

Simulation and Analysis of Hybrid Petri Nets

Using the Matlab Tool HYPENS

Alessandro Giua, Carla Seatzu, Fausto Sessego

Dept. of Electrical and Electronic Engineering, University of Cagliari, Italy

Email: {giua,seatzu,fausto.sessego}@diee.unica.it

Abstract

In this paper we consider a software tool for the simulation and analysis of discrete, continuous and hybrid Petri nets, called HYPENS. It is an open source tool we have recently developed in Matlab.

The main contribution of this paper is that of demonstrating the efficiency of HYPENS via real several non trivial examples. In particular, we consider a family of purely discrete event systems modeling a queueing network, and a job shop system with finite capacity buffers and unreliable multi-class machines. A detailed analysis of the computational times and of the accuracy of the results is proposed. In particular, in the first case example, that is modeled using discrete Petri nets, the correctness of the results is proved using well known analytical results from the queueing theory.

Published as:

F. Sessego, A. Giua, C. Seatzu, "Simulation and Analysis of Hybrid Petri Nets using the Matlab Tool HYPENS," *SMC08: 2008 IEEE Int. Conf. on Systems, Man, and Cybernetics* (Singapore), October 2008.

In this paper we consider a Matlab tool for the simulation and analysis of discrete, continuous and hybrid Petri nets, called HYPENS: *Hybrid Petri net Simulator*, that we first presented in [9].

In many applications dealing with complex systems, a plant has a discrete event dynamics whose number of reachable states is typically very large, and problems of realistic scale quickly become analytically and computationally untractable. To cope with this problem it is possible to give a continuous approximation of the "fast" discrete event dynamics by means of *continuous Petri nets*, i.e., nets obtained from discrete nets by "fluidification" [4], [10].

In general, different fluid approximations are necessary to describe the same system, depending on its discrete state. Thus, the resulting models can be better described as *hybrid Petri nets* (HPN) that combine discrete and continuous dynamics [4], [5]. Several HPN models have been defined (see [5]). The model considered in HYPENS is called *First-Order Hybrid Petri nets* (FOHPN) because its continuous dynamics are piece-wise constant. FOHPN were originally presented in [3] and have been successfully used to model and analyze manufacturing systems [2].

In the last years several tools for the simulation of timed discrete PN have been proposed. Very few tools on the contrary, also deal with hybrid PN: we are aware of *HISim* [6] and *SIRPHYCO* [8]. A detailed comparison among them is proposed in [9].

The main interesting features of HYPENS can be summarized as follows: it enables us to simulate and analyze within the same framework discrete, continuous and hybrid Petri nets; it has been developed in Matlab, thus allowing the user to take advantage of several functions and structures already present in Matlab; in the discrete subnet both finite and infinite server semantics can be implemented. Finally, for the *conflict resolution* of discrete transitions we use a general approach that combines both priorities and random weighted choices as in [1]: this is coded in the structure of the net. In the case of continuous transitions, on the contrary, we assume that the choice of the firing speed vector \mathbf{v} is made at run-time and represents a *control input*. Thus for solving conflicts among continuous transitions we use a more general optimization rule: at each step the net evolves so as to (myopically) optimize a linear performance index $J(\mathbf{v})$ [9].

In this paper we propose a detailed analysis of the efficiency of HYPENS via numerical simulations. In particular, we first consider a purely discrete Petri net modeling a queueing network, whose number of elementary modules (namely, the number of queues) vary, depending on certain parameters. Different simulations have been carried out for different values of such parameters, and the corresponding computer time is computed. Although HYPENS can simulate general stochastic networks, we considered in this

example markovian networks, so that the obtained results can be compared with those predicted using well known results from the queueing theory.

Finally, we consider a job shop system taken from the literature [2] and modeled using First-Order Hybrid Petri nets. We show the results of some numerical simulations and analysis carried out using HYPENS.

II. HYBRID PETRI NETS

We recall the FOHPN formalism used by HYPENS, following [3].

A. Net structure

A FOHPN is a structure $N = (P, T, Pre, Post, \mathcal{D}, \mathcal{C})$.

The set of *places* $P = P_d \cup P_c$ is partitioned into a set of *discrete* places P_d (represented as circles) and a set of *continuous* places P_c (represented as double circles). The cardinality of P , P_d and P_c is denoted n , n_d and n_c .

The set of *transitions* $T = T_d \cup T_c$ is partitioned into a set of discrete transitions T_d and a set of continuous transitions T_c (represented as double boxes). The cardinality of T , T_d and T_c is denoted q , q_d and q_c .

The *pre-* and *post-incidence functions* that specify the arcs are (here $\mathbb{R}_0^+ = \mathbb{R}^+ \cup \{0\}$): $Pre, Post : P_c \times T \rightarrow \mathbb{R}_0^+$, $P_d \times T \rightarrow \mathbb{N}$. We require that $\forall t \in T_c$ and $\forall p \in P_d$, $Pre(p, t) = Post(p, t)$, so that the firing of continuous transitions does not change the marking of discrete places.

Transitions in T_d may either be deterministic or stochastic. In the case of deterministic transitions the function $\mathcal{D} : T_d \rightarrow \mathbb{R}_0^+$ specifies the timing associated to timed discrete transitions. In the case of stochastic transitions \mathcal{D} defines the parameter(s) of the distribution function corresponding to the timing delay. The function $\mathcal{C} : T_c \rightarrow \mathbb{R}_0^+ \times \mathbb{R}_\infty^+$ specifies the firing speeds associated to continuous transitions (here $\mathbb{R}_\infty^+ = \mathbb{R}^+ \cup \{\infty\}$). For any continuous transition $t_j \in T_c$ we let $\mathcal{C}(t_j) = [V_j', V_j]$, with $V_j' \leq V_j$: V_j' represents the *minimum firing speed* (mfs), V_j represents the *maximum firing speed* (MFS).

The *incidence matrix* of the net is defined as $C(p, t) = Post(p, t) - Pre(p, t)$. The restriction of \mathbf{C} (Pre , $Post$, resp.) to P_x and T_y , with $x, y \in \{c, d\}$, is denoted \mathbf{C}_{xy} (Pre_{xy} , $Post_{xy}$, resp.).

A *marking* is a function that assigns to each discrete place a non-negative number of tokens, and to each continuous place a fluid volume. Therefore $M : P_c \rightarrow \mathbb{R}_0^+$, $P_d \rightarrow \mathbb{N}$. The marking of place p_i is denoted M_i , while the value of the marking at time τ is denoted $M(\tau)$. The restriction of M to P_d and P_c are denoted with M^d and M^c , resp.

A *system* $\langle N, M(\tau_0) \rangle$ is an FOHPN N with an initial marking $M(\tau_0)$.

The enabling of a discrete transition depends on the marking of all its input places, both discrete and continuous. More precisely, a discrete transition t is *enabled* at M if for all $p_i \in \bullet t$, $M_i \geq \text{Pre}(p_i, t)$, where $\bullet t$ denotes the preset of transition t . The *enabling degree* of t at M is equal to $\text{enab}(M, t) = \max\{k \in \mathbb{N} \mid M \geq k \cdot \text{Pre}(\cdot, t)\}$.

If t is *infinite-server semantics*, we associate to it a number of clocks that is equal to $\text{enab}(M, t)$. Each clock is initialized to a value that is equal to the time delay of t , if t is deterministic, or to a random value depending on the distribution function of t , if t is stochastic. If a discrete transition is *k-server semantics*, then the number of clocks that are associated to t is equal to $\min\{k, \text{enab}(M, t)\}$. The values of clocks associated to t decrease linearly with time, and t fires when the value of one of its clocks is null (if \bar{k} clocks reach simultaneously a null value, then t fires \bar{k} times). Note that here we are considering *enabling memory policy*, not *total memory policy*. This means that if a transition enabling degree is reduced by the firing of a different transition, then the disabled clocks have no memory of this in future enabling [1], [4].

If a discrete transition t_j fires k times at time τ , then its firing at $M(\tau^-)$ yields a new marking $M(\tau)$ such that $M^c(\tau) = M^c(\tau^-) + \mathbf{C}_{cd}\boldsymbol{\sigma}$, and $M^d(\tau) = M^d(\tau^-) + \mathbf{C}_{dd}\boldsymbol{\sigma}$, where $\boldsymbol{\sigma} = k \cdot \mathbf{e}_j$ is the *firing count vector* associated to the firing of transition t_j k times.

To every continuous transition t_j is associated an instantaneous firing speed (IFS) $v_j(\tau)$. For all τ it should be $V'_j \leq v_j(\tau) \leq V_j$, and the IFS of each continuous transition is piecewise constant between events.

A continuous transition is enabled only by the marking of its input discrete places. The marking of its input continuous places, however, is used to distinguish between *strong* and *weakly enabled*: if all input continuous places of t_j have a not null marking, then t_j is called *strongly enabled*, else t_j is called *weakly enabled*¹.

We can write the equation which governs the evolution in time of the marking of a place $p_i \in P_c$ as $\dot{m}_i(\tau) = \sum_{t_j \in T_c} C(p_i, t_j)v_j(\tau)$ where $\mathbf{v}(\tau) = [v_1(\tau), \dots, v_{n_c}(\tau)]^T$ is the IFS vector at time τ .

The enabling state of a continuous transition t_j defines its admissible IFS v_j . If t_j is not enabled then $v_j = 0$. If t_j is strongly enabled, then it may fire with any firing speed $v_j \in [V'_j, V_j]$. If t_j is weakly enabled, then it may fire with any firing speed $v_j \in [V'_j, \bar{V}_j]$, where $\bar{V}_j \leq V_j$ since t_j cannot remove more fluid from any empty input continuous place \bar{p} than the quantity entered in \bar{p} by other transitions. Linear inequalities can be used to characterize the set of *all* admissible firing speed vectors \mathcal{S} . Each vector $\mathbf{v} \in \mathcal{S}$ represents a

¹We are using an enabling policy for continuous transitions slightly different from the one proposed by David and Alla [4]. See [2] for a detailed discussion.

particular mode of operation of the system described by the net, and among all possible modes of operation, the system operator may choose the best, i.e., the one that maximizes a given performance index $J(\mathbf{v})$ [2].

We say that a *macro-event* (ME) occurs when: (a) a discrete transition fires, thus changing the discrete marking and enabling/disabling a continuous transition; (b) a continuous place becomes empty, thus changing the enabling state of a continuous transition from strong to weak; (c) a continuous place, whose marking is increasing (decreasing), reaches a flow level that increases (decreases) the enabling degree of discrete transitions.

Let τ_k and τ_{k+1} be the occurrence times of two consecutive ME as defined above; we assume that within the interval of time $[\tau_k, \tau_{k+1})$, denoted as a *macro-period* (MP), the IFS vector is constant and we denote it $\mathbf{v}(\tau_k)$. Then the continuous behavior of an FOHPN for $\tau \in [\tau_k, \tau_{k+1})$ is described by $M^c(\tau) = M^c(\tau_k) + \mathbf{C}_{cc}\mathbf{v}(\tau_k)(\tau - \tau_k)$, $M^d(\tau) = M^d(\tau_k)$.

Example 1: Consider the net system in Fig. 1.a. Place p_1 is a continuous place, while all other places are discrete. Continuous transitions t_1, t_2 have MFS $V_1 = 1, V_2 = 2$ and null mfs. Deterministic timed discrete transitions t_3, t_5 have timing delays 2 and 1.5, resp. Exponential stochastic discrete transitions t_4, t_6 have average firing rates are $\lambda_4 = 2$ and $\lambda_6 = 1.5$.

The continuous transitions represent two unreliable machines; parts produced by the first machine (t_1) are put in a buffer (p_1) before being processed by the second machine (t_2). The discrete subnet represents the failure model of the machines. When p_3 is marked, t_1 is enabled, i.e. the first machine is operational; when p_2 is marked, transition t_1 is not enabled, i.e. the first machine is down. A similar interpretation applies to the second machine.

Assume that we want to maximize the production rates of machines. In such a case, during any MP continuous transitions fire at their highest speed. This means that we want to maximize $J(\mathbf{v}) = v_1 + v_2$ under the constraints $v_1 \leq V_1, v_2 \leq V_2$, and — when p_1 is empty — $v_2 \leq v_1$.

The resulting *evolution graph* and the time evolution of M_1, v_1 and v_2 are shown in Fig. 1.b and c. During the first MP (of length 1) both continuous transitions are strongly enabled and fire at their MFS. After one time unit, p_1 gets empty, thus t_2 becomes weakly enabled and fires at the same speed of t_1 . At time $\tau = 1.5$, transition t_5 fires, disabling t_2 . ■

III. HYPENS TOOL

HYPENS has been developed in Matlab (Version 7.1). It is composed of 4 main files. The first two files, *make_HP_N.m* and *enter_HP_N.m* create the net to be simulated: the former requires input data from the workspace, while the latter is a guided procedure.

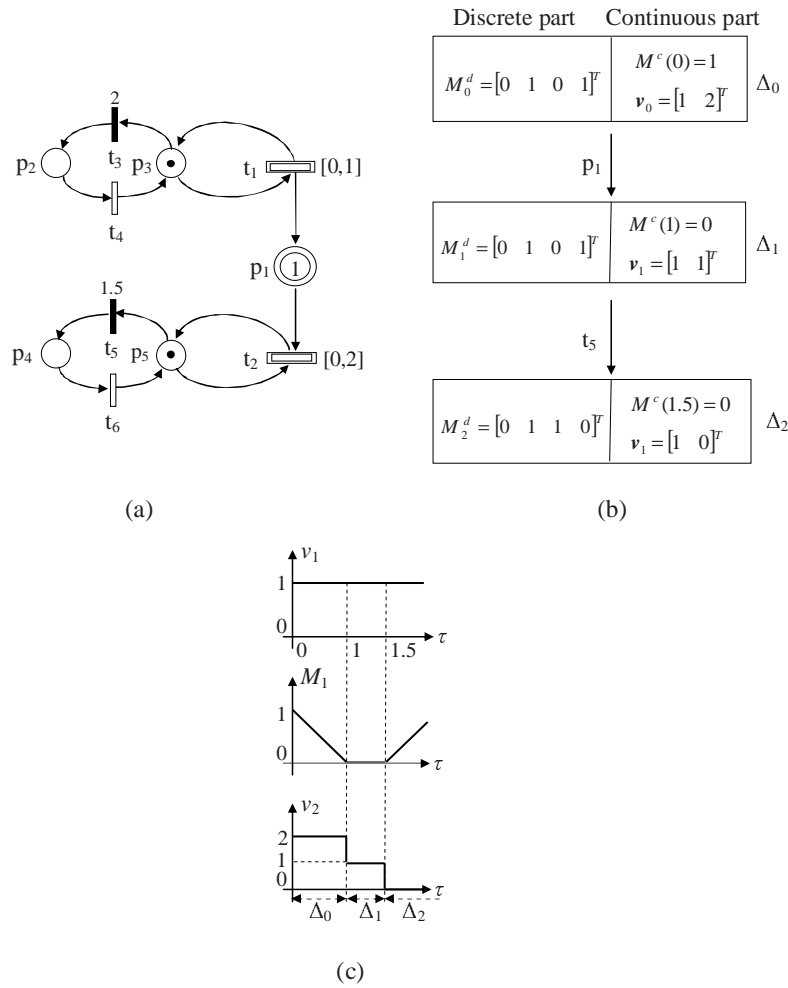


Fig. 1. (a) An FOHPN, (b) its evolution in time.

The file *simulator_HP.N.m* computes the timing evolution of the net that is summarized in an array of cells called *Evol*. This array contains the following information: *Type* (denotes if the net is continuous, discrete or hybrid), *M_Evol* (the marking of the net at each ME), *IFS_Evol* (IFS vectors during each MP), *P_macro* (the continuous places that causes a ME), *Event_macro_Evol* (the ME caused by continuous places), *firing_transition* (the discrete transition that fires at each ME), *timer_macro_event* (the length of each ME), τ (the total time of simulation at occurrence of each ME), *Q_Evol* (the values of clocks at each ME), n_c, n_d, q_c, q_d (the number of continuous/discrete places and transitions).

Based on this array, the file *analysis_HP.N.m* computes useful statistics and plots the simulation results.

For sake of brevity, here we limit to present the latter file, namely *analysis_HP.N.m*, that better enables us to appreciate the potentiality of the software. For more details on the other functions we address to the manual and demos of HYPENS that can be downloaded from [7].

$[P_{ave}, P_{max}, Pd_{ave_t}, IFS_{ave}, Td_{ave}, Pd_{freq}, Md_{freq}] = analysis_HPN (Evol, static_plot, graph, marking_plot, Td_firing_plot, up_marking_plot, Pd_prob_plot, Pd_{ave_t_plot}, IFS_plot, up_IFS_plot, IFS_{ave_plot}, Td_{freq_plot})$.

This function computes useful statistics and plots the results of the simulation run contained in *Evol*. The other input parameters are the following.

- *statistic_plot* $\in \{0, 1\}$: if 1 two histograms are created showing for each place, the maximum and the average value of the marking during the simulation.
- *graph* $\in \{0, 1\}$: if 1 the evolution graph is printed on screen.
- *marking_plot*: is a vector used to plot the marking evolution of selected places in separate figures. As an example, if we set *marking_plot* = $[x \ y \ z]$, the marking evolution of places p_x , p_y and p_z is plotted. If *marking_plot* = $[-1]$, then the marking evolution of all places is plotted.
- *Td_firing_plot* $\in \{0, 1\}$: if 1 a graph is created showing the time instants at which discrete transitions have fired.
- *up_marking_plot* is a vector used to plot the marking evolution of selected places in a single figure. The syntax is the same as that of *marking_plot*.
- *Pd_prob_plot* $\in \{0, 1\}$: 1 means that as many plots as the number of discrete places will be visualized, each one representing the frequency of having a given number of tokens during the simulation run.
- *Pd_ave_t_plot* is a vector used to plot the average marking in discrete places with respect to time. A different figure is associated to each place, and the syntax to select places is the same as that of *marking_plot*.
- *IFS_plot* (resp., *up_IFS_plot*): is a vector used to plot the IFS of selected transitions in separate figures (resp., in a single figure). The syntax is the same as that of *marking_plot* and *up_marking_plot*.
- *IFS_ave_plot* $\in \{0, 1\}$: if 1 an histogram is created showing the average firing speed of continuous transitions.
- *Td_freq_plot* $\in \{0, 1\}$: if 1 an histogram is created showing the firing frequency of discrete transitions during the simulation run.

These are the outputs of function *analysis_HPNN.m*.

- P_{ave} (P_{max}) $\in \mathbb{R}^{1 \times n}$: each entry is equal to the average (maximum) marking of the corresponding place during the simulation run.
- $Pd_{ave_t} \in \mathbb{R}^{n_d \times K}$: column k specifies the average marking of discrete places from time $\tau_0 = 0$ to time τ_k when the k -th ME occurs. Here K is the number of ME that occur during the simulation run.
- $IFS_{ave} \in \mathbb{R}^{1 \times q_c}$: each entry is equal to the average IFS of the corresponding continuous transition.

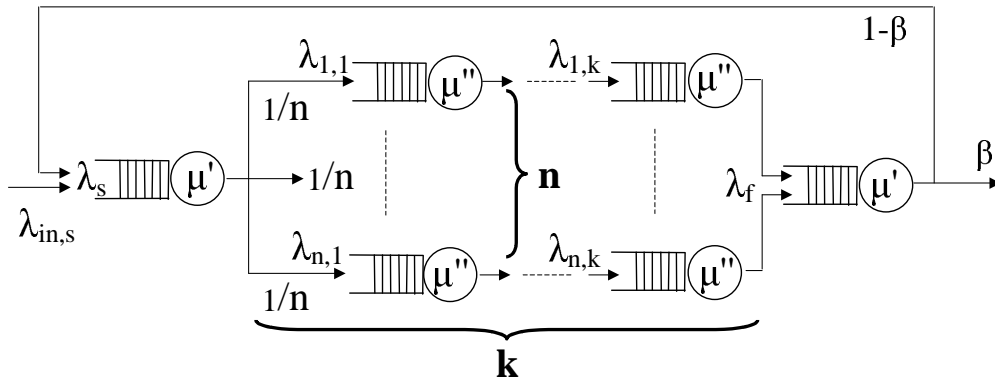


Fig. 2. The queuing network considered in Section IV.

- $Td_ave \in \mathbb{R}^{1 \times qd}$: each entry is equal to the average enabling time of the corresponding discrete transition.
- $Pd_freq \in \mathbb{R}^{n_d \times (z+1)}$ specifies for each discrete place the frequency of a given number of tokens during the simulation run. Here z is the maximum number of tokens in any discrete place during the simulation run.
- $Md_freq \in \mathbb{R}^{(n_d+1) \times \tilde{K}}$ where \tilde{K} denotes the number of different discrete markings that are reached during the simulation run. Each column of Md_freq contains one of such markings and its corresponding frequency as a last entry.

IV. A QUEUEING NETWORK

The objective of this section is to evaluate the performance of HYPENS in the case of purely discrete nets. We want to study the error between the results of the simulation and those predicted analytically after a sufficiently large simulation time, the computer time required to do the simulation, and the simulation time necessary to reach a steady state.

We consider the family of queuing networks sketched in Fig. 2 whose structure depends on two parameters, k and n , where n denotes the number of branches exiting from the first queue and k denotes the length of each branch. We assume that all queues are M/M/1. The values of the average external arrival rate and of the average service rates are respectively equal to $\lambda_{in,s} = n$, and $\mu' = 2n$, $\mu'' = 1.5$; finally we assume $\beta = 0.9$. This is an open network of markovian queues whose steady state behaviour can be computed analytically: this will allow us to compare the simulation results with those predicted by the model.

The Petri net model of this system is sketched in Fig. 3. Each queue is modeled as a place whose marking represents the number of users in the queue, and an exponential timed transition whose parameter μ denotes

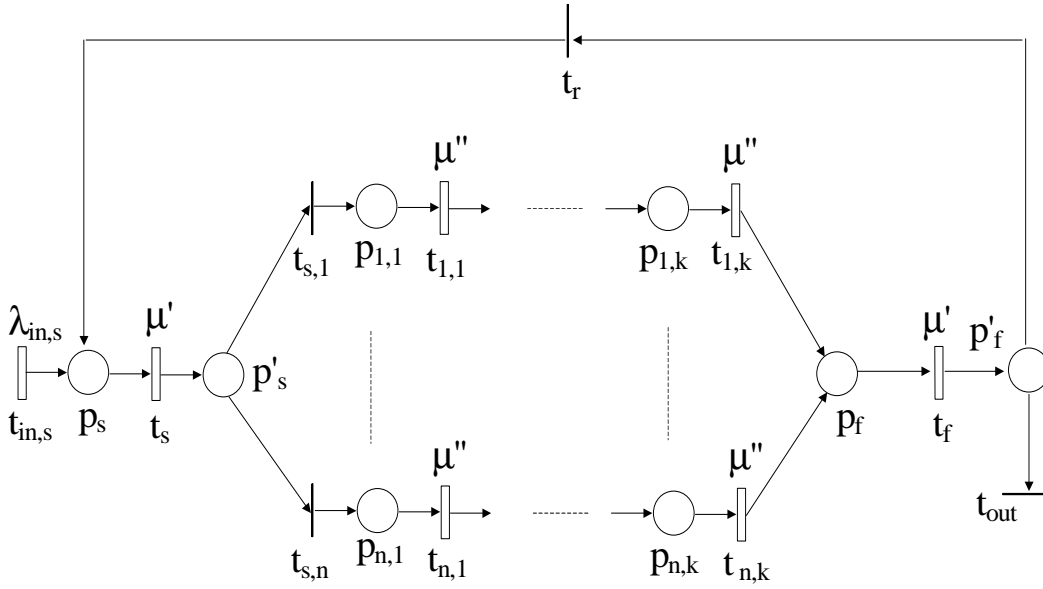


Fig. 3. The Petri net model of this queuing system in Fig. 2.

the average service rate. The external arrival has been modeled by an exponential timed transition whose parameter is equal to $\lambda_{in,s}$. The routing from the first queue to the n branches is modeled by immediate transitions $t_{s,1}, \dots, t_{s,n}$ with input place p'_s : conflicts among such transitions are solved assigning an equal weight to them [7]. This implies that, if a conflict among them arises, then all these transitions have the same probability to fire. Finally, immediate transitions t_{out} and t_r , sharing the same input place p'_f , model the routing from the last queue: a fraction equal to $\beta = 0.9$ exits from the system, the remaining part is recycled. Thus we assigned different weights, equal respectively to β and $1 - \beta$ to transitions t_{out} and t_r , that are used to solve conflicts.

Such a system is ergodic. In steady state the average firing rate of the transitions (equivalent to the arrival rates at the queues) and the average number of marking in the places (equivalent to the average number of customers in the queues) can be computed analytically and are shown in Table I.

Several numerical simulations have been carried out for different values of k and n . The most significant results are summarized in Table II. For all considered values of k and n , 10 different evolutions have been computed starting from the same initial condition corresponding to the empty system. In all cases we assumed a simulation time of 5000 units.

Simulations have been carried out on an Intel Core 2 with a clock of 2 GHz.

In particular, Table II shows the following results (average over 10 simulations).

- Columns 1 and 2 show the values of k and n .

$f^*(t_{in,s}) = f^*(t_{out})$	$f^*(t_s) = f^*(t_f)$	$f^*(t_{i,j})$	$f^*(t_r)$
n	$10n/9$	$10/9$	$n/9$

$M^*(p_s) = M^*(p_f)$	$M^*(p_{i,j})$	$M^*(p'_s) = M^*(p'_f)$
1.25	$20/7$	0

TABLE I

STEADY STATE AVERAGE FIRING RATES (TRANSITIONS) AND AVERAGE MARKINGS (PLACES) OF THE NET IN FIG. 3

k	n	$E[\tau_c]$	$Var[\tau_c]$	$E[\tau_a]$	$Var[\tau_a]$	$E[\varepsilon_P]$	$Var[\varepsilon_P]$	$E[\varepsilon_T]$	$Var[\varepsilon_T]$
1	1	14.1	0.08	2.1	0.008	0.080	0.004	0.015	0.0001
1	2	17.4	0.04	2.8	0.003	0.048	0.001	0.009	0.0000
1	3	20.7	0.16	3.6	0.013	0.058	0.002	0.015	0.0002
1	4	24.6	0.08	4.5	0.009	0.039	0.002	0.008	0.0000
2	1	30.8	0.27	8.4	0.98	0.060	0.002	0.009	0.0001
2	2	39.7	0.06	27.8	0.96	0.026	0.000	0.004	0.0000
2	3	49.8	0.12	50.6	3.0	0.038	0.001	0.005	0.0000
2	4	61.6	0.20	77.4	2.8	0.045	0.001	0.006	0.0000
3	1	50.1	0.38	74.7	6.0	0.044	0.001	0.008	0.0000
3	2	66.6	0.52	117.3	10.0	0.050	0.001	0.009	0.0000
3	3	88.0	0.96	170.5	14.9	0.038	0.001	0.007	0.0000
3	4	113.7	0.66	232.8	10.8	0.024	0.000	0.004	0.0000
4	1	72.2	0.24	172.3	10.0	0.039	0.001	0.005	0.0000
4	2	102.1	0.76	259.4	27.9	0.034	0.001	0.005	0.0000
4	3	140.2	1.42	364.1	33.6	0.023	0.001	0.005	0.0000
4	4	188.6	4.90	489.3	101	0.026	0.000	0.006	0.0000

TABLE II

THE RESULTS OF SOME NUMERICAL SIMULATIONS CARRIED OUT ON THE FOHPN IN FIG. 3.

- Columns 3 and 4 show respectively the average computation time $E[\tau_c]$ in seconds to run each simulation (namely to execute the file *simulator_HP.N.m*), and the corresponding variance $Var[\tau_c]$.
- Columns 5 and 6 show respectively the average time $E[\tau_a]$ in seconds we spent to perform each analysis (namely to execute the file *analysis_HP.N.m*), and the corresponding variance $Var[\tau_a]$.
- Columns 7 and 8 show a measure of the worst case place error, i.e., the worst case distance between the average steady state marking resulting from simulations and that one computed analytically (see Table I).

In particular, we first compute for each place $p \in P \setminus \{p'_s, p'_f\}$ the relative average marking error of p

$$\varepsilon_p = \frac{|M^*(p) - \bar{M}(p)|}{M^*(p)}.$$

where $\bar{M}(p)$ is the average marking of p during the simulation run, and $M^*(p)$ is the expected value shown in Table I.

Then we compute the maximum relative error on places, namely

$$\varepsilon_P = \max_{p \in P \setminus \{p'_s, p'_f\}} \varepsilon_p.$$

Finally, we compute the average error $E[\varepsilon_P]$ and the corresponding variance $Var[\varepsilon_P]$ over the 10 considered simulations.

- Columns 9 and 10 show a measure of the worst case transition error, i.e., the distance between the average firing rate of transitions resulting from simulations and that one computed analytically (see Table I).

We first compute for each transition $t \in T$ the relative average firing rate error of t

$$\varepsilon_t = \frac{|f^*(t) - \bar{f}(t)|}{f^*(t)}$$

where $\bar{f}(t)$ is the average firing rate of t during the simulation run, and $f^*(t)$ is the expected value shown in Table I.

Then, we compute the maximum relative error on transitions, namely

$$\varepsilon_T = \max_{t \in T} \varepsilon_t.$$

Finally, we compute the average error $E[\varepsilon_T]$ and the corresponding variance $Var[\varepsilon_T]$ over the 10 considered simulations.

Note that, as expected, the computation time τ_c and the time required to analyze the results τ_a grows with the size of the net. The error on the average place marking does not exceed the 8%, while error on

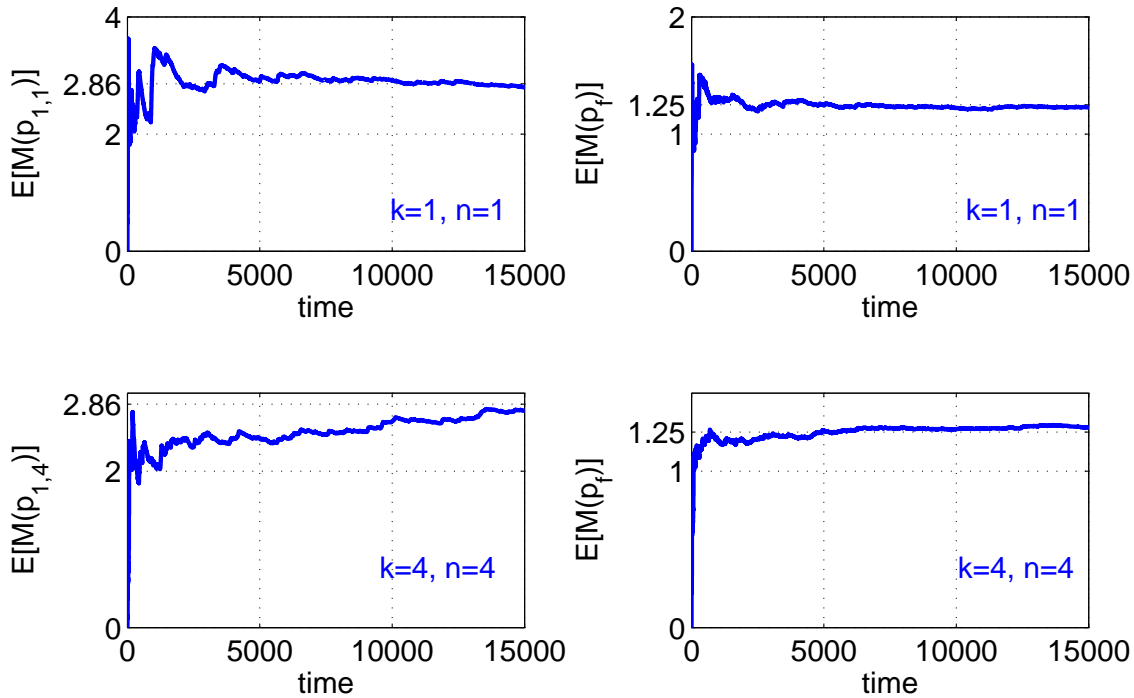


Fig. 4. The evolution of the average marking of places $p_{1,1}$, p_f ($k = n = 1$) and places $p_{1,4}$, p_f ($k = n = 4$).

the transition firing rate does not exceed the 1.5%. This shows that even for a relatively small simulation time, our tool provides reliable results.

Finally, Fig. 4 shows the evolution with respect to time of the average marking of some places, namely $p_{1,1}$, p_f in the case of $k = n = 1$, and $p_{1,4}$, p_f in the case of $k = n = 4$. As it can be observed, as time grows, all the above markings converge to the expected steady state value analytically computed.

V. A JOB SHOP SYSTEM

In this section we consider the FOHPN model of a production system consisting of a job-shop already presented in the literature [2], whose scheme is sketched in Fig. 5.

This job-shop system has 4 machines M_1 , M_2 , M_3 , M_4 , an assembly station M_5 , and seven buffers, B_0 , B_1 , B_2 , B_3 , B_4 , B_5 , B_6 . Machine M_i ($i = 1, 2, 3, 4, 5$) has a maximum machine production rate V_{M_i} . Buffers B_0 and B_6 are fictitious in the sense that B_0 is an infinite source containing all required raw materials and B_6 is a sink buffer with no constraints. On the contrary all other buffers have a finite capacity C_{B_i} .

There are two classes of products: class 1, whose flow is denoted in Fig. 5 by thin lines, and class 2,

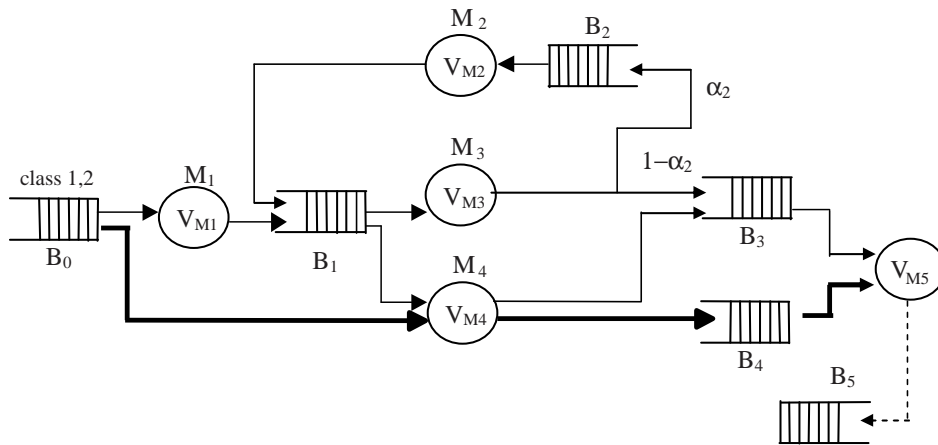


Fig. 5. A job-shop system.

whose flow is denoted by thick lines. The flow of assembled product is denoted in Fig. 5 by a dotted line. Row parts of both classes are initially contained in B_0 .

Initially, all buffers are empty (except the raw parts buffer B_0) and all machines are assumed to be operational.

To cope with the difficulty of large numbers of reachable markings, here we provide a "fluid" model of buffers and machines. Obviously, the state of machines, namely operational or down, is still a discrete state. Thus, the resulting model is hybrid. The FOHPN model of the production system under consideration is reported in Fig. 6. More detailed explanations on this can be found in [2].

For the sake of simplicity, we assume that only M_1 and M_2 are unreliable machines characterized by a time-dependent failure model. We also assume that all buffers have finite capacity except the two-class buffer B_0 and the final single-class buffer B_6 .

We consider the following numerical values: $V_{M1} = 10$, $V_{M2} = 15$, $V_{M3} = 30$, $V_{M4} = 10$, $V_{M5} = 10$; $\alpha_2 = 0.7$; $m_{B0}^1 = 10^3$, $m_{B0}^2 = 10^3$; $C_{B1} = 50$, $C_{B2} = 50$, $C_{B3} = 40$, $C_{B4} = 30$, $C_{B5} = 30$; $\lambda_{R,1} = 0.1$, $\lambda_{F,1} = 0.06$, $\lambda_{R,2} = 0.07$, $\lambda_{F,2} = 0.2$.

The optimal control policy we use in this example consists in the maximization of the system throughput (the IFS v_{M5}) while minimizing the number of input parts. More precisely, at each macro-period we solve an optimization problem of the form

$$\max_{\mathbf{v} \in \mathcal{S}} J = 1000 \cdot v_{M5} - (v_{M1} + v_{M4}^2) \quad (1)$$

where the different weighting coefficients have been chosen so as to consider the throughput maximization as the prior requirement, and the minimization of the flow of row parts taken from buffer B_0 as a secondary

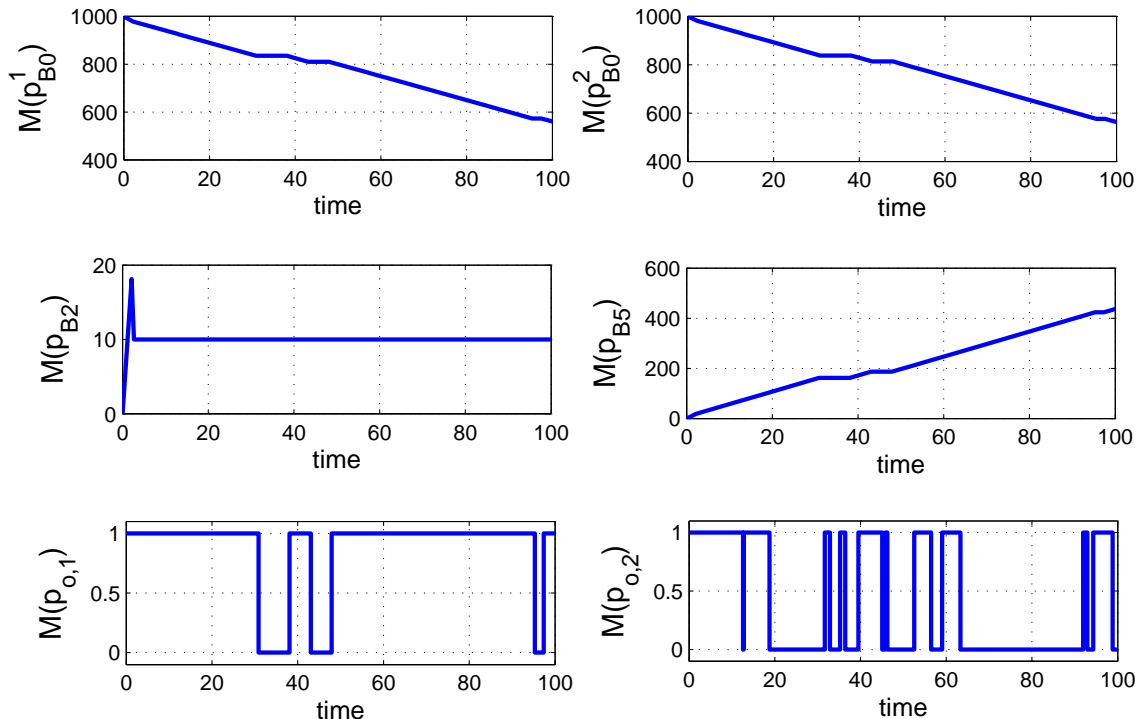


Fig. 7. The results of some numerical simulations carried out on the FOHPN in Fig. 6.

$$Td_{ave} = \begin{bmatrix} t_{R,1} & t_{F,1} & t_{R,2} & t_{F,2} \\ 0.03 & 0.03 & 0.06 & 0.06 \end{bmatrix}.$$

The evolution graph created by *simulator_HP*N, and matrices Md_{ave_t} , Md_{freq} are not reported here for sake of brevity.

The simulation time on an Intel Core 2 with a clock of 2 GHz is equal to 0.08 seconds. As expected, the simulation of a hybrid model is much faster than that of a purely discrete one, because there is a relatively small number of macro-events.

VI. CONCLUSION

In this paper we report the results of several tests performed with HYPENS, a tool for the analysis of hybrid Petri nets developed at the University of Cagliari. In a first case, we considered a family of queueing networks, described by a stochastic Petri net. In a second case, we considered a partially fluidized model of a job-shop system taken from the literature and described by a hybrid Petri net.

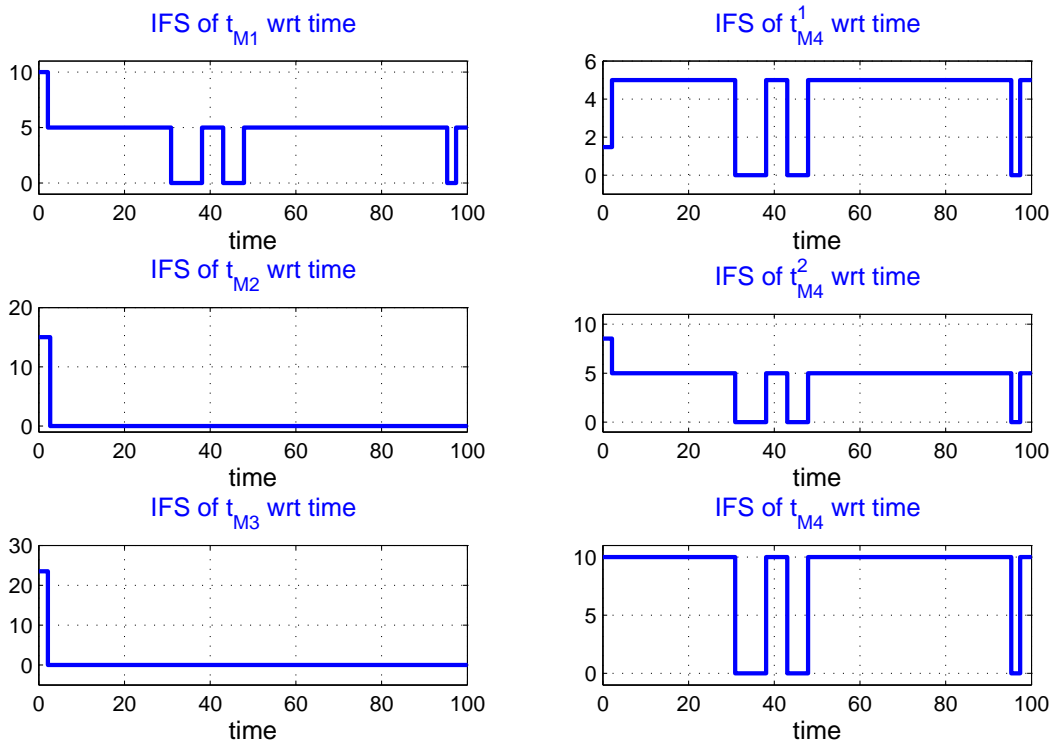


Fig. 8. The results of some numerical simulations carried out on the FOHPN in Fig. 6.

In the first case, the results obtained with HYPENS have been compared with the analytical results showing the correctness of the tool. The computation time and the time required for the analysis of the simulation results are significantly small to allow one to deal with non trivial examples.

In both cases, the tool allows one to build a model and test it with little effort, deriving a meaningful set of performance measures that can be graphically presented.

REFERENCES

- [1] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. Wiley, 1995.
- [2] F. Balduzzi, A. Giua, and C. Seatzu. Modelling and simulation of manufacturing systems with first-order hybrid Petri nets. *Int. J. of Production Research*, 39(2):255–282, 2001.
- [3] F. Balduzzi, G. Menga, and A. Giua. First-order hybrid Petri nets: a model for optimization and control. *IEEE Trans. on Robotics and Automation*, 16(4):382–399, 2000.
- [4] R. David and H. Alla. *Discrete, Continuous and Hybrid Petri Nets*. Springer, 2004.
- [5] A. Di Febbraro, A. Giua, G. Menga, and (Eds.). Special issue on Hybrid Petri nets. *Discrete Event Dynamic Systems*, 2001.
- [6] <http://sourceforge.net/projects/hisim>.
- [7] <http://www.diee.unica.it/~giua/TESI/Fausto.Sessegio/HYPENS>.

[8] <http://www.lag.ensieg.inpg.fr/sirphyco>.

18

[9] F. Sessego, A. Giua, and C. Seatzu. HYPENS: a matlab tool for timed hybrid Petri nets. In *Application and Theory of Petri nets, Kees M. van Hee, R. Valk (Eds.), Lecture notes in Computer Science, Springer Berlin/Heidelberg*, June 2008.

[10] M. Silva and L. Recalde. On fluidification of Petri net models: from discrete to hybrid and continuous models. *Annual Reviews in Control*, 28(2):253–266, 2004.