10th International Symposium on Process Systems Engineering - PSE2009
Rita Maria de Brito Alves, Claudio Augusto Oller do Nascimento and Evaristo Chalbaud Biscaia Jr. (Editors)

A Reactive Scheduling Approach Based on Domain-Knowledge

Juan M. Novas, Gabriela P. Henning

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

Abstract

Real world industrial environments frequently face unexpected events that generally disrupt in-progress production schedules. This contribution presents advances in the development of a support framework to address the repair-based reactive scheduling of industrial batch plants. When facing an unforeseen event, the framework is capable of capturing the current operational plan and its status. Based on this information, a rescheduling problem specification is developed. Tasks to be rescheduled are identified and, for them, the set of the most suitable rescheduling action types (e.g. shift, reassign, etc.) is specified. For a given specification, many solutions to the problem could exist. Then, the second step of this approach relies on the generation of a constraint programming (CP) model to address the rescheduling problem just specified. To create such model each rescheduling action type is automatically transformed into different constraints. In addition, a search strategy based on domain knowledge can also be developed. Finally, the solution of the CP model and its associated search strategy will render the repaired schedule in which the repair action types that were suggested will be instantiated. A case study of a multiproduct multistage batch plant is presented, where an events of unit failure is considered.

Keywords: Reactive Scheduling, Decision Support Systems, Batch Plants.

1. Introduction

Usually, predictive scheduling techniques generate production plans which assume stationary operating conditions along the whole scheduling horizon. However, real industrial environments are dynamic in nature; unforeseen events disrupt frequently the in-progress schedule. Reactive scheduling, or rescheduling, is then performed in order to update the current agenda. This task can be periodically executed or when an unexpected event, such as a unit breakdown, order cancellation or arrival, occurs, leading to periodic versus event-driven rescheduling, respectively. To be adopted, reactive scheduling systems must provide immediate responses to disruptions. Besides, minimum changes to the original schedule are desired to maintain a smooth plant operation. Hence, repair-based or partial rescheduling is preferred to a full-scale one.
Several works about reactive scheduling of batch plants have been reported. Janak et al. (2006) presented a review of approaches tackling this kind of problem and other papers in the field have recently been published (Ferrer-Nadal et al., 2007). Even though the review of Vieira et al. (2003) focused on discrete manufacturing plants, most of the issues described in it can be found in process industries too.
This contribution tackles the reactive scheduling problem by developing a support environment based on both, an explicit representation of the domain knowledge and a CP approach. The next section presents an overview which addresses these two components. Finally, Section 3 discusses an illustrative example.

2. Reactive Support Environment

When addressing a rescheduling situation most of the objectives and basic constraints that define the original problem still apply; however, the partially executed schedule and the perturbation or triggering event has also to be taken into account. This contribution presents a support framework able to represent this knowledge and use it in the development of a solution. It has been envisioned to operate under an event-driven policy. The framework explicitly captures the status of an in-progress schedule and characterizes the disrupting event, in order to specify the rescheduling problem to be faced and, afterwards, solve it. The environment integrates different modules, as it is shown in Fig.1, all based on an explicit domain representation.
As the first step of the proposed approach, tasks to be rescheduled are identified, and for each of them, the most suitable rescheduling action type is specified. An action type not only prescribes what to do (e.g. shift, reassign, etc.) on a processing task, but also a range of possibilities, i.e. feasible equipment for a reassignment, shifting interval, etc. For a given specification, corresponding to the set of action types associated to the tasks to be rescheduled, many solutions could exist. Then, the second step of this approach relies on the generation of a constraint programming (CP) model to address the rescheduling problem just specified. To create such model each rescheduling action type is automatically transformed into different constraints. A search strategy based on domain knowledge can be employed, too. Finally, the solution of the CP model and its associated search strategy will render the repaired schedule, in which the proposed repair action types will be instantiated.

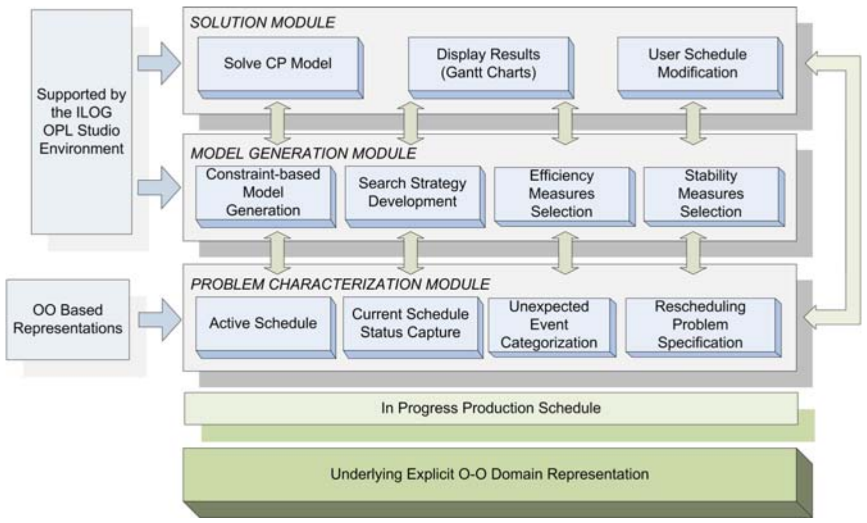
SOLUTION MODULE
Solve CP Model
Display Results (Gantt Charts)
User Schedule Modification
Supported by the ILOG OPL Studio Environment
MODEL GENERATION MODULE
Constraint-based Model Generation
Search Strategy Development
Efficiency Measures Selection
Stability Measures Selection
PROBLEM CHARACTERIZATION MODULE
OO Based Representations
Active Schedule
Current Schedule Status Capture
Unexpected Event Categorization
Rescheduling Problem Specification
In Progress Production Schedule
Underlying Explicit O-O Domain Representation

Fig.1. Components of the Reactive Scheduling Support Framework

2.1. Domain Knowledge Representation and Problem Characterization

Scheduling is a knowledge intensive activity in terms of domain knowledge. The representation of this type of knowledge is a critical issue in any scheduling support system. Fig. 2 presents a simplified version of the conceptual model of multiproduct batch plants scheduling knowledge included in the proposed framework. This information is organized into different conceptual levels. Generic concepts represent engineering or plant information that is used by entities which model a given scheduling scenario. Because of the fact that scheduling is an evolving activity, different versions of a schedule need to be captured and related among themselves. A schedule version is a snapshot of an active schedule, which is valid at a given moment, and is represented by means of scheduling domain entity versions (i.e., versions of the instances of *Task*, *ResourceSchedule*, *Schedule*, etc.). The handling of versions is not discussed here due to space limitations.

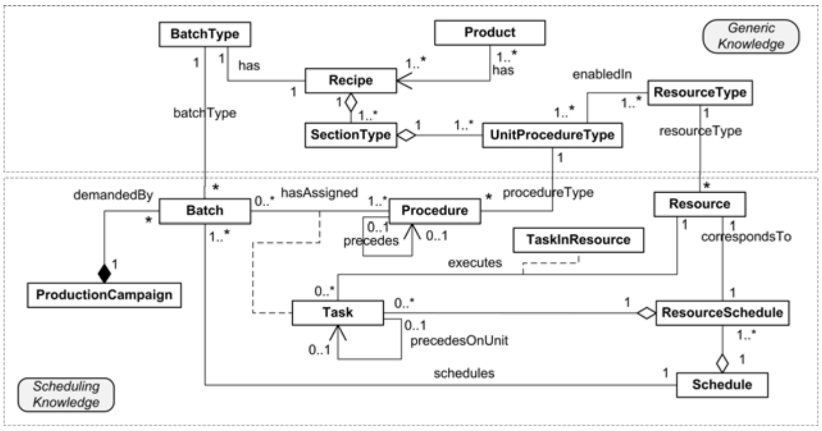

Fig.2. Partial view of batch scheduling domain concepts

Since the domain knowledge is explicitly captured, the problem characterization module can fulfill the following functional requirements that were imposed when designing the framework.

- *Capture the current operational plan and its status at the rescheduling point.* Instances of entities represented at the scheduling knowledge level, their relationships, as well as the values of their attributes, allow modeling each problem to be tacked. When addressing a rescheduling situation, the status of each task involved in the disrupted schedule must be known. At a rescheduling point *t*, a decision-making activity about tasks to be included in the reactive process, has to be made. Hence, all the tasks involved in the disrupted agenda, T , are classified in the following sets, depending on their status at time point *t*: (i) already executed tasks, T_t^{AE}, (ii) non-executed tasks, T_t^{NE}, and (iii) in-progress tasks, T_t^{IP}. In addition, if the disrupting event is a batch cancellation, its associated tasks are included in the set T_t^{CE}. In turn, new tasks to be considered due to a rush batch arrival, belong to the set T_t^{NW}. These sets are related in the following way: $T=\{T_t^{AE}\cup T_t^{IP}\cup T_t^{NE}\}$; $T_t^{CE}\subseteq T_t^{NE}$; $T_t^{NW}\not\subset T_t$. Batch and equipment status are also domain information which is taken into account in the rescheduling process. Based on the previous classification, tasks to be involved in the rescheduling process are identified. Thus, RT_t is the set of tasks that needs to be considered due to the unexpected event taking place at time *t*. The remaining tasks are those not involved in the reactive process, referred as NT_t.
- *Identification of the disruptive event type and its specific characteristics.* The categorization and modeling of events is also another critical aspect of the problem.

The problem characterization module uses information from the object instances pertaining to the sets previously described and from the entity representing the *UnexpectedEvent* in order to determine: (i) the updated equipment ready times, (ii) the earliest start times of those tasks that pertain to in-progress batches. This information is also employed to properly specify the reactive scheduling problem, as discussed below.

2.1.1. Rescheduling Problem Specification

For each task belonging to RT_t its associated rescheduling action type has to be decided. By means of the proposed methodology, each task does not have a specific and pre-defined rescheduling operation related to it; instead, it has a type of rescheduling action. Each action type prescribes a category of rescheduling operation and a range of possibilities to apply it. In this framework, the following types have been considered up to now: *a) Shift*, which simply "pushes" the task start time forward (right-shift) or backwards (left-shift) on the current unit. To apply such action on a task, or set of tasks, allowed left shift (*lsv*) and right shift values (*rsv*) are defined to specify a time window for the new start time. *b) Reassign*, which allows tasks to be allocated to a unit belonging to a set of feasible ones. *c) Freeze,* which is not a scheduling action itself, but is required for tasks that must keep their timings as well as equipment assignments.

In order to define the rescheduling action type to be associated with each task, they are previously classified into: (i) directly affected ones, comprising set RT_t^{DA}, (ii) indirectly affected tasks, represented by set RT_t^{IA}, and (iii) non-affected tasks, which are included in set RT_t^{NA}. The way tasks are categorized depends on the event type and to the extend tasks are disrupted by it. Directly and indirectly affected tasks are linked to specific revision actions, which depend on the event type. On the contrary, action types associated with non-affected tasks are defined employing domain knowledge of each specific problem. To illustrate these ideas, sets RT_t^{DA}, RT_t^{IA} and RT_t^{NA} are defined for a unit failure event and their associated rescheduling actions are discussed. Similar definitions can be made for a batch cancellation (not shown due to lack of space) or other disruptions.

Task Classification for a Unit Failure Event

RT_t^{DA}. Tasks in T_t^{NE}, are the ones assigned to the broken-down unit, which start later than *t* and earlier than the unit recovery time. Besides, if a task is interrupted by the unit failure and cannot be recovered, its batch is canceled and all the associated tasks are inserted as a new batch (*NewBatchArrival* event) to be included in the rescheduling process.

RT_t^{IA}. These indirectly affected tasks are linked with, at least, one of the directly affected activities. They are the ones located downstream in those batches associated with the directly disrupted tasks. Activities located upstream can be considered as indirectly affected too (as seen in the example presented in the next section).

RT_t^{NA}. Includes tasks which are not affected, neither directly nor indirectly, by the unit breakdown.

Rescheduling Action Types for Tasks Associated with a Unit Breakdown

Tasks in RT_t^{DA} are associated with a reassign action type that forces them to find another processing unit. Similarly, members of RT_t^{IA} are associated with a reassign action type that allows them to retain their current unit allocations (with a possible change in timings) or to choose different ones. Finally, different criteria were established to define the rescheduling action types associated with those tasks included in set RT_t^{NA}. In this contribution, both a scheduling horizon-based (*SHB*) and a batch information-based (*BIB*) criteria are briefly discussed.

SHB criterion: It takes into account the moment the disruptive event takes place in relation to the length of the whole scheduling horizon. Depending on the scheduling scenario and the costs of changing the current task allocations, it can be decided to: (i) freeze tasks, belonging to set RT_t^{NA}, whose current start times are very close to the

rescheduling point, (ii) enable shift movements over those tasks from set RT_t^{NA} having a start time not as close to the rescheduling point, but not located at the end of the horizon, (iii) enable reassign action types over those tasks placed at the end of the scheduling horizon. A parameter V is employed to identify these intervals.
BIB criterion: The status of the current schedule information is employed to define the rescheduling action types over those tasks in set RT_t^{NA}. For instance, batches having a high processing progress at the rescheduling point are not allowed to have their non-executed tasks reassigned, (ii) tasks associated with batches having small positive or negative values of their slack times are allowed to be reallocated to avoid violating their due dates.
In this way, a rescheduling problem specification gives rise to a set of rescheduling alternatives to every task, and many solutions to the problem could exist. Then, the second step of this approach relies on the generation of a constraint programming (CP) model to address the rescheduling problem just specified. To create such model each rescheduling action type is automatically transformed into different constraints.

2.2. Model Generation

In order to create a repaired schedule, the model generation module is in charge of setting up a constraint programming (CP) model, complying with the syntax of the OPL language, the underlying language of ILOG OPL Studio (ILOG, 2002). This model (not shown due to space limitations) comprises: (i) the set of basic constraints (assignment, precedence, timing constraints, etc.), which are part of any batch scheduling problem, and are generated for all tasks in RT_t and (ii) the set of event-dependent constraints. These last ones depend on the disrupting situation been faced, and on the adopted action types for tasks included in sets RT_t^{DA}, RT_t^{IA} and RT_t^{NA}.

3. Example

This proposal was tested with various case studies. One of them, introduced by Pinto and Grossmann (1997), is presented in this section. The plant involves 4 processing stages and 10 units. During the scheduling horizon, 20 orders have to be processed. Fig. 3.a shows an in-progress schedule, with tasks labeled by batch name. Let us assume that unit *u4* shuts down at t = 31.00 h and it will be unavailable till a *recovery time* = 134 h.
The problem was solved employing three different approaches in order to compare results: by resorting to a full-scale rescheduling (*FSR*) and to two partial ones. In the first case all tasks involved in the rescheduling process were allowed to be reassigned. The second approach was a partial reschedule, identified as *PR_1*, which employed the *SHB* criterion to classify tasks (see Fig. 3.a). Tasks in sets RT_t^{DA} and RT_t^{IA} were associated with a reassign rescheduling action type. For tasks in set RT_t^{NA}, a value of 24 h was given to parameter *V*, meaning that tasks having their planned start time within the rescheduling point, and 24 h after it, were associated with freeze action type. Tasks in RT_t^{NA}, starting at least 24 h after the rescheduling point, were associated with a shift action type. The new schedule, obtained under these conditions, which is not shown due to lack of space, was not a good quality one due to the topological constraints that had to be considered. Because of them, batches assigned to units *u1* and *u2* in the first stage, which were in set RT_t^{NA}, could not be assigned to unit *u5* in the second stage and had to wait for the recovery of *u4*. On the contrary, the other partial reschedule approach (*PR_2*) employed topological knowledge. The criterion used was also *SHB*, but now upstream tasks were considered in RT_t^{IA}, too. So, in order to give more flexibility to tasks

assigned to units *u1* and *u2*, they were associated with reassign action types. The solution that was obtained is shown in Fig. 3.b. This new schedule is of better quality than those obtained with the *PR_1* and *FSR* strategies. Schedules were assessed by means of a makespan performance measure, plus equipment and temporal stability functions that allow quantifying a smooth plant operation, among others.

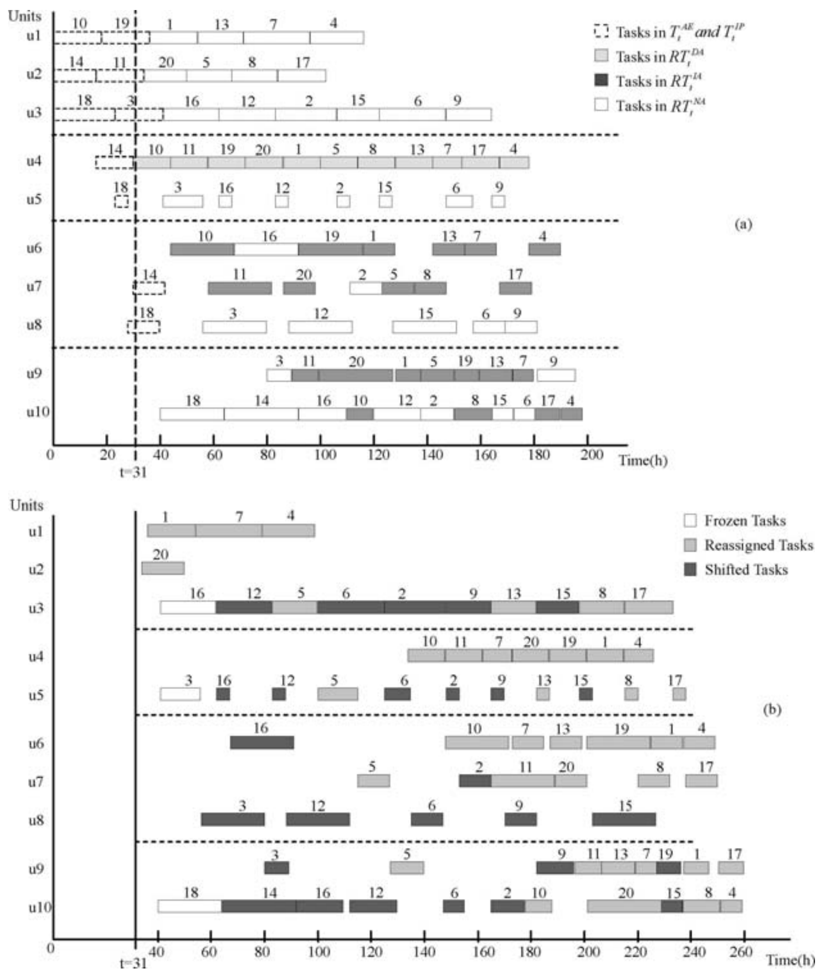

Fig.3. a) In-progress schedule (UIS policy). Classification of tasks for *PR-1* is depicted with grey shades. b) New schedule obtained after applying the *PR_2* approach.

References

S. Ferrer-Nadal, C. A. Méndez, M. Graells, L. Puigjaner, 2007, Optimal Reactive Scheduling of Manufacturing Plants with Flexible Bath Recipes, Ind. Eng. Chem. Res., 46, 6273-6283.

ILOG: ILOG OPL Studio 3.5. User's Manual, France, 2002.

S.A. Janak, C.A. Floudas, J. Kallrath, N. Vormbrock, 2006, Production Scheduling of a Large-Scale Industrial Batch Plant. II. Reactive Scheduling. Ind. Eng. Chem. Res., 45, 8253-8269.

J. M. Pinto, I. E. Grossmann, 1997, A logic-based approach to scheduling problems with resource constraints, Comp. Chem. Eng., 21, 801-818.

G. Vieira, J. Herrmann, E. Lin, 2003, Rescheduling Manufacturing Systems: A Framework of Strategies, Policies, and Methods, J. Sched., 6, 39-62.