CORE DISCUSSION PAPER

2007/26

A tighter continuous time formulation for the cyclic scheduling of a mixed plant

Yves Pochet[1], François Warichet[2]

March 2007

Abstract

In this paper, based on the cyclic scheduling formulation of Schilling and Pantelides [22], we propose a continuous time mixed integer linear programming (MILP) formulation for the cyclic scheduling of a mixed plant, i.e. a plant composed of batch and continuous tasks. The cycle duration is a variable of the model and the objective is to maximize productivity. By using strengthening techniques and the analysis of small polytopes related to the problem formulation, we strengthen the initial formulation by tightening some initial constraints and by adding valid inequalities. We show that this strengthened formulation is able to solve moderate size problems quicker than the initial one. However, for real size cases, it remains difficult to obtain the optimal solution of the scheduling problem quickly. Therefore, we introduce MILP-based heuristic methods in order to solve these larger instances, and show that they can provide quite good feasible solutions quickly.

Keywords : Cyclic scheduling, Continuous time formulation, Mixed integer programming heuristics, Strengthening

[1]CORE & IAG, Université Catholique de Louvain, Belgium, pochet@core.ucl.ac.be

[2]CORE & INMA, Université Catholique de Louvain, Belgium, warichet@core.ucl.ac.be

This work was supported by le Fond de Recherche Solvay. This work was partly carried out within the framework of ADONET, a European Network in Algorithmic Discrete Optimization, contract no. MRTN-CT-2003-504438. This text presents research results of the Belgian Program on Interuniversity Poles of Attraction initiated by the Belgian State, Prime Minister's Office, Science Policy Programming. The scientific responsibility is assumed by the authors.

1 Introduction

The problem that we address in this paper is the optimization of the cyclic schedule of a mixed plant in order to maximize its productivity. A mixed plant is composed of batch and continuous tasks. A batch task has a fixed processing time and produces at the end a fixed amount of product. A continuous task is processed continuously and its decision variable is the speed at which it is performed. Both types of tasks consume resources with limited capacity or availability.

Two types of formulations are typically proposed in the literature to model such scheduling problems as mixed integer programs : the discrete time formulation and the continuous time formulation.

Initially, scheduling problems were modeled by discrete time formulations (see for example Kondili et al. [12]) using time intervals of fixed duration and a state task network representation to model the process network. To better model the various types of resources used, the resource task network was introduced (see Pantelides [18]) to generalize the concept of state task network. Typically discrete time formulations require a large number of small time intervals to model the problem accurately and obtain realistic solutions. This gives rise to large size models in terms of number of variables. However, given that the formulation is usually strong (i.e. the duality gap is rather small), one can hope to solve moderate size problems to near optimality.

Even though the number of variables is very large, the number of time intervals in which an event occurs in an optimal solution is usually quite small. Therefore, continuous time formulations were proposed to reduce the size of the formulation, see among others Zhang and Sargent [28], Pinto and Grossmann [19] and Mockus and Reklaitis [16]. The characteristic of these types of formulations is that time intervals have variable duration. Consequently, the number of time intervals required to model the scheduling problem accurately can be much smaller and is close to the number of events that really occur. The continuous time formulations can be based on time slots or on events.
For the slot-based formulation, time is decomposed into a set of consecutive time slots of variable duration and a batch task is assigned to a set of consecutive time slots. The representation of time can be global, i.e synchronized for all units (see for example Schilling and Pantelides [21]) or unit specific,

i.e asynchronous (see for example Karimi and McDonald [10]).
For the event-based formulations, the variables correspond to the starting and the ending time of each batch task expressed in absolute time units, and model events that occur at different moments. Here also, the representation of time can be global (see for example Zhang and Sargent [28] and Mockus and Reklaitis [16]) or unit specific (see for example Ierapetritou and Floudas [8]).
The main interest of continuous time formulations is that the number of variables is very small. The mixed integer program is compact but typically weak in the sense that the duality gap is large, and therefore the number of Branch and Bound nodes needed to obtain the optimal solution is large. This is due, very often, to the necessary introduction of so-called big M type of constraints to obtain a correct model formulation.

For recent literature reviews about scheduling formulations for chemical processing systems, see Floudas and Lin [7] and Mendez et al. [15].

In the problem studied here, the product demand is relatively stable which implies that we are interested by an optimal cyclic schedule where the objective is to maximize the long term productivity. In the literature, this cyclic scheduling problem has been modeled by discrete time formulations (see for example Shah et al. [23]) and continuous time formulations (see for example Wu and Ierapetritou [27], Schilling and Pantelides [22]). The paper by Castro et al. [2] models a case study problem using both types of formulations and concludes that the discrete time formulation gives, in a reasonable amount of time, a solution of better quality than the continuous time formulation.

In this paper, we study a continuous time slot-based formulation to model the productivity optimization problem for the cyclic schedule of a mixed plant. In our model, some tasks have to start after others without any waiting time. In such a case, it is not possible to remove the big M constraints as proposed in the paper by Sundaramoorthy and Karimi [25]. Therefore, the continuous time formulation becomes weak.

One way to improve the model formulation is to get rid of the big M constraints by decomposing the problem into an assignment master problem and a sequencing subproblem. In the paper by Maravelias and Grossmann [14], for example, the assignment of production units to tasks is modeled by a mixed integer linear programming (MILP) formulation and the sequenc-

ing subproblem is modeled and solved by constraint programming (CP). Another related decomposition approach is proposed by Maravelias in [13]. The assignment problem is also modeled as a mixed integer linear program, but the feasibility check and the deduction of feasible schedules are performed by combinatorial sequencing algorithms.

Another way to improve model formulations with big M constraints is to tighten the model formulation. In general, to tighten a formulation , we can use strong or facet defining valid inequalities for the problem studied (see Nemhauser and Wolsey [17]). We can also tighten formulations by using an extended space of variables (see Pochet and Wolsey [20]) or by strengthening techniques (see Andersen and Pochet [1]).

Our first contribution is to show that by using a tightened continuous time formulation, we solve problem instances quicker and with less Branch-and-Bound nodes than if we use the initial continuous time formulation. The tightened formulation is obtained by using a combination of strengthening techniques and the analysis of small polytopes, see Christof and Loebel [3], applied to the formulation of the scheduling of the batch tasks.

We also tried to improve the model formulation of the continuous part of the problem. The results obtained so far suggest that only one of the valid inequalities found can help to solve the problem instances more rapidly, see Warichet and Pochet [26].

Using this tightened formulation, we detect that for many instances, the CPU solution time and the total number of nodes in the branch and bound algorithm are drastically reduced. Nevertheless the time needed in some cases to find the optimal solution remains very long because of the difficulty of finding good feasible solutions even though the duality gap is small. Therefore, we pay attention to some MILP heuristic techniques in order to obtain good feasible solutions quickly, while trying to take advantage of the improved formulation obtained. The MILP heuristic methods can be subdivided into two groups. The fist type are the construction heuristic methods that construct a feasible solution from scratch (LP-and-Fix or Cut-and-Fix, Relax-and-Fix (see Stadtler [24]), ...) and the second type are the improvement heuristic methods that try to improve some initial feasible solution (Relaxation Induced Neighborhood Search (RINS) (see Danna et al. [4]), Local Branching (LB) (see Fischetti and Lodi [6]), Exchange (EXCH)). For more details about these heuristic methods, see for instance Pochet and

Wolsey [20].

In Kelly and Mann [11], the use of a Relax-and-Fix heuristic is proposed in order to speed up the resolution of production scheduling problems. They showed that with this heuristic, it is possible to find a good feasible solution quickly.

Our second contribution is that the heuristic method used, a combination of various well-known MILP heuristic methods such as Relax-and-Fix and Local Branching, allows us to find better feasible solutions than exact solution methods for a fixed computing time and for some large instances.

The outline of the paper is the following. In Section 2, we present an initial formulation of our scheduling problem. We show in Section 3, how some of the constraints can be strengthened. We define in Section 4 the heuristic methods used. In Section 5, we present some computational results and eventually in the last section, we conclude and propose some directions for future work.

2 Initial model formulation

The objective is to obtain a cyclic schedule of the mixed plant maximizing its productivity. The mixed plant is composed of batch and continuous tasks that consume resources. The formulation of the scheduling problem has to take into account the limited availability of the resources. The initial formulation proposed and the time decomposition are related to the model formulation presented in Schilling and Pantelides [22].

For the continuous time formulation used here, the time is decomposed in a fixed and finite number of time slots and the duration of every time slot is a variable of the model. The decomposition of time is common to all processing units. In the literature, this is known as a global time slot based model.
We represent in Figure 1 the time decomposition into time slots. An event is here defined as the beginning or end of a batch task, and a time slot is the time between two events. Events and time slots are numbered from 1 up to T. In cyclic scheduling, the event at the end of time slot T coincides with the event occurring at the beginning of time slot 1, and is numbered as event 1.

In this section, we define the sets and the variables used and we present

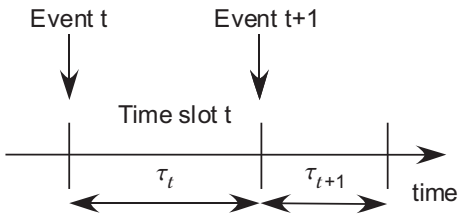

Figure 1: Event and Time slot

an initial formulation for our problem.

2.1 Sets and Variables

Taking into account the process description, we model batch and continuous tasks. We are given a set of batch tasks BT, a set of continuous tasks CT and a maximum number T of events. For generality, we suppose that the capacity of each resource is represented using its own unit of measure that we call ru. The size of a batch of task i is denoted by $BS_i[ru]$ and expresses the quantity of product produced at the end of the task. The processing time of a batch task $i \in BT$ is constant and is given by $p_i[h]$. The continuous tasks $j \in CT$ are performed continuously. The lower and upper bounds on the speed or processing rate of the continuous task $j \in CT$ are $\underline{\rho}_j[ru/h]$ and $\overline{\rho}_j[ru/h]$ respectively. There are precedence constraints between some definite batch tasks and some resources $r \in \{1, \ldots, R\}$ are shared.

The four types of indices and sets are the following :

i: is the index of a batch task, $i \in BT$

j: is the index of a continuous task, $j \in CT$

t: is the index of a time slot, $t \in \{1, \ldots, T\}$

r: is the index of a resource, $r \in \{1, \ldots, R\}$

The five types of decision variables are the following :

τ_t: is the duration of time slot t $[h]$, $\tau_t \geq 0$.

$z_{i,t,t'}$: =1 if a batch of task i starts at the beginning of time slot t and finishes at the end of time slot t',
=0 otherwise.

$q_{j,t}$: is the quantity processed by the continuous task j during time slot t $[ru]$, $q_{j,t} \geq 0$.

$r_{r,t}$: is the rate $[ru/h]$ or the quantity $[ru]$ of resource r available at the beginning of time slot t just after the event occuring at the beginning of time slot t, $r_{r,t} \geq 0$.

$rf_{r,t}$: is the rate $[ru/h]$ or the quantity $[ru]$ of resource r available at the end of time slot t just before the event occuring at the end of time slot t, $rf_{r,t} \geq 0$.

To reduce the number of binary variables $z_{i,t,t'}$, we impose the following restriction. A batch task i can only be performed during a maximum number of consecutive time slots denoted by d. This means that if task i starts at time slot t, this task has to end at or before time slot $t+d-1$. This implies that $z_{i,t,t'}$ exists for all $i \in BT, t \in \{1,\ldots,T\}$ and $t' \in \{t,\ldots,t+d-1\}$.

We also impose, without loss of generality, $\tau_t \leq p^{\max}$ for all $t \in \{1,\ldots,T\}$, where $p^{\max} = \max_{i \in BT} p_i$ is the maximum processing time among all batch tasks.

2.2 Initial continuous time formulation

We model a cyclic schedule. In general, it is possible that a task starts in the current cycle and finishes in the next cycle. As observed in Shah et al. [23], it is possible in this case to optimize the schedule over one cycle by using the concept of task 'wrap-around', because the end of the task in the next cycle has a corresponding end in the beginning of the current cycle. We use the 'wrap-around' time operator $\Omega(t)$ defined in Schilling and Pantelides [22] :

$$\begin{aligned} &\Omega(t) = t \qquad \forall t : 1 \leq t \leq T, \\ &\Omega(t) = \Omega(t-T) \text{ for } t > T, \\ &\Omega(t) = \Omega(t+T) \text{ for } t < 1. \end{aligned}$$

So, a batch task i starting in time slot t and finishing in time slot $t+k$ corresponds to $z_{i,t,\Omega(t+k)} = 1$ as illustrated in Figure 2. We present now the various type of constraints that we need in order to model the problem.

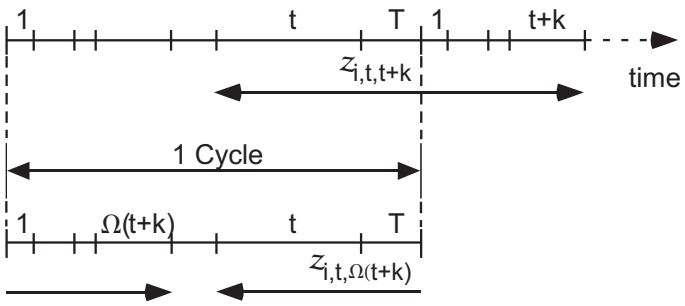

Figure 2: The 'wrap-around' time operator $\Omega(t)$

2.2.1 Duration constraints for batch tasks

The following timing constraints (1)-(2) impose that if a batch task i starts at time slot t and finishes at time slot l then the sum of the time slot durations from t to l is equal to the processing time of batch task i, denoted by p_i. The constant $p^{\max}$ allows one to obtain a non positive right hand side in (2) when $z_{i,t,\Omega(l)} = 0$, because $\tau_t \leq p^{\max}$ holds for all t.

For all $i, t, l : i \in BT, 1 \leq t \leq T, t \leq l \leq t + d - 1$, we have that

$$p_i z_{i,t,\Omega(l)} \leq \sum_{k=t}^{l} \tau_{\Omega(k)} \tag{1}$$

$$p_i z_{i,t,\Omega(l)} \geq \sum_{k=t}^{l} \tau_{\Omega(k)} - p^{\max}(l - t + 1)\left(1 - z_{i,t,\Omega(l)}\right). \tag{2}$$

Constraint (2) is called a big M constraint where the big M constant is $p^{\max}(l - t + 1)$. This big M constraint is usually not strong (non facet-defining).

The constraints (3)-(4) impose that we can start or finish only one batch task at each time slot. When adding this type of constraints, one needs to increase the size of the formulation (larger T) in order to obtain an equivalent formulation because we limit to one the number of tasks starting or finishing at each time event. However, as observed in practice, this new formulation is stronger, i.e. the duality gap is smaller, and allows one to strengthen the

formulation of the timing constraints (see Section 3.1).

$$\sum_{i \in BT} \sum_{l=t}^{t+d-1} z_{i,t,\Omega(l)} \leq 1 \qquad \forall t : 1 \leq t \leq T \tag{3}$$

$$\sum_{i \in BT} \sum_{t=l-d+1}^{l} z_{i,\Omega(t),l} \leq 1 \qquad \forall l : 1 \leq l \leq T \tag{4}$$

Note that constraints (3)-(4) are facet-defining for the subproblem in which we consider only one batch task and no resource restriction.

2.2.2 Processing constraints for the continuous tasks

Constraints (5)-(6) limit the amount processed by the continuous tasks

$$q_{j,t} \leq \overline{\rho}_j \tau_t \qquad \forall j, t : j \in CT, 1 \leq t \leq T \tag{5}$$

$$q_{j,t} \geq \underline{\rho}_j \tau_t \qquad \forall j, t : j \in CT, 1 \leq t \leq T \tag{6}$$

where $\underline{\rho}_j[ru/h]$ and $\overline{\rho}_j[ru/h]$ are the lower and the upper bounds for the rate of material that is processed by the continuous task j, respectively.

Once started, the continuous tasks have to be active all the time. In the case of a cyclic schedule, this means that the continuous tasks have to be active all the time.

2.2.3 Resource constraints

As proposed in Pantelides [18], the problem formulation is based on the resource task network (RTN) representation. The set {1,...,R} of resources r is composed of all the materials (products produced, consumed, or stored into storage tanks) (R_m), utilities (R_u), and processing equipments (production units) (R_e) involved in the process. The precedence constraints between batch tasks are modeled by resource constraints as well. A detailed example is given in Section 5.1.

The general material balance constraint can be written as follows. It determines the resource availability level, for each resource r, at the transition from the end of the time slot $t-1$ to the start of time slot t. Only batch tasks produce or consume resources at these events.

$$r_{r,t} = rf_{r,\Omega(t-1)} + \sum_{i \in BT} \bar{\mu}_{i,r} \sum_{t'=t-d}^{t-1} z_{i,\Omega(t'),\Omega(t-1)} - \sum_{i \in BT} \mu_{i,r} \sum_{t'=t}^{t+d-1} z_{i,t,\Omega(t')} \tag{7}$$

$\forall r, t : 1 \leq r \leq R, 1 \leq t \leq T$ and where

$\bar{\mu}_{i,r}$: is the rate $[ru/h]$ of renewable resource $r \in R_u$ released at the end of task i, or
the quantity $[ru]$ of non renewable resource $r \in R_m$ produced at the end of task i, or
the number of units $[ru]$ of non renewable resource $r \in R_e$ released at the end of task i.

$\mu_{i,r}$: is the rate $[ru/h]$ of renewable resource $r \in R_u$ consumed at the beginning of task i, or
the quantity $[ru]$ of non renewable resource $r \in R_m$ consumed at the beginning of task i, or
the number of units $[ru]$ of non renewable resource $r \in R_e$ consumed at the beginning of task i.

Continuous tasks are in process during time slots and therefore we have to control the level of all the resources $r \in R_m$ related to the continuous tasks at the end of every time slot τ_t just before receiving possible resource releases or consumptions occuring at the beginning of the next time slot. The resource level at the end of every time slot can be expressed as

$$rf_{r,t} = r_{r,t} + \sum_{j \in CT} \lambda_{j,r} q_{j,t} \qquad \forall r, t : r \in R_m, 1 \leq t \leq T \tag{8}$$

where

$$\begin{aligned} \lambda_{j,r} &= 1 \text{ if the continuous task } j \text{ produces resource } r \in R_m \\ &= -1 \text{ if the continuous task } j \text{ consumes resource } r \in R_m \\ &= 0 \text{ otherwise.} \end{aligned}$$

The levels of resources $r \in R_e \cup R_u$ are not changed during time slots, therefore $rf_{r,t} = r_{r,t} \; \forall \; t, \; \forall \; r \in R_e \cup R_u$.

We have to verify at the beginning and at the end of each time slot that the resource capacity limitations are satisfied.

$$Rmin_r \leq r_{r,t} \leq Rmax_r \qquad \forall r, t : 1 \leq r \leq R, 1 \leq t \leq T \tag{9}$$

$$Rmin_r \leq rf_{r,t} \leq Rmax_r \qquad \forall r, t : r \in R_m, 1 \leq t \leq T \tag{10}$$

where $Rmin_r, Rmax_r$ represent the lower and upper limits on the level of resource r, respectively, in $[ru/h]$ for $r \in R_u$ and in $[ru]$ for $r \in R_m \cup R_e$.

However if there exists a time event t at which there exists both a release of resource r at the end of time slot $t-1$ and a consumption of r at the start of time slot t, then we need an additional set of constraints in order to guarantee that the maximum resource capacity usage is satisfied. This is why we have to also impose the following condition

$$rf_{r,t} + \sum_{i \in BT} \bar{\mu}_{i,r} \sum_{t'=t-d+1}^{t} z_{i,\Omega(t'),t} \leq Rmax_r \tag{11}$$

for all $r, t : 1 \leq r \leq R, 1 \leq t \leq T$.

Finally, if for a certain resource $\tilde{r} \in R$, $\bar{\mu}_{i,\tilde{r}} = \mu_{i,\tilde{r}}$, for all $i \in BT$, then we do not use the constraints (7)-(11) for that resource $\tilde{r}$. We only impose the following set of constraints :

$$Rmin_{\tilde{r}} \leq \sum_{i \in BT} \sum_{t1=t-d+1}^{t} \sum_{t2=t}^{t1+d-1} \bar{\mu}_{i,\tilde{r}} z_{i,\Omega(t1),\Omega(t2)} \leq Rmax_{\tilde{r}} \tag{12}$$

for all $t \in \{1, \ldots, T\}$. This occurs typically for the modelling of the availability of the utilities (i.e $\tilde{r} \in R_u$).

Figure 3 represents the two variables $r_{r,t}$ and $rf_{r,t}$ for the three types of resources.

2.2.4 No waiting time between two batch tasks

Some batch tasks have to be performed directly after others without any waiting time. For instance, in a chemical reactor, regulation has to start immediately after heating. This is modeled as follows :

$$\sum_{l=t}^{t+d-1} z_{i',t,\Omega(l)} \geq \sum_{l=t-d}^{t-1} z_{i,\Omega(l),\Omega(t-1)} \tag{13}$$

for all i, i' such that i' follows directly i without any waiting time, for all $t \in \{1, \ldots, T\}$.

This constraint is the reason why we need to know or to model exactly the duration of a batch task as the sum of time slots durations.

2.2.5 Breaking symmetry in the cyclic solution

To remove the symmetry in the cyclic solution, we usually fix the initial starting time slot of one batch task that has to be processed during the

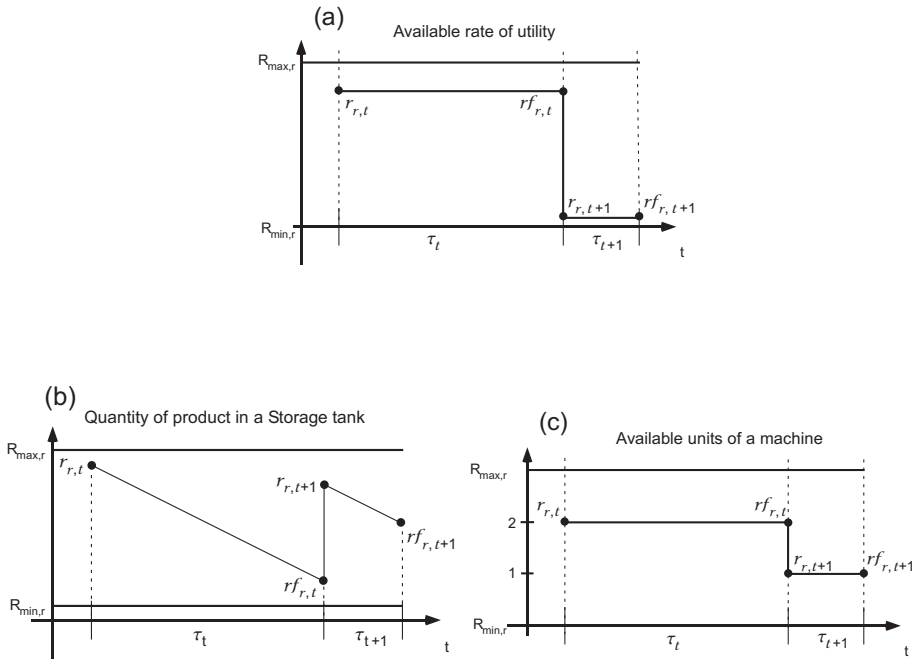

Figure 3: Representation of the resource $r \in R_u$ (a), $r \in R_m$ (b) and $r \in R_e$ (c), respectively.

cyclic schedule.
For example, if we know that a specific batch task i_1 has to be processed during the cyclic schedule, the corresponding constraint will be : $\sum_{t'=1}^{d} z_{i_1,1,t'} = 1$. This can usually be done without loss of generality or optimality.

2.2.6 The objective function

The objective is to maximize the average production per unit of time over the entire cycle, or the productivity. The non linear objective function is the following

$$\max \frac{\sum_{j \in OUT} \sum_{t=1}^{T} q_{j,t}}{\sum_{t=1}^{T} \tau_t}, \tag{14}$$

where $OUT \subseteq CT$ is the set of output products of the plant. It is here assumed that the output products are produced by some of the continuous tasks. This restriction can be relaxed easily.

It was shown by Isbell and Marlow [9], and also extended by Dinkelbach [5], that this nonlinear objective function can be optimized by solving a sequence (iterations $p = 1, 2, \ldots$) of linear optimization problems where the objective function at iteration p is

$$\max \sum_{j \in OUT} \sum_{t=1}^{T} q_{j,t} - \mu^p \sum_{t=1}^{T} \tau_t, \tag{15}$$

and where μ^p is a constant which is computed before iteration p as

$$\mu^p = \frac{\sum_{j \in OUT} \sum_{t=1}^{T} q_{j,t}^{\star,p-1}}{\sum_{t=1}^{T} \tau_t^{\star,p-1}},$$

where $q_{j,t}^{\star,p-1}$, for $j \in OUT$ and for all t, and $\tau_t^{\star,p-1}$, for all t, are the optimal solution of the problem at iteration $p-1$. The initialization is $\mu^1 = \sum_{j \in OUT} \underline{\rho}_j$. This iterative process stops at the end of iteration p when $\mu^{p+1} = \mu^p$. In that case, the optimal objective function (15) is equal to 0. As explained and proved in Isbell and Marlow [9], the corresponding optimal solution is also optimal for the problem with the nonlinear objective function (14). We show in **Appendix A** another way to prove this result.

This sequence of μ^p converges because the value of μ^p increases monotonically and is bounded from above. The sequence of μ^p increases monotonically because if $\mu^{p+1} < \mu^p$ then

$$\sum_{j \in OUT} \sum_{t=1}^{T} q_{j,t}^{*,p} - \mu^p \sum_{t=1}^{T} \tau_t^{*,p} < 0$$

and this implies

$$\sum_{j \in OUT} \sum_{t=1}^{T} q_{j,t}^{*,p} - \mu^p \sum_{t=1}^{T} \tau_t^{*,p} < 0 = \sum_{j \in OUT} \sum_{t=1}^{T} q_{j,t}^{*,p-1} - \mu^p \sum_{t=1}^{T} \tau_t^{*,p-1}$$

and therefore $(q^{*,p}, \tau^{*,p})$ cannot be optimal for iteration p because it is dominated by $(q^{*,p-1}, \tau^{*,p-1})$, and this is a contradiction.
Moreover, the value of the μ's is bounded from above by the upper bound on the rate of material that is processed by the continuous tasks in the set OUT, that is $\sum_{j \in OUT} \overline{\rho}_j$.

Finally, as explained in Isbell and Marlow [9], this algorithm converges in a finite number of iterations. The optimal solution of each problem with the linear objective function (15) is a vertex of the feasible set and there are only a finite number of vertices for this set. If the same vertex is returned for two successive iterations, the corresponding μ's are equivalent and the algorithm terminates.

3 Strengthened continuous time formulation

In this section, we describe the improved formulation of some constraints and some valid inequalities added to tighten the MILP formulation. The proofs of validity and the detailed derivation for the various results of this section are given in Warichet and Pochet [26].
These results have been obtained by the analysis and generalization of the reformulation of small polytopes (see Christof and Loebel [3]), and by strengthening techniques such as the one described in Andersen and Pochet [1].

3.1 Strengthening of the timing constraints for the batch tasks

The initial timing constraint (1)

$$\sum_{k=t}^{l} \tau_{\Omega(k)} \geq p_i z_{i,t,\Omega(l)} \qquad \forall i \in BT, \forall t,l : 1 \leq t \leq T, t \leq l \leq t+d-1$$

can be strengthened as

$$\sum_{k=t}^{l} \tau_{\Omega(k)} \geq \sum_{i \in BT} \sum_{k=t|t\neq l}^{l} p_i z_{i,t,\Omega(k)} + \sum_{i \in BT} p_i z_{i,l,l} \tag{16}$$

$$\sum_{k=t}^{l} \tau_{\Omega(k)} \geq \sum_{i \in BT} \sum_{k=t|t\neq l}^{l} p_i z_{i,\Omega(k),l} + \sum_{i \in BT} p_i z_{i,t,t} \tag{17}$$

for all $t,l : 1 \leq t \leq T, t \leq l \leq t+d-1$.

Constraints (16) and (17) are both facet-defining for the subproblem in which we consider only one batch task and no resource restriction. Note that the size of the formulation has been reduced by avoiding the definition

of separate timing constraints for each batch task $i \in BT$.

To illustrate the difference between the initial and the strengthened formulation, we assume that we have only one batch task and we write the initial constraint (1) for $t = 1$ and $l = 3$:

$$\tau_1 + \tau_2 + \tau_3 \geq p_1 z_{1,1,3}.$$

The corresponding strengthened expression (16) can be written as

$$\tau_1 + \tau_2 + \tau_3 \geq p_1 \left(z_{1,1,1} + z_{1,1,2} + z_{1,1,3}\right) + p_1 z_{1,3,3}.$$

In Figure 4, taking into account the fact that we can start or finish at most one batch task at each time slot, we can see that we may add the variables corresponding to the dashed intervals in the right hand side, and keep the inequality valid. This holds because $z_{1,1,3} + z_{1,1,2} + z_{1,1,1} \leq 1$, and if $z_{1,1,3} + z_{1,1,2} + z_{1,1,1} = 1$ then $\tau_1 + \tau_2 + \tau_3 \geq p_1$. Similarly, if $z_{1,1,3} + z_{1,3,3} = 1$ then we must also have $\tau_1 + \tau_2 + \tau_3 \geq p_1$. Finally, when $z_{1,1,1} + z_{1,1,2} + z_{1,1,3} + z_{1,3,3} = 2$ (for instance when $z_{1,1,1} = z_{1,3,3} = 1$) then the two batches do not overlap and we must have $\tau_1 + \tau_2 + \tau_3 \geq 2p_1$.

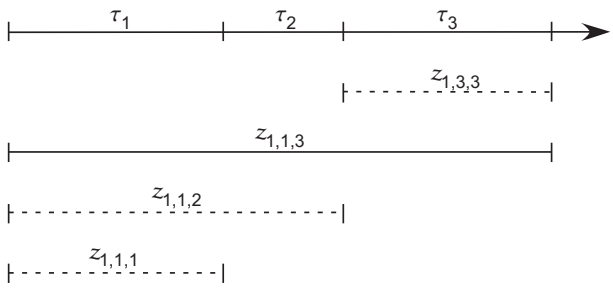

Figure 4: Strengthened constraint (16) : an example

By using the same idea, we can generalize this result even more by considering that if there are two (or more) batch tasks, we can sum their processing times in the right hand-side :

$$\begin{aligned} \tau_1 + \tau_2 + \tau_3 \quad & \geq p_1 \left(z_{1,1,1} + z_{1,1,2} + z_{1,1,3}\right) + p_1 z_{1,3,3} \\ & + p_2 \left(z_{2,1,1} + z_{2,1,2} + z_{2,1,3}\right) + p_2 z_{2,3,3}. \end{aligned}$$

The second strengthened inequality (17) is closely related to the first one and can be explained in the same way.

Similarly, the timing constraint (2)

$$\sum_{k=t}^{l} \tau_{\Omega(k)} \leq p_i z_{i,t,\Omega(l)} + p^{\max}(l-t+1)\left(1 - z_{i,t,\Omega(l)}\right)$$

for all $i, t, l : i \in BT, 1 \leq t \leq T, t \leq l \leq t+d-1$ can be strengthened as

$$\begin{aligned}
&\sum_{k=t}^{l} \tau_{\Omega(k)} \leq \sum_{i \in BT} \sum_{k=l-d+1}^{l} p_i z_{i,\Omega(k),\Omega(l)} \\
&+p^{\max}\Big((l-t+1) - \sum_{i \in BT} \sum_{k=l-d+1}^{l} \min\{l-t+1, l-k+1\} z_{i,\Omega(k),\Omega(l)}\Big) \\
&+ \sum_{i \in BT} (p_i - p^{\max}) \sum_{t'=t|l-t+1=d}^{l-1} z_{i,t,\Omega(t')} \qquad (18)
\end{aligned}$$

for all $t, l : 1 \leq t \leq T, t \leq l \leq t+d-1$, and

$$\begin{aligned}
&\sum_{k=t}^{l} \tau_{\Omega(k)} \leq \sum_{i \in BT} \sum_{k=t}^{t+d-1} p_i z_{i,t,\Omega(k)} \\
&+p^{\max}\Big((l-t+1) - \sum_{i \in BT} \sum_{k=t}^{t+d-1} \min\{l-t+1, k-t+1\} z_{i,t,\Omega(k)}\Big) \\
&+ \sum_{i \in BT} (p_i - p^{\max}) \sum_{t'=t+1|l-t+1=d}^{l} z_{i,\Omega(t'),\Omega(l)} \qquad (19)
\end{aligned}$$

for all $t, l : 1 \leq t \leq T, t \leq l \leq t+d-1$.

Constraints (18) and (19) are facet-defining for the subproblem in which we consider only one batch task and no resource restriction. Again, the size of the formulation has been reduced compared to (2) when there are multiple batch tasks.

The two strengthened inequalities are closely related and we explain here intuitively how to interpret one of them. When $l < t+d-1$, the general

form of the inequality (19) is

$$\sum_{k=t}^{l} \tau_{\Omega(k)} \leq \sum_{i \in BT} \sum_{k=t}^{t+d-1} p_i z_{i,t,\Omega(k)} +$$
$$p^{\max}\Big((l-t+1) - \sum_{i \in BT} \sum_{k=t}^{t+d-1} \min\{l-t+1, k-t+1\} z_{i,t,\Omega(k)}\Big) \quad (20)$$

for all $t, l : 1 \leq t \leq T, t \leq l < t+d-1$.

To illustrate the difference between the initial and the strengthened formulation when $l < t+d-1$, we assume that we have only one batch task and we write the initial constraint (2) for $t = 1$, $l = 2$ and $d = 3$:

$$\tau_1 + \tau_2 \leq p_1 z_{1,1,2} + p^{\max}(2 - 2z_{1,1,2}). \quad (21)$$

The corresponding strengthened expression (20), can be written as

$$\tau_1 + \tau_2 \leq p_1(z_{1,1,1} + z_{1,1,2} + z_{1,1,3}) + p^{\max}(2 - z_{1,1,1} - 2z_{1,1,2} - 2z_{1,1,3})$$

In Figure 5, taking into account the fact that we can start at most one batch task at each time slot, we can see that we may subtract in the right hand side of (21) a positive multiple of the variables corresponding to the dashed intervals, and keep the inequality valid. This holds because $Z = z_{1,1,1} + z_{1,1,2} + z_{1,1,3} \leq 1$ and if $Z = 0$, then $\tau_1 + \tau_2 \leq 2p^{\max}$ by the initial upper bound on τ_t. Similarly, if $Z = 1$ and $z_{1,1,1} = 1$ then $\tau_1 = p_1$ and $\tau_2 \leq p^{\max}$ and finally if $Z = 1 = z_{1,1,2} + z_{1,1,3}$ then $\tau_1 + \tau_2 \leq p_1$ because a batch of task 1 starts in $t = 1$ and finishes at or after $l = 2$.

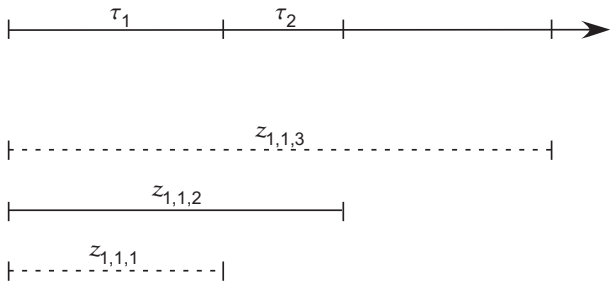

Figure 5: Strengthened constraint (20) : an example

If there are several batch tasks, the inequality (20) becomes

$$\begin{aligned}\tau_1 + \tau_2 \leq \quad & p_1(z_{1,1,1} + z_{1,1,2} + z_{1,1,3}) + p^{\max}(2 - z_{1,1,1} - 2z_{1,1,2} - 2z_{1,1,3}) \\ & + p_2(z_{2,1,1} + z_{2,1,2} + z_{2,1,3}) + p^{\max}(-z_{2,1,1} - 2z_{2,1,2} - 2z_{2,1,3})\end{aligned}$$

which is stronger because the terms added in the right hand side are non positive.

When $l = t + d - 1$, one term from (19) is missing in inequality (20). We explain now that, when $l = t + d - 1$, we can introduce the term $\sum_{i \in BT}(p_i - p^{\max}) \sum_{t'=t+1}^{l} z_{i,\Omega(t'),\Omega(l)}$ in the right hand side of (20) and keep the inequality valid.

We continue with an example involving one batch task. Suppose that $d = 3, t = 1$ and $l = 3$ $(= t + d - 1)$, the strengthened inequality (19) for that example can be written as

$$\begin{aligned}\tau_1 + \tau_2 + \tau_3 \leq \quad & p_1(z_{1,1,1} + z_{1,1,2} + z_{1,1,3}) + p^{\max}(3 - z_{1,1,1} - 2z_{1,1,2} - 3z_{1,1,3}) \\ & + (p_1 - p^{\max})(z_{1,2,3} + z_{1,3,3})\end{aligned}$$

The only new or additional case to consider is : $z_{1,2,3} + z_{1,3,3} = 1$. In that case, $z_{1,1,2} + z_{1,1,1} \leq 1$ and $z_{1,1,3} = 0$.
Suppose first that $z_{1,1,2} + z_{1,1,1} = 1$. If $z_{1,1,2} = 1, \tau_1 + \tau_2 = p_1$ and $\tau_3 \leq p_1$, then $\tau_1 + \tau_2 + \tau_3 \leq 2p_1$. If $z_{1,1,1} = 1, \tau_1 = p_1$ and $\tau_2 + \tau_3 \leq p_1 + p^{\max}$, then $\tau_1 + \tau_2 + \tau_3 \leq 2p_1 + p^{\max}$.
Suppose now that $z_{1,1,2} + z_{1,1,1} = 0$, then $\tau_1 + \tau_2 \leq 2p^{\max}$, $\tau_3 \leq p_1$ and $\tau_1 + \tau_2 + \tau_3 \leq 2p^{\max} + p_1$.
So the inequality is satisfied in all possible cases.

3.2 Strengthening of the constraints for the continuous tasks

If the continuous tasks $j \in CT$ consume or produce a product stored into a resource $r \in R_m$, i.e if $\lambda_{j,r} \neq 0$ in (8), then the constraint (5) can be strengthened as

$$q_{j,t} \leq \overline{\rho}_j \tau_t - \sum_i \max\Big(p_i \overline{\rho}_j - (Rmax_{r_j} - Rmin_{r_j}); 0\Big) z_{i,t,t} \qquad \forall t \in \{1, \ldots, T\} \tag{22}$$

where r_j is the resource that stores the product produced or consumed by the continuous task j.

The continuous task j has a maximum speed ($\overline{\rho}_j[ru/h]$) and the resource r_j has a limited capacity ($[Rmin_{r_j}; Rmax_{r_j}][ru]$). If a batch task i starts

at time slot t and finishes at time slot t, the duration τ_t of the time slot t is equal to the processing time of the batch task p_i. In that case, if $p_i\overline{\rho}_j > Rmax_{r_j} - Rmin_{r_j}$ then the continuous task cannot be performed at maximum speed and $q_{j,t} \leq Rmax_{r_j} - Rmin_{r_j}$. Constraint (22) imposes this additional restriction.

3.3 Other valid inequalities improving the model formulation

The next valid inequality (23) explicitly takes into account the fact that the batch tasks can be performed in parallel when they can last for several consecutive time slots. The maximum number of consecutive slots on which any batch task can be performed is denoted by d. This implies that any time slot can be used by various batch tasks performed in parallel. This time slot is in a sense "shared" by the various batch tasks.

We define the discrete array $coeff$, for fixed $t \in \{1, \ldots, T\}$ and $l \in \{t+2, \ldots, t+T+d-3\}$. For each $k \in \{t, \ldots, l\}$, $coeff_k$ represents the maximum number of batch tasks performed completely (i.e., starting and finishing) in the interval $\{t, \ldots, l\}$ that can be active during time slot k, and is expressed as

$$\begin{aligned} coeff_k = \quad & \min(k-t+1; l-k+1) \qquad \forall k \in \{t, \ldots, l\} \\ = \quad & 0 \text{ otherwise.} \end{aligned}$$

Then, we can add the following valid inequality for $1 \leq t \leq T$ and $t \leq l \leq t+d-1$, (see Warichet and Pochet [26])

$$\sum_{k=t}^{l} \min(coeff_k, d)\tau_{\Omega(k)} \geq \sum_{i \in BT} p_i \left(\sum_{k=t}^{l} \sum_{f=k}^{\min(k+d-1,l)} z_{i,\Omega(k),\Omega(f)} \right). \qquad (23)$$

To illustrate this valid inequality, we consider one batch task, $t = 1$, $d = 3$ and $l = 4$. For this small example, $Coeff = (1, 2, 2, 1)$. Only one batch task can be active during time slot 1 because only one batch task can start at time slot 1. For the time slot 2, 2 batch tasks can be active because we can start a batch task at time slot 1 and finish this batch task after time slot 2 and we can start a batch task at time slot 2 also. For the time slot 3, at most 2 batch tasks can be active because we can finish a batch task at time slot 3 and a batch task at time slot 4. Only one batch task can be active during time slot 4 because only one batch task can finish at time slot

4. A batch task can last only for $d = 3$ consecutive time slots and therefore at most 3 batch tasks can overlap the same time slot. This is why we take in the left-hand side of the inequality (23) the minimum between $coeff_k$ and d. We can write the corresponding valid inequality as

$$\begin{aligned}\tau_1 + 2\tau_2 + 2\tau_3 + \tau_4 \quad \geq \quad & p_1(z_{1,1,1} + z_{1,1,2} + z_{1,1,3} + z_{1,2,2} + z_{1,2,3} \\ & + z_{1,2,4} + z_{1,3,3} + z_{1,3,4} + z_{1,4,4}).\end{aligned}$$

The inequality (23) is valid because the left hand side is the production time available in $\{t, \ldots, l\}$ for batches performed completely in $\{t, \ldots, l\}$ and the right hand side is the production time effectively used by such batches.

4 Heuristic Methods

We introduce below a heuristic method that combines the use of various well known MIP based heuristic methods. Our objective in using MIP based heuristics is to find good feasible solutions quickly by taking advantage of the improved formulations described above. We outline here the well known heuristic methods and we explain how we combine them. All these heuristics are described in Pochet and Wolsey [20].

The main heuristic method used is **Relax-and-Fix**, see Stadtler [24]. The various steps of our specific implementation are the following :

1. We decompose the set of binary variables $z_{i,t,\Omega(l)}$ in various non-disjoint sets $S_1, S_2, \ldots, S_T$. In our case, the set S_t is composed of the binary variables that can be active during time slot t. Every variable will be part of at least one set. In other words, for each $t \leq k \leq l \leq t + d - 1$, the variable $z_{i,t,\Omega(l)} \in S_{\Omega(k)}$.

2. We start by imposing the integrality restriction for the binary variables in the sets S_1 and S_2 and we relax this constraint (i.e. $0 \leq z \leq 1$) for variables in the sets $(S_3 \cup \ldots \cup S_T) \setminus (S_1 \cup S_2)$. We solve the corresponding relaxed MIP problem and we obtain an upper bound for the original problem (for the maximization of the linear objective function).
   We fix the binary variables in the set S_1 at their optimal values.
   This is the end of the first iteration of Relax-and-Fix.

3. Then, we impose the integrality condition for the binary variables in the sets S_2 and S_3 and we relax the integrality constraints for the rest of the variables, i.e. for variables in the sets $(S_4 \cup \ldots \cup S_T) \setminus$

$(S_1 \cup S_2 \cup S_3)$. We solve the corresponding relaxed MIP problem and we fix the binary variables in the set S_2 at their optimal value. This is the end of the second iteration.

4. And so on, up to the last iteration where variables in $(S_{T-1} \cup S_T) \setminus (S_1 \cup \ldots \cup S_{T-2})$ are binary, the other variables being fixed by previous iterations.

The difficulty is that at each step, it is possible that the MIP problem becomes infeasible because of previous variables being fixed at inconsistent values. In such a case, we combine this first heuristic method with a neighborhood search method similar to the **Local Branching** heuristic method (see Fischetti and Lodi [6]). More specifically, if the MIP problem is infeasible at iteration p of Relax-and-Fix, we solve a relaxation of this MIP problem where we impose that the variables previously fixed ($z_{i,t,\Omega(l)} = z^{\star,f}_{i,t,\Omega(l)}$ for $(i,t,l) \in F$) have to remain binary but we allow a limited number k of them to change their values.
This can be modeled by adding the following constraint :

$$\sum_{(i,t,l)\in F | z^{\star,f}_{i,t,\Omega(l)}=0} z_{i,t,\Omega(l)} + \sum_{(i,t,l)\in F | z^{\star,f}_{i,t,\Omega(l)}=1} (1 - z_{i,t,\Omega(l)}) \leq k$$

This additional Local Branching step is performed to improve the robustness of the Relax-and-Fix heuristic method.

An additional way for improving the quality of the feasible solutions obtained by the heuristic is to use the Local Branching heuristic method at the end of the Relax-and-Fix algorithm. It consists in allowing k variables $z_{i,t,\Omega(l)}$ to change their values compared to the final Relax-and-Fix solution. This Local Branching step is repeated until no significant improvement to the objective is obtained. This final Local Branching step defines an improvement heuristic starting from the Relax-and-Fix solution.

5 Computational results

In order to test the efficiency of the strengthened model formulation and the heuristic, we define first a simple test case problem. Then, we present specific valid inequalities for the test case, and some computational results for the test case and some variants of it. Finally, we show how the improved formulation and heuristic can be used to model and solve an industrial larger size scheduling case.

5.1 Test case

The simple test case is composed of one continuous task and of a finite number of units (reactors) denoted by *nbr_unit* on which a set of batch tasks are performed. The reactors are identical and produce the batches in parallel. The batches produced are stored in a limited capacity tank before being processed by the continuous task. The test case is represented in Figure 6. The size of a batch is fixed and is equal to 8 ru.

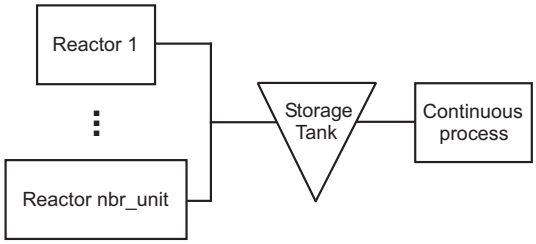

Figure 6: The test case

The polymerization process (I) that takes place in the reactors is subdivided in five consecutive batch tasks : the filling of the reactor (I,1), the heating of the raw material (I,2,h), the exothermic reaction (I,2,r), the cooling (I,2,c) and the discharge (I,3). In our model, the exothermic reaction (I,2,r) is further subdivided in 4 batch subtasks because the consumption of resources varies too much during the temperature regulation task. We assume that the filling of the reactor takes 0.166 h, the heating of the raw material takes 0.4522 h, the exothermic reaction is subdivided in 4 subtasks taking 0.5 h, 0.5 h, 1 h and 1.44125 h respectively, the cooling takes 0.919 h and the discharge 0.166 h.

The resources are the intermediate storage tank (IS), the hot water (H), the cold water (C) and a stock attached to each batch task (a1-a5) in order to model the number of reactors available for performing the corresponding batch task, i.e *nbr_unit* of resources a_1 to a_5 are available in total.
The rate of hot water needed to perform the heating task is 3 $[ru/h]$. The rates of cold water needed to perform the four subtasks of the exothermic reaction are 3.7 $[ru/h]$, 1.64 $[ru/h]$, 0.92 $[ru/h]$ and 0.41 $[ru/h]$, respectively. The rate of cold water needed for the cooling task is 2 $[ru/h]$.

The lower and upper limits on the level of product in the storage tank, on the rate of hot water and cold water are $[0, 15]$ ru,$[0, 3]$ $[ru/h]$ and $[0, 4.2]$

$[ru/h]$, respectively.

The lower and upper bounds for the rate of material that is processed by the continuous task are $\underline{\rho} = 1[ru/h]$ and $\overline{\rho} = 6[ru/h]$, respectively.

The resource task network of the test case problem is represented in Figure 7, where the continuous task is denoted by II.

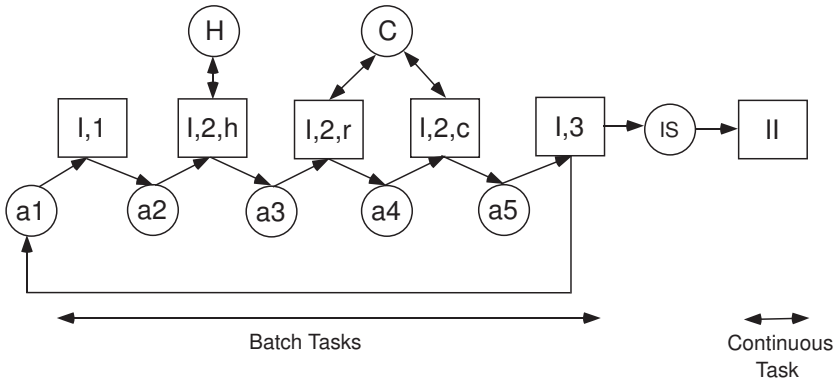

Figure 7: The resource task network representation for the test case

The two main characteristics relative to this test case are that the continuous task cannot be interrupted because we try to find a cyclic schedule and the second is that we cannot wait between the heating, the exothermic reaction and the cooling tasks.

Suppose that we have 2 reactors ($nbr\_unit = 2$), 17 time slots ($T = 17$) and a batch task can last for 4 time slots ($d = 4$). A typical evolution of the resources level of the storage tank, of the hot water rate used and of the cold water rate used over the scheduling cycle are represented in Figure 8. The dots represent the event times over the scheduling cycle.

The corresponding values of the $z_{i,t,\Omega(l)}$ batch variables are represented in the Gantt chart in Figure 9, with time on the horizontal axis and batch tasks on the vertical axis.

5.2 Specific reformulation

In this section, we describe three additional valid inequalities that are specific for this test case problem.

Every batch task can be performed on every reactor. The first valid inequality takes into account the fact that the time needed to complete all

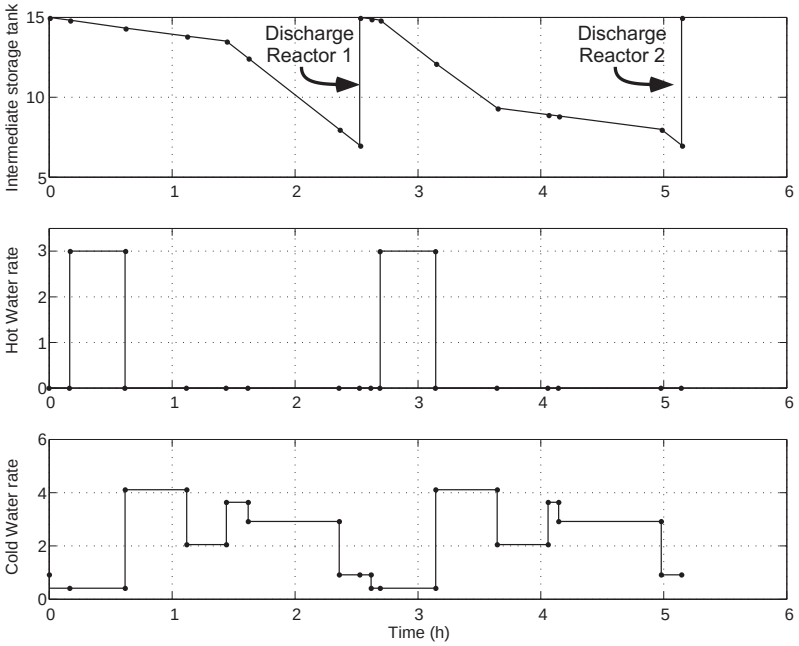

Figure 8: Evolution of the resources over the scheduling cycle

the tasks (i.e. the duration of the cycle) is greater or equal to the time needed if the batch tasks and processing times are evenly allocated among all the *nbr_unit* reactors.

The constraint can be written as

$$\frac{1}{nbr\_unit}\left(\sum_{i\in BT}\sum_{t=1}^{T}\sum_{t'=t}^{t+d-1} p_i z_{i,t,\Omega(t')}\right) \leq \sum_{t=1}^{T}\tau_t. \tag{24}$$

Because of the cyclic schedule, all the batch tasks have to be processed the same number of times. The total number of time slots is limited and therefore, we can restrict the number of times that a batch task can be performed on a cycle as follows

$$\sum_{t=1}^{T}\sum_{t'=t}^{t+d-1} z_{i,t,\Omega(t')} \leq \left\lfloor \frac{T}{|BT|} \right\rfloor \qquad \forall i \in BT. \tag{25}$$

For this specific test case, inequality (23) can be strengthened if $nbr\_unit <$

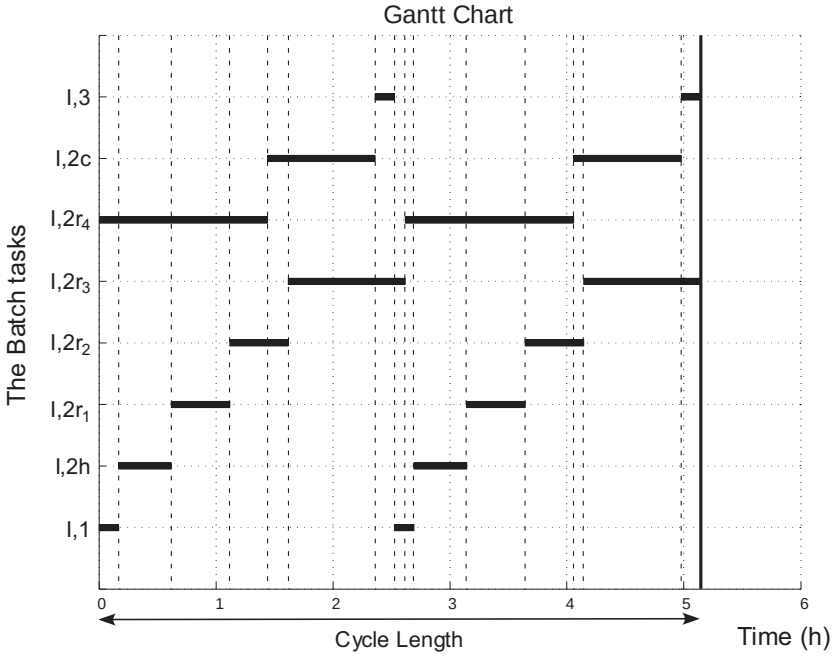

Figure 9: The scheduling of the batch tasks (2 reactors)

d. If this is the case, the inequality becomes

$$\sum_{k=t}^{l} \min(coeff_k, nbr\_unit)\tau_{\Omega(k)} \geq \sum_{i \in BT} p_i \left(\sum_{k=t}^{l} \sum_{f=k}^{\min(k+d-1,l)} z_{i,\Omega(k),\Omega(f)} \right). \tag{26}$$

The inequality (26) is still valid because the left hand side is the production time available in $\{t, \ldots, l\}$ for batches performed completely in $\{t, \ldots, l\}$ and the right hand side is the production time effectively used by such batches. We take the minimum between $coeff_k$ and nbr_unit in the left-hand side of the inequality (26) because the number of active batch tasks in any time slot cannot be larger than the number of reactors available.

5.3 Results for the test case

We report on tests on the efficiency of the exact and heuristic methods in order to solve the test case problem. We solve the improved formulation

of this problem by using a standard Branch and Bound package. In order to improve the Branch and Bound performance for the cases studied, we set priorities to branch first on the $z_{i,t,\Omega(l)}$ binary variables. The reason is that the other integer variables (units of resources available corresponding to machines and equipments) depend only of the binary variables $z_{i,t,\Omega(l)}$, and take automatically integer values when the $z_{i,t,\Omega(l)}$ variables take integral values.
We also impose to branch up first because this influences more the objective function value of the corresponding subproblem.

We show first that the exact solution method can only be used for limited size problem instances. We show also that for some larger instances, our MIP heuristic methods taking advantage of the strengthened formulation find good feasible solutions quickly. All the results in this paper have been obtained by using the Xpress MP software on a pentium 4, running at 3 GHz.

5.3.1 Exact solution methods

We pay attention first at the performance of the initial (F1) and of three strengthened formulations (F2)-(F3)-(F4). The initial formulation (F1) is based on the constraints (1)-(13), plus the specific constraints (24) and (25) for the test case. The basic strengthened formulation (F2) is based on the constraints (3)-(4), (6)-(13), (16),(18) and (22), with the case specific constraints (24), (25). The complete strengthened formulation (F3) is composed of (F2) and in addition the constraints (17), (19) and (26). The last strengthened formulation (F4) considers the additional constraints (17), (19) and (26) as model cuts, i.e these constraints are removed from the initial formulation, are added to the cut pool, and generated as cuts when they are violated.

The characteristics of the small (resp. large) instance in Table 1 are the following : $T = 10(resp.17), d = 4, nbr\_unit = 2$, There are 80 (resp. 136) integer variables, 320 (resp. 544) binary variables and 61 (resp. 103) continuous variables. The number of constraints in the strengthened formulations (F2)-(F4) is smaller because the timing constraints have been aggregated over all batch tasks.

Table 1 reports on the solution of the test case using the four formula-

Small Instance (T=10)				
nbr unit=2	F1	F2	F3	F4
Constraints	889	339	499	499
Nodes	118	21	3	**1**
Time	2.3 s	1.4 s	2 s	**0.7 s**
Productivity	3.11	3.11	3.11	**3.11**
Large Instance (T=17)				
nbr unit=2	F1	F2	F3	F4
Constraints	1498	563	954	954
Nodes	**274**	334	174	644
Time	**13.3 s**	13.8 s	44 s	153.7 s
Productivity	**3.11**	3.11	3.11	3.11

Table 1: First comparison between the initial (F1) and the strengthened formulations (F2)-(F4)

tions and a standard MIP solver. For both instances and all formulations, the maximal productivity for the test case is obtained by solving two mixed integer optimization problems with the objective function (15). We start with $\mu^1 = \underline{\rho}$.
In Table 1, "Nodes" represents the total number of Branch-and-Bound nodes needed in order to solve the two iterations up to optimality, "Time" represents the corresponding total CPU time and "Productivity" is the maximal productivity obtained for the test case.

In Table 1, we can see that the small and the large instances are solved easily by the four formulations. The small instance is solved by using the formulation (F4) at the root node. However, for the large instance, (F1) and (F2) give the best results and are quite comparable. In order to really test the quality of the two formulations, we need to solve more instances.

In Table 2, we consider a second case where the instances characteristics and the number of variables are the same as those in Table 1, except that $nbr\_unit = 3$. In the next Tables, the star (*) after the productivity measure means that the iterative procedure (that gives the optimal solution of the problem with the nonlinear objective function) is stopped at the end of some iteration before obtaining the optimal productivity solution. This occurs when the CPU solution time is too large.

Small Instance (T=10)				
nbr unit=3	F1	F2	F3	F4
Constraints	889	339	499	499
Nodes	1342	1186	700	630
Time	26.1 s	**14.5 s**	20.4 s	18 s
Productivity	3.27	**3.27**	3.27	3.27
Large Instance (T=17)				
nbr unit=3	F1	F2	F3	F4
Constraints	1498	563	954	954
Nodes	48524	24009	12169	16271
Time	1744 s	**496 s**	1617 s	2275 s
Productivity	3.21*	**3.21***	3.21*	3.21*
Nbr of iterations	1	1	1	1

Table 2: Second comparison between the initial (F1) and the strengthened formulations (F2)-(F4), where Nbr of iterations is the number of iterations of the objective linearization procedure

For both instances in Table 2, the strengthened formulations (F2)-(F4) obtain the optimal solution in fewer nodes. The strengthened formulation (F2) solves both instances quicker than the other three formulations. Not surprisingly, the strengthened formulations (F3)-(F4) taking into account all valid inequalities need fewer nodes than the other two formulations.

Finally, the characteristics and the number of variables of the small and the large instances in Table 3 are the same as the one in Table 1, except that $nbr\_unit = 4$. For the large instance and for all formulations except (F2), we are even not able to solve the first iteration of the linearized objective up to optimality. For such cases, a star is added to the CPU solution time of the corresponding instance in Table 3 and we calculate the remaining duality gap for that iteration as $\frac{\text{Best bound - Best solution}}{\text{Best bound}}$ and the productivity is the one corresponding to the best schedule obtained with respect to the linearized objective.

For the small instance in Table 3, the strengthened formulation (F2) gets the optimal solution quicker and with less nodes. For the large instance, only the strengthened formulation (F2) gives the optimal solution for the first iteration of the linearization procedure in a reasonable amount of time.

Small Instance (T=10)				
nbr unit=4	F1	F2	F3	F4
Constraints	889	339	499	499
Nodes	1365	**766**	772	1114
Time	20.61 s	**8.9 s**	18 s	25 s
Productivity	3.27	**3.27**	3.27	3.27
Nbr of iterations	2	2	2	2
Large Instance (T=17)				
nbr unit=4	F1	F2	F3	F4
Constraints	1498	563	954	954
Nodes	60900	**25609**	13000	14000
Time	2000 s*	**516 s**	2000 s*	2000 s*
Productivity	3.21*	**3.21***	3.21*	3.21*
Nbr of iterations	1	1	1	1
Remaining Duality gap	10.72 %	**0 %**	5.56 %	2.5 %

Table 3: Third comparison between the initial (F1) and the strengthened formulations (F2)-(F4)

In Tables 1 - 3, we can observe that for small instances, the four formulations give good results. For the large instances, the strengthened formulation (F2) provides almost always the optimal solution with respect to the linearized objective in less computing time and is more efficient than the initial one (F1). The strengthened formulations (F3)-(F4) solve the problem instances with almost always fewer nodes but are on average slower than (F2) in term of CPU time. A better cutting plane strategy should be developed in order to take advantage of the valid inequalities found in a reduced amount of computing time.

However, we can observe that for larger instances, the exact methods cannot solve the problems in a reasonable amount of time. Therefore, we pay attention to heuristic methods in order to obtain good feasible solutions for these larger instances quickly.

5.3.2 Heuristic solution methods

Given the superiority of reformulation (F2), the 5 heuristic methods used and compared to solve larger instances are the following :

1. Truncated B&B (F1) : Based on the initial formulation (F1), we solve the problem and we stop the branch and bound algorithm before the end of the resolution.

2. Truncated B&B (F2) : Based on the basic strengthened formulation (F2), we solve the problem and we stop the branch and bound algorithm before the end of the resolution.

3. Relax-and-Fix (F1) : The heuristic method described in Section 4 based on the initial formulation (F1). The parameter k for Local Branching, during and at the end of Relax-and-Fix, takes a value in the set $\{6, 9, 12\}$. We always start with $k = 6$. During Relax-and-Fix, we increase k only if we could not find a feasible solution, otherwise we stop Local Branching. At the end of Relax-and-Fix, we increase k as long as a significant improvement is observed.

4. Relax-and-Fix (F2*) : The heuristic method described in Section 4 based on the formulation (F2) plus constraints (17) and (19). The parameter k follows the same rules as for method 3.

5. Relax-and-Fix/LB (F2*) : It is a variant of method 4. The differences are that, at each iteration p of Relax-and-Fix : (i) we allow to change $k1$ values of variables in S_{p-1} ($p \geq 2$), (ii) the binary variables in the set $\{z_{i,t,p-2} : i \in BT, t \in \{p-d-1, \ldots, p-2\}\}$(if $p \geq 3$) are fixed (for all subsequent iterations of Relax-and-Fix) at the optimal value obtained at iteration $p-1$, (iii) the z variables in the set $(S_p \cup S_{p+1})$ have to be binary, and finally (iv) the integrality condition is relaxed on the others z variables. Here, we set $k1 = 3$ and the parameter k follows the same rules as for methods 3 and 4.

For the last three heuristic methods, we have imposed a maximum time for solving each MIP optimization subproblem in the heuristic. We choose to set this parameter to 500 sec. If at the end of the 500 sec., we have obtained a feasible solution, we stop the resolution of the current problem and proceed to the next step of the heuristic method. Otherwise, we continue to solve the current subproblem until a first feasible solution is obtained.

For Tables (4)-(6), we only try to solve the first iteration of the linearization of the objective function. Therefore, we put a star (*) after the productivity obtained because we could not prove with one iteration of the linearization that the productivity is optimal. Moreover, for the heuristic 1 or 2, the resolution of the first linearized objective problem was sometimes stopped before optimality was proved. For such cases, a star (*) is added to the CPU solution time of the corresponding instance in the Table. The duality gap indicates the remaining gap with respect to the linearized objective and is defined as $\frac{\text{Best bound - Best solution}}{\text{Best bound}}$. For the Relax-and-Fix methods, the best bound is the optimal solution obtained at the first iteration (before any fixing), and the best solution is the final Relax-and-Fix solution obtained.

The first heuristic comparison is proposed in Table 4. The characteristics of the small (resp. large) instance in Table 4 are the following : $T = 18(resp.26), d = 7, \text{nbr unit} = 2, \underline{\rho} = 1[ru/h], \overline{\rho} = 6[ru/h]$. There are 144 (resp. 208) integer variables, 1008 (resp. 1456) binary variables and 109 (resp. 157) continuous variables.

2 units		B&B (F1)	B&B (F2)	R&F (F1)	R&F (F2*)	R&F/LB (F2*)
Small Instance T=18	# Constr.	2449	703	2449	919	919
	Nodes	1643	**1**	18524	10	23
	D. Gap (%)	0	**0**	50.4	0	0
	Time	96 s	**2 s**	484 s	9 s	22 s
	Prod.	3.11*	**3.11***	3.05*	3.11*	3.11*
Large Instance T=26	# Constr.	3529	1007	3529	1319	1319
	Nodes	6200	9300	15192	493	**48**
	D. Gap (%)	14.8	0.74	40	1	**0**
	Time	1000 s*	1000 s*	556 s	105 s	**51 s**
	Prod.	2.37*	3.06*	2.51*	3*	**3.11***

Table 4: First heuristic comparison, $nbr\_unit = 2$

In Table 4, and for the small instance, the heuristic method 2 provides the optimal solution of the first linearization iteration quicker and with fewer nodes than the other heuristic methods. Regarding the large instance, the heuristic methods 5 provides quicker and with fewer nodes the optimal solution of the first linearization.

The second heuristic comparison is proposed in Table 5. The characteristics of the small (resp. large) instance in Table 5 are the same as the one given for Table 4, except that $nbr\_unit = 3$. Again, for every instance of

this table, we only perform one linearization iteration.

3 units		B&B (F1)	B&B (F2)	R&F (F1)	R&F (F2*)	R&F/LB (F2*)
Small Instance T=18	# Constr.	2449	703	2449	919	919
	Nodes	13400	7490	12918	10647	2575
	D. Gap (%)	11	8.13	11	**7.4**	8.13
	Time	1000 s*	517 s*	589 s	403 s	**196 s**
	Prod.	3.32*	3.6*	3.32*	**3.67***	3.6*
Large Instance T=26	# Constr.	3529	1007	3529	1319	1319
	Nodes	4300	5400	2019	3762	2558
	D. Gap (%)	48.9	40.67	48.9	**11.56**	22.2
	Time	1000 s*	1000 s*	109 s	700 s	**332 s**
	Prod.	2.51*	3.32*	2.51*	**3.27***	2.57*

Table 5: Second heuristic comparison, $nbr\_unit = 3$

The heuristic methods 4 or 5 seem to outperform the other three methods, a better solution in terms of remaining duality gap is obtained quicker. The heuristic 4 gives for the two instances a solution with a smaller remaining duality gap and a higher productivity than heuristic 5 but need more running time to compute these better results. Therefore, both heuristic methods 4 and 5 can be interesting in order to compute good feasible solutions quickly.

The third heuristic comparison is proposed in Table 6. The characteristics of the small (resp. large) instance in Table 5 are the same as the one of Table 4 except that $nbr\_unit = 4$. Again, for every instance of this table, we only perform one linearization iteration.

In Table 6, we can see that for the small instance, the heuristic method 1 gives a good feasible solution quicker. For the large instance, the heuristic method 4 gives a good solution with a smaller duality gap quicker than the other heuristic methods. We can observe in Table 6 that the heuristic methods 4 and 5 give good solutions for the small instance but are quite slow. For the large instance, the heuristic method 5 finds a good solution in terms of productivity but a very bad solution (and large duality gap) in terms of the linear objective. In order to obtain a better feasible solution (with respect to the linear objective) using the heuristic method 5, we change two parameters of the heuristic. The maximum time for solving each optimization subproblem is now set to 100 sec. in order to obtain quicker a feasible solution and we introduce more flexibility in the resolution of each subproblem by using the parameter $k1 = 5$ in order to obtain a better feasible solution.

4 units		B&B (F1)	B&B (F2)	R&F (F1)	R&F (F2*)	R&F/LB (F2*)
Small Instance T=18	# Constr.	2449	703	2449	919	919
	Nodes	**489**	2545	1901	11960	13969
	D. Gap (%)	**16.1**	16.1	16.1	17.33	16.1
	Time	**58 s***	270 s*	95 s	1167 s	1265 s
	Prod.	**3.32***	3.32*	3.32*	3.21*	3.32*
Large Instance T=26	# Constr.	3529	1007	3529	1319	1319
	Nodes	8800	9300	2302	6420	5816
	D. Gap (%)	44.06	42.63	51.8	**16.3**	42.2
	Time	2000 s*	2000 s*	256.4 s	**1421 s**	1490.3 s
	Prod.	3.32*	3.53*	2.51*	**3.3***	3.6*

Table 6: Third heuristic comparison, $nbr\_unit = 4$

The results obtained for the modified heuristic method 5 are presented in Table 7.

4 units		R&F/LB(F2*)
Small Instance T=18	# Constr.	919
	Nodes	5584
	D. Gap (%)	16.1
	Time	551 s
	Prod.	3.32*
Large Instance T=26	# Constr.	1319
	Nodes	4388
	D. Gap (%)	16.5
	Time	823 s
	Prod.	3.28*

Table 7: Modified heuristic method 5, $nbr\_unit = 4$

In Table 7, we can observe that for the small instance, the solution obtained with the heuristic method 5 and the new parameters is the same as the one obtained with the initial parameters but the CPU time is reduced. For the large instance, we get quicker a very good solution with a small remaining duality gap. With these new parameters, the heuristic method 5 provides a very good solution quicker than the other heuristic methods.

To summarize, we can conclude that for the small instances in Tables 4-6, the heuristic method 4 does not (except once) obtain a better feasible solution than the heuristics 1 and 2, with a smaller duality gap. However for the large instances in the same Tables, the heuristic methods 1 and 2 do not provide good solutions quickly anymore. For large instances, the use of the heuristic method 4 or 5 is quite interesting because we obtain on average better feasible solutions in less computing time. Moreover, heuristic method 5 can be more interesting than heuristic method 4, because, on average, this heuristic provides good feasible solutions quicker and with a small duality gap. However, for one instance, the results obtained by the heuristic method 5 are not satisfactory. For this case, we have seen that the parameters of the algorithm can be adapted in order to provide better feasible solutions quicker.

In the next section, we try to solve a larger industrial scheduling instance and we compare the efficiency of heuristic method 5 with truncated branch and bound corresponding to heuristic methods 1 and 2.

5.4 An industrial scheduling instance

In this section, we present a basic industrial scheduling problem and we use the heuristic methods 1,2 and 5 defined previously.

5.4.1 Problem description

There are three polymerization lines (denoted by I-II-III) with three, three and one unit respectively. The volume of the reactors for the first two lines ($V1$) is 27 ru. The volume of the reactor of line three ($V2$) is 140 ru. The polymerization task performed in the reactors is decomposed into the same number of batch tasks as in Section 5.1. Moreover, the same resources as for the test case are shared among the processes of a same line, but not shared between lines. The maximum rate of hot water for each line is 3 [ru/h]. The maximum rate of cold water is 7.4 $[ru/h]$ for line 1, 8 $[ru/h]$ for line 2 and 3.7 $[ru/h]$ for line 3.

After each reactor, there is one tank for each line where the product is discharged. The capacity of the tanks are 35 ru, 35 ru and 140 ru, respectively. The discharge of these tanks into the common buffer to all lines is modeled as a continuous task. We assume that the valve is processing like a continuous task and that the rate of material processed by the valves of the two first lines is in the interval $[0ru/h, 15ru/h]$ and by the valves of

the third line in the interval $[0ru/h, 10ru/h]$. The valves of the three lines do not have to process material all the time. The capacity of the common buffer is 40 ru and the continuous task after the buffer is a stripping task that cannot be stopped. For this task, the rate of material processed is in the interval $\rho \in [10ru/h, 55ru/h]$.

This process is represented in Figure 10.

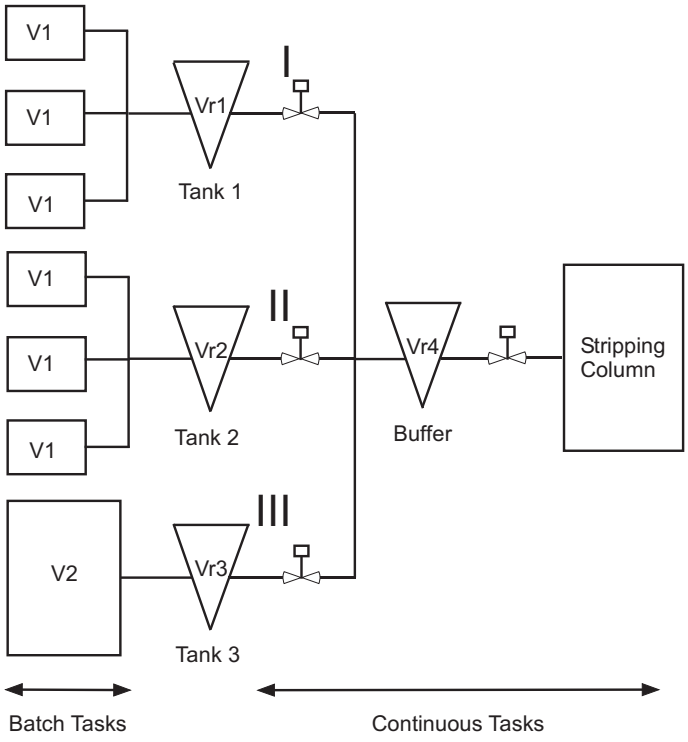

Figure 10: The basic industrial case

5.4.2 Comparison of Heuristics

We first model the problem with the initial formulation (F1) and with the strengthened formulation (F2) and we try to solve it up to optimality. We decide to take 20 time slots (i.e., $T = 20$) and a batch task can last for 10 time slots (i.e, $d = 10$). The problem has 10973 constraints for the initial formulation and 1833 for the strengthened one. For both formulations, we have

4800 binary variables, 480 integer variables, and 381 continuous variables. We stopped the resolution of the problem modeled by the initial formulation at the first linearization iteration after 20000 sec. and 21500 nodes, we got a remaining duality gap of 31.3%. The productivity corresponding to the best feasible solution is 20.6 $[ru/h]$. This solution method corresponds to heuristic method 1.
Also, we stopped the resolution of the problem modeled by the strengthened formulation at the first linearization iteration after 20000 sec. and 57600 nodes, we got a remaining duality gap of 27.2%. The productivity corresponding to the best feasible solution is 22 $[ru/h]$. This solution method corresponds to heuristic method 2.

For the same problem, we used the heuristic method 5 defined previously with the parameter $k1 = 8$ and the maximum time for solving each optimization subproblem is now set to 200 sec. The feasible solution obtained has a remaining duality gap of 26.9%. The corresponding number of nodes is 3126 and the CPU solution time is 1232 *sec.*. The productivity corresponding to this feasible solution is 22.12 $[ru/h]$.

After 20000 sec. of CPU time, the two methods firstly proposed, based on the initial (F1) and the strengthened (F2) formulation, were not able to provide a better feasible solution than the one given by the heuristic method 5 in 1232 sec. of CPU time. Therefore, heuristic methods such as heuristic 5 can be interesting for large instances of the linearized objective problem, both in terms of solution quality and running time.

5.4.3 Bounds on the maximal productivity

However, the final objective is to maximize productivity. As explained in section 2.2.6, the optimal productivity is obtained by solving a sequence of mixed integer linear programs where the value of μ is updated at each iteration.

Therefore, in order to obtain a good productivity for this basic industrial case, if the feasible solution obtained at the first iteration has a productivity significantly higher than the one fixed by default for iteration 1 (μ^1), we can solve a second iteration where μ is updated. We will obtain a feasible solution with a productivity at least as good as the one obtained at the first iteration. We can continue this procedure until no significant improvement of the productivity is achieved.

In this case, $\mu^1 = \underline{\rho} = 10$ and the heuristic 5 gives, for the first iteration, a feasible solution with a duality gap of 26.9% (for the linearized objective) and a productivity of $22.12[ru/h]$.

Actually, we cannot prove that this solution is optimal for the problem with the nonlinear objective function (14) maximizing productivity because the solution of the linear relaxation of the second linearized problem with objective function (15) and with $\mu^2 = 22.12$ is 14.18, and is not 0.

To measure the quality of the solution obtained for the problem with the linearized objective function in term of productivity (i.e. with respect to the non-linear objective), we show that we can bound the maximal improvement of productivity that can be obtained by the optimal solution at each iteration, and that this bound on the maximal improvement per iteration is monotonically non increasing over the linearization iterations.

The objective value for the optimal solution of the problem with the linear objective function at iteration p $(q^{*,p}, \tau^{*,p})$ is bounded by :

$$\sum_{j \in OUT} \sum_{t=1}^{T} q_{j,t}^{*,p} - \mu^p \sum_{t=1}^{T} \tau_t^{*,p} \leq UB^p$$

where UB^p is the LP relaxation bound obtained at the root node of the Branch-and-Bound algorithm. We show now how an upper bound on the maximal productivity improvement can be obtained for this iteration.

We divide every term of the previous inequality by the unknown $\sum_{t=1}^{T} \tau_t^{*,p} (> 0)$, that is by the optimal cycle duration at iteration p for the problem with the linear objective function, and we get :

$$\frac{\sum_{j \in OUT} \sum_{t=1}^{T} q_{j,t}^{*,p}}{\sum_{t=1}^{T} \tau_t^{*,p}} - \mu^p = \mu^{p+1} - \mu^p \leq \frac{UB^p}{\sum_{t=1}^{T} \tau_t^{*,p}}$$

This defines an upper bound on the productivity improvement $(\mu^{p+1} - \mu^p)$. As $\sum_{t=1}^{T} \tau_t^{*,p}$ is unknown, we look for a known lower bound on $\sum_{t=1}^{T} \tau_t^{*,p}$ in order to obtain a known upper bound on $\mu^{p+1} - \mu^p$.

First note that $\sum_{t=1}^{T} \tau_t^{*,p} \geq p_{i_1}$ because we know that in order to break the symmetry, we have imposed that at least one batch task i_1 of the line I' has to be performed during the cycle. Moreover, in our case, we have some precedence constraints between batch tasks and therefore we know that if a batch task i_1 starts on line I', at least a set of batch tasks on line I' $(BT_{I'})$ has to be processed during the cycle. The minimum time required, in order

to complete all the batch tasks in the set $BT_{I'}$, is obtained when all the possible reactors of line I' are performing the tasks in $BT_{I'}$ in parallel. This is why $(\sum_{t=1}^{T} \tau_t) \geq \frac{1}{nbr\_unit\_I'} \sum_{i \in BT_{I'}} p_i$ where nbr_unit_I' is the number of reactors available on line I' able to perform a batch tasks of the set $BT_{I'}$.

This lower bound on the cycle length can actually be improved. If a feasible solution at iteration p exists for the problem instance, it must satisfy $\sum_{t=1}^{T} \tau_t \geq LB^{CL,p}$ where

$$
\begin{aligned}
& LB^{CL,p} = \min \sum_{t=1}^{T} \tau_t \\
st \quad & \sum_{j \in OUT} \sum_{t=1}^{T} q_{j,t} - \mu^p \sum_{t=1}^{T} \tau_t \geq 0 \\
& \text{and all the constraints of the basic strengthened formulation (F2)} \\
& \text{are satisfied}
\end{aligned}
$$

Therefore, the upper bound on the maximal improvement of productivity that can be obtained by the optimal solution at iteration p can be bounded by

$$\mu^{p+1} - \mu^p \leq \frac{UB^p}{\sum_{t=1}^{T} \tau_t^{*,p}} \leq \frac{UB^p}{LB^{CL,p}} \tag{27}$$

Now we prove why the upper bound on the maximal productivity improvement that can be obtained by the optimal solution at each iteration is monotonically non increasing over the iterations.

By starting with $\mu^1 = \sum_{j \in OUT} \underline{\rho}_j$, the value of μ^p is monotonically non decreasing (see Section 2.2.6).

Let $\tilde{q}$ and $\tilde{\tau}$ be the solution of the linear relaxation at iteration $p+1$ that defines the upper bound UB^{p+1} on the optimal objective value. Then, this solution can also be a relaxed solution for iteration p. $UB^{p+1} = \sum_{j \in OUT} \sum_{t=1}^{T} \tilde{q}_{j,t} - \mu^{p+1} \sum_{t=1}^{T} \tilde{\tau}_t \leq \sum_{j \in OUT} \sum_{t=1}^{T} \tilde{q}_{j,t} - \mu^p \sum_{t=1}^{T} \tilde{\tau}_t$ because $\mu^p \leq \mu^{p+1}$ and $\sum_{t=1}^{T} \tilde{\tau}_t \geq 0$. As $\tilde{q}$ and $\tilde{\tau}$ are feasible for iteration p, we have $\sum_{j \in OUT} \sum_{t=1}^{T} \tilde{q}_{j,t} - \mu^p \sum_{t=1}^{T} \tilde{\tau}_t \leq UB^p$. This proves that $UB^{p+1} \leq UB^p$.

Moreover, since the value of μ^p is monotonically non decreasing, the value of $LB^{CL,p}$ is also monotonically non decreasing since the constraint $\sum_{j \in OUT} \sum_{t=1}^{T} q_{j,t} - \mu^p \sum_{t=1}^{T} \tau_t \geq 0$ will restrict more and more the feasible solution set.

Then, by dividing UB^{p+1} by $LB^{CL,p+1}$ and UB^p by $LB^{CL,p}$, and by using (27), the upper bound on the maximal productivity improvement that

can be obtained by the optimal solution at iteration $p+1$ is less or equal than the one at iteration p.

Therefore, if the upper bound on the maximal improvement of productivity that can be obtained by the optimal solution between two iterations is small, we know that for the next step this bound will be even smaller, and we can stop when we find that the improvement will not be significant enough.

In our industrial case, we tried to solve the optimization problem in order to determine $LB^{CL,1}$ but we could not solve it to optimality. The lower bound at the root node obtained on the minimum duration of the cycle is 1.8 h. The upper bound obtained for the first linearization iteration is 40.5 for the three heuristic methods. Therefore, the maximal improvement of productivity that can be obtained by the optimal solution for the first linearization iteration is $\frac{40.5}{1.8} = 22.5[ru/h]$. So the maximal productivity that can be obtained by the optimal solution at iteration 1 is 32.5 $[ru/h]$ (μ^1+22.5). Heuristic 5 produces a solution whose productivity is 22.12 ru/h ($= \mu^2$).

The upper bound obtained for the second linearization iteration is 14.18 and the lower bound at the root node obtained on the minimum duration of the cycle at iteration 2 is $LB^{CL,2} \geq 1.8$ h. The maximal productivity improvement that we can obtain by solving the second iteration up to optimality is $\frac{14.18}{1.8} = 7.87[ru/h]$. Therefore, the maximal productivity that can be obtained by the optimal solution at iteration 2 is 30 $[ru/h]$ ($\mu^2(22.12)$+7.87). We observe that the maximal improvement per iteration is non-increasing.

We can also mention here the fact that during the Branch and Bound algorithm, every time a feasible solution is obtained with a larger productivity than the best current one, we could add a cut valid for the whole formulation imposing that the productivity of the next feasible solution has to be greater or equal to this larger productivity. This could give a better value of μ for the next iteration and therefore speed up the convergence of the μ's. Unfortunately, we have observed that by adding this type of constraints, the resolution of the problem with the linear objective function was slower.

5.4.4 Solution of the Case

To conclude, we describe the best solution obtained in terms of productivity by heuristic method 5. The schedule obtained for line I and II is given in Figures 11-12 respectively. We can observe that the three reactors of line I and the three reactors of line II are processing during the cycle. Line III is not used.

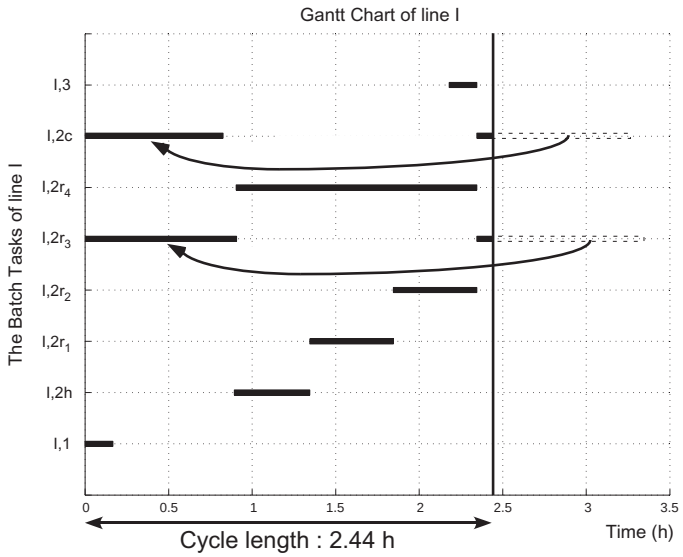

Figure 11: The schedule of the batch tasks of line I

For line 1 and 2, the resource levels are represented in Figures 13-14, respectively.

The evolution of the storage tank level of the common buffer (Vr4) is represented in Figure 15.

6 Conclusion and Future Work

In this paper, we introduced a tightened continuous time MILP formulation for solving the cyclic scheduling problem of a mixed plant. We showed that the improved formulation gave better results (quality and/or running times) than the initial one but the resolution of large instances remains difficult.

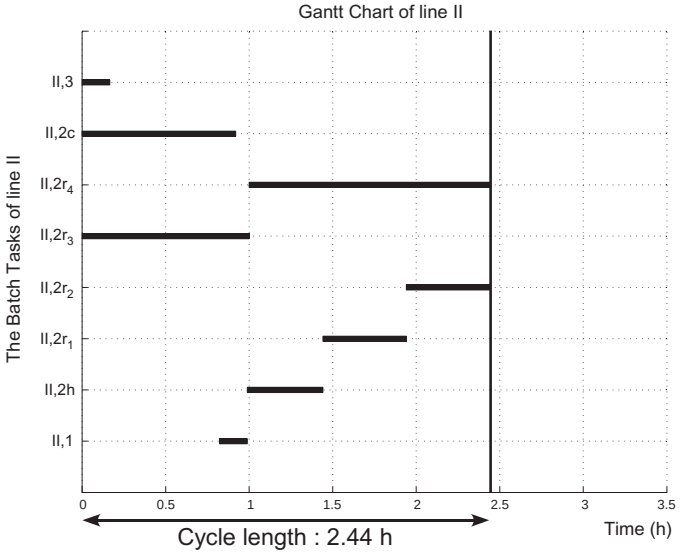

Figure 12: The schedule of the batch tasks of line II

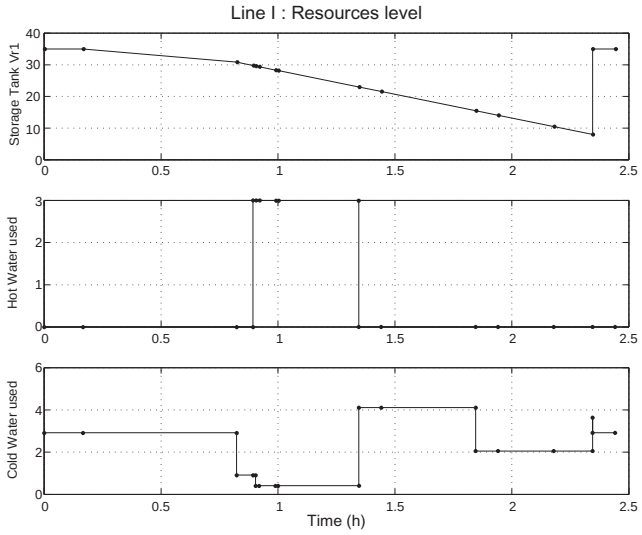

Figure 13: The resources for line I

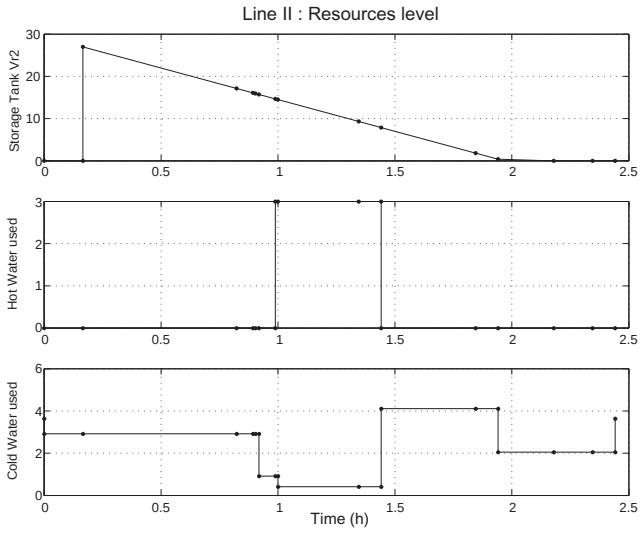

Figure 14: The resources for line II

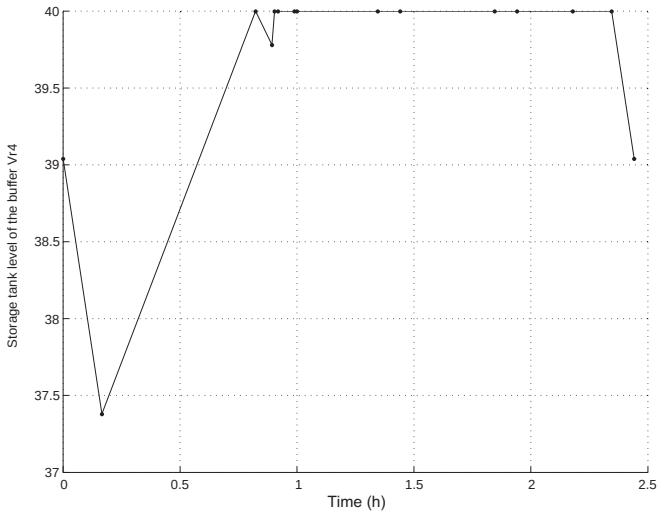

Figure 15: The evolution of the storage tank level of the buffer

We were still not able to solve realistic industrial cases with exact Branch-and-Bound methods. So, we investigated MIP based heuristic methods in order to obtain good feasible solutions quickly. We showed that, for some large instances, the heuristic solutions given by the exact methods (truncated Branch-and-Bound) were not better than the feasible solutions given by the MIP based heuristic methods, and the latter use less CPU solution time. Finally, we solved a basic industrial case using the initial and the strengthened formulations by truncated branch and bound, and also by a MIP based heuristic method. We got quicker and better feasible solutions by using the MIP based heuristic method, showing that such heuristic methods seem important for large instances.

As future work, we would like to find a tighter reformulation for the interaction between batch and continuous tasks in order to speed up the problem resolution. Also, an efficient separation algorithm should be implemented in order to choose which valid inequalities should be added to the model formulation during Branch and Bound, in order to reduce the formulation size and be able to solve larger instances faster.

Another objective for the future is to improve the heuristic methods found in order to find better feasible solutions for large instances quicker.

We would like also to extend the formulation in order to be able to model the problem where the batch sizes and the batch processing times are variables of the model.

Finally, the best cyclic schedule calculated here is independent of the initial state of the production lines. In order to be able to apply a good cyclic schedule for the plant processes, we need a transient schedule that brings the system from some initial state (possibly bad in terms of long term productivity) to a state from which a cyclic schedule with good productivity can be applied. Our model formulation should be adapted to contain a transient schedule before the cyclic one.

Appendix A

Suppose that for the problem composed of the linear objective function (15) and a given set of constraints, the optimal solution at iteration p is $q_{j,t}^{\star,p}$, for all t and $j \in OUT$, and $\tau_t^{\star,p}$, for all t. Suppose also that the $\sum_t \tau_t^{\star,p} > 0$, and that

$$\mu^p = \mu^{p+1} = \frac{\sum_{j \in OUT} \sum_t q_{j,t}^{\star,p}}{\sum_t \tau_t^{\star,p}}.$$

The optimal objective value at iteration p for the objective function (15) is then zero because the solutions of the optimization problems at iterations $p-1$ and p have the same productivity.

Suppose also that for the problem composed of the same constraints and of the non-linear objective function (14), the optimal solution is $\tilde{q}_{j,t}$, for all $j \in OUT$ and t, and $\tilde{\tau}_t$ for all t. Suppose also that the $\sum_t \tilde{\tau}_t > 0$.

Let $\tilde{\mu}$ be the optimal productivity

$$\tilde{\mu} = \frac{\sum_{j \in OUT} \sum_t \tilde{q}_{j,t}}{\sum_t \tilde{\tau}_t}$$

We have then that : $\sum_{j \in OUT} \sum_t \tilde{q}_{j,t} - \tilde{\mu} \sum_t \tilde{\tau}_t = 0$.

We cannot have $\mu^p > \tilde{\mu}$, because $\tilde{q}$ and $\tilde{\tau}$ define the optimal productivity and $q^{\star,p}$ and $\tau^{\star,p}$ are part of a feasible schedule.

Suppose now the $\tilde{\mu} > \mu^p$, i.e :

$$\tilde{\mu} = \frac{\sum_{j \in OUT} \sum_t \tilde{q}_{j,t}}{\sum_t \tilde{\tau}_t} > \frac{\sum_{j \in OUT} \sum_t q^{\star,p}_{j,t}}{\sum_t \tau^{\star,p}_t} = \mu^p$$

Then, $\sum_{j \in OUT} \sum_t \tilde{q}_{j,t} - \mu^p \sum_t \tilde{\tau}_t > \sum_{j \in OUT} \sum_t q^{\star,p}_{j,t} - \mu^p \sum_t \tau^{\star,p}_t = 0$

This is a contradiction because the optimal solution at iteration p of the problem with the objective function (15) is $q^{\star,p}$ and $\tau^{\star,p}$, and $\tilde{q}$, $\tilde{\tau}$ is feasible for this problem.

Therefore $\tilde{\mu} = \mu^p$ and the optimal solution of the problem with the objective function (15), $q^{\star,p}$ and $\tau^{\star,p}$, is also optimal for the problem with the nonlinear objective function (14).

Acknowledgement

We would like to especially thank I. Simeonova, G. Bastin and D. Dochain, from the INMA departement at University of Louvain in Belgium, for all the fruitful comments they have made about this work. We want to also thank our colleagues, Peter Malkin, Michel Baes and Ruslan Sadykov for the interesting questions asked and remarks made about this paper.

References

[1] K. Andersen and Y. Pochet. Coefficient strengthening : a tool for formulating mixed integer programs. *CORE Discussion Paper*, (2007/24), 2007.

[2] P.M. Castro, A.P. Barbosa-Povoa, and H.A. Matos. Optimal periodic scheduling of batch plants using RTN-based discrete and continuous-time formulations : a case study approach. *Ind. Eng. Chem. Res.*, 42:3346–3360, 2003.

[3] T. Christof and A. Loebel. Porta - a polyhedron representation transformation algorithm. *available via http://www.zib.de/Optimization/Software/Porta/*, 1997.

[4] E. Danna, E. Rothberg, and C. Le Pape. Exploring relaxation induced neighborhoods to improve mip solutions. *Mathematical Programming*, 102:71–90, 2005.

[5] W. Dinkelbach. On nonlinear fractional programming. *Management Science*, 13:492–498, 1967.

[6] M. Fischetti and A. Lodi. Local branching. *Mathematical Programming*, 98:23–48, 2003.

[7] C.A. Floudas and X. Lin. Mixed integer linear programming in process scheduling : modeling, algorithms, and applications. *Annals of Operations Research*, 139:131–162, 2005.

[8] M.G. Ierapetritou and C.A. Floudas. Effective continuous-time formulation for short-term scheduling. 1. multipurpose batch processes. *Ind. Eng. Chem. Res.*, 37:4341–4359, 1998.

[9] J.R. Isbell and W.H. Marlow. Attrition games. *Naval Research Logistics Quarterly*, 3:71–93, 1956.

[10] I.A. Karimi and C.M. McDonald. Planning and scheduling of parallel semicontinuous processes. 2. short-term scheduling. *Ind. Eng. Chem. Res.*, 36:2701–2714, 1997.

[11] J.D. Kelly and J.L. Mann. Flowsheet decomposition heuristic for scheduling : a relax-and-fix method. *Computers chem. Engng*, 28:2193–2200, 2004.

[12] E. Kondili, C.C. Pantelides, and R.W.H. Sargent. A general algorithm for short-term scheduling of batch operations - I. MILP formulation. *Computers chem. Engng*, 17:211–227, 1993.

[13] C.T. Maravelias. A decomposition framework for the scheduling of single- and multi-stage processes. *Computers chem. Engng*, 30:407–420, 2006.

[14] C.T. Maravelias and I.E. Grossmann. A hybrid MILP/CP decomposition approach for the continuous time scheduling of multipurpose batch plants. *Computers chem. Engng*, 28:1921–1949, 2004.

[15] C.A. Mendez, J. Cerda, I.E. Grossmann, I. Harjunkoski, and M. Fahl. State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Computers chem. Engng*, 30:913–946, 2006.

[16] L. Mockus and G.V. Reklaitis. Mathematical programing formulation for scheduling of batch operations based on nonuniform time discretization. *Computers chem. Engng*, 21:1147–1156, 1997.

[17] G.L Nemhauser and L.A Wolsey. *Integer and combinatorial optimization*. John Wiley & Sons, 1988.

[18] C.C. Pantelides. Unified frameworks for the optimal process planning and scheduling. *Proceedings on the second conference on foundations of computer aided operations*, pages 253–274, 1994.

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

[20] Y. Pochet and L.A. Wolsey. *Production planning by mixed-integer programming*. Springer, 2006.

[21] G. Schilling and C.C. Pantelides. A simple continuous-time process scheduling formulation and a novel solution algorithm. *Computers chem. Engng*, 20:S1221–S1226, 1996.

[22] G. Schilling and C.C. Pantelides. Optimal periodic scheduling of multipurpose plants. *Computers chem. Engng*, 23:635–655, 1999.

[23] N. Shah, C.C. Pantelides, and R.W.H. Sargent. Optimal periodic scheduling of multipurpose batch plants. *Annals of Operations Research*, 42:193–228, 1993.

[24] H. Stadtler. Multilevel lot sizing with setup times and multiple constrained resources: Internally rolling schedules with lot-sizing windows. *Operations Research*, 51:487–502, 2003.

[25] A. Sundaramoorthy and I.A. Karimi. A simpler better slot-based continuous-time formulation for short-term scheduling in multipurpose batch plants. *Chemical Eng. Science*, 60:2679–2702, 2005.

[26] F. Warichet and Y. Pochet. A continuous time formulation for the cyclic scheduling of a mixed plant : valid and facet defining inequalities. Technical report, Center for operations research and econometrics (CORE), Belgium, 2006,in preparation.

[27] D. Wu and M. Ierapetritou. Cyclic short-term scheduling of multiproduct batch plants using continuous-time representation. *Computers chem. Engng*, 28:2271–2286, 2004.

[28] X. Zhang and R.W.H. Sargent. The optimal operation of mixed production facilities - a general formulation and some approaches for the solution. *Computers chem. Engng*, 20:897–904, 1996.