Dynamic hybrid simulation of batch processes driven by a scheduling module

Florian Fabre[a], Gilles Hétreux[a,*], Jean-Marc Le Lann[a], Pascale Zaraté[b]

[a] *Laboratoire de Génie Chimique (PSI–GI), UMR-CNRS 5503/INPT-ENSIACET, 4, allée Emile Monso, 31030 Toulouse, Cedex 4, France*
[b] *IRIT/INPT-ENSIACET, 118 Route de Narbonne, 31062 Toulouse, Cedex 9, France*

Keywords:
Dynamic hybrid simulation Environment
Batch processes
Scheduling
Petri nets
Mixed Integer Linear Programming
Object oriented modeling

A B S T R A C T

Simulation is now a *CAPE* tool widely used by practicing engineers for process design and control. In particular, it allows various offline analyses to improve system performance such as productivity, energy efficiency, waste reduction, etc. In this framework, we have developed the dynamic hybrid simulation environment *PrODHyS* whose particularity is to provide general and reusable object-oriented components dedicated to the modeling of devices and operations found in chemical processes. Unlike continuous processes, the dynamic simulation of batch processes requires the execution of *control recipes* to achieve a set of production orders. For these reasons, *PrODHyS* is coupled to a scheduling module (*ProSched*) based on a *MILP* mathematical model in order to initialize various operational parameters and to ensure a proper completion of the simulation. This paper focuses on the procedure used to generate the simulation model corresponding to the realization of a scenario described through a particular scheduling.

1. Introduction

For several decades, processing and recovery of raw materials has caused a tremendous expansion of industrial chemistry. If the units in this sector traditionally operate continuously, food, biotechnology, pharmaceutical or electronics industries are functioning preferentially in a batch mode. Indeed, located on markets subject to high turnover of products and fluctuating or unpredictable demand, batch processes are characterized by these qualities of flexibility. Generally used to manufacture high added value products, profits made so far were such that it seemed somewhat interesting to develop tools and methodologies to improve the performance of these units. But the internationalization of markets and the growing needs of society have led to new industrial strategies. Located in highly competitive markets, the function of the process is then complicated by a desire to consolidate production facilities and reduce costs. These new constraints are reflected today by an undeniable interest of industrial and scientific community to better design and more importantly, to better exploit these batch processes.

Among the available *CAPE* tools (*Computer Aided Process Engineering*), process engineers are showing a growing interest in *dynamic simulation* for its ability to carry out various analyses (configurations, operating policies, etc.) on a "*virtual*" plant, extremely useful to process engineers in their daily work to improve system performance (productivity, energy efficiency, waste reduction, etc.). During the development of a new process, mass and energy balances, equipments sizing, utilities needs assessment, estimation of time cycle or cost analysis are generally performed and these tools can significantly reduce design variance and the laboratory work on pilot which is often costly and time consuming. In operation, having a reliable simulation model improves understanding of the whole process by the operators and facilitates communication. Production engineers can assess in a few minutes the impact of critical parameters on key indicators such as production costs, time cycle, energy efficiency or productivity. The simulation also provides the means to monitor the occupancy of all tanks during a campaign and verify that the minimum and maximum loads are always met in all parts of the process. It can validate operating conditions of each task and sets the control loops required to maintain these operating conditions. Finally, in a safety point of view, the impact of defaults in the operative or command part can be quickly estimated by simulation and corrective actions can be tested.

In this context, the unification of research in modeling and simulation of processes carried out for many years in the *LGC* has led to the development of *PrODHyS* (Fabre, 2009; Hétreux, Théry, Perret, Lelann, & Joulia, 2002; Jourda, Joulia, & Koehret, 1996; Moyse, 2000; Olivier-Maget, 2007; Perret, 2003; Sargousse, 1999), a dynamic hybrid simulation environment dedicated to chemical processes (Fig. 1).

Based on *object concepts*, this environment offers extensible and reusable software components allowing a rigorous and systematic modeling of the topology and the behavior of processes. The hybrid feature is managed with the *Object Differential Petri Nets* (*ODPN*)

* Corresponding author. Tel.: +33 5 62 88 36 60.
E-mail address: gilles.hetreux@ensiacet.fr (G. Hétreux).

doi:10.1016/j.compchemeng.2011.04.007

Nomenclature

Indices

i	processing tasks
st	storage tasks
j	units
n	event points representing the beginning of a task
s	states

Sets

I	set of processing tasks i
ST	set of storage tasks st
ST_s	set of storage tasks st for state s
I_j	tasks i that can be performed in unit j
I_s	set of tasks i that use state s
$I_s{}^{\mathrm{p}}$	set of tasks i that produce state s
$I_s{}^{\mathrm{c}}$	set of tasks i that consume state s
J	set of units j
J_i	set of units j that are suitable for performing task i
N	set of event points n within the time horizon
S	set of states s
S^f	states with finite intermediate storage
S^n	states with no intermediate storage
S^p	states that are final products
S^r	states that are raw materials
S^z	states with zero-wait policy

Parameters

$V_i{}^{\min}$, $V_i{}^{\max}$	minimum and maximum capacity for task i
$C_s{}^{\max}$	maximum amount of state s that can be stored
$\rho^{\mathrm{p}}{}_{i,s}/\rho^{\mathrm{c}}{}_{i,s}$	proportion of state s produced or consumed by task i
pf_i	fixed part of the processing time of task i
pv_i	variable part of the processing time of task i
H	maximum duration of the campaign → time horizon
h_s	storage cost of state s
D_s	amount of state s delivered at the end of the campaign

Variables

$B_{i,n}$	amount of material undertaking by task i at event point n
$Bs_{i,n}$	amount of material starting processing by task i at event point n
$Bf_{i,n}$	amount of material finishing processing by task i at event point n
$Bst_{st,n}$	amount of material stored by storage task st at event point n
$S_{s,n}$	amount of state s at event point n
SF_s	final amount of state s at the end of the time horizon
$S0_s$	initial amount of state s at the beginning of the time horizon
$Ts_{i,n}$	time at which task i starts at event point n
$Tf_{i,n}$	time at which task i finishes at event point n
$pt_{i,n}$	processing time of task i at event point n
$Tsst_{st,n}$	time at which storage tasks st starts at event point n
$Tfst_{st,n}$	time at witch storage tasks st finishes at event point n
$W_{i,n}$	1 if task i is activated at event point n, else 0
$Ws_{i,n}$	1 if the task i begins at event point n else 0 (binary variable)
$Wf_{i,n}$	1 if the task i ends at event point n else 0 (binary variable)
$Plan$	duration of the production scheduling

Table 1
Batch management with optimization or dynamic simulation approach.

	Optimisation method	Dynamic simulation
Advantages	Exhaustive exploration of candidate solutions Global consideration of all the constraints Efficient solving method	More realistic modeling of processes Processing times determined by phenomenological models
Drawbacks	Modeling often based on simplifying assumptions, which do not permit to exploit the entire flexibility of the process Fixed and often overestimated processing times	Evaluation of a candidate solution $\Leftrightarrow$ simulation of the process for a given sequence and a given batch sizes ↳ limited exploration of candidate solutions Myopic view $\Leftrightarrow$ difficulties to take into account time constraints (*no-wait*, *conditioning calendar*, *cleaning policy*)

formalism. It combines in the same structure, a set of differential and algebraic equations (*DAE*) systems which describe the continuous evolution of the system (primarily based on the thermodynamic and physicochemical laws) and high level Petri nets which define the legal commutation sequences between states (i.e. one of the possible configurations of *DAE* systems).

Nevertheless, in opposite to continuous processes, studies on batch units often necessitate to take into account both the physicochemical phenomena that take place in each device (*local vision*) and the management of batches (nature, size, number and starting date) passing through the unit (*global vision*). Obviously, these two features have a significant impact on the performances and induce that the system has to be tackled as a whole to establish a consistent analysis. In this context, the simulator must be able to run a scenario described by a production plan including production orders (*PO*), each *PO* indicating among other things, the type of product, the quantity to be produced and the period of execution (starting and ending date of the jobs).

To achieve the production plan and meet the various constraints, a temporal and quantitative synchronization must be ensured. But, the management of batches only by simulation does not always give satisfactory results and may even lead to abort an execution. First, in order to take into account the capacity of equipment, it is often necessary to split production orders into several batches. The number and size of theses batches have to be calculated. In addition, the myopic view of the simulation prevents a proper handling of time constraints (delivery dates, zero-wait policy, maximum delay, etc.), resource allocation constraints or cleaning constraints. In many dynamic simulators, these calculations are either assumed by the user, either based on heuristics or simple priority rules. So, in order to tackle rigorously each part of the problem and improve the solutions, the strategy adopted in *PrODHyS* consists in driving the simulation by following a production scenario obtained from a scheduling module based on optimization techniques. Table 1 summarized the main characteristics of these two kinds of tool.

The purpose of this paper is to present the tools and methodologies used to implement the interface between this scheduling module and the simulation model. The rest of the paper is organized as follow. In Section 2, the problem statement and the principle of the proposed approach are described. Each module of this tool is then described in the following order. In Section 3, the *ERTN* graphical formalism is briefly presented and illustrated. Section 4 describes the implemented mathematical formulation. Section 5 presents the major concepts on which the dynamic hybrid simulator is based. Finally, Section 6 deals with the command level of the simulation model and it highlights the necessity of coupling

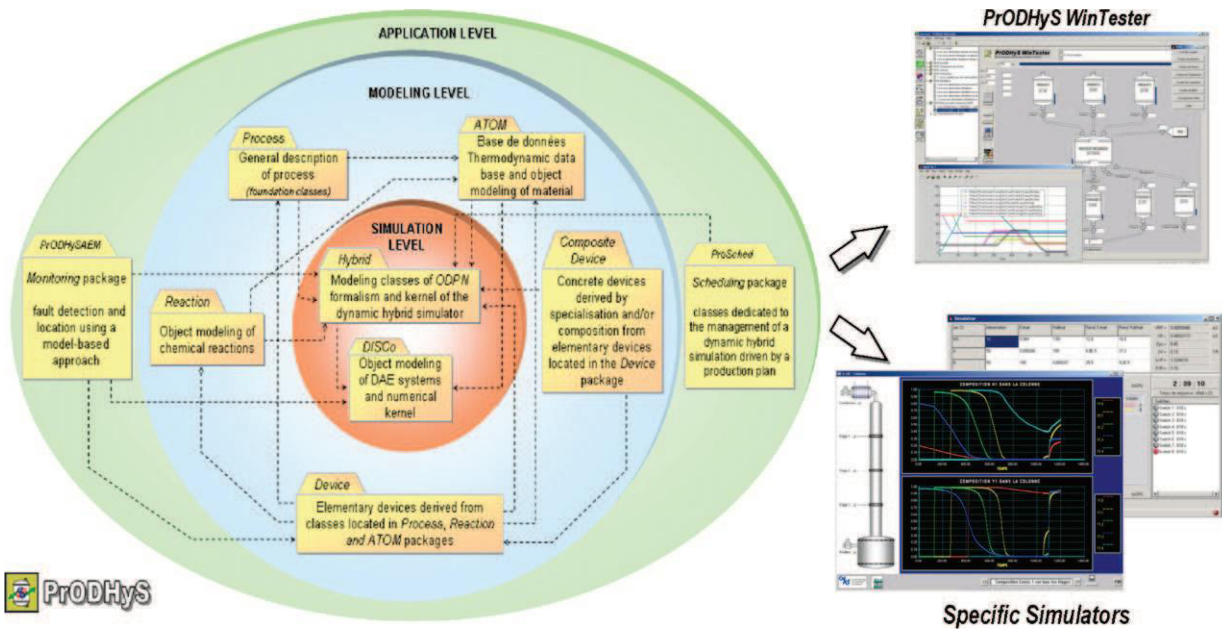

APPLICATION LEVEL
MODELING LEVEL
SIMULATION LEVEL
Process
General description of process (foundation classes)
ATOM
Base de données Thermodynamic data base and object modeling of material
ProDHySAEM
Monitoring package
fault detection and location using a model-based approach
Reaction
Object modeling of chemical reactions
Hybrid
Modeling classes of ODPN formalism and kernel of the dynamic hybrid simulator
DISCo
Object modeling of DAE systems and numerical kernel
Composite Device
Concrete devices derived by specialisation and/or composition from elementary devices located in the Device package
ProSched
Scheduling package
classes dedicated to the management of a dynamic hybrid simulation driven by a production plan
Device
Elementary devices derived from classes located in Process, Reaction and ATOM packages
PrODHyS
PrODHyS WinTester
Specific Simulators

Fig. 1. Packages of the dynamic hybrid simulation environment *PrODHyS*.

the simulator with a scheduling module. The interface with the simulator is then presented and some remarks are discussed.

2. Main steps of the recipe-driven dynamic hybrid simulation

2.1. General features of the considered batch processes

The addressed processes are *multi-purpose batch or semi-continuous plant*. In this kind of unit, each product follows a specific sequence of operations and is produced using shared processing equipment. These *general network processes* correspond to the more general case in which batches can be merged and/or split. This feature induces that material balances must be taken into account explicitly (in opposite to *sequential processes* that are order- or batch-oriented and do not require the consideration of mass balances). Consequently, the corresponding simulation models have to incorporate several general characteristics that include:

- disjunctive (devices, operators, etc.) and cumulative (materials, utilities, etc.) resources constraints,
- various storage and transfer policies (*UIS*: Unlimited Intermediate Storage, *FIS*: Finite Intermediate Storage, *NIS*: No Intermediate Storage, *ZW*: Zero-Wait, etc.)
- fixed and/or dependent processing times (depending on batch size),
- mixing and splitting of batches, inducing variable batch size along the production

2.2. Modeling of recipes

Recipe is an entity that describes the *formulation* (set of chemical substances and proportions), the *procedure* (set of physical steps required to make the product) and the required *equipment*. To tackle complex processes, the standard *ISA/SP88* (www.isa.org) has specified a hierarchical model including 4 levels (Fig. 2), each one providing information in an appropriate granularity:

- *Generic (or general) recipe* specifies the manufacturing method of each finished product. It contains information about the materials (raw materials and intermediates), proportions, operating parameters, but, no data about the equipment of the production system is provided.
- *Site recipe* is an instantiation of the *generic recipe* in which the details about the production site are incorporated. This involves the general topology of the process and clear definitions of the characteristics of the processing equipment (capacity, energy consumption, etc.).

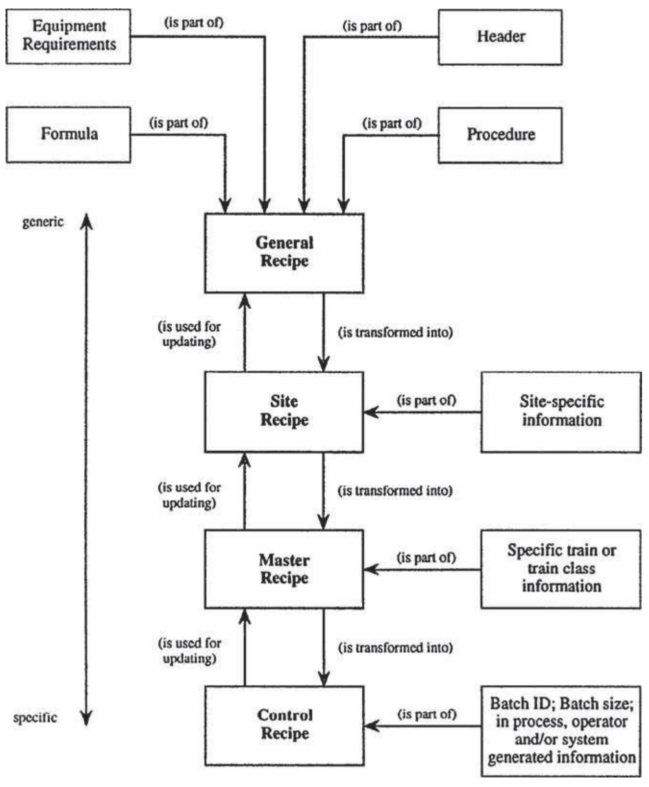


Fig. 2. Hierarchical modeling of the recipe.

- *Master recipe* is an instantiation of the *site recipe* that sets the type and amount of finished product(s) to be produced in a given operational horizon. It therefore clarifies the production orders to be achieved. This level of recipe makes use of scheduling, which calculates the number and size of each batch as well as the sequence of these batches on equipment.
- *Control recipe* is applied to a particular batch or lot and describes the execution of each task in detail. It is implemented at a supervision level.

2.3. *Graphical framework for the modeling of recipes*

To facilitate the modeling phase by non-expert users in optimization and simulation, a way is to build optimization and simulation models which are structurally generic and configurable with parameters entered through a well-defined graphical formalism. Indeed, the implementation and the tuning of a *MILP* or simulation model can become rather technical and complex in some cases. Thus, the support of a visual representation can be very helpful for the modeling of the production system. Provided that the semantic of this graphical formalism is sufficiently general, it allows the user to describe a problem in a simple and intuitive way while ignoring the mathematical support useful to its resolution. Another advantage of such formalism is the ability to unambiguously model a problem by adding specific construction rules. It reduces (but it does not avoid) potential modeling mistakes and users can spend more time in analysing the system rather than developing the model.

In this framework, the *Extended Resource Task Network* (*ERTN*) formalism has been developed for the modeling of recipes. Based on the well-known *Resource Task Network* (*RTN*) formalism proposed in (Agha, 2009; Fabre, 2009; Pantelides, 1994) have introduced new semantic elements (see Section 3) notably among others, in order to handle explicitly cumulative resources (such as utilities for example) and multi-modal resources. In these works, the *ERTN* formalism is more precisely used to model the *procedure* part of the *site recipe*. It is constructed from the *procedure* of the *generic recipe* and the topology of the unit chosen to execute this recipe.

2.4. *Main steps of the dynamic simulation procedure*

Fig. 3 shows a schematic diagram of the procedure implemented to run a dynamic simulation of a complete process for a given production campaign in *PrODHyS*.

Given the *generic recipe* of the manufactured products and the topology of the unit, the *procedure* of the *site recipe* is modeled in our tool using the *ERTN* (*Extended Resource Task Network*) graphical formalism.

To manage overall flows passing through the unit, a "*simplified*" but structurally generic scheduling model based on a *MILP* formulation is set and instantiated with data provided through the *ERTN* view (set of estimated parameters for duration, capacity of devices according to the stored material, etc.). Thus, given a *time horizon* and a *production plan* (obtained by a *MRP* procedure for example), the package *ProSched* calculates a scheduling by calling the commercial solver *XPRESS-MP* for a given computation effort. This treatment gives rise to the *master recipe* and the resulting list of tasks can be depicted on a *Gantt chart*. Data characterizing each task are transmitted via a file to the dynamic simulator *PrODHyS* in order to parameterize the *command level* of the simulation model (i.e. the *control recipe*), previously constructed in accordance to the *ERTN* view by assembling predefined *operation* objects. The *process level* of the simulation model is built according to the topology of the unit with *device* or *composite device* objects. The simulation of this "detailed" model is then executed until the completion of the production plan.

A normally ended simulation indicates that all capacity and time constraints are met. The production plan is validated and the analysis of the operational and physicochemical properties can be made. If a simulation fails then it means that constraints are violated and the user has to analyze the simulation results (via the evaluation of various indicators) to undertake corrective actions. Nevertheless, according to the objective of the study, some parameters of the (simplified and/or detailed) model may be modified or refined by exploiting the simulation of the previous iteration. This procedure can be restarted until the user finds satisfactory results.

In summary, the main idea of this combined approach is to take advantage of the strengths of dynamic simulation and mathematical programming to achieve a consistent batch management in the workshop and thus, to enhance the achievement of the dynamic simulation.

3. The *ERTN* graphical formalism

3.1. *Brief description of the ERTN formalism*

The expressive quality of formalism is judged by its aptitude to summarize on a single graph the information necessary to represent a process. In this context, *State Task Network* (*STN*) proposed by (Kondili, Pantelides, & et Sargent, 1993) has been a first step toward developing a universal representation for a batch plant. Later, (Pantelides, 1994) has proposed the *Resource Task Network* (*RTN*) formalism, an extension of the *STN* that contains more information about processing equipment and their connectivity. Based upon the major concepts of the well-know *RTN* formalism (Agha, 2009; Fabre, 2009) have introduced new semantic elements and the resulting framework is called *Extended Resource Task Network* (*ERTN*). Thus, this graph represents the main features encountered in batch processes. The underlying semantic elements are listed in Fig. 4. Accompanied by well-established construction rules, it clearly and unambiguously represents production procedures (precedence constraints), materials and energy flows (ratio of inlet and outlet flows, free flows, recycling, separation and mixing of batches) and resource constraints (topology of unit, capacity of devices, fixed or dependent operating time, shared and multimodal devices, etc.). The generic nature of the *ERTN* formalism offers a direct correspondence between the graphical elements and mathematical constraints. So, several formulations can be associated with the *ERTN* framework.

3.2. *Example of batch process modeling*

To illustrate a subpart of the *ERTN* semantic, a typical batch process is presented. In this example, the production of two final products is considered. Product **P1** necessitates three successive operations: a preheating of reactant **A**, next a reaction (*reaction 1*: $\mathbf{A}+\mathbf{B} \rightarrow \mathbf{IntAB}$) and finally, a distillation to separate final product **P1** and residue **P2**. If we suppose that intermediate **IntAB** already exists, the recipe of product **P3** is composed of two operations: a preheating of reactant **C**, followed by a reaction (*reaction 2*: $\mathbf{C}+\mathbf{IntAB} \rightarrow \mathbf{P3}$).

The topology of the unit is shown in Fig. 5. It consists of a *preheater/mixer*, two reactors (called *REACTOR 1* and *REACTOR 2*), a *column* (ensures the separation of reaction products) and several *storage tanks* (for raw materials **A**, **B**, **C**, intermediate product **IntAB**, residue product **P2** and final products **P1**, **P3**). To control these devices, the unit is equipped with several actuators (pumps P*i*, valves V*i*, heating systems Q*i*, electric motors M*i*) and sensors (retention U*i*, temperature T*i*, composition XP*i*, flow F*i*). **REACTION1** can be performed indifferently in the two reactors while **REACTION2** can be performed only in *REACTOR 2*. In addition,

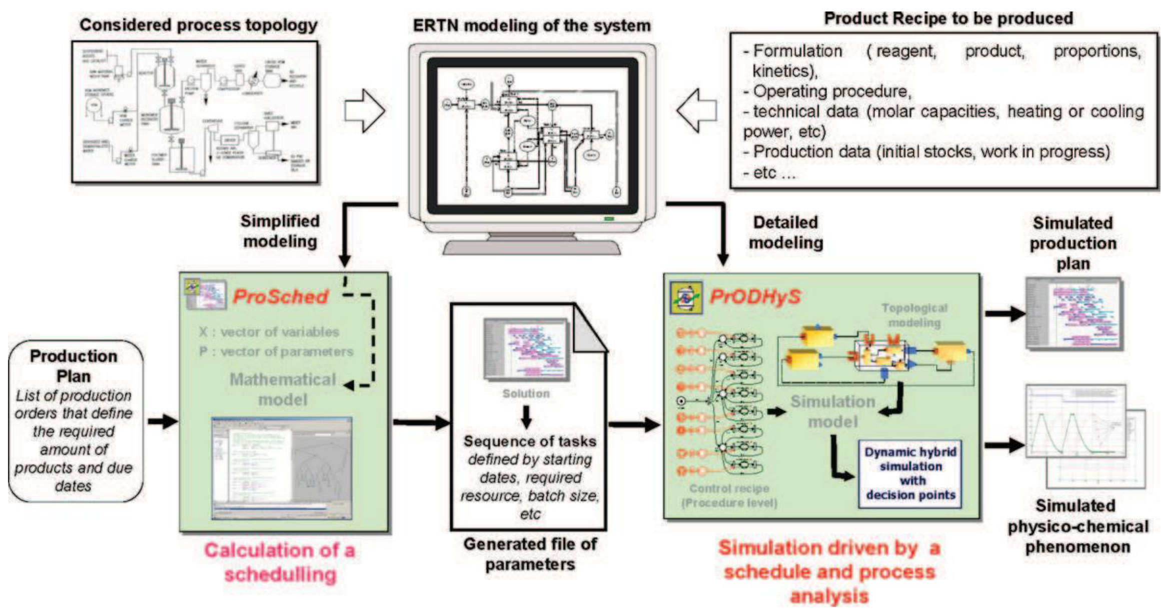

Considered process topology
ERTN modeling of the system
Product Recipe to be produced
- Formulation (reagent, product, proportions, kinetics),
- Operating procedure,
- technical data (molar capacities, heating or cooling power, etc)
- Production data (initial stocks, work in progress)
- etc ...
Simplified modeling
Detailed modeling
Simulated production plan
Production Plan
List of production orders that define the required amount of products and due dates
ProSched
X : vector of variables
P : vector of parameters
Mathematical model
Solution
Sequence of tasks defined by starting dates, required resource, batch size, etc
Generated file of parameters
PrODHyS
Topological modeling
Simulation model
Dynamic hybrid simulation with decision points
Control recipe (Procedure level)
Calculation of a schedulling
Simulation driven by a schedule and process analysis
Simulated physico-chemical phenomenon

Fig. 3. General procedure of a dynamic simulation in *PrODHyS*.

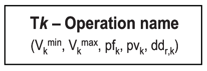
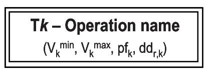
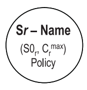
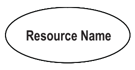
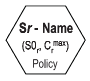
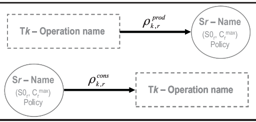
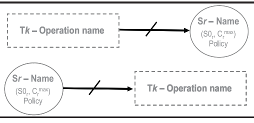
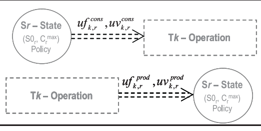
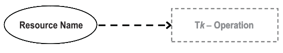
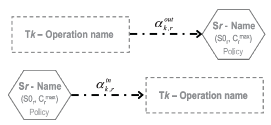
NOM	SYMBOLE	REPRESENTS
Batch task Node	 T*k* – Operation name (V_k^{min}, V_k^{max}, pf_k, pv_k, $dd_{r,k}$)	**Discontinuous processing task *k*** the batch size $B_{k,t}$ is such that $V_k^{min} \le B_{k,t} \le V_k^{max}$, the processing time is $p_k = pf_k + pv_k B_{k,t}$ and the delivery time of resource r is $dd_{r,k}$ (by default, $dd_{r,k} = p_k$)
Continuous task Node	 T*k* – Operation name (V_k^{min}, V_k^{max}, pf_k, $dd_{r,k}$)	**Continuous processing task *k*** the flow rate $B_{k,t}$ is such that $V_k^{min} \le B_{k,t} \le V_k^{max}$, the processing time is $p_k = pf_k$ and the delivery time of resource r is $dd_{r,k}$ (by default, $dd_{r,k} = 0$)
Cumulative resource Node		**Cumulative resource *r*** the amount $S_{r,t}$ of stored resource r is such that $S_{r,t} \le C_r^{max}$, the initial amount is $S0_r$, the *storage policy* is *UIS* or *NIS* or *FIS* (by default, *FIS*) and the *transfer policy* can be *ZW* (by default, *none*)
Disjuntive resource Node		**Disjunctive resource *r*** resource which can be used by only one processing task at a given time
State resource Node		**State resource *r*** the amount $S_{r,t}$ is an integer indicating the actual state of the disjunctive resource r. It is such that $S_{r,t} \le C_r^{max}$, the initial marking is $S0_r$ and the *transfer policy* can be *ZW* (by default, *none*)
Fixed flow Arc	 T*k* – Operation name $\rho_{k,r}^{prod}$ S*r* – Name ($S0_r$, C_r^{max}) Policy S*r* – Name ($S0_r$, C_r^{max}) Policy $\rho_{k,r}^{cons}$ T*k* – Operation name	**Fixed proportion flow of cumulative resource** Cumulative resource flow governed by a conservative mass balance. $\rho_{k,r}^{cons}$ (resp. $\rho_{k,r}^{prod}$) is the fixed proportion of resource r consumed (resp. produced) with respect of $B_{k,t}$ (by default, $\rho_{k,r}^{cons} = 1$ (resp. $\rho_{k,r}^{prod} = 1$))
Free flow Arc	 T*k* – Operation name S*r* – Name ($S0_r$, C_r^{max}) Policy S*r* – Name ($S0_r$, C_r^{max}) Policy T*k* – Operation name	**Free proportion flow of cumulative resource** Cumulative resource flow governed by a conservative mass balance. The / on arc indicates a free proportion of resource r consumed (resp. produced) with respect of $B_{k,t}$. $\mu_{k,r}^{prod}$ or $\mu_{k,r}^{cons}$ is equal to 1 if a free flow arc exists between cumulative resource r and task k, 0 otherwise.
Production / consumption Arc	 S*r* – State ($S0_r$, C_r^{max}) Policy $uf_{k,r}^{cons}, uv_{k,r}^{cons}$ T*k* – Operation T*k* – Operation $uf_{k,r}^{prod}, uv_{k,r}^{prod}$ S*r* – State ($S0_r$, C_r^{max}) Policy	**Production/consumption flow of cumulative resource** Cumulative resource flow not governed by a conservative mass balance. The produced (resp. consumed) amount of cumulative resource r by task k is $u_{k,r}^{prod} = uf_{k,r}^{prod} + uv_{k,r}^{prod} B_{k,t}$ (resp. $u_{k,r}^{cons} = uf_{k,r}^{cons} + uv_{k,r}^{cons} B_{k,t}$)
Use Arc		**« Use » relationship between a processing task and a disjunctive resource** Indicates that the disjunction resource r has the capability to perform the processing task k.
State transition Arc	 T*k* – Operation name $\alpha_{k,r}^{out}$ S*r* - Name ($S0_r$, C_r^{max}) Policy S*r* - Name ($S0_r$, C_r^{max}) Policy $\alpha_{k,r}^{in}$ T*k* – Operation name	**In/out flow of state resource** Indicates an evolution of the actual state (modeled by state resources r) of the disjunctive resource which performes the processing task k. The integer $\alpha_{k,r}^{in} \ge 1$ (resp. $\alpha_{k,r}^{out} \ge 1$) if a transition state arc exists between state resource r and task k, 0 otherwise. By default, $\alpha_{k,r}^{in} = 1$ (resp. $\alpha_{k,r}^{out} = 1$).

Fig. 4. Semantic elements of the *ERTN* graphical formalism.

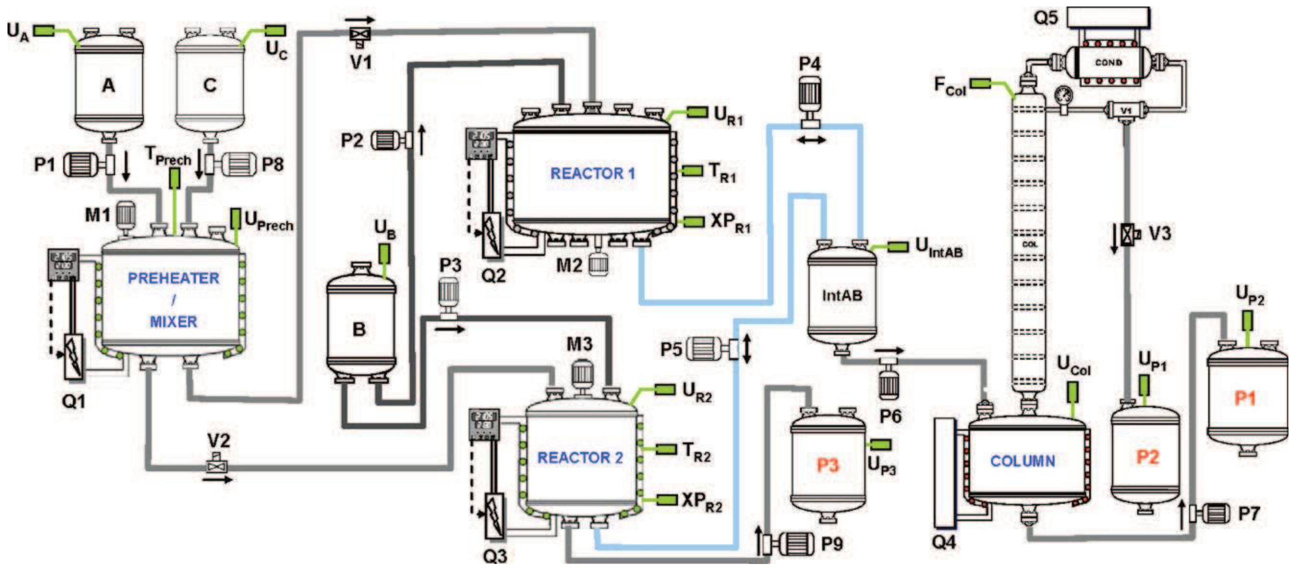


Fig. 5. Topology of the unit.

regarding the energy point of view, reactors and heater consume electricity to maintain the operating conditions and the column requires high pressure steam (**HP**) at boiler and a coolant (**CW**) at condenser.

In these conditions, the procedure of the *site recipe* is then modeled by the *ERTN* shown in Fig. 6. The capacity of tanks is given in kg and the processing times include a fixed and a variable (dependent of batch size) part. The absence of storage tank between the preheater and reactors induces that the state **HotA** has a "capacity" equal to zero and a zero-wait transfer policy. Moreover, note that devices are disjunctive resources while the different kinds of energy are considered as cumulative resources.

4. *MILP* formulation of the scheduling problem

4.1. *Key features of the considered scheduling model*

Different methods are proposed in the literature to solve these problems recognized as NP-complex. Given our goal, any method allowing the simultaneous determination of the starting date and the batch-size of each task is a candidate, providing that a good solution is obtained in a reasonable time. So, the quality of the plan and the provided computational effort are parameters for which a compromise must be found. Several excellent reviews (Burkard & Hatzl, 2005; Floudas & Lin, 2004; Kallrath, 2002; Méndez, Cerdá, Grossmann, Harjunkoski, & Fahl, 2006) clearly point out that Mixed Integer Linear Programming (*MILP*) has been widely used for solving the batch process scheduling problem. In this framework, various formulations of the problem are proposed in the literature (Kondili et al., 1993; Maravelias & Grossmann, 2003). Globally, we can distinguish *MILP* models based on *discrete time* formulation (such as *Global time intervals*) or based on *continuous time* formulation (such as Global time points, Unit- specific time event, Time slots, Unit-specific immediate precedence, Immediate precedence, General precedence, etc.). In our study, the best suited models regarding the combination of optimization and simulation are those based on a continuous-time formulation. A detailed comparison of these continuous-time models can be found in (Shaik et al., 2005). Notably, it presents through several examples of benchmark the good compromise of the *Unit Specific Event* formulation in term of resolution time and robustness.

4.2. *Description of the optimization model*

Originally, the *Unit Specific Event* formulation has been developed by (Ierapetritou & Floudas, 1998). This continuous-time formulation for short-term scheduling introduces the original concept of event points, which are a sequence of time instances located along the time axis of a unit, each representing the beginning of a task or the utilization of the unit. The location of the event points is different for each unit, allowing different tasks to start at different times in each unit for the same event point. The timings of task are then accounted for through special sequencing constraints. Because of the heterogeneous locations of the event points for different units, as well as the definition of an event as only the starting of a task, for the same scheduling problem, the number of event points required is smaller than others continuous-time formulations and subsequently, reduces notably the number of binary variables.

The model currently implemented in *ProSched* corresponds to the formulation found in (Janak, Lin, & Floudas, 2004) with a limited use of "*Big M*" constraints and the aggregation of sequence constraints. Moreover, the capacity limits of storage tank are taken into account partially by the mathematical model described in (Ierapetritou & Floudas, 1998). Indeed, the material balances are calculated only at the beginning of tasks. In some cases, this can locally lead to overflow the capacity of storage tanks (Fig. 7a). However, that is unacceptable in terms of simulation since physical constraints are violated.

Thus, additional constraints have been implemented by (Janak et al., 2004) to tackle this feature. For this, storage tasks are defined. The sequence and timing of these new tasks and the processing tasks are then related so that the amounts of states will be consistent and specified limits can be enforced (Fig. 7b).

Note however that in our formulation, the variables are only indexed by a event number n and a task i that corresponds to a couple (operation, EquipementUnit) and not by an operation i, a device j and event n as in (Janak et al., 2004). This reduces the number of variables and it is consistent with the *ERTN* formalism. As describing in detail the whole *MILP* model is not the aim of this paper, only the fundamental equations are reported below grouped by functions (for example, utility constraints are not given here although they are included in our model). An exhaustive description of these constraints can be found in (Janak et al., 2004).

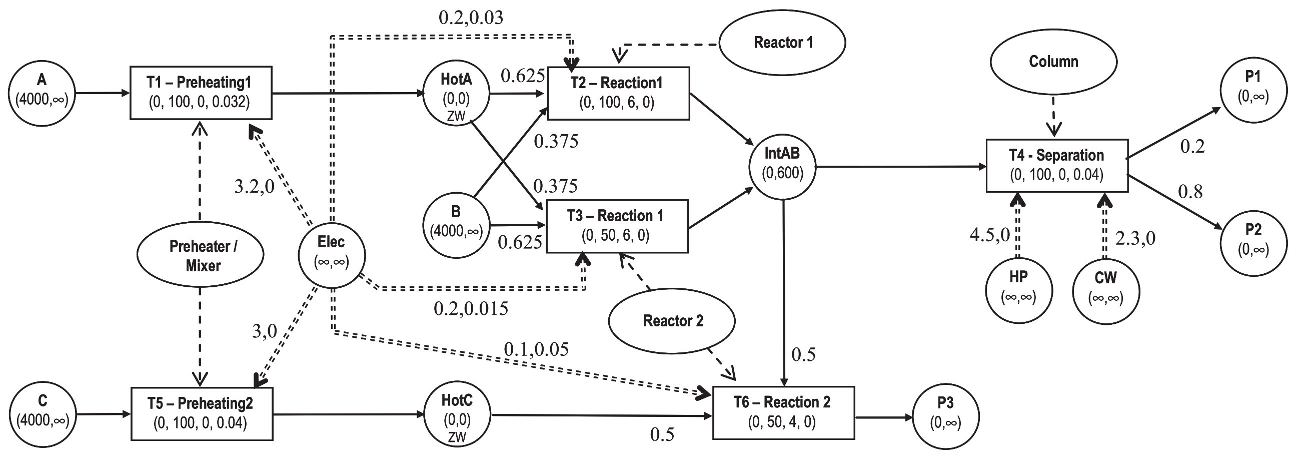
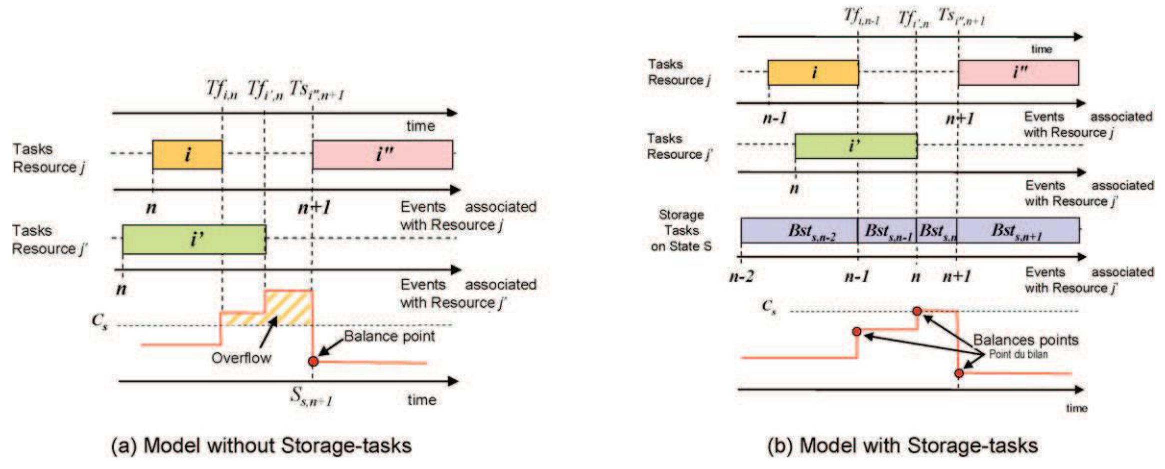


Fig. 6. *ERTN* view of the *site recipe* of the process.



Tf_{i,n} Tf_{i',n} Ts_{i'',n+1}
time
Tasks Resource j
i
i''
n
n+1
Events associated with Resource j
Tasks Resource j'
i'
n
Events associated with Resource j'
C_s
Overflow
Balance point
S_{s,n+1}
time
(a) Model without Storage-tasks
Tf_{i,n-1} Tf_{i',n} Ts_{i'',n+1}
time
Tasks Resource j
i
i''
n-1
n+1
Events associated with Resource j
Tasks Resource j'
i'
n
Events associated with Resource j'
Storage Tasks on State S
Bst_{s,n-2}
Bst_{s,n-1}
Bst_{s,n}
Bst_{s,n+1}
n-2
n-1
n
n+1
Events associated with Resource j'
C_s
Balances points
Point du bilan
time
(b) Model with Storage-tasks

Fig. 7. Refined storage tank capacity constraints.

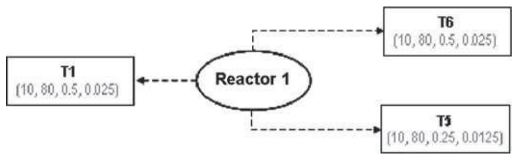


Fig. 8. Allocation constraints (disjunctive resources).

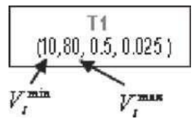


Fig. 9. Capacity constraints of processing tasks.

The nomenclature associated with this model is given in the appendix at the end of the article. On the basis of this notation, the mathematical model involves the following constraints:

Allocation constraints (cf. Fig. 8):

$$\sum_{i\in I_j} W_{i,n} \le 1 \quad \forall_j \in J, \quad \forall n\in N \tag{A.1}$$

$$W_{i,n} = \sum_{n'\le n} Ws_{i,n'} - \sum_{n'<n} Wf_{i,n'} \quad \forall_i \in I, \quad \forall n \in N \tag{A.2}$$

$$\sum_{n\in N} Ws_{i,n} = \sum_{n\in N} Wf_{i,n} \quad \forall_i \in I \tag{A.3}$$

$$Ws_{i,n} \le 1 - \sum_{n'<n} Ws_{i,n'} + \sum_{n'<n} Wf_{i,n'} \quad \forall_i \in I, \quad \forall n \in N \tag{A.4}$$

$$Wf_{i,n} \le \sum_{n'\le n} Ws_{i,n'} - \sum_{n'<n} Wf_{i,n'} \quad \forall_i \in I, \quad \forall n \in N \tag{A.5}$$

Capacity constraints and Batch-size matching constraints of Processing Tasks (cf. Fig. 9)

$$V_i^{\min} W_{i,n} \le B_{i,n} \le V_i^{\max} W_{i,n} \quad \forall_i \in I, \ \forall n \in N \tag{A.6}$$

$$B_{i,n} \le B_{i,n-1} + V_i^{\max}(1 - W_{i,n-1} + Wf_{i,n-1}) \quad \forall_i \in I, \ \forall n \in N | n > 1 \tag{A.7}$$

$$B_{i,n} \ge B_{i,n-1} - V_i^{\max}(1 - W_{i,n-1} + Wf_{i,n-1}) \quad \forall_i \in I, \ \forall n \in N | n > 1 \tag{A.8}$$

$$Bs_{i,n} \le B_{i,n} \quad \forall_i \in I, \ \forall n \in N \tag{A.9}$$

$$Bs_{i,n} \le V_i^{\max} Ws_{i,n} \quad \forall_i \in I, \ \forall n \in N \tag{A.10}$$

$$Bs_{i,n} \ge B_{i,n} - V_i^{\max}(1 - Ws_{i,n}) \quad \forall_i \in I, \ \forall n \in N \tag{A.11}$$

$$Bf_{i,n} \le B_{i,n} \quad \forall_i \in I, \ \forall n \in N \tag{A.12}$$

$$Bf_{i,n} \le V_i^{\max} Wf_{i,n} \quad \forall_i \in I, \quad \forall n \in N \tag{A.13}$$

$$Bf_{i,n} \ge B_{i,n} - V_i^{\max}(1 - Wf_{i,n}) \quad \forall_i \in I, \ \forall n \in N \tag{A.14}$$

Capacity constraints of storage tasks

$$Bst_{st,n} \le C_s^{\max} \quad \forall_s \in S, \ \forall_{st} \in ST_s, \forall n \in N \tag{A.15}$$

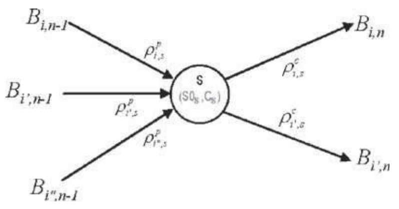


Fig. 10. Material balances on state.

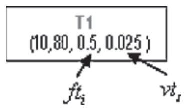


Fig. 11. Duration of processing tasks.

Material Balances including Storage Tasks (cf. Fig. 10)

$$S_{s,n} = S_{s,n-1} + \sum_{i \in I_s} \rho^p_{i,s} Bf_{i,n-1} - \sum_{i \in I_s} \rho^c_{i,s} Bs_{i,n} + \sum_{st \in ST_s} Bst_{st,n-1} - \sum_{st \in ST_s} Bst_{st,n} \quad \forall s \in S, \quad \forall n \in N | n > 1 \tag{A.16}$$

$$S_{s,1} = S0_s - \sum_{i \in I_s} \rho^c_{i,s} Bs_{i,1} - \sum_{st \in ST_s} Bst_{st,1} \quad \forall_s \in S \tag{A.17}$$

$$SF_s = S_{s,N} - D_s + \sum_{i \in I_s} \rho^p_{i,s} Bf_{i,N} + \sum_{st \in ST_s} Bst_{st,N} \quad \forall_s \in S, \tag{A.18}$$

$$S_{s,n} \le C_s \quad \forall_s \in S, \forall n \in N \tag{A.19}$$

$$S_{s,n} = 0 \quad \forall_s \in S^e \cup S^f \cup S^n, \ \forall_n \in N \tag{A.20}$$

Duration constraints of Processing tasks (cf. Fig. 11)

$$Tf_{i,n} \ge Ts_{i,n} \quad \forall i \in I, \forall n \in N \tag{A.21}$$

$$Tf_{i,n} \le Ts_{i,n} + MW_{i,n} \quad \forall i \in I, \forall n \in N \tag{A.22}$$

$$Ts_{i,n} \le Tf_{i,n-1} + M(1 - W_{i,n-1} + Wf_{i,n-1}) \quad \forall i \in I, \forall n \in N | n > 1 \tag{A.23}$$

$$pt_{i,n} = pf_i Ws_{i,n} + pv_i Bs_{i,n} \quad \forall i \in I, \forall n \in N \tag{A.24}$$

$$Tf_{i,n'} - Ts_{i,n} \ge pt_{i,n} - M(1 - Ws_{i,n}) - M(1 - Wf_{i,n'}) - M \sum_{n \le n'' < n'} Wf_{i,n''} \quad \forall i \in I, \forall n \in N, \forall n' \in N, n \le n' \tag{A.25}$$

$$Tf_{i,n'} - Ts_{i,n} \le pt_{i,n} + M(1 - Ws_{i,n}) + M(1 - Wf_{i,n'}) + M \sum_{n \le n'' < n'} Wf_{i,n''} \quad \forall i \notin I^{ps}, \forall n \in N, \forall n' \in N, \quad n \le n' \tag{A.26}$$

Duration Constraints of Storage Tasks

$$Tfst_{st,n} \ge Tsst_{st,n} \quad \forall st \in ST, \forall n \in N \tag{A.27}$$

Sequence constraints of processing tasks: same task in the same unit

$$Ts_{i,n} \ge Tf_{i,n-1} \quad \forall i \in I, \forall n \in N | n > 1 \tag{A.28}$$

Sequence constraints of processing tasks: different tasks in the same unit

$$Ts_{i,n} \ge Tf_{i',n-1} + \tau_{i',i} - M(1 - Wf_{i',n-1} - Ws_{i,n}) \quad \forall j \in J, \ \forall i \in I_j, \forall i' \in I_j | i \ne i', \quad \forall n \in N | n > 1 \tag{A.29}$$

Sequence constraints of processing tasks: different tasks in different units

$$Ts_{i,n} \ge Tf_{i',n-1} - M(1 - Wf_{i',n-1}) \quad \forall s \in S, \forall i \in I^c_s, \forall i' \in I^p_s, \forall j \in J_i, \forall j' \in J_{i'} | j \ne j', \forall n \in N | n > 1 \tag{A.30}$$

Sequence constraints of processing tasks: no-wait condition (ZW transfer policy)

$$Ts_{i,n} \le Tf_{i',n-1} + M(2 - Wf_{i',n-1} - Ws_{i,n}) \quad \forall s \in S^z \cup S^n \cup S^f, \forall i \in I^c_s, \forall i' \in I^p_s, \forall j \in J_i, \forall j' \in J_{i'} | j \ne j', \forall n \in N | n > 1 \tag{A.31}$$

Sequence constraints of storage tasks

$$Ts_{i,n} \ge Tfst_{st,n-1} \quad \forall s \in S, \forall i \in I^c_z, \forall st \in ST_s, n \in N | n > 1 \tag{A.32}$$

$$Ts_{i,n} \le Tfst_{st,n-1} + M(1 - Ws_{i,n}) \quad \forall s \in S^f, \quad \forall i \in I^c_z, \quad \forall st \in ST_s, \quad n \in N | n > 1 \tag{A.33}$$

$$Tsst_{st,n} \ge Tf_{i',n-1} - M(1 - Wf_{i',n-1}) \quad \forall s \in S, \forall i' \in I^p_s, \quad \forall st \in ST_s, \quad n \in N | n > 1 \tag{A.34}$$

$$Tsst_{st,n} \le Tf_{i',n-1} + M(1 - Wf_{i',n-1}) \quad \forall s \in S^f, \forall i' \in I^p_s, \quad \forall st \in ST_s, \quad n \in N | n > 1 \tag{A.35}$$

$$Tsst_{st,n} = Tfst_{st,n-1} \quad \forall st \in ST_s, \quad n \in N | n > 1 \tag{A.36}$$

Bound constraints

$$Tf_{i,n} \le H \quad \forall i \in I, \quad \forall n \in N \tag{A.37a}$$

$$Ts_{i,n} \le H \quad \forall i \in I, \quad \forall n \in N \tag{A.37b}$$

$$0 < W_{i,n} < 1 \quad \forall i \in I, \quad \forall n \in N \tag{A.38}$$

Plan duration constraint:

$$Tf_{i,n} \le \text{Plan} \quad \forall i \in I, \quad \forall n \in N \tag{A.39}$$

Objective function

$$\min \left(a. \ Plan + \sum_{s \in S} h_s SF_s + \sum_{s \in S} \sum_{n \in N} h_s S_{s,n} \right) \tag{A.40}$$

4.3. Complementary tools developed for ProSched module

As mentioned previously, any semantic element of *ERTN* formalism described in Section 3 has a direct translation with sets of constraints of the mathematical model. This gives the generic nature of this model since each problem instance is simply defined through a data file (Fig. 12).

In order to facilitate parameters entry, a “drag and drop” tool has been developed. The user can create its *ERTN* graphically and choose all the parameters (units, tasks, durations, sequences...) of its model. After an automatic verification of the validity of the *ERTN*, the program creates the initialization file compatible with the model of optimization implemented in *Xpress MP*.

After the scheduling phase, the user can proceed directly to a first analysis based on Gantt diagram. A second tool can interpret directly the data provided by the optimizer and display it as a Gantt chart with the evolution of amount of states and batches on the time horizon (see Fig. 13).

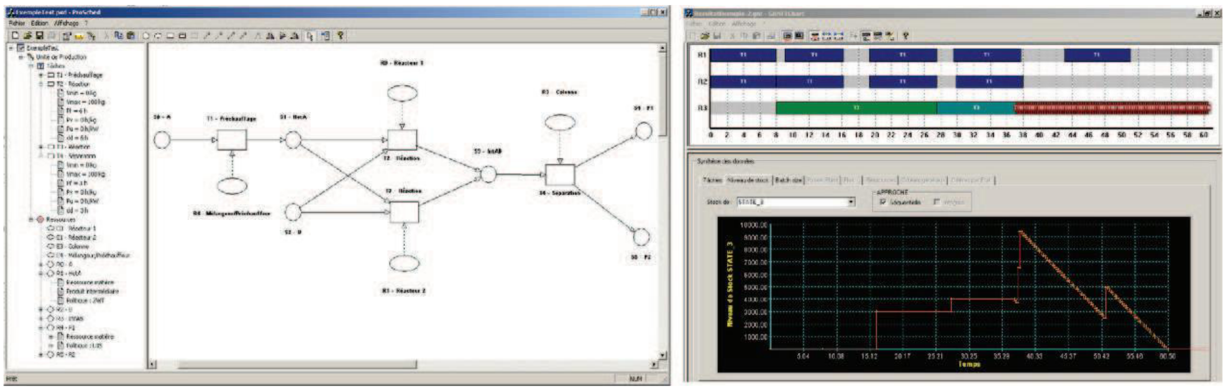


Fig. 12. Tools associated to the first step of the procedure of dynamic simulation.

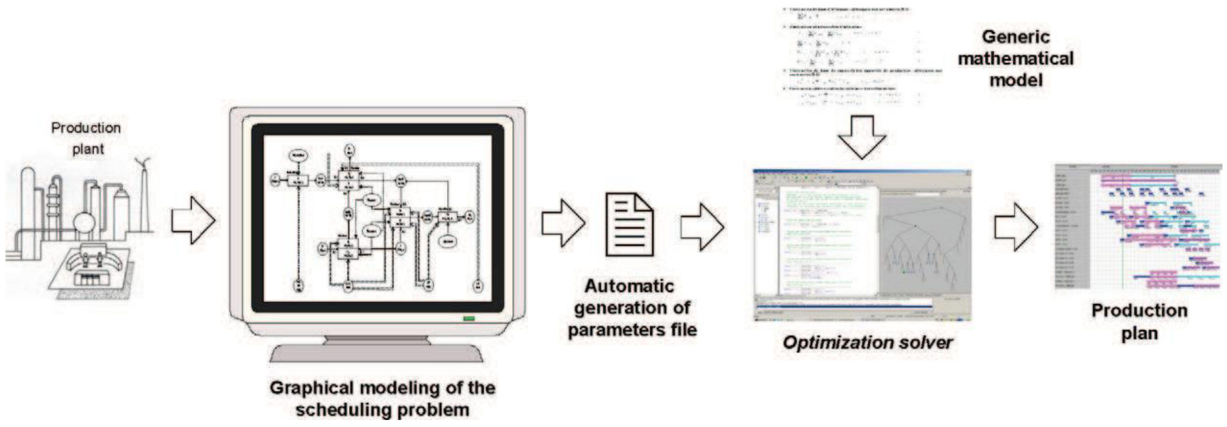

Production plant
Graphical modeling of the scheduling problem
Automatic generation of parameters file
Generic mathematical model
Optimization solver
Production plan

Fig. 13. *ProSched* Generator and Gantt Chart Manager.

5. The dynamic hybrid simulator *PrODHyS*

This section presents briefly the main characteristics of the dynamic simulator *PrODHyS* and then, describes the structure of the simulation model, and especially the command level.

5.1. *The ODPN formalism (Object Differential Petri Net)*

Batch processes are generally classified as dynamic hybrid systems (Zaytoon, 2001). This kind of system requires specific dynamic simulators able to handle rigorously both the continuous evolution of the state variables (temperature increase, chemical kinetics, etc.) and the discrete behavior (on/off pump, open/close valve, etc.). In this framework, the platform *PrODHyS* uses the *Object Differential Petri Net* formalism (*ODPN*) to model the hybrid behavior of both devices and material it contains. Fig. 14 recalls the semantic elements of *ODPN*. The formal definition and evolution rules of this formalism and its implementation within *PrODHyS* are described in detail in (Perret, 2003) and presented in (Hétreux, Perret, & Le Lann, 2003; Hétreux, Thery, Olivier, & Le Lann, 2007; Perret, Hétreux, & Le Lann, 2004; Hétreux, Perret, & Le Lann, 2004).

5.2. *General structure of the simulation model*

To make the simulation of a discontinuous process, it is necessary to model both the control part (the *supervisor*) and the operative part (the *process*). In *PrODHyS*, the simulation model located at the *command* level (presumably specific to the recipe, the topology of the considered process and the production plan to achieve) is completely separated from the simulation models of devices. Indeed, models of devices must be reusable regardless of the context (concept of *component*). Thus, different recipes can be implemented and tested without changing the models associated with the devices (i.e. the *process* level).

The model of the *command* level is the *master ODPN* (called *recipe Petri Net*) whose evolution causes changes in the *ODPN* of the entities located at the *process* level. This *ODPN* corresponds somehow to

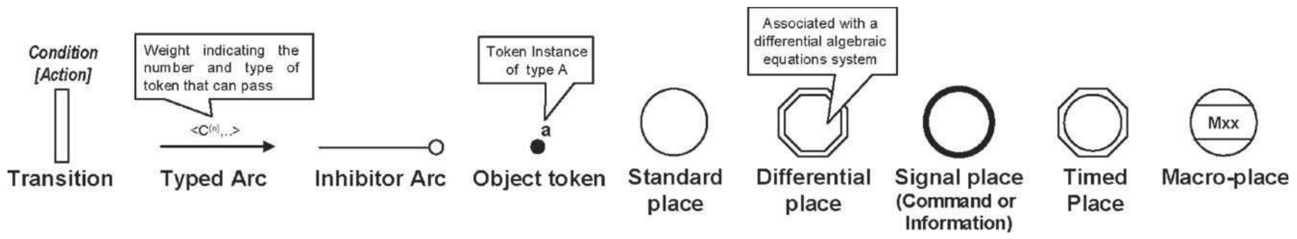

Condition [Action]
Transition
Weight indicating the number and type of token that can pass
<C(n),..>
Typed Arc
Inhibitor Arc
Token Instance of type A
a
Object token
Standard place
Associated with a differential algebraic equations system
Differential place
Signal place (Command or Information)
Timed Place
Mxx
Macro-place

Fig. 14. Semantic elements of *ODPN* formalism.

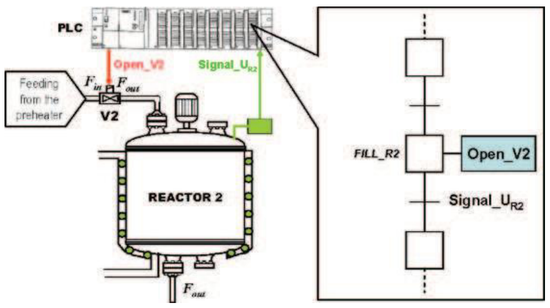


Fig. 15. Subpart of the process shown in Fig. 4 and GRAFCET sequence.

the *control recipe* or *GRAFCET* program executed by a Programmable Logic Controller (see example in Fig. 15).

The signals exchanged between the *command* part and the *process* section correspond either to transmit a command (signal Open_V2), or to receipt an information from a sensor (signal Signal_U_{R2}). A signal is modeled by a place (called respectively, *command* or *information* place) and its status is associated with the marking of this place. These places are the unique link between *process* and *command* levels.

Regarded as black boxes, there are two types of device (see Fig. 16):

- active devices: objects whose Petri net has one or more command and/or information places such as actuators (cf. VALVE V2) and sensors (see CAPTOR UR2)
- passive devices: objects whose Petri net has no direct link with the recipe Petri net such as tanks, reactors (see REACTOR 2) or material.

The marking of a command place of an *active* device induces generally changes in its own Petri net, itself causing evolution in cascade in *passive* devices through the network formed by the connection of the various material or energy ports. In consequence, the evolution of *ODPN* models is conditioned by two distinct types of event:

- first, the *external events* that cause *controlled* switching. These events are issued from the *recipe* Petri net to drive the *active* devices or it is the occurrence of a *state event* (threshold) or a *temporal event*. Specified by the user, these events appear explicitly on the *recipe* Petri net.
- secondly, the *intrinsic events* whose occurrence depends only on the spontaneous evolution of the process. These autonomous switches are for example, a change in material state (the transition from liquid to liquid/vapor when the boiling point is reached) or a commutation in a *passive* device. They therefore do not appear explicitly in the *recipe* Petri net (the user does not have to specify them) and are treated solely within the model of the entity.

Interactions between *recipe* Petri net and *process* Petri net are illustrated in Fig. 16. This is the translation of the system shown in Fig. 15 and it represents an operating sequence in which a reactor is fed until a fixed volume is reached. The *filling* operation is controlled by the *recipe* Petri net by placing a token on the *command* place of the *valve* object (place Open_V2). The feed of reagent is kept open while this *command* place is marked. To detect the ending time of the transfer, a *level detector* object is used. The marking of the *information* place (place Signal_D1) of the *level detector* object indicates that the volume of reagent has reached the target value. The transition is fired. The absence of token on the *command* place then causes the closure of the *valve* object. In the following, only *recipe* Petri net is shown (sequence of operations) and equipment are seen only through their *signal* places.

6. Driving a dynamic hybrid simulation

6.1. Hierarchical modeling of the recipe

The *recipe* Petri net is the link between the optimization model and the simulation model of the operative part. However, when the production system includes extensive facilities or the product development requires many operations, the size of *ODPN* associated with the *command* level can grown quickly. In this case, it is necessary to structure the *control* recipe in successive refinements.

6.1.1. Notion of parameterised macro-place

Based on the decomposition advocated by the *ISA-SP88* (Fig. 17a), the control recipe has a hierarchy on several levels (*procedure, operation, phase, step, instruction*, etc.). To implement this structure in the *ODPN* of the command level, the notion of *macro-places* is exploited. It replaces a sequence of places and transitions relative to an *operation* or a *phase* by a single macro-place. This sequence is then delimited by two special places **E** and **S** between which all types of places defined in the *ODPN* formalism may arise, including other macro-places (Fig. 17b).

At the highest level of the hierarchy (the *procedure*), a macro-place represents the execution of a unit operation. However, some operations may take place in different devices. For example, in the process shown in Fig. 5, the operation called **REACTION 1** can run in the **REACTOR 1** and/or **REACTOR 2**. For this reason, any macro-place can be set with an *EquipmentUnit* object. This object represents the main device (for example, the vessel of the reactor 1) and all the actuators and control equipment (here, the valve **V1**, pumps **P2** and **P4**, heating system **Q2**, engine **M2** and captors **UR1**, **TR1** and **XPR1**). An instance of this object defines a unit of equipment and attributes of this instance is then used to define the commands or signals required for each sequence (Fig. 17b). This hierarchical structure facilitates the specification of the *control* recipe and the setting-up of the simulation model by the use of reusable sequences stored in macro-place (see **Feed** phase in Fig. 17b). An example of *recipe* control is given in (Hétreux, Théry, & Le Lann, 2006). This functionality is rather important since the model of the *command* level is completely disconnected from the models of the *process* level.

6.1.2. Notion of task token

The macro-place *operation* are parameterized by the used devices (see Section 6.1.1) but also by the characteristics of tasks to perform. For example, a reactor in which several reactions can take place requires in each case different operating conditions (temperature, pressure, composition, etc.). It is the case of the reactor called **REACTOR 2** in the process shown on Fig. 5. Similarly, two tasks performing the same operation in the same unit may still have different settings, especially when they depend on the batch size. To address these issues and define more generic *operation* objects, a *task* object has been introduced. The attributes of a *task* object include, among others things:

- the earliest starting date of the task,
- the batch size,
- a reference to the *EquipmentUnit* object allocated to this task,
- a reference to the *operation* object to perform, including all operational parameters (temperature, pressure, composition, etc.) necessary to define the conditions and actions of the sequence (state events associated with continuous variables of the *DAE* systems).

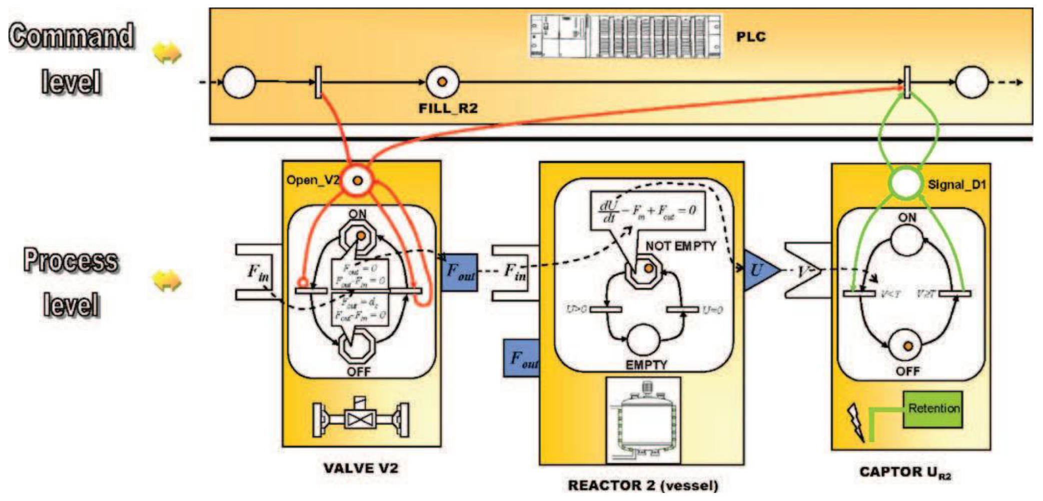
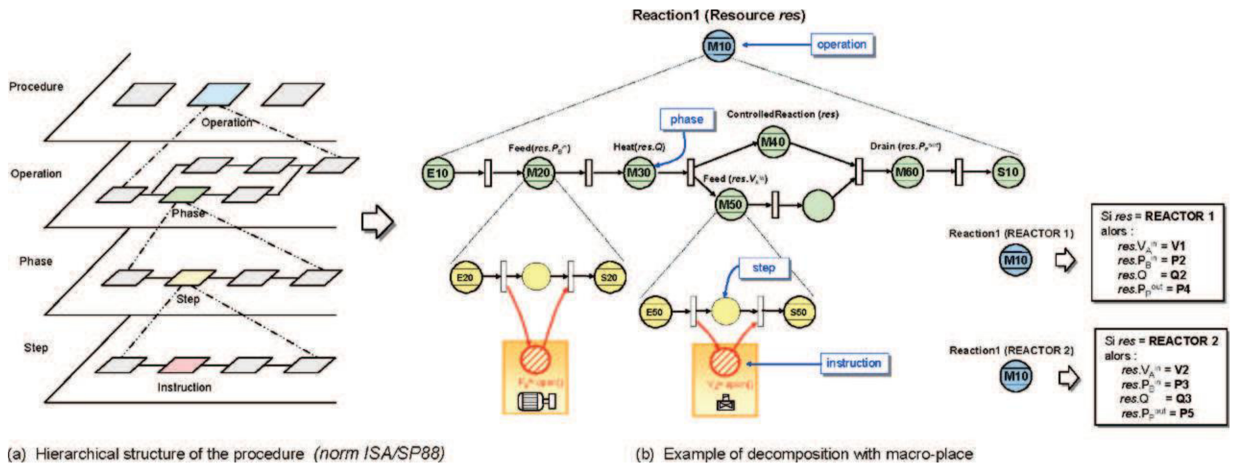


Fig. 16. Interaction between *process* and *command* level.



Procedure
Operation
Phase
Step
Instruction
Reaction1 (Resource res)
M10
operation
phase
ControlledReaction (res)
Feed(res.P_B^in)
Heat(res.Q)
Drain (res.P_P^out)
Feed (res.V_A^in)
E10
M20
M30
M40
M50
M60
S10
E20
S20
E50
S50
step
instruction
Reaction1 (REACTOR 1)
Si res = REACTOR 1 alors : res.V_A^in = V1 res.P_B^in = P2 res.Q = Q2 res.P_P^out = P4
Reaction1 (REACTOR 2)
Si res = REACTOR 2 alors : res.V_A^in = V2 res.P_B^in = P3 res.Q = Q3 res.P_P^out = P5
(a) Hierarchical structure of the procedure (norm ISA/SP88)
(b) Example of decomposition with macro-place

Fig. 17. Hierarchical structure of the *control* recipe using macro-places.

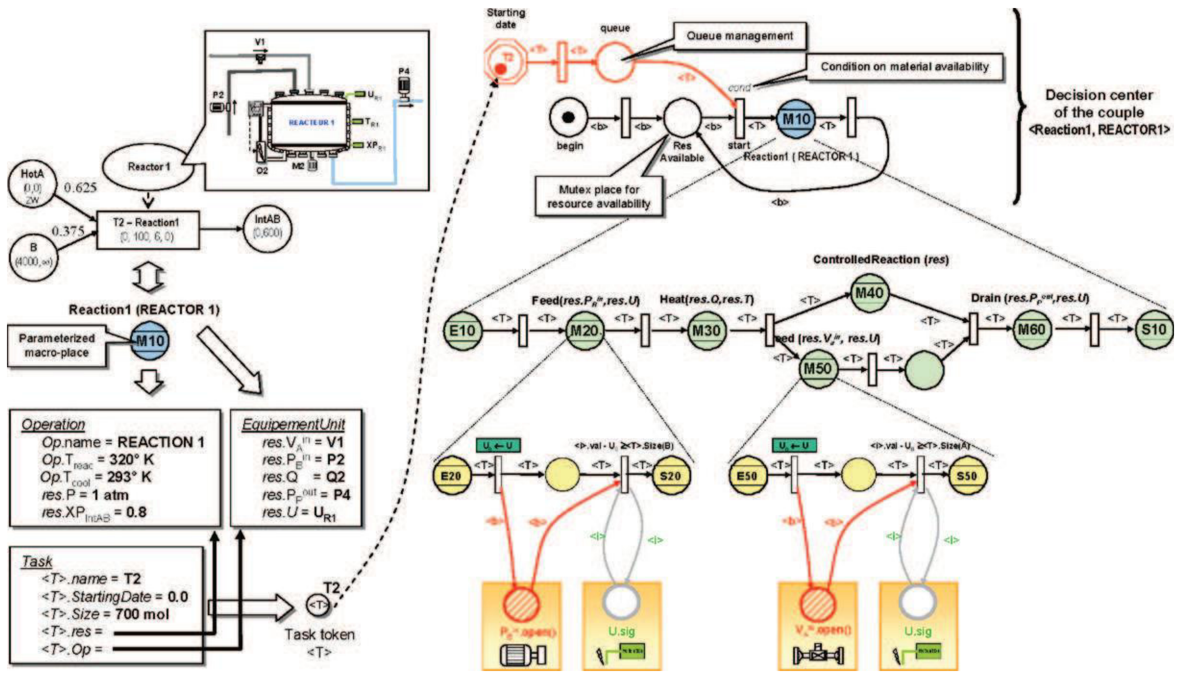


Fig. 18. Part of the *ODPN* (*operation* level) relative to the execution of task <**REACTION** 1, **REACTOR 1**, **700 mol**>.

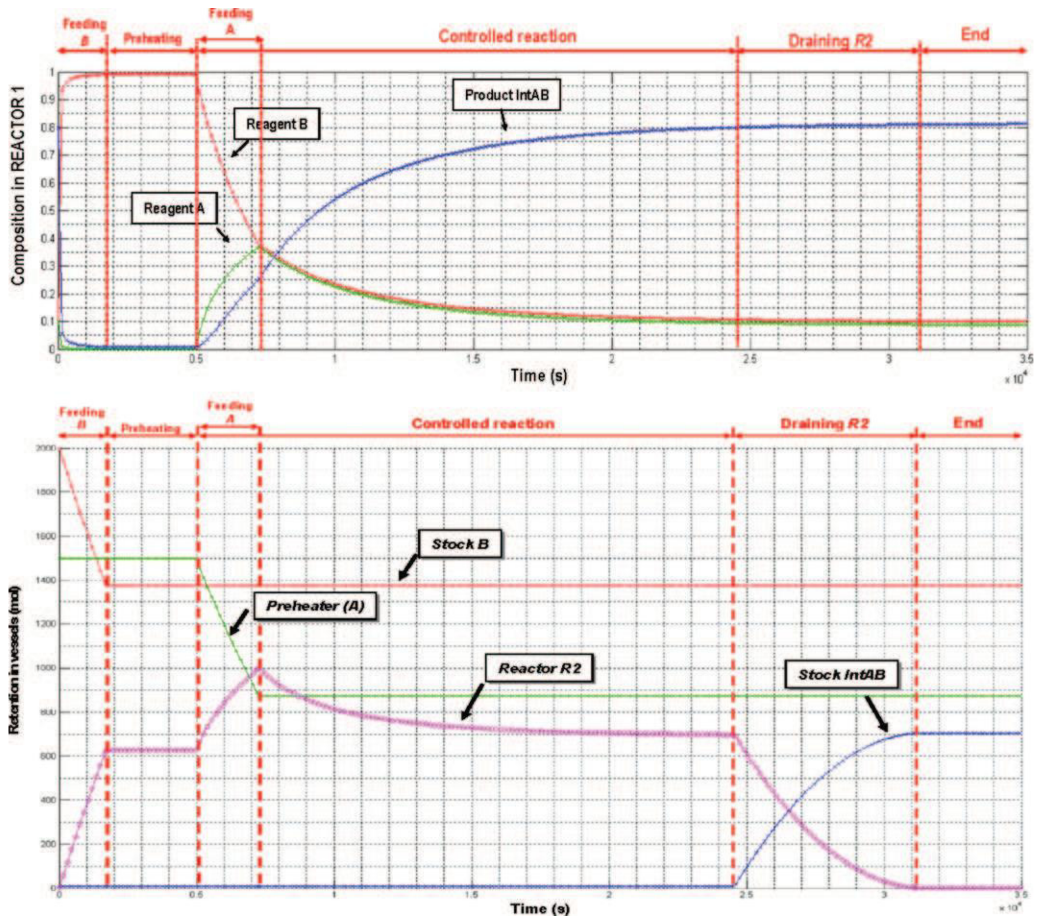


Fig. 19. Simulation results of task <**REACTION 1**, **REACTOR 1**, **700 mol**>.

A *task* object thus defines the triplet **<Operation, EquipmentUnit, BatchSize>**. However, this information should be disseminated to instantiate the conditions and actions of the transitions dispatched on the *ODPN* of an operation. For this reason, the *task* object is associated with a *token* object of type *TaskToken* (noted **<T>**). When an instance of a token **<T>** sensitizes a transition, then the formal variables used to define the conditions or actions are replaced by its attributes (Fig. 18). Finally, note that this token does not materialize a lot of material, but an informational entity used to launch a task. It can therefore be assimilated to an execution order.

6.1.3. *Simulation of a unit operation*

To illustrate the above discussion, a simulation limited to the task <**REACTION** 1, **REACTOR 1**, **700 mol**> is executed. The *ODPN* associated with this operation is shown in Fig. 18. For the launch of a single batch (700 mol of **IntAB**), Fig. 19a and b shows respectively the evolution of the composition in **REACTOR 1** and the retention in the various concerned devices.

6.2. *Structure of the ODPN of the control recipe within the procedure level*

For each unit operation *op* of the procedure carried out on the equipment unit *res* (called couple **<operation, EquipmentUnit>**), a structure called "*decision center*" is implemented as shown in Fig. 18. Furthermore, an instance of *TaskToken* object **<T>** is created for each task corresponding to the triplet (*op, res, size*). This *ODPN* manage both the temporal and the resource availability:

- the *temporal aspect* is supported by a timed place (place **StartingDate** in Fig. 18) for managing the launch of each task. The *delay* parameter of the place is equal to the starting date of the task carried by the token **<T>** ($P(<T>).delay \leftarrow <T>.StartingDate$). When the starting date has expired, the token is released and marks the place dedicated to the management of a queue (place **queue** in Fig. 18) when necessary.
- a *mutex* place is associated with each disjunctive resource (shared devices between operations or not) and manage its availability (place **ResAvailable** in Fig. 18). When this place is not marked, this indicates that the resource is already requisitioned by another task and prevents the crossing of the transition called **start**. So, it avoids the starting of a new task before the end of the previous one.
- a task can be started only after ensuring the availability of materials. Indeed, at the simulation level, the real duration of operations can be shorter or longer than the mean delay taken into account at the scheduling level. For this, a condition placed on the transition located before the **operation** macro-place verifies that the amount of materials are equal to or greater than the proportion required for the batch size carried by the token.

Note also that:

- all tasks that do not share the same equipment unit can potentially be performed in parallel, many tasks associated with the same couple **<Operation, EquipmentUnit>** can exist. In this context, the timed place **StartingDate** is simply marked with a number of token **<T>** equal to the corresponding number of tasks.

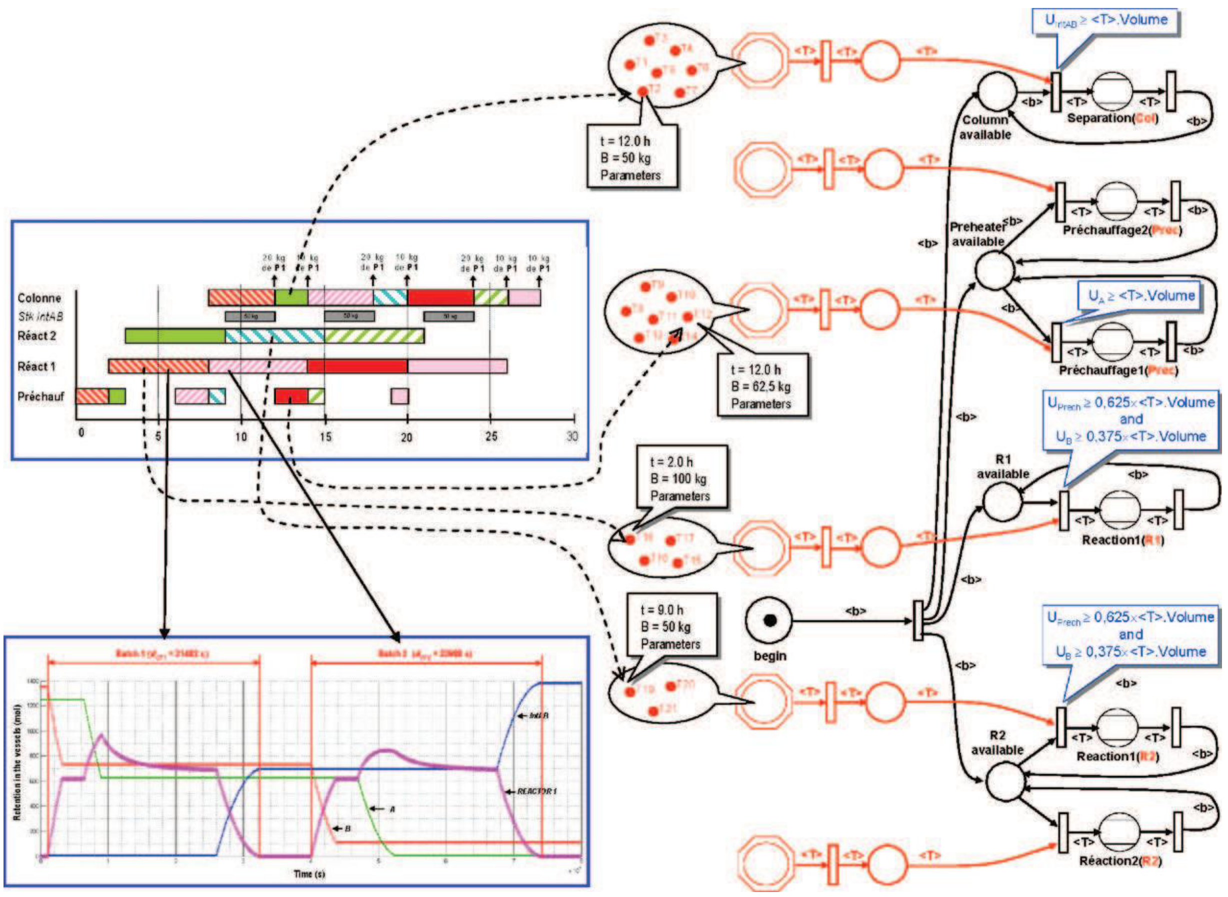


Fig. 20. *ODPN* of the *control* recipe (*procedure* level).

- a *taskToken* object **<T>** represents the data relative to a single task. It becomes obsolete when the task is completed. In other words, the same token can not be used for two successive operations (even if they were identical). As a result, no precedence relationship appears explicitly in the *ODPN* of the *control* recipe at the *procedure* level.

6.3. *Application on the process example*

Based on the *ERTN* shown in Fig. 6, the optimization module establishes a scheduling with the *MILP* model solved with *XPRESS-MP*. The parameters of the mathematical model are initialized with estimated average durations and linearized parameters. Given the characteristics of the process in Fig. 5, the scheduling of a single production order equal to 100 kg of **P1** is shown in Fig. 20.

After the scheduling step, the sequence on each processing unit, the starting dates as well as the number and the batch sizes are transmitted to the simulator. Each task is instantiated and associated with a *taskToken* object **<T>**. Fig. 20 shows the *ODPN* of the *control* recipe at the *procedure* level corresponding to the *ERTN* in Fig. 5 instantiated with the aforementioned scheduling.

The *ODPN* of the control recipe is built by assembling a set of *decision center*, each one associated with a couple **<Operation, EquipmentUnit>**. Thus, operations carried out by several processing units must be duplicated as it is done in the *ERTN* formalism. This case concerns the operation **REACTION1** performed either in **REACTOR1** or **REACTOR2**. In addition, if the same resource *res* is used by several operations op_i then each decision center associated with a couple **<$\mathbf{op_i}$,res>** shares the same *mutex* place (named **ResAvailable**) which models the availability of the resource *res*. This case concerns for example **REACTOR2** which performs both **REACTION1** and **REACTION2**.

The simulation is then performed by following the production plan so defined. Performance indicators can be calculated in order to evaluate the quality of the solution. Fig. 20 shows the successive execution of two batches of identical size in the same device (here, **REACTOR1**). The curves show that the durations of each batch are different (change in feed rate due to a gravity transfer). This example highlights the modeling gap (models used are different by nature) existing between the two modules (optimization/simulation) and the need to provide decisional autonomy to the simulator for the starting (or not) of production tasks. As a result, schedules obtained by simulation and those obtained by optimization are not directly comparable.

Several cases have been solved and generally, the simulations have been correctly completed. Nevertheless, some time constraints may not be completely fulfilled due in most cases to a inaccurate estimation of the processing times at the scheduling level. Indeed, if the duration taken into account in the optimization model is *underestimated*, the simulator starts the task at the earliest when the allocated resource and the required amount of material are available. Nevertheless, future time constraints could not be met. In the opposite, if the duration taken into account in the optimization model is *overestimated*, the simulator is forced to wait the expiry of the scheduled starting date. Here again, future time constraints cannot be guarantied. Fig. 21 illustrates this case.

In fact, as established in (Méndez et al., 2006), a gap always exists between theory and practical due to the simplifying assumptions sometimes introduced to make the problem tractable. This is the reason why the model is called "*simplified*" for the scheduling part, in opposition to the "*detailed*" model for the simulation part which

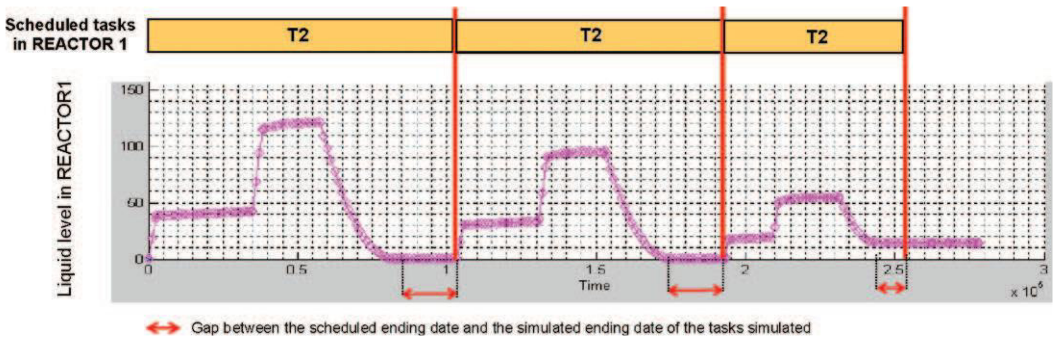


Fig. 21. Simulation results for 3 tasks **<REACTION 1**, **REACTOR 1>** with different batch sizes.

describes the physicochemical phenomenon by differential algebraic equations systems. Moreover, this induces that the search of a mathematical optimum of simplified models can seem useless, in practice. For various reasons, the implementation of such scheduling is often limited when it is confronted with the simulation model of the process. In particular, optimization model are often established under the assumption of constant and known processing times. However, this represents a severe restriction toward the sensitivity of certain operations to the adjustments of the operating conditions. The batch column is an example where the processing time depends on several parameters: the quality of the initial load, the heating policy of the boiler, the reflux policy, the racking side flows, the thermal losses, etc. In addition, the duration of a task can also depend on the state of the system at a given time. For example, the duration of a transfer by gravity is dependent on the retention in the source tank. In the same way, the heating duration of a product depends on the initial temperature, itself being able to depend on the waiting duration of the product in the upstream storage tank if thermal losses exist. Finally, criterion is often reduced to a subpart of the overall objectives considered by end-users. So, if a scheduling is only a "good" solution of the problem, it is not a drawback and the user can adjust some parameters at the simulation level. For these reasons, in this procedure, the optimization calculations are often stopped when a fixed time delay or an integrality gap is met.

In order to refine the results, the above simulation results can be used to reset the data of the mathematical model and thus improve the production plans obtained through an iterative procedure. An another strategy is the simulation of each operation independently for a set of parameters in order to obtained accurate initial data for the scheduling module. Nevertheless, it seems likely that the simulated plans are more easily exploitable because they are based on a more accurate representation of the real phenomena and can provide reference points (temperature, pressure, composition, etc.) during the progression of the in situ operations.

7. Conclusion

Based on object concepts, *PrODHyS* provides software components for the modeling and the dynamic simulation of industrial processes (Hétreux et al., 2002; Hétreux et al., 2003; Perret et al., 2004). The implementation of a high level formalism (*Object Differential Petri Net*) associated with efficient numerical methods (Gear, 1971) has led to the development of a hybrid dynamic simulator numerically robust. In order to deal efficiently with the simulation of batch process, this paper presents a package whose role is to build automatically optimized production scenarios that should run the simulator. For this, several key issues have been addressed. First, it has been introduced the *ERTN* graphical formalism that models the main characteristics of a process. This formalism is used in the software *ProSched Generator* designed to generate the input parameters of the scheduling model. This generic mathematical model (*MILP*) is based on a continuous time formulation called *Unit-Specific Time Event*. This module calculates all input data useful to the simulation model. Secondly, the interface between the optimization model and the simulation model has been established.

For this, the *ODPN* of the control recipe is structured into several levels by using parameterized macro-places. Moreover, information associated with each task is distributed throughout the network thanks to *task token* object.

Currently, the effectiveness of this framework has been proved and several studies on batch processes have been conducted with success. Nevertheless, it might be interesting to test other optimization models to improve the quality of the scheduling obtained in the first step of the procedure. Especially, many robust optimization techniques can be applied in order to explicitly model system uncertainty and generate a schedule which is not only feasible for the nominal system conditions but also robust when considering the distribution of the unknown system parameters (Lin, Janak, & Floudas, 2004; Janak, Lin, & Floudas, 2007; Shaik & Floudas, 2009).

To conclude, note that this procedure is included as a part of a more general method dedicated to the scheduling of batch processes. The fundamental principle is to suppose that an "approximate" solution (in term of behavior) provided by an optimization model with a reduced computational effort, is compensated by a finer modeling of the process carried out at the simulation level. This approach should make more robust the production plans and facilitates the physicochemical analysis of phenomena. However, in order to validate this approach and evaluate quantitatively its effectiveness, several modules are currently in development.

References

Agha M. (2009). Integrated management of energy and production: scheduling of batch processes and CHP plants, Thèse de Doctorat, INP de Toulouse, France.

Burkard, R. E., & Hatzl, J. (2005). Review, extensions and computational comparison of *MILP* formulations for scheduling of batch processes. *Computers and Chemical Engineering*, *29*, 1752–1769.

Fabre F. (2009). Conduite orientée ordonnancement d'un simulateur dynamique hybride: Application aux procédés discontinus, Thèse de Doctorat, INP, Toulouse, France.

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

Gear C.W. (1971), The Simultaneous Numerical Solution of Differential-Algebraic Equations, IEEE Transaction on Circuit Theory, CT 18 (1), Ed. Academic Press.

Ierapetritou, M. G., & Floudas, C. A. (1998). Effective continuous-time formulation for short-term scheduling: 1. Multipurpouse batch processes. *Industrial and Engineering Chemistry Research*, *37*, 4341.

Hétreux, G., Théry, R., Perret, J., Lelann, J. M., & Joulia, X. (2002). *Bibliothèque orientée-objet pour la simulation dynamique des procédés: architecture et mise en oeuvre*. Toulouse: Congrès SIMO.

Hétreux, G., Perret, J., & Le Lann, J. M. (2003). Object hybrid formalism for modelling and simulation of chemical processes. In *ADHS'03* Saint-Malo, France, Hétreux, G., Perret, J., & LeLann, J. M. (2004). Composant based approach for simulation of dynamic hybrid systems. In *Conference on Conceptual Modeling and Simulation (CMS'04)* Genoa, Italy.

Hétreux, G., Théry, R., & Le Lann, J. M. (2006). *Outil d'aide à la décision pour l'ordonnancement des procédés semi-continus: le noyau de simulation dynamique hybride*. Toulouse, France: Congrès SIMO.

Hétreux, G., Thery, R., Olivier, N., & Le, Lann. (2007). De la simulation dynamique hybride vers la conduite de procédés batch et semi-continus. *Journal Européen des Systèmes Automatisés (J.E.S.A) Edition LAVOISIER*, *41*(5), 585–616.

Janak, S. L., Lin, X., & Floudas, C. A. (2004). Enhanced Continuous-time unit-specific event-based formulation for short-term scheduling of multipurpose batch processes: resource constraints and mixed storage policies. *Industrial Engineering and Chemical Research*, *43*(10), 2516–2533.

Janak, S. L., Lin, X., & Floudas, C. A. (2007). A new robust optimization approach for scheduling under uncertainty: II. Uncertainty with known probability distribution. *Computers and Chemical Engineering*, *31*, 171–195.

Jourda, L., Joulia, X., & Koehret, B. (1996). Introducing ATOM, the applied thermodynamic object-oriented model. *Computers and Chemical Engineering*, *20A*, S157–S164.

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

Kondili, E., Pantelides, C. C., & et Sargent, R. W. H. (1993). A general algorithm for short-term scheduling of batch operations – I *MILP* formulation. *Computers and Chemical Engineering*, *17–2*, 211–227.

Lin, X., Janak, S. L., & Floudas, C. A. (2004). A new robust optimization approach for scheduling under uncertainty: I. Bounded uncertainty. *Computers and Chemical Engineering*, *28*, 1069–1085.

Maravelias, C. T., & Grossmann, I. E. (2003). Minimization of the makespan with a discrete-time State-Task network formulation. *Industrial and Engineering Chemistry Research*, *42*, 6252–6257.

Méndez, C. A., Cerdá, J., Grossmann, I. E., Harjunkoski, I., & Fahl, M. (2006). State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Computers and Chemical Engineering*, *30*(6–7), 913–946.

Moyse, A. (2000), Odysseo: plate-forme orientée-objet pour la simulation dynamique des procédés. Thèse de Doctorat, INP de Toulouse, France.

Olivier-Maget N. (2007). Surveillance des systèmes dynamiques hybrides: application aux procédés, Thèse de Doctorat, INP de Toulouse, France.

Pantelides C.C. (1994). Unified framework for the optimal process planning and scheduling, *CAHE Publications, Proceedings of the second Conference on Foundations of Computer Aided Operations*, pp. 253–274.

Perret J. (2003), Intégration des Réseaux de Petri Différentiels à Objets dans une plateforme de simulation dynamique hybride: application aux procédés industriels, Thèse de Doctorat, INP, Toulouse, France.

Perret, J., Hétreux, G., & Le Lann, J. M. (2004). Integration of an object formalism within a hybrid dynamic simulation environment. *Control Engineering Practice (Elsevier)*, *12/10*, 1211–1223.

Sargousse A. (1999). Noyau numérique orientée-objet dédié à la simulation des systèmes dynamiques hybrides, Thèse de Doctorat, INP de Toulouse, France.

Shaik M.A., Janak S.L. et Floudas C.A. (2005), Continuous-Time Models for Short-Term Scheduling in Multipurpose Batch Plants: a Comparative Study, Chemical Engineering, Princeton University, Princeton, NJ 08540.

Shaik, M. A., & Floudas, C. A. (2009). Novel Unified Modeling Approach for Short-Term Scheduling. *Industrial and Engineering Chemistry Research*, *48*, 2947–2964.

Zaytoon, J. (2001). *Systèmes dynamiques hybrides*. HERMES Sciences publications.