A CONSTRAINT PROGRAMMING MODEL FOR SOLVING REACHABILITY PROBLEM IN TIMED PETRI NETS

Yongliang Huang, Thomas Bourdeaud'Huy, Pierre-Alain Yvars, Armand Toguyeni

9^{th} *International Conference of Modeling, Optimization and Simulation - MOSIM'12*
June 6-8, 2012 - Bordeaux - France
"Performance, interoperability and safety for sustainable development"

A CONSTRAINT PROGRAMMING MODEL FOR SOLVING REACHABILITY PROBLEM IN TIMED PETRI NETS

Yongliang HUANG[†], Thomas BOURDEAUD'HUY[†]
Pierre-Alain YVARS[‡], Armand TOGUYENI[†]

[†] LAGIS, Ecole Centrale de Lille
Avenue Paul Langevin
59650 Villeneuve d'Ascq - France
{yongliang.huang, thomas.bourdeaud_huy, armand.toguyeni}@ec-lille.fr

[‡] LISMA, Supmeca
Saint Ouen - France
payvars@supmeca.fr

ABSTRACT: *In this paper, we propose to use a constraint programming approach to address the reachability problem in Timed Petri Nets (TPNs). TPNs can be used to model a wide class of systems, from manufacturing issues to formal verification of embedded systems. Many of the considered problems can be modeled as reachability problems in TPNs. To reduce the space state explosion brought by the exploration of the TPN behaviour, we propose to follow the incremental methodoly proposed by (Bourdeaud'huy & Hanafi 2006), who translate the reachability graph of Timed PNs into a mathematical programming model using linearization techniques. We show how to adapt this model to the constraint programming paradigm by using reified contraints and compare the respective efficiency of the two models.*

KEYWORDS: *Timed Petri Nets, Reachability Problem, Constraint Programming, Reified Constraints.*

1 INTRODUCTION

Petri Nets (PNs) have been well used for modeling, analyzing, synthesizing and implementating Manufacturing systems for decades, see (Recalde, Silva, Ezpeleta & Teruel 2004), (Zhou & Venkatesh 1999). The modeling problem in manufacturing systems is usually characterized by concurrent and synchronous discrete events. PNs are well suited for this type of modeling because they capture the precedence relations and interactions among these events. Since PNs have a strong mathematical foundation, they allow both qualitative and quantitative analysis of the properties of such systems.

Using PNs, many problems in manufacturing systems can be expressed as reachability problems, which consist in finding transition firing sequences leading from the initial state to a given target state. When considering time delays for operations or events is needed, several classes of PNs can be proposed, like *Timed PNs* which are considered in this paper.

We consider the reachability problems in TPNs for two contexts: task scheduling or reconfiguring a system. The scheduling problem can be reduced to finding an optimal sequence of transitions leading from an initial marking to a final marking. This problem can be solved by integer linear programming. However, in the case of reconfiguration, two issues must be considered:

- Dynamic reconfiguration (on line) requires to find quickly a possible sequence;
- Off-line analysis of the possibilities of reconfiguration requires to enumerate all the sequences.

The latter is inadequately addressed in linear programming. This is why we choose to evaluate the capacity of constraint programming to solve these problems.

Practically, it is not possible to explore the reachability graph exhaustively due to the well known problem of combinatorial explosion. Many methods have been studied to limit this explosion (Michel & Vernadat 1998). Here, we follow the approach given by (Bourdeaud'huy & Hanafi 2006), who use a unique sequence of steps to capture all the behaviors of the system of a given size, avoiding to build the whole reachability graph.

In this study we focus specifically on the use of *reified constraints*, a way to add constraints dynamically during the search space exploration. Such constraints act as a substitute for the linearization techniques used before and allow to reduce the number of variables needed to express the TPN behaviour.

The previous mathematical model is improved and adapted to constraint programming paradigm. A practical study is made using *Ilog Solver*, and several case studies.

The paper is organized as follows. In section 2, we give the definition of Timed PNs and its reachability problem. In section 3, we introduce the mathematical programming model given in (Bourdeaud'huy & Hanafi 2006). In section 4, we adapt it using reified constraints as an alternative to the linearization technique used before, and propose additional improvements allowing to reduce the search space. In section 5, we assess the respective efficiency of both models by giving some numerical experiments. Finally, we conclude by giving a few promising research perspectives.

2 TIMED PETRI NETS

2.1 Petri Nets

Petri Nets were first proposed by (Petri 1962), they are a graphical and mathematical modeling tool used for many classes of systems. They are a promising tool for describing and studying systems characterized as being concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic (Murata 1989).

2.1.1 Definition

A Places/Transitions Petri net $(R = (\mathbb{P}, \mathbb{T}, C), m_0)$, see (Murata 1989), is a bipartite directed graph where:

- $\mathbb{P}$ and $\mathbb{T}$ are two finite sets of nodes denoted respectively *places* and *transitions* with $|\mathbb{P}| = \text{M}$ and $|\mathbb{T}| = \text{N}$. Places generally represent conditions, and transitions represent events (Murata 1989).
- C is the incidence matrix which is defined as: $\forall p \in \mathbb{P}, \forall t \in \mathbb{T}, C(p,t) = C^+(p,t) - C^-(p,t)$, in which $C^-(p,t)$ and $C^+(p,t)$ respectively represent the weight of arcs from places to transitions and from transitions to places.
- $m_0 : \mathbb{P} \to \mathbb{N}$ associates to each place $p \in \mathbb{P}$ an integer $m_0(p)$ called the *marking* of the place p.

2.1.2 Graphic and Matrix representation

In the graphic representation of a Petri Net, places are represented as circles and transitions as rectangles. Each place p of the net is associated with the number of tokens corresponding to its marking $m(p)$. Tokens are represented as full disks inside places, or simply by an integer label.

In the following, we will use linear algebra formulations to make the presentation more concise:

- The vector $\overrightarrow{e_{t_k}}$ will denote the Parikh vector associated to the transition t_k, the k^{th} component of which takes the value 1 and others zero;
- The vector $\overrightarrow{m_0}$ will denote the marking vector of the net, defined by $\overrightarrow{m_0} = (m(p_1), m(p_2), \ldots, m(p_{\text{M}}))^{\intercal} \in \mathbb{N}^{\text{M}}$.

2.1.3 State equation

A transition t_k is said to be *fireable* from a given marking m_0 (denoted by $m_0[t_k\rangle$), if and only if its *upstream* places have enough tokens: $m_0[t_k\rangle \quad \Leftrightarrow \quad \overrightarrow{m_0} \geq C^- \cdot \overrightarrow{e_{t_k}}$.

The firing of t_k from m_0 produces a new marking m_1 (denoted by $m_0[t_k\rangle m_1$) such that:

$$m_0[t_k\rangle m_1 \quad \Leftrightarrow \quad m_0[t_k\rangle \wedge \overrightarrow{m_1} = \overrightarrow{m_0} + C \cdot \overrightarrow{e_{t_k}}$$

The Parikh vector $\overrightarrow{\sigma}$ associated to the sequence $\sigma = t_{\sigma_1} t_{\sigma_2} \ldots t_{\sigma_r}$ is defined as the vector from $\mathbb{N}^{\text{N}}$ the j^{th} component of which takes a value equal to the number of occurences of the transition t_j in σ: $\overrightarrow{\sigma} = \sum_{j=1}^{r} \overrightarrow{e_{t_{\sigma_j}}}$.

A *fireable transition sequence* (aka. *firing sequence*) σ leading from m_0 to m_f is denoted $m_0[\sigma\rangle m_f$. Using its Parikh vector and the previous linear algebra notations, one can get a classical property of PNs:

$$m_0[\sigma\rangle m_f \Rightarrow \overrightarrow{m_f} = \overrightarrow{m_0} + C.\overrightarrow{\sigma}.$$

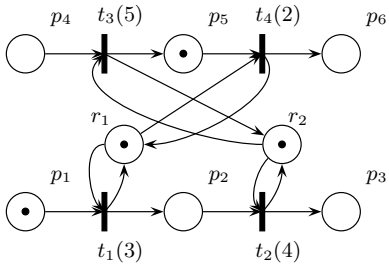
This equation is known as the *state equation* of PNs. The reverse implication (which is not true in general) allows to formulate the well known *PNs reachability problem* in the following way:

"Given an initial marking m_0 and a final marking m_f, find a firing sequence σ such that $m_0[\sigma\rangle m_f$".

Note that in the general case, several transitions may be fired simultaneously. Such a multiset of transitions is called a *step*, and denoted also by $\overrightarrow{\sigma}$ hereafter. Steps firings and fireable steps sequences can be defined in the same way as above.

2.2 Timed Petri Nets

Many PNs classes have been proposed to handle time extensions. For adding time extensions to the PNs, time can be associated with transitions, places and arcs, and time annotations can correspond to durations (*Timed PNs*) or time intervals (*Time PNs*), see (Wang 1998). The following definition of Timed PNs is derived from (Chretienne 1984).



	Date	Marking
Initial date	0	$(1,0,0,0,1,0,1,1)^\intercal$
Firing of $t_1 \rightarrow$	1	$(0,0,0,0,1,0,0,1)^\intercal$
	2	$(0,0,0,0,1,0,0,1)^\intercal$
	3	$(0,0,0,0,1,0,0,1)^\intercal$
End of $t_1 \rightarrow$	4	$(0,1,0,0,1,0,1,1)^\intercal$
Firing of $t_4 \rightarrow$	5	$(0,1,0,0,0,0,0,1)^\intercal$
	6	$(0,1,0,0,0,0,0,1)^\intercal$
End of $t_4 \rightarrow$	7	$(0,1,0,0,0,1,1,1)^\intercal$
	8	$(0,1,0,0,0,1,1,1)^\intercal$

Figure 1: Timed Petri net and its firings

2.2.1 Timed PNs

A Timed Petri Net $(R = (\mathbb{P}, \mathbb{T}, C, d), m_0)$ is a PN where a *duration* $d(t) \in \mathbb{N}^*$ is associated to each transition t. The vector $\overrightarrow{d} = \sum_{t \in \mathbf{T}} d(t) \cdot \overrightarrow{e_t}$ is called the *duration vector* of the Timed PN.

To simplify the study, we restrict ourselves to TPNs *without immediate transitions* (transitions where $d(t) = 0$), which is not so restrictive in real world practice.

TPNs have the same representation as classical PNs, to which is added a *labeling* on transitions. An example of Timed PN is given in (Fig. 1). We have: $d(t_1) = 3$, $d(t_2) = 4$, $d(t_3) = 5$, $d(t_4) = 2$.

2.2.2 Firing Policy

The firing durations associated to transitions modify the *marking validity conditions*. Since durations are associated to transitions, the TPN acts as if tokens *"disappeared"* at the time the transition is fired, and then *"reappeared"* after a delay corresponding to the duration of the fired transition. Thus, the marking of a Timed Petri net evolves with the occurences of an external timer.

For instance, let's consider the Timed Petri net of (Fig. 1). At date 1, the transition t_1 (duration: 3 *t.u.* (time units)) is fired. Then the transition t_4 (duration: 2 *t.u.*) is fired at date 5. The evolution of marking with time is given in Fig. 1.

Note that one could have fired transition t_4 at date 4, since the resource r_1 had been released at the end of the firing of transition t_1. However, the same transition was not fireable at date 3, since the firing of t_1 was not finished.

Finally, the firing and ending dates of transitions play a fundamental role in the behaviour of the Timed Petri net. It is thus necessary to *adapt* the firing equations according to these firing dates.

2.3 Timed Petri Nets Reachability Problem

Using the previous notations, the TPNs reachability problem consists of searching for a feasible *controlled execution* allowing to reach a final state $\overrightarrow{m_f}$ from an initial state $\overrightarrow{m_0}$.

It is quite simple to see a parallelism between a scheduling problem and a TPN reachability problem. Indeed, let's consider for instance the Timed Petri net of (Fig. 2). One remarks obviously that solving a reachability problem between markings $m_0 = \{p_1, p_2, p_3, m_1, m_2, m_3\}$ and $m_f = \{p_4, p_5, p_6, m_1, m_2, m_3\}$ means exactly finding the scheduling of the production.

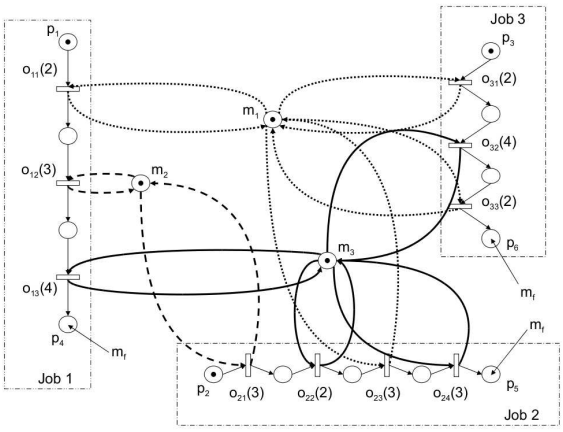


Figure 2: Timed PNs modeling a job shop.

Practically, it is not possible to explore the reachability graph exhaustively due to the well known problem of combinatorial explosion.

Several approaches have been proposed to solve the Timed Petri net reachability problem, or by restricting their study to a subclass of Time PNs, like Timed Event Graphs, either by using dedicated heuristics. A complete bibliography can be found in (Richard 1997).

Note the approach proposed by (Carlier & Chretienne 1988) using *earliest firing dates* cannot handle the general Timed Reachability Problem since firing transitions as soon as they are enabled may lead to miss some optimal firing sequences. For instance, let's consider the TPN of (Fig. 3). The earliest firing sequence $(t_1 t_4, t_5, t_2 t_6, t_3)$ takes 22 t.u. However, the optimal solution is given by the sequence $(t_1 t_4, t_2, t_3 t_5, t_6)$ which needs 14 t.u.

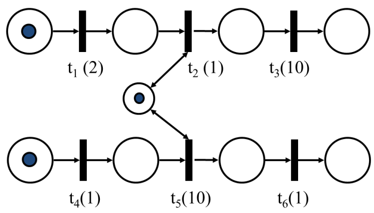


Figure 3: Earliest firing date doesn't mean optimality.

More general solutions can be found in (Driss, Yim, Korbaa & Ghedira 2004), where untimed firing sequences are generated first, and then dates are associated to firings by propagating time constraints. Unfortunately, such method requires to enumerate all firing sequences first, which leads to combinatorial explosion, and it may exist an infinite number of associations of dates to firings (if a transition can be fired at date d, it is still fireable any moment later) .

In this paper, we propose to use a direct approach proposed by (Bourdeaud'huy & Hanafi 2006) who capture the timed behavior using a mathematical model allowing its exact resolution.

3 INTEGER LINEAR PROGRAMMING MODEL

3.1 Incremental Model

In (Bourdeaud'huy & Hanafi 2006), the model is based on an *incremental* propagation of constraints corresponding to an increasing number of *firing dates* in *"Controlled Executions"* which have been introduced by (Carlier & Chretienne 1988) to represent the *history* of the transition firings. It allows to express the PNs state reached at any date.

The *state* of a Timed PNs behaving under a controlled execution is given at any time v by its classical *marking vector* $\overrightarrow{e_m}(v) \in \mathbb{Z}^{\mathrm{M}}$ and a *residual durations vector* $\overrightarrow{e_r}(v) \in \mathbb{N}^{\mathrm{N}}$ associating to each *active* transition its *remaining* duration. Obviously, such representation of the vector of residual durations is based on the assumption that reentrance is forbidden: a transition that is currently firing cannot start another firing.

Let $\mathbb{V}$ be the set of firing dates v_i of the considered controlled execution, with $|\mathbb{V}| = \mathsf{V}$. We express at any firing date $v_i \in \mathbb{V}$ (denoted by the time $\Delta_{v_{i-1}}$ elapsed from the previous firing) the induced modifications on the vectors $\overrightarrow{e_m}(v_i)$ and $\overrightarrow{e_r}(v_i)$ by the firing of step $\overrightarrow{\sigma}(v_i)$.

A synoptic of notations is given in Table 1.

Following the behaviour given in section 2.2, the state vectors $\overrightarrow{e_m}(vi)$ and $\overrightarrow{e_r}(vi)$ must be updated in the following way when the step $\overrightarrow{\sigma_i}$ is fired at date v_i:

Table 1: Synoptic table of notations.

e_{m_0}	$\longrightarrow$	$e_{m_1} \cdots e_{m_k}$	$\longrightarrow$	$e_{m_{k+1}} \cdots e_{m_{\mathsf{V}-1}}$	$\longrightarrow$	e_{m_V}
$v_0 = 0$	σ_1 $v_1 = v_0 + \Delta_{v_0}$		σ_{k+1} $v_{k+1} = v_k + \Delta_{v_k}$		σ_V $v_\mathsf{V} = v_{\mathsf{V}-1} + \Delta_{\mathsf{V}-1}$	
e_{r_0}	$\longrightarrow$	$e_{r_1} \cdots e_{r_k}$	$\longrightarrow$	$e_{r_{k+1}} \cdots e_{r_{\mathsf{V}-1}}$	$\longrightarrow$	e_{r_V}

- The residual duration vector $\overrightarrow{e_r}(v_i)$ contains the durations of new transitions fired by step $\overrightarrow{\sigma_i}$;
- The residual duration vector at date $\overrightarrow{e_r}(v_i)$ contains *also* the durations of transitions that were active at the previous date v_{i-1} and are still firing;
- Tokens consumed by $\overrightarrow{\sigma_i}$ disappear from the marking vector $\overrightarrow{e_m}(v_i)$;
- The marking vector $\overrightarrow{e_m}(v_i)$ contains *also* the tokens produced by transitions that were active at the previous date v_{i-1} and are no more firing at date v_i.

Note that we do not consider intermediate states between two firings, even if a transition started at date v_{i-1} has finished its firing at a date strictly lower than v_i. This is not needeed to check if the sequence of steps $\sigma_{i,i\in[\![1,\mathsf{V}]\!]}$ is fireable, since tokens are only consumed at firing dates. Moreover, it allows to reduce the number of states to consider, thus reducing the combinatorial explosion.

To express the update of state vectors in a linear way, compatible with using integer linear programming solvers, (Bourdeaud'huy & Hanafi 2006) use the following conventions. If $\overrightarrow{x}$ is a vector from $\mathbb{Z}^k$, they denote by:

- $\overrightarrow{x}^+ \in \mathbb{N}^k$ the vector of its positive components, such that $\overrightarrow{x}^+(i) = \overrightarrow{x}(i)$ if $\overrightarrow{x}(i) > 0$ and 0 otherwise;
- $\overrightarrow{x}^s \in \mathbb{B}^k$ the vector representing its *sign*, such that: $\forall c \in [\![1,k]\!], \overrightarrow{x}^s(c) = 0$ if $\overrightarrow{x}(c) \leq 0$ and $\overrightarrow{x}^s(c) = 1$ otherwise.

The previous notations leads to:

$$\overrightarrow{e_r}(v + \Delta_v) = \sum_{t \in \mathbb{T}} d(t) \cdot \overrightarrow{\sigma_{v+\Delta_v}}(t) \cdot \overrightarrow{e_t} + \left(\overrightarrow{e_r}(v) - \Delta_v \cdot \overrightarrow{Id}\right)^+ \quad (22)$$

$$\overrightarrow{e_m}(v + \Delta_v) = \overrightarrow{e_m}(v) - C^- \cdot \overrightarrow{\sigma_{v+\Delta_v}} + C^+ \cdot \left(\overrightarrow{e_r}(v)^s - \left(\overrightarrow{e_r}(v) - \Delta_v \cdot \overrightarrow{Id}\right)^s\right) \quad (23)$$

Obviously, the same equations remain valid if no transition is fired at the date $v + \Delta_v$ (i.e. $\overrightarrow{\sigma_{v+\Delta_v}} = \overrightarrow{0}$), and allow to evaluate the *instantaneous state* at this date.

Let $(R, \overrightarrow{m_0}, \overrightarrow{r_0}, \overrightarrow{d})$ be a Timed Petri Net with $\mathbb{P} = \{p_1, \ldots, p_{\mathrm{M}}\}$, $\mathbb{T} = \{t_1, \ldots, t_{\mathrm{N}}\}$. Let $C^+ \in \mathbb{N}^{\mathbb{P}\times\mathbb{T}}$ and $C^- \in \mathbb{N}^{\mathbb{P}\times\mathbb{T}}$ be its incidence matrices. Let $\overrightarrow{m_f}$ be the target marking vector, and $\overrightarrow{r_f}$ be the target residual durations vector. The integer linear programming model $IP(\mathsf{V})$ is defined by:

$$\forall k \in [\![0, \mathrm{M}-1]\!], \quad e_{m0k} = \overrightarrow{m_0}(k) \tag{1}$$
$$\forall k \in [\![0, \mathrm{M}-1]\!], \quad e_{m\mathsf{V}k} = \overrightarrow{m_f}(k) \tag{2}$$
$$\forall j \in [\![0, \mathrm{N}-1]\!], \quad e_{r0j} = \overrightarrow{r_0}(j) \tag{3}$$
$$\forall j \in [\![0, \mathrm{N}-1]\!], \quad e_{r\mathsf{V}j} = \overrightarrow{r_f}(j) \tag{4}$$

$$\forall i \in [\![0, \mathsf{V}-1]\!], \forall j \in [\![0, \mathrm{N}-1]\!], \quad B \cdot a_{ij} - e_{rij} \leq B-1 \tag{5}$$
$$\forall i \in [\![0, \mathsf{V}-1]\!], \forall j \in [\![0, \mathrm{N}-1]\!], \quad e_{rij} - B \cdot a_{ij} \leq 0 \tag{6}$$
$$\forall i \in [\![0, \mathsf{V}-1]\!], \forall j \in [\![0, \mathrm{N}-1]\!], \quad B \cdot \alpha_{ij} - e_{rij} + \Delta_{v_i} \leq B-1 \tag{7}$$
$$\forall i \in [\![0, \mathsf{V}-1]\!], \forall j \in [\![0, \mathrm{N}-1]\!], \quad e_{rij} - \Delta_{v_i} - B \cdot \alpha_{ij} \leq 0 \tag{8}$$
$$\forall i \in [\![0, \mathsf{V}-1]\!], \forall j \in [\![0, \mathrm{N}-1]\!], \quad e_{rij} - \Delta_{v_i} - \beta_{ij} \leq 0 \tag{9}$$
$$\forall i \in [\![0, \mathsf{V}-1]\!], \forall j \in [\![0, \mathrm{N}-1]\!], \quad \beta_{ij} - B \cdot \alpha_{ij} \leq 0 \tag{10}$$
$$\forall i \in [\![0, \mathsf{V}-1]\!], \forall j \in [\![0, \mathrm{N}-1]\!], \quad \beta_{ij} + B \cdot \alpha_{ij} - e_{rij} + \Delta_{v_i} \leq B \tag{11}$$

$$\forall i \in [\![1, \mathsf{V}]\!], \forall k \in [\![0, \mathrm{M}-1]\!], e_{mik} - e_{m(i-1)k} + \sum_{c=0}^{\mathrm{N}-1} C^-_{kc} \cdot \sigma_{ic} - \sum_{c=0}^{\mathrm{N}-1} C^+_{kc} \cdot (a_{(i-1)c} - \alpha_{(i-1)c}) = 0 \tag{12}$$
$$\forall i \in [\![1, \mathsf{V}]\!], \forall j \in [\![0, \mathrm{N}-1]\!], \quad e_{rij} - \beta_{(i-1)j} - \overrightarrow{d}(j) \cdot \sigma_{ij} = 0 \tag{13}$$

$$\forall i \in [\![1, \mathsf{V}]\!], \forall j \in [\![0, \mathrm{N}-1]\!], \quad \sigma_{ij} + \alpha_{(i-1)j} \leq 1 \tag{14}$$

$$\forall i \in [\![1, \mathsf{V}]\!], \forall j \in [\![0, \mathrm{N}-1]\!], \quad \sigma_{ij} \in \{0,1\} \tag{15}$$
$$\forall i \in [\![0, \mathsf{V}-1]\!], \forall j \in [\![0, \mathrm{N}-1]\!], \quad a_{ij} \in \{0,1\} \tag{16}$$
$$\forall i \in [\![0, \mathsf{V}-1]\!], \forall j \in [\![0, \mathrm{N}-1]\!], \quad \alpha_{ij} \in \{0,1\} \tag{17}$$
$$\forall i \in [\![0, \mathsf{V}-1]\!], \forall j \in [\![0, \mathrm{N}-1]\!], \quad \beta_{ij} \in \mathbb{N} \tag{18}$$
$$\forall i \in [\![0, \mathsf{V}]\!], \forall j \in [\![0, \mathrm{N}-1]\!], \quad e_{rij} \in \mathbb{N} \tag{19}$$
$$\forall i \in [\![0, \mathsf{V}]\!], \forall k \in [\![0, \mathrm{M}-1]\!], \quad e_{mik} \in \mathbb{N} \tag{20}$$
$$\forall i \in [\![0, \mathsf{V}-1]\!], \quad \Delta_{v_i} \in \mathbb{N} \tag{21}$$

Figure 4: Integer linear programming model

The *physical sense* of the equations is explained below:

- The quantity $\sum_{t\in\mathbb{T}} d(t) \cdot \overrightarrow{\sigma_{v+\Delta_v}}(t) \cdot \overrightarrow{e_t}$ represents the *new residual durations* coming from the execution of the step $\sigma_{v+\Delta_v}$ at the date $v + \Delta_v$;
- The quantity $\left(\overrightarrow{e_r}(v) - \Delta_v \cdot \overrightarrow{Id}\right)^+$ represents the *update* of the residual durations vector at the date $v + \Delta_v$, from its value $\overrightarrow{e_r}(v)$ at the date v. The *"+"* operator allows to take into account only positive values;
- The quantity $C^- \cdot \overrightarrow{\sigma_{v+\Delta_v}}$ represents the number of tokens removed from places upstream to the transitions of the step $\sigma_{v+\Delta_v}$, the execution of which starts at the date $v + \Delta_v$;
- Finally, the quantity $\overrightarrow{e_r}(v)^s - \left(\overrightarrow{e_r}(v) - \Delta_v \cdot \overrightarrow{Id}\right)^s$ represents the Parikh vector of the transitions, the firing of which ends at the date $v + \Delta_v$. This expression is made from the comparison between the Parikh vector of the transitions that were pending at the date v (vector $\overrightarrow{e_r}(v)^s$), and the Parikh vector of the transitions that *will be* active at the date $v + \Delta_v$ (vector $\left(\overrightarrow{e_r}(v) - \Delta_v \cdot \overrightarrow{Id}\right)^s$).

3.2 Formulation of *"+"* and *"s"* operators

The operators *"+"* and *"s"* introduced in the previous section acts on *vectors objects*. However, they acts uniformly on each component of the input vector, and can be defined in a natural way using the corresponding *scalar* operators. (Bourdeaud'huy & Hanafi 2006) propose to address the calculation of values X^s and X^+ associated to an integer $X \in \mathbb{Z}$, such that:

$$\begin{cases} X > 0 & \Rightarrow \quad X^s = 1 \text{ and } X^+ = X \\ X \leq 0 & \Rightarrow \quad X^s = 0 \text{ and } X^+ = 0 \end{cases}$$

The linearization is based on the use of 2 additional discrimination variables (α and β) and 5 additional equations. Let $X \in \mathbb{Z}$ and $B \in \mathbb{N}^*$ be *"sufficiently large"*. Let $\alpha \in \{0,1\}$ and $\beta \in \mathbb{N}$ such that:

$$\begin{cases} B \cdot \alpha - X & \leq & B-1 \\ X - B \cdot \alpha & \leq & 0 \\ X - \beta & \leq & 0 \\ \beta - B \cdot \alpha & \leq & 0 \\ \beta + B \cdot \alpha - X & \leq & B \end{cases}$$

$$\text{Then: } \begin{cases} X > 0 \Leftrightarrow & \alpha = 1 \\ \text{and} & \beta = X \\ X \leq 0 \Leftrightarrow & \alpha = 0 \\ \text{and} & \beta = 0 \end{cases}$$

Using additional variables and constraints, it is thus possible to propose a general linear mathematical model to solve the Timed PNs reachability problem.

The full model denoted as $IP(\mathsf{V})$ is given in figure 4. Equations (1) to (4) correspond to conditions over initial and final states. Equations (5) to (11) express the constraints over discrimination variables used to compute the *"+"* and *"s"* operators. Variables (a_i), (α_i) and (β_i) denote respectively the values of $\overrightarrow{e_r}(v_i)^s$, $\left(\overrightarrow{e_r}(v) - \Delta_v \cdot \overrightarrow{Id}\right)^s$ and $\left(\overrightarrow{e_r}(v) - \Delta_v \cdot \overrightarrow{Id}\right)^+$ from equations (22) and (23). Equations (12) and (13) correspond to intermediate state computation equations (22) and (23). . Equation (14) is used to forbid reentrant steps: if transition j is already active at date v_i (ie. $\alpha_{(i-1)j} = 1$), it cannot be fired again at the same date (ie. σ_{ij} cannot be equal to 1). Finally, equations (15) to (21) define the domain of variables.

The above model has been solved by its authors using integer linear programming with promising results, to handle scheduling problems searching to obtain the final marking while minimizing the total duration of the firing sequence. In our work, we are more particularly interested in safety issues:

- Finding as soon as possible a firing sequence representing a *counter example* to a safety specification;
- Enumerating all paths between two markings, in order to verify a safety property is always true.

Thus, we propose to use *constraint programming* to solve the previous model. Of course, one could use constraint programming solvers to solve the linear model given in Fig. 4 directly. However, constraint programming allows to use *reification techniques* in order to avoid the use of linearization variables. In the next section, we show how to use such reified constraints and give some improvements allowing to reduce the search space.

4 TOWARDS A CONSTRAINT PROGRAMMING MODEL

4.1 Reified Model

In (Bourdeaud'huy & Hanafi 2006) authors proposed to add some intermediate variables to express the impact of *"finishing transitions"* on the state vectors $\overrightarrow{e_m}(v)$ and $\overrightarrow{e_r}(v)$ in a linear way, using + and s operators in equations (22) and (23).

In this section, we propose to express the non-linear parts directly, by reifing the constraints corresponding to these equations.

Since reentrance is forbidden, a transition t_j which was *already firing* at date v_i cannot start a new fire at step v_{i+1} if it has not finished its previous firing before. Thus, at date v_{i+1}, the corresponding component of the residual durations vector, $e_{r(i+1)j}$, can have two values:

$$e_{rij} - \Delta_{v_i} > 0 \Rightarrow \begin{cases} e_{r(i+1)j} = e_{rij} - \Delta_{v_i} \\ \sigma_{(i+1)j} = 0 \end{cases} \quad (41)$$

$$e_{rij} - \Delta_{v_i} \leq 0 \Rightarrow e_{r(i+1)j} = d(j) \cdot \sigma_{(i+1)j} \quad (42)$$

Equation (41) means that t_j is still firing and can't be fired anymore, so $e_{r(i+1)j}$ is equal to the residual time to elapse. Following equation (42), if t_j is not firing (or has finished) at date v_{i+1}, $e_{r(i+1)j}$ is decided by the new step $\sigma_{(i+1)j}$.

In our reified model, the set of equations (41) and (42) replace equation (22).

To express the quantity of tokens added to to $e_{m(i+1)}$ by transitions that have finished between date v_i and v_{i+1}, we add intermediate variables $TF_{(i+1)j}$ to denote these transitions, using the following set of reified constraints:

$\forall j \in [\![1, \mathrm{N}]\!]$,

$$\begin{cases} e_{rij} > 0 \\ \text{and } e_{rij} - \Delta_{v_i} \leq 0 \end{cases} \Rightarrow TF_{(i+1)j} = 1 \quad (43)$$

$$\begin{cases} e_{rij} \leq 0 \\ \text{or } e_{rij} - \Delta_{v_i} > 0 \end{cases} \Rightarrow TF_{(i+1)j} = 0 \quad (44)$$

Equations (43) denote transitions j that were firing at date v_i and have finished at date v_{i+1}. Conversely, equations (44) denote transitions that were not firing at date v_i or that have finished at date v_{i+1}

Equation (23) is then replaced in the previsous mathematical programming model by the new equation:

$$\overrightarrow{e_m}(v + \Delta_v) = \overrightarrow{e_m}(v) - C^- \cdot \overrightarrow{\sigma_{v+\Delta_v}} + C^+ \cdot \overrightarrow{TF_{v+\Delta_v}} \quad (45)$$

4.2 Improvements of Model

In order to reduce the search space and improve the efficiency of our CP model, we reduce the domain of variables and add more constraints, according to some observations made on the considered TPNs.

Firing Policy Since all transitions have a finite duration, to fire the next step σ_{i+1}, it is not necessary to *wait* after the firing of σ_i at date v_i more

Let $(R, \overrightarrow{m_0}, \overrightarrow{r_0}, \overrightarrow{d})$ be a Timed Petri Net with $\mathbb{P} = \{p_1, \ldots, p_{\mathrm{M}}\}$, $\mathbb{T} = \{t_1, \ldots, t_{\mathrm{N}}\}$. Let $C^+ \in \mathbb{N}^{\mathbb{P}\times\mathbb{T}}$ and $C^- \in \mathbb{N}^{\mathbb{P}\times\mathbb{T}}$ be its incidence matrices. Let $\overrightarrow{m_f}$ be the target marking vector, and $\overrightarrow{r_f}$ be the target residual durations vector. The constraint programming model $CP(\mathsf{V})$ is defined by:

$$\forall k \in [\![0, \mathrm{M}-1]\!], \quad e_{m0k} = \overrightarrow{m_0}(k) \tag{24}$$
$$\forall k \in [\![0, \mathrm{M}-1]\!], \quad e_{m\mathsf{V}k} = \overrightarrow{m_f}(k) \tag{25}$$
$$\forall j \in [\![0, \mathrm{N}-1]\!], \quad e_{r0j} = \overrightarrow{r_0}(j) \tag{26}$$
$$\forall j \in [\![0, \mathrm{N}-1]\!], \quad e_{r\mathsf{V}j} = \overrightarrow{r_f}(j) \tag{27}$$

$$\forall i \in [\![0, \mathsf{V}-1]\!], \forall j \in [\![0, \mathrm{N}-1]\!], \quad e_{rij} - \Delta_{v_i} > 0 \Rightarrow e_{r(i+1)j} = e_{rij} - \Delta_{v_i}, \sigma_{(i+1)j} = 0 \tag{28}$$

$$\forall i \in [\![0, \mathsf{V}-1]\!], \forall j \in [\![0, \mathrm{N}-1]\!], \quad e_{rij} - \Delta_{v_i} \leq 0 \Rightarrow e_{r(i+1)j} = \overrightarrow{d}(j) \cdot \sigma_{(i+1)j} \tag{29}$$

$$\forall i \in [\![0, \mathsf{V}-1]\!], \forall j \in [\![0, \mathrm{N}-1]\!], \quad e_{rij} > 0 \text{ and } e_{rij} - \Delta_{v_i} \leq 0 \Rightarrow TF_{(i+1)j} = 1 \tag{30}$$

$$\forall i \in [\![0, \mathsf{V}-1]\!], \forall j \in [\![0, \mathrm{N}-1]\!], \quad e_{rij} \leq 0 \text{ or } e_{rij} - \Delta_{v_i} > 0 \Rightarrow TF_{(i+1)j} = 0 \tag{31}$$

$$\forall i \in [\![0, \mathsf{V}-1]\!], \forall k \in [\![0, \mathrm{M}-1]\!], \quad e_{m(i+1)k} - e_{mik} = \sum_{c=0}^{\mathrm{N}-1} C^+_{kc} \cdot TF_{(i+1)c} - \sum_{c=0}^{\mathrm{N}-1} C^-_{kc} \cdot \sigma_{(i+1)c} \tag{32}$$

$$\forall i \in [\![1, \mathsf{V}-1]\!], \quad \sum_{j=0}^{\mathrm{N}-1} \sigma_{ij} \neq 0 \tag{33}$$

$$\forall i \in [\![0, \mathsf{V}-1]\!], \quad \Delta_{v_i} \leq \max_{j \in [\![1,\mathrm{N}]\!]} e_{rij} \tag{34}$$
$$\forall i \in [\![1, \mathsf{V}-1]\!], \quad \Delta_{v_i} \neq 0 \tag{35}$$

$$\forall i \in [\![1, \mathsf{V}]\!], \forall j \in [\![0, \mathrm{N}-1]\!], \quad \sigma_{ij} \in \{0, 1\} \tag{36}$$
$$\forall i \in [\![1, \mathsf{V}]\!], \forall j \in [\![0, \mathrm{N}-1]\!], \quad TF_{ij} \in \{0, 1\} \tag{37}$$
$$\forall i \in [\![0, \mathsf{V}]\!], \forall j \in [\![0, \mathrm{N}-1]\!], \quad e_{rij} \in \{0, D_{max}\} \tag{38}$$
$$\forall i \in [\![0, \mathsf{V}]\!], \forall k \in [\![0, \mathrm{M}-1]\!], \quad e_{mik} \in \{0, M_{max}\} \tag{39}$$
$$\forall i \in [\![0, \mathsf{V}-1]\!], \quad \Delta_{v_i} \in \{0, D_{max}\} \tag{40}$$

Figure 5: Constraint programming model

than the time needed to finish all currently firing transitions, i.e. the maximum component of the residual duration e_{ri}, denoted as $\max_{j \in [\![1,\mathrm{N}]\!]}(\overrightarrow{e_{rij}})$. Indeed: we are sure that all transitions previously active at date v_i have finished, thus all tokens needed to fire σ_{i+1} are available at date $v_i + \max_{j \in [\![1,\mathrm{N}]\!]}(\overrightarrow{e_{ri}})$.

Using the same considerations, variables Δ_{v_i} and e_{rij} will never be greater than D_{max}, denoting the longest duration. Such considerations lead to the following additional constraints:

$$\forall i \in [\![0, \mathsf{V}]\!], \forall j \in [\![0, \mathrm{N}-1]\!], \; e_{rij} \in \{0, D_{max}\} \tag{46}$$
$$\forall i \in [\![0, \mathsf{V}-1]\!], \; \Delta_{v_i} \in \{0, D_{max}\} \tag{47}$$
$$\forall i \in [\![0, \mathsf{V}-1]\!], \; \Delta_{v_i} \leq \max_{j \in [\![1,\mathrm{N}]\!]}(\overrightarrow{e_{ri}}) \tag{48}$$

In order to reduce the search space, we forbid the firing of *empty* steps, i.e. steps containing no transition firing, which were allowed in the previous work. This avoid to find solution built from another one by simply adding an empty step. Only the last steps σ_{V} is allowed to be empty, in order to be able to consider the date V for which the final marking is reached but no transition is still active. Note this implies to choose the total number of steps accurately since if one choose more steps than the number needed to reach the final marking, no solution can be found. To the contrary, empty steps allow to use dichotomic search to find the minimal number of steps needed to reach the final marking.

In order to avoid solutions built one from another by splitting steps into individual transition firings, we forbid the elapsed time Δ_i between steps σ_i and σ_{i+1} to be equal to 0, except from first firing date Δ_0. Thus, if a solution exists where two transitions t_i, t_j must be fired simultaneously at date v_i, they will be fired in an unique step $\overrightarrow{\sigma_i} = \overrightarrow{t_j} + \overrightarrow{t_k}$ and not using two steps $\overrightarrow{\sigma_i} = \overrightarrow{t_j}$ and $\overrightarrow{\sigma_{i+1}} = \overrightarrow{t_k}$ with $\Delta_i = 0$.

Such improvements reduce the search space by deleting solutions that will be found anyway. The corresponding constraints are given in Equations (49) and (50).

$$\forall i \in [\![1, \mathsf{V}-1]\!], \sum_{j=0}^{\mathrm{N}-1} \sigma_{ij} \neq 0 \tag{49}$$

$$\forall i \in [\![1, \mathsf{V}-1]\!], \Delta_{v_i} \neq 0 \quad (50)$$

Structural Properties Since PNs express the behaviour of physical systems, they have generally an infinite behaviour, made of cyclic executions, but their number of states is *bounded*. Thus, it is possible to compute an upper bound of the marking of each place, allowing to reduce the domain of variables e_{mik} in the considered model, as shown in equation (51). Such bounds can be computed using P-flows of the net, see (Krukeberg & Jaxy 1987). For instance, the number of tokens in each place of the TPNs of (Fig. 2) is bounded by 1, which is the max number in initial marking.

$$\forall i \in [\![0, \mathsf{V}]\!], \forall k \in [\![0, \mathrm{M}-1]\!],\ e_{mik} \in \{0, M_{max}\} \quad (51)$$

The full model denoted as $CP(\mathsf{V})$ is given in figure 5. Equations (24) to (27) correspond to conditions over initial and final states. Equations (28) to (31) correspond to the reified constraints introduced above. Equation (32) is used to get the marking at each step. Equations (33) to (35) are used to reduce the search space: avoid *empty* steps, forbid redundant solutions with *empty wait* and split steps. Finally, equations (36) to (40) define the domains of variables.

In the next section, we give numerical experiments in order to compare our new model to the integer programming model mentioned in (Bourdeaud'huy & Hanafi 2006).

5 NUMERICAL EXPERIMENTS

Experiments were carried out using a 2.93 Ghz Pentium with 4 Gb of RAM, using the constraint programming tool *Ilog Solver*. We have used two Timed PNs reachability problems to verify the efficiency and the robustness of the CP model with comparison to the IP model, which had been improved following section 4.2.

The first TPN corresponds to the scheduling problem given in Fig. 2. The goal was to to get solutions leading first from $m_0^1 = \{p_1, p_2, p_3, m_1, m_2, m_3\}$ to $m_f^1 = \{p_4, p_5, p_6, m_1, m_2, m_3\}$, then from $m_0^2 = \{2 \cdot p_1, 2 \cdot p_2, 2 \cdot p_3, m_1, m_2, m_3\}$ to $m_f^2 = \{2 \cdot p_4, 2 \cdot p_5, 2 \cdot p_6, m_1, m_2, m_3\}$.

The second one is given in Fig. 6. It represents philosophers around a table, who spend time eating spaghettis and thinking. To eat, each one needs two forks, but there is only one available for two people. Each philosopher is provided with a control place, allowing us to quantify how many times it has been active. The presence of this unbounded place makes the corresponding reachability graph unbounded.

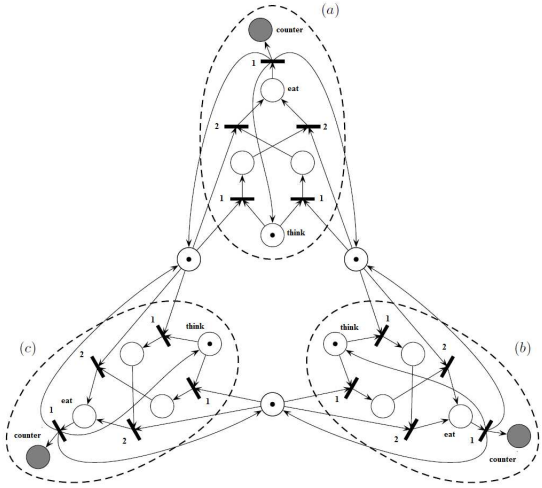


Figure 6: Dining philosophers problem

The corresponding reachability problem was to find firing sequences allowing to make each philosopher eat a given number of times (2 then 4). Such problem is characterized by the existence of *deadlocks*, i.e. states from where no transition is fireable. Such situation can be obtained if each philosopher decide to take the fork on his left, for example. The presence of *deadlocks* make the problem harder to solve.

The results are given in tables 2 to 5. We give the time to obtain all solutions and first feasible solution, the number of inconsistent choices (*fails*), the number of solutions, variables and constraints for each problem.

As can be seen in tables 2 and 3, our reified model behaves better than the IP one for the first example, particularly when the search space is larger. CP model is more efficient than IP model although they are meeting the same number of fails when searching with the same number of firing dates.

In tables 4 and 5, the philosophers problem shows that CP model search for solutions a little bit slower than IP model, although it meets far more fails during search. More experiments must be done to understand the reason why the reified model doest not behave better. This could be due to the existence of deadlocks.

The use of reifed constraints allows us to reduce the number of variables and constraints of the model. One should note that the first solution given by each model using the default labeling strategy is the same: linearization variables were not choosen to be labeled in the search tree, they have been instanciated thanks to the enumeration of other variables. The default strategy was to label variables in the order they occur in the sequence $\sigma_1 \Delta_{v_0} e_{m_1} e_{r_1} \sigma_2 \Delta_{v_1} e_{m_2} e_{r_2} \ldots \sigma_{\mathsf{V}} \Delta_{v_{\mathsf{V}-1}} e_{m_{\mathsf{V}}} e_{r_{\mathsf{V}}}$.

Table 2: Scheduling problem - finding all solutions (m_0^1 to m_f^1)

V	Time	Fails	variables	constraints	Nb of solutions
CP model					
6	0 s	36	333	0	33
7	0.094 s	567	381	0	1479
8	0.702 s	3884	429	0	23280
9	4.337 s	17525	477	0	165088
10	14.227 s	72673	525	0	538288
11	22.105 s	238690	573	0	656524
12	15.85 s	447419	621	0	0
IP model					
6	0 s	36	503	0	33
7	0.125 s	567	581	0	1479
8	1.060 s	3884	659	0	23280
9	6.536 s	17525	737	0	165088
10	21.918 s	72673	815	0	538288
11	33.977 s	238690	893	0	656524
12	21.934 s	447419	971	0	0

Table 3: Scheduling problem - finding first feasible solution (m_0^2 to m_f^2)

V	Time	Fails	variables	constraints	makespan
CP model					
9	27.160 s	545210	477		no solution
10	8.486 s	171494	525	885	30
11	1.950 s	39341	573	974	30
12	0.577 s	10817	621	1057	34
13	0.172 s	1944	669	1147	34
14	0.125 s	1076	717	1235	38
15	0.047 s	77	765	1317	40
16	0.078 s	31	813	1408	41
17	0.031 s	26	861	1500	41
IP model					
9	33.821 s	532375	737		no solution
10	10.483 s	167508	815	1179	30
11	2.465 s	38646	893	1297	30
12	0.686 s	10650	971	1415	34
13	0.156 s	1919	1049	1533	34
14	0.094 s	1062	1127	1651	38
15	0.109 s	78	1205	1769	40
16	0.031 s	28	1283	1887	41
17	0.062 s	20	1361	2005	41

Table 4: Dining philosophers problem - finding all solutions (target counter =2)

V	Time	Fails	variables	constraints	Nb of solutions
CP model					
13	1.856 s	22084	907	0	0
14	6.224 s	76286	972	0	2880
15	19.797 s	222456	1037	0	47328
16	55.458 s	435438	1102	0	291552
IP model					
13	1.716 s	17054	1477	0	0
14	6.006 s	56522	1587	0	2880
15	20.374 s	165514	1697	0	47328
16	61.808 s	356742	1807	0	291552

Table 5: Dining philosophers problem - finding first feasible solution (target counter =4)

V	Time	Fails	variables	constraints	makespan
CP model					
29	15.351 s	216325	1947	3610	40
30	3.136 s	42257	2012	3735	41
31	2.434 s	30001	2077	3860	42
32	1.170 s	13671	2142	3985	43
IP model					
29	14.399 s	175378	3237	4929	40
30	2.964 s	35238	3347	5099	41
31	2.152 s	24086	3457	5269	42
32	1.061 s	10519	3567	5439	43

6 CONCLUSION AND FUTURE WORK

In this paper, we are interested in solving the Timed Petri Nets Reachability Problem using Constraint Programming Approach.

Our goal is to develop efficient methodologies allowing to address safety issues in manufacturing and transport systems by enumerating firing sequences. This work is the first step towards this direction.

We propose to use an incremental model allowing to capture the timed behaviour of the TPN in a linear system of equations. A dedicated constraint programming model is proposed, using reified constraints, to find solutions efficiently. This model is evaluated on academic examples with promising results and behaves better than the original one.

In the future, we propose to follow interesting directions raised by this first work:

- Complete numerical experiments to understand for which classes of problems each model is better fitted;
- Improve the obtained model adding bounds, valid inequalities, global constraints traducing structural properties of the PN;
- Adapt CP exploration techniques to the specificities of the considered problem by designing filtering algorithms, improving labeling techniques. One idea could be to search only for one solution in each equivalence class defined by a preliminary analysis of the underlying PN, without considering times, thus reducing the search space.

Finally, it must be said that the mathematical model described in (Fig. 5) leads only to a semi-decision algorithm. Indeed, in the context of unbounded TPNs, the number of firing dates V is set arbitrarily, as we do not know any information on the number of steps needed to find a possible solution. Thus, if no solution

is obtained using this value V, one cannot conclude on the reachability property. To the contrary, when dealing with bounded TPNs, it is possible to set V to the value of the *"sequential depth"* of the net, a parameter defined in (Bourdeaud'huy, Hanafi & Yim 2004) and which guarantee the complete exploration of the reachability graph. Using this parameter as search depth, it is always possible to conclude when the algorithm stops. A promising research track could be to define efficient procedures to evaluate the value of this parameter.

References

Bourdeaud'huy, T. & Hanafi, S. (2006). Scheduling of flexible manufacturing systems using timed petri nets and mathematical programming, *Proceedings of the 8th International Workshop on Discrete Event Systems* **3**: 94–99.

Bourdeaud'huy, T., Hanafi, S. & Yim, P. (2004). Efficient reachability analysis of bounded petri nets using constraint programming, *SMC'04, International Conference on Systems, Man and Cybernetics*, La Hague, Hollande.

Carlier, J. & Chretienne, P. (1988). Timed petri net schedules, *Advances in Petri Nets 1988* pp. 62–84.

Chretienne, P. (1984). Executions controls dans les reseaux de petri temporises, *T.S.I.* **3**.

Driss, O., Yim, P., Korbaa, O. & Ghedira, K. (2004). Reachability search in limed petri nets using constraint programming, pp. 4923–4928.

Krukeberg, F. & Jaxy, M. (1987). Mathematical methods for calculating invariants in petri nets, *in* G. Rozenberg (ed.), *Advances in Petri Nets 1987*, Vol. 266 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 104–131.

Michel, F. & Vernadat, F. (1998). Maitrise de l'explosion combinatoire. reduction du graphe de comportement, *RAIRO Technique et Science Informatiques* **17**(7): 805–837.

Murata, T. (1989). Petri nets: Properties, analysis and applications, *Proceedings of the IEEE* **77**(4): 541–580.

Petri, C. (1962). *Kommunikation mit Automaten*, PhD thesis, University of Bonn, Bonn, West Germany.

Recalde, L., Silva, M., Ezpeleta, J. & Teruel, E. (2004). Petri nets and manufacturing systems: An examples-driven tour, *Lectures on Concurrency and Petri Nets* pp. 71–89.

Richard, P. (1997). *Contribution des reseaux de Petri a l'etude de problemes de recherche operationnelle*, PhD thesis, Universit?de Tours.

Wang, J. (1998). *Timed Petri Nets, Theory and Application*, Kluwer Academic Publishers.

Zhou, M. & Venkatesh, K. (1999). *Modeling, Simulation, and Control of Flexible Manufacturing Systems: A Petri Net Approach*.