SimHPN: a MATLAB toolbox for continuous Petri nets *

Jorge Júlvez and Cristian Mahulea
Aragón Institute of Engineering Research (ISA), University of Zaragoza, Spain (e-mails: julvez, cmahulea@unizar.es)

Abstract: This paper presents a MATLAB embedded package for continuous Petri nets called SimHPN. It offers a collection of tools devoted to simulation, analysis and synthesis of dynamical systems modeled by continuous Petri nets. Its embedding in the MATLAB environment provides the considerable advantage of creating powerful algebraic, statistical and graphical instruments exploiting the high quality routines available in MATLAB.

1. INTRODUCTION

Petri nets (PN) are a mathematical formalism for the description of discrete-event systems, successfully used for modeling, analysis and synthesis of such systems. One of its main features is that their state space belongs to the set of non-negative integers (Murata (1989)). Another key feature of PN is their capacity to represent graphically and visualize primitives such as parallelism, synchronization, mutual exclusion, etc.

As any other formalism for discrete event systems, PN suffer from the state explosion problem especially when the system is heavily populated. Among the different procedures to overcome this problem, fluidification is a promising one. In the case of PN this leads to continuous Petri nets (ContPN) (David and Alla (2010)). In ContPN the marking of places can be any non-negative real value. As a consequence of this, a transition can fire in any real amount between zero and its enabling degree. Different time interpretations can be considered, being infinite and finite server semantics the most popular. A third firing semantics, called product semantics, can be used to study nets obtained by decolorization and population dynamics.

In this paper we present a new MATLAB embedded software capable to simulate and analyze ContPN systems with different firing semantics. Up to our knowledge this is the first package dealing with ContPN that includes facilities for the three most used firing semantics in literature. In MATLAB there exists a toolbox dealing with discrete Petri nets (Matcovschi et al. (2003)) and one for the so-called first order hybrid Petri nets (Sessego et al. (2008)), but until now no one deals with ContPNs.

The main features of the new created toolbox are:

- simulation of continuous Petri nets under the following firing semantics: infinite server, finite server, product semantics.
- import models from different graphical Petri net editors
- different visualization options
- computation of throughput bounds
- computation of P-T semiflows
- optimal sensor placement

The paper is organized as follows: Section 2 introduces the formal definition of ContPN. Section 3 defines the three different firing semantics for the timed interpretation. Section 4 briefly presents some of the algorithms implemented in the package. In Section 5 the user interface of the simulator as well as several examples are depicted. Finally, Section 6 sketches the main features of the package.

2. UNTIMED CONTINUOUS PETRI NETS

Definition 1. A contPN system is a pair \( \langle N, m_0 \rangle \), where: \( N = \langle P, T, \text{Pre}, \text{Post} \rangle \) is a net structure (with set of places \( P \), set of transitions \( T \), pre and post incidence matrices \( \text{Pre}, \text{Post} \in \mathbb{R}^{\left|P\right| \times \left|T\right|} \)) and \( m_0 \in \mathbb{R}_{\geq 0}^\left|P\right| \) is the initial marking.

The token load of the place \( p_i \) at marking \( m \) is denoted by \( m_i \) and the preset and postset of a node \( X \in P \cup T \) are denoted by \( \text{Pre}(X) \) and \( \text{Post}(X) \), respectively. For a given incidence matrix, e.g., \( \text{Pre}(p_i, t_j) \) denotes the element of \( \text{Pre} \) in row \( i \) and column \( j \).

A transition \( t_j \in T \) is enabled at \( m \) iff \( \forall p_i \in \text{Pre}(t_j), m_i > 0 \). The enabling degree of \( t_j \) is:

\[
\text{enab}(t_j, m) = \min_{p_i \in \text{Pre}(t_j)} \left\{ \frac{m_i}{\text{Pre}(p_i, t_j)} \right\}
\]

An enabled transition \( t_j \) can fire in any real amount \( 0 \leq \alpha \leq \text{enab}(t_j, m) \) leading to a new marking \( m' = m + \alpha \cdot C(t_j) \), where \( C = \text{Post} - \text{Pre} \) is the token-flow matrix and \( C(t_j) \) is its \( j \) column. If \( m \) is reachable from \( m_0 \) through a finite sequence \( \sigma \), the state (or fundamental) equation, \( m = m_0 + C \cdot \sigma \), is satisfied, where \( \sigma \in \mathbb{R}_{\geq 0}^{\left|T\right|} \) is the firing count vector.

Right and left non negative annihilers of the token flow matrix \( C \) are called T- and P-semiflows, respectively. If there exists \( y > 0 \) such that \( y \cdot C = 0 \), the net is said to be conservative, and if there exists \( x > 0 \) satisfying \( C \cdot x = 0 \), the net is said to be consistent.

3. TIMED CONTINUOUS PETRI NETS

Under any timed interpretation of the net model, the fundamental equation depends on time: \( m(t) = m_0 + C \cdot \sigma(t) \). Through time differentiation, the following equation is obtained: \( \dot{m}(t) = C \cdot \dot{\sigma}(t) \). The derivative of the firing
count vector will be called the (firing) flow of the timed model: \( f(\tau) = \dot{\sigma}(\tau) \). Different definitions for the flow of continuous timed transitions have been proposed in the literature, being the three most important finite server (or constant speed), infinite server (or variable speed) and product semantics (Alla and David (1998); Silva and Recalde (2002)).

### 3.1 Infinite Server Semantics

In discrete Markovian Petri nets systems, infinite server semantics means that the delay between the enabling and the firing of a transition is the minimum of \( n \cdot t \) independent exponentially distributed variables, all of them with the same parameter, \( \lambda \). This is an exponential distribution too, with parameter \( n \cdot \lambda \) (by flow additivity for exponential distribution or, more generally, by order statistics (Feller (1950))). If we make a first-order approximation, i.e., we just take into account the mean values, the firing of the transition takes \( 1/(n \cdot \lambda) \) time units. This can be interpreted from a continuous point of view as \( f_j = \lambda \cdot \text{enab}(t_j) \), which corresponds to the variable speed of (Alla and David (1998)).

Under infinite server (variable speed) the flow of a transition \( t_j \) is:

\[
 f_j = \lambda_j \cdot \text{enab}(t_j, m) = \lambda_j \cdot \min_{p_i \in t_j} \left\{ \frac{m_i}{\text{Pre}(p_i, t_j)} \right\}
\]  

(1)

where \( \lambda_j \) is a real positive number associated to \( t_j \).

The enabling degree of the transition \( t_j \) represents the number of active servers for that transition at \( m \). The flow will be the number of active servers times the work each one does per time unit (\( \lambda_j \)). Notice that the number of active servers in a transition (station) depends only on the marking of its input places.

### 3.2 Finite Server Semantics

In discrete systems, finite server semantics can be implemented by means of infinite server semantics by adding a self-loop place marked with as many tokens as the number of servers. However, this does not represent finite server semantics if these tokens are interpreted as a fluid. To have a correct approximation, the “server tokens” should be considered different from the tokens in the rest of the marking, because they cannot be split. From a discrete point of view, if the marking is multiplied by a natural number (greater than one), the tokens representing the servers should not be multiplied, i.e., we may have more customers, but keep the number of resources.

Let us consider single-server semantics first. In discrete Markovian Petri nets systems, when a transition is enabled, it fires with a rate exponentially distributed with parameter \( \lambda \). Let \( t_i \) be the delay of the \( i \)-th firing. Then, the average delay after \( k \) firings, \( \frac{1}{k} \cdot \sum_{i=1}^{k} x_i \), is a k-Erlang distribution with mean \( 1/\lambda \) (thus variance \( 1/(\lambda^2 k) \)). That is, if we integrate the flow along a large period of time, we obtain approximately the same as if the flow had been constant, because the variance tends to vanish when \( k \) is large. So, if the transition is always firing, i.e., it is always enabled, the firing can be approximated by a constant flow with speed \( \lambda \). If it is not, instead of having idle periods, we may approximate the server by a slower one that is always busy, i.e., one that works at the same speed as the incoming flow of tokens. In any case, if we let the system evolve for a large enough period, the error will be small.

This corresponds to the constant speed of (Alla and David (1998)), with \( V_j = \lambda_j \), that is:

\[
 f_j = \begin{cases} 
 \lambda_j, & \text{if } \forall p_i \in t_j, m_i > 0 \\
 \min_{p_i \in t_j} \min_{m_i = 0} \left\{ \sum_{t_i \in p_i} \frac{f_{t_i} \cdot \text{Post}(t_i, p_i)}{\text{Pre}(p_i, t_j)} \right\}, & \text{otherwise} 
\end{cases}
\]  

(2)

If for all \( p_i \in t_j, m_i > 0 \) then \( t_j \) is said to be strongly enabled, it is weakly enabled otherwise. Let us now consider the k-server semantics. In this case, the firing rate of the transition is exponentially distributed with parameter \( \lambda_j \cdot \min \{ k, \text{enab}(t_j) \} \). If the number of servers, \( k \), is small with respect to the (fluid) marking of the system, it is reasonable to consider that this minimum will (often) be \( k \). Then, the same reasoning as in the previous case can be applied, obtaining a maximal flow of \( k \cdot \lambda_j \).

Observe that (2) is not defining completely the flow of a ContPN under finite server semantics. In the case of conflict, a resolution policy should be specified, otherwise many solutions of the flows are possible (Balduzzi et al. (2000)). Therefore, this semantics is non-deterministic as defined in (2).

### 3.3 Product semantics

Continuous Petri nets can be used to model the evolution of a system of populations, for example the Lotka-Volterra predator/prey model (Cellier (1991)). Such systems are naturally modeled by coloured nets, and when they are decoloured the flow of a transition becomes the product of the enabling degrees of every input place (Silva and Recalde (2000)):

\[
 f_j = \lambda_j \cdot \prod_{p_i \in t_j} \left\{ \frac{m_i}{\text{Pre}(p_i, t_j)} \right\}
\]

4. METHODS FOR THE ANALYSIS OF CONTPN

#### 4.1 Performance bounds

After a transient state, a continuous Petri net system under infinite and finite server semantics reaches a steady state when its marking, and so the flow (or throughput) through transitions, remains constant. Observe that if a steady state is reached, \( \dot{m} = 0 \), and so \( C \cdot f = 0 \). That is, the flow vector in the steady state is a T-semiflow.

**Infinite Server Semantics.** The following programming problem can be used to compute an upper bound for the throughput of a transition (Jülvez et al. (2005)):

\[
 \max \{ \phi_j \mid \mu_j^{\infty} = m_0 + C \cdot \sigma_{ss}, \quad \phi_j^{ss} = \lambda_j \cdot \min_{p_i \in t_j} \left\{ \frac{\mu_j^{\infty}}{\text{Pre}(p_i, t_j)} \right\} \forall t_j \in T, \quad C \cdot \phi_{ss} = 0, \quad \mu_{ss}, \sigma_{ss} \geq 0 \}
\]  

(3)

Nevertheless, this non-linear programming problem is difficult to solve due to the minimum operator. When a transition \( t_j \) has a single input place, the equation reduces to (4). And when \( t_j \) has more than an input place, it can be relaxed (linearized) as (5).
Finite Server Semantics. With finite server semantics, the flow of a transition \( t \) must be always less than or equal to \( \lambda_j \). Moreover, for consistent nets with only one T-semiflow \( x \), in steady state the flow vector is necessarily proportional to such T-semiflow. Hence, we just need to observe the transitions and find the bottleneck, that is, a certain \( t_j \) whose utilization is equal to 1. Therefore, an upper performance bound, \( \chi_j \), can be computed as:

\[
\chi_j = \max \left\{ k \mid k \cdot x^{(j)} \leq \lambda \right\}
\]

where \( x^{(j)} \) is the T-semiflow normalized, i.e., \( x^{(j)}(t_j) = 1 \), for transition \( t_j \).

4.2 Computation of minimal P-T semiflows

Following the steps proposed in (Silva (1985)) an algorithm to compute the minimal P and T-semiflows is proposed. The input parameter of the procedure is the incidence matrix \( C \), and the output is a matrix containing the vectors that represent the minimal semiflows.

4.3 Optimal Sensor Placement

Assuming that each place can be measured at a different cost, the optimal sensor placement problem of continuous Petri Nets under infinite server semantics is to decide the set of places to be measured such that the net system is observable at minimum cost. Measuring a place allows the observation of a set of others ("covered" by that measure) but, the problem is not a simple covering one (Garey and Johnson (1979)). The question is studied at the structural level in (Mahulea et al. (2005)) and the results obtained are used in the implementation of an algorithm to reduce the computational burden.

4.4 Optimal Steady-State

The only action that can be performed on a ContPN is to slow down the flow of its transitions. If a transition can be controlled (its flow reduced or even stopped), we will say that it is a controllable transition. The forced flow of a controllable transition \( t_j \) becomes \( f_j - u_j \), where \( f_j \) is the flow of the unforced system, i.e., without control, and \( u_j \) is the control action \( 0 \leq u_j \leq f_j \).

In production control is frequent the case that the profit function depends on production (benefits in selling), working process and amortization of investments. Under linear hypothesis for fixed machines, i.e., \( \lambda \) defined, the profit function may have the following form:

\[
w^T \cdot f - z^T \cdot m - q^T \cdot m_0
\]

where \( f \) is the throughput vector, \( m \) the average marking, \( w \) a gain vector w.r.t. flows, \( z^T \) is the cost vector due to immobilization to maintain the production flow and \( q^T \) represents depreciations or amortization of the initial investments.

The algorithm used to compute the optimal steady state flow (and marking) is very much alike the one used to compute the performance bounds, with the difference that the linear programming problem that needs to be solved is:

\[
\begin{align*}
\max \{ w^T \cdot f - z^T \cdot m - q^T \cdot m_0 \mid & C \cdot f = 0, \\
m = m_0 + C \cdot \sigma, \\
f_j = \lambda_j \cdot \frac{m_j}{Pre(p_i,t_j)} - v(p_i,t_j), \\
& \forall p_i \in \bullet t_j, \forall v(p_i,t_j) \geq 0 \\
f, m, \sigma & \geq 0
\end{align*}
\]

where \( v(p_i,t_j) \) are slack variables. These slack variables give the control action for each transition. For more details on this topic, see (Mahulea et al. (2008)).

5. THE SIMHPN PACKAGE

The SimHPN (http://webdiis.unizar.es/GISED/?q=tool/simhpn) provides a Graphical User Interface (see Fig. 1) to achieve the simulations and computations described in the previous sections. The data of the net system can be introduced either manually or through Petri nets editors: P Mohammed or TimeNet (Zimmermann and Knoke (1995)). The data needed for a system to be simulated is: \texttt{Pre} and \texttt{Post} matrices, initial marking \( m_0 \) and the internal speeds of transitions \( \lambda \).

5.1 Graphical interface

![Fig. 1. Sketch of the main window of SimHPN](Image 507x141 to 553x322)

The SimHPN (http://webdiis.unizar.es/GISED/?q=tool/simhpn) provides a Graphical User Interface (see Fig. 1) to achieve the simulations and computations described in the previous sections. The data of the net system can be introduced either manually or through Petri nets editors: P Mohammed or TimeNet (Zimmermann and Knoke (1995)). The data needed for a system to be simulated is: \texttt{Pre} and \texttt{Post} matrices, initial marking \( m_0 \) and the internal speeds of transitions \( \lambda \).

![Fig. 1. Sketch of the main window of SimHPN](Image 507x141 to 553x322)

It is also possible to adjust the maximum absolute and relative errors of the simulation as well as the simulation length. As plot options, the simulator allows one to plot the evolution of the marking of the places, the evolution of the
flow of the transitions and the evolution of the marking of
one place vs. the marking of other place. When more than
one plot, i.e., marking and throughput evolution, is desired
by the user, new windows are opened by the simulator to
show them.

By means of two different buttons, the P(T)-semiflows
of the net as well as the throughput bounds of the system,
both for infinite and finite servers semantics, can be
computed. The results are displayed on the MATLAB
command window and can be used for future analysis
tasks.

5.2 Case study 1

Let us consider the ContPN system in Fig. 2 modeling a
resource (place p_6) shared between two processes, and let
us simulate its evolution with two continuous approxima-
tions: finite and infinite server semantics. The following to-
ken conservation laws hold: m_1 + m_3 + m_4 = 1, m_2 + m_5 = 1
and m_4 + m_5 + m_6 = 1. Thus the markings of p_2, p_3, p_6
are sufficient to represent the evolution of all states (mark-
ings). Let \( \lambda = [1, 2, 1, 1, 0.5]^T \) and \( \mathbf{m}_0 = [1, 1, 0, 0, 0, 1]^T \).
Observe that in this case the behavior of the discrete PN
is the same for finite and infinite server semantics because the
(singles) servers are implicit in the model, in other words
the upper bounds of the marking of all places is 1. On the
contrary, continuous finite and infinite server semantics do
not lead to the same values.

**Infinite server semantics.** First, observe that p_2 is implicit,
i.e., it is never the only place constraining the firing
of t_4 (DiCosare et al. (1993)), and its marking verifies:
m_2 = m_4 + m_6. Hence, \( f_4 = \min\{m_2, m_6\} = m_6 \),
and so only two linear systems can govern the evolution. At
\( \tau = 0 \), m_3 < m_6, therefore the evolution is governed by
the following linear system:

\[
\Sigma_1 = \begin{cases}
\dot{m}_2 = \lambda_5 \cdot m_5 - \lambda_4 \cdot m_6 = \frac{1}{2} m_5 - m_6 \\
\dot{m}_3 = \lambda_1 \cdot m_1 - \lambda_2 \cdot m_3 = m_1 - 2m_3 \\
\dot{m}_6 = \lambda_3 \cdot m_6 + \lambda_5 \cdot m_5 - \lambda_2 \cdot m_3 - \lambda_4 \cdot m_6 \\
= m_4 + \frac{1}{2} m_5 - 2m_3 - m_6
\end{cases}
\]

The evolution of the ContPN system is sketched in Fig. 3.
It evolves according to \( \Sigma_1 \) until \( \tau \approx 1.14 \) t.u. when
m_3(\tau) = m_6(\tau). At that point, a switch occurs and the
new linear system is:

\[
\Sigma_2 = \begin{cases}
\dot{m}_2 = \lambda_5 \cdot m_5 - \lambda_4 \cdot m_6 = \frac{1}{2} m_5 - m_6 \\
\dot{m}_3 = \lambda_1 \cdot m_1 - \lambda_2 \cdot m_6 = m_1 - 2m_6 \\
\dot{m}_6 = \lambda_3 \cdot m_6 + \lambda_5 \cdot m_5 - \lambda_2 \cdot m_6 - \lambda_4 \cdot m_6 \\
= m_4 + \frac{1}{2} m_5 - 3m_6
\end{cases}
\]

The system evolves according to \( \Sigma_2 \) and reaches the
steady state marking \([0.4, 0.6, 0.2, 0.4, 0.4, 0.2]^T \) with the
corresponding flow: \([0.4, 0.4, 0.4, 0.2, 0.2, 0.2]^T \).

**Finite server semantics.** The evolution of the system under
finite server semantics is presented in Fig. 4. At m_0, the
input places of t_1 and t_4 are marked, therefore t_1 and
t_4 are strongly enabled and \( f_3 = f_4 = 1 \). The other
transitions are weakly enabled and their flows depend
on the input flows to the empty input places. For t_2,
the input flow to p_3 (the only empty input place) is 1,
then \( f_2 = \min\{\lambda_2, 1\} = 1 \). Transition t_3 will work at the
maximum speed because the input flow in p_4 is \( f_3 = 1 \).
For t_5, the input flow to p_5 is 1, then its flow is limited by
its maximal firing speed \( \lambda_5 = 0.5 \).
The net in Figure 5 has been loaded from the PMEditeur steady-state markings are obtained.

According with these flow equations, the evolution of the system will be governed by the linear system, $\Sigma_3$, until $\tau = 2$, when $m_6$ and $m_2$ become empty. At this time instant, the marking is $[1, 0, 0, 0, 1, 0]^T$. Now, $t_1$ and $t_5$ are strongly enabled, therefore $f_1 = 1$ and $f_5 = 0.5$. The weakly enabled transitions $t_2$ and $t_4$ are in conflict and a resolution policy must be specified. Assume, for example, that both transitions have the same priority, i.e., the incoming flow to $p_6$ is equally split to $t_2$ and $t_4$. Moreover, the output flow of $p_4$ is upper bounded by the flow of $t_3$. Then, the resulting flow is $f_2 = f_3 = f_4 = 0.5$. So, the system of equations that defines the evolution after $\tau = 2$ is the new linear system $\Sigma_4$.

$$
\Sigma_4 = \begin{cases}
\dot{m}_1 = f_3 - f_1 = 1 - 1 = 0 \\
\dot{m}_3 = f_1 - f_2 = 1 - 1 = 0 \\
\dot{m}_6 = f_3 + f_5 - f_2 - f_4 = 1 + 0.5 - 1 - 1 = -0.5
\end{cases}
$$

At $\tau = 4$ the marking is $[0, 0, 1, 0, 1, 0]^T$. Thus $p_4$ is empty and a new flow computation has to be done. The only strongly enabled transition is $t_5$, hence $f_5 = 1$. After solving the associated equations, $f_1 = f_2 = f_3 = f_4 = 0.5$ is obtained. These values correspond to a steady state marking ($\mathbf{m}(\tau) = 0$).

Clearly, the evolution of a ContPN system is quite different under both semantics: different transitory regimes and steady-state markings are obtained.

### 5.3 Case study 2

The net in Figure 5 has been loaded from the PMEditeur and model a table factory system ((adapted from Teruel et al. (1997))). The system is composed of the following items: two different machines ($t_1$ and $t_2$) to make table-legs, a new fast one ($t_1$) which produces two legs at a time, and the old one ($t_2$), which makes legs one by one; a machine ($t_3$) to produce table boards; a machine ($t_4$) to assemble a four legs and a board; and a big painting line ($t_6$) which paints two tables at once. The painting line has more capacity than the other machines, so more unpainted tables are brought ($t_1$) from a different factory. The different products are stored in buffers: Table-legs are stored in $p_1$, boards are stored in $p_5$, and $p_7$ is devoted to the storage of unpainted tables. The rest of the places contains work orders: whenever the painting line finishes a couple of tables, it delivers work orders to the leg-makers, the board-maker, and the other factory. Due to some commercial considerations, it is desired 50% of the tables to be assembled in the factory and 50% to be brought from the other factory (this order is represented by equal weights of the arcs going from $t_4$ to $p_3$ and $p_4$). It is also required that 75% of the legs are produced by the new machine and 25% by the old one (this is modelled by the arc weights going from $t_6$ to $p_1$ and $p_2$).

The SimHPN package computes the P-semiflows: $y_1 = [1, 0, 0, 4, 1, 0, 4]^T$ and $y_2 = [0, 0, 1, 0, 1, 0, 1]^T$, and the unique T-semiflow as: $x_1 = [3, 2, 2, 2, 2, 2]^T$. Let us assume that $\lambda = 1$. The simulator computes the following steady state throughput bounds for the six transitions of the system: upper and lower bounds under infinite server semantics: $[0.6, 0.4, 0.4, 0.4, 0.4, 0.4]^T$

---

**Fig. 5.** A system that models a manufacturing application

**Fig. 6.** Predator/prey model

while under finite server semantics the upper bound is: $[1, 0.667, 0.667, 0.667, 0.667, 0.667]^T$.

Since this net system is monotonic w.r.t. the initial marking and the firing rate vector $\lambda$, the optimal steady-state corresponds to $u = 0$ for the gain vector $w = 1$, i.e., $f^{ss} = [0.6, 0.4, 0.4, 0.4, 0.4, 0.4]^T$. This is confirmed by the corresponding procedure implemented in SimHPN. The steady state marking is obtained as: $[1.2, 0.4, 0.4, 0.4, 2.6, 0.4, 0.8]^T$. For the same net, assuming a measuring cost of each place equal to 1, the optimal sensor placement obtained from the SimHPN is $\{p_2, p_5, p_6\}$, i.e., the set of measuring place. Obviously, the optimal measuring cost is 3.

### 5.4 Case study 3

In Figure 6, a Petri net model of a predator/prey system is depicted. The number of preys is represented by the marking of $p_1$ and the number of predators by the marking of $p_2$. In this ecological model, the enabling of $t_2$ represents encounters between predators and preys and therefore must be proportional to the product of both markings. Let the vector $\lambda$ be $\lambda = [0.25, 0.01, 1]^T$ and the initial markings be $m_0 = [90, 40]^T$.

The product semantics produces an oscillatory behavior of the markings and the throughputs. Figure 7 shows the evolution of the populations of predators and preys while figure 8 shows how the throughput of the three transitions evolve through time. If the marking of one place (population of predators) is plotted vs. the marking of the other (population of preys), the result is the closed orbit loop shown in Figure 9.
6. CONCLUSIONS

This paper has presented a new MATLAB package, called SimHPN, that allows us to perform several analysis and synthesis tasks on continuous Petri nets working under different server semantics. In particular, SimHPN provides procedures to compute minimal P and T-semiflows, throughput bounds, optimal steady state and optimal sensor placement.

Additionally, SimHPN is able of simulating continuous Petri nets evolving under any of the following semantics: infinite server, finite server and product semantics. The package is equipped with a Graphical User Interface that offers a friendly interaction with the user.

REFERENCES


Silva, M. (1985). Las Redes de Petri: en la Automática y la Informática. AC.


