A slot-based formulation for the short-term scheduling of multi-stage, multi-product batch plants with resource constraints and sequence-dependent changeovers

Muge Erdirik-Dogan and Ignacio E. Grossmann*

Department of Chemical Engineering, Carnegie Mellon University,

***Pittsburgh, Pennsylvania 15213**

June 2007

Abstract

In this paper we address the short-term scheduling of multi-product batch plants with parallel identical/non-identical reactors, a challenging problem that has been motivated by a real world application at the Dow Chemical Company. The main challenges include handling sequence-dependent changeovers with high variance as well as two-stage production with limited, shared intermediate storage, and batch splitting. In order to deal with these issues, we propose a new continuous time MILP model based on the concept of asynchronized time slots. The proposed model has the unique feature that it incorporates slot based mass balances and accounts for sequence-dependent changeovers. While effective for small problems, the proposed model becomes computationally expensive to solve for larger problems, mainly due to the fact that the necessary number of slots is not known a priori and the model requires postulating an upper bound on the slots. Therefore, we propose a bi-level decomposition algorithm in which the original problem is decomposed into an upper level scheduling and a lower level scheduling and sequencing problem. The upper level model is a recent planning model where mass balances are aggregated over time periods and detailed timing constraints are dropped. However, the effects of changeovers are accounted for by incorporating scheduling constraints. Thus the upper level model yields very accurate predictions and a tight upper bound. In the lower level the original problem is solved for the number of slots and subsets of products as predicted by the upper level yielding a lower bound. These subproblems are solved iteratively until the bounds converge. In order to reduce the number of total iterations, we propose a new class of symmetry breaking constraints. The application of the proposed approach is illustrated with several examples.

1. Introduction

The problem of short-term scheduling of batch plants has received considerable attention in the literature over the last decade with the focus ranging from single stage to multiple stage processes, from specific to general types of network configurations, and from fixed equipment connectivity to flexible arrangements. Extensive reviews can be found in Reklaitis (1992), Pekny and Reklaitis (1998), Pinto and Grossmann (1998), Shah (1998), Kallrath (2002), Floudas and Lin (2004), and Mendez et al.(2006).

In this paper we address the short-term scheduling of parallel identical/non-identical multi-product batch reactors, a challenging problem that has been motivated by a real world application at the Dow Chemical Company. This specific problem not only poses the difficulty of handling sequence-dependent changeovers with high variance for an industrial-sized data, but also has the difficulty of handling two-stage production with limited and shared intermediate storage. Due to topological restrictions within the plant, batch splitting of the intermediate is required once its production is

* Author to whom all correspondence should be addressed. Telephone: (412)-268-2230. Email: grossmann@cmu.edu.

completed. This combined with having to monitor the level of intermediates in the tanks to avoid shortages or overloads, poses several new difficulties in handling the mass and inventory balances.

This special kind of resource constrained batch scheduling problem can in principle be tackled through discrete time models such as the State Task Network (STN) (Kondilli et al., 1993, Shah et al., 1993, Rodrigues et al., 2000) or Resource Task Network (RTN) (Pantelides, 1994), which are quite general and effective in monitoring the level of resources. However, the weakness of these discrete time models is that handling of sequence-dependent changeovers is cumbersome. Specifically, STN requires a finer discretization of time in order to accommodate small changeovers. Moreover, the number of constraints quickly becomes extremely large when problems involve a significant number of changeovers. RTN requires defining explicitly additional tasks associated with each type of cleaning as well as different states of cleanliness for each processing units. The definition of cleaning tasks significantly increases the model size making the problem intractable very quickly. These difficulties can in principle be overcome with continuous time STN (Schilling and Pantelides, 1996, Zhang and Sargent, 1996, Ierapetritou and Floudas, 1998, Mockus and Reklaitis, 1999, Giannelos and Georgiadis, 2002, and Maravelias and Grossmann, 2003) and RTN (Pantelides, 1994, Castro et al., 2001,and Castro et al., 2004) models based on the definition of global time points. However, these have the disadvantage of being potentially expensive to solve since the use of common time grid for all shared resources requires a large number of time points to be defined in order to consider exact transition times and to handle multiple due dates.

Another alternative to consider for the modeling is to use formulations that are based on the concept of time slots. The main idea is to postulate an appropriate number of slots for each processing order to allocate them to the batch tasks to be performed. Relevant work in this area is represented by the formulations of Pinto and Grossmann (1995, 1996), Chen at al., 2002 and Lim and Karimi (2003). While sequence-dependent changeover times can be treated in a straightforward manner, the handling of mass balances is difficult as well as the monitoring of the shared resources. Recently Sundaramoorthy and Karimi (2005) have developed a novel slot-based model that can handle mass balances, but that formulation is built on the assumption of sequence-independent changeovers which makes it non-applicable to this specific problem.

Other alternative approaches for sequential processes are based on the concept of immediate or general batch predecessor. Relevant work involves the formulations by Cerda et al. (1997), Mendez et al. (2000), Mendez et al. (2001), Mendez and Cerda (2003, 2004a, 2004b) and Gupta and Karimi (2003). A common limitation of precedence-based formulations is that the number of sequencing variables scales with the number of batches of products to be scheduled, which can increase significantly the model size. More importantly, batch precedence formulations cannot handle mass balances explicitly.

Thus, there is a motivation for exploring special purpose models that might be able to more readily exploit the structure of the problem. In order to tackle this challenging problem, we propose a new continuous time MILP optimization model that uses the representation of time slots. The proposed model has the unique feature that it incorporates mass balances and accounts for sequence-dependent changeovers. Each slot represents one potential batch of the product that is assigned to that slot. Since the number of batches of each product is a variable to be determined by the model, the exact number of slots to be utilized is not known a priori. Hence, we postulate an upper bound for the number of slots, which is more than the necessary for each unit. The assignments of products to these slots are to be determined to define the sequence of production on each unit and each time period. The length of each slot is equal to the batch time of the product that is assigned to that slot plus the corresponding transition time. In order to avoid violations in the use of the limited shared resources, the mass balances are performed on a slot basis. Moreover, since the time slots are not synchronized over the units, we introduce constraints to determine the relative locations of the slots across parallel units to ensure feasible transfer of material.

While effective for small problems and short time periods, the proposed model is computationally expensive to solve for larger problems and longer time periods. His is largely due to the fact that it requires postulating an upper bound on the slots for each unit and each time period. The solution time increases rapidly with the number of slots because the number of binary variables and the transition variables increase directly with increasing number of slots. Therefore, we propose a rigorous bi-level decomposition algorithm that reduces the computational expense of the model. The proposed approach involves decomposing the problem into an upper level sequencing and a lower level scheduling model. The upper level which is based on the planning model of Erdirik-Dogan and Grossmann (2007), is an aggregation of the original problem where the detailed timing of production and changeovers are replaced with time balances and slot based mass balances are aggregated over time periods. The main decisions in this level are the assignments of products to available equipment, number of batches of each product as well as the production and inventory levels. Since the upper level is a relaxation of the original model, it yields a valid upper bound on the profit. Furthermore, it also yields a tighter upper bound on the number of slots to postulate for the lower level through the number of batches. In the lower level, the proposed MILP scheduling model is solved by excluding the products that were not selected by the upper level for the number of slots as determined by the upper level. Since its solution corresponds to a feasible solution of the original problem, it yields a lower bound on the profit. This procedure iterates until the difference between the upper bound and the lower bound is less than a specified tolerance. To expedite the search, we add logic cuts that are derived from the structure of the problem which help to eliminate symmetric solutions.

This paper is organized as follows. In the following section the problem definition is presented. This is followed by the proposed MILP scheduling model. Section 4 consists of a small example demonstrating the need for a specialized solution algorithm. The solution strategy, which is based on the bi-level decomposition algorithm, is discussed in section 5. In sections 6 and 7, the upper level and lower level sub-problems are presented, respectively. This is followed by section 8 where the logic cuts are presented. Application of the proposed solution approach is illustrated in section 9 with several examples that show that the slot based scheduling model can be solved effectively for a variety of problems. Finally, we summarize our findings in section 10.

2. Problem Definition

The scheduling problem we will address in this paper is as follows (see Figure 1). Given is a plant that contains identical and/or non-identical batch reactors that operate in parallel. The batch reactors are to be used to manufacture intermediates and final products. A subset of the final products is produced in a single reaction stage, while the remaining final products require intermediates, thus involving two reaction stages with finite intermediate storage. Each final product is fed to a dedicated storage tank. Due to the topology of the plant, once an end product is sent to the corresponding dedicated storage tank, it can not be retrieved back to the plant. Therefore, handling of the end products that are required as intermediates involves batch splitting. Each reactor can produce a subset of products for which corresponding batch times and batch sizes are given. We are given transition times and costs which are both unit and sequence dependent. Given are also storage tanks with associated capacities, production costs and selling prices for products. Specified is a production horizon composed of a certain number of time periods defined by due dates on which product demands are specified. The demands are point demands and product shipments occur only at the end of each period.

The goal of this scheduling problem is then to determine: (1) the assignment of products to available units; (2) number of batches of each product; (3) the optimal sequence of production on each unit; (4) start and end times of each batch; (5) the amounts of intermediates and end products produced and sold in each time period; (6) the inventory levels. The objective is to maximize the profit in terms of sales,

operating costs, inventory holding costs and transition costs, while satisfying production demands at the specified due dates.

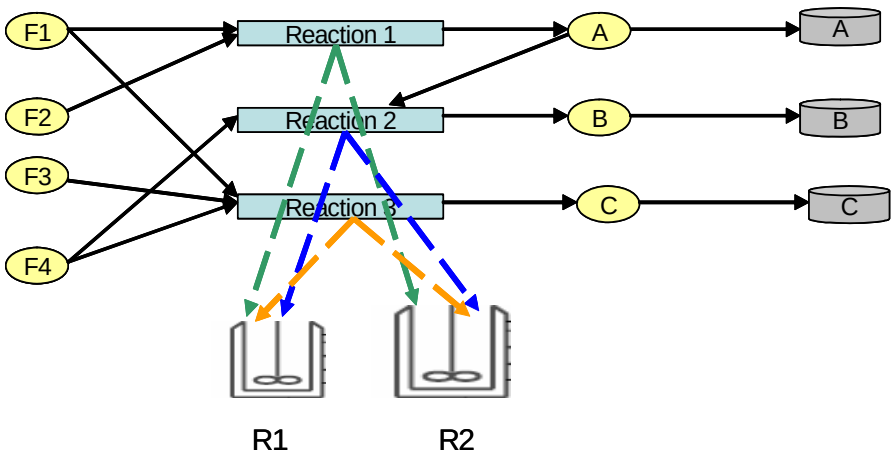


Figure 1. Representation of the problem

3. Mixed-Integer Linear Programming (MILP) Model

The proposed MILP model is based on a continuous time domain representation and has the following features:

(a) The model is based on a time slot representation where each slot represents one potential batch of the product that is assigned on that slot (see Figure 2). Since the number of batches of each product is a variable to be determined by the model, the exact number of slots to be utilized is not known a-priori. In order to avoid infeasible or suboptimal solutions, we postulate a valid upper bound on the number of slots for each unit. Due to the overestimation of the slots, some slots may be left unoccupied (null slots).

(b) The selection of number of slots to postulate for each unit depends on the products that can be processed on each unit, corresponding batch times, $BT_{i,m}$ and the length of the time period, H_t. We then define the number of slots postulated for each unit as $NS_{m,t} = \left\lfloor \underset{i}{Max}\left\{H_t / BT_{i,m}\right\} \right\rfloor$. Since the time horizon is divided into time periods of equal length, this reduces to $NS_m = \left\lfloor \underset{i}{Max}\left\{H_t / BT_{i,m}\right\} \right\rfloor$. In other words, we postulate NS_m slots for unit m for each time period (see Figure 2).

(c) The assignments of products to these slots define the sequence of production on each unit and at each time period (see Figure 3). The length of each slot is equal to the batch time of the product assigned to that slot plus the corresponding transition time. If none of the products is assigned to a particular slot, then the length of the slot is forced to zero.

(d) In each slot at most one product can be produced. However, the same product can be produced on multiple slots.

(e) The number of postulated slots is the same for a specific unit for all time periods. However, start and end times can vary for each unit and each time period. In other words, the slots are asynchronous across the parallel units.

(f) The final times of periods are synchronized for all units.

(g) Production occurs in batches and multiple units may process separate batches of the same product in order to satisfy the demands on the due dates. Note that the batch reactors that operate in parallel do not necessarily have to be in phase nor identical.

(h) As mentioned in the problem statement, production involves intermediate products with finite storage. Since intermediate products can be consumed by more than one batch of the same end product and/or multiple end products in the same time period, it is a limited shared resource. In order to prevent violations in the use of these shared resources, mass balances are performed on a slot basis. Moreover, since the time slots are not synchronized over the units, we introduce constraints to determine the relative locations of slots across parallel units to ensure feasible material transfer.

(i) Batch size and processing times of each batch are fixed parameters. However, they are dependent on both the nature of the product and on the unit on which it is produced.

(j) Each unit is suitable for producing a subset of the products which is predefined.

(k) Transition times required when switching from one product to another are dependent on both the type of equipment and also on the production sequence. Note that both the transition times within each time period and across adjacent time periods must be taken into account.

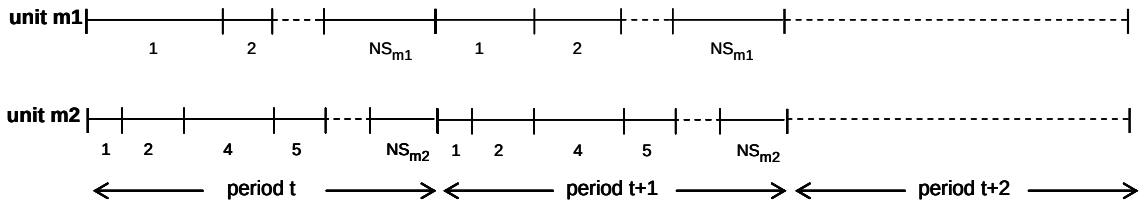


Figure 2. Prepostulating time-slots for each unit

Some of the major assumptions include the following:

- Demands, due dates, batch times and sizes, transition times are deterministic.
- Transition costs are directly proportional to transition times.
- After the processing of a product is completed, the product materials are transferred to the storage tanks.
- The transfer times are negligible.
- The operations are assumed to be non-preemptive.
- The postulated slots are restricted to be in a specific period. In other words, the length of a slot cannot exceed the length of the corresponding time period.

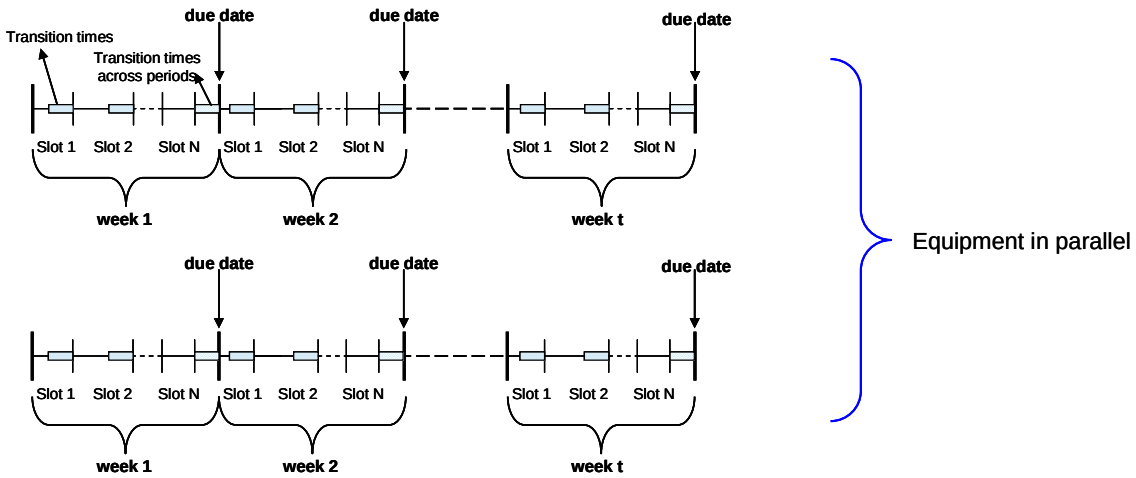


Figure 3. Time slots postulated for each unit at each time period

The MILP model (OP) is as follows:

3.1. Assignments and Processing Times:

$$\sum_{i \in I_m} W_{i,m,l,t} \leq 1 \quad \forall\, l \in L_m, m, t \tag{1}$$

$$\sum_{i \in I_m} W_{i,m,l,t} \geq \sum_{i \in I_m} W_{i,m,l+1,t} \quad \forall l \in L_m - \{\overline{l}_m\}, m, t \tag{2}$$

$$PT_{m,l,t} = \sum_{i \in I_m} BT_{i,m} \cdot W_{i,m,l,t} \quad \forall l \in L_m, m, t \tag{3}$$

$$X_{i,m,l,t} = Q_{i,m} \cdot W_{i,m,l,t} \quad \forall i \in I_m, \forall l \in L_m, m, t \tag{4}$$

Constraint (1), which is expressed in terms of the assignment variables $W_{i,m,l,t}$, represents the condition that at most one product can be assigned to each slot *l* of each unit *m* at each time period *t*. Note that not all slots postulated on unit in time period *t* have to be utilized. Constraint (2) enforces the consecutive utilization of each slot in each unit and each time period. However, idle times are allowed for the units. Constraint (2) also ensures that the unused (null) slots are placed at the end of the corresponding time periods. According to constraint (3), $PT_{m,l,t}$, the length of slot *l* of unit *m* in time period *t*, is equal to the batch time of the product that is assigned on that slot. Note that the length of the slot is forced to zero if there is no assignment made to that slot. Constraint (4) determines $X_{i,m,l,t}$, the amount of product *i* produced in slot *l* of unit *m* at each time period *t*, which is equal to the batch size, $Q_{i,m}$, of product *i* in unit *m*.

3.2. Sequence Dependent Transitions and Detailed Timing Relations

Transition times occur when the production on a unit is changed from one product *i* to another *i'* in order to avoid off-specification products. These transitions may involve cleaning or operational changes in the equipment and depend on the sequence of production as well as the equipment on which the products are assigned (see Figure 4).

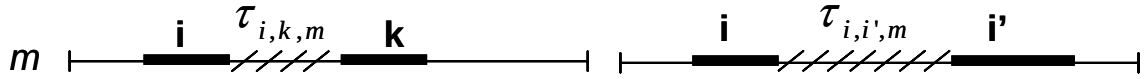


Figure 4. Sequence-dependent transition time

3.2.1. Transition times within each time period:

$$Z_{i,i',m,l,t} \geq W_{i,m,l,t} + W_{i',m,l+1,t} - 1 \qquad \forall i, i' \in I_m, i' \neq i, \forall l \in L_m - \{\overline{l}_m\}, m, t \tag{5}$$

Constraint (5) defines the transition variable $Z_{i,i',m,l,t}$ which represents the transition that occurs within each period (Figure 5). The value of $Z_{i,i',m,l,t}$ is 1 if product *i* assigned to slot *l* is followed by product *i'*

at slot *l*+1 on unit *m* at time period *t*; otherwise it is zero. Because the transition costs are minimized in the objective function, the variables $Z_{i,i',m,l,t}$ can be treated as continuous variables, $0 \le Z_{i,i',m,l,t} \le 1$.

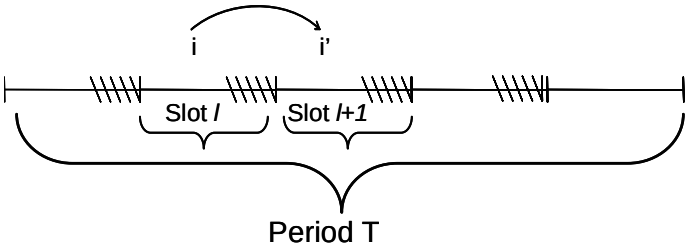


Figure 5. Sequence-dependent transition times within each time period

The transition time corresponding to slot *l* can then be defined as:

$$TR_{m,l,t} = \sum_{i \in I_m} \sum_{i' \in I_m} \tau^m_{i,i'} \cdot Z^t_{i,i',m,l,t} \quad \forall l \in L_m - \{\bar{l}_m\}, m, t \tag{6}$$

Note that according to (1) and (5) at most one of the terms becomes nonzero in the summation term of (6).

3.2.2. Transition times across adjacent weeks:

Transitions across adjacent time periods take place between the product that is the last one to be assigned on the current time period, and the product that is the first one to be assigned on the subsequent time period. If all the postulated slots had been utilized, constraints (7) and (8) would be sufficient to model the transitions across adjacent weeks.

$$\hat{Z}_{i,i',m,l,t} \ge W_{i,m,l,t} + W_{i',m,l',t+1} - 1 \qquad \forall i, i' \in I_m, i' \ne i, l = \bar{l}_m, l' = 1, m, t \in T - \{\bar{t}\} \tag{7}$$

where $\bar{l}_m$ corresponds to the last slot of time period *t* and *l'* corresponds to the first slot of the subsequent time period. The transition time required would then be determined by:

$$\hat{TR}^{across}_{m,l,t} = \sum_{i \in I_m} \sum_{i' \in I_m} \tau_{i,i'} \cdot \hat{Z}_{i,i',m,l,t} \quad l = \bar{l}_m, m, t \in T - \{\bar{t}\} \tag{8}$$

Since the number of slots postulated for each unit at each time period will be larger than the number of actual slots that is needed, this makes the modeling of transitions across adjacent weeks a nontrivial task. For instance, if the last slot ($\bar{l}_m$) is empty the transition in ($\hat{Z}$) reduces to zero neglecting the transition between the product assigned to slot ($\bar{l}_m$-1) of time *t* and the product assigned to the first slot of period *t*+1. In order to model the case when there is at least one empty slot, we first define all the potential transitions, $\tilde{Z}_{i,i',m,l,t}$, from period *t* to the first slot of period *t*+1 (see Figure 6):

$$\tilde{Z}_{i,i',m,l,t} \ge W_{i,m,l,t} + W_{i',m,l',t+1} - 1 \quad \forall i, i' \in I_m, i' \ne i, \forall l \in L_m - \{\bar{l}_m\}, l' = 1, m, t \in T - \{\bar{t}\} \tag{9}$$

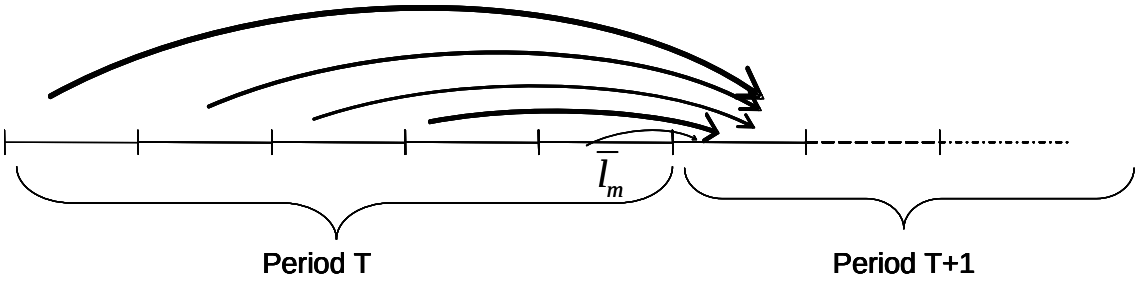


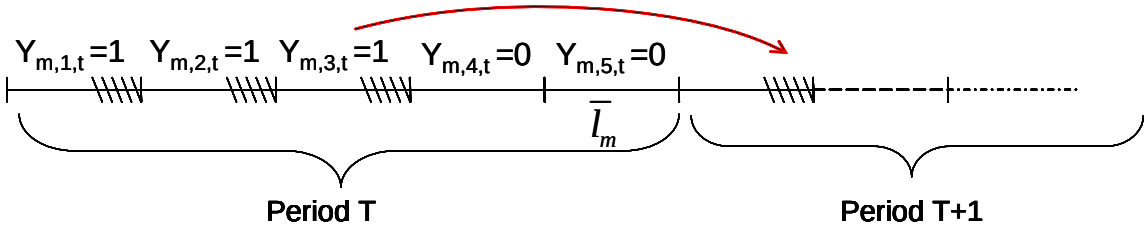
Figure 6. Potential transitions across adjacent periods

Among these transitions only the one corresponding to the transition between the last utilized slot of period *t* and the first slot of period *t*+1 should be activated. This requires identifying the last slot utilized in each time period.

In order to determine whether a slot is being utilized or not, we introduce the variable $Y_{m,l,t}$. According to (10), the value of $Y_{m,l,t}$ is 1 if slot *l* of unit *m* is being used at time period *t*; otherwise, it is zero.

$$Y_{m,l,t} = \sum_{i \in I_m} W_{i,m,l,t} \quad \forall l \in L_m, m, t \tag{10}$$

Since by constraint (2) we ensure that the unused (null) slots will be placed as last slots in each time period, we identify the last slot (*l*) to be utilized as the one where the succeeding slot (*l*+1) is left unused, $Y_{m,l+1,t}$=0 (see Figure 7).



Only the transition from the 3rd slot is taken into account.

Figure 7. Transitions across adjacent weeks

The transitions that occur across adjacent time periods can then be handled with the following constraint:

$$\tilde{TR}^{across}_{m,l,t} = \left(\sum_{i \in I_m} \sum_{i' \in I_m} \tau^m_{i,i'} \cdot \tilde{Z}^t_{i,i',m,l,t}\right) \cdot \left(1 - Y_{m,l+1,t}\right) \quad \forall l \in L_m - \{\bar{l}_m\}, m, t \tag{11}$$

Note that the equality in constraint (11) will be non-zero for slot *l* only if slot *l* is the last slot that is to be utilized in time period *t*. In other words, if the immediate successor of slot *l* is occupied, then $Y_{m,l+1,t}$ becomes 1 forcing $\tilde{TR}^{across}_{m,l,t}$ to be zero. On the other hand, if the immediate successor of slot *l* is not occupied, then $Y_{m,l+1,t}$ becomes zero, the term $\left(1 - Y_{m,l+1,t}\right)$ becomes 1 and constraint (11) becomes $\tilde{TR}^{across}_{m,l,t} = \sum_{i \in I_m} \sum_{i' \in I_m} \tau^m_{i,i'} \cdot \tilde{Z}^t_{i,i',m,l,t}$. Note that an unoccupied slot might be followed by another unoccupied slot

(see slot 4 of Figure 6). In this case the term $\left(1-Y_{m,l+1,t}\right)$ will not be equal to zero, but the transition variable $\tilde{Z}^{t}_{i,i',m,l,t}$ will become zero for this slot through constraint (9) and reduce constraint (11) to zero. The equality in constraint (11), however, holds only if there are unused slots left within time period *t*. For the time periods where all the postulated slots are utilized, modeling of transitions across adjacent time periods include only the last slot of time period *t* and the first slot of the subsequent time period. In this case, the transition variable, $\hat{Z}_{i,i',m,l,t}$, and the transitions across adjacent periods, $\hat{TR}^{across}_{m,l,t}$, are represented by constraints (7) and (8), respectively.

According to constraint (12) the end time of a slot is equal to the starting time plus the batch time of the product that is assigned to that slot plus the corresponding transition times. Third to last term represents the transitions within the time periods (see (6)); second to last term represents the transitions across adjacent periods when all the postulated slots are utilized (see (8)); the last term is valid if there are null slots present in the period (see (11)). For each slot *l* depending on whether the successive slot (*l*+1) is used or not, the term $\sum_{i}\sum_{i'}\tau_{i,i'}\cdot\tilde{Z}_{i,i',m,l,t}$ is dropped or activated.

$$Te_{m,l,t} = Ts_{m,l,t} + PT_{m,l,t} + \sum_{i\in I_m}\sum_{i'\in I_m}\tau^{m}_{i,i'}\cdot Z^{t}_{i,i',m,l,t} + \sum_{i\in I_m}\sum_{i'\in I_m}\tau^{m}_{i,i'}\cdot\hat{Z}^{t}_{i,i',m,l,t} + \left(\sum_{i\in I_m}\sum_{i'\in I_m}\tau^{m}_{i,i'}\cdot\tilde{Z}^{t}_{i,i',m,l,t}\right)\cdot\left(1-Y_{m,l+1,t}\right) \quad \forall l\in L_m, m, t \tag{12}$$

Note that equation (12) is nonlinear due to the term $\left(\sum_{i}\sum_{i'}\tau_{i,i'}\cdot\tilde{Z}_{i,i',m,l,t}\right)\cdot\left(1-Y_{m,l+1,t}\right)$. To avoid the difficulty of handling such terms, we linearize that term using a convex hull formulation. We first define

$$TRT_{m,l,t} = \sum_{i\in I_m}\sum_{i'\in I_m}\tau^{m}_{i,i'}\cdot\tilde{Z}^{t}_{i,i',m,l} \quad \forall l\in L_m-\{\bar{l}_m\}, m, t\in T-\{\bar{t}\} \tag{13}$$

Equation (12) then becomes:

$$Te_{m,l,t} = Ts_{m,l,t} + PT_{m,l,t} + \sum_{i}\sum_{i'}\tau^{m}_{i,i'}\cdot(Z_{i,i',m,l,t}+\hat{Z}_{i,i',m,l,t}) + TRT_{m,l,t} - \left(TRT_{m,l,t}\cdot Y_{m,l+1,t}\right) \ \forall l\in L_m, \forall m, \forall t \tag{14}$$

We then define

$$TX_{m,l,t} = TRT_{m,l,t}\cdot Y_{m,l+1,t} \quad \forall l\in L_m-\{\bar{l}_m\}, m, t\in T-\{\bar{t}\} \tag{15}$$

The term defined above can be expressed as the following disjunction:

$$\begin{bmatrix} Y_{m,l+1,t} \\ TX_{m,l,t} = TRT_{m,l,t}\end{bmatrix} \vee \begin{bmatrix} \neg Y_{m,l+1,t} \\ TX_{m,l,t} = 0\end{bmatrix} \tag{16}$$

The disjunction in constraint (16) states that if $Y_{m,l+1,t}$ is true, then $TX_{m,l,t}$ is equal to $TRT_{m,l,t}$; otherwise $TX_{m,l,t}$ is zero. Using the convex hull transformation (Raman and Grossmann, 1994) of constraint (16) we obtain the following constraints:

$$TRT_{m,l,t} = TRT1_{m,l,t} + TRT2_{m,l,t} \quad \forall l\in L_m-\{\bar{l}_m\}, m, t\in T-\{\bar{t}\} \tag{17}$$

$$TX_{m,l,t} = TRT1_{m,l,t} \quad \forall l\in L_m-\{\bar{l}_m\}, m, t\in T-\{\bar{t}\} \tag{18}$$

$$TRT1_{m,l,t} \le TRBound_m\cdot Y_{m,l+1,t} \quad \forall l\in L_m-\{\bar{l}_m\}, m, t\in T-\{\bar{t}\} \tag{19}$$

$$TRT2_{m,l,t} \le TRBound_m \cdot (1 - Y_{m,l+1,t}) \quad \forall l \in L_m - \{\overline{l}_m\}, m, t \in T - \{\overline{t}\} \tag{20}$$

where $TRBound_m = \underset{i \in I_m}{Max}\{\underset{i' \in I_m}{Max}\, \tau^m_{i,i'}\}$ is a parameter defining the maximum transition time for unit *m*.

$$Te_{m,l,t} \le Ts_{m,l+1,t} \quad \forall l \in L_m - \{\overline{l}_m\}, m, t \tag{21}$$

Constraint (21) ensures that the end time of one slot is less than or equal to the start time of the preceding slot. Note that idle times are allowed. The inequality in constraint (22) guarantees that the end time of the last slot in time period *t* cannot exceed the duration of time period t. Constraints (23) and (24) ensure that the start time of the first slot in each period is greater than or equal to the start of the corresponding period. Note that we are not forcing the first slot to begin at the start time of the related time period.

$$Te_{m,l,t} \le HT_t \quad l = \overline{l}_m, \forall m, t \tag{22}$$

$$Ts_{m,l,t} \ge 0 \quad l = 1, \forall m, t = 1 \tag{23}$$

$$Ts_{m,l,t+1} \ge HT_t \quad l = 1, \forall m, t \in T - \{\overline{t}\} \tag{24}$$

3.2.3. Transition costs within each time period:

As we have mentioned before, transitions also involve costs, which reflect utilities used and the disposal costs for the transition product. The parameter $C\tau^m_{i,i'}$ defines the cost required to switch from product *i* to *i'* on unit *m*. The transition costs within each period are then represented by variable $TRACOST1$ as defined in constraint (25).

$$TRACOST1 = \sum_i \sum_{i'} \sum_m \sum_l \sum_t C\tau^m_{i,i'} \cdot Z^t_{i,i',m,l} \tag{25}$$

3.2.4. Transition costs across adjacent weeks:

Calculation of the transition costs for changeovers across adjacent periods follows a similar treatment as the calculation of the transition times.

$$TRACOST2 = \sum_i \sum_{i'} \sum_m \sum_l \sum_t C\tau^m_{i,i'} \cdot \hat{Z}^t_{i,i',m,l} + (\sum_i \sum_{i'} \sum_m \sum_l \sum_t C\tau_{i,i'} \cdot \tilde{Z}^t_{i,i',m,l}) \cdot (1 - Y_{m,l+1,t}) \tag{26}$$

The first term in constraint (26) reflects the transition cost for the case when all the postulated slots are utilized, while the second term is valid if there are null slots present in the end of the time period. Note, however, that the second term is nonlinear. We circumvent this difficulty in an analogous way to constraints (15)-(20). We define the variables $TXT_{m,l,t}$ and $TXX_{m,l,t}$ as follows.

$$TXT_{m,l,t} = \sum_i \sum_{i'} C\tau_{i,i'} \cdot \tilde{Z}_{i,i',m,l,t} \quad \forall l \in L_m - \{\overline{l}_m\}, m, t \in T - \{\overline{t}\} \tag{27}$$

$$TXX_{m,l,t} = TXT_{m,l,t} \cdot Y_{m,l+1,t} \quad \forall l \in L_m - \{\overline{l}_m\}, m, t \in T - \{\overline{t}\} \tag{28}$$

Constraint (26) then becomes:

$$TRACOST2 = \sum_i \sum_{i'} \sum_m \sum_l \sum_t C\tau^m_{i,i'} \cdot \hat{Z}_{i,i',m,l,t} + \sum_m \sum_l \sum_t (TXT_{m,l,t} - TXX_{m,l,t}) \tag{29}$$

$$\begin{bmatrix} Y_{m,l+1,t} \\ TXX_{m,l,t} = TXT_{m,l,t} \end{bmatrix} \vee \begin{bmatrix} \neg Y_{m,l+1,t} \\ TXX_{m,l,t} = 0 \end{bmatrix} \tag{30}$$

Applying the convex hull yields,

$$TXT_{m,l,t} = TXT1_{m,l,t} + TXT2_{m,l,t} \quad \forall l \in L_m - \{\overline{l}_m\}, m, t \in T - \{\overline{t}\} \tag{31}$$

$$TX_{m,l,t} = TRT1_{m,l,t} \quad \forall l \in L_m - \{\overline{l}_m\}, m, t \in T - \{\overline{t}\} \tag{32}$$

$$TXT1_{m,l,t} \le TRCBound_m \cdot Y_{m,l+1,t} \quad \forall l \in L_m - \{\overline{l}_m\}, m, t \in T - \{\overline{t}\} \tag{33}$$

$$TXT2_{m,l,t} \le TRCBound_m \cdot (1 - Y_{m,l+1,t}) \quad \forall l \in L_m - \{\overline{l}_m\}, m, t \in T - \{\overline{t}\} \tag{34}$$

where $TRCBound_m = \underset{i \in I_m}{Max}\{\underset{i' \in I_m}{Max} C\tau^m_{i,i'}\}$ defines the maximum transition cost for unit m.

3.3. Mass and Inventory Balances:

A subset of the end products is produced in a single reaction stage while the remaining final products require intermediates, thus involving two reaction stages with intermediate storage.
We group the products produced in the plant into three categories:

(i) End products that are also intermediate products $i \in IFINT_i$

(ii) End products that are produced in two stages $i \in IE_i$

(iii) End products that are produced in a single stage $i \in IF_i$

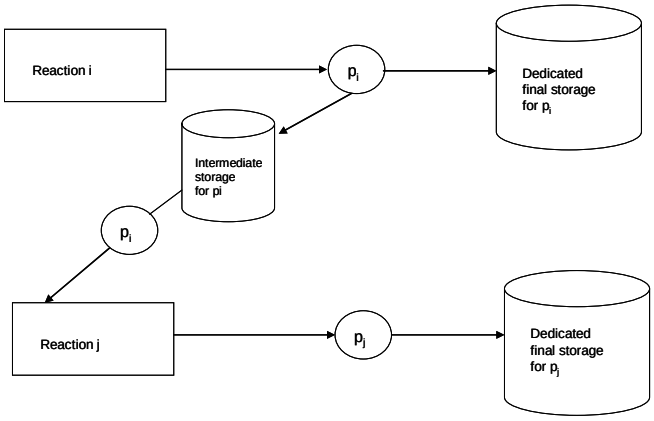


Figure 8. Intermediates and End Products

Due to the layout of the plant, once an end product is transferred to the dedicated storage tank, it can not be retrieved back to the plant. Therefore, after the processing of products ($i \in IFINT_i$), which are both intermediates and end products is completed, each batch is split and transferred to the intermediate storage tanks and the dedicated storage tanks (see Figure 8). End products that require intermediate products as raw materials and end products that are produced in a single stage on the other hand are transferred directly to the dedicated final storage tanks and distributed to satisfy customer demands.

An intermediate product can be consumed by more than one batch of the same or different end products with different mass balance coefficients. Although the batch sizes are constant, the intermediate usage levels should be monitored since the intermediate involves batch splitting. In order to prevent any violations of the use of intermediate product, we perform mass and inventory balances based on time slots instead of time periods.

3.3. 1. Mass Balances for end products that are also intermediate products:

Once the production of the intermediate product in slot *l* of unit *m* at time period *t* is completed, it can:
(1) be transferred ($INVP^{FIN}_{i,m,l,t}$) to the dedicated final storage tanks so as to meet the customer demands.
(2) be transferred ($INVINT^{TRA}_{i,m,l,t}$) to the intermediate storage tanks to satisfy the raw material requirements of the corresponding end products ($i \in IE_i$) that are produced within the plant (see Figure 9).

$$X_{i,m,l,t} = INVP^{FIN}_{i,m,l,t} + INVINT^{TRA}_{i,m,l,t} \quad \forall i \in (IFINT_i \cap I_m), l \in L_m, m, t \tag{35}$$

The amount that is transferred to the intermediate storage tank during period *t* ($INVINT^{TRA}_{i,m,l,t}$) can then:
Case 1: be used to satisfy the raw material requirements of the end product that is being produced in the same unit (*m*), in slots (*l'>l*) and in the same time period (*t*) as the intermediate.
Case 2: be used to satisfy the raw material requirements of the end product that is being produced in another unit (*m'),* in available slots and in the same time period (*t*) as the intermediate.
Case 3: be accumulated in the intermediate storage tank to be used to satisfy raw material demands in the subsequent time periods.

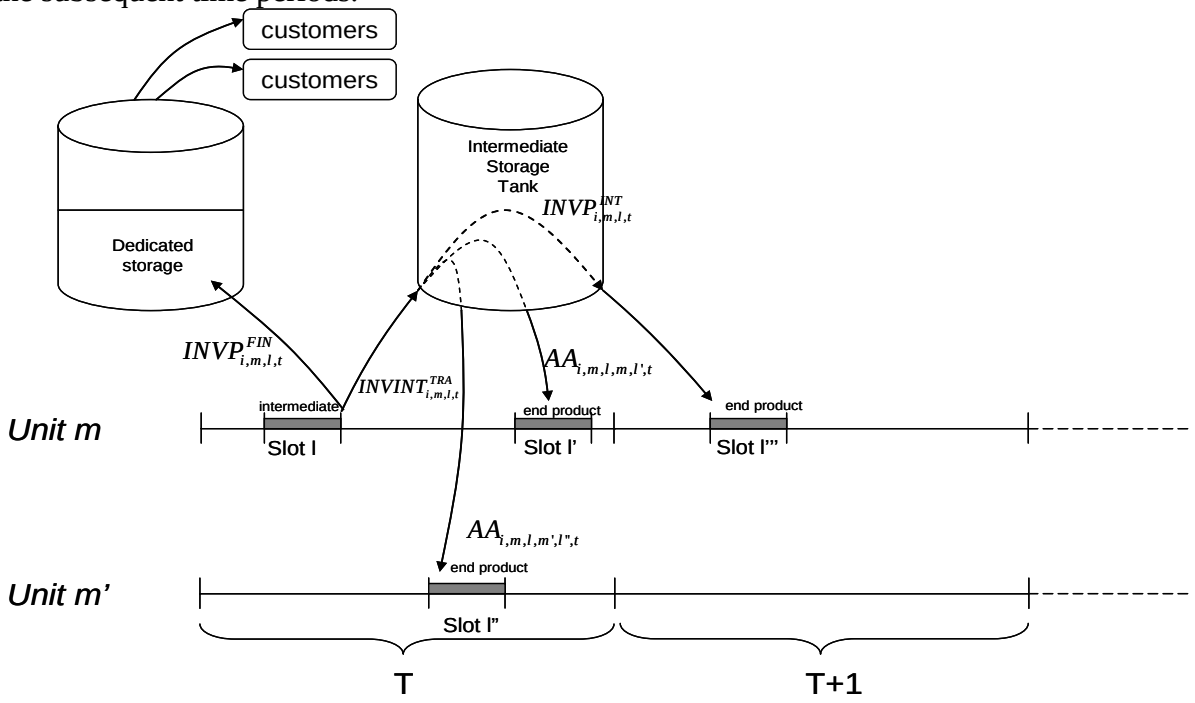


Figure 9. Distribution of the intermediate product

In order to be able to monitor the changes in the intermediate storage tank we define the following variables (see Figure 9).
$AA_{i,m,l,m,l',t}$: This represents the amount of $i \in IFINT_i$ produced in slot *l* of unit *m* at time period *t* that can be transferred to slot *l'* of unit *m* at time *t* to satisfy the raw material requirements for the end product assigned to slot *l'* of unit *m* (case 1).

$AA_{i,m,l,m',l',t}$: This represents the amount of $i \in IFINT_i$ produced in slot *l* of unit *m* at time period *t* that can be transferred to slot *l'* of unit *m'* ($m' \neq m$) at time *t* to satisfy the raw material requirements for the end product assigned to slot *l'* of unit *m'* (case 2).

$INVP^{INT}_{i,m,l,t}$: This variable represents the amount of $i \in IFINT_i$ produced in slot *l* of unit *m* at time period *t* that can be accumulated in the intermediate storage tank so as to satisfy the raw material demands in the subsequent time periods (case 3).

$$INVINT^{TRA}_{i,m,l,t} = INVP^{INT}_{i,m,l,t} + \sum_{\substack{l'>l \\ l' \in L(m)}} AA_{i,m,l,m,l',t} + \sum_{m' \neq m} \sum_{l' \in L(m')} AA_{i,m,l,m',l',t} \quad i \in (IFINT_i \cap I_m), l \in L_m, m, t \qquad (36)$$

Constraint (36) represents the distribution of the intermediate product that has been transferred to the intermediate storage tank as seen in Figure 9. Since the time required to transfer materials to and from storage is assumed to be negligible, the slot based mass balance for feasible allocation of the intermediate products $i \in IFINT_i$ can be defined as follows:

$$X_{i,m,l,t} = INVP^{FIN}_{i,m,l,t} + \sum_{\substack{l'>l \\ l' \in L(m)}} AA_{i,m,l,m,l',t} + \sum_{m' \neq m} \sum_{l' \in L(m')} AA_{i,m,l,m',l',t} + INVP^{INT}_{i,m,l,t} \quad i \in (IFINT_i \cap I_m), l \in L_m, m, t \qquad (37)$$

The first term of constraint (37) represents the amount that is transferred to the dedicated storage tanks to satisfy customer demands whereas the rest of the terms represent the amount that is transferred to satisfy the raw material requirements within the plant and to the intermediate storage tanks (see Figure 9).

Note that an intermediate product assigned to slot *l* can supply for the production of an end product that is assigned to slot *l'* at time period *t* only if slot *l* is completed before slot *l'* starts. For intermediates and end products that are assigned on the same unit, forcing *l'* to be greater than *l* will guarantee that this requirement is satisfied (See Figure 10).

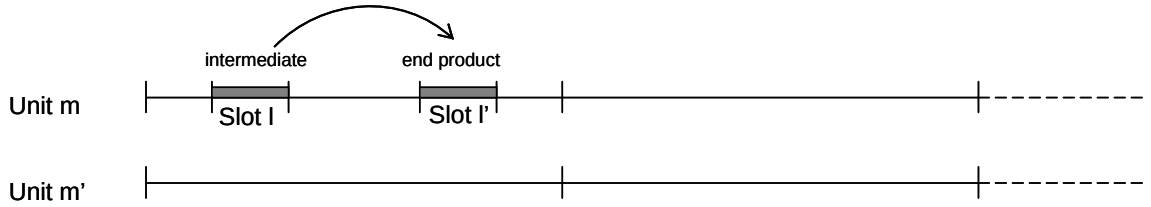


Figure 10. Material transfer within the same unit in the same time period

However, for intermediates and end products which are assigned to different units, the postulated slots *l* and *l'* have variable lengths and are not synchronized across the parallel units. Thus, we need to determine their relative location. Therefore we introduce the binary variable $YY_{l,l',m,m',t}$ which becomes 1 if slot *l* of unit *m* is completed before slot *l'* of unit *m'* starts at time period *t* (see Figure 11a).

The following big-M constraints determine the relative location of slots belonging to parallel units *m* and *m'*:

$$Ts_{m,l,t} + PT_{m,l,t} \leq Ts_{m',l',t} + HT_t \cdot (1 - YY_{l,l',m,m',t}) \quad \forall l \in L_m, l' \in L_{m'}, m, m' \neq m, t \qquad (38)$$

$$Ts_{m',l',t} \leq Ts_{m,l,t} + PT_{m,l,t} + HT_t \cdot (YY_{l,l',m,m',t}) \quad \forall l \in L_m, l' \in L_{m'}, m' \neq m, t \qquad (39)$$

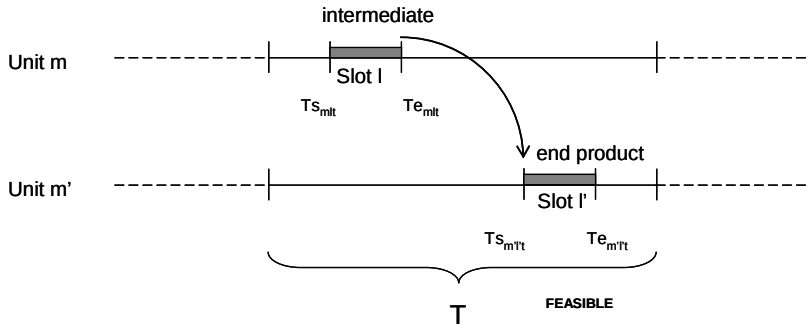


Figure 11a. Slot *l* completed before slot *l'* starts, feasible transfer

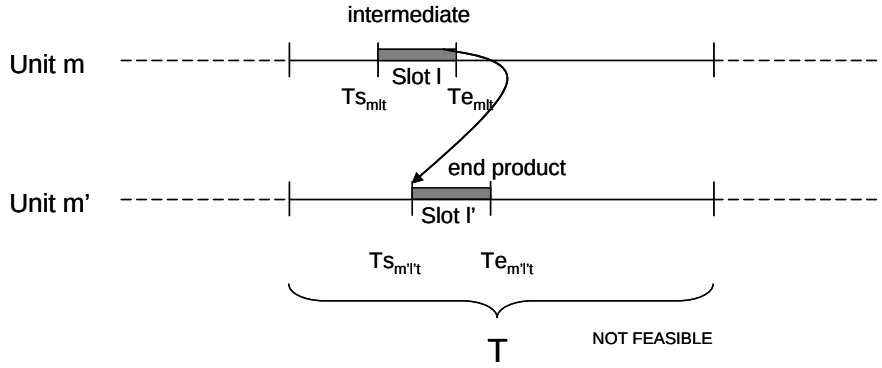


Figure 11b. Slot *l'* starts before slot *l* is completed, infeasible transfer

According to (38), if slot *l* of unit *m* is completed before slot *l'* of unit *m'* starts then the end time of slot *l* of unit *m* is less than or equal to the start time of slot *l'* of unit *m'* during time period *t* (see Figure 11a). If not then according to (39) the start time of slot *l'* of unit *m'* is less than or equal to the end time of slot *l* of unit *m* at time period *t* (see Figure 11b).

$$AA_{i,m,l,m',l',t} \le Q_{i,m} \cdot (YY_{l,l',m,m',t}) \quad i \in (IFINT_i \cap I_m), l \in L_m, l' \in L_{m'}, m' \ne m, t \tag{40}$$

Constraint (40) states that, the amount of intermediate that is produced in slot *l* of unit *m* to supply for the end product that is assigned to slot *l'* of unit *m'* during period *t* is zero unless slot *l* is completed before slot *l'* starts. The batch size of reactor *m* for product *i* is a valid upper bound on the amount of product *i* that can be transferred.

$$AA_{i,m,l,m',l',t} \le Q_{i,m} \cdot \sum_{\substack{i' \in SEI_{i',i} \\ i' \in (I_{m'} \cap IE_{i'})}} (W_{i',m',l',t}) \quad i \in (IFINT_i \cap I_m), \forall l \in L_m, \forall l' \in L_{m'}, \forall m, \forall t \tag{41}$$

Note that, constraint (37) ensures that the amount of intermediate product *i* transferred from slot *l* of unit *m* during time period *t* to any slot *l'* of any unit *m'* is zero unless the corresponding intermediate is assigned to slot *l* of unit *m* during time *t*. Constraint (41), on the other hand, ensures that the transfer of intermediate product from any slot *l* of unit *m* to slot *l'* of unit *m'* during time period *t* is zero unless an end product requiring the corresponding intermediate product is assigned to slot *l'* of unit *m'* during time period *t*.

3.3. 2. Inventory Balances for end products that are also intermediate products:

We introduce the following variables to establish the inventory balances (see Figure 12):

$INV_{i,t}^{INT}$: The inventory level of $i \in IFINT_i$ in the intermediate storage tank at the end of time period *t*.
$INVID_{i,m,l,t}^{INT}$: This variable represents the amount of intermediate that is transferred from the initial inventory in the intermediate storage tank ($INVI_i^{INT}$) to slot *l* of unit *m* during the first time period in order to supply for the end product assigned to slot *l* of unit *m* in the first time period.
$INVC_{i,m,l,t+1}$: This variable represents the amount of intermediate consumed from the intermediate storage tank in order to supply for the end product that is assigned to slot *l* of unit *m* in the subsequent time period.
$INV_{i,t}^{FIN}$: This variable stands for the inventory level of $i \in IFINT_i$ in the dedicated final storage tank at the end of time period *t*.

Constraint (42) represents the inventory balances for the intermediate storage tanks for the first time period. The inventory level of product $i \in IFINT_i$ at the end of the first time period in the intermediate storage tank is given by the initial inventory level in the intermediate storage tank, $INVI_i^{INT}$, plus the amount produced in slot *l* of unit *m* and transferred to the intermediate storage tank in the first time period, $INVP_{i,m,l,t}$, minus the amount transferred from the initial inventory to satisfy the raw material demands for the end products that are assigned to slot *l'* of unit *m* in the first time period, $INVID_{i,m,l',t}^{INT}$ minus the amount consumed from the intermediate storage tank to satisfy the raw material demands for the end products that are assigned to slot *l"* of unit *m* in the subsequent time period, $INVC_{i,m,l",t+1}$ (see Figure 12). The inventory balances for the intermediate storage tanks for time periods greater than one is expressed with constraint (43). Note that the term $INVID_{i,m,l',t}^{INT}$ drops in constraint (43) since the variable $INVID_{i,m,l',t}^{INT}$ represents the amount of intermediate that is transferred from the initial inventory to supply for the end products that are assigned to slot l' of unit m only in the first time period.

$$INV_{i,t}^{INT} = INVI_i^{INT} + \sum_m \sum_{\substack{l \in L_m \\ l < |L_m|}} INVP_{i,m,l,t} - \sum_m \sum_{\substack{l' \in L_m \\ l' < |L_m|}} INVID_{i,m,l',t}^{INT} - \sum_m \sum_{\substack{l" \in L_m \\ l" < |L_m|}} INVC_{i,m,l",t+1} \qquad i \in (IFINT_i \cap I_m), t = 1 \tag{42}$$

$$INV_{i,t}^{INT} = INV_{i,t-1}^{INT} + \sum_m \sum_{\substack{l \in L_m \\ l < |L_m|}} INVP_{i,m,l,t} - \sum_m \sum_{\substack{l" \in L_m \\ l" < |L_m|}} INVC_{i,m,l",t+1} \qquad i \in (IFINT_i \cap I_m), t > 1 \tag{43}$$

$$\sum_m \sum_{\substack{l \in L_m \\ l < |L_m|}} INVID_{i,m,l,t}^{INT} \le INVI_i^{INT} \qquad i \in (IFINT_i \cap I_m), t = 1 \tag{44}$$

Constraint (44) imposes an upper bound on $INVID_{i,m,l,t}^{INT}$ such that the amount of intermediate product distributed cannot exceed the amount that was present initially.

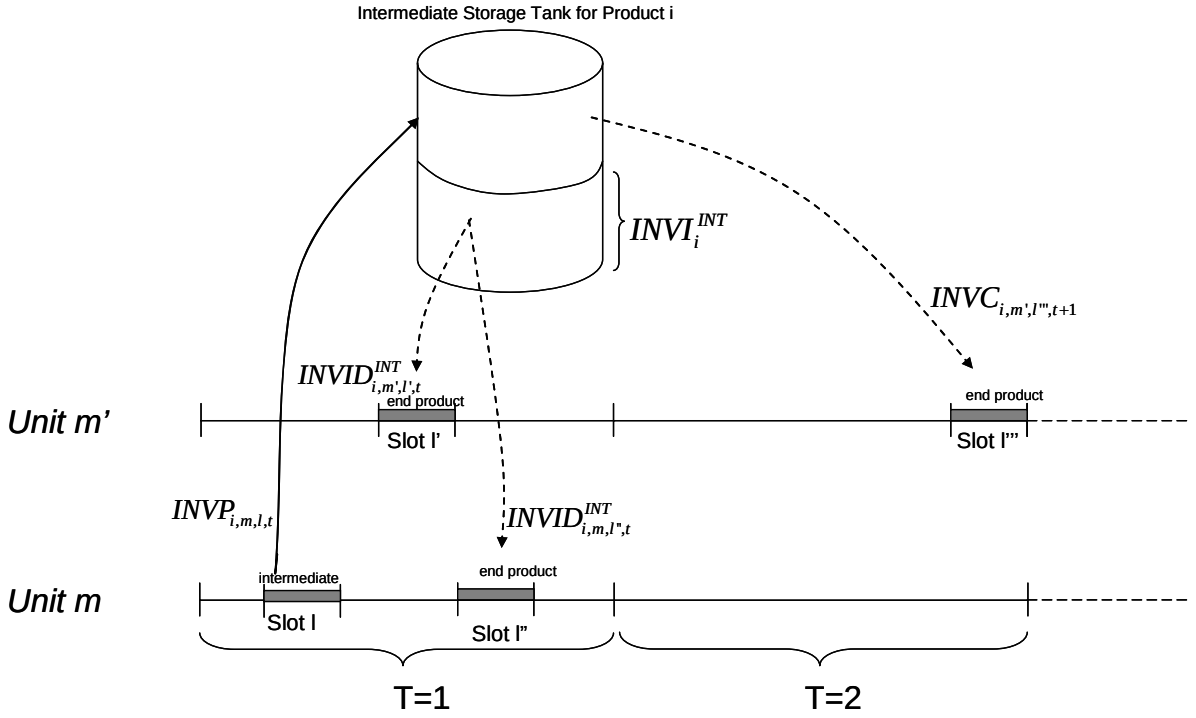


Figure 12. Schematic representation of inventory balances

$$INVC_{i,m,l,t} \le CAP_i^{INT} \cdot \sum_{\substack{i' \in SEI_{i',i} \\ i' \in (I_{m'} \cap IE_{i'})}} W_{i',m,l,t} \quad \forall i \in (IFINT_i \cap I_m), l \in L_m, m, t \tag{45}$$

According to constraint (45), the amount of intermediate product *i* transferred from the intermediate storage tank to slot *l* of unit *m* during time period *t* is zero unless an end product requiring the corresponding intermediate is assigned to slot *l* of unit *m* during time period *t* and also is bounded by the capacity of the intermediate storage tank, CAP_i^{INT}, for the intermediate product *i*.

Given $S_{i,t}$, the sales of product *i* in period *t*, constraints (46) and (47) represent the inventory balances for the dedicated final storage tank.

$$INV_{i,t}^{FIN} = INVI_i^{FIN} + \sum_m \sum_{l \in L_m} INVP_{i,m,l,t}^{FIN} - S_{i,t} \qquad \forall i \in (IFINT_i \cap I_m), t = 1 \tag{46}$$

$$INV_{i,t}^{FIN} = INV_{i,t-1}^{FIN} + \sum_m \sum_{l \in L_m} INVP_{i,m,l,t}^{FIN} - S_{i,t} \qquad \forall i \in (IFINT_i \cap I_m), t > 1 \tag{47}$$

From constraints (46) and (47), the inventory level $INV_{i,t}^{FIN}$ of product $i \in IFINT_i$ at the end of time *t* in the final dedicated storage tank is equal to the sum of the initial inventory plus the amount produced and transferred to the final dedicated storage tank minus the sales to satisfy the customer demands.

3.3. 3. Mass and Inventory Balances for end products that are produced in 2 stages:

The mass and inventory balances presented in this section are included to guarantee that enough intermediate $i' \in IFINT_{i'}$ is available before the production of the end product $i \in IE_i$ requiring the intermediate $i' \in IFINT_{i'}$ starts.

The sources of intermediate $i' \in IFINT_{i'}$ required to produce the end product $i \in IE_i$ that is assigned to slot *l* of unit *m* during time period *t* can be categorized as follows (see Figure 13 and Figure 14):

(i) the initial inventory in the intermediate storage tank ($INVID^{INT}_{i',m,l,t}$) for $t = 1$.

(ii) the accumulated inventory in the intermediate storage tank ($INVC_{i',m,l,t}$) for $t > 1$.

(iii) the intermediate that was produced in the same unit in the same time period but in a slot (*l'*<*l*) that precedes slot *l* ($AA_{i',m,l',m,l,t}$).

(iv) the intermediate that was produced in another unit in the same time period but in a slot that was completed before slot *l* starts ($AA_{i',m',l'',m,l,t}$).

Constraints (48) and (49) ensure that the production of the end product $i \in IE_i$ in slot *l* of unit *m* in time period *t* will not be performed unless the required intermediates $i' \in IFINT_{i'}$ are present in the required amounts for $t = 1$ and $t > 1$, respectively. The parameter $\alpha_{i,i'}$ represents the amount of intermediate $i' \in IFINT_{i'}$ consumed by a unit size of the end product $i \in IE_i$.

$$X_{i,m,l,t} = \sum_{\substack{i' \in SEI_{i,i'} \\ i' \in (IFINT_i \cap I_m)}} (1/\alpha_{i,i'}) \cdot INVID^{INT}_{i',m,l,t} + \sum_{\substack{i' \in SEI_{i,i'} \\ i' \in (IFINT_i \cap I_m)}} (1/\alpha_{i,i'}) \cdot \sum_{l' > l \atop l' \in (L(m) \cap L(t))} AA_{i',m,l',m,l,t}$$
$$+ \sum_{\substack{i' \in SEI_{i,i'} \\ i' \in (IFINT_i \cap I_m)}} (1/\alpha_{i,i'}) \cdot \Big(\sum_{\substack{m' \neq m \\ m \in M_{i'}}} \sum_{l'' \in (L(m') \cap L(t))} AA_{i',m',l'',m,l,t} \Big) \quad \forall i \in (IE_i \cap I_m), l \in (L(m) \cap L(t)), m, t = 1 \tag{48}$$

The first term of constraint (48) represents the amount of intermediate $i' \in IFINT_{i'}$ that is transferred from the initial inventory present in the intermediate storage tank to supply for the end product $i \in IE_i$ that is assigned to slot *l* of unit *m* in the first time period, $INVID^{INT}_{i',m,l,t}$; the second term stands for the intermediate produced in an earlier slot *l'* of the same unit *m* in the first time period and transferred to supply for the end product assigned to slot *l* of unit *m*, $AA_{i',m,l',m,l,t}$ and finally the last term represents the intermediate that is produced in slot *l''* of a different unit *m'* and transferred to supply for the end product that is assigned to slot *l* of unit *m*, $AA_{i',m',l'',m,l,t}$ (see Figure 13).

Constraint (49) is in essence the same as constraint (48) except that it is defined for time periods greater than one and hence the first term $INVID^{INT}_{i',m,l,t}$ is replaced by the term $INVC_{i'm,l,t}$, which stands for the intermediate $i' \in IFINT_{i'}$ transferred from the intermediate storage tank to supply for the end product that is assigned to slot *l* of unit *m* during time period *t* (see Figure 14).

$$X_{i,m,l,t} = \sum_{\substack{i' \in SEI_{i,i'} \\ i' \in (IFINT_i \cap I_m)}} (1/\alpha_{i,i'}) \cdot INVC_{i'm,l,t} + \sum_{\substack{i' \in SEI_{i,i'} \\ i' \in (IFINT_i \cap I_m)}} (1/\alpha_{i,i'}) \cdot \sum_{\substack{l' > l \\ l' \in (L(m) \cap L(t))}} AA_{i',m,l',m,l,t}$$
$$+ \sum_{\substack{i' \in SEI_{i,i'} \\ i' \in (IFINT_i \cap I_m)}} (1/\alpha_{i,i'}) \cdot \Big(\sum_{\substack{m' \neq m \\ m \in M_{i'}}} \sum_{l'' \in (L(m') \cap L(t))} AA_{i',m',l'',m,l,t} \Big) \quad \forall i \in (IE_i \cap I_m), l \in (L(m) \cap L(t)), m, t > 1 \tag{49}$$

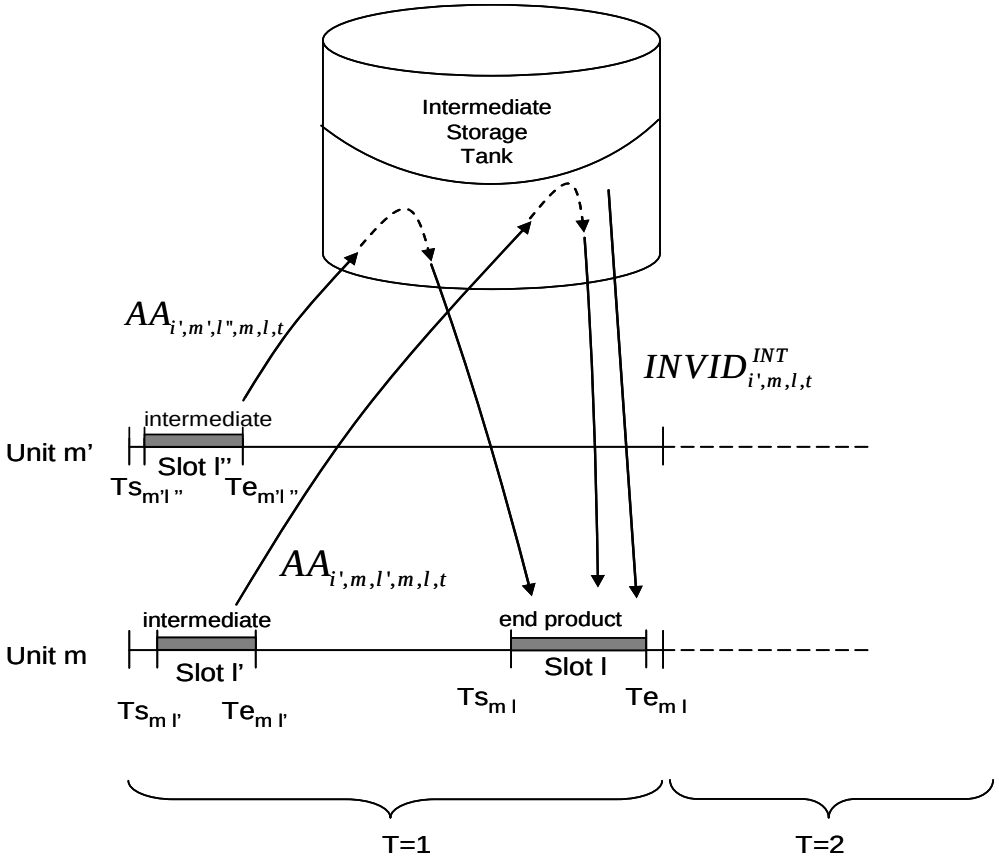


Figure 13. Material transfer for the end product assigned to the first time period

$$INV^{FIN}_{i,t} = INV^{FIN}_{i,t-1} + \sum_{m \in MI(i)} \sum_{l \in (L(m) \cap L(t))} X_{i,m,l,t} - S_{i,t} \quad i \in IE_i, \forall t \tag{50}$$

Constraint (50) represents the overall inventory balance for the end products that are produced in 2 stages. The inventory level of product *i* at the end of time period *t* in the final storage tank is equal to the initial inventory level plus the amount of product $i \in IE_i$ produced in time period *t* minus the sales of *i* at the end of time period *t*.

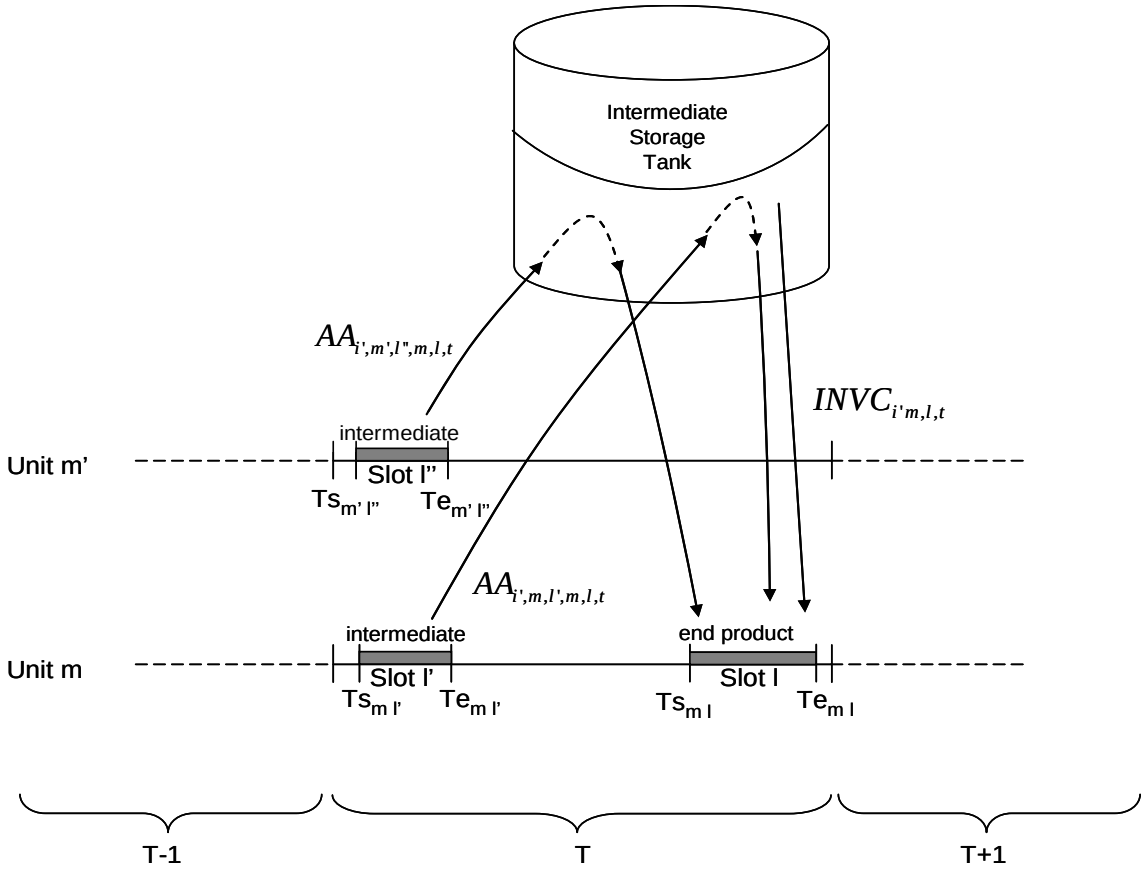


Figure 14. Material transfer for the end product assigned to periods greater than one

3.3. 4. Mass and Inventory Balances for end products that are produced in a single stage:

Products that are produced in a single stage are directly transferred to the dedicated final storage tanks upon the completion of the production so as to satisfy the customer demands. Since handling of these products do not include batch splitting and also since sales occur only at the end of each time period, there is no need to keep track of changes on a slot basis. Hence, mass and inventory balances hold for each time period as shown by constraint (51). The change in the inventory level of product $i \in IF_i$ is equal to the difference between the production and the sales.

$$INV_{i,t-1}^{FIN} + \sum_{m \in MI(i)} \sum_{l \in (L(m) \cap L(t))} X_{i,m,l,t} = S_{i,t} + INV_{i,t}^{FIN} \quad i \in IF_i, \forall t \tag{51}$$

3.4. Demand Fulfillment:

Product demands can be posed either as lower bounds or upper bounds. Constraint (52a) specifies that, demands must be satisfied for all products and that amount of products shipped to the customers can exceed the demands. Constraint (52b) on the other hand specifies the maximum projected demands that can be sold in the market for each time period, hence amounts of products shipped must not exceed the specified demands.

$$S_{i,t} \geq D_{i,t} \qquad \forall i,t \tag{52a}$$

$$S_{i,t} \le D_{i,t} \qquad \forall i,t \tag{52b}$$

3.6. Minimum number of batches:

Due to operational reasons, the minimum number of batches, MRL_i, can be defined for each product. Constraint (53) imposes this condition.

$$\sum_{l} W_{i,m,l,t} \ge MRL_i \cdot W_{i,m,l,t} \qquad \forall i \in I_m, m, t \tag{53}$$

3.7. Objective Function:

$$\begin{aligned} \Pr ofit = & \sum_{i}\sum_{t} C^{P}_{i,t} \cdot S_{i,t} - \sum_{i}\sum_{m}\sum_{l}\sum_{t} C^{OP}_{i,t} \cdot XB_{i,m,l,t} - \sum_{i}\sum_{t} C^{INV}_{i,t} \cdot INV^{FIN}_{i,t} \\ & - \sum_{i \in IFINT}\sum_{t} C^{INV}_{i,t} \cdot INV^{INT}_{i,t} - \sum_{i \in IFINT_i}\sum_{t \ne 1}\sum_{m \in M_i}\sum_{\substack{l \in L_m \\ l < |L_m|}} INVC_{i,m,l,t} - TRACOST1 - TRACOST2 \end{aligned} \tag{54}$$

The profit is given by the sum of sales revenues, the operating costs, the inventory costs, the total transition costs within each time period and transition costs across adjacent weeks. The third, fourth and fifth terms of the objective function represent the inventory costs. While the third and fourth terms correspond to the final and intermediate inventory costs, respectively, the fifth term reflects the amount of intermediate products that are to be consumed in the subsequent period but stored in the current time period.

3.8. Tightening Constraint:

In principle, when two or more batches of the same product are assigned to the same unit in the same time period, it is guaranteed implicitly by optimality that the corresponding product will be assigned to adjacent slots due to the trade-off between inventory and transition costs. Nevertheless, we can exploit this property to improve the tightness of the model by enforcing the condition that there can not be more than one transition for any given product that is assigned to unit *m* in time period *t*.

$$\sum_{\substack{i' \ne i \\ i' \in I_m}} \sum_{\forall l \in L_m - \{\bar{l}_m\}} Z_{i,i',m,l,t} \le 1 \quad \forall i, \in I_m, m, t \tag{55}$$

4. Motivating Example:

In order to analyze the computational performance of the proposed MILP formulation, we solve a small example consisting of 5 Products and 2 Reactors for a time horizon of up to 2 weeks. The schematic representation of this example is shown in Figure 15. Each reactor can process any of the products. All products except product B are produced in a single stage whereas product B requires A as an intermediate.

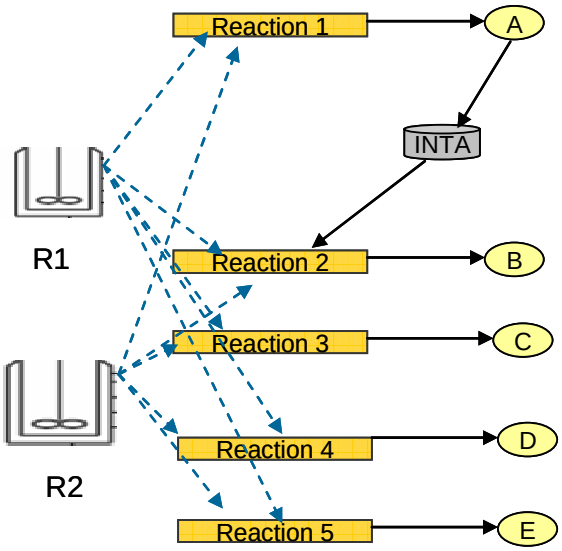


Figure 15. Schematic representation for the motivating example

The batch times for the motivating example are illustrated in Table 1 and the rest of the data for this example can be found in Appendix A1.

Table 1. Batch Times for the Motivating Example

	Batch Time (hrs)	
	R1	R2
A	16	16
B	10	10
C	25	25
D	20	20
E	15	15

In order to avoid infeasible or suboptimal solutions, the minimum number of slots that must be postulated for each unit and each time period, $NS_{m,t} = \left\lfloor \underset{i}{Max}\left\{H_t / BT_{i,m}\right\} \right\rfloor$, is 16 in this case ($NS_{m,t} = \left\lfloor \underset{i}{Max}\{10.5, 16.8, 6.72, 8.4, 11.2\} \right\rfloor = \lfloor 16.8 \rfloor$). Table 2 shows the problem sizes and solution times for this example for 16 slots for 1 week and 2 weeks.

Table 2. Model and Solution Statistics for the Motivating Example

length of time horizon	number of binary variables	number of continuous variables	number of equations	time (CPU s)	solution ($)
1 Week	1,326	5,355	5,111	77.9	302,200
2 Weeks	2,652	10,967	11,088	10,000*	933,255

*Search terminated after 10,000 CPUs, the best feasible solution is posted.

As can be seen from Table 2, even for this small instance addition of another time period nearly doubles the size of the model to 11,088 constraints, 10,967 continuous variables and 2,652 discrete variables and also results in more than 2 orders of magnitude increase in the solution time. Also, as can be seen from Table 3, the number of slots has a significant impact on the problem size and the difficulty of the solution. Hence, there is a need for more efficient solution methods. The rest of the paper focuses

on an alternative solution strategy to address the computational difficulties of solving the proposed model.

Table 3. Variation of the Problem Size with Time Slots for the Motivating Example

1 Week			
number of slots	number of binary variables	number of continuous variables	number of equations
5	150	1,083	767
10	500	2,583	2,157
15	1,050	4,483	4,147
20	1,800	6,783	6,737
2 Weeks			
number of slots	number of binary variables	number of continuous variables	number of equations
5	300	2,231	1,776
10	1,000	5,311	4,816
15	2,100	9,191	9,056
20	3,600	13,871	14,497

5. Solution Strategy:

The main idea of the proposed strategy is to reduce the upper bound for the number of slots. Our goal is to develop a decomposition algorithm to lower the computational expense of this problem by: (1) tightening the upper bounds on the number of slots, (2) reducing the search space of the binary variables. We propose a bi-level decomposition algorithm that involves decomposing the original model into an upper level sequencing and a lower level scheduling model. The upper level model (ULP) is an aggregation of the original problem (OP) where the detailed timing of production and changeovers are replaced with time balances and the slot based mass balances are aggregated over time periods. The main decisions in this level are the assignments of products to available equipment, number of batches of each product as well as the production and inventory levels. Since (ULP) is a relaxation of the original model (OP), it yields a valid upper bound on the profit. Moreover, it also yields a very good upper bound on the number of slots to be postulated for the lower level through the number of batches of each product determined at this level. The lower level problem (LLP) is essentially the same as the original problem (OP) except that it is solved by excluding the products that were not selected by the upper level. Since its solution corresponds to a feasible solution of the original problem, it yields a lower bound on the profit. The decomposition is based on an iterative scheme where the upper level model (ULP) is solved first to determine upper bounds on the profit and number of slots for the lower level, and to predict the products to be produced. The lower level model is solved next for the subset of products and number of slots as determined by (ULP) to obtain a lower bound on the profit. The procedure iterates until the difference between the upper and lower bounds is less than a specified tolerance. Integer cuts are added to the upper level at each iteration to ensure global optimality. The decomposition algorithm is illustrated schematically in Figure 16.

6. Upper Level Model:

The MILP model for the upper level is motivated by the detailed planning model proposed in Erdirik-Dogan and Grossmann, 2007. Although posed initially as a long term planning model by the authors, this model can actually be considered as an aggregation of the original model (OP). In particular, the slot time representation (the index *l*) that has been used in the original model is dropped

hence the assignment and production variables are obtained by aggregating the variables of the original model over time slots. Moreover, detailed timing variables and constraints are ignored since this model deals with determining the sequence of batches of products rather than the detailed schedule. Finally, the mass and inventory balances are performed over time periods as opposed to time slots as in the original model (OP). This however implies that while the mass balances will continue to hold for the products that are produced in two stages, the assignment of an intermediate before the assignment of an end product is not guaranteed.

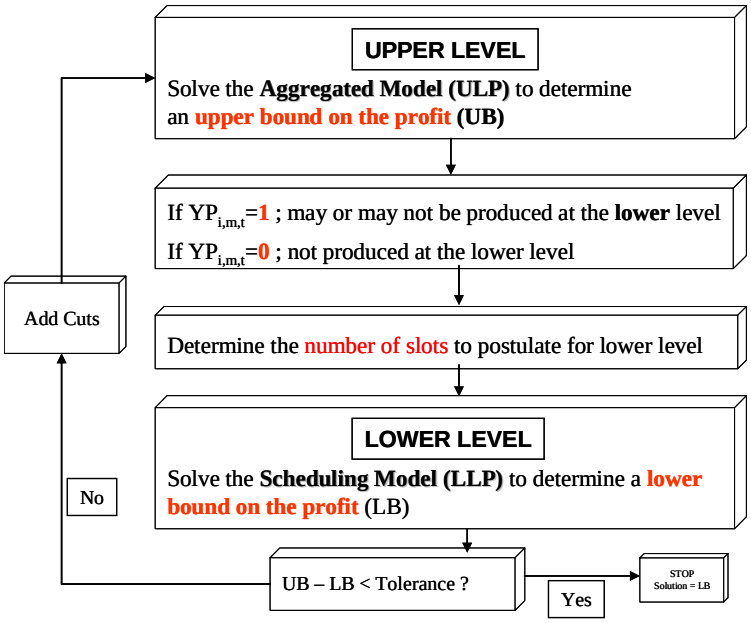


Figure 16. Flow chart for the proposed algorithm

A detailed description of this model can be found in Erdirik-Dogan and Grossmann (2007). However, we include here a brief summary of this model. The general form of the MILP model is follows:

maximize profit
subject to mass balances on each state
sequencing constraints
time balances on each equipment for each time period

The major decisions that the model is concerned with are (i) the assignment of product *i* to unit *m* at each time period *t*, $YP_{i,m,t}$, (ii) number of batches of each product *i* in each unit *m* at each period *t*, NB_{imt}, (iii) corresponding amount of material processed by each task *i*, FP_{imt} (see Figure 17).

The goal of the model is to determine the minimum transition time sequence within the assigned products within each unit and time period, while maximizing the profit and satisfying the demands at the due dates. In this way, the determination and allocation of number of batches of each product and their sequencing are determined simultaneously. The basic idea relies on accounting for the sequence-dependent changeover times and costs without determining the detailed timing of the operations but through sequencing constraints and time balances. This is handled by generating cyclic schedules for each unit and each time period using asymmetric traveling salesman constraints to minimize overall transition times amongst the assigned products, and then by breaking one of the links in the cycle to obtain the optimal sequences.

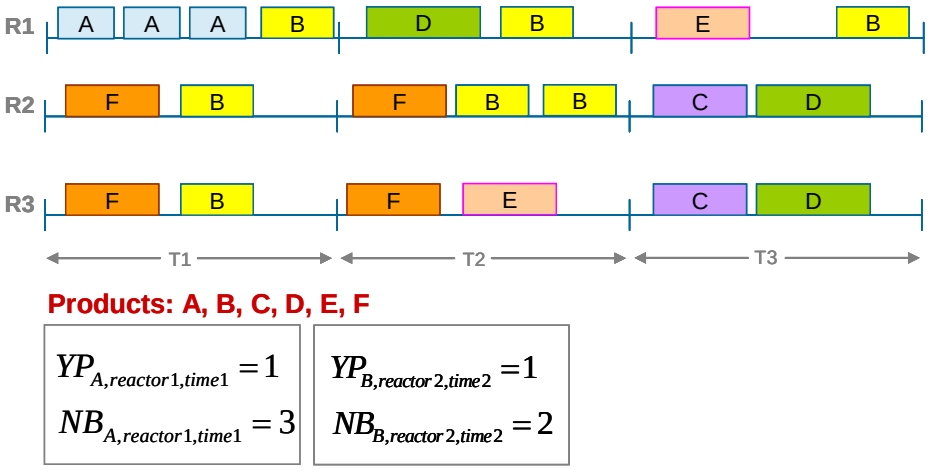


Figure 17. Schematic representation of the variables for the upper level model

Remarks:

1. The model (ULP) is combinatorially less complex than the original problem (OP) due to the fact that its formulation requires three indices *i*, *m* and *t* while the formulation of (OP) requires an additional index *l*.
2. Despite the fact that (ULP) is a lower dimensional problem with respect to (OP), it is still an expensive problem to solve. This is due to the fact that, although the detailed timing of operations is neglected at this level, the model requires additional 0-1 variables and constraints to explicitly account for sequence-dependent changeovers.
2. The formulation represented by (ULP) might exhibit subcycles. In this case, although an actual sequence cannot be obtained since the formulation represents a relaxation, it still yields a valid upper bound on the profit.
3. As stated before, the detailed timing of production is not considered at this level. Therefore, the model (ULP) cannot guarantee the assignment of the intermediate before the associated end product within each time period. This however could lead to discrepancies between the schedules obtained by the upper and lower level.

7. Lower Level:

The lower level model (LLP) is essentially the same as the original scheduling model (OP) except for the following:

(i) Recall that (OP) required an a priori selection of number of slots. Hence, an upper bound for number of slots, NS_m, were postulated at each time period for unit *m*. In order to tighten this upper bound, we exploit knowledge on the number of batches of each product as predicted by the upper level. Specifically, since each time slot *l* is associated with one potential batch of any of the products, the number of batches of product *i* in unit *m* during time period *t*, $NB_{i,m,t}$, obtained at the upper level will represent an upper bound on the number of slots for product *i* in unit *m* during time period *t* in the lower level.

$$NPslot^{r}_{i,m,t} = NB^{r}_{i,m,t} \quad \forall i,m,t \tag{56}$$

According to constraint (56), the number of slots that are allocated to product *i* in unit *m* during time period *t* is set to the number of batches of product *i* in unit *m* during time period *t* as predicted by the upper level (ULP) at iteration *r*.

$$NUslot_{m,t}^{r} = \sum_{i} NPslot_{i,m,t}^{r} \quad \forall m,t \tag{57}$$

The total number of slots postulated in unit *m* for time period *t* at iteration r is then given by constraint (57). Since the model has the flexibility of assigning a particular product to any of the associated identical and/or non-identical units, for a specific time period *t*, the maximum of the number of slots specified for the associated units are postulated as shown in constraint (58). In other words instead of specifying the number of slots for each unit, NS_m, as done in the original model, we specify the number of slots for each time period, at each iteration, $NTslot_t^r$ (see Figure 18).

$$NTslot_{t}^{r} = \underset{m}{Max}\left\{ NUslot_{m,t}^{r} \right\} \quad \forall t \tag{58}$$

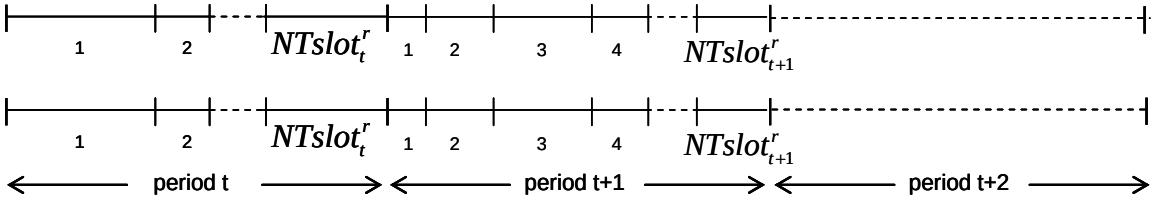


Figure 18. Time slots postulated for each time period in the lower level

We should note that, constraint (57) will result in tighter upper bounds on the number of slots compared to constraint (58). This is because constraint (57) assigns distinct number of slots on each unit and each time period whereas constraint (58) plays it on the safe side and for each time period assigns the maximum value among the associated units. As a result, optimality is not guaranteed with constraint (57). We should also note that, among all the instances we have solved so far, constraint (57) was able to find the global optimal solution.

(ii) In this level, we solve the original model (OP) for a subset of products as predicted by the upper level (UPL). In particular, products that are predicted to be produced by the upper level ($YP_{i,m,t}^{r} = 1$), may or may not be produced. However, those products which were predicted as not to be produced by the upper level ($YP_{i,m,t}^{r} = 0$), are excluded from the lower level by setting to zero the corresponding assignment variables ($W_{i,m,l,t} = 0$). This condition is enforced by the following constraint:

$$\sum_{l} W_{i,m,l,t} \le YP_{i,m,t}^{r} \quad \forall i,m,t \tag{59}$$

8. Integer Cuts:

In this section we consider the integer cuts that expedite the convergence of the proposed decomposition algorithm.

8.1. Cover Cuts:

Subproblems (UL) and (LL) are solved iteratively until the bounds converge. If at iteration *r* the bounds do not lie within a specified tolerance, we need to generate new solutions at the upper level in order to ensure an optimal solution. Also, the solution predicted by the upper level at iteration *r* may lead to an infeasibility in the lower level. In such a case, we need to exclude the corresponding configuration from the upper level. In order to explicitly exclude the current configuration and to forbid infeasible configurations, we add the following integer cut to the upper level:

$$\sum_{(i,m,t)\in Z^r} YP_{i,m,t} \leq \left|Z_1^r\right| - 1 \qquad (60)$$

where Z_1^r is the set of *(i,m,t)* for which the assignment variable $YP_{i,m,t} = 1$.

We should note that, the integer cut in (60), not only excludes the current configuration of iteration *r* but also any other configuration that is a superset of assignment *r* (i.e. any configuration *k* for which $Z_1^k \supseteq Z_1^r$). Hence, by eliminating a large number of feasible configurations, it helps reducing the search space for the binary variables.

8.2. Symmetry Breaking Cuts:

8.2.1. The Source of Symmetries:

As mentioned in the problem statement, the plant can consist of a subset of identical units within each stage. Units are identical when the subset of products they can produce as well as the batch times, batch sizes and operational costs for these products are the same. This means that, the assignment of the same configuration of products (preserving the sequence) on two identical units *R1* and *R2* will result in two distinct solutions which are indeed equivalent and have the same objective value (see Figure 19).

Figure 19. Two equivalent schedules

In other words, given a specific feasible solution, an alternative solution can be obtained by simply renaming/reindexing the units. As an example consider the solution given in Figure 20 for two identical units R1 and R2. The solutions shown as schedule 1 and schedule 2 are globally the same and they differ only by the ordering of the identical units.

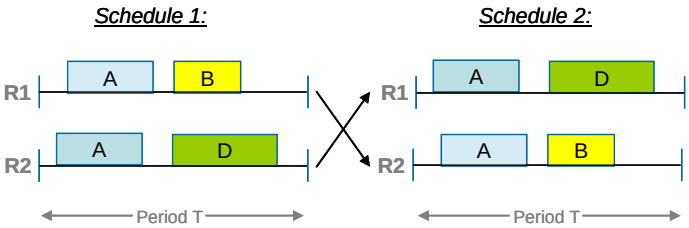


Figure 20. Equivalent solutions for two identical units

To generalize, any permutation of the array of identical units, $R = \{R_1, R_2, R_3, ...R_m\}$, associated with a specific solution will result in an alternative solution. The total number of alternative solutions for this specific solution then will depend on the total number of associated identical units, *M*, and will be given by *M!*.

This is illustrated in Figure 21 for 3 identical reactors, R1, R2 and R3 where 6 (3!) alternative schedules are shown. Given any of these 6 schedules, the remaining 5 can be obtained by permuting/reordering units R1, R2 and R3. Specifically, R1 of schedule 1 was renamed as R2 in schedule 2; R2 of schedule 1 was renamed as R3 in schedule 2 and finally R3 of schedule 1 was renamed as R1 in schedule 2.

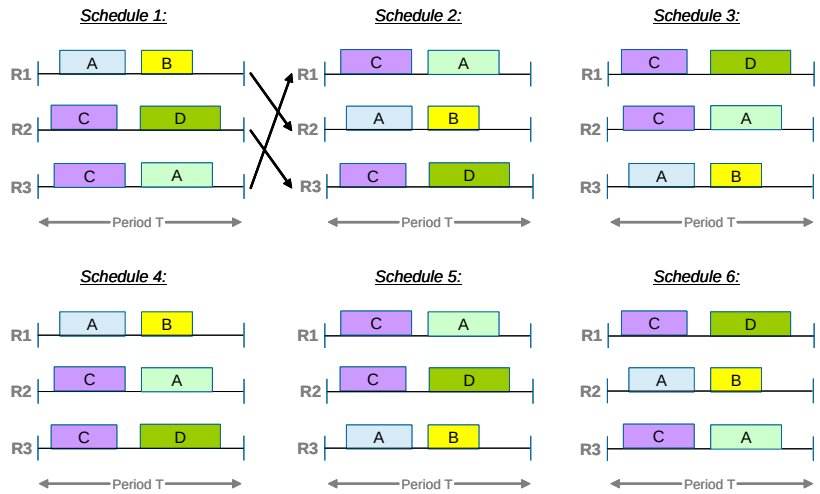


Figure 21. Six equivalent schedules for three identical units

Since identical units are indistinguishable, the proposed model possesses a large number of symmetries that lead to many alternative optimal solutions and several clones of feasible solutions. These symmetries can a substantial adverse effect on the computational effort since they increase the size of the search space in the branch and bound algorithm as well as increase the total number of iterations of the proposed decomposition algorithm. We eliminate these redundant solutions from the search space by adding symmetry breaking constraints that ensure that any solution symmetric to one already considered will not be explored again during the search.

8.2.2. Breaking Symmetries

Our strategy of adding symmetry breaking cuts in a systematic manner is as follows. Since the symmetry arises from the fact that identical units can be permuted to yield equivalent solutions, the basic idea is to add constraints that will be satisfied by only one of the solutions in each equivalence class. More precisely, we propose adding constraints such that the remaining (M!-1) permutations of the solution in consideration become infeasible. In order to be able to propose a general framework for generating these constraints, it is necessary to have a procedure for generating all possible permutations of an array of M elements systematically. In the next section, we describe a new systematic method of generation of permutations.

8.2.2.1. Generation of Permutations by Circular Shifts

The algorithm we propose to generate permutations is motivated by the Nested Cyclic Algorithm of Langdon and is strongly related to the shift cursor algorithm of Gao and Wang.

The proposed algorithm generates all the *(M+1)!* permutations of a list consisting of *(M+1)* elements by using circular right shifts as the basic operation. In the context of this paper, a list $P = \{i_k, i_{k+1}, i_{k+2}, i_{k+3}.....i_{k+M}\}$ will refer to an ordered set of locations where i_k stands for the content of position *k*. Position *k* stands for the leftmost position whereas position *k+M* stands for the rightmost position. The length of the list, *(M+1)*, is the number of elements considered.

The algorithm is as follows. It starts with a particular ordering of *(M+1)* elements and generates the next *(M)* permutations (total of *M+1* permutations) by moving the element occupying the leftmost position (*k*) in the original list to the right by 1 place, 2 places,….., *M* places until it reaches the rightmost position (*k+M*) (see Figure 22). The remaining elements of the list are shifted to the right by 1

place, 2 places,….., *M* places, respectively, in effect performing a circular shift. When the element located in position k shifts to position *k+1*, the elements in positions *k+1, k+2,…,k+M-1* each move one position to the right. Since the element located in position *k+M* occupies the rightmost position, it is moved by one place to the right in a circular manner to position *k*. Let $P = \{i_k, i_{k+1}, i_{k+2}, i_{k+3}.....i_{k+m}\}$ be the original sort, then the 1st permutation (P_1), 2nd permutation (P_2),…,M^{th} permutation (P_m) on P are as follows:

$$P_1 = \{i_{k+M}, i_k, i_{k+1}, i_{k+2}, i_{k+3}....., i_{k+M-2}, i_{k+M-1}\}$$
$$P_2 = \{i_{k+M-1}, i_{k+M}, i_k, i_{k+1}, i_{k+2}, i_{k+3}....., i_{k+M-2}\}$$
$$\vdots$$
$$P_m = \{i_{k+1}, i_{k+2}, i_{k+3}....., i_{k+M-2}, i_{k+M-1}, i_{k+M}, i_k\}$$

For each *(M+1)* circular orderings generated in this manner, another *(M+1)* can be obtained by fixing the element occupying the leftmost position in place in each of the orderings and reapplying the algorithm on the subsort of the elements located in positions *k+1* to *k+M*. Since each ordering leads to *(M+1)* permutations, the algorithm will generate all *(M+1)!* permutations with this recursive scheme. Please see Appendix B for the flowchart of the algorithm.

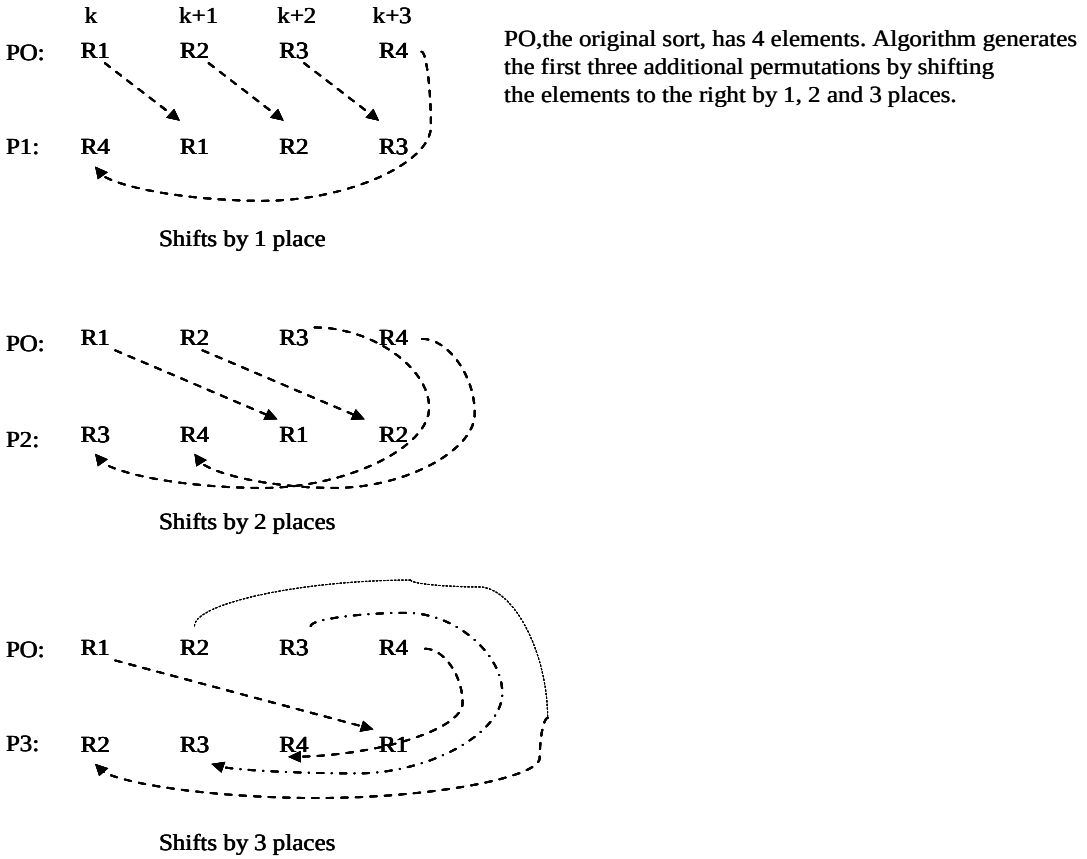


Figure 22. Step 1 of the permutation generation algorithm for (M+1) = 4 elements

8.2.2.2. Generation of Symmetry Breaking Constraints

In the next set of constraints, we eliminate the alternative optima among iterations by preventing shifts by one place, two places,…etc between identical units. We should note that it is sufficient to add

these constraints only to the ULP since LLP is solved for the subsets of assignments as predicted by ULP.

i) Case of one unit / non-identical units:

Symmetry is not an issue for the case of a single unit or multiple non-identical units.

ii) Case of two identical units:

Since the number of identical units is two, there are two possible permutations. Each unit can only be shifted by one place to yield the only alternative solution (see Figure 23).

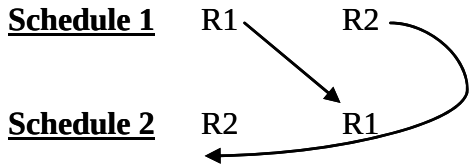


Figure 23. Shifts by one place for 2 identical units

If schedule 1 is obtained as the optimal solution of UPL at iteration *r*, the following constraint will exclude schedule 2 from the search space of UPL at iterations *s*>*r*.

$$\sum_{(i,t)\in Z_m^r} YP_{i,m',t} + \sum_{(i,t)\in Z_{m'}^r} YP_{i,m,t} \le \left|Z_m^r\right| + \left|Z_{m'}^r\right| - 1 \quad \forall m, m', m < m' \tag{61}$$

where $Z_m^r = \left\{i,t,m \,\middle|\, YP_{i,m,t} = 1\right\}$, $Z_{m'}^r = \left\{i,t,m' \,\middle|\, YP_{i,m',t} = 1\right\}$ and *m* and *m'* are identical units. Note that Z_m^r and $Z_{m'}^r$ are obtained from the optimal solution at the upper level, in terms of the assignment variable in iteration *r*. Constraint (61) will eliminate all symmetric solutions that can be constructed by swapping the assignments between two identical units.

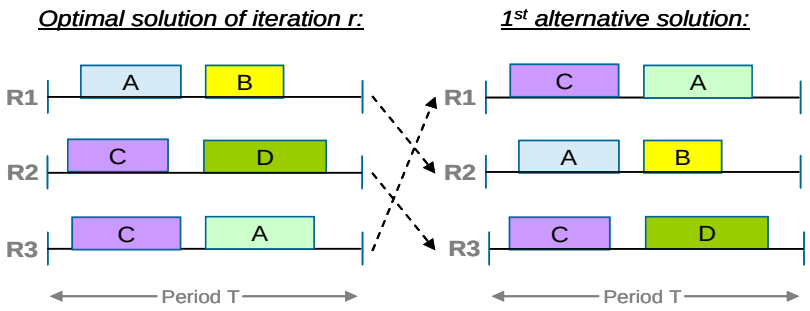
iii) Case of three identical units:

Presence of three identical units results in five additional alternative solutions. From the permutation generation algorithm proposed in the previous section, we know that the first two alternative solutions will be constructed from the original solution by shifting the elements to the right by one place and two places. The next two constraints exclude these two alternatives from further consideration.

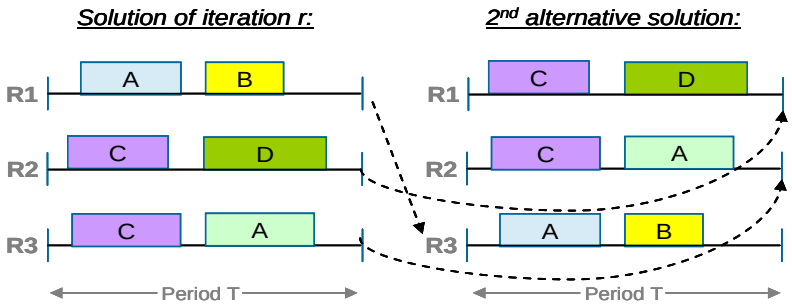
$$\sum_{(i,t)\in Z_m^r} YP_{i,m',t} + \sum_{(i,t)\in Z_{m'}^r} YP_{i,m",t} + \sum_{(i,t)\in Z_{m"}^r} YP_{i,m,t} + \le \left|Z_m^r\right| + \left|Z_{m'}^r\right| + \left|Z_{m"}^r\right| - 1 \quad \forall m, m', m", m < m' < m" \tag{62}$$

$$\sum_{(i,t)\in Z_m^r} YP_{i,m",t} + \sum_{(i,t)\in Z_{m'}^r} YP_{i,m,t} + \sum_{(i,t)\in Z_{m"}^r} YP_{i,m',t} + \le \left|Z_m^r\right| + \left|Z_{m'}^r\right| + \left|Z_{m"}^r\right| - 1 \quad \forall m, m', m", m < m' < m" \tag{63}$$

Constraint (62) will eliminate all the symmetric solutions that can be constructed by shifting the units by one place among three identical units by making such configurations infeasible. Similarly, constraint (63) will eliminate all symmetric solutions that can be constructed by shifting the units by two places among three identical units.



$$Cut1: YP_{A,R2,T} + YP_{B,R2,T} + YP_{C,R3,T} + YP_{D,R3,T} + YP_{C,R1,T} + YP_{A,R1,T} \leq 6-1$$

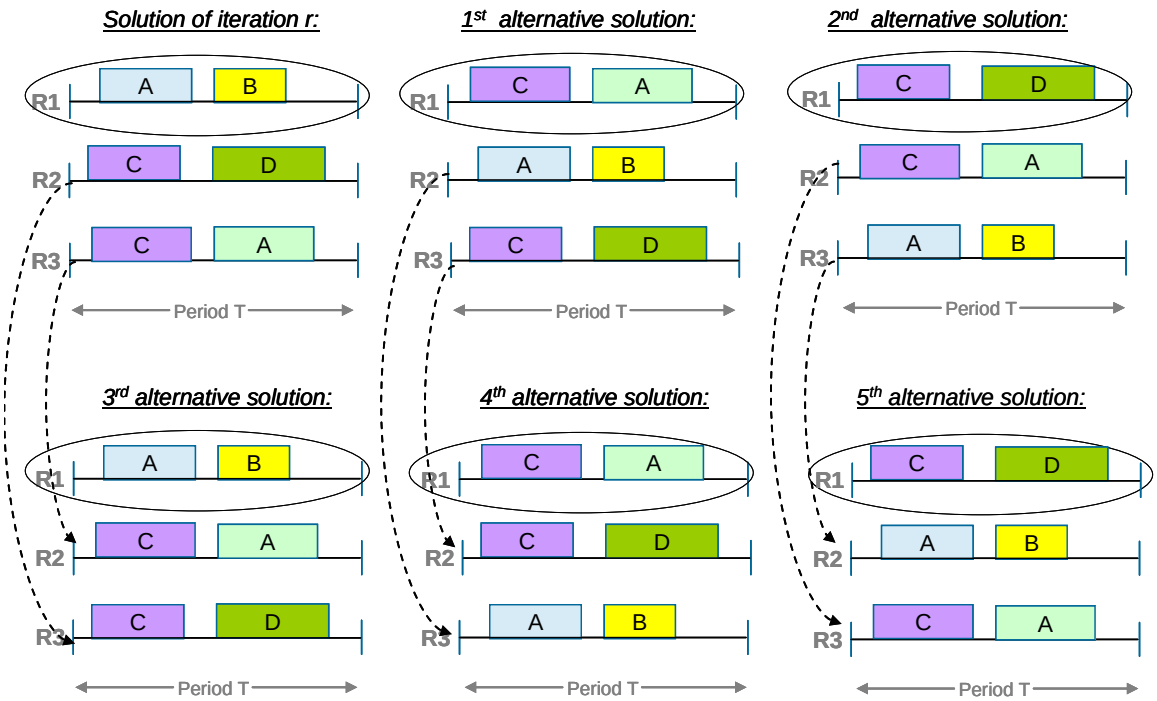


$$Cut2: YP_{A,R3,T} + YP_{B,R3,T} + YP_{C,R1,T} + YP_{D,R1,T} + YP_{C,R2,T} + YP_{A,R2,T} \leq 6-1$$

Figure 24. Construction of the first two alternative solutions for three identical units

This is illustrated in Figure 24 for three identical units and four products. Given the optimal solution at iteration *r*, cut1 (constraint 62) will exclude the first alternative solution which is constructed by shifting the units by one place among three identical units. Note that, in the presence of cut1, first alternative solution becomes an infeasible choice. Likewise, for the same solution, addition of cut2 (constraint 63) will exclude the second alternative solution which is constructed by shifting the units by two places. Cut2 achieves this by imposing infeasibility on the second alternative solution.

Recall from the permutation generation algorithm that the next three solutions (3rd, 4th and 5th alternative solutions) will be generated from the previous three solutions (optimal solution, 1st and 2nd alternative solutions) by keeping the first units in place and reapplying the algorithm. As can be seen from Figure 25, 3rd, 4th and 5th alternative solutions are constructed from the optimal solution, 2nd and 3rd alternative solutions, respectively, simply by performing interchanges between two identical units. However, the interchanges between two identical units are already covered by constraint (61). Therefore, in order to eliminate these three solutions constraint (61) will be sufficient.



$$Cut3: YP_{C,R3,T} + YP_{D,R3,T} + YP_{C,R2,T} + YP_{A,R2,T} \le 3 \quad Excludes\ 3^{rd}\ alternative\ solution$$

$$Cut4: YP_{A,R3,T} + YP_{B,R3,T} + YP_{C,R1,T} + YP_{A,R1,T} \le 3 \quad Excludes\ 4^{th}\ alternative\ solution$$

$$Cut5: YP_{A,R2,T} + YP_{B,R2,T} + YP_{C,R1,T} + YP_{D,R1,T} \le 3 \quad Excludes\ 5^{th}\ alternative\ solution$$

Figure 25. Generation of all alternative solutions for three identical units

Extension of the derivation of the symmetry breaking constraints to the case of four identical units and five identical units is given in Appendix B. To generalize, to eliminate symmetry when there are *M* identical units present, we will introduce (*M-1*) constraints that eliminate one place, two places,….., (*M-1*) place shifts among the *M* units plus all the constraints required to eliminate the symmetric solutions when (*M-1*) identical units are present.

We should note that having multiple time periods does not have any effect on the total number of alternative solutions. This is due to the fact that adjacent time periods are interdependent by the sequence dependent changeovers. As can be seen from Figure 26a, the solutions given in iteration 1 and 2 are symmetric since the units changed places as a row. However, the solutions given in Figure 26b are asymmetric due to the fact that the sequence of products is now different. Had the problem have sequence independent changeovers between adjacent periods then we would have $(M!)^T$alternative solutions instead of *(M!)* where M is the number of identical units and T is the number of time periods.

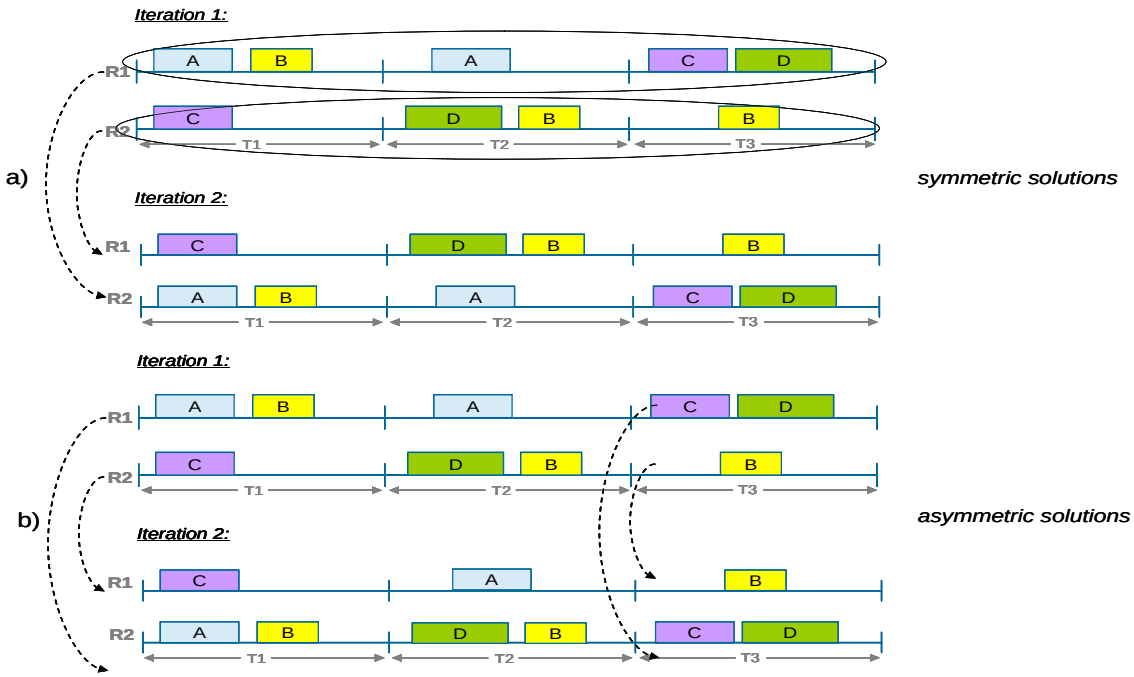


Figure 26. Symmetric and asymmetric solutions due to changeovers across adjacent weeks

9. Results:

In this section we present five different examples, with the last example corresponding to an industrial-sized problem. It should be noted that all the models presented in this paper have been implemented in GAMS 22.3 and solved with CPLEX 10.1 on an 2X Intel Xeon 5150 at 2.66 GHz machine. The data for Examples 1, 2, 3 and 4 are given in Appendix A and the data for all the other examples are available upon request.

9.1. Example 1: Motivating Example Revisited

We apply the proposed decomposition algorithm to both one week and two weeks instances from the motivating example to compare the computational performance with the full-space solution. Table 4 shows the problem sizes, solution times and the objective values for both the proposed approach and the full-space solution for 1 week time horizon, whereas Table 5 shows to the corresponding results obtained for 2 weeks of time horizon. The first row corresponds to solutions obtained by the full-space method whereas the second row summarizes the results obtained by the proposed algorithm. Rows three and four show in detail the problem statistics for the upper level and lower level sub-problems, respectively. For the cases where the algorithm requires more than one iteration to converge, the sizes reported for the upper level and lower level sub-problems are from the last iteration whereas the corresponding times are the total times.

As shown in Table 4, for the 1 week instance the proposed algorithm found the optimal solution of $302,200 in less than a second while the full-space method obtained the same solution in 78 CPUs. The dramatic difference between the sizes of the full space model and the lower level model, LLP is due to difference in the number of slots. While for the full space method 16 slots are postulated for each unit and time period, only 4 slots are postulated for each unit and each time period for LLP. We should also note that the proposed algorithm converged in one single iteration and that the solutions obtained by the upper level and the lower level were identical. The last row of Table 4 illustrates the results obtained

with RTN model with a discretization of 2 hours (Castro et al., 2003). In the specified time limit of 10,000 s, RTN failed to prove the optimality of the solution.

Table 4. Problem Sizes and Results for Example 1 for 1 Week

method	number of binary variables	number of continuous variables	number of equations	time (CPUs)	Solution ($)
full space	1,326	5,355	5,111	77.9	302,200
proposed algorithm				0.624	302,200
ULP	140	207	210	0.359	302,200
LLP	50	483	303	0.265	302,200
RTN-d=2 hr	3,168	5,447	2,112	10,000*	302,200

*Search terminated, best feasible solution posted

Table 5. Problem Sizes and Results for Example 1 for 2 Weeks

method	number of binary variables	number of continuous variables	number of equations	time (CPUs)	Solution ($)
full space	2,652	10,967	11,088	10000*	933,255
proposed algorithm				6.155	933,436
ULP	280	419	435	4.593	933,436
LLP	216	1,723	1,297	1.562	933,436

While for the 1 week instance, the proposed algorithm shows significantly improved solution times, the computational benefit of the algorithm is demonstrated better for the 2 week instance. The proposed algorithm yields a solution of $933,436 in only 6 CPUs in one single iteration where as the full-space method fails to terminate within the specified limit (10,000 CPUs), only managing to produce a feasible solution of $933,255. This is more than three orders of magnitude reduction in the solution time.

This significant reduction in the solution time is mainly due to the fact that the original problem is decomposed into smaller sub-problems that are easier to solve. Having time balances instead of detailed timing constraints, and aggregate mass balances instead of slot based explicit mass balances, the ULP is much smaller in size. The LLP not only has the advantage of having to handle fewer products at each time period compared to the original model, but it is also solved for fewer slots which in turn reduces the number of constraints, continuous and binary variables. Moreover, the total number of iterations required for obtaining convergence is reduced by the ULP's ability to explicitly taking into account the sequence-dependent changeover times and costs.

9.2. Example 2:

This is essentially the same as Example 1 for the 1 week horizon instance, except that the demand rates are increased as shown in Appendix A. As expected, this results in a larger LLP compared to that of Example 1 due to the fact that the number of slots to be postulated for the lower level is larger now.

As seen in Table 6, the proposed algorithm yields a profit of $1,070,660 requiring 2 major iterations in only 14 CPUs, whereas the full-space method failed to terminate in 20,000 CPUs obtaining a feasible solution of $1,028,280. The problem sizes reported in the third and fourth rows are from the second iteration, while the corresponding times are the total times. The last row of table 6 shows the

results obtained with RTN model using discretizations of 2 hours (Castro et al., 2003). A feasible solution of $ 733,580 was obtained in 20,000 seconds.

Table 6. Problem Sizes and Results for Example 2

method	number of binary variables	number of continuous variables	number of equations	time (CPUs)	Solution ($)
full space	1326	5355	5276	20,000*	1,028,280
proposed algorithm				14.2	1,070,660
ULP	140	207	212	2.23	1,070,660
LLP	456	2,415	2,080	11.97	1,070,660
RTN-d=2 hr	3,168	5,447	2,112	20,000*	733,580

*Search terminated, best feasible solution posted

Table 7 shows the progress of iterations with the proposed method. Note that the upper level and the lower level produced identical solutions at the second iteration.

Table 7. Progress of Iterations for Example 2

Iteration	Upper Bound (UB)	Lower Bound (LB)
1	1,072,580	1,028,320
2	1,070,660	1,070,660*

*Optimal solution

Figure 27 shows the Gantt chart and the flow to and from the intermediate storage tanks. As can be seen, 10 slots are being utilized in R1 and 9 slots are being utilized in R2. 38,400 lbs of product A is transferred to the intermediate storage tank from the 5th slot in R2 to supply for the raw material requirements of Product B assigned to slots 5 and 6 in unit R1. Note that the end time of slot 5 in R1 (104 hrs) is less than the start time of slots 5 (107) and 6 (117) of R1. Similarly, 57,600 lbs of product A is transferred from slot 6 of unit R2 to supply for product B assigned to slots 7, 8 and 9 in unit R1. Finally, 19,200 lbs of A is transferred from the 8th slot of unit R2 to the intermediate storage tank which is then transferred to slot 10 of unit R1.

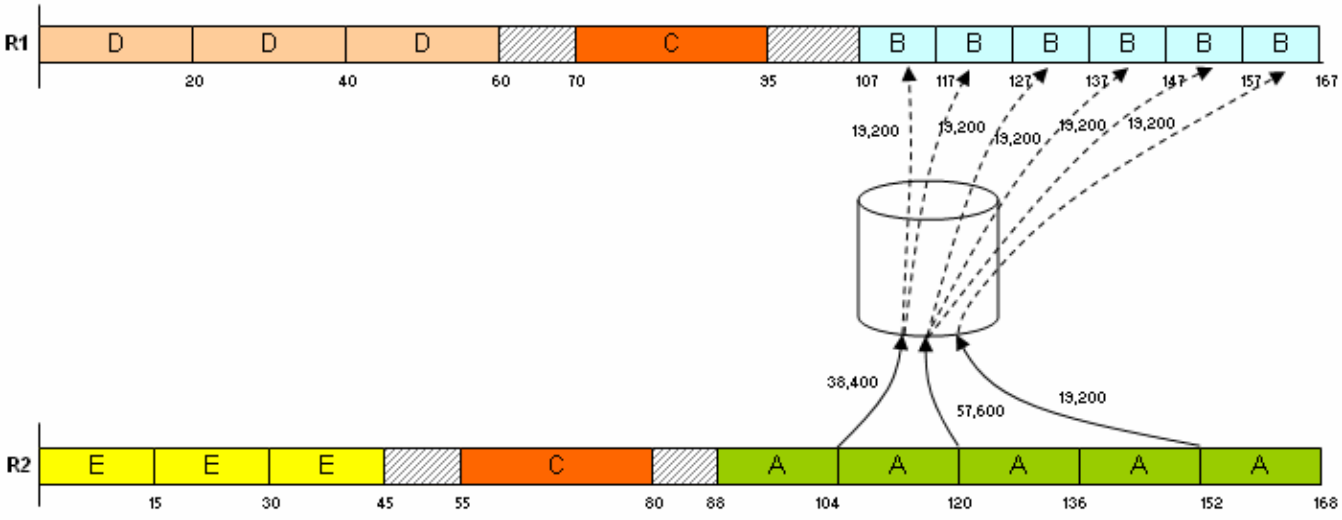


Figure 27. Gantt chart and flow diagram for Example 2

9.3. Example 3:

This example is an extension of Example 2 to a time horizon of 2 weeks. Table 8 shows the comparison of the problem sizes and the corresponding solutions of the proposed algorithm with the full-space solution. While it took the proposed approach only 22.22 CPUs to find the optimal solution of $2,223,654, the full-space solution only managed to find a feasible solution of $2,007,969 in 20,000 CPUs.

Table 8. Problem Sizes and Results for Example 3

method	number of binary variables	number of continuous variables	number of equations	time (CPUs)	Solution ($)
full space	2,652	10,967	11,418	20,000*	2,007,969
proposed algorithm				26.22	2,223,654
ULP	280	419	435	11.52	2,223,759
LLP	956	5,140	4,810	14.70	2,223,654

*Search terminated, best feasible solution is posted.

As shown in the last column of Table 8, the solutions obtained by the upper level ($2,223,759) and the lower level ($2,223,654) are slightly different despite the fact that the same optimal schedules are generated by the ULP and the LLP models (see Figure 28 and 29). From Table 9, it is clear that this is due to the difference in the inventory costs; the inventory cost determined by the upper level is $105.3 less than the value determined by the lower level. This underestimation of the inventory cost by the upper level results from the fact that mass balances are handled in an aggregate manner, and also that ULP does not take into account the fact that either the intermediate should be produced before the production of the corresponding end product starts within the same time period or there should be sufficient material in the intermediate storage tank.

Table 9. Comparison of the Objective Function Items for the Upper and Lower Level Solutions

($)	ULP	LLP
sales	3,606,100	3,606,100
operating costs	1,381,480	1,381,480
inventory costs	220.9	326.2
transition costs	640	640

Note that in the optimal schedules obtained by both ULP and LLP, in week 2 of unit R2 product B is assigned before product A. In order to ensure feasible mass balances, the LLP model ensures that there is sufficient product A in the intermediate storage tank at the end of the first week to supply for product B that is assigned in week 2, whereas since in the ULP model the mass balances are aggregated over time periods, it underestimates the inventory.

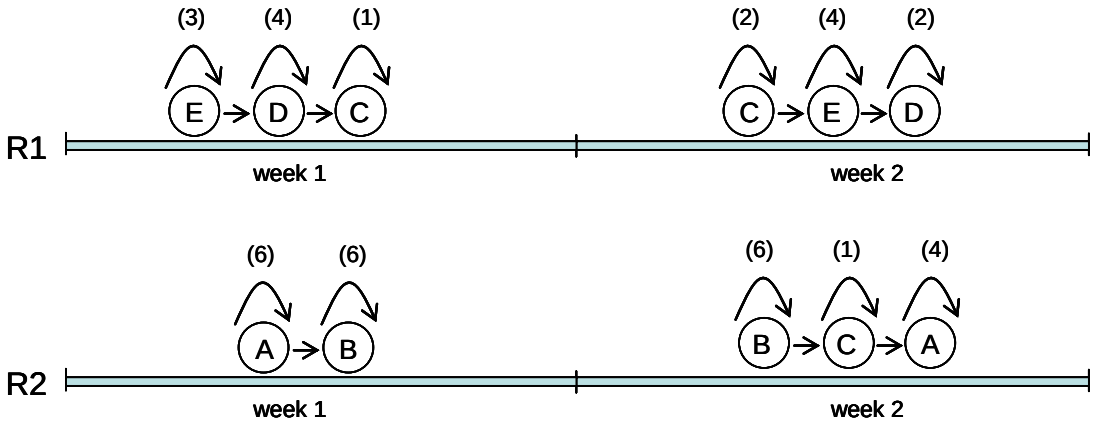


Figure 28. Optimal schedule generated by the ULP for Example 3

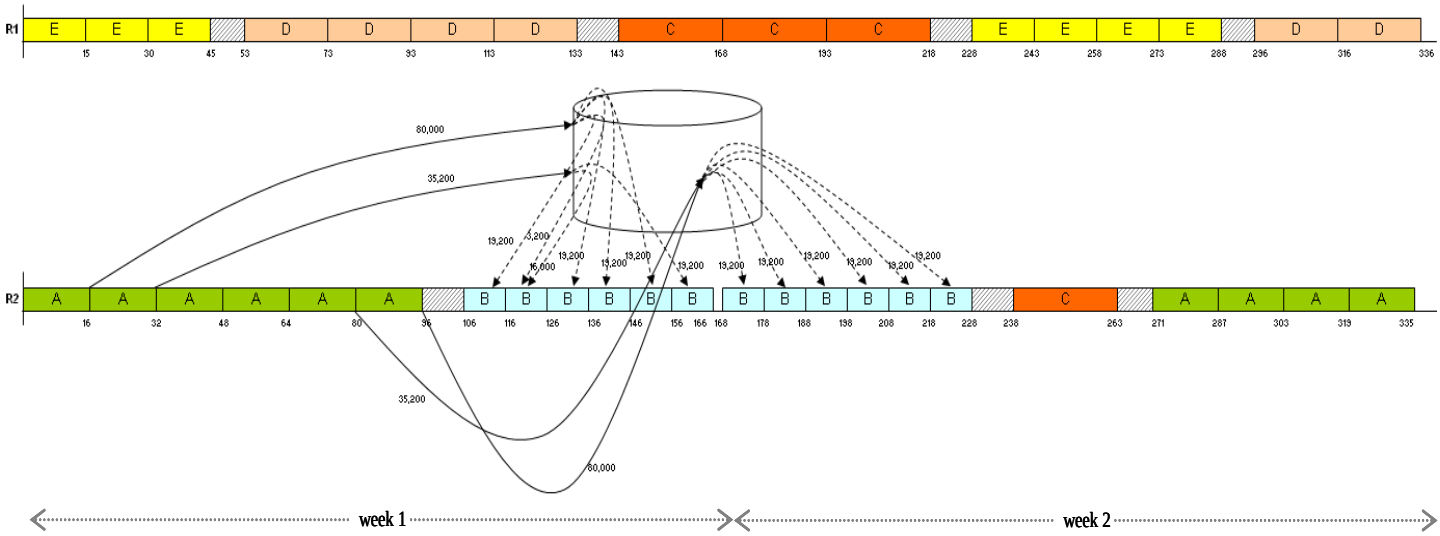


Figure 29. Gantt chart and flow diagram generated by the LLP for Example 3

9.4. Example 4:

This example consists of six different products, A-F and four identical reactors, R1-R4 with a horizon time of 1 week. Each reactor can process any of the products. Products B and E are produced in two stages and require intermediates A and F, respectively, whereas the other products are produced in a single production stage. The schematic representation of the problem is given in Figure 30.

Table 10 shows the comparison of the problem sizes and solution times of the proposed method with the full-space method. The proposed method obtained the profit of $1,448,542.98 in 464 CPUs in 4 major iterations. The full-space method failed to terminate within the specified time limit (30,000 CPUs), yielding a feasible solution of $1,287,315.29.

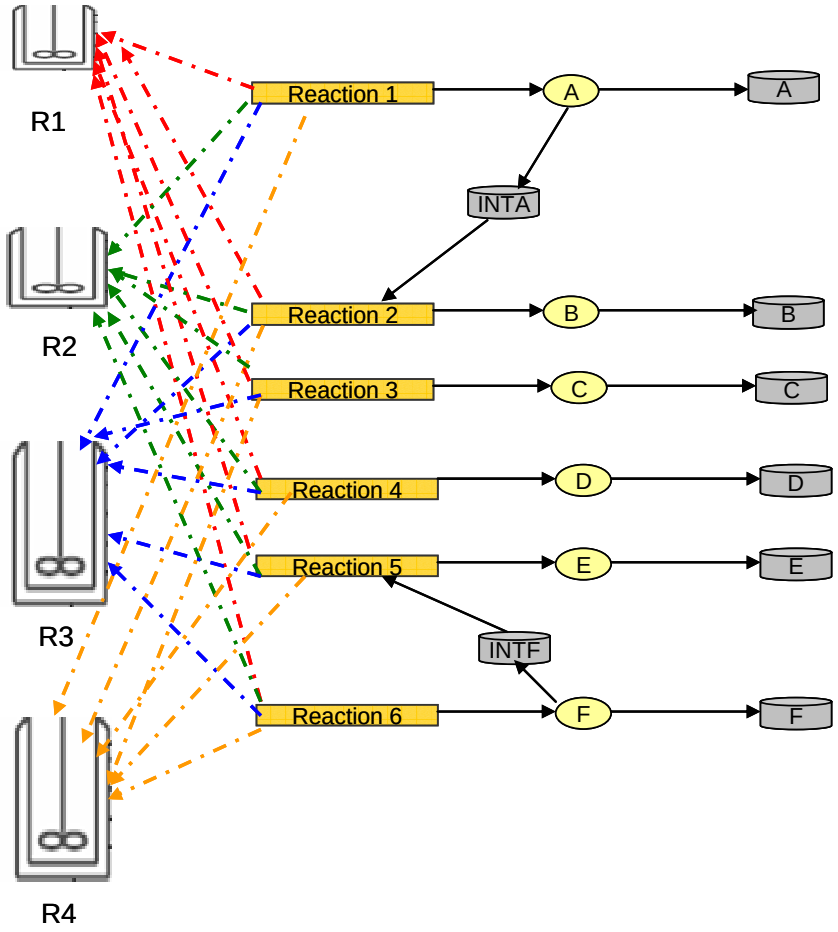


Figure 30. Schematic representation for Example 4

Table 10. Problem Sizes and Results for Example 4

method	number of binary variables	number of continuous variables	number of equations	time (CPUs)	Solution ($)
full space	5,032	22,417	30,231	30,000*	1,287,315.29
proposed algorithm				464.48	1,448,542.98
ULP	384	549	631	366.95	1,448,542.98
LLP	1,435	8,029	9,151	97.53	1,448,542.98

Table 11 shows the progress of the iterations for Example 4. The optimal solution is obtained at the 4th iteration and the upper and lower bounds are identical. We should note that in the absence of the symmetry breaking cuts, the optimal solution of $1,448,542.98 is not obtained even after 50 major iterations. In this case, the proposed approach yields the solution of $1,399,010.32 for the first 24 iterations, then the solution of $1,424,400.21 for the next 24 iterations, and finally the solution of $1,448,250.21 for the last 2 iterations. The Gantt chart and the flow diagram for Example 4 is illustrated in Figure 31.

Table 11. Progress of Iterations for Example 4

Iteration	Upper Bound (UB)	Lower Bound (LB)
1	1,448,632.98	1,399,010.32
2	1,448,612.98	1,424,400.21
3	1,448,552.98	1,424,250.21
4	1,448,542.98	1,448,542.98*

*Optimal solution

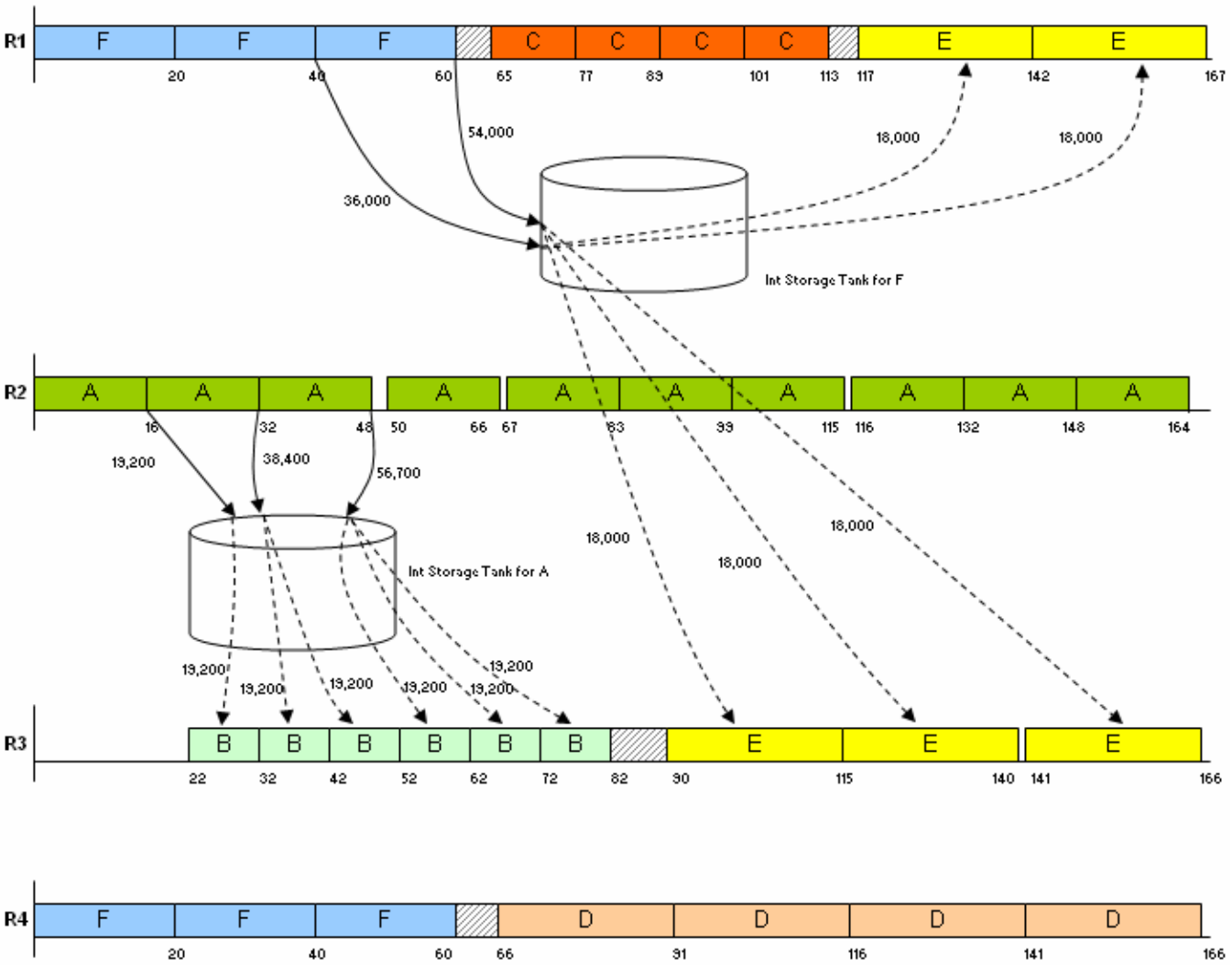


Figure 31. Gantt chart and flow diagram generated by the LLP for Example 4

9.5. Example 5:

In this example we consider four products, A-D and three identical reactors, R1-R3 for a time horizon of two weeks. Each reactor can process any of the products. Product B is produced in two stages and require intermediate A with a ratio of 10 to 1. While in the previous examples the due dates were defined only at the end of each time periods, this example includes multiple intermediate due dates. Specifically, there are six due dates where the products are required to be shipped as can be seen from Figure 32.

Table 12. Problem Sizes and Results for Example 5

method	number of binary variables	number of continuous variables	number of equations	time (CPUs)	Solution ($)
full space	23,040	67,469	92,289	10,000*	**
proposed algorithm				55.50	6,767,952
ULP	864	1283	1463	52.80	6,767,952
LLP	2884	12,759	14,326	2.70	6,767,952

*Search terminated, best feasible solution is posted.

** No feasible solution found in the given time limit

Table 12 shows the comparison of problem sizes and solution times of the proposed method with the full space method. While the proposed method found the optimal solution of $6,767,952 in 55.5 CPUs in a single iteration, the full space method failed to find a feasible solution within the specified time limit of 10,000s. We should note that, while the proposed method can handle the issue if intermediate due dates efficiently in terms of computational effort, the slots are restricted to associated time periods and they cannot spill over the time periods. Therefore, suboptimal solutions can be obtained. Figure 32 shows the Gantt chart and the flow diagram. The two weeks of time horizon is divided into six time periods as defined by the due dates which are specified at times 70 hours, 110 hours, 158 hours, 200 hours, 260 hours and 336 hours.

10. Conclusions:

A new continuous-time MILP formulation based on asynchronous slots has been presented for the short-term scheduling of multi-product batch plants with parallel units. The proposed formulation incorporates several realistic features such as limited, shared intermediate storage, batch splitting, intermediate due dates and sequence-dependent changeover times and costs. Due to the combinatorial nature of the problem, the solution times increase significantly with the problem size. In order to be able to efficiently solve larger problems of practical interest, we have proposed a bi-level decomposition scheme based on the work of Erdirik-Dogan and Grossmann (2006) that guarantees optimality. We have also proposed novel symmetry breaking cuts that exploit the inherent symmetry of the problem arising from parallel identical units. The application of the algorithm was illustrated with several examples where it can be seen that compared to the full space method, the proposed approach decreased the solution times by more than an order of magnitude. The proposed decomposition approach has proved to be very efficient requiring only a few iterations between the master and the subproblems. This achievement is mainly due to incorporating the effects of changeovers at the upper level which leads to very accurate predictions and tight upper bounds thus decrease in the number of iterations.

Acknowledgments. The authors thank John Wassick from the Dow Chemical co. for defining the problem in this paper. Also, special thanks to Pedro Castro from INETI for comparing our method with the discrete RTN model. The authors would also like to acknowledge financial support from the Pennsylvania Infrastructure Technology Alliance, Institute of Complex Engineered Systems, from the National Science Foundation under Grant No. DMI-0556090 and from the Dow Chemical Company.

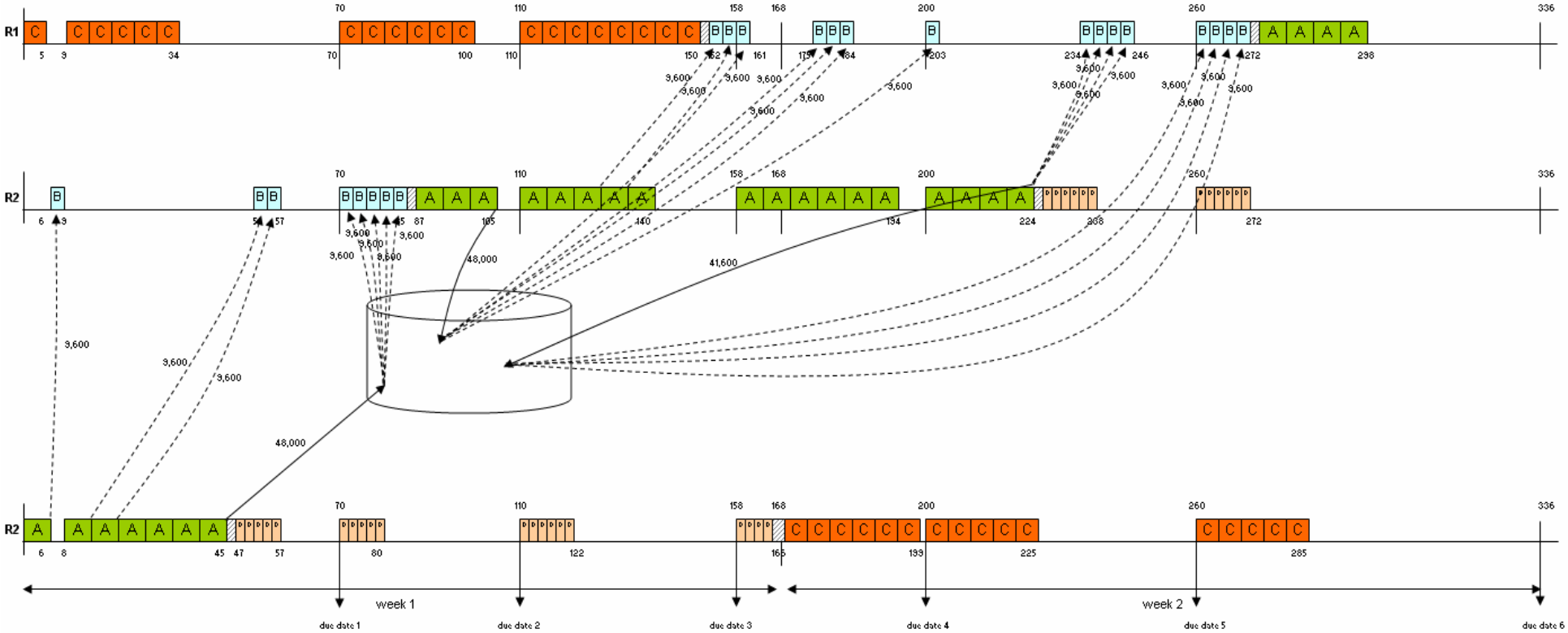


Figure 32. Gantt chart and flow diagram generated by the LLP for Example 5

Appendix A
Data for Example 1:

Table A1. Batch Sizes and Batch Times for Example 1

	Batch Size (lb)	
	R1	R2
A	80,000	80,000
B	96,000	96,000
C	120,000	120,000
D	100,000	100,000
E	150,000	150,000
	Batch Time (hrs)	
	R1	R2
A	16	16
B	10	10
C	25	25
D	20	20
E	15	15

Table A2. Selling Price and Cost Data for Example 1

Product	operating costs ($/lb)	selling price ($/lb)	inventory costs ($/lb w)
A	0.35	0.95	0.001496
B	0.34	0.99	0.001339
C	0.36	0.9	0.001418
D	0.37	1	0.001539
E	0.3	0.85	0.001618

Table A3. Changeover Times and Costs for Example 1

Product	A	B	C	D	E
			Transition times (hrs)		
A	0	10	16	12	12
B	20	0	10	15	12
C	8	12	0	20	10
D	12	22	10	0	25
E	30	15	10	8	0
			Transition costs ($/1000)		
A	0	100	160	120	120
B	200	0	100	150	120
C	80	120	0	200	100
D	120	220	100	0	250
E	300	150	100	80	0

Table A4. Lower Bounds for Demands for 1 Week Horizon Example 1

	Demand (lb/w)
Product	time period 1
A	80,000
B	96,000
C	120,000
D	100,000
E	150,000

Table A5. Lower Bounds for Demands for 2 Weeks Horizon Example 1

	Demand (lb/w)	
Product	time period	time period 2
A	80,000	160,000
B	96,000	192,000
C	120,000	240,000
D	100,000	200,000
E	150,000	300,000

Data for Example 2:

Table A6. Lower Bounds for Demands for 1 Week Horizon Example 2

Demand (lb/w)	
Product	time period 1
A	320,000
B	576,000
C	360,000
D	300,000
E	450,000

Data for Example 3:

Table A7. Lower Bounds for Demands for 2 Weeks Horizon Example 3

	Demand (lb/w)	
Product	time period 1	time period 2
A	320,000	320,000
B	576,000	576,000
C	360,000	480,000
D	300,000	300,000
E	450,000	600,000

Data for Example 4:

Table A8. Batch Sizes and Batch Times for Example 4

	Batch Size (lb)			
	R1	R2	R3	R4
A	80,000	80,000	80,000	80,000
B	96,000	96,000	96,000	96,000
C	84,000	84,000	84,000	84,000
D	125,000	125,000	125,000	125,000
E	90,000	90,000	90,000	90,000
F	80,000	80,000	80,000	80,000
	Batch Time (hrs)			
	R1	R2	R3	R4
A	16	16	16	16
B	10	10	10	10
C	12	12	12	12
D	25	25	25	25
E	25	25	25	25
F	20	20	20	20

Table A9. Selling Price and Cost Data for Example 4

Product	operating costs ($/lb)	selling price ($/lb)	inventory costs ($/lb w)
A	0.35	0.95	0.001496
B	0.34	0.99	0.001339
C	0.36	0.9	0.004018
D	0.36	0.8	0.0025
E	0.32	0.7	0.001
F	0.45	1	0.001596

Table A10. Changeover Lower Bounds for Demands for Example 4

Demand (lb/w)	
Product	time period 1
A	640,000
B	576,000
C	336,000
D	500,000
E	450,000
F	400,000

Table A11. Minimum Number of Batches for Example 4

Product	Min. number of batches
A	3
B	2
C	2
D	2
E	2
F	2

Table A12. Changeover Times and Costs for Example 4

Product	A	B	C	D	E	F
Transition times (hrs)						
A	0	10	16	20	10	30
B	20	0	10	5	8	18
C	8	12	0	3	4	20
D	11	14	16	0	12	22
E	4	23	19	3	0	8
F	7	5	5	6	8	0
Transition costs ($/1000)						
A	0	100	160	200	100	300
B	200	0	100	50	80	180
C	80	120	0	30	40	200
D	110	140	160	0	120	220
E	40	230	190	30	0	80
F	70	50	50	60	80	0

Appendix B

The flowchart of the permutation generation algorithm is given in Figure B1. The cyclic-right rotation operation is defined as follows:

$$rotate[1](i):\{k \to k+1, k+1 \to k+2, ..., k+m \to k\}$$
$$rotate[2](i):\{k \to k+2, k+1 \to k+3, ..., k+m \to k+1\}$$
$$rotate[3](i):\{k \to k+3, k+1 \to k+4, ..., k+m \to k+2\}$$
$$\vdots$$
$$rotate[m](i):\{k \to k+m, k+1 \to k, ..., k+m \to k+m-1\}$$

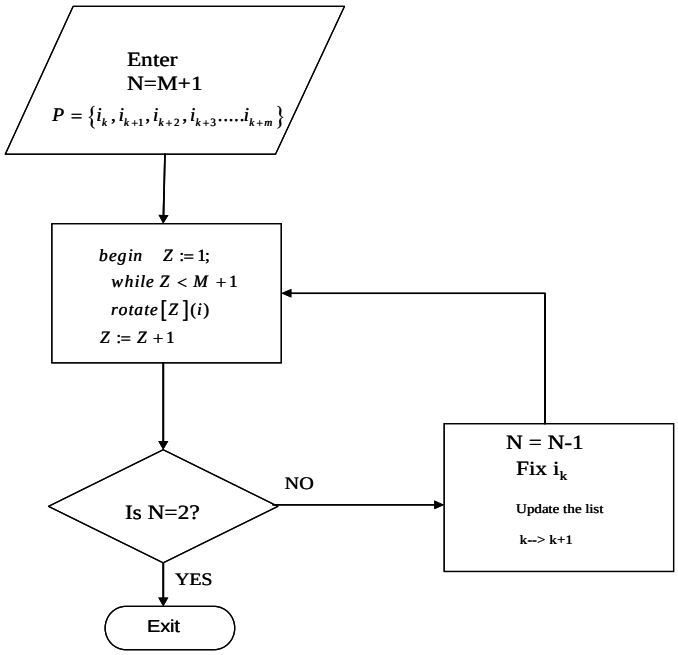


Figure B1. Flowchart of permutation generation algorithm

Derivation of the symmetry breaking cuts for the case of four identical units:

iv) Case of four identical units:

Generation of all the alternative solutions in the case of four identical units is performed by:

1) right shifting by one place, two places and three places among four identical units
2) fixing the leftmost element in all the three solutions obtained in step 1 and the original solution and shifting the units by one place and two places among three identical units
3) fixing the leftmost two positions in the solutions obtained at step 2 and shifting the units by one place among two identical units.

In order to eliminate the alternative solutions obtained at step 1, we introduce three additional constraints which are derived in analogy to constraints (62) and (63).

$$\sum_{(i,t)\in Z^r_m} YP_{i,m',t} + \sum_{(i,t)\in Z^r_{m'}} YP_{i,m'',t} + \sum_{(i,t)\in Z^r_{m''}} YP_{i,m''',t} + \sum_{(i,t)\in Z^r_{m'''}} YP_{i,m,t} \le \left|Z^r_m\right| + \left|Z^r_{m'}\right| + \left|Z^r_{m''}\right| + \left|Z^r_{m'''}\right| - 1 \quad \text{(B1)}$$
$$\forall m, m', m'', m''', m < m' < m'' < m'''$$

Constraint (B1) eliminates the symmetric solutions that are constructed by one place shifts between four identical units.

$$\sum_{(i,t)\in Z^r_m} YP_{i,m'',t} + \sum_{(i,t)\in Z^r_{m'}} YP_{i,m''',t} + \sum_{(i,t)\in Z^r_{m''}} YP_{i,m,t} + \sum_{(i,t)\in Z^r_{m'''}} YP_{i,m',t} \le \left|Z^r_m\right| + \left|Z^r_{m'}\right| + \left|Z^r_{m''}\right| + \left|Z^r_{m'''}\right| - 1 \quad \text{(B2)}$$
$$\forall m, m', m'', m''', m < m' < m'' < m'''$$

Constraint (B2) eliminates the symmetric solutions that are constructed by two place shifts between four identical units.

$$\sum_{(i,t)\in Z^r_m} YP_{i,m''',t} + \sum_{(i,t)\in Z^r_{m'}} YP_{i,m,t} + \sum_{(i,t)\in Z^r_{m''}} YP_{i,m',t} + \sum_{(i,t)\in Z^r_{m'''}} YP_{i,m'',t} \le \left|Z^r_m\right| + \left|Z^r_{m'}\right| + \left|Z^r_{m''}\right| + \left|Z^r_{m'''}\right| - 1 \quad \text{(B3)}$$

$$\forall m, m', m'', m''', m < m' < m'' < m'''$$

Constraint (B3) eliminates the symmetric solutions that are constructed by three place shifts between four identical units.

For the elimination of the alternative solutions generated by step 2, the problem reduces to the case of three identical units and thus can be handled with constraints (62) and (63). Likewise for exclusion the symmetries generated by step 3, the problem reduces to the case of two identical units and therefore adding constraint (61) will be sufficient to eliminate these solutions. To summarize in order to avoid all the symmetric solutions for the case of four identical units, we have introduced three constraints (B1, B2 and B3) in addition to all the constraints that have been introduced for the three identical units case (61, 62 and 63).

For the case of five identical units, we will need to add four additional constraints (shifts by one place, two places, three places and four places among five identical units) plus all the constraints that have been added for the four identical units case (61, 62, 63,B1, B2, and B3).

Nomenclature

Indices

i, i' products

l, l',l'',l''' slots

m,m',m'',m''' units

t time periods

$\overline{l}_m$ last slot of unit *m*

$\overline{t}$ last time period

Sets

I set of products

I_m set of products that can be processed on unit *m*

$IFINT_i$ set of end products that are also intermediate products

IE_i set of end products that are produced in two stages

IF_i set of end products that are produced in a single stage

L set of slots

L_m set of slots that belong to unit m

T set of time periods

Parameters

$NS_{m,t}$ number of slots postulated for each unit, each time period

NS_m number of slots postulated for each unit

$BT_{i,m}$ batch processing time of task i in unit m

$Q_{i,m}$ batch size of task i in unit m

$\alpha_{i,i'}$ mass balance coefficient for the production of product i from intermediate i'

$D_{i,t}$ demand for product i at the end of time period t

MRL_i minimum number of batches for each product

CAP_i^{INT} capacity of the intermediate storage tank for the intermediate product *i*

$\tau_{i,i'}^m$ changeover time required to change the operation from product i to product i' in unit m

$TRBound_m$ maximum transition time for unit *m*

$TRCBound_m$ maximum transition cost for unit m

H_t duration of the t^{th} time period

HT_t time, in terms of hours, at the end of the t^{th} time period

$C_{i,t}^{OP}$ operating cost of product i at the end of time period t

$C_{i,t}^{INV}$ inventory cost of product i at the end of time period t

$C_{i,t}^P$ selling price of product i at the end of time period t

$C\tau_{i,i'}^m$ changeover costs of changing the production from task i to i' in unit m

$NPslot_{i,m,t}^r$ upper bound on the number of slots for product *i* in unit *m* during time period *t* at iteration *r*

$NUslot_{m,t}^r$ upper bound on the number of slots in unit *m* during time period *t* at iteration *r*

$NTslot_t^r$ upper bound on the number of slots for each time period

Variables

$W_{i,m,l,t}$ 0-1 variable to denote the assignment of product i to slot l of unit m during time period t

$YP_{i,m,t}$ 0-1 variable to denote the assignment of product *i* to unit *m* at each time period *t* (upper level)

$PT_{m,l,t}$ the length of slot *l* of unit *m* in time period *t*

$X_{i,m,l,t}$ the amount of product *i* produced in slot *l* of unit *m* at each time period *t*

NB_{imt} number of batches of each product *i* in each unit *m* at each period *t* (upper level)

FP_{imt} amount of material processed by each task *i* (upper level)

$Z_{i,i',m,l,t}$ to denote if product *i* assigned to slot *l* is followed by product *i'* at slot *l*+1 on unit *m* at time period *t*

$TR_{m,l,t}$ transition time that will be applied after slot *l* of unit *m* during time period *t*

$\hat{Z}_{i,i',m,l,t}$ transition variable responsible from transition across adjacent weeks when all slots are occupied

$\hat{TR}_{m,l,t}^{across}$ transition time that will be applied at the end of period t when all slots are occupied

$\tilde{Z}_{i,i',m,l,t}$ transition variable responsible from transition across adjacent weeks in the presence of unoccupied slots

$\tilde{TR}_{m,l,t}^{across}$ transition time that will be applied at the end of period t in the presence of unoccupied slots

$Y_{m,l,t}$ to denote if slot *l* of unit *m* is occupied during time period *t*

$Te_{m,l,t}$ end time of slot *l* of unit *m* in period *t*

$Ts_{m,l,t}$ start time of slot *l* of unit *m* in period *t*

$TRACOST1$ transition costs within each period

$TRACOST2$ transition costs for changeovers across adjacent periods

$INVP^{FIN}_{i,m,l,t}$ amount of intermediate produced in slot *l* of unit *m* at time period *t* that is transferred to the dedicated final storage tanks

$INVINT^{TRA}_{i,m,l,t}$ amount of intermediate produced in slot *l* of unit *m* at time period *t* that is transferred to the intermediate storage tanks to satisfy the raw material requirements of the corresponding end products

$AA_{i,m,l,m,l',t}$ amount of intermediate transferred from slot *l* of unit *m* at time period *t* to slot *l'* of unit *m* at time *t*

$AA_{i,m,l,m',l',t}$ amount of intermediate transferred from slot *l* of unit *m* at time period *t* to slot *l'* of unit *m'* ($m' \neq m$) at time *t*

$INVP^{INT}_{i,m,l,t}$ intermediate transferred from slot *l* of unit *m* at time period *t* to the intermediate storage tank

$YY_{l,l',m,m',t}$ 0-1 variable to denote if slot *l* of unit *m* is completed before slot *l'* of unit *m'* starts at time period *t*

$INV^{INT}_{i,t}$ inventory level of $i \in IFINT_i$ in the intermediate storage tank at the end of time period *t*

$INVID^{INT}_{i,m,l,t}$ amount of intermediate transferred from the initial inventory in the intermediate storage tank to slot *l* of unit *m* during the first time

$INVC_{i,m,l,t+1}$ amount of intermediate consumed from the intermediate storage tank in order to supply for the end product that is assigned to slot *l* of unit *m* in the subsequent time period

$INV^{FIN}_{i,t}$ inventory level of $i \in IFINT_i$ in the dedicated final storage tank at the end of time period *t*

$S_{i,t}$ sales of product *i* in period *t*

REFERENCES

1.Reklaitis, G.V. (1992). Overview of scheduling and planning of batch process operations. *NATO Advanced Study Institute—Batch process systems engineering*. Turkey: Antalya.

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

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

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

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

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

7.Méndez, C.A.; Cerdá, J.; Grossmann, I.E.; Harjunkoski, I.; Fahl, M. State-of-the-art Review of Optimization Methods for Short-Term Scheduling of Batch Processes. Comput. Chem. Eng. 2006, 30, 913.

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

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

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

11. Pantelides, C.C. (1994). Unified frameworks for optimal process planning and scheduling. *Foundations of Computer-Aided Process Operations,* Cache publications, New York, 253 – 274.

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

13. Zhang, X. & Sargent, W.H. (1996). The optimal operation of mixed production facilities – A general formulation and some approaches for the solution. *Computers and Chemical Engineering*, 20, 897 – 904.
14. Ierapetritou, M.G., & Floudas C.A. (1998) Effective Continuous-Time Formulation for Short-Term Scheduling: I. Multipurpose Batch Processes. *Ind. & Eng. Chem. Res.*, 37, 11, 4341-4359.
14. Mockus, L. & Reklaitis, G.V. (1999b). Continuous time representation approach to batch and continuous process scheduling. 2. Computational issues. *Industrial and Engineering Chemistry Research*, 38, 204 – 210.
15. Giannelos, N.F. & Georgiadis, M.C. (2002). A simple new continuous-time formulation for short-term scheduling of multipurpose batch processes. *Industrial and Engineering Chemistry Research*, 41, 2178 – 2184.
16. Maravelias, C.T. & Grossmann, I.E. (2003). New general continuous-time state-task network formulation for short-term scheduling of multipurpose batch plants. *Industrial and Engineering Chemistry Research*, 42, 3056 – 3074.
17. Pantelides, C.C. (1994). Unified frameworks for optimal process planning and scheduling. *Foundations of Computer-Aided Process Operations*, Cache publications, New York, 253 – 274.
18. Castro, P., Barbosa-Póvoa, A.P.F.D & Matos, H. (2001). An improved RTN continuous-time formulation for the short-term scheduling of multipurpose batch plants. *Industrial and Engineering Chemistry Research*, 40, 2059 – 2068.
19. Castro, P.M.; Barbosa-Póvoa, A.P.; Matos, H.A. & Novais, A.Q. (2004) Simple continuous-time formulation for short-term scheduling of batch and continuous processes. *Industrial and Engineering Chemistry Research*, 43, 105 – 118.
20. Pinto, J.M. & Grossmann, I.E. (1995). A continuous time mixed integer linear programming model for short-term scheduling of multistage batch plants. *Industrial and Engineering Chemistry Research*, 34, 3037 – 3051.
21. Pinto, J.M. & Grossmann, I.E. (1996). An alternate MILP model for short-term scheduling of batch plants with preordering constraints. *Industrial and Engineering Chemistry Research*, 35, 338 – 342.
22. Chen, C.; Liu, C.; Feng, X. & Shao, H. (2002). Optimal short-term scheduling of multiproduct single-stage batch plants with parallel lines. *Industrial and Engineering Chemistry Research*, 41, 1249 – 1260.
23. Lim, M. & Karimi, I.A. (2003). Resource-constrained scheduling of parallel production lines using asynchronous slots. *Industrial and Engineering Chemistry Research*, 42, 6832 – 6842.
24. Sundaramoorthy, A. & Karimi, I.A. (2005). A simpler better slot-based continuous-time formulation for short-term scheduling in multiproduct batch plants. *Chemical Engineering Science*, 60, 2679 – 2702.
25. Cerdá, J.; Henning, G.P. & Grossmann, I.E. (1997). A mixed-integer linear programming model for short term scheduling of single-stage multiproduct batch plants with parallel lines. *Industrial and Engineering Chemistry Research*, 36, 1695 – 1707.
26. Méndez, C.A.; Henning, G.P. & Cerdá, J. (2000). Optimal scheduling of batch plants satisfying multiple product orders with different due-dates. *Computers and Chemical Engineering*, 24, 2223 – 2245.
27. Méndez, C.A.; Henning, G.P. & Cerdá, J. (2001). An MILP continuous-time approach to short-term scheduling of resource-constrained multistage flowshop batch facilities. *Computers and Chemical Engineering*, 25, 701 – 711.
28. Méndez, C.A. & Cerdá, J. (2003a). An MILP continuous-time framework for short-term scheduling of multipurpose batch processes under different operation strategies. *Optimization & Engineering*, 4, 7 – 22.
29. Méndez, C.A. & Cerdá, J. (2004a). An MILP framework for batch reactive scheduling with limited discrete resources. *Computers and Chemical Engineering*, 28, 1059 – 1068.
30. Méndez, C.A. & Cerdá, J. (2004b). Short-term scheduling of multistage batch processes subject to limited finite resources. *Computer-Aided Chemical Engineering*, 15B, 984-989.
31. Gupta, S. & Karimi, I.A. (2003). An improved MILP formulation for scheduling multiproduct, Multistage Batch Plants. *Industrial and Engineering Chemistry Research*, 42, 2365 – 2380.

32. Erdirik-Dogan, M. & Grossmann, I.E. (2007) Optimal Production Planning Models for Parallel Batch Reactors with Sequence-dependent Changeovers. *To appear in AIChE Journal.*
33. Sedgewick, R. (1977) Permutation Generation Methods. *Computing Surveys*, 9, 2, 137-164.
34. Langdon, G. G. (1967) An algorithm for generating permutations. *Commun. ACM* 10, 5, 298-299.
35.Gao, J. & Wang, D. (2003) Permutation Generation: Two New Permutation Algorithms. *eprint arXiv:cs/0306025*.
36. Erdirik-Dogan, M. & Grossmann, I.E. (2006) Simultaneous Planning and Scheduling for Multiproduct Continuous Plants. *Ind. Eng. Chem. Res*, 45. 299-315.
37. Castro, P.M.; Novais, A.Q. & Henrique A.M. (2003) 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