Real time scheduling of batch systems

Stéphane JULIA*, Robert VALETTE**

*Depto. de Informática,
Universidade Federal de Uberlândia, P.O. Box 593, 38400-902 Uberlândia, MG, Brazil

** LAAS-CNRS, 7 Avenue du Colonel-Roche, 31077 Toulouse Cedex France

e-mail: stephane@triang.com.br robert@laas.fr

Abstract – The approach presented in this article is based on the real time simulation of p-time Petri net for the real time scheduling of batch systems. After defining the different kinds of constraints which can exist in a linear hybrid production system, we present the conflict resolution principle used by a token player algorithm at the global coordination level.

Keywords – Petri nets, batch processes, real-time scheduling.
1 Introduction

The globalization of markets and the ever changing needs of society generated the substitution of mono product processes by flexible multi product processes called: batch systems. These systems are used in particular by the food and fine chemistry industries. They operate on continuous raw materials (generally fluids) and are composed of continuous equipment (a continuous flow of material goes through them, as in thermal exchanges for example) and also discrete equipment (reactors, intermediary buffers etc.). A batch is a quantity of material which is characterized by real numbers (volume units etc.). The act of manipulating diverse batches which can be treated simultaneously and which are transferred from one reactor to another, shows that these systems look to a certain extent like flexible manufacturing systems [15]. As a matter of fact, in these kind of systems, there exists similar resource allocation mechanisms. For example, the same reactor can be necessary to treat different batches. If the reactor works in a closed mode (a single batch, at the same time, in a single reactor), it can be seen as a discrete resource like in a flexible manufacturing system. This means that these resources are non preemptive disjunctive ones.

It is at the global coordination level of batch systems, during the real time scheduling [14], that the resource allocation mechanism problems appear. The general principle of the real time scheduling of production systems is based on the real time simulation of the system model which is carried out in parallel with the real working of the system. Then, the real time scheduling consists of making decisions in real time each time a conflict appears.

The scheduling problem of flexible production systems has been studied by several researchers. When we consider flexible manufacturing systems, the principal purpose is to find a solution which minimizes the makespan. For example, in [12], Lee and DiCesare use a t-timed Petri net model to
represent a flexible manufacturing system. In particular, the model allows one to represent the
different production routes and the shared resources (a more general model than an event graph). To
find a sequence of operations which minimizes the makespan, the authors apply a branch and bound
algorithm to the model and transitions are fired as soon as they are enabled. With such an approach,
several limitations have to be taken into account. Even using a branch and bound algorithm, there
exists the problem of the state explosion and if we manage to obtain an optimal solution, in any
way, it will not be possible to obtain it in real time. Another problem is that firing transitions as
soon as they are enabled will not give us necessarily the optimal schedule as Silva and Valette have
shown it on a simple example in [14]. For batch systems, obtaining a feasible schedule consistent
with a set of constraints (not necessarily optimal) is more relevant. Therefore the scheduling
problem has to be seen more as an analysis of a system under a set of constraints as in [7]. As a
matter of fact, in a batch system, a waiting time too long between two operations can damage the
batch. Of course, the same does not happen with a part of a flexible manufacturing system where
only the makespan is, most of the time, the criterion to optimize. Another problem which has to be
highlighted is that working at the global coordination level, the solution of the scheduling problem
will have to be calculated in real time to take into account the possible perturbations which can
appear during the real operation of the system. Actually, the operation durations are ill known in
food and fine chemical industry because they depend on raw material quality. Consequently, it
would not be realistic to follow a perfect sequence of operations calculated off-line.

To follow, the authors will define a model adapted to the representation of the constraints which are
able to exist within a batch system. A specialized inference mechanism will be presented too.
Applied to the model of the system, this inference mechanism is used to solve in real time the
conflict situations.
2 Real time simulation model

The model used for the real time scheduling of batch systems has to describe in real time the operation sequences to be made on the batches, to communicate with the external world, and to be submitted to explicit temporal constraints. In particular, it has to evolve in synchronization with the physical system. The global coordination and real time monitoring of production systems based on a model has been presented in [5]. Each time, the state of the system changes (each time an operation ends), the detection module verifies if the state of the simulation model is compatible with the actual state of the system and brings up to date the state of the simulation model. The decision module indicates then which operation (or which operations) has to be initiated (figure 1).

In the case of hybrid systems, it is common to separate the model of the system in two distinct models [1]: the reference model and the control module which specifies the control policy. The reference model represents a qualitative view of the behavior of the system independently of any control policy (model of the global behavior of a reactor for example) and is used at the detection level to verify if a reactor does not overflow for example. The control module controls the flow of products which have to be treated and is used more for production management.

Figure-1. Global coordination of production systems founded on a model

When the representation of the resource allocation mechanism is essential, then, Petri nets are well adapted. Extended models have been proposed [3] to consider the hybrid aspect of batch systems. When batch systems can be seen as linear hybrid systems [6], then, by approximation, it is generally possible to remain within the context of an event driven simulation and to use a discrete model with time considerations. Then, the activities are characterized by durations. In place of monitoring
thresholds for determining the dates of the end of activity events, they are derived by simulating a p-time Petri net by means of a token player algorithm.

In this paper, the main objective is to treat the real time scheduling problem of batch systems and, for that, we will only work using the control module of the model which will be represented by a discrete model (discrete states and continuous time) obtained by approximation of a linear hybrid model.

First of all, several kinds of constraints which exist in batch systems have to be defined.

2.1. Recipe

The production route constraints (the recipes of the batch process) describe the set of operations, organized in time, for the realization of products. They are represented by sequences of places and transitions. The tokens which go through these sequences represent the products in transformation (the treated batches). The beginnings and the ends of the operations are attached to the transitions (the events) and the operations (treatment and transfer operations of batches) are attached to the places.

Figure-2. Recipe
The net of figure 2 gives an example of such a recipe which is applied to the batch modeled by the token in place Bi. This place represents the entry buffer of the production route. Place Bo represents the exit buffer. Places O1, O2, and O3 represent the three treatments which have to be applied to the batch. And, finally, the transitions represent the beginnings and the ends of operations O1, O2, and O3.

2.2. Resource allocation constraint

The resource allocation constraints permit the representation of flexible behavior in hybrid systems when the discrete resources (reactors in a closed mode and intermediary buffers) are shared among the diverse recipes. The resources are represented by tokens of the places which are connected to the production route constraints. For example, the constraint given by figure 3 means that the reactor, represented by the token in the place R, will be used as well to realize treatments of the Recipe 1 as to realize treatments of the Recipe 2.

*Figure-3. Resource allocation constraint*

The global model of the system is obtained by merging (place and transition merging) the diverse constraints (all the Petri net templates). It will be important to verify that the global model is deadlock free. Unfortunately, in the case of batch systems, for any transfer operation, we have to request a new resource before releasing the preceding one. This can lead to a deadlock resulting from a siphon which is empty. A classical example of deadlock in the case of batch systems is presented in [16]. An approach for deadlock avoidance is shown in [4,15]. It consists of adding a new place to the global model that controls the maximum number of tokens in the siphon.

2.3. Continuous aspect of the model
A typical way to encapsulate continuous phenomena within a discrete model is to turn explicit their durations. These durations are continuous values and in consequence a kind of hybrid model is obtained. In batch systems, durations of transfer operations and durations of transformation operations can be associated with places or transitions of the Petri net model. For example, Azzopardi and Lloyd [2] use, in the case of batch systems, a p-timed Petri net model to solve the optimum scheduling problem. Nevertheless, to use a timed Petri net model, it is necessary to know the exact operation durations for all the configurations. In batch systems, it will be in practice impossible to know the exact durations of the transformation operations. For example, for a batch of milk to be boiled, it will have a minimum duration before which the milk will not boil and a maximum duration after which the milk will burn. Because of that, a simple timed model will not be useful neither to compute a realistic previsional schedule, nor to compute the real time schedule. It is then necessary to introduce uncertain durations in the formal model. The p-time Petri net model defined by Khansa [11], will allow us to introduce the uncertain durations of the transformation operations of the batches. The definition of this model is the following one:

**Definition 1**: A p-time Petri net is a pair <R,I_p>, where:

- R is a marked Petri net,
- I_p is the application defined as:

  \[ I_p : P \rightarrow (Q^+ \cup 0) \times (Q^+ \cup +\infty) \]

  \[ p_i \rightarrow I_{pi} = [a_i, b_i] \text{ with } 0 \leq a_i \leq b_i \]

\[ I_{pi} = [a_i, b_i] \] is the static interval which represents the permanency duration (sojourn time) of a token in pi. Before the duration a_i, the token in pi is in the non-available state. After a_i and before
bi, the token in pi is in the available state for the firing of a transition. After bi, the token in pi is again in the non-available state and cannot enable any transition anymore: “token death”. In the batch system case, the “death” of a token has to be seen as a constraint which is not respected.

The dynamic evolution of a p-time Petri net depends on the marking of the net and on the temporal situation of the tokens (non-available, available, dead). For the real time scheduling of hybrid systems, when the model is a Petri net, the problem is then to simulate in real time the temporal model, to highlight the conflict situations and to verify that the local conflict resolution will not have as a direct consequence the “death” of a token (constraint violation).

3 General approach for the real time scheduling

The principal aim of this approach is to take into account the state of the system not only at a single instant corresponding to the reception date of the last message of an operation’s end. We work on a short temporal projection which permits one to emphasize possible future conflicts.

3.1. Conflict definition

First of all, we have to define what a conflict is for a p-time Petri net. For an autonomous Petri net, only qualitative time information is represented; for example, two operations executed at the same time (parallelism) or two operations in mutual exclusion (transitions in conflict for a given marking). When explicit quantitative time is represented on the model, the problem is much more complex. With a p-time Petri net, the resource conflicts are visible during a time interval and not only at a single time point. It is then possible to generate more fired sequences which can lead to better solutions. Indeed, it is not always a good strategy to fire transitions as soon as they are
enabled as *Silva and Valette* showed in a simple example [14]. The necessary definitions to understand the conflict notion of a p-time Petri net were introduced in [8,9]. We recall them here.

**Definition 2:** Visibility interval of a token: A visibility interval \([ (\delta_p)_{\text{min}} ; (\delta_p)_{\text{max}} ]\) associated with a token of a place \(p\) of a p-time Petri net defines:

- the earliest date \((\delta_p)_{\text{min}}\) when the token in \(p\) becomes available for the firing of an output transition of \(p\).
- the latest date \((\delta_p)_{\text{max}}\) after which the token becomes non-available (“dead”) and cannot be used for the firing of any transition.

In the context of a batch system, the “death” of a token means that a temporal constraint has been violated.

**Definition 3:** Enabling interval of a transition: if a transition has \(n\) input places and if each one of these places has several tokens in it, then the enabling interval \([ (\theta_t)_{\text{min}} ; (\theta_t)_{\text{max}} ]\) of this transition is obtained by choosing for each one of these \(n\) input places a token, the visibility interval associated to this token, and making the intersection of all the obtained visibility intervals.

**Definition 4:** Conflict time interval: if two transitions \(t_1\) and \(t_2\) are in structural conflict, then the conflict time interval of these transitions is obtained by making the intersection of the enabling intervals of \(t_1\) and \(t_2\). Inside the obtained time interval, \(t_1\) and \(t_2\) are in effective conflict for the marking; outside this time interval, they are not in effective conflict.

*Figure-4. Conflict for a p-time Petri net*
Let us consider the example of figure 4, with the following static intervals associated with the places p1, p2, and p3:

\[
[(d_{p1})_{\text{min}} ; (d_{p1})_{\text{max}}] = [1;6],
\]
\[
[(d_{p2})_{\text{min}} ; (d_{p2})_{\text{max}}] = [0;7],
\]
\[
[(d_{p3})_{\text{min}} ; (d_{p3})_{\text{max}}] = [2;6].
\]

At date 0 a token arrives in p1, at date 2 a token arrives in p3, and, finally, at date 3, a token arrives in p2. The visibility intervals of these tokens are:

\[
[(\delta_{p1})_{\text{min}} ; (\delta_{p1})_{\text{max}}] = [1;6],
\]
\[
[(\delta_{p2})_{\text{min}} ; (\delta_{p2})_{\text{max}}] = [3;10],
\]
\[
[(\delta_{p3})_{\text{min}} ; (\delta_{p3})_{\text{max}}] = [4;8].
\]

The enabling intervals are [3;6] for t1 and [4;8] for t2. The conflict time interval associated to the pair (t1;t2) is [4;6].

### 3.2. Conflict state

When a token, which represents a batch, enables a transition which is in structural conflict with other transitions, then, considering the visibility interval of this token, we have to calculate the possible arrival dates of other tokens which are in the input places of the transitions in structural conflict with the enabled transition. According to the possible arrival dates of these tokens, it is then possible to define the diverse visibility intervals of the diverse tokens and to draw up the possible conflict time intervals. From the obtained results, the useful part of the net, necessary to study locally the possible conflicts for the involved shared resource, can be isolated.
Figure-5. Conflict for a shared resource

We can consider for example the net of figure 5 at date $\delta=6$. The token in $O_{31}$ becomes available for the firing of $t_{41}$. $t_{41}$ is in structural conflict with $t_{42}$ and $t_{43}$. We have then to calculate the possible arrival dates of tokens in $O_{32}$ and in $O_{33}$ between the date 6 and 14 which correspond to the minimum and maximum bounds of the visibility interval of the token in $O_{31}$. Here, we only want to establish the possible existence of an effective conflict between several transitions in a p-time Petri net. Then, we choose as arrival dates, the earliest arrival dates of the tokens. At date 10 (minimum bound of the visibility interval of the token of place $O_{22}$), if the transition $t_{32}$ is fired, then a token appears in $O_{32}$ and its visibility interval is $[8+10;10+10] = [18;20]$. If the transitions of the recipe 3 are fired as soon as possible, at date $7+3 = 10$ with 7 the minimum bound of the visibility interval of the token of place $O_{13}$ and 3 the minimum bound of the static interval of the place $O_{23}$, a token appears in $O_{33}$ and its visibility interval is $[2+10;5+10] = [12;15]$. Since the visibility interval of the resource is $[0;+\infty[$, the enabling interval of $t_{41}$ is $[6;14]$, the enabling interval of $t_{42}$ is $[18;20]$, and the enabling interval of $t_{43}$ is $[12;15]$. The conflict time interval of the pair $(t_{41};t_{43})$ is equal to $[6;14] \cap [12;15] = [12;14]$ and we will eventually be able to have an effective conflict between $t_{41}$ and $t_{43}$ in a p-time Petri net. The conflict time interval of the pair $(t_{41};t_{42})$ is equal to $[6;14] \cap [18;20] = \emptyset$ and we will not have an effective conflict between $t_{41}$ and $t_{42}$. It means that $t_{41}$ will be necessarily fired before $t_{42}$. The part of the net which represents the recipe 2 will not be directly involved in the firing of $t_{41}$ and we will not consider it at the local level of the conflict analysis. The useful part of the net for the conflict analysis of the shared resource of place R is represented on figure 6.

Figure-6. Useful part of the conflict
3.3. Conflict resolution

As the conflict resolution technique must be used in real time, instead of enumerating the set of the acceptable sequences we can obtain from the state of the Petri net fragment of figure 6, we look at the local level of the conflict and the consequences of a particular decision. If we consider the net of figure 6, at date $\delta=6$, the token in $O_{31}$ becomes available for the firing of $t_{41}$. It seems normal then to fire $t_{41}$ as soon as possible since there is no token in $O_{33}$ at that time. If we fire $t_{41}$ at date $\delta=6$, a new token appears in $O_{41}$ and its visibility interval is equal to $[2+6;3+6] = [8;9]$. From this new state (a token in $O_{41}$ with its visibility interval equal to $[8;9]$ and a token in $O_{13}$ with its visibility interval equal to $[7;12]$, we want to verify the consequences of this decision (firing $t_{41}$ at date $\delta=6$) on the rest of the net of figure 6. In particular, we want to verify that this decision will not cause a constraint violation i.e. the “death” of a token. The tool which allows one to represent all the possible evolutions of a p-time Petri net is the class graph defined by Khansa [10] from that of [13]. Such a graph allows one to define the “death token classes” corresponding to the possibility of existence of “death tokens”.

Figure-7. Class graph without death token classes.

Figure 7 represents the class graph of the net of figure 6 obtained after the firing of $t_{41}$ at date $\delta=6$.

The classes of the graph are the following ones :

\[ C_0 : M_0=\{O_{41},O_{13}\},(2\leq do_{41}\leq 3)\land(1\leq do_{13}\leq 6) \]
\[ C_1 : M_1=\{O_{13},R\},(0\leq do_{13}\leq 4)\land(0\leq dr\leq +\infty) \]
\[ C_2 : M_2=\{O_{41},O_{23}\},(0\leq do_{41}\leq 2)\land(3\leq do_{23}\leq 4) \]
\[ C_3 : M_3=\{O_{23},R\},(1\leq do_{23}\leq 4)\land(0\leq dr\leq +\infty) \]
\[ C_4 : M_4=\{O_{33},R\},(2\leq do_{33}\leq 5)\land(0\leq dr\leq +\infty) \]
For example, the class C0 means that we have for the net of figure 6, after the firing of t41, a token in O41 and a token in O13. The token in O41 will have to stay in this place, before being used for the firing of a transition, at least a duration equal to 2 and at most a duration equal to 3. The token in O13 will have to stay in this place at least a duration equal to 1 and at most a duration equal to 6. On the graph, we have two arcs from the class C0. The one which connects C0 to C1 means that if we choose to fire t51 before firing t23, the transition t51 will be fired necessarily between the minimum duration 2 and the maximum duration 3. The one which connects C0 to C2 means that if we choose to fire t23 before firing t51, the transition t23 will be fired necessarily between the minimum duration 1 and the maximum duration 3. The detailed algorithm used to construct this graph is presented in the Khansa PhD thesis [10].

There is no “death class” in this graph and consequently, we can fire the transition t41 at date $\delta=6$.
$$C_2 : M = \{O_{41}, O_{33}\}, (8 \leq d_{O_{41}} \leq 16) \land (2 \leq d_{O_{33}} \leq 5)$$

The class $C_2$ is a “death token class” which leads us to a deadlock. In fact, we can see that in this state class, the token in $O_{41}$ has to stay in this place at least a duration equal to 8. The shared resource will not be available in place $R$ before this duration. On the other hand, the token in $O_{33}$ only stays in this place at most a duration equal to 5. If we remain for a longer duration in this particular place, then we will have the “death” of the token and a constraint violation for the batch system. We know that after having waited at the maximum a duration equal to 5 in the place $O_{33}$, the resource in $R$ will not be available and the “death” of the token in $O_{33}$ will occur.

We deduce from this class graph that the transition $t_{41}$ cannot be fired at date $\delta = 6$ when the token in $O_{31}$ becomes available and we will have to try in any way to fire the transition $t_{43}$ before $t_{41}$, expecting then that we will not have the “death” of the token in $O_{31}$.

### 3.4. Token player algorithm

When the model of the system is a Petri net, then, the real time simulation, which is carried out in parallel with the real operation of the system, is made using a special inference mechanism called: token player.

*Figure-9. Token player algorithm for a p-time Petri net.*

Figure 9 represents such an algorithm when the model is a p-time Petri net. The communication process between the real system and the model is made by sending and receiving messages. The received messages generally represent operation endings and are the events attached to the operation ending transitions. When we receive one of these messages, if the corresponding
transition is not enabled i.e. if the token which is in the input place of this transition (the one associated to the external event) is not available at this moment, then, the token player algorithm detects an abnormal behavior. The detection function is then followed by a diagnosis function in order to identify the reasons of such an inconsistency between the state of the system and the state of the model. The problem of the detection based on a model of the process is presented in details in [16].

Each time a token becomes available, we verify if it enables a transition. If it is the case and if the transition is not in structural conflict, then we fire it and we send a new message to the system to initiate a new operation (an event attached to the fired transition and corresponding to the beginning of an operation). If the transition is in structural conflict, we isolate the conflict state (the corresponding Petri net fragment) and then we verify if we can fire the transition (conflict resolution mechanism). If the conflict resolution mechanism allows one to fire the corresponding transition as soon as the token becomes available, then we fire the transition and a new message is sent to the system to initiate a new operation.

Finally, if there is “death” of a token (detection of an internal event which represents the maximum bound of a visibility interval), then, there is a constraint violation which could have been caused by bad management of the available resources (wrong decisions) or by a physical problem at the system level. We have then to relax a constraint (increase the value of the maximum bound of a static interval) each time it is possible. On the contrary, the batch is damaged : in particular, this situation happens when the maximum bound represents the maximum treatment duration of a batch.
4 Conclusion

The presented approach can be seen as a simulation with imprecise dates for the firing of transitions. The advantage is that the token player algorithm of a p-time Petri net can be seen as a special inference mechanism which can be used for the real time scheduling and which can accept several acceptable sequences. It is thus a tool very well adapted to the real time monitoring of batch systems which are flexible production systems. Another great interest of this approach is that the calculation of the conflict time intervals allows us to isolate the useful part of the net when a decision has to be made (conflict resolution). The class graph (each node of the class graph encapsulates an infinite number of states) is then generated only from a Petri net fragment and not from the entire Petri net model. The combinatorial problem corresponding to the explosion of the number of states which exists each time the automata corresponding to the marked Petri net is generated is thus avoided.

As a future work proposal, we will study the situations where the corresponding class graphs contain both “death token classes” and safe ones. In particular, it will be interesting to show how, changing the minimum and the maximum bounds of certain static intervals (the ones associated to the batch transformation operation), we could eliminate the “death token classes” and keep at the same time most of the safe classes. In this way, we could keep only the sequences without deadlock.
References


Figure 1
Figure 2
Figure 3
Figure 4
Figure 7
Figure 8