A MILP REACTIVE SCHEDULING FRAMEWORK FOR RESOURCE-CONSTRAINED MULTISTAGE BATCH FACILITIES

Carlos A. Méndez and Jaime Cerdá*
INTEC (UNL - CONICET)
Güemes 3450 - 3000 Santa Fe – ARGENTINA
E-mail: jcerda@intec.unl.edu.ar

Abstract

This work introduces a MILP reactive scheduling framework for resource-constrained multistage batch facilities with multiple parallel units at each stage. The approach is based on a continuous time domain representation that takes into account the schedule currently in progress, the updated information on the old production orders still to be processed, new order arrivals, the present plant state and the limited availability of renewable discrete resources like processing units and manpower. Order due dates and sequence-dependent changeovers can also be considered. The proposed technique is able to generate updated schedules when unforeseen events like deviations in processing times, equipment breakdown or batch reprocessing occur. To avoid full-scale rescheduling, the approach just allows schedule modifications involving starting time shifting, limited resource reallocation and local batch reordering at any discrete resource. The rescheduling algorithm is iteratively performed to restore feasibility at minimum increase of the objective function. Performance measures like make-span or average order lateness can be used. A large-scale case study was successfully solved at low computational time.

Keywords

MILP model, Discrete resources, Reactive scheduling, Rescheduling algorithm, Multistage batch plants

Introduction

A dynamic industrial environment usually requires a constant updating of the production schedule in progress. Since unexpected events continually arise throughout the time horizon, some modifications in the proposed schedule must be introduced on a regular basis. Instead of performing full-scale rescheduling, it is better performed limited resource-task reallocation and local reordering of tasks at any discrete resource to regain feasibility at minimum additional cost. Consequently, one of the major differences between predictive and reactive scheduling is that a large fraction of the scheduling decisions already taken will remain the same or at most experience limited changes during the rescheduling process. This fact can be used to sharply reduce the batch rescheduling problem size.

Previous work on batch scheduling mostly assumed that discrete renewable resources other than processing equipment are available in unlimited amounts. However, production tasks usually require renewable discrete resources like manpower, tools, etc., that are available in finite quantities. Sometimes, they severely limit the number of simultaneous processing tasks and, consequently, the production schedule found by applying those scheduling methodologies could become infeasible. In fact, such resource constraints force the sequential execution of some process operations leading to a significant increase in the schedule make-span or the overall tardiness. A few number of continuous-time predictive scheduling approaches for resource-constrained multistage batch facilities have already been published (Pekny and Reklaitis, 1998; Pinto and Grossmann, 1998). They generate optimal production schedules for the next time horizon even if limited amounts of discrete resources other than equipment are available. However, an industrial

* To whom all correspondence should be addressed

environment continually changes and the original schedule must be re-optimized on a daily or hourly basis. Some work on reactive scheduling of batch processes without resource constraints has already been done. Hasebe et al. (1991) proposed a reordering algorithm to update the schedule of multiproduct batch plants involving parallel production lines with a shared unit. Two types of reordering operations, i.e. the insertion of a job and the exchange of two jobs, were just allowed. However, they should be performed one at a time, since otherwise the algorithm would become computationally expensive. In turn, Roslöf et al. (2001) developed an MILP reordering algorithm to improve a non-optimal schedule or update the current schedule by iteratively releasing and reassigning or resequencing a small number of jobs. Again, the simultaneous reordering of two or more jobs may produce a strong increase in the problem size.

This work extends the MILP reactive scheduling approach for single-stage batch plants, introduced by Méndez and Cerdá (2001), to resource-constrained multistage batch facilities. The proposed MILP framework relies on three key elements: (a) separate handling of allocation and sequencing decisions; (b) a problem representation describing the processing task sequence at any resource through the full set of (direct/non-direct) predecessors for each task; (c) a uniform treatment of different discrete resources to define a common set of 0-1 sequencing variables for all of them. In this way, a low-size problem formulation allowing to perform multiple rescheduling moves at the same time has been derived. The approach was successfully applied to a real-world industrial case study.

Problem Statement

Given: (i) a multistage multiproduct batch plant with multiple parallel units $j \in J_s$ at each processing stage *s;* (ii) the set of processing units $j \in J_s$ available at each stage *s*; (iii) the set of available discrete resources *R* other than equipment; (iv) the set of old production orders $i \in I^{old}$ still to be completed, including those partially processed; (v) the new order arrivals $i \in I^{new}$; (vi) the sequence of processing stages $s \in S_i$ for each old/new order $i \in I$; (vii) the set of alternative units $j \in J_{is}$ and discrete resources $r \in R_{is}$ that can be allocated to each task (*i,s*); (viii) the changes in the plant state because of unexpected events like equipment breakdowns, worker absenteeism, etc.; (ix) up-to-date processing times, sequence-dependent setup times and unit-dependent resource requirements for every task (*i,s*); (x) the production schedule in progress, by providing the discrete resource items currently assigned to each task (*i,s*) and the processing task sequence at any available resource before rescheduling; (xi) the last processing stage already completed or currently in progress for old production orders at the rescheduling time; (xii) the expected completion times for ongoing processing tasks; (xiii) the set of production tasks that can be just locally reordered and the reordering extent during rescheduling; (xiv) the set of tasks $(i,s) \in TA$ that can be reassigned to other resource items and the reallocation alternatives during rescheduling and (xv) the remaining time horizon.

The problem goal is to reschedule old orders still to be processed and insert the new ones through the allowed rescheduling actions in such a way that all production orders are completed in time and every resource constraint is satisfied at minimum makespan.

The Mathematical Framework

The knowledge of the current production schedule becomes an important piece of information to be explicitly considered by the problem representation. The proposed mathematical framework describes the processing task sequence at every resource item by providing the full set of (direct/non-direct) predecessors for any task (*i',s'*) through the sequencing variables $X_{is,i's'}$. If the relative locations of tasks (*i,s*) and (*i',s'*) assigned to the same resource item is frozen and task (*i,s*) is currently processed before, then $X_{is,i's'} = 1$. As a result, $X_{is,i's'}$ is no longer a problem variable. Moreover, dispatch rules like EDD or SPT rules can easily be embedded in the problem formulation. Let us assume that the relative ordering of tasks (*i,s*) and (*i',s'*) should comply the EDD rule and $d_i < d_{i'}$. Then, $X_{is,i's'}$ is equal to 1 and such a variable can be eliminated. In both cases, a problem size reduction is achieved.

Resource allocation constraints

A single unit $j \in J_{is}$ and the required amount of discrete resources like manpower should be assigned to every processing task (*i,s*).

$$\sum_{j \in J_{is}} Y_{isj} = 1 \qquad \forall (i,s) \in (T^{new} \cup TA) \tag{1}$$

$$\sum_{r \in R_{is}} Y_{isr} = \sum_{j \in J_{is}} b_{isrj} \; Y_{isj} \qquad \forall (i,s) \in (T^{new} \cup TA) \tag{2}$$

where b_{isr} is the amount of resource *r* required by *(i,s).*

Sequencing constraints

Let us assume that production tasks (*i,s*) and (*i',s'*) belong to the set ($T^{new} \cup$ TA). Then, sequencing constraints must be imposed on any pair of tasks (*i,s*) and (*i',s'*) only if they have been allocated to the same resource item $r \in R_s$, i.e. $Y_{isr} = Y_{i's'r} = 1$. If task (*i,s*) is performed before task (*i',s'*), i.e. $X_{is,i's'} = 1$, then constraint (3.1) will hold to ensure that task (*i',s'*) begins after completing task (*i,s*). In such a case, the other sequencing constraint (3.2) becomes redundant. If instead task (*i,s*) is processed later ($X_{is,i's'} = 0$), then constraint (3.2) will be enforced to prevent from starting (*i,s*) before ending (*i',s'*). Therefore, a single variable $X_{is.i's'}$ is required to control the relative ordering of any pair of tasks (*i,s*) and (*i',s'*) at the processing sequence of any shared resource item *r*.

$$C_{is} + \tau_{isi's'r} \le S_{i's'} + M(1 - X_{isi's'}) + M(2 - Y_{isr} - Y_{i's'r}) \quad \forall (i,s),(i',s') \in (T^{new} \cup TA)$$
$$r \in (R_{is} \cap R_{i's'}) : (i < i')\ or\ (i = i'\ and\ s < s') \qquad (3.1)$$

$$C_{i's'} + \tau_{i's'isr} \le S_{is} + M\ X_{isi's'} + M(2 - Y_{isr} - Y_{i's'r}) \quad \forall (i,s),(i',s') \in (T^{new} \cup TA)$$
$$r \in (R_{is} \cap R_{i's'}) : (i < i')\ or\ (i = i'\ and\ s < s') \qquad (3.2)$$

Let us now assume that the processing tasks (*i,s*) and (*i',s'*) belong to the set T^{old} and both are currently assigned to the same resource item *r*. Moreover, the reassignment of either one to an alternative resource item of the same type is not permitted. In such a case, three different cases can be defined: (A) The relative ordering of tasks (i,*s*) and (*i',s'*) at the rth-processing sequence is frozen. Assuming that the task (*i',s'*) is currently processed before, then constraints (3.1) and (3.2) reduce themselves to:

$$C_{i's'} + \tau_{i's'isr} \le S_{is} \quad \forall (i,s),(i',s') \in T^{old}$$
$$r \in (R_{is} \cap R_{i's'}) : (i < i')\ or (i = i'\ and\ s < s') \qquad (3.3)$$

Therefore, the related sequencing variables and one of the sequencing constraints are withdrawn from the problem formulation. (B) A dispatch rule has been embedded in the problem formulation to preordering tasks (*i,s*) and (*i',s'*). This particular case is similar to the previous one and the relative ordering is known beforehand. If either one of the tasks or both belong to the set ($T^{new} \cup TA$), then the RHS of the related sequencing constraint will include the allocation variables Y_{isr}. (c) The relative ordering of tasks (*i,s*) and (*i',s'*) can be changed during rescheduling. In such a case, the last RHS term in both constraints (3.1) and (3.2) must be withdrawn.

Timing constraints

The starting time for every required task (*i,s*) can be computed as follows:

$$S_{is} = C_{is} - \sum_{j \in J_{is}} pt_{isj}\ Y_{isj} \quad \forall (i,s) \in T \qquad (4)$$

$$S_{is} \ge \sum_{j \in J_{is}} rt_j\ Y_{isj} \quad \forall (i,s) \in T \qquad (5)$$

Moreover, *t*he task (*i,s*+1) can never begin before the preceding task (*i,s*) has been completed. Then,

$$CT_{is} \le ST_{i(s+1)} \quad \forall (i,s) \in T : s \ne \{s_i^{\ell}\} \qquad (6)$$

Problem objective function

The problem goal is to minimize the make-span.

$$CT_{is} \le MK \quad \forall (i,s) \in T : s = \{s_i^{\ell}\} \qquad (7)$$

$$Min \quad MK \qquad (8)$$

The Rescheduling Algorithm

(1) Define the task sets T^{old}, T^{new} and $TA \subseteq T^{old}$ to be considered during the rescheduling process.
(2) Allocate discrete resources to new processing tasks (*i,s*) and/or old tasks (*i,s*)∈TA currently assigned to resources no longer available because of equipment breakdown or operator absenteeism. The arrangement of processing tasks $(i,s) \in (T^{new} \cup TA)$ is initially assumed to follow a specified dispatch rule embedded in the problem formulation, like the minimum slack time rule. Not only the dispatch rule controls the relative ordering of such tasks among themselves but also with regards to those required by old tasks $(i,s) \notin TA$. During this step, the relative ordering of any pair of tasks $(i,s) \notin TA$, at any resource item is assumed to be frozen. Therefore, the related sequencing variables are eliminated and the problem formulation just includes the allocation variables Y_{isr}. Solve the resulting MILP model so as to minimize the order in-process time.
(3) Rearrange processing tasks allocated to the same resource item by allowing any pair of consecutive tasks to switch their locations at the processing sequence. During this step, resource reallocation is not permitted. A small-size MILP formulation just involving sequencing variables is so defined to improve the schedule found in Step (2).
(4) Repeat iteratively Step (3) until no further improvement in the objective function is achieved.

Illustrative Example

The proposed rescheduling approach has been applied to a large-scale resource-constrained multiproduct batch plant involving 4 processing stages and 10 units. Problem data can be found in Pinto and Grossmann (1997). Twenty production orders are to be processed during the time horizon. No more than 6 units can be simultaneously operated since a single operator is required to run each unit and a crew of 6 operators is just available. Figure 1 shows the original production schedule. It was found by following a solution strategy quite similar to the proposed rescheduling algorithm. The only difference is that I^{old} and *IA* are both empty sets and all production orders are included in the set I^{new}. Let us now suppose that the following unexpected events occurred at time t= 57 h: (i) two late orders O21 and O22, not included in the original problem, which are similar to orders O19 and O20, respectively, have arrived; (ii) unit E7 is no longer available because of equipment malfunctioning; (iii) the operator R6 is not at work because of sickness. As a result, a new unit is to be assigned to perform the third processing stage on production orders {O2, O4, O5, O11, O13}. In addition, a new operator is to be allocated to different processing stages on the production orders {O2, O5, O8, O9, O13, O14, O15, O17, O18, O20. The proposed rescheduling algorithm has been applied to restore feasibility at minimum cost. Figure 2 shows the revised scheduling after the resource allocation step. Next, the reordering step was iteratively applied until no further improvement in the objective function was achieved. Despite the processing tasks previously assigned to operator R6 imply a total processing time of 175 h, the schedule make-span just grows from 232 h to 304 h, i.e. an increase of 72 h. Table 1 shows the computational requirements and the best objective value at each rescheduling step.

Conclusions

A new MILP reactive scheduling algorithm has been developed to update the short-term schedule of resource-constrained multiproduct batch facilities.

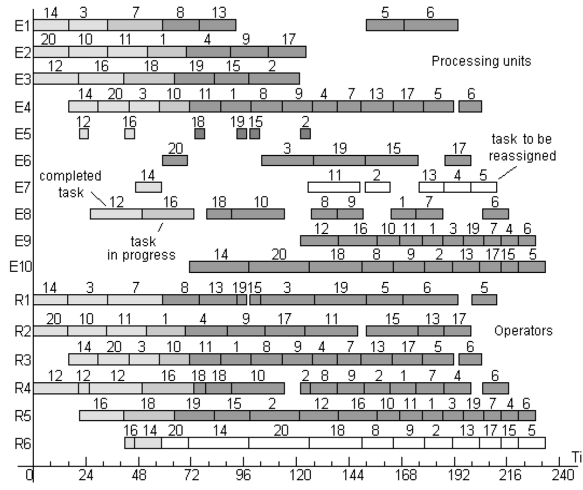

Figure 1. Schedule in progress

Nomenclature

Sets

I	production orders
I^{new}	late production orders to be inserted into the current schedule
I^{old}	production orders belonging to current schedule
T	production task (i,s) for every order I
T^{new}	production tasks (i,s) for late orders I
T^{old}	production tasks (i,s) still to be processed
TA	production tasks (i,s) to be reassigned to alternative resource items
J	processing units
S	processing stages
S_i	sequence of stages for order i
R	resource items other than equipment

Parameters

d_i	due date of order i
pt_{isj}	processing time for task (i,s) in unit j
s_i^{ℓ}	last processing stage for order i
$\tau_{isi's'r}$	sequence-dependent setup time between task (i,s) and task (i',s') at resource r

Variables

MK	make-span
CT_{is}	completion time for task (i,s)
ST_{is}	starting time for task (i,s)
$X_{is\,i's'}$	binary variable denoting that task (i,s) is processed before ($X_{isi's'}=1$) or after ($X_{isi's}=0$) task (i',s')
Y_{isr}	binary variable denoting that task (i,s) is allocated to resource item r

Table 1. Model size and computational requirements

binary vars, cont. vars, constraints	objective function	CPU time[a]
123, 176, 3232	294.51	202.84

[a] Seconds on Pentium III PC (933 MHz) with ILOG OPL 3.1/CPLEX 7.0

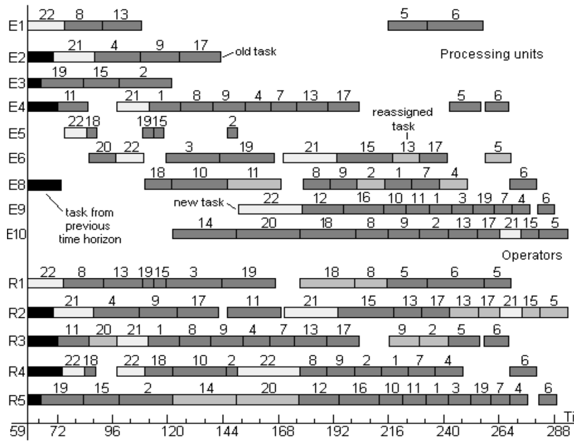

Figure 2. Updated schedule

References

ILOG OPL Studio 3.1 (2000). *User´s manual*. ILOG S.A. France

Hasebe, S., Hashimoto, I. and Ishikawa, A. (1991). *Japan Chemical Engineering*. Japan, **24**, 483

Méndez, C. A. and Cerdá J. (2001). *Proceedings of the* II *Pan American Workshop on Process Systems Engineering.* "*CEPAC'2001*", September, Guarujá, SP- Brazil.

Méndez, C. A. and Cerdá J. (2002). *Computer-Aided Chemical Engineering*, **8**, 701.

Pekny, J.F. and Reklaitis, G.V. (1998). *Proceedings of the Third International Conference on Foundations of Computer-Aided Process Operations,* 91, Snowbird, Utah.

Pinto, J. M. and Grossmann I. E. (1998). *Annals of Operations Research*, **81**, 433.

Pinto, J. M. and Grossmann I. E. (1997). *Comput. Chem. Eng.*, **21**, 801.

Roslöf, J., Harjunkoski I., Björkqvist, J., Karlsson, S. and Westelund, T. (2001). *Comput. Chem. Eng.*, **25**, 821.