MATLAB tools for the analysis of Petri net models*

Jorge Júlvez
Dept. of Computer Science and Systems Engineering
Univ. of Zaragoza, Spain
{julvez}@unizar.es

Mihaela H. Matcovschi, Octavian Pastravanu
Dept. of Automatic Control and Applied Informatics, Technical University
"Gh. Asachi" of Iasi, Romania
{mhanako,opastrav}@ac.tuiasi.ro

Abstract

The paradigm of Petri nets offers a rich modeling power that has been successfully exploited in many different application domains as manufacturing, logistic, traffic, computer and biological domains. In order to develop and analyze Petri models in an efficient and systematic way, appropriate software tools are required. Among the existing software platforms for scientific software development, MATLAB places at the user's disposal a large set of toolboxes with an inutitive graphical interface. This paper describes some of the existing Petri net toolboxes that have been developed on MATLAB, and that cover the needs for the modeling and analysis of discrete, continuous and hybrid systems.

1 Introduction

The theory of Petri nets (abbreviated PNs) emerged from the computer-science-oriented research of the early sixties [20] and two decades later became an effective instrument for the exploration of a large class of systems, generically referred to as discrete-event and hybrid systems. The original idea of a bipartite graph able to model properties such as process synchronization, asynchronous events, concurrent operations, and conflicts or resource sharing, proved to be seminal for further developments, in the sense of the theoretical extension, concomitantly with the accommodation of new types of practical models. Thus, the conceptual evolution allowed the incorporation of time information into transitions [17, 21], places [24] or arcs [31], broadening the usage of PNs from qualitative analysis of discrete-event systems to quantitative analysis and performance modeling. Different types of time constraints were allowed: deterministic values [21], time intervals [17], stochastic [1] or possibilistic forms [19]. Dealing with analysis or synthesis of complex discrete-event systems based on PN models often leads to computational infeasibility due to the so called state explosion when the set of reachable markings increases exponentially with respect to the initial marking. To overcome this drawback in 1987 two different relaxation techniques were independently introduced in order to study discrete event systems through a continuous approximated model [7] and [26]. If only a few transitions in a PN model are fluidized, i.e., some transitions remain discrete while some become continuous, the obtained models form a class of hybrid Petri nets [3]. An overview of the development of the field of Petri nets in the half a century from the first introduction in Carl Adam Petris Ph.D. thesis is presented in the recent works [25, 22].

Besides computer science, manufacturing engineering was one of the most important beneficiaries of the scientific potential offered by PNs at the beginning of the nineties. This is the period when robust instruments supported by the PN theory were devised for flexible manufacturing systems analysis and de-

*Submitted to the *19th IEEE International Conference on Emerging Technologies and Factory Automation; ETFA'2014* as invited paper to track 4: *"Automated Manufacturing Systems"*.

2014 IEEE Emerging Technology and Factory Automation (ETFA)
978-1-4799-4845-1/14/$31.00

sign, which yielded the publication of the first monographs [9, 32]. The results of the mentioned period stimulated a considerable increase of the effort invested during the next two decades, as reflected by the state of the art presented in the book [6].

The modeling power of the PNs combined with the convenient handling of their graphical representations motivated many academic or research groups to create a wide variety of software tools, as summarized by section Tools and Software of Petri Net World (`http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/db.html`). On the other hand, most of these tools were not meant as general-purpose packages, dealing with particular types of PNs and focusing on sets of specialized problems in accordance with the key interests of the developers. Therefore, the selection of such a package may become rather difficult for any user who cannot find a comfortable match between the available software facilities and his/her investigation objectives.

Users preferences in choosing PN tools also depend on their familiarity with the software environments hosting those tools, because many applications require supplementary studies preceding or following the PN-oriented tasks. Subsequently, for students and specialists working in different areas of engineering (particularly skilled in the exploitation of MATLAB and its specialized toolboxes), bridging the PN-formalism with the generous computational resources of MATLAB appeared as an extremely profitable approach to discrete-event and hybrid systems.

Within the above commented context, our work aims to reinforce the dissemination of the capabilities offered for PN applicative research by two software products endorsed by the MathWoks Connections Program, namely *Petri Net Toolbox* developed at "Gheorghe Asachi" Technical University of Iasi (`http://www.ac.tuiasi.ro/pntool/`) and *SimHPN* developed at Zaragoza University (`http://webdiis.unizar.es/GISED/?q=tool/simhpn`). It is worth saying that in 2001, besides the release of *Petri Net Toolbox* version 1 presented in [13], some PN analysis instruments for MATLAB were also described in paper [29], but their use was drastically limited by the lack of a graphical user interface.

To ensure a comprehensive picture of the MATLAB-embedded (MATLAB-compatible) PN tools, our work refers additionally to another two packages, namely HYPENS developed at the University of Cagliari (`http://www.diee.unica.it/automatica/hypens/`) and PNSB - developed at "Gheorghe Asachi" Technical University of Iasi (`http://www.ac.tuiasi.ro/~pnsb/`); these packages allow the simulation of hybrid systems described via PN models.

Our exposition is structured as follows. The theoretical background needed for the presentation of the two software products is ensured by Section 2. The main features of Petri Net Toolbox and SimHPN are discussed in Sections 3 and 4, respectively. Section 5 gives brief overviews of HYPENS and PNSB. In Section 6, a relevant example illustrates the software use in addressing problems typical to systems engineering.

2 Discrete, Continuous and Hybrid Petri nets

In the classical formalism of discrete Petri nets, transitions are fired in natural amounts, that leads to discrete event dynamic systems. In contrast to this, the firing amounts of transitions in continuous Petri nets is allowed to take real numbers, that leads to a continuous state space. On the other hand hybrid Petri nets [8, 5] integrate both discrete and continuous transitions in a single net model. In fact, both discrete and continuous Petri nets can be seen as particular cases of hybrid Petri nets in which all transitions are considered as discrete and continuous respectively.

This section defines the main concepts related to the hybrid nets that are supported by the MATLAB toolboxes under consideration. In the following, the reader is assumed to be familiar with Petri nets (see [18, 10] for a gentle introduction).

2.1 Untimed Hybrid Petri nets

Definition 2.1 *A* Hybrid Petri Net (HPN) system *is a pair* $\langle \mathcal{N}, \boldsymbol{m_0} \rangle$*, where:* $\mathcal{N} = \langle P, T, \boldsymbol{Pre}, \boldsymbol{Post} \rangle$ *is a* net structure*, with set of places* P*, set of transitions* T*, pre and post incidence matrices* $\boldsymbol{Pre}, \boldsymbol{Post} \in \mathbb{R}_{\geq 0}^{|P| \times |T|}$*, and* $\boldsymbol{m}_0 \in \mathbb{R}_{\geq 0}^{|P|}$ *is the* initial marking.

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

In a HPN, the set of transitions T is partitioned in two sets $T = T^c \cup T^d$, where T^c contains the set of continuous transitions and T^d the set of discrete transitions. In contrast to other works, the set of places P is not explicitly partitioned, i.e., the marking of a place is a natural or real number depending on the firings of its input and output transitions. Nevertheless, in order to make net models easier to understand, those places whose marking can be a real non-integer number will be depicted as double circles, and the rest of places will be depicted as simple circles (such places will have integer markings). Continuous transitions are graphically depicted as two bars, while discrete transitions are represented as empty bars.

Right and left non negative annullers of the token flow matrix $\boldsymbol{C}$, where $\boldsymbol{C} = \boldsymbol{Post} - \boldsymbol{Pre}$, are called T- and P-*semiflows*, respectively. A semiflow $\mathbf{v}$ is *minimal* when its support, $\|\mathbf{v}\| = \{i \mid \mathbf{v}(i) \neq 0\}$, is not a proper superset of the support of any other semiflow, and the greatest common divisor of its elements is one. If there exists $\boldsymbol{y} > 0$ such that $\boldsymbol{y} \cdot \boldsymbol{C} = 0$, the net is said to be *conservative*, and if there exists $\boldsymbol{x} > 0$ satisfying $\boldsymbol{C} \cdot \boldsymbol{x} = 0$, the net is said to be *consistent*.

The enabling degree of a transition $t_j \in T$ is:

$$enab(t_j, \boldsymbol{m}) = \begin{cases} \min\limits_{p_i \in {}^\bullet t_j} \left\lfloor \dfrac{m(p_i)}{\boldsymbol{Pre}(p_i, t_j)} \right\rfloor & \text{if } t_j \in T^d \\ \min\limits_{p_i \in {}^\bullet t_j} \dfrac{m(p_i)}{\boldsymbol{Pre}(p_i, t_j)} & \text{if } t_j \in T^c \end{cases} \tag{1}$$

where $\lfloor x \rfloor$ denotes the largest integer not greater than x.

Transition $t_j \in T$ is *enabled* at $\boldsymbol{m}$ iff $enab(t_j, \boldsymbol{m}) > 0$. An enabled transition $t_j \in T$ can fire in any amount α such that $0 \leq \alpha \leq enab(t_j, \boldsymbol{m})$, where $\alpha \in \mathbb{N}$ if $t_j \in T^d$ and $\alpha \in \mathbb{R}$ if $t_j \in T^c$. Such a firing leads to a new marking $\boldsymbol{m'} = \boldsymbol{m} + \alpha \cdot \boldsymbol{C}(\cdot, t_j)$, where $\boldsymbol{C} = \boldsymbol{Post} - \boldsymbol{Pre}$ is the token-flow matrix and $\boldsymbol{C}(\cdot, t_j)$ is its j^{th} column. If $\boldsymbol{m}$ is reachable from $\boldsymbol{m}_0$ through a finite sequence σ, the *state (or fundamental) equation*, $\boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}$ is satisfied, where $\boldsymbol{\sigma} \in \mathbb{R}_{\geq 0}^{|T|}$ is the firing count vector. According to this firing rule the class of nets defined in Def 2.1 is equivalent to the class of nets defined in [8, 5].

Two enabled transitions t_i and t_j are in conflict at $\boldsymbol{m}$ if there exists $p \in {}^\bullet t_i \cap {}^\bullet t_j$ such that $Pre(p, t_i) + Pre(p, t_i) > m(p)$. For this, it is necessary that ${}^\bullet t_i \cap {}^\bullet t_j \neq \emptyset$, and in that case it is said that t_i and t_j are in structural conflict relation. In order to establish a conflict resolution policy, a natural number η_j can be associated to each transition t_j. This number can represent either priority or probability. If it is a priority, the transition with lowest value in a set of conflicting transition will fire first. If it is a probability, the probability to fire t_j first is equal to η_j divided by the sum of the η_i of the conflicting transitions.

2.2 Timed Hybrid Petri nets

Different time interpretations can be associated to the firing of transitions. Once an interpretation is chosen, the state equation can be used to show the dependency of the marking on time, i.e., $\boldsymbol{m}(\tau) = \boldsymbol{m_0} + \boldsymbol{C} \cdot \boldsymbol{\sigma}(\tau)$. The term $\boldsymbol{\sigma}(\tau)$ is the firing count vector at time τ. Depending on the chosen time interpretation, the firing count vector $\sigma_j(\tau)$ of a transition $t_j \in T^c$ is differentiable with respect to time, and its derivative $f_j(\tau) = \dot{\sigma_j}(\tau)$ represents the *continuous flow* of t_j. As for the timing of discrete transitions, several definitions exist for the flow of continuous transitions.

Definition 2.2 *A* Timed Hybrid Petri Net (THPN) system *is a 4-tuple* $\langle \mathcal{N}, \boldsymbol{m}_0, Type, \boldsymbol{\lambda} \rangle$ *where* $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ *is a HPN,* $Type : T \to \{id, pd, dd, ic, pc\}$ *establishes the time semantics of transitions and* $\boldsymbol{\lambda} : T \to \mathbb{R}_{\geq 0}$ *associates a real parameter to each transition related to its semantics.*

Any of the following semantics is allowed for a discrete transition $t_i \in T^d$:

- *Infinite server semantics* ($Type(t_i) = id$): Under infinite server semantics, the time delay of a transition t_i, at a given marking $\boldsymbol{m}$, is an exponentially distributed random variable with parameter $\lambda_i \cdot enab(t_i, \boldsymbol{m})$, where the integer enabling $enab(t_i, \boldsymbol{m})$ represents the number of active servers of t_i at marking $\boldsymbol{m}$.

- *Product server semantics* ($Type(t_i) = pd$): Under product server semantics, the time delay of a transition t_i at $\boldsymbol{m}$ is an exponentially distributed random variable with parameter $\lambda_i \cdot \prod_{p_j \in {}^\bullet t_i} \left\lfloor \frac{\boldsymbol{m}(p_j)}{\boldsymbol{Pre}(p_j, t_i)} \right\rfloor$, where $\prod_{p_j \in {}^\bullet t_i} \left\lfloor \frac{\boldsymbol{m}(p_j)}{\boldsymbol{Pre}(p_j, t_i)} \right\rfloor$ is the number of active servers.

- *Deterministic delay* ($Type(t_i) = dd$): A transition t_i with deterministic delay is scheduled to fire $1/\lambda_i$ time units after it became enabled.

Conflict resolution: When several discrete exponential transitions, under either infinite or product server semantics, are in conflict, a racing policy is adopted, i.e., the one with smaller time delay will fire first.

If a discrete transition with deterministic delay is not in conflict with other transitions, it is fired as scheduled, if it is in conflict then it is fired only if its schedule firing time is less than the firing time of the conflicting transition. In the case of several conflicting deterministic transitions with same scheduled firing instance, the transition to fire is chosen probabilistically assigning the same probability to each conflicting transition. Furthermore after the firing of a deterministic transition, the timers of all the transitions in the same conflict are discarded.

For a continuous transition $t_i \in T^c$ the following semantics are allowed:

- *Infinite server semantics* ($Type(t_i) = ic$): Under *infinite server* the flow of a transition t_i is:

$$f_i = \lambda_i \cdot enab(t_i, \boldsymbol{m}) = \lambda_i \cdot \min_{p_j \in {}^\bullet t_i} \left\{ \frac{m(p_j)}{\boldsymbol{Pre}(p_j, t_i)} \right\} \tag{2}$$

  Such an expression for the flow is obtained from a first order approximation of the discrete case [28] and corresponds to the *variable speed* of [3].

- *Product server semantics* ($Type(t_i) = pc$): In a similar way to discrete transitions, the continuous flow under product server semantics is given by:

$$f_i = \lambda_i \cdot \prod_{p_j \in {}^\bullet t_i} \left\{ \frac{m(p_j)}{\boldsymbol{Pre}(p_j, t_i)} \right\}$$

The described supported semantics cover the modeling of a large variety of actions usually associated to transitions. For instance, infinite server semantics, which is more general than finite server semantics, is well suited for modeling actions in manufacturing, transportation and logistic systems [8]; product server semantics is especially useful to modeling population dynamics [27] and biochemical reactions [11]; and deterministic delays allow one to represent pure delays and clocks that appear, for instance, when modeling traffic lights in automotive traffic systems [30].

3 Discrete nets in Petri Net Toolbox

The *Petri Net Toolbox* [15, 14, 13] (abbreviated as *PN Toolbox*) for MATLAB bridges the Petri net formalism with the widely spread usage of MATLAB. It accepts three types of PN models, namely: untimed, transition-timed and place-timed. The timed nets can be deterministic or stochastic, and the stochastic case allows using the appropriate probability distribution function with positive support. The capacity of a place can be finite or infinite. The user can set priorities / probabilities for conflicting transitions. Both single and infinite server semantics can be used for transition firings. The *PN Toolbox* can also operate with stochastic and generalized stochastic PNs [2].

An easy to exploit *Graphical User Interface* GUI (see Fig. 1) allows the user to get hold of the facilities offered by the PN Toolbox. The GUI may be utilized in two modes: first, the *Draw Mode* allows the user to draw a new PN model or modify the properties of an existing one in a natural fashion; second, the *Explore Mode* enables access to simulation, analysis and design tools, exploiting the computational resources of the MATLAB environment.

The GUI exhibits eight control panels (see Fig. 1): *Menu Bar* (1), *Quick Access Toolbar* (2), *Drawing Area* (3), *Drawing Panel* (4), *Draw/Explore Switch* (5), *Simulation Panel* (6), *Status Panel* (7) and a *Message Box* (8).

The user can access all the facilities available in the PN Toolbox from the *Menu Bar* placed horizontally, on top of the main window of the GUI. It displays a set of nine drop-down menus, namely *File, Modeling, View, Properties, Simulation, Per-*

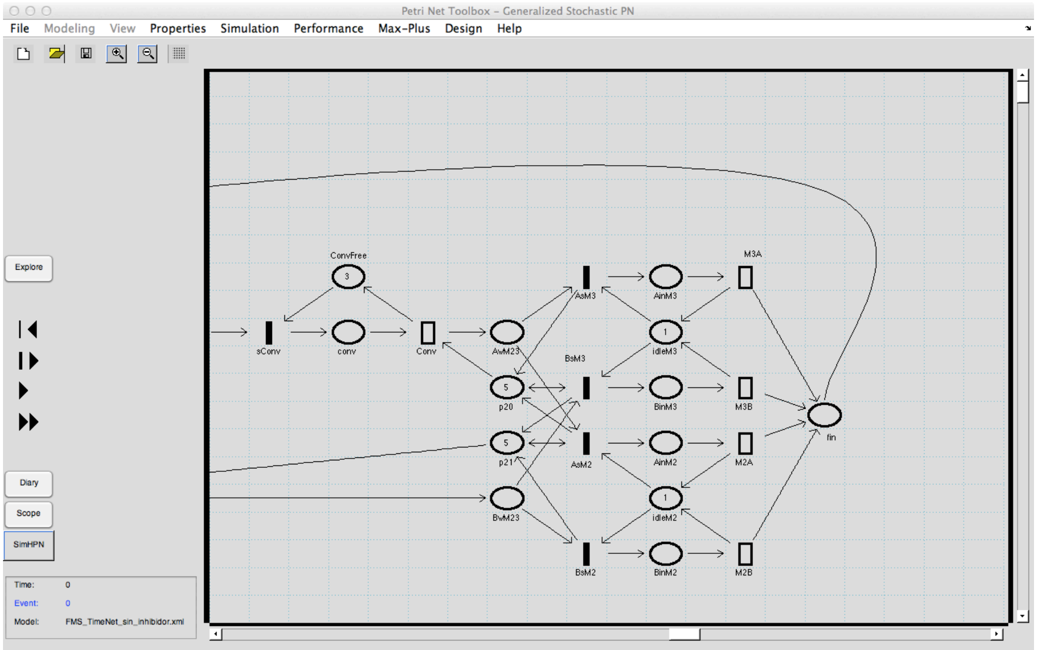

Figure 1: GUI of the Petri net Toolbox.

formance, Max-Plus, Design and *Help*, that are enabled depending on the exploitation mode of the PN Toolbox.

The *File* menu offers facilities for file-handling operations such as creating a new model, loading a previously saved model, saving, printing or closing a model, and exiting the GUI. The *Modeling* menu provides tools for graphical editing a model in the *Drawing Area*. After adding places or transitions and drawing arcs between two different nodes of the current PN model, the user can add tokens in a specific place or edit the properties of different net objects. The user may also set the policy for the resolution of conflicting transitions assigning priorities or probabilities to conflicting transitions.

The *View* menu allows choosing specific conditions for visualization of the current model by zooming in or out, by displaying or hiding the current values of arc weights. The desired representation of a transition as a square, rectangle or bar can also be selected by the user.

The *Properties* menu provides computational tools for the analysis of the behavioral and structural properties of the current PN model. After drawing a PN model, the user can: a) visualize the *Incidence Matrix*, which is automatically built from the net topology; b) explore the *Behavioral Properties* (such as liveness, boundedness, reversibility etc.) by consulting the *Coverability Tree*, which is automatically built from the net topology and initial marking; c) explore the *Structural Properties* (such as structural boundedness, repetitiveness, conservativeness and consistency): d) calculate T- and P-*semiflows*.

Using the *Simulation* menu the user can simulate the PN model in three ways: *step by step* and *run slow*, which are accompanied by animation, and *run fast*. This menu also allows the user to set the conditions of the simulation and to store or retrieve the simulation log file.

At the end of a simulation experiment, the *Performance* menu allows the visualization of the *global performance indices* that are separately recorded for transitions (such as total number of firings, average frequency of firings, etc.) and for places (such as total number of arrived/ departed tokens, average frequency of token arrival/ departure, average number of tokens in place, average waiting time per token, etc.). These indices may be saved in *HTML* format in a file placed in the working directory.

The *Max-Plus* menu allows performing the simulation and analysis of a place-timed event graph based on its max-plus state-space model derived directly from its topology and initial marking in an implicit form [4].

For the synthesis, via automated iterative simulations, of timed or (generalized) stochastic PN models, the *PN Toolbox* offers the menu (command) *Design*, which allows exploring the dependence of a *Design Index* on one or two *Design Parameters* that vary within intervals defined by the user. A *Design Parameter* may be selected as (*i*) the initial marking of a place, (*ii*) a parameter of the distribution function defining the duration associated with a place or a transition in timed PNs, (*iii*) the mean value of the exponential distribution function associated with a transition in (generalized) stochastic PNs. The *Design Index* may be selected as a global performance index associated with any node of the net.

The *Help* menu provides complete information about the *PN Toolbox* and allows visualization of four movies initiating the user in its exploitation.

The *Quick Access Toolbar* is placed horizontally, below the *Menu Bar*, and presents six image buttons that give access to the highly used commands *New Model, Open Model, Save, Zoom In, Zoom Out* and *Show Grid*.

The *Drawing Area* is located in the central and right side of the main window. It is provided with a grid, where the nodes of the PN graph are to be placed, and with two scrollbars (on the right and bottom sides) for moving the desired parts of the graph into view.

The *Draw/Explore Switch* allows switching between the *Draw Mode* (in which the user can draw a new model or modify an existing one) and the *Explore Mode* (in which the user can access all functions available for simulation and analysis). When switching from one exploitation mode to the other, different menus become available in the *Menu Bar*.

In *Draw Mode*, for a quick access to the main commands used for drawing a model, the *PN Toolbox* is equipped with a *Drawing Panel* placed vertically, in the left side of the main window, above the *Draw/Explore Switch*. In *Explore Mode*, the *Drawing Panel* is hidden and the user gains access to the *Simulation Panel* placed vertically, in the left side of the main window, just below the *Draw/Explore Switch*.

The *Status Panel* is a message board (placed in the bottom left-hand corner of the main window), where the *PN Toolbox* displays the current simulation time and the total number of events. In addition, the *Status Panel* displays the file name corresponding to the current model.

The *Message Box* is a MATLAB text object used by the *PN Toolbox* to display different messages to the user, depending on the exploitation mode.

4 Continuous and hybrid nets in SimHPN

The *SimHPN* simulator [12] supports infinite server and product server semantics for both discrete and continuous transitions. Moreover, deterministic delays with single server semantics are also supported for discrete transitions. Both the data related to the model description, i.e., net structure, initial marking and timing parameter, and the output results, i.e., time trajectories, are MATLAB variables. At the end of the simulation, the user can export the data to the MATLAB workspace where it can be used for further analysis. The next subsection describes the functionality of the tool as shown in its graphical interface.

The *SimHPN* toolbox provides a Graphical User Interface (GUI) that enables the user to easily perform simulations and carry out analysis methods. This GUI consists of a MATLAB figure window, exhibiting a *Menu bar* and three control panels: (i) *Drawing Area*, (ii) *Options panel*, and (iii) *Model Management panel*. Fig. 2 presents a hard-copy screenshot of the main window opened by *SimHPN* toolbox, where all the component parts of the GUI are visible.

The *Menu bar* (placed horizontally, on the top of the window in Fig. 2) displays a set of four drop-down menus at the top of the window, where the user can select different features available in the *SimHPN* toolbox. These menus are: *Model*, *Options*, *Simulation*, and *Optimal*.

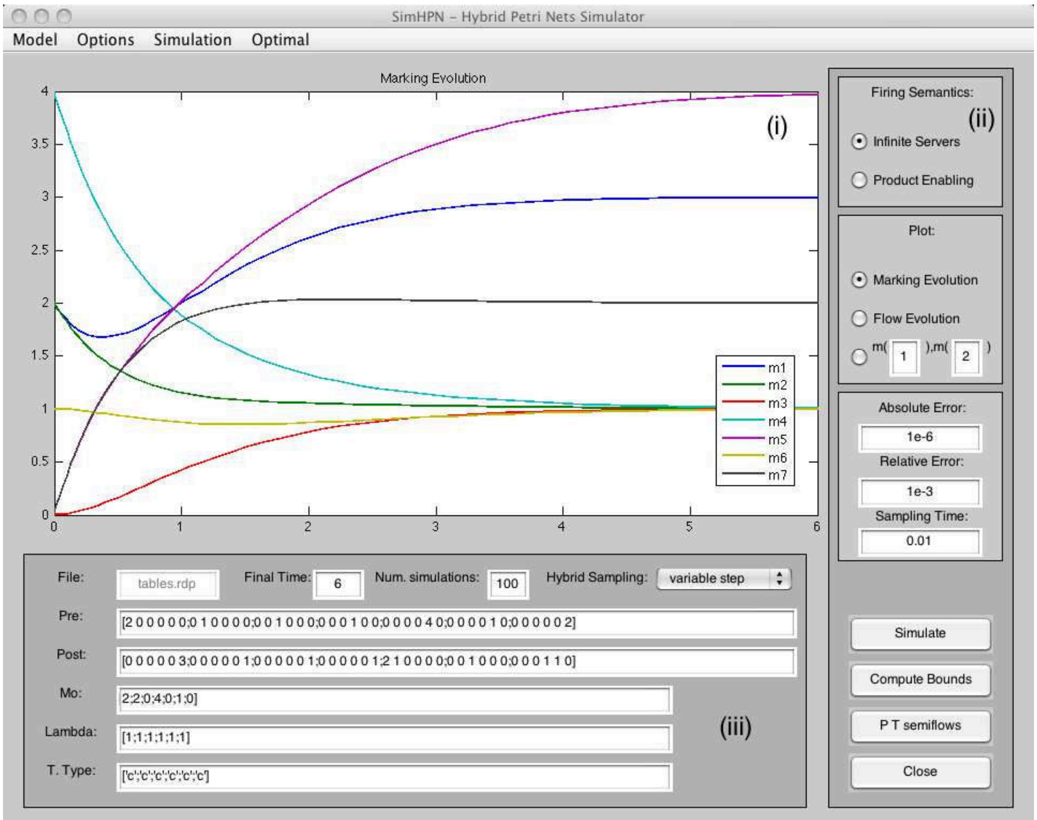

Figure 2: Sketch of the main window of *SimHPN*

The *Model menu* contains the pop-up menus *Import from Pmeditor*, *Import from TimeNet* and *Import from .mat file* that implement several importing options for the matrices, $\boldsymbol{Pre}$, $\boldsymbol{Post}$, $\boldsymbol{m}_0$, etc, that describe the net system: Such matrices can be introduced manually or through two Petri nets editors: PMEditeur and TimeNet [34]. Moreover, the matrices can be automatically loaded from a *.mat* file (MATLAB file format) or loaded from variables defined in the workspace, this is done just by writing the name of the variable to be used in the corresponding edit boxes.

The *Options* menu contains only the pop-up menu *Show Figure Toolbar* allows to show the characteristic toolbar of the MATAB figure object that permits, for example, the use of zoom tool on the displayed graphic in the *Drawing Area*.

The *Simulation menu* contains the pop-up menus *Markings to plot*, *Flows to plot*, and *Save results to workspace*. The pop-up menus *Markings to plot*, *Flows to plot* allow the user to select the components of marking vector and flow vector that will be plotted after simulation in the *Drawing area*. The pop-up menu *Save results to workspace* permits to export, after simulation, the marking and flow evolution to variables in the MATLAB workspace.

The *Optimal* menu contains the pop-up menus *Optimal Observability* and *Optimal Control*. Such pop-up menus perform calls to the algorithms for computing optimal steady state and optimal sensor placement for continuous Petri nets with infinite server semantics.

The *Drawing area* (located in the left and central side of the window in Fig. 2), is a MATLAB axes object where the trajectories of the simulation results are plotted. The components of markings and flows that will be represented are selected from the menu.

The *Options panel* (placed, as a horizontal bar, on the right part of the window Fig. 2) presents a number of options related to the model. From top to bottom: (a) two radio buttons to select the firing semantics for continuous and discrete exponential transitions; (b) three radio buttons allowing to select the variables to be plotted in the *Drawing Area*, 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; (c) three edit boxes to fix the maximum absolute and relative errors allowed by the simulated trajectory and the sampling time used in simulations (see next subsection for more details on the selection of the sampling time); (d) a *Simulate* button to start a new simulation; (e) a *Compute Bounds* button that computes performance bounds for continuous nets under infinite server semantics; (f) a *P T semiflows* button to compute the minimal P- and T-semiflows of the net, the results are displayed on the MATLAB command window and can be used for future analysis tasks; and (g) a *Close* button to close the $SimHPN$ toolbox.

The *Model Management Panel* panel is composed of different edit boxes (placed in the bottom left corner of the window in Fig. 2), where the $SimHPN$ toolbox displays the current values of the matrices describing the net system and permits to select the simulation time and the number of simulations to be performed (this last parameter is ignored if the net contains no stochastic transitions). The required matrices for a system in order to be simulated are: $\boldsymbol{Pre}$ and $\boldsymbol{Post}$ matrices, initial marking $\boldsymbol{m}_0$, the parameter $\boldsymbol{\lambda}$ of each transition, and the type of each transition. This last parameter is equal to 'c' for continuous transitions, to 'd' for stochastic discrete transitions and to 'q' for deterministic discrete transitions. Notice that if the type of a transition is 'q' then single server semantics is adopted for its firing and therefore the selection of firing semantics in the *Options panel* will be ignored for this transition.

5 Hybrid models in HYPENS and PNSB

The *HYbrid PEtri Net Simulator* (HYPENS) [23] is a tool that allows to simulate several classes of timed discrete, continuous and hybrid Petri nets. More precisely, the class of First Order Hybrid Petri Nets is considered. In this class of nets, the flow of the continuous transitions is piecewise constant and is obtained by solving a linear programming problem. Both finite and infinite server semantics can be associated to the firing of discrete transitions. The tool can be easily interfaced with other MATLAB programs and can also be used for analysis and optimization via simulation. The large set of plot functions available in MATLAB allows one to represent the results of the simulation in a clear and intuitive way.

The *Petri Net Simulink Block* (PNSB) [16, 14] ensures the integration of the simulation facilities available in the *PN Toolbox* with Simulink and facilitates the simulation of hybrid systems when the event-driven part of their dynamics is modeled using the Petri net formalism. The PN model stored in a PNSB can be untimed, P-timed or T-timed. It contains transitions whose firing is synchronized (triggered) with external events generated by a set of input signals. A *synchronized transition* is fired whenever (i) it is enabled by the net marking and (ii) one of its associated triggering events, defined at the PN level, occurs. Both finite and infinite server semantics can be used for transition firings. A Simulink model can contain any number of PNSBs needed to model a complex system. The PNSB is equipped with a GUI that allows a user to draw a PN model (PNSB Editor), define the triggering events (PNSB Event Explorer) and debug the Simulink model (PNSB Debugger). The functions of the PNSB Editor are similar to the editing facilities available in the *Draw Mode* of the PN *Toolbox*.

This approach to hybrid systems modeling and analysis allows incorporating accurate nonlinear models for the continuous dynamics (built from blocks available in the standard Simulink libraries) with PN models for discrete event dynamics.

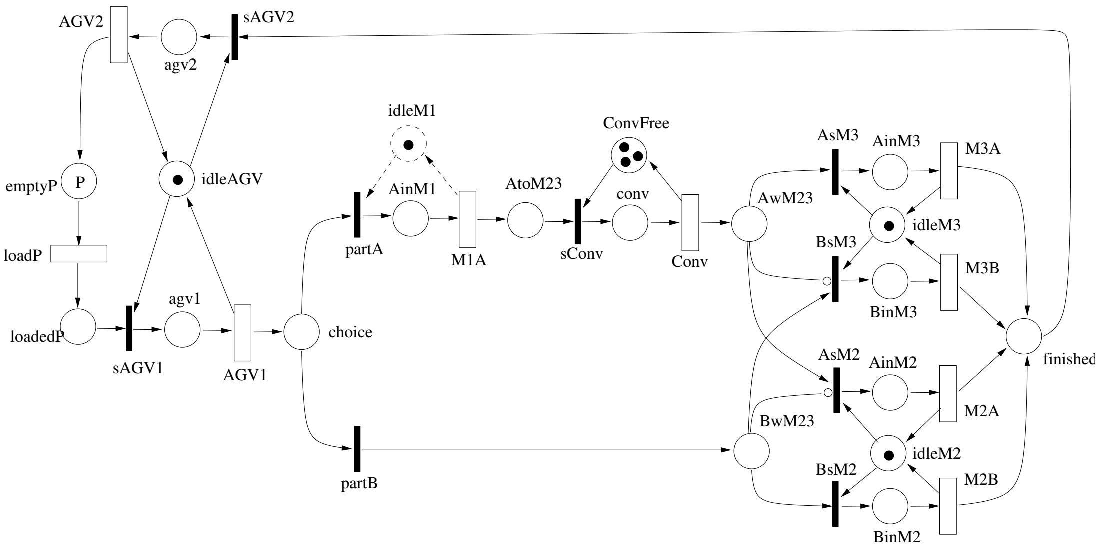

Figure 3: A GSPN model of a FMS taken from [33].

6 Case study

Let us consider the Generalized Stochastic Petri Net (GSPN) [2] model of a Flexible Manufacturing System (FMS) adapted from [33] and presented in Fig. 3. The FMS consists of three machines called M_1, M_2 and M_3, and handles two types of products named A and B. Parts of type A are processed first by machine M_1. Then, they are processed by M_3, but if M_2 is not processing a product of type B, i.e., M_2 is idle, it can be processed on M_2 as well. On the other hand, parts of type B are processed by M_2 but if M_3 is idle, it can be processed on M_3 as well. The production of a product of type A is modeled by place $AinM1$ (production by M_1) while the activity of manufacturing is modeled by transition $M1A$. The products are then transported by a conveyor with capacity of three products (place *conv*) and then in place $AwM23$ a decision is taken. Products are moved to M_3 by firing $AsM3$ or to M_2 by firing $AsM2$. Notice that transition $AsM2$ has an input inhibitor arc (an inhibitor arc from p_i to t_j with weight w disables t_j if $m(p_i) > k$) enabling the transition only if there are no products of part B in place $BwM23$. For products of type B, in place $BwM23$ a decision is taken. The products are produced on M_2 by firing $BsM2$ or on M_3 by firing $BsM3$ (if no products of type A are waiting to be processed in place $AwM23$).

In place *choice* there is a decision on which type of product should be produced. There is a conflict between transitions *partA* and *partB* and a fixed percentage is given: 30% to produce products of part A (transition *partA*) and 70% for *partB*. An *AGV* is available in order to unload the finished products and load the first machines.

Since the model contains inhibitor arcs, many structural properties cannot be directly computed because this information is not presented in the token flow matrix. If the net is bounded, the inhibitor arcs can be removed by using a complementary place. To apply this construction the maximum number of tokens in the input place of the inhibitor arcs must be known. This can be obtained from *PN Toolbox* by exploiting the reachability graph. For a number of pallets $P = 5$, the reachability graph has 6582 nodes and the maximum number of tokes in places $AwM23$ and $BwM23$ is 5. Inhibitor arcs $(AwM23, BsM3)$ and $(BwM23, AsM2)$ can be replaced by using the complementary places p_{20} and p_{21} and connected to the other nodes as in Fig. 1. Notice that transition $BsM3$ can be fired only if place p_{20} has 5 tokes that is equivalent to have zero tokens in $AwM23$.

The *PN Toolbox* allows the check of different structural properties. For example, the P and T semiflows. In this case, there are 7 P-semiflows (associated to the conservation of the following resources: main process stream, AGV, conveyor, M_2, M_3 and 2 associated to the complementary places introduced: p_{20} and p_{21}) and 4 T-semiflows (corresponding to the four repetitive components). Each place is in the support of at least one P-semiflow meaning that the net is conservative. Furthermore, all transitions are in the support of at least one T-semiflow, hence the net is consistent. These structural propertes can be easily checked by the *PN Toolbox*.

The main facility of the *PN Toolbox* is the simulation of the GSPN system. Assuming the same firing rate for all transition equal to 1, single server semantics for transition $M1A$ and infinite server semantics to the other transitions, and simulating for $10,000$ time units, the number of products produced is equal to $4,846$ (*Service Sum* index of transition $sAGV2$), from which $1,454$ products of part A (*Service Sum* of transitions $M3A$ and $M2A$) and $3,393$ products of part B (*Service Sum* of transitions $M3B$ and $M2B$). The throughput of the system is obtained by consulting the index *Service rate* of transition $AGV2$. In this case, it is equal to 0.4845.

An important feature of the *PN Toolbox* is the possibility of running a design experiment. Assume that we want to see how the throughput of the system changes when the firing rate of transition $M1A$ decreases from 1 to 0.2 (the average of the firing delay is varying from 1 to 5). The results are sketched in Fig.4 where it is easy to see that the throughput is almost constant when the firing rate of $M1A$ is between 0.33 and 1.

The GSPN can be automatically exported to *SimHPN* by using the corresponding pushdown button from *PN Toolbox*. However, in the considered continuous nets, all transitions must have the same firing semantics, in particular infinite server semantics. For this reason, in the GSPN in Fig. 3, place $idleM1$ is introduced to model the finite server semantics of $M1A$. Since the initial marking of $idleM1$ is one, the enabling degree of $M1A$ is upper bounded by 1, and the resulting net has the desired behavior. The throughput of the continuous system in the steady state computed by $SimHPN$ is 0.497512, which is very similar to the one of the discrete system.

7 Conclusions

Petri nets are widely used in industry and academy for the modeling and analysis of dynamical systems. Although Petri nets were originally oriented (and somehow constrained) to discrete event systems, the advent of continuous and hybrid Petri nets has further widened the application scope of the paradigm and has simplified the study of large discrete systems as they may admit a suitable continuous approximation.

This paper has presented some tools that offer modeling, analysis and simulation functions for discrete, continuous and hybrid Petri nets. More precisely, the main focus has been on the *Petri Net Toolbox* which is mainly oriented to discrete Petri nets, and on $SimHPN$, which is a complementary tool mainly oriented to continuous and hybrid Petri nets. Thus, the choice of the appropriate software tool strongly depends on the nature (discrete, continuous or hybrid) of the model to be analyzed. Besides the existing algorithms for analysis and simulation, both tools offer a clear an intuitive graphical interface which makes the user easier to grasp the way the tools work. In addition to the mentioned tools, a brief discussion of two other MATLAB tools has been included, namely *Hybrid Petri Net Siulator* (HYPENS) and *Petri Net Simulink Block* (PNSB).

References

[1] M. Ajmone Marsan, G. Balbo, and G. Conte. A class of generalized stochastic Petri nets for the performance analysis of multiprocessor systems. *ACM Trans. on Computer Systems*, 2(2):93–122, 1984.

[2] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. Wiley, 1995.

[3] H. Alla and R. David. Continuous and hybrid Petri nets. *Journal of Circuits, Systems, and Computers*, 8(1):159–188, 1998.

[4] F. Baccelli, G. Cohen, G. J. Olsder, and J.-P. Quadrat. *Synchronization and linearity : an algebra for discrete event systems*. Wiley series in

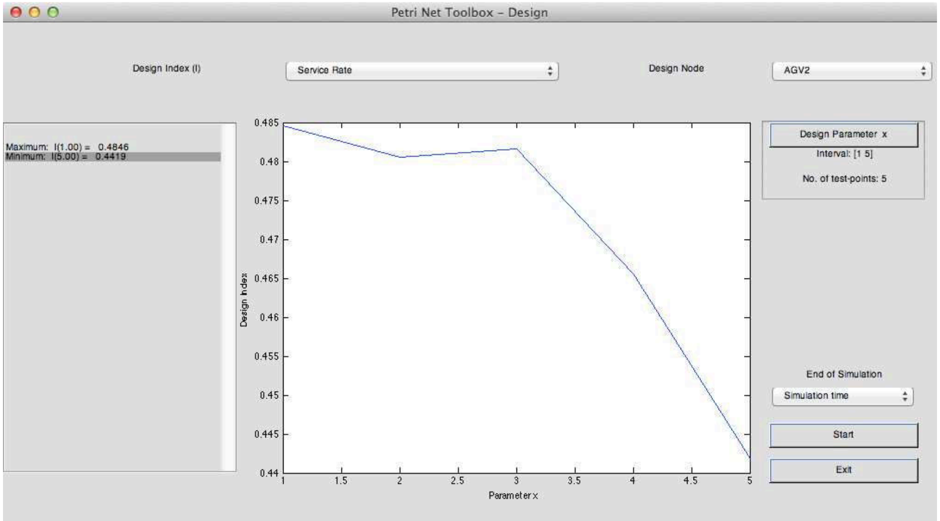

Figure 4: A design experiment carried out by the Petri net Toolbox.

probability and mathematical statistics. J. Wiley & Sons, Chichester, New York, 1992.

[5] 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.

[6] J. Campos, C. Seatzu, and X. Xie, editors. *Formal Methods in Manufacturing.* CRC Press, 2014.

[7] R. David and H. Alla. Continuous Petri nets. In *Proc. of the 8th European Workshop on Application and Theory of Petri Nets*, pages 275–294, Zaragoza, Spain, 1987.

[8] R. David and H. Alla. *Discrete, Continuous and Hybrid Petri Nets.* Springer-Verlag, 2010. 2^{nd} edition.

[9] A. A. Desrochers and R. Y. Al-Jaar. *Applications of Petri nets in manufacturing systems : modeling, control, and performance analysis.* IEEE. IEEE Press, Piscataway, 1995.

[10] F. DiCesare, G. Harhalakis, J. M. Proth, M. Silva, and F. B. Vernadat. *Practice of Petri Nets in Manufacturing.* Chapman & Hall, 1993.

[11] M. Heiner, D. Gilbert, and R. Donaldson. Petri nets for systems and synthetic biology. In M. Bernardo, P. Degano, and G. Zavattaro, editors, *Formal Methods for Computational Systems Biology*, Lecture Notes in Computer Science, pages 215–264. Springer Berlin, Heidelberg, 2008.

[12] J. Júlvez, C. Mahulea, and C.-R. Vázquez. SimHPN: A MATLAB toolbox for simulation, analysis and design with hybrid Petri nets. *Nonlinear Analysis: Hybrid Systems*, 6(2):806 – 817, 2012.

[13] C. Mahulea, L. Barsan, and O. Pastravanu. Matlab tools for Petri-net-based approaches to flexible manufacturing systems. In *In: F.G. Filip, I. Dumitrache and S. Iliescu (Eds.), 9th IFAC Symposium on Large Scale Systems LSS 2001*, pages 18–20, 2001.

[14] M. H. Matcovschi, C. Mahulea, C. Lefter, and O. Pastravanu. Petri net toolbox in control engineering education. In *2006 IEEE International Conference on Computer Aided Control System Design*, pages 2298–2303, Munich, Germany, 2006.

[15] M. H. Matcovschi, C. Mahulea, and O. Pastravanu. Petri Net Toolbox for MATLAB. In *11th Mediterranean Conference on Control and Automation MED'03*, Rhodes, Greece, 2003.

[16] M. H. Matcovschi, C. Popescu, and O. Pastravanu. A new approach to hybrid system simulation: Development of a Simulink library for Petri net models. *Control Engineering and Applied Informatics*, 7(4):55 – 62, 2005.

[17] P. Merlin. *A study of the Recoverability of Computer Systems.* PhD thesis, Univ. California, Irvine, CA, USA, 1974.

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

[19] T. Murata. Temporal uncertainty and fuzzy-timing high-level Petri nets. In J. Billington and W. Reisig, editors, *Application and Theory of Petri Nets 1996*, volume 1091 of *Lecture Notes in Computer Science*, pages 11–28. Springer, 1996.

[20] C. Petri. *Kommunikation mit Automaten (Communication with Automata).* PhD thesis, Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2, 1962. Second Edition:, New York: Griffiss Air Force Base, Technical Report RADC-TR-65–377, Vol.1, 1966, Pages: Suppl. 1, English translation.

[21] C. Ramchandani. Analysis of asynchronous concurrent systems by Petri nets. Technical Report Project MAC, TR-120, M.I.T., Cambridge, MA, USA, 1974.

[22] C. Seatzu, M. Rez, M. Silva, and J. Van Schuppen. *Control of Discrete-Event Systems: Automata and Petri Net Perspectives.* Lecture Notes in Control And Iinformation Sciences. Springer London, Limited, 2012.

[23] F. Sessego, A. Giua, and C. Seatzu. HYPENS: a Matlab tool for timed discrete, continuous and hybrid Petri nets. In 29^{th} *International Conference on Application and Theory of Petri Nets (ICATPN 2008)*, volume 5062 of *Lecture Notes in Computer Science*, pages 419–428. Springer, Xi'an, China, June 2003.

[24] J. Sifakis. Uses of Petri nets for performance evaluation. In *Measuring, Modelling, and Evaluating Computer Systems*, pages 75–93. North-Holland, 1977.

[25] M. Silva. Half a century after Carl Adam Petri's Ph.D. thesis: A perspective on the field. *Annual Reviews in Control*, 37(2):191 – 219, 2013.

[26] M. Silva and J. Colom. On the structural computation of synchronic invariants in P/T nets. In *Proc. of the 8th European Workshop on Application and Theory of Petri Nets*, pages 237–258, Zaragoza, Spain, 1987.

[27] M. Silva and L. Recalde. Réseaux de Petri et relaxations de l'integralité: Une vision des réseaux continus. In *Conférence Internationale Francophone d'Automatique (CIFA 2000)*, pages 37–48, 2000.

[28] M. Silva and L. Recalde. Petri nets and integrality relaxations: A view of continuous Petri nets. *IEEE Trans. on Systems, Man, and Cybernetics*, 32(4):314–327, 2002.

[29] M. Svádová and Z. Hanzálek. Matlab toolbox for Petri nets. In *22nd International Conference ICATPN 2001*, pages 32–36, 2001.

[30] C. Vázquez, H. Sutarto, R. Boel, and M. Silva. Hybrid Petri net model of a traffic intersection in an urban network. In *2010 IEEE Multiconference on Systems and Control*, Yokohama, Japan, September 2010.

[31] B. Walter. *Transaktionsorientierte Recovery-Konzepte für verteilte Datenbanksysteme.* Inst.für Informatik, Universitat Stuttgart, 1982.

[32] M. C. Zhou and F. DiCesare. *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems.* Kluwer Academic Publishers, 1993.

[33] A. Zimmermann. *Stochastic Discrete Event Systems - Modeling, Evaluation, Applications.* Springer, Berlin Heidelberg New York, 2007.

[34] A. Zimmermann and M. Knoke. Timenetsim - a parallel simulator for stochastic Petri nets. In *Proc. 28th Annual Simulation Symposium*, pages 250–258, Phoenix, AZ, USA, 1995.