A CONSTRAINT PROGRAMMING APPROACH TO THE MULTI-STAGE BATCH SCHEDULING PROBLEM

L. Zeballos and G.P. Henning
INTEC (Universidad Nacional del Litoral – CONICET)
Güemes 3450, 3000 – Santa Fe, Argentina

Abstract

A hybrid Constraint Programming (CP) formulation able to address the multi-stage batch scheduling problem is presented. This approach handles different features found in industrial environments such as finite unit ready-times, dissimilar parallel equipment in each stage, sequence-dependent changeover times, topology constraints, forbidden job-equipment assignments, etc. Moreover, it renders good computational results with a variety of objective functions. However, the most important point about the proposed approach is its ability to handle constrained situations regarding both renewable and non-renewable resources. When dealing with non-renewable resources, critical raw materials having more than one delivery date can be modeled. The consumption of raw materials can be described by two distinct patterns: (i) the whole amount is needed at beginning of the job demanding the raw material and (ii) the consumption occurs at a constant rate along the processing of the demanding task.

Keywords

Multistage scheduling, Constraint programming, Renewable and nonrenewable resource constraints

Introduction

The short-tem scheduling problem of multistage batch plants is a difficult problem in itself, that becomes much more complex when resource constraints are handled. Reklaitis (1992) made a comprehensive review of the resource constraint scheduling problem. Among the most recent contributions in this field, the works of Pinto and Grossmann (1997) and Méndez et al. (2001) can be cited.

A wide variety of solution methodologies have been employed to address scheduling problems. One that has gained increased attention in recent years and that has been successfully applied to the scheduling domain is Constraint Programming (CP), either alone or combined with MILP in hybrid formulations (Jain and Grossmann, 2001; Harjunkoski and Grossmann, 2001).

The goal of this work is to propose a new hybrid CP model for the multistage batch scheduling problem able to deal with features found in industrial environments, the most important of them being resource constraints. It is based on the use of ILOG OPL Studio (ILOG, 2002) and resorts to some special ILOG Scheduler constructs (ILOG, 2000).

Hybrid Constraint Programming Model

The model will be gradually introduced; starting with the basic model, it will latter incorporate additional features. Then, constraints for handling both, renewable and non-renewable resources will be presented and finally, different objective functions will be posed.

In the following CP model, the set I corresponds to orders to be scheduled, set J refers to processing units, L is the set of stages and K is the set of critical raw material deliveries. Moreover, *activities* and *resources* are two modeling elements of ILOG OPL Studio (2002) to be used in this formulation. *Task* and *DelivTask* are the main activities to be included in the model, the first one representing processing tasks associated to orders and the second one deliveries of raw materials that occur during the planning horizon. While a *Task* is characterized by means of duration, starting time and ending time variables

(*Task.duration*, *Task.start* and *Task.end*, respectively), a *DelivTask* is considered to be instantaneous. Moreover, use_{ij} is a binary variable standing for the assignment of order i to unit j, $x_{ii'l}$ is a binary sequencing variable representing the precedence of order i to order i' in stage l.

Resources to be handled can be of different type. Processing units and manpower are unary resources, i.e. resources that can process just one activity at a time. Unary resources having similar capabilities can be grouped into alternative resource sets. Another type of resource is the so-called reservoir (e.g. raw materials) having a capacity that can be consumed by certain activities (like processing tasks) and replenished by others (such as raw material deliveries, i.e. *DelivTask*).

Assignment, Timing, Sequencing and Precedence constraints

$$Task_{il} \ requires \ s_l \quad \forall i \in I, \forall l \in L \tag{1}$$

$$\sum_{j \in J_l} use_{ij} = 1 \ \forall i \in I, \forall l \in L \tag{2}$$

$$use_{ij} = 1 \Rightarrow Task_{il}.duration = t_{ij} \tag{3}$$
$$\forall i \in I, \forall j \in J_l, \forall l \in L$$

$$Task_{il}.end \geq \sum_{j \in J_l} use_{ij} \left[Max\left(ru_j, r_i\right) + su_j + t_{ij} \right] \tag{4}$$
$$\forall i \in I, \forall l \in L$$

$$Task_{il} \ precedes \ Task_{il'} \tag{5}$$
$$\forall i \in I, \forall l, l' \in L, Ord(l') = Ord(l) + 1$$

$$Task_{i'l}.end - t_{i'j} \geq Task_{il}.end + tt_{ii'l} + su_j - M\left(1 - x_{ii'l}\right) \tag{6}$$
$$- M\left(2 - use_{ij} - use_{i'j}\right) \ \forall i, i' \in I, i' > i, \forall j \in J_l, \forall l \in L$$

$$Task_{il}.end - t_{ij} \geq Task_{i'l}.end + tt_{i'il} + su_j - M\left(1 - x_{i'il}\right) \tag{7}$$
$$- M\left(2 - use_{i'j} - use_{ij}\right) \ \forall i, i' \in I, i' > i, \forall j \in J_l, \forall l \in L$$

Constraints (1) and (2) are assignment constraints that prescribe that each processing task l associated to an order i must be assigned to just one processing unit j. Constraint (1) resorts to the *requires* special construct, provided by ILOG Scheduler (ILOG, 1992), to enforce the assignment of $Task_{il}$ to one of the j processing units that comprise the set of alternative resources s_l. Variable use_{ij}, that captures which is the particular unit that was assigned, is needed in Eq. (3) to select the proper task duration t_{ij}. Obviously, if similar units comprise each processing stage, Eq. (3) is not required. Variable use_{ij} is also employed in Eq. (4), which sets a lower bound on the completion time of order i at stage l, due to the release time of the order r_i and the ready and setup times of the assigned unit, ru_j and su_j respectively.

Constraint (5) resorts to *precedes*, another special construct of the programming environment, to impose a proper succession among the processing tasks that comprise each order i. Eqs. (6)-(7) were drawn from Méndez et al. (2001) to handle sequence dependent changeover times $tt_{ii'l}$. This feature could have been modeled with some special elements of the language, i.e. by associating *states* to tasks and using the *transitionType* special construct, like it was done in a previous work (Zeballos and Henning, 2002). However, computational results have shown that Eqs. (6)-(7) are more efficient. Unfortunately, this comparison study cannot be presented due to lack of space.

Topology, Forbidden Assignment and Sequencing Constraints

$$ActivityHasSelectedResource\left(Task_{il}, s_l, tool_j\right) \Leftrightarrow use_{ij} = 1 \tag{8}$$
$$\forall i \in I, \forall l \in L, \forall j \in J_l$$

$$not \ ActivityHasSelectedResource\left(Task_{il}, s_l, tool_j\right) \tag{9}$$
$$\forall i \in I, \forall l \in L, \forall j \in J_l, (i, j) \in B$$

$$ActivityHasSelectedResource\left(Task_{il}, s_l, tool_j\right) \Rightarrow$$
$$not \ ActivityHasSelectedResource\left(Task_{il'}, s_{l'}, tool_{j'}\right) \tag{10}$$
$$\forall i \in I, \forall l, l' \in L, \forall j \in J_l, \forall j' \in J_{l'}, ord(l') = ord(l) + 1, j, j' \in C$$

$$ActivityHasSelectedResource\left(Task_{il}, s_l, tool_j\right) \Rightarrow x_{ii'j} = 0 \tag{11}$$
$$\forall i, i' \in I, \forall l \in L, \forall j \in J_l, (i, i') \in D$$

In order to handle topology and forbidden assignment restrictions, the *ActivityHasSelectedResource* special construct is employed. It handles alternative resources by acting as a predicate that evaluates to true when $Task_{il}$ has been assigned to $tool_j$ belonging to the set of alternative units s_l. Constraint (8) links this special construct to the variable use_{ij}. As seen in Eq. (9) it can be negated; in this particular case to prevent the assignment of unit j to order i if they belong to the set of forbidden unit-order pairs, called B. The same construct is also used in Eq. (10) to rule out the allocation of unit j' to the processing of order i at stage l', if unit j has been assigned to the processing of order i at stage l, and units j and j' belong to the set of non-connected units C. Similarly, constraint (11) easily prohibits certain order sequences (i.e., pairs i, i' belonging to the set of forbidden sequences D).

Renewable resource constraints

Equipment units are typical renewable resources and have already been modeled by constraint (1). By using the same approach, qualified manpower m can be represented as indicated in constraint (12). Once again, the *requires* special construct takes into account the overall resource availability and is in charge of freeing manpower when the task is finished.

$$Task_{il} \ requires \ m \quad \forall i \in I_m, \forall l \in L_m \tag{12}$$

Non-renewable resource constraints

Since these resources are depleted by those tasks requiring them, it is necessary to declare (i) how the consumption occurs, and also (ii) how the resource reservoir is replenished. As mentioned before, two usage patterns will be modeled. Pattern I, employing constructs provided by the language, assumes that the whole amount of the non-renewable resource is consumed at the start of the task, and Pattern II considers that consumption progresses at a constant rate along the duration of the task, as in continuous or semi-continuous tasks. For simplicity reasons, constraints to be presented in this section model the consumption of one critical raw material. However, they can be easily extended to handle several raw materials.

Raw Material Consumption Pattern I

$$DelivTask_k.end \; produces \; qd_k \; rm \;\; \forall k \in K \tag{13}$$

$$DelivTask_k.end \geq td_k \;\; \forall k \in K \tag{14}$$

Constraints (13)-(14) represent activities of raw material delivery, assumed to be instantaneous. While (13) models the amount *qd* of raw material *rm* that is allocated in each delivery *k*, constraint (14) accounts for the time at which the activity *DelivTask* takes place. As seen, the *produces* special construct is included in (13) to directly model the activity of replenishing the reservoir of *rm.*

$$Task_{il} \; consumes \; qr_i \; rm \;\; \forall i \in I_r, \forall l \in L_r \tag{15}$$

Constraint (15) resorts to the *consumes* special construct, which is the counterpart of *produces* to enforce the consumption of raw material *rm* at the start of $Task_{il}$.

Raw Material Consumption Pattern II

ILOG's Scheduler package does not provide any specific construct to handle this type of consumption, which is found quite often in the industrial practice. Before introducing the balance equation that prevents launching orders without having enough amount of the critical raw material, constraints (16)-(19) will be presented. They correspond to orders *i* demanding the critical raw material in certain stages and their purpose is to determine the status (i.e., being processed, finished) of each processing task of such orders at the time the *k*-th delivery takes place. This is achieved by means of binary variables FO_{ilk} and PO_{ilk}. The first one is equal to one if the processing of stage *l* of order *i* is finished before the time the *k*-th raw material delivery takes place. On the contrary, PO_{ilk} is equal to one if the order is being processed at such a time.

$$Task_{il}.end \;\leq td_k \Rightarrow FO_{ilk} = 1 \wedge PO_{ilk} = 0 \tag{16}$$
$$\forall\, i \in I_r, \forall l \in L_r, \forall\, k \in K$$

$$Task_{il}.end \;\geq td_k \Rightarrow FO_{ilk} = 0 \tag{17}$$
$$\forall\, i \in I_r, \forall l \in L_r, \forall\, k \in K$$

$$Task_{il}.end > td_k \wedge Task_{il}.start > td_k \Rightarrow \tag{18}$$
$$FO_{ilk} = 0 \wedge PO_{ilk} = 0 \; \forall\, i \in I_r, \forall\, k \in K, \forall l \in L_r$$

$$Task_{il}.end > td_k \wedge Task_{il}.start < td_k \Rightarrow \tag{19}$$
$$FO_{ilk} = 0 \wedge PO_{ilk} = 1 \; \forall\, i \in I_r, \forall\, k \in K, \forall l \in L_r$$

Raw Material Balance Equation for Pattern II

This is represented by constraint (20). Its left hand side accounts for the raw material availability due to deliveries, whereas the right hand side represents the raw material depletion. The first term on the right hand side considers the consumption due to orders which have already finished at the time the *k*-th delivery occurs. The second term accounts for the consumption of orders that are being manufactured by that time. Such consumption is proportional to the elapsed time.

$$\sum_{f,k\in K, f<k} qd_f \geq \sum_{i\in I_r, l\in L_r} qr_i\, FO_{ilk} + \tag{20}$$
$$\sum_{i\in I_r, l\in L_r} qr_i\, PO_{ilk} \left[\left(td_k - Task_{il}.start \right) / \, Task_{il}.duration \right] \forall k \in K$$

Objective Functions and Associated Constraints

The CP model can deal with distinct objective functions such as makespan *MK*, performance indexes related to order tardiness T_i and earliness E_i, number of tardy orders NO_i, unit utilization costs, etc.

$$Task_{il} \; precedes \; MK \quad \forall i \in I, l = last(L) \tag{21}$$

$$Min \;\; MK \tag{22}$$

If Eq. (22) is chosen as the problem objective function, constraint (21) should also be included in the model since it defines the makespan concept.

$$T_i \geq Max(0, Task_{il}.end - d_i) \quad \forall i \in I, l = last(L) \tag{23}$$

$$Min \sum_{i\in I} p_i\, T_i \tag{24}$$

Constraint (23) assigns a positive value to the tardiness T_i of order *i* if the finishing time of its last processing stage is greater than its due-date d_i. This additional equation is to be included in the model if the weighted total tardiness defined in Eq. (24) becomes the problem performance measure. Similar constraints (not shown due to lack of space) can be defined to consider the total weighted earliness as the problem objective function.

$$0 < Task_{il}.end - d_i \Rightarrow TO_i = 1 \;\; \forall i \in I, l = last(L) \tag{25}$$

$$Min \sum_{i\in I} TO_i \tag{26}$$

Constraint (25), that defines whether a task is tardy or not, is to be included in the model if the number of tardy orders is minimized, as in (26).

$$use_{il} \leq usetool_j \;\; \forall i \in I, \forall l \in L, \forall j \in J_l \tag{27}$$

$$Min \sum_{i\in I, j\in J} c_{ij}\, use_{ij} + Min \sum_{j\in J} q_j\, usetool_j \tag{28}$$

Finally, Eq. (27) allows to define the $usetool_j$ binary variable that reflects the usage of unit *j*. It is included in the model if the equipment usage cost is minimized as in constraint (28) (Harjunkoski and Grossmann, 2001).

Examples and Computational Results

Due to lack of space the application of the model is briefly illustrated by means of the case study shown in Tables 1 and 2, consisting of 5 orders that should be processed in three stages, having two dissimilar units each one. All of them but order 2, require a critical raw material in the second processing stage, which is a continuous one. This raw material is replenished at the start of the planning horizon and at two other additional times (See gray vertical lines in Fig. 1). A comparison of the solutions obtained with the two consumption patterns showed that Pattern II renders better results. For instance, if Makespan is the chosen objective function, Pattern I does not allow the start of Tasks 12 and 52 until restocking that arrives at time t = 375 takes place. As seen in Fig. 1 this does not occur for Pattern II that can start processing these tasks previously, thus reducing the makespan value in 12 units. However, Pattern II is computationally more demanding (25.1s versus 0.99s). Nevertheless, its performance improves dramatically (i.e. it goes down to 1s again) if the solution obtained with Pattern I is run before and by means of a script, such solution is employed to provide bounds on the orders' finishing time of Pattern II.

Conclusions

A new hybrid CP formulation has been developed. It can handle many features usually found in real industrial problems such as sequence dependent changeover times, finite release times, topology constraints, forbidden order-unit assignments, prohibited order sequences, etc., and the existence of limited resources of both, renewable and non-renewable type. It has been tested with a variety of case studies and has rendered good computational results for medium size examples, with several objective functions.

Acknowledgments

Authors acknowledge financial support from ANPCyT under Grant 14-070040, from CONICET and "Universidad Nacional del Litoral".

References

ILOG. (2000). ILOG Scheduler 5.0. *User's Manual*, France.

ILOG. (2002). ILOG OPL Studio 3.5. *User's Manual*, France

Jain, V. and I.E. Grossmann. (2001). Algorithms for hybrid MILP/CP models for a class of optimization problems. *INFORMS Journal of Computing*, **13**, 258-276.

Harjunkoski, I. and I.E. Grossmann. (2001). Decomposition Techniques for Multistage Scheduling Problems Using Mixed-Integer and Constraint Programming Methods, accepted in *Computers and Chemical Engineering*. http://egon.cheme.cmu.edu/Group/Papers.html

Méndez, C.A, G.P. Henning and J. Cerdá (2001). An MILP continuous-time approach to short-term scheduling of resource-constraint multistage flowshop batch facilities, *Computers and Chemical Enginnering*, **25**, 701-711.

Pinto, J.M. and I.E. Grossmann (1997). A logic-based approach to scheduling problems with resource constraints, Computers and Chemical Engineering, **21**, 801-818.

Reklaitis, G.V. (1992). Overview of scheduling and planning of batch process operations, *NATO Advanced Study Institute–Batch Process Systems Engineering*, Antalya, Turkey.

Zeballos, L. and G. P. Henning. (2002). A Constraint Programming Approach to the Single-Stage Batch Scheduling Problem with Resource Constraints, presented in *JAIIO' 02*, Santa Fe, Argentina, September 2002.

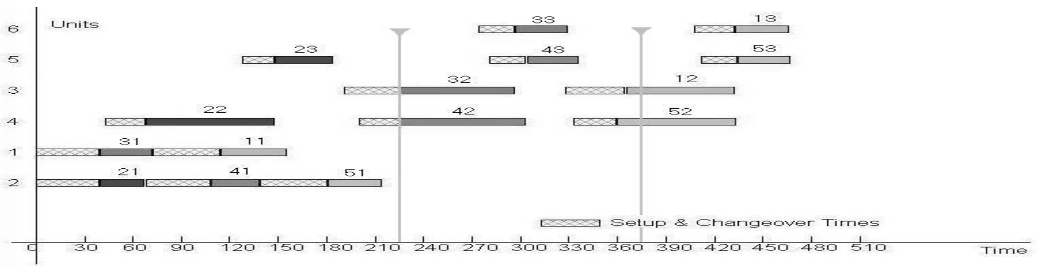
Table 1. Production orders data

Order	Release Time	Due Date	Raw material required at Stage 2	Task Duration/Cost at Unit 1	2	3	4	5	6
1	8	300	64	33/2	31/2	71/1	70/1	35/1	33/1
2	5	350		0/0	28/1	75/1	80/1	37/1	36/1
3	6	400	75	41/1	35/1	68/1	75/1	40/1	34/1
4	10	400	71	42/2	30/1	73/1	78/1	32/1	30/1
5	6	350	67	30/1	33/1	70/1	74/1	33/1	35/1
		Setup		40	40	35	25	20	23

Table 2. Sequence dependent changeover times

Order	Stage 1					Stage 2					Stage 3				
	1	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
1	1	3	2	1	1	1	3	2	4	1	5	3	2	2	1
2	3	2	1	1	2	3	4	3	1	2	3	2	1	3	2
3	2	2	1	1	3	2	2	1	1	3	2	3	1	1	1
4	3	1	1	3	2	1	2	1	4	2	2	1	2	3	2
5	3	2	1	3	1	3	4	3	2	3	4	5	1	3	1

Figure 1. Gantt Diagram corresponding to consumption Pattern II showing raw material deliveries.


Units
Setup & Changeover Times
Time