Production Chain Modeling based on Learning Flow Stochastic Petri Nets

Walid Ben Mesmia, Kamel Barkaoui

Production Chain Modeling based on Learning Flow Stochastic Petri Nets

Walid Ben Mesmia (benmesmiawalid77@yahoo.fr)
ENIT: Ecole Nationale d'Ingenieurs de Tunis https://orcid.org/0000-0003-1270-5916
Kamel Barkaoui
CNAM: Conservatoire National des Arts et Metiers

Research Article

Keywords: Machine learning, Multi-agent system, Stochastic Petri Nets, Specification, Verification

Posted Date: January 24th, 2024

DOI: https://doi.org/10.21203/rs.3.rs-3839753/v1

Production Chain Modeling based on Learning Flow Stochastic Petri Nets

Walid Ben Mesmia[1*] and Kamel Barkaoui[2†]

[1*]LR-Sys'Com-ENIT, ENIT, Tunis, 1002,Tunisia.
[2]CEDRIC-CNAM, Paris, France.

*Corresponding author(s). E-mail(s): benmesmiawalid77@yahoo.fr;
Contributing authors: kamel.barkaoui@cnam.fr;
†

Abstract

In this study, we propose a model called ***LFSPN***, which serves as an extension of stochastic Petri nets dedicated to the multi-agent systems paradigm. The main objective is to specify, verify, validate, and evaluate the flow of materials within an automated production chain. We illustrate the practicality of our model by engaging in a systematic process of modeling and simulation of a production chain involving material flow. To evaluate the performance, we employ a mobile learning agent, which has distinct characteristics, namely mobility and learning. The distinctive characteristics of the learning agent are manifested in two key behaviors: mobility and learning. Notably, the learning agent is equipped with a flexible learning algorithm that integrates stochastic elements based on transitions. We validate the effectiveness of our model by performing a comparative analysis with similar existing works. The advantages of our ***LFSPN*** model are twofold. Firstly, it offers a representation with two levels of abstraction: a graph representing the classic components of an SPN, and an additional layer encompassing the learning and migration aspects inherent to a mobile learning agent. Secondly, our model stands out for its flexibility and simulation simplicity.

Keywords: Machine learning, Multi-agent system, Stochastic Petri Nets, Specification, Verification

1 Introduction

According to the great development of the machine learning technics, they were used to assessing corporate activity in many competitive areas in order to gain an advantage on multiple levels, including productivity, performance, and even performability. As a result, machine learning is applicable in the fields of healthcare, manufacturing, real estate, and logistics [1, 2]. Machine learning [1], or other annotated automatic learning, is one of the rational fields of science and is defined as a subclass of artificial intelligence. So, it is based on allowing algorithms to discover patterns.In addition, Machine Learning algorithms perform their own learning in order to perform a task or make predictions from data. So they improve their performance over time. Once trained, the algorithm will be able to find patterns in new data. We can annotate that learning depends on the time and the learning algorithm efficiency [2, 3].
In [4], the authors propose a modeling approach to visualize the machine learning prediction process using colored Petri nets. They base their learning phases on a Petri net structure by feature selection and extract the arc functions specifying the transition trigger [2].

The effectiveness of simulation models for planning and analyzing material flow systems in industrial automation is highlighted by the authors' works in [6, 7] and [4]. Notable simulation models are physics engines, which precisely capture system behavior by incorporating physical data like weight, speed, or friction to represent systems at a fine level [6]. The Petri net model [6], which is frequently used to describe system behavior with concurrent processes [7], is another simulation model that is widely used in production systems.

The action space is specified by transitions, while locations and tokens represent the observation space of the reinforcement learning agent. To confirm the suitability of the timed colored Petri net model, a case study of a simple production facility was carried out. On the other hand, a Q-learning agent was trained on the Petri Net simulation model to perform a simple task. Finally, the authors show the usability of their model through the case study, showing that Petri nets constitute a type of model suitable for training a reinforcement learning algorithm and are capable of faithfully modeling all relevant components of a material flow system [8].
In this model, the action space is defined by transitions, while locations and tokens collectively form the observation space for the reinforcement learning agent. The timed colored Petri net model's applicability was tested by the authors using a case study involving a basic production facility. In this scenario, a Q-learning agent is trained to perform a fundamental task using the Petri Net simulation model. The case study results highlight the usefulness of Petri nets as a model that can be used to train a reinforcement learning algorithm, successfully capturing all relevant elements of a material flow system [8]. By incorporating a stochastic paradigm through the introduction of a random delay in Timed Petri Nets (TPN) [13, 14], which are designed to create a comprehensive model enabling both verification and qualitative and quantitative system analysis, stochastic Petri Nets ($SPNs$) emerge. An SPN is essentially a timed PN with a probability measure applied to the space trajectories. The firing sequences become measurable by considering a random space. Furthermore, the SPN

model is formally defined as a subset of timed petri nets (TPN). This petri nets subclass, introduced by Merlin [16], incorporates the concept of timed transitions within TPNs. The representation of TPN [13], [14] is rooted in the idea of associating a time interval $[a_j, b_j]$ with each transition T_j. If this interval is continuously validated for at least a_j units of time, the transition can potentially be triggered. Additionally, if it is continuously validated for $(b_j - a_j)$ units of time, it must be in the firing state. Moreover, there exist several subclasses of TPN [16], where time can be associated with various elements such as places [13, 14], transitions [17], arcs [19], or combinations of these simultaneously. According to the TPN notation, the firing transitions are not deterministic but occur within firing intervals [20, 21].

While the conventional modeling approach has been advocated for its applicability to global data and the constraints of information processing systems, it faces challenges in assessing performance and adaptability, particularly in the face of technological advancements driving the production chain towards complete automation, adhering to established standards of learning and simulation. In response to these challenges, we present a model utilizing stochastic Petri nets that delineates the production system into two levels. The first level encapsulates work units through transitions, places (stochastic or immediate) and employs classic tokens to represent materials and finished products. The second level introduces mobile autonomous learning agents, capable of dynamic interactions such as integration and migration, as observed in [22], and learning processes as outlined in [9] .

Within this framework, we propose a novel model, named Learning Flow Stochastic Petri Nets ($LFSPN$), which aims to objectively model and assess the performance of the production chain. Our model, grounded in trained agents, facilitates the decomposition of the production system into components aligning with the learning agents. This approach minimizes the autonomy of agents to the intricacies of control within their respective components. Decisions are locally determined based on self-control, allowing agents to make pertinent decisions in real time. While learning agents autonomously influence the nature and extent of interactions, this simplifies the modeling and simulation of the intricate production system. The subsequent section introduces fundamental concepts, followed by the illustration of the $LFSPN$ model in the third section. The subsequent section demonstrates the application of the model in the specification and verification process within an real production system case. The fifth section outlines the simulation methodology, incorporating the Q-learning algorithm, and presents a comparative analysis with similar studies. The conclusion summarizes our findings.

2 Basic Concepts

2.1 Material flow control

The materials flow [30] is a representation that allows us to describe material dynamics as well as interactions with the environment at different scales. Also it is interested at the micro level for an industrial process or an organization (company, administration, etc.)[9], [30]. However, the concept of material flow [9, 10] can be composed into three disciplines:

- Freight transportation, this involves the goods movement via conveyor belts or autonomous vehicles, etc. [9, 10];
- Goods handling, which describes the positioning of an object while maintaining a certain orientation [9, 10];
- Stockpiling is defined as an intentional material flow interruption. [9, 10].

Multi-level automation [30] provides different layers of abstraction to analyze industrial systems. These levels range from the equipment-focused field level to the business planning-focused management level. At the supervisory level, data is collected and monitored. This level also contains supervisory control systems (SC) [30], which are useful in coordinating between processes in such a distributed system. This can ensure the planning and production process orchestration or the storage units coordination [10, 11].
Material flow systems can be distributed systems, which are endowed by a high degree of competition. According to [11], [12], the objective is summarized in the control of simultaneous and distributed systems by supervisory control units.

2.2 Mobile agent

The mobile agent [23], [24] is a logical (software represented by a script) or physical (embedded system) entity instantiated in an environment with [29]:

- Goals: a set of objectives and tendencies sent to the system;
- Capabilities;
- Skills;
- Means of communication;
- A certain degree of intelligence and autonomy [29].

So far, we are still in the case of an agent in the classical definition. But these mobile agents have the opportunity to leave their places of birth to migrate from one site to another in search of information not found on its site of creation or meet other mobile agents to better perform its tasks [22], [24]. We adopt the definition of [22], [24], and [29] which presents a mobile agent as any physical or logical entity satisfying the properties of a classical agent and capable of changing its environment. In the present context, each mobile agent is assigned to a trained agent. The mobile agent checks the owners of a cognitive agent as in [26], [25].
To specify the production system, we integrate SPN design modeling with the semantics of multi-agent systems. Furthermore, our model encompasses the life cycle of mobile agents, employing it to specify the behavior of trained mobile agents in production systems. The subsequent section introduces the proposed model and its associated methods.

3 Proposed model: Learning Flow Stochastic Petri Nets

The **Learning Flow Stochastic Petri Nets**, denoted $LFSPN$, is the 11-uplets: $LFSPN$**:**
$=< P, T, A, \varphi, M_0, MTyp-tk, GTyp-dis, TFringT, S, AC, GLearning,>$
Where:

- P: represents the finished places set;
- T: represents the finished transition set;
  $T= T_i \cup T_s$
  - T_i: immediate transitions set,
  - T_s: stochastic transitions set.
- A: The agents non-finite sets (training type or random type).
- φ: function that determines the firing transition rate,
- M_0: Petri Nets initial marking,
- $MTyp-tk$: The function makes a distinction between the different token types: (trained agent, random agent, upper goods, lower goods and rivets),

$$\textbf{MTyp-tk: } tk \rightarrow \{1,\ 2,\ 3,\ 4,\ 5,\ 6\}\,;\, tk \mapsto \begin{cases} 1,\ tk \\ \in trained\ agent \\ 2,\ tk \\ \in random\ agent \\ 3,\ tk \\ \in upper\ goods \\ 4,\ tk \\ \in lower\ goods \\ 5,\ tk \\ \in rivets \\ 6,\ tk \\ \in Tr-Ready-type \end{cases} \tag{1}$$

- $GTyp-dis$: the $Typ-dis$ function allowing to distinguish between the three distribution types assigned to the transitions.

$$GTyp-dist : (T_s) \rightarrow \{e,\ n,\ ln\}$$
$$k \mapsto GTyp-dist(k) = \begin{cases} e, & (Exponential) \\ n, & (Normal) \\ ln, & (Log-Normal) \end{cases} \tag{2}$$

- $TFringT$: a function, associated with the model's non-immediate transitions, that defines the firing time following the three laws of stochastic transition probability

(Normal, Log-Normal, or Exponential).

$$
\begin{aligned}
&TFringT : T_s \times \{e,\ n,\ ln\} \to \mathrm{R}^+ \\
&typ = Typ - dist(ts) \\
&(ts, typ) \mapsto TFringT(ts) =
\end{aligned}
$$

$$
\begin{cases}
typ = e, \frac{\int_{-\infty}^{t} (\lambda \cdot e^{-\lambda x}) \ \cdot \ x \ \ dx}{\int_{-\infty}^{+\infty} (\lambda \cdot e^{-\lambda x}) \ \ dx}; \\
typ = n, \frac{\int_{-\infty}^{+\infty} ((1/_{(2\sqrt{2\pi})} \cdot e^{-1/2.x^2}) \cdot x) \ \ dx}{\int_{-\infty}^{+\infty} ((1/_{(2\sqrt{2\pi})} \cdot e^{-1/2.x^2}) \ \ dx}; \\
typ = ln, \frac{\int_{-\infty}^{+\infty} (1/_{x\sigma\sqrt{(2\pi)}} \cdot e^{-((\ln x - \mu)^2/_{2\sigma^2}) \cdot \ x)} \ \ dx}{\int_{-\infty}^{+\infty} (1/_{x\sigma\sqrt{(2\pi)}} \cdot e^{-((\ln x - \mu)^2/_{2\sigma^2})} \ \ dx}; \\
(\ x \in \mathrm{R}^+, \lambda \in \mathrm{R}^+,\ \sigma \in \mathrm{R}^+,\ \mu \in \mathrm{R}).
\end{cases}
\tag{3}
$$

- S: The non-finite set of states is assigned to a trained agent to take into account in its life cycle;
- AC: The non-finite set of actions that a learning agent can execute in its behavior.
- $Q - Learning$: is a function based on a Markov Decision Process. This function is executed by the trained agent, depending on the current state and the action to be performed. Formally, this function is presented as follows:

$$
\begin{aligned}
&GLearning :\ (A \times S \times AC \times T_s) \to (A \times S \times AC) \\
&tka = \ MTyp - tk(A_i) = 1 \\
&(s,\ s') \in (S \times S) \\
&(a,\ a') \in (AC \times AC) \\
&t \ \in T_s \\
&tka \ \in \ M(p) \ \in (P \cap \ ^I tr)
\end{aligned}
$$

$$
\begin{aligned}
Q(A_i,\ t,\ s,\ a) = \\
(1-\alpha) \cdot Q(A_i,\ t,\ s,\ a) + \ \alpha(r + \gamma \max Q(A_i,\ t,\ s',\ a')) \qquad (4) \\
\alpha = \frac{1}{(1 + nbfiring(t)} \qquad (5)
\end{aligned}
$$

- $GLearning$: The specified function $(GLearning)$ describes the forced learning mobile agent behavior by defining the validation to leave place and integrate into another place following the transition firing. In other words, it instantly describes the behavior of a mobile-trained agent in the perception, output, and integration phases. This is based on a stochastic transition $t \ \ (t \ \in \ Ts)$, a token $A_i \ \ (MTyp - tk \ (A_i) \ = 1)$, and another random agent type token such that

$(MTyp - tk(AR_j) = 2)$. We note the trained mobile agent learning function generalized by $GLearning(A_i,\ AR_j,\ t,\ p)$. Formally, we define the learning function $Glearning$ as follows:

$$
\begin{aligned}
&GLearning: \ (A \times A \times T_s \times P \times \mathrm{R}^+ \times \mathrm{R}^+) \rightarrow (\mathrm{R}^+ \times \mathrm{R}^+)\\
&tka = \ MTyp - tk(A_i) = 1\\
&tkr = \ MTyp - tk(AA_j) = 2\\
&p \ \in (P \cap \ {}^I t)\\
&t \ \in T_s\\
&(s, a) \in (S \times AC)\\
&tka \ \in \ M(p) \ \in (P \cap \ {}^I tr)\\
&T_{begin} = \min(TFiring(t)) \ (\forall \ t \ \in \ {}^I p)\\
&T_{end} = \min(A_i \ \cdot \ history(t,\ times))
\end{aligned}
$$

$$
\begin{aligned}
GLearning(A_i,\ AR_j,\ t,\ p,\ T_{begin},\ T_{end}) = &\qquad (6)\\
Q - Learning(A_i,\ t,\ < s,\ a >) &\qquad (7)
\end{aligned}
$$

The proposed model $(LFSPN)$ is designed to depict the execution of a production chain process. Its primary objectives are to specify the model, ensuring alignment with the learning objectives of an agent equipped with behavior that integrates mobility and learning. Additionally, the model aims to verify properties and predict delays associated with learning behavior. To illustrate the adopted process of specification, verification, and prediction, we apply the model to a production chain example. It's worth noting that we utilize the same application example as presented in [8].

4 Case study: Reinforcement Learning of Production Systems

In this section, we demonstrate the potential of our modeling approach utilizing the $LFSPN$ model through an industrial case study. Our methodology comprises specification, verification, and learning prediction. Within this case study, we leverage the $LFSPN$ to specify, verify, analyze, and simulate the production flow control process. It's important to highlight that our application of the model and its associated tools to a material flow process example aligns with the production chain section depicted in [8]. In this work, we employ a combination of specifications, analysis, and verification techniques involving timed transitions, places (with random dwell times), and lifecycle tokens to identify sources of delay. Our modeling approach for a production chain, based on $LFSPN$, encompasses concepts related to delays, temporal properties of transitions, and the circulation of tokens with randomly timed life cycles (supervision agents, materials, etc.). We applied our model on the same factory example, studied in [8], and which ensures the capture of different aspects of material flow systems, see Figure1. This manufacturing line is composed of an output and assembly

section responsible for assigning product modifications. The internal transport aspect is achieved and captured with the rotary table, while the entry and storage sections each contain a storage unit [8].

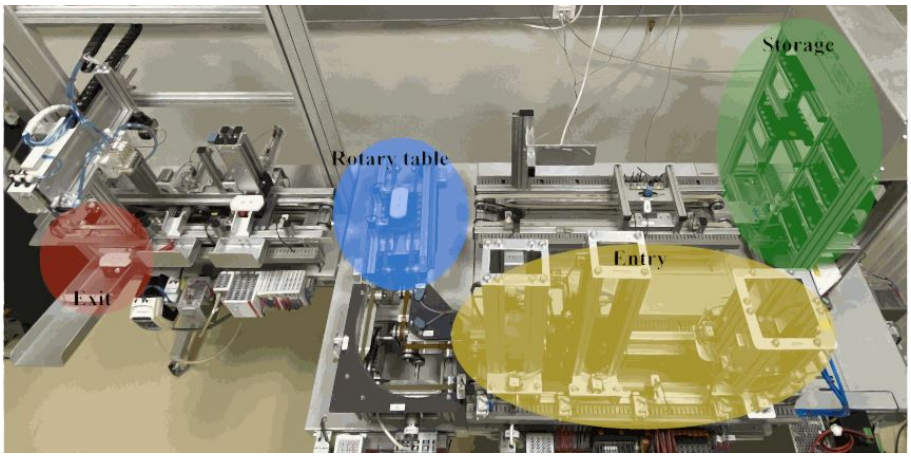

Fig. 1 Production chain [8]

The production chain is structured with an initial warehouse serving as the entry point for materials designated for product manufacturing (refer to Figure 1). Various types of components are distinguished upon entry, including transport carts, the lower half, and the upper half of the product. Notably, the bottom half of the product may fall into two distinct categories [8]. Following the assembly of product components, the items are conveyed to a rotary table. This table facilitates the movement of goods towards either the outlet or a storage unit, enabling the transportation of goods between the storage unit and the outlet. The factory exit is composed of a product depot and an assembly station. It is noteworthy that the final phase entails the addition of rivets to the product at the assembly station [8].

4.1 Places Specification

The proposed model is composed of 13 places with indices from 0 to 12. The table1 illustrates their role description in the model.

4.2 Transitions Specification

The representative model of the production chain is composed of 13 transitions endowed with indices numbered from 1 to 12. We distinguish between two categories according to the role (transport transition or action transition (assembly/installation)).

- Stochastic action transitions: this subset is composed by $\{t_1,\ t_{10}\}$;
- Stochastic transitions of goods transport: this subset is composed by $\{t_2,\ t_3,\ t_4..t_9\}$;

Palce	Role	${}^{I}p$	p^{O}
p_0	trained mobile agent park	$t_{12},\ t_{13}$	t_1
p_1	lower goods park	$t_{12},\ t_{13}$	t_1
p_2	upper goods park	$t_{12},\ t_{13}$	t_1
p_3	transport park	$t_{12},\ t_{13}$	t_1
p_4	goods after assembly operation	t_1	t_2
p_5	RT Input	t_2	$t_3,\ t_4$
p_6	RT Ready	$t_3,\ t_4,\ t_5$	$t_3,\ t_4,\ t_5$
p_7	RT Storage	t_3, t_5	$t_6,\ t_7$
p_8	storage park	t_6	t_13
p_9	RT Output	$t_4,\ t_6, t_8$	$t_5,\ t_9$
p_{10}	Assembly station	$t_9,\ t_{10}$	$t_8,\ t_{10},\ t_{11}$
p_{11}	Rivet park	$\emptyset$	t_{10}
p_{12}	Out put system	t_{11}	t_{12}

Table 1 Places specification of production system

- Immediate transitions: the model includes two immediate transitions to recover the trained agent role $\{t_{12},\ t_{13}\}$.

4.3 Tokens specification

In delineating the set of tokens, it becomes essential to differentiate between model agents (whether mobile-trained or random) and conventional tokens, the latter specifying the goods undergoing processing within the production chain. Then, according to this hypothesis, the tokens set is presented as follows:

- A trained agent responsible for carrying out the training by moving with the assets that flow through a transition firing sequence;
- A random agent assigned to a stochastic transition to detect firing time and predict delays or failure;
- A set of lower-goods-type tokens;
- A set of upper-goods-type tokens;
- A set of Tr-Ready-type tokens;
- A set of rivet-type tokens.

4.4 LFSPN Model Formal Checking

The formal verification using our LFSPN model revolves around assessing the learning and success aspects, encompassing both performance mode and performability (system failure detection). This verification is grounded in the examination of the token set (consumption and production) resulting from the firing of transition sequences, as depicted in the marking graph (refer to Figure 2). In essence, the properties of the stochastic petri net (SPN) that can be scrutinized include boundary conditions, liveness, blocking states, reachability, and other relevant criteria. According to the marking graph, the firing sequences, which ensure the trained agent completes learning in the finished product process (production or storage), are presented as follows:

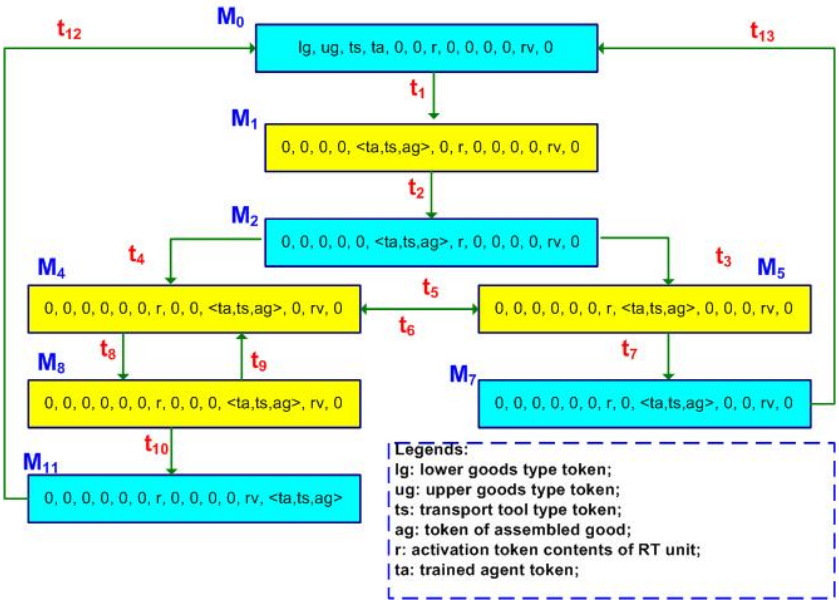

Fig. 2 Representative marking graph of the production chain modeled via $LFSPN$

- q_1=$\{t_1,\ t_2,\ t_4,\ t_9,\ t_{11},\ t_{12}\}$;
- q_2=$\{t_1,\ t_2,\ t_3,\ t_7,\ t_{13}\}$;
- q_3=$\{t_1,\ t_2,\ t_3,\ t_7,\ t_{13}\}$;
- q_4=$\{t_1,\ t_2,\ t_4,\ t_9,\ t_8,\ t_9,\ t_{11},\ t_{12}\}$;
- q_5=$\{t_1,\ t_2,\ t_3,\ t_9,\ t_7,\ t_6,\ t_5,\ t_7,\ t_{13}\}$;

To check the $LFSPN$' properties, we adopt an analogy with classical concepts of SPN verification.

- The $LFSPN$ model associated with the production chain process for all firing sequences transitions (q_1; q_2; q_3; q_4) is 6-bounded;
- The $LFSPN$ model is not pure;
- The $LFSPN$ model has no source transition.
- All the transitions are $quasi-live$ so the $LFSPN$ model is almost alive.
- Note that the $LFSPN$ model does not have an accessibility problem.
-
- We see that M_0, M_{12} and M_{13} are equivalent to what a firing circuit shows. This explains that learning is continuous.
- The $LFSPN$ model is $quasi-live$, not balanced and not sound.

5 Simulation and reinforcement learning

To show the applicability $LFSPN$ model as a multi-agent environment for an agent with reinforcement learning behavior, we propose a Q-Learning algorithm that has been implemented with Matlab (see algorithm1). We adopt the same hypothesis as

the authors in [8]. This hypothesis assumes that the agent is rewarded for correctly depositing a product and punished for depositing a product by mistake or causing a collision. Also, we encourage effective solutions, and each transition firing is slightly punished. This means that a solution using as few transitions as possible is preferred, such as the sequence already chosen (q_1). Finally, a timeout was defined at the end of which the scenario was reset.

- Let S: the states set resulting from the mapping between a marking graph and the state graph.
- Let AC: the actions set performed by a trained agent.
- Let SQ: the transition set sequences capable of ensuring the operations production success.
- Let $NBIter$: the number of learning iterations.

Algorithm 1 Learning Algorithm

$LetaasTrainedagent$
$Let\ i\ AsInteger$
$Let\ s\ As\ state$
$Let\ ac\ ASAction$
$Let\ sq\ \in\ SQ$
$Lett\ \in\ T$
$Letp\ \in\ I^t$
$i\ \leftarrow\ 0$ NBIter=70000
While (i < NBIter)
If $(\varphi\ =\ 0)$
$r\ \leftarrow\ 0$
break;
ELse
$tb \leftarrow \min(Tfiring(t))$
$te \leftarrow\ \min(a\ .\ \ thistory(t, time))$
$a.r\ \leftarrow\ Glearning(a,\ \varphi,\ t,\ tb,\ te)$
$t \leftarrow\ Next$
$i\ \leftarrow\ i+1$
End IF
EndWhile

5.1 Simulation

We note that the $LFSPN$ model simulation is subdivided into two axes:

- A classic simulation axis calculates the firing time of each transition, which makes up a firing sequence. This is achieved via the functions of the model, which are based on the laws of probability. We note that this simulation (see figure3 and figure4) is

similar to that presented in [27, 28]. The objective of this simulation is to evaluate the execution time and the delay.

- Via this interface, we can configure the transitions and determine the most suitable probability law to minimize the sequence firing time. A learning-based simulation axis which adopts the learning agent implementation (Q-learning based on the algorithm1). Figure4 illustrate this simulation interface is presented. The second simulation evaluation is intended to study the modeled chain performance.

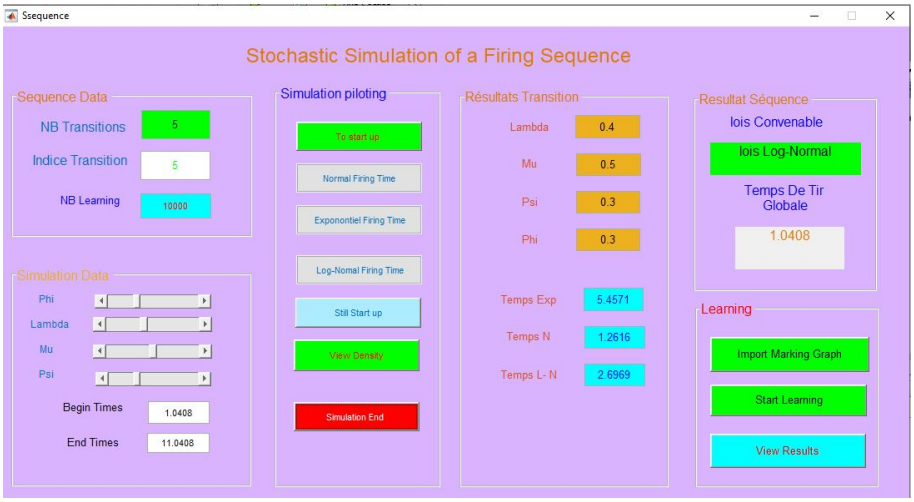

Fig. 3 Simulation interface

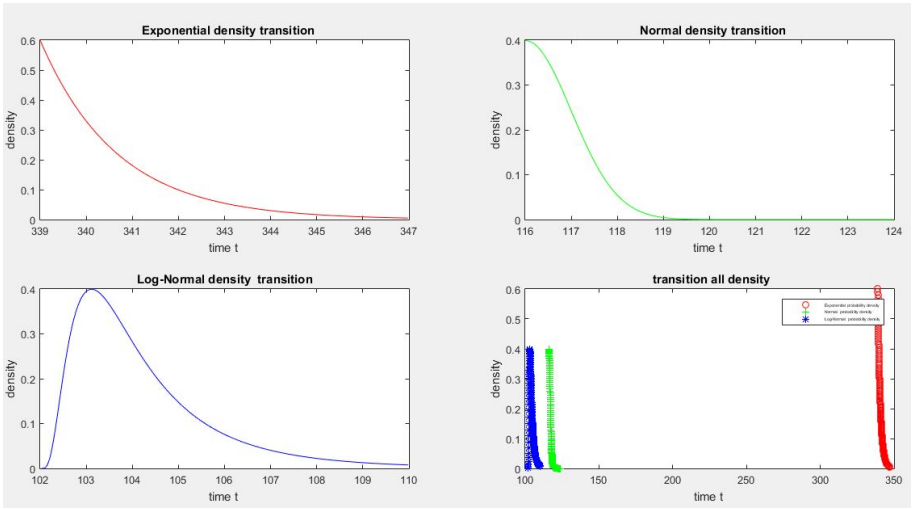

Fig. 4 Density Transition

To simplify the learning simulation (Q-learning) of the agent, we assume that all the transitions in the $LFSPN$ model are stochastically homogeneous. According to the simulation, we found that the values ($\mu = 0.4$, $\psi = 0.6$, $\lambda = 0.6$) are the most suitable to achieve significant results.
We see that the learning of our developed agent depends on the stochastic-based abstraction aspect provided by the LFSPN model. To ensure this learning simulation phase, we trained the algorithm 70000 times for each type of probability density (exp: $\lambda = 0.6$, Log-Normal: ($\mu = 0.4$, $\psi = 0.6$, Normal (0, 1)). Figure5 illustrates the training results of the agent (Q-learning) and a classic probability agent (called purely random). We note that the latter is endowed by a randomly aspect when choosing an action, so it never improves its behavioral aspect. Also, figure5 shows the floating average of the token number correctly deposited over episodes according to the three distributions (exponential, normal, and log normal) adopted by $LFSPN$. We notice:

- Normal-Trained agent: the curve colored in red represents the floating average of the token number correctly deposited in the case of Normal;
- Exponential-Trained agent: the curve colored in blue represents the floating average of the token number correctly deposited in the case of exponentials.
- Log-Normal-Trained agent: the curve colored in green represents the floating average of the token number correctly deposited in the case of Log-normal.

According to figure5 we can notice that:

△ The optimal number of correctly deposited products is around 3.0.
△ The Log-Normal-trained agent has this average after 50 000 training sessions;
△ The two trained agents (normal and exponential) reach an average of around 2 products correctly deposited.

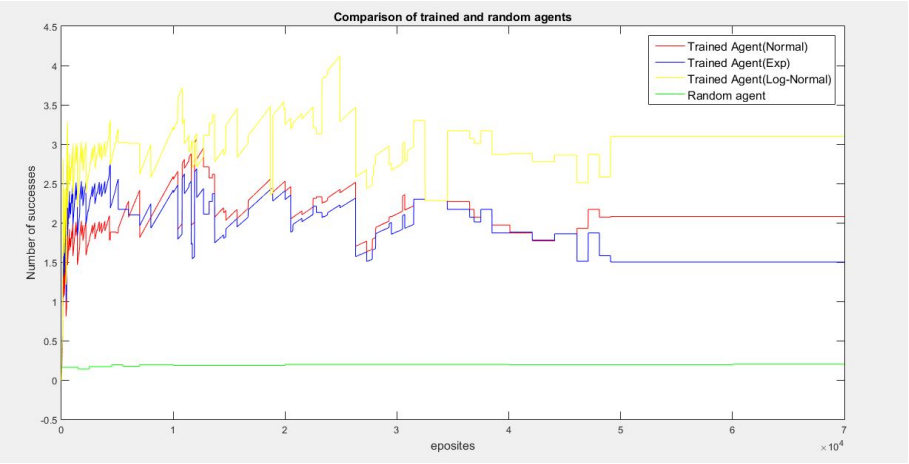

Fig. 5 Results of all trained Q-learning agents granted in $LFSPN$

5.2 Comparison and discussion results

By comparing ours with that presented in [8], we can deduce:

△ The authors in [8] (see figure6) presented a comparison between a simulation of a trained agent based on a timed colored petri net model and a random agent.
△ The latter is not capable of improving its behavior because of its random appearance. According to figure6 the average number of successes is 1.5 products. This value is reached after 50 000 learning trials.
△ Our two trained agents (normal and exponential) reach around 2 and 1.5 properly deposited on average on average.
△ Our learning behavior of the trained agent (exponential) is similar to the trained agent developed in [8].
△ Our simulation based on our proposed $LFSPN$ model provided learning stability after 50000 trials. Our three trained agents (exponential, normal, and log-normal) reach their performance values (number of successes), which are clearly higher compared to those provided in [8].
△ Our two trained agents learning performance is explained by the characteristics of the (normal and log-normal) laws, which reflect the memory of the stochastic transition firing and the system history represented by $LFSPN$.
△ This is explained by the addition of two immediate transitions (t_{12} and t_{13}).

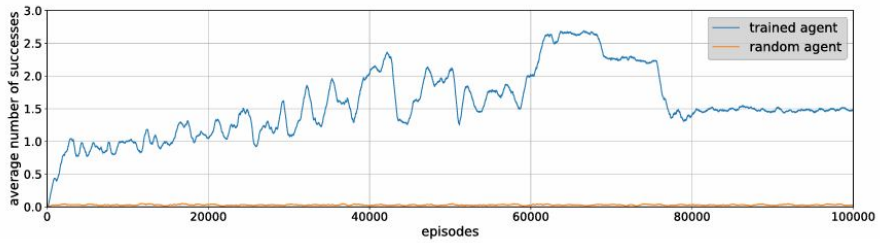

Fig. 6 Results of trained agent in [8]

6 Conclusion

In this study, we propose a model named $LFSPN$ that serves as a comprehensive extension integrating stochastic Petri nets with the multi-agent system paradigm. Our model introduces a trained agent responsible for the learning process. The primary objective of the model is to define, verify, and assess the material flow within an automated production line. We demonstrated the applicability of our model through a process encompassing the specification, verification, and simulation of a production chain with material flow. The distinctiveness of our model lies in the characteristics of the learning agent, which is summarized by two key behaviors: mobility and learning. The learning algorithm, integrated as a tool within our model, exhibits flexibility by incorporating a stochastic aspect based on the transitions. We demonstrate the

proposed model effectiveness by presenting a comparison between our learning simulation results and those presented by the authors in [8]. We note that we applied $LFSPN$ model to the same production chain presented in [8]. $LFSPN$ model advantage lies in a representation framework with two abstraction levels: a graphic level via the classic components of a SPN and another learning and migration level of which characterizes a mobile learning agent.

7 Declarations

- ⋆ **Conflict of interest** : The autors declares that they have not confils of interestes.
- ⋆ **Ethical approval** : All procedures performed in studies involving human participants met the ethical standard of institutional.
- ⋆ **Funding** : No funding was received to assist with the preparation of this manuscript.
- ⋆ All authors contributed to the study conception and design.

8 References

References

[1] machine-learning, 2023 https://datascientest.com/machine-learning-tout-savoir[accessed: 2023-10-18]

[2] artificial-intelligence/machine-learning, 2023 https://www.oracle.com/fr/artificial-intelligence/machine-learning/what-is-machine-learning/[accessed: 2023-09-11]

[3] Skorikov, M., Momen, S., (2020) Machine learning approach to predicting the acceptance of academic papers. In IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT), pp 113-117, 2020 10.1109/IAICT50021.2020.9172011

[4] Kawamitsu, I., Nakamura, M. (2020). Colored Petri Net Modeling for Prediction Processes in Machine Learning. In: Fujita, H., Fournier-Viger, P., Ali, M., Sasaki, J. (eds) Trends in Artificial Intelligence Theory and Applications. Artificial Intelligence Practices. IEA/AIE 2020. Lecture Notes in Computer Science(), vol 12144. Springer, Cham. https://doi.org/10.1007/978-3-030-55789-8_57.

[5] Skorikov, M., Momen, S., (2020). Machine learning approach to predicting the acceptance of academic papers, 2020 pp. 113-117, 10.1109/IAICT50021.2020.9172011

[6] Gottlich, S., Hoher, S., Schindler, P., Schleper, V., Verl, A.,(2013). Modeling, simulation and validation of material flow on conveyor belts. In Applied Mathematical Modelling, pp. 3295-3313, 2013.

[7] Glatt, M., Kull, D. , Ravani B., Aurich, J. C., (2019). Validation of a physics engine for the simu- lation of material flows in cyber-physical production systems, In CIRP Conference on Manufacturing Systems, Ljubljana, Slovenia, 2019.

[8] Riedmann, S., Harb, J., Hoher, S.,(2022). Timed Coloured Petri Net Simulation Model for Reinforcement Learning in the Context of Production Systems, In Production at the Leading Edge of Technology, pp. 457–465, 2022, Springer International Publishing, isbn 978-3-030-78424-9.

[9] Benoit, R., Dominique, B., Rene, K.,(2018). Fondements des analyses de flux de matieres et d energie et typologies d applications pour la gouvernance des territoires et des organisations, journals.openedition, https://doi.org/10.4000/vertigo.20822[accessed: 2023-08-11]

[10] V., G., P., ADB,(1990). VDI2860 Montageund Handhabungstechnik, Dusseldorf: Beuth Verlag GmbH, 1990.

[11] Jensen, K.,(1997). A brief introduction to coloured Petri Nets. In, Springer, 1997.

[12] Li, Z., Wang, S., Zhao, T., Liu, B.,(2016). A hazard analysis via an improved timed colored petri net with time-space coupling safety constraint,Tools and Algorithms for the Construction and Analysis of Systems, Berlin, Heidelberg. In Chinese Jounal of Aeronautics, pp. 1027-1041, 2016.https://doi.org/10.4000/vertigo.20822[accessed: 2023-08-11]

[13] Lefebvre, D., (2017).Probability of faults for partially observed Timed PNs with temporal constraints. In Networking, Sensing and Control (ICNSC), 2017 IEEE 14th International Conference. IEEE. Calabria, Southern Italy, 2017 doi: 10.1155/2017/2821078.

[14] Rachidi, S., Leclercq, E., Pigne, Y., Lefebvre, D., (2019). PN modeling of discrete event systems with temporal constraints. In 21th International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, Romania, 2017, pp. 103–108, doi: 10.1155/2017/2821078

[15] Peterson, J.L., (1982). Petri net theory and the modeling of systems.In The Computer Journal, 25(1), pp 129–135, 1982 doi: doi.org/10.1093/comjnl/25.1.129

[16] Merlin, P. M., (1974). A study of recoverability of communication protocols, Ph.D. Thesis, University of California, 1974.

[17] Holliday, M. A., Vernon, M.K., (1987). A generalized timed Petri net model for performance analysis. In IEEE Transactions in Software Engineering, 13. IEEE Computer Society, 1987.

[18] Lefebvre, D., (2017). Detection of Temporal Anomalies for Partially Observed Timed PNs. In Mathematical Problems in Engineering, 7, 2017. doi:10.1155/

2017/2821078.

[19] Walter, B.,(1983). Timed Petri-nets for modelling and analyzing protocols with real-time characteristics. In Third IFIP workshop on protocols specification, testing and verification. North-Holland, 1983.

[20] Ghomri, L., Alla, H.,(2008). Modelling and analysis of hybrid dynamic systems using hybrid Petri nets.In Vedran Kordic , Publisher: InTech, pp 113–130, 2008. doi:978-3-902613-12-7.

[21] Ghomri, L., Alla, H., (2011). Continuous flow Systems and Control Methodology Using Hybrid Petri nets. In IEEE Conference on Automation Science and Engineering, Trieste, Italy, 2011.

[22] Ben Mesmia, W., Marzougui, B., Barkaoui, K., (2016). Petri Nets for Mobile Agent: Theory and Application.In Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016, Springer International Publishing, doi: 10.1007/978-3-319-56991-817.

[23] Idrissi, H., (2016) Contributions to the security of mobile agent systems. Cryptography and Security [cs.CR]. University of Rochelle; University of Mohammed V (Rabat), 2016 https://tel.archives-ouvertes.fr/tel-01661378 [accessed: 2022-04-27]

[24] Zhiyu, Q.,(2002) Mobile Agents. DEA from the university of Paris Sud, Paris, 2002.

[25] Marzougui, B., Hassine, K., Barkaoui, K., (2001). A New Formalism for Modeling a Multi Agent Systems: Agent Petri Nets. In Journal of Software Engineering and Applications (JSEA),3(12), pp 1118-1124, 2001

[26] Ferber, J.(1995). The multi-agent systems to collective intelligence.In Inter Editions, Paris, pp. 33–47, 1995

[27] Ben Mesmia, W., Escheikh, M., Barkaoui, K.,(2020). DevOps Workflow verification and duration prediction using non-markovian stochastic petri nets. J Softw Evol Proc, 2020.https://doi.org/10.1020/smr.2329

[28] Mesmia, W., Barkaoui, K., Escheikh, M., (2023) FMS-Workflow Modeling Based on P-Timed Stochastic Petri Net. Journal of Software Engineering and Applications, 16, pp 443-482. doi: 10.4236/jsea.2023.169022.

[29] Dimeas, A.L. and Hatziargyriou, N.D.,2007, IEEE Power Engineering Society General Meeting, Agent based Control for Microgrids, 1-5, doi=10.1109/PES.2007.386064.

[30] Telek, M. and Horvath, A., Proceedings. IEEE International Computer Performance and Dependability Symposium. IPDS'98 (Cat. No.98TB100248),

Supplementary variable approach applied to the transient analysis of age-MRSPNs,1998,44-51, doi10.1109/IPDS.1998.707708