Time (Hours)							
product	*unit1*	*unit2*	*unit3*	*unit4*	*unit5*	*unit6*	*unit7*	*unit8*
1	0.5	0.5	0.8	0.8	0.5	0.5	0.5	0.5
2	0.5	0.5	0.8	0.8	0.5	0.5	0.5	0.5
3	0.5	0.5	0.8	0.8	0.5	0.5	0.5	0.5

Table A3. Processing Time for Example 2

	Processing Time (Hours)							
product	*unit1*	*unit2*	*unit3*	*unit4*	*unit5*	*unit6*	*unit7*	*unit8*
1	0.6	0.5	0.5	0.85	0.85	0.6	0.5	0.5
2	0.5	0.5	0.7	0.75	0.75	0.5	0.5	0.5
3	0.5	0.6	0.5	0.85	0.85	0.6	0.5	0.5

Table A4. Processing Time for Example 3

	Processing Time (Hours)							
product	*unit1*	*unit2*	*unit3*	*unit4*	*unit5*	*unit6*	*unit7*	*unit8*
1	0.5	0.5	0.5	0.75	0.75	0.75	0.6	0.7
2	0.6	0.4	0.6	0.65	0.55	0.55	0.5	0.5
3	0.4	0.6	0.4	0.45	0.75	0.75	0.6	0.6
4	0.3	0.6	0.5	0.75	0.65	0.65	0.7	0.5
5	0.5	0.4	0.5	0.65	0.75	0.75	0.5	0.7
6	0.5	0.6	0.6	0.75	0.65	0.65	0.5	0.5

Appendix A

Table A1 gives the relationships between the tasks and processing units for Examples 1, 2, and 3. Figure A1 shows the state task network for Examples 1 and 2, while Figure A2 shows the state task network for Example 3. The processing times for Examples 1, 2, and 3 are given in Tables A2, A3, and A4, respectively. The suitability of moveable vessels is given in Table A5.

Table A5. Suitability of Moveable Vessels

product	Example 1	Example 2	Example 3
1	mv_1 and mv_2	mv_1 and mv_2	mv_1–mv_5
2	mv_3	mv_1 and mv_2	mv_1–mv_5
3	mv_1 and mv_2	mv_1 and mv_2	mv_1–mv_5
4			mv_1–mv_5
5			mv_1–mv_5
6			mv_1–mv_5

Literature Cited

(1) Méndez, C. A.; Cerdá, J.; Harjunkoski, I.; Grossmann, I. E.; Fahl, M. State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Comput. Chem. Eng.* **2006**, *30* (6), 913.

(2) Huang, W.; Chung, P. W. H. Integrating routing and scheduling for pipeless plants in different layouts. *Comput. Chem. Eng.* **2005**, *29* (5), 1069.

(3) Pantelides, C. C.; Realff, M. J.; Shah, N. Short-term scheduling of pipeless batch plants. *Chem. Eng. Res. Des.* **1995**, *73* (A4), 431.

(4) Realff, M. J.; Shah, N.; Pantelides, C. C. Simultaneous design, layout and scheduling of pipeless batch plants. *Comput. Chem. Eng.* **1996**, *20* (6/7), 869.

(5) Gonzalez, R.; Realff, M. J. Operation of pipeless batch plants. I. MILP schedules. *Comput. Chem. Eng.* **1998**, *32* (7−8), 841.

(6) Bok, J.; Park, S. Continuous-time modeling for short-term scheduling of multipurpose pipeless plants. *Ind. Eng. Chem. Res.* **1998**, *37*, 3652.

(7) Lee, K. H.; Chung, S.; Lee, H. K.; Lee, I. B. Continuous time formulation of short-term scheduling for pipeless batch plants. *J. Chem. Eng. Jpn.* **2001**, *34* (10), 1267.

(8) Van Hentenryck, P. *The OPL Optimization Programming Language*; The MIT Press: Cambridge, MA, 1999.

(9) Marriot, K.; Stuckey, P. *Programming with Constraints. An Introduction*; The MIT Press: Cambridge, MA, 1999.

(10) Brailsford, S. C.; Potts, C. N.; Smiths, B. M. Constraint Satisfaction Problems: Algorithms and Applications. *Eur. J. Oper. Res.* **1999**, *119*, 557.

(11) *ILOG OPL Studio 3.7 User's Manual, 2003*; ILOG, Inc., 2003.

(12) Nuijten, W. P. M.; Aarts, E. H. L. A computational study of constraint satisfaction for multiple capacitated job shop scheduling. *Eur. J. Oper. Res.* **1996**, *90*, 269.

(13) Le Pape, C. Implementation of resource constraints in ILOG schedule: A library for the development of constrained-based scheduling systems. *Int. Syst. Eng.* **1998**, *3* (2), 55.

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

(15) Harjunkoski, I.; Jain, V.; Grossmann, I. E. Hybrid Mixed-Integer/Constraint Logic Programming Strategies for Solving Scheduling and Combinatorial Optimization Problems. *Comput. Chem. Eng.* **2000**, *24*, 337.

(16) Jain, V.; Grossmann, I. E. Algorithms for hybrid MILP/CP model for a class of optimization problems. *INFORMS J. Comput.* **2001**, *13*, 258.

(17) Harjunkoski, I.; Grossmann, I. E. Decomposition Techniques for Multistage Scheduling Problems Using Mixed-integer and Constraint Programming Methods. *Comput. Chem. Eng.* **2002**, *26*, 1533.

(18) Maravelias, C. T.; Grossmann, I. E. A Hybrid MILP/CP Decomposition Approach for the Continuous Time Scheduling of Multipurpose Batch Plants. *Comput. Chem. Eng.* **2004**, *28*, 1921.

(19) Roe, B.; Papageorgiou, L. G.; Shah, N. A hybrid MILP/CLP algorithm for multipurpose batch process scheduling. *Comput. Chem. Eng.* **2005**, *29*, 1277.

(20) Zeballos, L. J.; Henning, G. P. A CP Approach to the Scheduling of Resource-Constrained Multiproduct Continuous Facilities. *Lat. Am. Appl. Res.* **2006**, *36* (4), 205.

(21) Huang, W.; Chung, P. W. H. Scheduling of pipeless batch plants using constraint satisfaction techniques. *Comput. Chem. Eng.* **2000**, *24* (2−7), 377.

(22) *ILOG Scheduler 5.0 User's Manual*, ILOG, Inc., 2003.

(23) *ILOG Solver 5.0 User's Manual*, ILOG, Inc., 2003.

Received for review July 23, 2009
Revised manuscript received November 26, 2009
Accepted November 26, 2009

IE901176N