Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers and Chemical Engineering 27 (2003) 1247–1259
Computers & Chemical Engineering
www.elsevier.com/locate/compchemeng

Dynamic scheduling in multiproduct batch plants

Carlos A. Méndez, Jaime Cerdá *

INTEC (UNL-CONICET), Güemes 3450, 3000 Santa Fe, Argentina

Received 6 February 2003; accepted 7 February 2003

Abstract

This work introduces a novel MILP formulation for reactive scheduling of multiproduct batch plants to optimally generate updated schedules due to the occurrence of unforeseen events. It can also be used to improve a non-optimal production schedule before it is executed. The approach is based on a continuous-time problem representation that takes into account the schedule in progress, the updated information on the batches still to be processed, the present plant state and the time data. To limit the changes on the current schedule, rescheduling operations involving local reordering and unit reallocation of old batches as well as the insertion of new batches are just permitted. In contrast to previous contributions, multiple rescheduling operations can be performed at the same time. The MILP problem formulation is iteratively solved until no further improvement on the current schedule is obtained. Three large-size example problems were successfully solved with low computational cost.

Keywords: Reactive scheduling; Multiproduct batch plants; MILP Optimization model; Unforeseen events

1. Introduction

Most of the work reported so far on scheduling techniques for multiproduct batch plants (MBP) is aimed at generating a priori production schedules assuming that plant parameters and production requirements will remain without changes throughout the time horizon. However, an industrial environment is dynamic in nature and, therefore, the initial schedule must usually be updated in midweek because of different kinds of unexpected events. For instance, changes in batch processing/setup times, unit breakdown/startup, late order arrivals, orders cancellations, reprocessing of batches, delayed raw material shipments, modifications in order due dates and/or customer priorities and so on. As a result, the proposed schedule may become inefficient or even infeasible. So, the ability to rapidly react to such unforeseen events and periodically re-optimize the schedule on a daily or hourly basis is a key issue in batch plant operation. Proper adjustments to the current schedule may include simultaneous local reordering of batches at some equipment units, reassignment of certain batches to alternative equipment items due to unexpected unit failures and/or simply batch starting time shifting. Frequently, however, batches to be processed in the current shift may not be switched to another unit because required raw materials have already been sent in advance to the assigned unit (Musier and Evans, 1990). As a result, a portion of the schedule should be frozen while the remainder can be subject to re-optimization.

In the last decade, some new reactive scheduling methodologies have been reported. Hasebe, Hashimoto, and Ishikawa (1991) proposed a reordering algorithm for the scheduling of multiproduct batch plants consisting of parallel production lines with a shared unit. The algorithm involved two reordering operations, the insertion of a job and the exchange of two jobs. Because exchange operations required longer computational times and produced worse results, the authors finally applied a reordering algorithm performing just insertion operations, one at a time. Though simultaneous reordering of jobs generally lead to much better production sequences, such a rescheduling option was not allowed mostly because the search space became extremely large and the algorithm would be computationally very

* Corresponding author. Tel.: +54-342-599-174/77; fax: +54-342-550-944.
E-mail address: jcerda@intec.unl.edu.ar (J. Cerdá).

0098-1354/03/$ - see front matter © 2003 Elsevier Science Ltd. All rights reserved.
doi:10.1016/S0098-1354(03)00050-4

Nomenclature

Sets

I	orders to be scheduled ($I = I^{new} \cup I^{old}$)
I^{old}	orders belonging to the current schedule ($I^{old} \subseteq I$, $I^{old} = IS \cup IA$)
I^{new}	new orders to be inserted into the current schedule ($I^{new} \subseteq I$)
IA	subset of old orders that can be reassigned to other processing units during rescheduling ($IA \subseteq I^{old}$)
IS	orders that can be rearranged in the current processing sequence, i.e. without unit reallocation ($IS \subseteq I^{old} - IA$)
IS_i	old orders whose relative locations with regards to an existing order i may change during the rescheduling process: $IS_i = \{i' \in IS / i \neq i', j_i^{old} = j_{i'}^{old}, ((r_i^{old} - n_i \leq r_{i'}^{old} \leq r_i^{old} + n_i)$ or $(r_{i'}^{old} - n_{i'} \leq r_i^{old} \leq r_{i'}^{old} + n_{i'}))\}$
J	processing units
J_i	Available units to process order i ($J_i \subseteq J$)
$J_i^{\#}$	available units to reallocate order $I \in IA$ ($J_i^{\#} \subseteq J_i$)

Parameters

d_i	due date of order $i \in I$
j_i^{old}	unit allocated to old order $i \in I^{old}$ in the current schedule
M	a very large number
pt_{ij}	processing time of order $i \in I$ in unit j
r_i^{old}	position of the existing order $i \in I^{old}$ on the processing sequence of unit j_i^{old} before rescheduling
ro_i	release time of order $i \in I$
ru_j	ready time of unit $j \in J$
sl_i	slack time of order i, $sl_i = d_i - \text{Min}\{pt_{ij}, j \in J_i\}$
su_{ij}	setup time of order $i \in I$ in unit j
$\tau_{ii'j}$	sequence-dependent setup time between orders $i \in I$ and $i' \in I$ in unit j
α_i	weighting coefficient for earliness of order $i \in I$
β_i	weighting coefficient for tardiness of order $i \in I$
n_i	small integer representing the maximum number of closer predecessors and successors that can change location with order $i \in IS$ on the processing sequence

Variables

C_i	completion time for order i
E_i	earliness for order i
T_i	tardiness for order i
$X_{ii'}$	binary variable denoting that order $i \in I$ is processed before ($X_{ii'} = 1$) or after ($X_{ii'} = 0$) order $i' \in I$, when both were allocated to the same unit
Y_{ij}	binary variable denoting the allocation of order $i \in (I^{new} \cup IA)$ to unit j

expensive. Instead, the authors considered the possibility of aggregating consecutive jobs of the same type before assigning them as a block to a new location in the same processing sequence. Afterwards, the aggregated job was divided again and the insertion of jobs was continued until the performance index could no longer be improved.

Roslöf, Harjunkoski, Björkqvist, Karlsson, and Westerlund (2001) developed an MILP reordering algorithm to improve a non-optimal schedule or update the schedule in progress because of unforeseen events. The approach was applied to improve a manually generated production schedule for a paper-converting mill producing papers of different qualities. The example involved 61 jobs to be processed on a single processing unit with sequence-dependent setup times. Test runs were performed by releasing either one or two jobs at a time. As expected, the strategy of simultaneously releasing two jobs led to a stronger decrease of the objective function than the single-job option but the problem size and the computational effort both showed much bigger increases.

This paper introduces a novel MILP mathematical formulation for the MBP reactive scheduling problem that considers the information about: (i) the current schedule; (ii) the present plant state; and (iii) the deviations in plant parameters, order availabilities and time data from those used for generating the schedule in progress. The approach is based on a problem representation describing the production schedule by specifying the whole set of predecessors for every batch at the assigned equipment unit (Méndez, Henning, & Cerdá, 2001). It can still be applied if setup times are sequence-dependent. In contrast to previous contributions, the proposed approach allows to perform multiple resche-

duling operations at the same time. Among the operations being considered, it can be mentioned the insertion of new order arrivals, the reassignment of existing batches to alternative units due to equipment failures and the reordering and time-shifting of old batches at the current processing sequences. To prevent rescheduling actions from disrupting a smooth plant operation, limited changes in batch sequencing and unit assignment are just permitted. The goal is to meet all the production requirements at either maximum customer satisfaction or minimum total cost by making limited batch relocations to lower the overall impact on the schedule in progress. The new reactive scheduling strategy has been applied to large-scale industrial examples under different types of unexpected events with great success.

2. Problem definition

Given:

a) a single-stage multiproduct batch plant with several units $j \in J$ working in parallel.
b) a set of single-batch orders $i \in I^{old}$ (excluding cancelled orders) still to be processed during the current scheduling horizon.
c) the production schedule in progress, by providing the assigned unit j_i^{old} and the full set of preceding batches for any existing batch $i \in I^{old}$ on the processing sequence, before performing the rescheduling process.
d) the present state of the plant including information on equipment breakdown and anticipated startup of units coming from maintenance.
e) up-to-date processing times, setup times, release times and due dates for old batches.
f) new single-batch order arrivals $i \in I^{new}$ and their processing/setup times, release times and due dates.
g) the time at which each available equipment unit $j \in J$ will be ready to process the next batch $i \in I = (I^{old} \cup I^{new})$ on the processing sequence.
h) new equipment unit maintenance periods defined by their initial/final times.
i) the set of (released) old batches $i \in IS \subseteq I^{old}$ that can be locally reordered in the current processing sequence, i.e. without unit reallocation.
j) the subset of batches $i' \in IS_i$ with which a batch $i \in IS$ can change its relative location, i.e. switching from being a predecessor to becoming a successor or vice versa.
k) the set of (released) old batches $i \in IA \subseteq I^{old}$ that can be reassigned to alternative processing equipment items as well as the allowed unit options ($j \in J_i^{\#} \subseteq J_i$) for each released batch i.
l) the remaining time horizon at the rescheduling time.

The problem goal is to optimally rescheduling the old batches and inserting new ones in such a way that all production requirements be completed in a timely fashion and every unit-allocation and sequencing constraint be satisfied at the minimum of a weighted summation of batch earliness/tardiness over the batch set $i \in I$. In this way, the average inventory level is also minimized.

The sets of old batches $\{IS, IA\}$ being released for reordering and unit reallocation, respectively, the subset of old batches IS_i with which a batch $i \in IS$ can exchange location and the set of units $J_i^{\#}$ to which a batch $i \in IA$ can be reassigned can be all arbitrarily chosen by the user. The set IS_i is easily defined by specifying the parameter n_i, a small integer giving the number of closer predecessors or successors with which a batch $i \in IS$ can switch location in the processing sequence of the assigned unit j_i^{old}. In fact, one can freeze the location of an old batch $i \in IS$ by simply making IS_i equal to the empty set and, consequently, $n_i = 0$. If $n_i = 1$, then the set IS_i will just include the direct predecessor and the direct successor of batch i in the current processing sequence. Let us assume that r_i^{old} denotes the current position of the old batch i in the processing sequence. When $n_i = 1$, then the set IS_i will comprise the old batches located at positions $r_i^{old} \pm 1$ in the current batch sequence of unit j_i^{old}. When $n_i = 2$, then IS_i will include the old batches located at positions $r_i^{old} \pm 1$ and $r_i^{old} \pm 2$. In the general case, the set IS_i will contain the batches with positions $r_i^{old} \pm 1$, $r_i^{old} \pm 2, \ldots r_i^{old} \pm n_i$ in the batch sequence of unit j_i^{old}. Therefore, the set IS comprises all the batches $i \in I^{old}$ with $n_i > 0$. The number of reordering options for batch i rises with n_i but at the same time the problem size and the computational requirements will both show a stronger increase. The desired value for n_i is the one producing the best trade-off between those opposite trends. In turn, the set $IA \subseteq I^{old}$ just comprises every old batch with a set $J_i^{\#}$ including unit options other than j_i^{old}.

3. Model assumptions

1) The multiproduct batch plant is operated on an order-driven basis.
2) Model parameters are all deterministic.
3) Setup times are sequence-dependent.
4) Single-batch orders are just considered. Otherwise, a batch-sizing procedure should be performed to transform new product demands into batches of given sizes before applying the rescheduling algorithm.

5) Once the processing of an order starts, it must not be interrupted.
6) Batch split is not allowed.
7) No resource constraints except equipment are considered.

4. The mathematical model

4.1. Allocation constraints

4.1.1. For new batches $i \in I^{new}$ to be scheduled

$$\sum_{j \in J_i} Y_{ij} = 1 \quad \forall i \in I^{\text{new}} \tag{1.1}$$

4.1.2. For old batches $i \in IA$ that can be reassigned to other units $j \in J_i^{\#}$ during rescheduling

$$\sum_{j \in J_i^{\#}} Y_{ij} = 1 \quad \forall i \in IA \tag{1.2}$$

For an existing batch $i \in IA$, the assigned unit in the updated schedule (j_i^{new}) may be different from the current one (j_i^{old}). However, the unit j_i^{old} may also belong to $J_i^{\#}$, unless j_i^{old} is no longer available during the rescheduling horizon. The small sets $IA \subseteq I^{\text{old}}$ and $J_i^{\#} \subseteq J_i$ are both selected by the user.

4.2. Timing constraints

For an old/new batch being first processed at the assigned unit j after performing the rescheduling process, one of the following constraints should be complied depending on the batch set to which it belongs.

$$C_i \geq \text{Max}[ru_j, ro_i] + pt_{ij} + su_{ij} \quad \forall i \in IS \subseteq I^{\text{old}}, \quad j = j_i^{\text{old}} \tag{2.1}$$

$$C_i \geq \sum_{j \in J_i^{\#}} (\text{Max}[ru_j, ro_i] + pt_{ij} + su_{ij}) Y_{ij} \quad \forall i \in IA \subseteq I^{\text{old}} \tag{2.2}$$

$$C_i \geq \sum_{j \in J_i} (\text{Max}[ru_j, ro_i] + pt_{ij} + su_{ij}) Y_{ij} \quad \forall i \in I^{\text{new}} \tag{2.3}$$

where ru_j is the release time for unit j, i.e. the time at which unit j can process the next batch after rescheduling. Generally, ru_j is the completion time of the batch being processed in unit j at the rescheduling time.

4.3. Sequencing constraints

4.3.1. For every pair of existing batches i, $i' \in IS \subseteq I^{old}$ that are currently assigned to the same unit j_i^{old} and may undergo just batch reordering operations

$$C_i + \tau_{ii'j} + su_{i'j} \leq C_{i'} - pt_{i'j} + M(1 - X_{ii'}) \quad \forall i \in IS, \; i' \in IS_i, \quad i < i', \quad j = j_i^{\text{old}} = j_{i'}^{\text{old}} \tag{3.1}$$

$$C_{i'} + \tau_{i'ij} + su_{ij} \leq C_i - pt_{ij} + MX_{ii'} \quad \forall i \in IS, \quad i' \in IS_i, \quad i < i', \quad j = j_i^{\text{old}} = j_{i'}^{\text{old}} \tag{3.2}$$

The notation $i < i'$ in Eqs. (3.1) and (3.2) means that ord(i) is less than ord(i') in the set IS.

The set IS_i for batch $i \in IS$ is defined by specifying the value of n_i, i.e. the number of closer predecessors or successors on the processing sequence that can change location with batch i. If batch $i' \in IS_i$ is currently a successor of batch i at the processing sequence of unit j^{old}, it may become a predecessor of i after the rescheduling process. Nonetheless, an asymmetric set IS_i can also be handled by defining a pair of parameters (n_i^-, n_i^+) for each old batch $i \in I^{\text{old}}$. For instance, if batch i' is currently a successor of batch i but $n_i^+ = 0$, then the batch i' does not belong to IS_i and should be still processed after batch i in the new schedule as required by Eq. (3.3). For frozen old batches $i \notin IS$, the sequencing constraints are given by,

$$C_i + \tau_{ii'j} + su_{i'j} \leq C_{i'} - pt_{i'j} \quad \forall i, \quad i' \in IS, \quad i' \notin IS_i, \quad j = j_i^{\text{old}} = j_{i'}^{\text{old}}, \; (r_i^{\text{old}} < r_{i'}^{\text{old}}) \tag{3.3}$$

where r_i^{old} and $r_{i'}^{\text{old}}$ are the positions of the existing orders i, i' on the processing sequence of the unit j_i^{old} before performing rescheduling operations.

4.3.2. For a pair of batches involving an old batch $i \in IS$ currently assigned to unit j_i^{old} and another old/new batch that can be allocated to the same unit

$$C_i + \tau_{ii'j} + su_{i'j} \leq C_{i'} - pt_{i'j} + M(1 - X_{ii'}) + M(1 - Y_{i'j}) \quad \forall i \in IS, \quad i' \in IA, \quad j = j_i^{\text{old}} \in J_{i'}^{\#} \tag{4.1}$$

$$C_{i'} + \tau_{i'ij} + su_{ij} \leq C_i - pt_{ij} + MX_{ii'} + M(1 - Y_{i'j}) \quad \forall i \in IS, \quad i' \in IA, \quad j = j_i^{\text{old}} \in J_{i'}^{\#} \tag{4.2}$$

Since $J_{i'}^{\#}$ is a small set of units to which the old batch $i' \in IA$ can be reallocated, then this case applies only if $j_i^{\text{old}} \in J_{i'}^{\#}$. For a new batch $i' \in I^{\text{new}}$, it is assumed that the set of units $J_{i'}$ where i' can be processed also includes the unit j_i^{old}.

$$C_i + \tau_{ii'j} + su_{i'j} \leq C_{i'} - pt_{i'j} + M(1 - X_{ii'}) + M(1 - Y_{i'j}) \quad \forall i \in IS, \quad i' \in I^{\text{new}}, \quad j = j_i^{\text{old}} \in J_{i'} \tag{4.3}$$

$$C_{i'} + \tau_{i'ij} + su_{ij} \le C_i - pt_{ij} + MX_{ii'} + M(1 - Y_{i'j}) \quad \forall i \in IS, \quad i' \in I^{\text{new}}, \quad j = j_i^{\text{old}} \in J_{i'} \tag{4.4}$$

4.3.3. For a pair of old batches i, $i' \in IA$ that can be reassigned to the same equipment item during the rescheduling process

$$C_i + \tau_{ii'j} + su_{i'j} \le C_{i'} - pt_{i'j} + M(1 - X_{ii'}) + M(2 - Y_{ij} - Y_{i'j}) \quad \forall i, \ i' \in IA, \quad i < i', \quad j \in (J_i^{\#} \cap J_{i'}^{\#}) \tag{5.1}$$

$$C_{i'} + \tau_{i'ij} + su_{ij} \le C_i - pt_{ij} + M\ X_{ii'} + M(2 - Y_{ij} - Y_{i'j}) \quad \forall i, \ i' \in IA, \quad i < i', \quad j \in (J_i^{\#} \cap J_{i'}^{\#}) \tag{5.2}$$

4.3.4. For an existing batch $i \in IA$ and a new batch $i' \in I^{new}$ that can be assigned to the same unit during the rescheduling process

$$C_i + \tau_{ii'j} + su_{i'j} \le C_{i'} - pt_{i'j} + M(1 - X_{ii'}) + M(2 - Y_{ij} - Y_{i'j}) \quad \forall i \in IA, \quad i' \in I^{\text{new}}, \quad j \in (J_i^{\#} \cap J_{i'}) \tag{6.1}$$

$$C_{i'} + \tau_{i'ij} + su_{ij} \le C_i - pt_{ij} + M\ X_{ii'} + M(2 - Y_{ij} - Y_{i'j}) \quad \forall i \in IA, \quad i' \in I^{\text{new}}, \quad j \in (J_i^{\#} \cap J_{i'}) \tag{6.2}$$

4.3.5. For a pair of new batches that can be assigned to the same unit during the rescheduling process

$$C_i + \tau_{ii'j} + su_{i'j} \le C_{i'} - pt_{i'j} + M(1 - X_{ii'}) + M(2 - Y_{ij} - Y_{i'j}) \quad \forall i, \ i' \in I^{\text{new}}, \quad i < i', \quad j \in (J_i \cap J_{i'}) \tag{7.1}$$

$$C_{i'} + \tau_{i'ij} + su_{ij} \le C_i - pt_{ij} + M\ X_{ii'} + M(2 - Y_{ij} - Y_{i'j}) \quad \forall i, \ i' \in I^{\text{new}}, \quad i < i', \quad j \in (J_i \cap J_{i'}) \tag{7.2}$$

4.4. Order tardiness

$$T_i \ge C_i - d_i \quad \forall i \in I \tag{8}$$

4.5. Order earliness

$$E_i \ge d_i - C_i \quad \forall i \in I \tag{9}$$

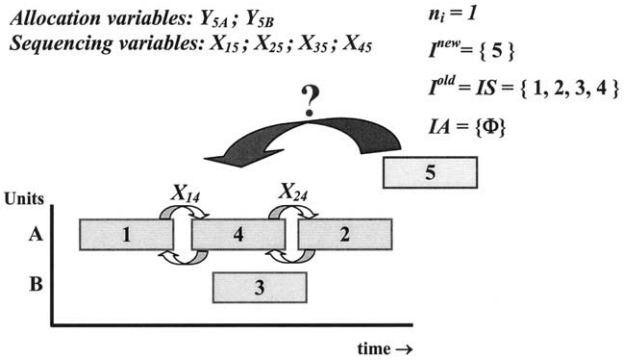

Fig. 1. A small example.

4.6. Problem objective function

$$\text{Min} \sum_{i \in I} \alpha_i E_i + \beta_i T_i \tag{10}$$

where α_i, β_i are weighting coefficients. When every order can be completed without tardiness, an equivalent objective function is given by,

$$\text{Max} \sum_{i \in I} C_i \tag{11}$$

5. An illustrative example

In order to illustrate the proposed MILP reactive scheduling approach, a small example will be studied (see Fig. 1). Let us consider a single-stage multiproduct batch plant with two parallel units. The current schedule specifies the processing of batches {1, 4, 2} in unit A and batch {3} in unit B, but rescheduling operations are required to insert a new single-batch order {5}. Re-ordering of old batches with just their direct predecessor or successor are only allowed ($n_i = 1$) for any batch $i \in IS$ while the new batch {5} can be assigned to any of the available units. Therefore: $IS = I^{\text{old}} = \{1, 2, 3, 4\}$ and $I^{\text{new}} = \{5\}$. Moreover, two assignment variables (Y_{5A}, Y_{5B}) and six sequencing variables (X_{14}, X_{24}, X_{15}, X_{25}, X_{35}, X_{45}) are to be defined (see Fig. 1). The problem constraint set for this small example is given below.

5.1. Allocation constraints

$$Y_{5A} + Y_{5B} = 1$$

5.2. Sequencing constraints between any pair of old batches

$$C_1 \le C_4 - pt_{4A} + M(1 - X_{14})$$

$$C_4 \le C_1 - pt_{1A} + MX_{14}$$
$$C_1 \le C_2 - pt_{2A}$$
$$C_2 \le C_4 - pt_{4A} + M(1 - X_{24})$$
$$C_4 \le C_2 - pt_{2A} + MX_{24}$$

5.3. Sequencing constraints among old and new batches

$$C_1 \le C_5 - pt_{5A} + M(1 - X_{15}) + M(1 - Y_{5A})$$
$$C_5 \le C_1 - pt_{1A} + MX_{15} + M(1 - Y_{5A})$$
$$C_2 \le C_5 - pt_{5A} + M(1 - X_{25}) + M(1 - Y_{5A})$$
$$C_5 \le C_2 - pt_{2A} + MX_{25} + M(1 - Y_{5A})$$
$$C_3 \le C_5 - pt_{5B} + M(1 - X_{35}) + M(1 - Y_{5B})$$
$$C_5 \le C_3 - pt_{3B} + MX_{35} + M(1 - Y_{5B})$$
$$C_4 \le C_5 - pt_{5A} + M(1 - X_{45}) + M(1 - Y_{5A})$$
$$C_5 \le C_4 - pt_{4A} + MX_{45} + M(1 - Y_{5A})$$

6. Using the MILP rescheduling approach to improving a non-optimal solution

By embedding preordering rules in the rigorous mathematical formulation of the batch scheduling problem, compact models can easily be derived. The use of MILP compact models represents a suitable alternative to quickly generate good production schedules for large-scale batch facilities. Preordering rules are indeed sequencing rules that establish the relative ordering of batches at every equipment unit beforehand. Depending on the rule being applied, the batches are sequenced by increasing processing times (SPT-rule), by increasing due dates (EDD-rule), by increasing slack-times (MST-rule) and so on. In this way, the compact scheduling problem formulation will just include the allocation variables since the 0–1 sequencing variables are no longer required. In other words, the major difference between the rigorous mathematical formulation and a compact scheduling model is that the sequencing constraints in the latter case are only expressed in terms of assignment variables.

If a good production schedule is to be generated from scratch by solving an MILP compact formulation, then there is no existing batch ($I^{old} = \varnothing$) and every batch belongs to the set I^{new}. Then, the sequencing constraints in Eqs. (7.1) and (7.2) are just needed. Let us assume that the batches will be arranged by increasing slack times (sl_i) at every equipment unit. In such a case, the sequencing constraints in Eqs. (7.1) and (7.2) in the resulting compact formulation will reduce to a single one with the following form:

$$C_i + \tau_{ii'j} + su_{i'j} \le C_{i'} - pt_{i'j} + M(2 - Y_{ij} - Y_{i'j}) \quad \forall i,\ i' \in I^{new},\ j \in (J_i \cap J_{i'}),\ (sl_i < sl_{i'}) \tag{7'}$$

However, compact scheduling models usually generate good, but non-optimal, solutions since the preordering rule may exclude the optimal schedule from the compact feasible region. Nevertheless, the initial schedule provided by the compact schedule model can be improved through reordering and reassignment operations by applying the proposed MILP rescheduling approach. During the rescheduling stage, all batches belong to the set I^{old} and, therefore, $I^{new} = \varnothing$. In this way, efficient production schedules for real-world multiproduct batch facilities can be found with very low computational cost.

7. The rescheduling algorithm

a) Define the sets I^{old} and I^{new} by incorporating the existing batches still to be processed in I^{old} and the new single-batch orders in I^{new}.
b) Define the set of available units $J_i^{\#}/J_i$ for every old/new batch i and the time periods during which each one can be used.
c) Define the production schedule to be updated by specifying the current assigned unit and the position r_i^{old} in the processing sequence for every old batch $i \in I^{old}$.
d) Define the small sets IS_i and IA by specifying the parameter n_i and the unit option set $J_i^{\#}$ for every old batch $i \in I^{old}$.
e) Generate the MILP rescheduling problem formulation based on the sets $\{IS, IS_i, IA, J_i^{\#}, J_i, I^{old}, I^{new}\}$.
f) Solve the MILP rescheduling formulation to find the best-updated schedule through local reordering and unit reassignment operations. If the solution found is an improvement with regards to the one identified in the previous iteration, go to step (g). Otherwise, either stop the procedure or enlarge the sets $\{IS_i, IA, J_i^{\#}\}$ by increasing n_i or the number of options in the set $J_i^{\#}$. In the latter case, go to step (g).
g) Update the sets $\{IS_i, IA, J_i^{\#}\}$ and return to step (e).
h) Each execution of steps (e)–(f) stands for a major iteration of the proposed rescheduling algorithm.

8. Results and discussion

8.1. Example 1

The proposed MBP reactive scheduling approach will be illustrated by tackling three example problems. Example 1, first introduced by Pinto and Grossmann (1995) and later studied by Ierapetritou, Hené, and

Table 1
Order data

Order	Due date (day)	Slack time (day)	Processing time (day)				Order	Due date (day)	Slack time (day)	Processing time (day)			
			U_1	U_2	U_3	U_4				U_1	U_2	U_3	U_4
O_1	15	13.806	1.538			1.194	O_{21}	30	26.386	7.317		3.614	
O_2	30	29.211	1.500			0.789	O_{22}	20	19.136			0.864	
O_3	22	21.182	1.607			0.818	O_{23}	12	8.376			3.624	
O_4	25	23.436			1.564	2.143	O_{24}	30	27.333			2.667	4.000
O_5	20	19.264			0.736	1.017	O_{25}	17	13.552	5.952		3.448	4.902
O_6	30	26.800	5.263			3.200	O_{26}	20	18.243	3.824			1.757
O_7	21	17.975	4.865		3.025	3.214	O_{27}	11	7.063	6.410			3.937
O_8	26	24.560			1.500	1.440	O_{28}	30	26.765	5.500			3.235
O_9	30	28.131			1.869	2.459	O_{29}	25	20.714				4.286
O_{10}	29	27.718		1.282			O_{30}	26	23.846				2.154
O_{11}	30	27.000		3.750		3.000	O_{31}	22	20.637		1.569		1.363
O_{12}	21	15.400		6.796	7.000	5.600	O_{32}	18	15.302			2.698	3.654
O_{13}	30	23.284	11.25			6.716	O_{33}	15	12.853		2.147		
O_{14}	25	23.473	2.632			1.527	O_{34}	10	7.342	3.265		2.658	
O_{15}	24	21.015	5.000			2.985	O_{35}	10	7.450	3.480			2.550
O_{16}	30	29.217	1.250			0.783	O_{36}	14	11.742		2.258		
O_{17}	30	26.964	4.474			3.036	O_{37}	24	21.855		2.145	2.194	
O_{18}	30	28.571		1.429			O_{38}	16	14.150	2.365			1.850
O_{19}	13	10.313		3.130		2.687	O_{39}	22	19.970	2.030			
O_{20}	19	17.926	2.424		1.074	1.600	O_{40}	23	21.110		1.890		
Setup time			0.180	0.175		0.237				0.180	0.175		0.237

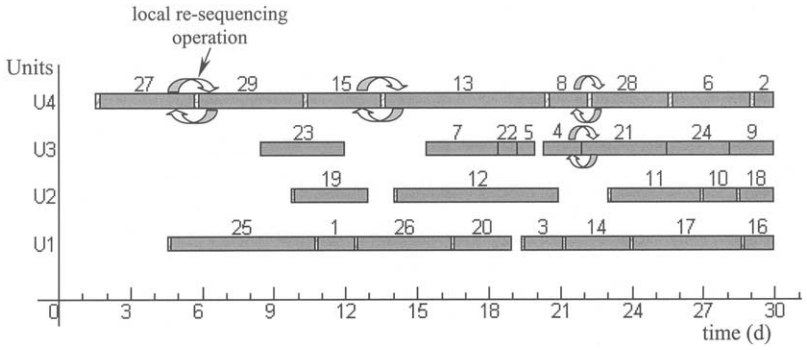

Fig. 2. Production schedule for example 1a reported by Pinto and Grossmann (1995).

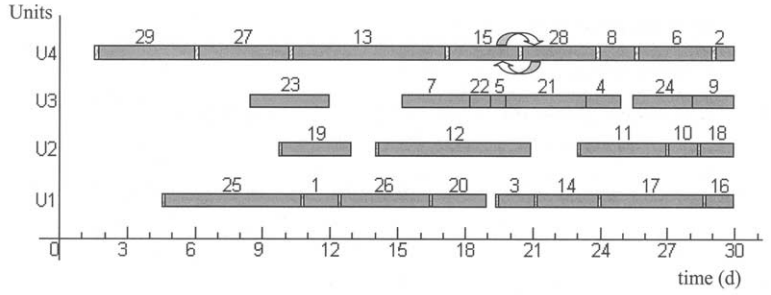

Fig. 3. Improved schedule for example 1a after the first rescheduling step.

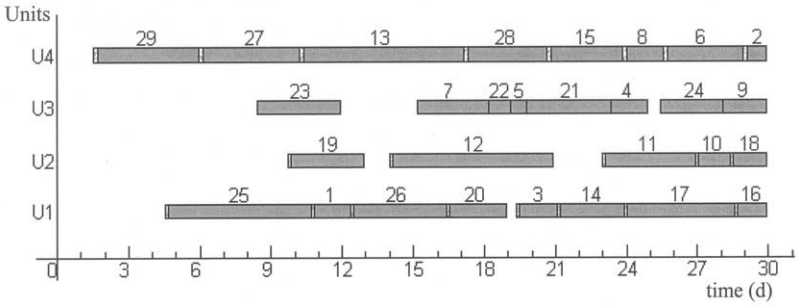

Fig. 4. Improved schedule for example 1a after the second rescheduling step.

Floudas (1999), involves the scheduling of a single-stage multiproduct batch plant with four parallel units. Twenty-nine single-batch orders O_1–O_{29} are to be processed within a 30-day scheduling horizon. Order due dates and unit-dependent processing and setup times are all given in Table 1. The proposed MILP rescheduling approach has first been applied to improve the best solution reported by Pinto and Grossmann (1995) and shown in Fig. 2 (example 1a). Since these authors applied preordering rules to solve the problem with a more compact mathematical formulation, one can expect that the solution reported by them can be improved through the proposed MILP rescheduling algorithm. Ierapetritou, Hené, and Floudas (1999) have also solved Example 1 by using an event-based approach with preordering rules. To improve the schedule reported by Pinto and Grossmann (1995), we will assume that reordering operations with $n_i = 1$ are just allowed during rescheduling. Therefore, all the batches belong to the set *IS* and each batch can only exchange location with either its direct predecessor or its direct successor on the processing sequence. A total of 25 reordering operations (rather than one or two operations at a time) can potentially be performed simultaneously. By solving the proposed MILP approach, four re-sequencing operations described in Fig. 2 are really accomplished to improve the objective function from 612.80 to 619.26, thus yielding the production schedule shown in Fig. 3. Another major iteration of the approach produces an additional increase of the objective function and, consequently, an improved schedule (see Fig. 4). A further application

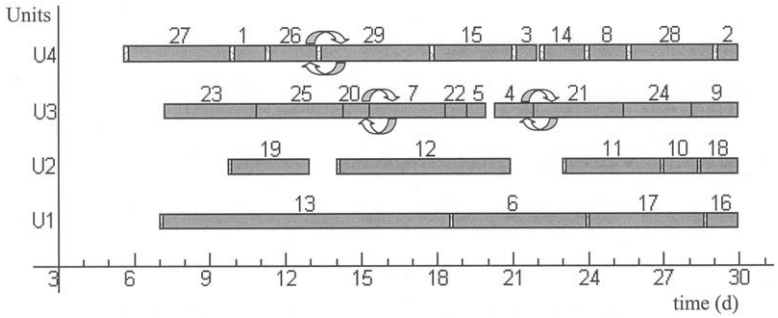

Fig. 5. Initial schedule for example 1b by embedding the MST-rule in the proposed approach.

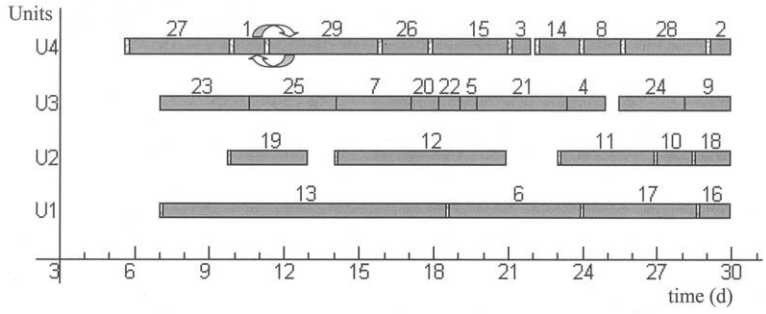

Fig. 6. Improved schedule for example 1b after the first rescheduling step.

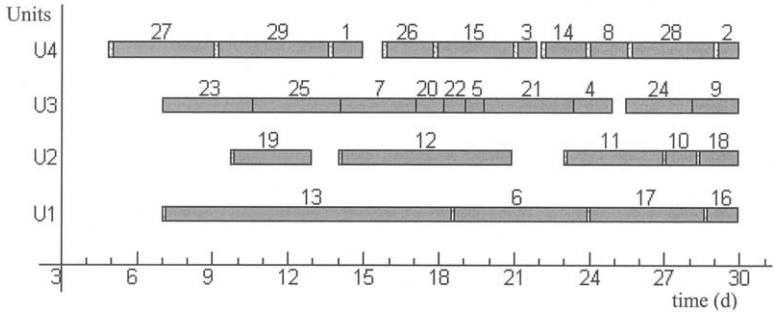

Fig. 7. Improved schedule for example 1b after the second rescheduling step.

of the rescheduling algorithm does not yield any extra improvement at all.

The MILP formulation was applied again to example 1 to this time establish a very good solution by optimally allocating units to the whole set of batches while assuming that they are sequenced at every unit by increasing slack times (sl_i). To do that, the minimum-slack-time rule (MST) was embedded in the MILP model. Moreover, all the batches are assumed to belong to I^{new} and, consequently, I^{old} is an empty set. Similarly to Pinto and Grossmann (1995), order completion times were maximized to reduce order earliness as much as possible. In this way, the production schedule shown in Fig. 5 was found (example 1b). Since the application of the minimum-slack-time (MST) preordering rule for batch sequencing does not always lead to the best schedule, the proposed approach was once more applied to improve it. This time we assume that all the batches are included in the set IS and local reordering operations with $n_i = 1$ for any batch $i \in IS$ are just allowed during rescheduling. To further reduce the total order earliness, three re-sequencing operations outlined in Fig. 5 were performed to generate a better schedule shown in Fig. 6. Another major iteration of the approach to further improve the updated schedule produces another increase in the objective function by performing a single re-sequencing operation as indicated in Fig. 6. The improved schedule described in Fig. 7 and Table 2 cannot be upgraded through another rescheduling step. Overall, the objective function increases from 627.08 (initial schedule) to 632.52 (final schedule) by making four re-sequencing operations. Therefore, any of the schedules generated by the proposed approach is better than the one reported by Pinto and Grossmann (1995)

Table 2
Summary of results for examples 1–3

Order	Due date (day)	Completion time (day)			Order	Due date (day)	Completion time (day)		
		Example 1b (Fig. 7)	Example 2 (Fig. 9)	Example 3 (Fig. 11)			Example 1b (Fig. 7)	Example 2 (Fig. 9)	Example 3 (Fig. 11)
O_1	15	15.000	13.338		O_{21}	30	23.436	25.464	26.370
O_2	30	30.000	30.000	31.193	O_{22}	20	19.086	19.264	21.192
O_3	22	22.000	22.000	20.863	O_{23}	12	10.675	5.4970	
O_4	25	25.000	21.850	22.756	O_{24}	30	28.131	28.131	29.037
O_5	20	19.822	20.000	19.808	O_{25}	17	14.123	11.603	
O_6	30	23.916	23.916	26.257	O_{26}	20	17.723	18.554	18.554
O_7	21	17.148	14.628		O_{27}	11	9.046	4.5970	
O_8	26	25.502	25.502	26.695	O_{28}	30	28.974	28.974	30.167
O_9	30	30.000	30.000	30.906	O_{29}	25	13.569	11.907	
O_{10}	29	28.396	28.396	28.396	O_{30}	26		20.945	23.254
O_{11}	30	26.939	26.939	26.939	O_{31}	22		20.935	20.935
O_{12}	21	21.000	16.871		O_{32}	18		17.326	20.328
O_{13}	30	18.473	13.455		O_{33}	15		9.900	
O_{14}	25	23.825	23.825	25.018	O_{34}	10		8.155	
O_{15}	24	20.945	16.560		O_{35}	10		7.384	
O_{16}	30	30.000	30.000	27.687	O_{36}	14		7.578	
O_{17}	30	28.570	28.570	32.341	O_{37}	24		19.191	19.191
O_{18}	30	30.000	30.000	30.000	O_{38}	16		16.000	
O_{19}	13	13.000	5.1450		O_{39}	22		18.473	20.814
O_{20}	19	18.222	18.400	18.604	O_{40}	23		23.000	23.000

Table 3
Model sizes and computational requirements for examples 1–3

Example	Binary vars, cont. vars, rows	Objective function	CPU time	Nodes
No. 1a—solution reported by Pinto and Grossmann (1995) (Fig. 2)	441, 875, 1791	612.798	1257.17[a]	204
No. 1a—solution reported by Ierapetritou, Hené, and Floudas (1999)	57,172, 559	600.186	0.28[a]	5
No. 1a—first rescheduling step (Fig. 3)	25, 29, 157	619.26	0.77[b]	289
No. 1a—second rescheduling step (Fig. 4)	25, 29, 157	619.51	0.66[b]	223
No. 1b—this approach with an embedded MST-rule (Fig. 5)	57, 29, 559	627.082	49.49[b]	11941
No. 1b—first rescheduling step (Fig. 6)	25, 29, 157	631.616	1.21[b]	592
No. 1b—second rescheduling step (Fig. 7)	25, 29, 157	632.521	0.77[b]	363
No. 2—insert 11 new orders into current schedule (Fig. 8)	147, 40, 479	760.957	67.72[b]	13238
No. 2—first rescheduling step (Fig. 9)	36, 40, 212	762.273	16.64[b]	7207
No. 3—reschedule due to unit maintenance (Fig. 11)	108, 25, 319	73.735	29.93[b]	11180

[a] Seconds on HP 9000-730 with GAMS/OSL.
[b] Seconds on Pentium II PC (400 MHz) with ILOG/CPLEX.

Table 4
Impact of the parameter n_i on the problem size and results for example 1b

n_i	Binary vars, cont. vars, rows	Objective function	CPU time[a]	Nodes
1	25, 29, 157	631.616	1.21	592
2	46, 29, 202	632.623	24.80	11922
3	63, 29, 233	632.623	75.58	30565
4	76, 29, 252	632.623	98.54	33412

[a] Seconds on Pentium II PC (400 MHz) with ILOG/CPLEX.

featuring a value of 612.8. Moreover, the problem size shows a drastic reduction of almost one-order-of-magnitude at any iteration not only in the number of binary and continuous variables but also in the overall CPU time (see Table 3).

Let now study the effect of adopting values of n_i greater than 1 on the rescheduling problem size, the CPU requirements and the objective function improvement per iteration. To do that, it was chosen the production schedule shown in Fig. 5 as the initial schedule. We will try to improve such an initial solution by applying the rescheduling algorithm with $n_i = 2$, 3 and 4, respectively. In this way, the allowed number of simultaneous reordering operations will almost grow by a factor n_i; i.e. two times, three times and four times, respectively. Table 4 presents the results of such test runs after a single major iteration of the algorithm, including the rescheduling problem size, the objective function, the CPU time requirement and the number of explored nodes. In all cases, the same updated schedule featuring an objective function equal to 632.623 was found. It represents a slight improvement with regards to the one provided by the rescheduling algorithm with $n_i = 1$ after two major iterations; i.e. just a 0.016% increase. On the other hand, the problem size and the CPU requirements both show a polynomial growth with n_i. From these results, it should be concluded that the selection of low values for n_i should be favored since it leads to a similar improvement with much lower computational cost.

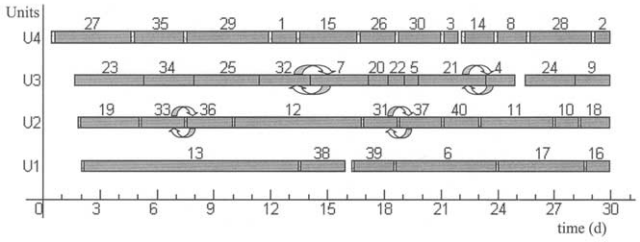

Fig. 8. Initial schedule for example 2.

8.2. Example 2

Example 2 assumes that new late single-batch orders O_{30}–O_{40} have arrived before starting the execution of the production schedule found for Example 1 (Fig. 7). Due dates and processing times for the new orders are given in Table 1. In this case, the proposed rescheduling approach is initially applied to fully optimize batch-unit allocation and batch sequencing for the new single-batch orders $i \in I^{new}$ while just permitting local re-sequencing operations with $n_i = 1$ for any old order $i \in I^{old} = IS$. The new schedule found in 67.7s on a Pentium II PC (400 MHz) with ILOG OPL studio 2.1 (ILOG, 1999), using the embedded CPLEX mixed-integer optimizer 6.5.2 release, is depicted in Fig. 8. By performing a second major iteration, but this time allowing re-sequencing operations with $n_i = 1$ for every old/new batch $i \in IS$, a better schedule has been generated (see Fig. 9 and Table 2). No further improvement was achieved by executing another major iteration if batch reordering with $n_i = 1$ is only permitted.

8.3. Example 3

Finally, Example 3 assumes that the best solution for the 40-order scheduling problem considered in Example

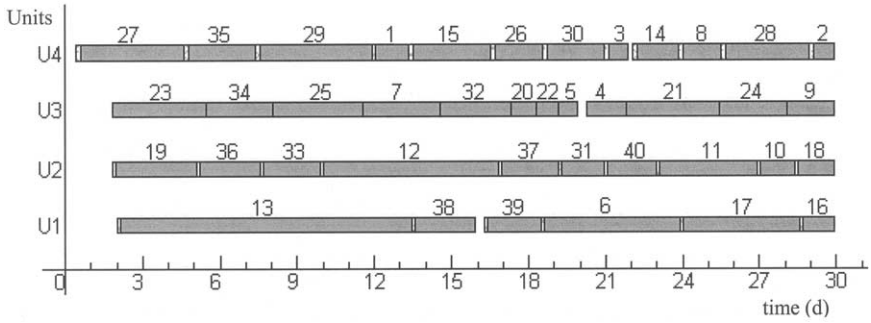

Fig. 9. Improved schedule for example 2 after the first rescheduling step.

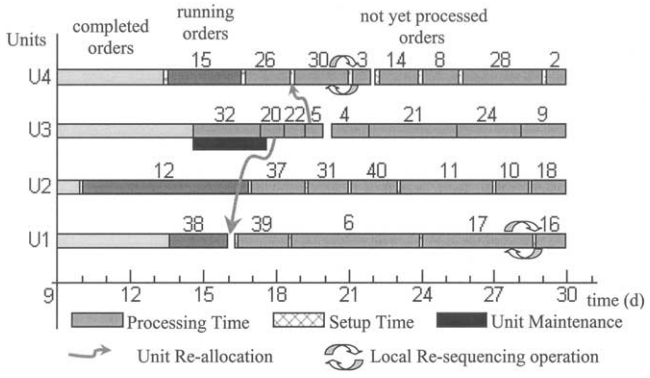

Fig. 10. Schedule in progress and rescheduling operations for example 3.

2 has been partially executed as predicted. However, an unexpected 3-day maintenance period for unit U_3 at time $t = 14.6$ days, after processing order no. 7, makes necessary to perform a rescheduling process over the set of 25 orders not yet processed (see Fig. 10). The problem goal is to minimize a weighted combination of order earliness and tardiness, with higher penalties associated to tardy orders ($\alpha_i = 1$, $\beta_i = 5$). During rescheduling, the batches currently allocated to unit U_3 may be reassigned to other units and sequenced in the best possible way (set IA) while local re-sequencing operations with $n_i = 1$ are just allowed for the remaining orders (set IS). If rescheduling is not performed and the orders allocated to U_3 should wait until the equipment unit resumes operation, then the overall tardiness for such orders will be as large as 13.55 days. Certainly, any unit different from U_3 is processing a batch at the rescheduling time $t = 14.6$ days and consequently its ready time will be later than 14.6. Through a single major iteration of the proposed MILP rescheduling approach, the best-updated schedule shown in Fig. 11 and Table 2 was found in 30 s. In this way, the total tardiness is decreased from 13.55 to 8.84 days while the maximum order tardiness dropped from 2.71 to 2.34 (see Table 5).

9. Conclusions

Large-scale scheduling packages have usually no real provision for cheaply correcting the current schedule for small to middle-size changes, except to make full rescheduling. This work intends to provide a flexible MILP systematic tool to efficiently update schedules by simultaneously making limited reallocation and reordering operations, while keeping the extent of the schedule adjustments under user control. The rescheduling tool can still be used even if sequence-dependent setup times are to be considered. The proposed MILP based algorithmic approach should be iteratively applied on the current schedule until no further improvement is obtained. Successful solution to three large-size examples involving from 29 to 40 orders shows that, in any case, the rescheduling steps requires a very low CPU time. Such examples dealt with the improvement of an available non-optimal schedule, the insertion of new

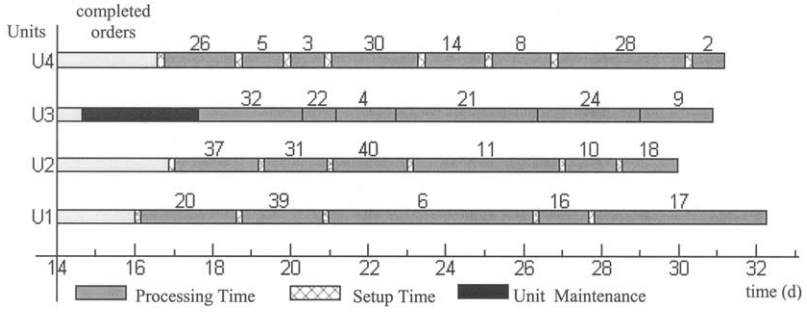

Fig. 11. Improved schedule in progress for example 3.

Table 5
Comparative table of performance measures for examples 1–3

Performance measure (days)	Example 1a, Pinto and Grossmann (1995)	Example 1b (Fig. 7)	Example 2 (Fig. 9)	Example 3 (Fig. 11)
Maximum earliness	14.787	11.527	16.545	4.809
Total earliness	82.202	62.479	132.727	29.535
Average earliness	2.834	2.154	3.318	1.181
Maximum tardiness				2.341
Total tardiness				8.84
Average tardiness				0.3536

jobs and the update of a schedule already in progress because of an unexpected unit maintenance period.

Acknowledgements

The authors acknowledge financial support from FONCYT under Grant 14-07004, and from 'Universidad Nacional del Litoral' under CAI+Ds 048 and 121.

References

Hasebe, S., Hashimoto, I., & Ishikawa, A. (1991). General reordering algorithm for scheduling of batch process. *Journal of Chemical Engineering of Japan 24* (4), 483–489.

Ierapetritou, M. G., Hené, T. S., & Floudas, C. A. (1999). Effective continuous-time formulation for short-term scheduling. 3 Multiple intermediate due dates. *Industrial and Engineering Chemistry Research 38*, 3445–3461.

ILOG OPL studio 2.1 user's manual (1999). ILOG S.A., France.

Méndez, C. A., Henning, G. P., & Cerdá, J. (2001). A continuous-time approach to short-term scheduling of resource-constrained multistage batch facilities. *Computers and Chemical Engineering 25*, 701–711.

Musier, R. F. H., & Evans, L. B. (1990). *Batch process management. Chemical Engineering Progress 87* (6), 66–77.

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.

Roslöf, J., Harjunkoski, I., Björkqvist, J., Karlsson, S., & Westerlund, T. (2001). An MILP-based reordering algorithm for complex industrial scheduling and rescheduling. *Computers and Chemical Engineering 25*, 821–828.