DOI: 10.1111/mice.12427

ORIGINAL ARTICLE

WILEY COMPUTER-AIDED CIVIL AND INFRASTRUCTURE ENGINEERING

Plausible Petri nets as self-adaptive expert systems: A tool for infrastructure asset monitoring

Manuel Chiachío[1,2] | Juan Chiachío[1] | Darren Prescott[1] | John Andrews[1]

[1]Resilience Engineering Research Group, Faculty of Engineering, University of Nottingham, Nottingham, UK

[2]Department of Structural Mechanics and Hydraulic Engineering, University of Granada, Granada, Spain

Correspondence
Manuel Chiachío, Resilience Engineering Research Group, Faculty of Engineering, University of Nottingham, Nottingham NG7 2RD, UK.
Email: Manuel.Chiachio-Ruano1@nottingham.ac.uk

Funding information
Lloyd's Register Foundation, Grant/Award Number: RB4539; Engineering and Physical Sciences Research Council, Grant/Award Number: EP/M023028/1

Abstract

This article provides a computational framework to model self-adaptive expert systems using the Petri net (PN) formalism. Self-adaptive expert systems are understood here as expert systems with the ability to autonomously learn from external inputs, like monitoring data. To this end, the Bayesian learning principles are investigated and also combined with the Plausible PNs (PPNs) methodology. PPNs are a variant within the PN paradigm, which are efficient to jointly consider the dynamics of discrete events, like maintenance actions, together with multiple sources of uncertain information about a state variable. The manuscript shows the mathematical conditions and computational procedure where the Bayesian updating becomes a particular case of a more general basic operation within the PPN execution semantics, which enables the uncertain knowledge being updated from monitoring data. The approach is general, but here it is demonstrated in a novel computational model acting as expert system for railway track inspection management taken as a case study using published data from a laboratory simulation of train loading on ballast. The results reveal self-adaptability and uncertainty management as key enabling aspects to optimize inspection actions in railway track, only being adaptively and autonomously triggered based on the actual learnt state of track and other contextual issues, like resource availability, as opposed to scheduled periodic maintenance activities.

1 | INTRODUCTION

Self-adaptability is an important intrinsic property that is displayed by many natural systems to deal with the challenges presented by changing environments. In engineering, the need to incorporate self-adaptation has been acknowledged as important in allowing engineered systems to modify their behavior in response to changing conditions with little or no human input, hence increasing efficiency, safety, and availability while minimizing the possibility of human errors (Krupitzer, Roth, VanSyckel, Schiele, & Becker, 2015). Intelligent systems like *expert systems* have the ability to emulate the human capacity to make decisions within a specific application domain using execution rules and knowledge (Adeli & Balasubramanyam, 1988; Kastner & Hong, 1984; Wagner, 2017), which does not necessarily include self-adaptation. Through self-adaptation, this knowledge is updated to dynamically accommodate environmental and contextual changes, therefore increasing the system efficiency and making it more resilient to the new conditions, which is one of the key aspects of intelligent systems.

In the literature, numerous knowledge-based models are available to represent expert systems (Adeli & Balasubramanyam, 1988; Paek & Adeli, 1990). Of the knowledge-based system approaches, Petri nets (PNs) (Petri, 1962) are typically regarded as a powerful modeling tool for expert systems,

especially when system-level operational nonlinearities (e.g., resource availability, concurrency and synchronization of components, etc.) need to be considered in the analysis (M. Chiachío, J. Chiachío, Sankararaman, & Andrews, 2017). PNs provide a graphical and mathematical language with well-established execution semantics, which can be combined with other computational techniques such as object-oriented programming, fuzzy sets, neural networks, etc., all of which greatly increase their suitability for modeling knowledge-based engineering problems (Li & Lara-Rosano, 2000). The basic concepts relative to the theory of PNs are summarized in Murata (1989). One of the main challenges in incorporating self-adaptation in expert systems is the handling of uncertain information during runtime, and the ability to update such information when new evidence becomes available (Bencomo & Belaggoun, 2013). However, the existing PN formalisms do not provide direct means to efficiently consider uncertain information (M. Chiachío, J. Chiachío, Prescott, & Andrews, 2016; M. Chiachío, J. Chiachío, Prescott, & Andrews, 2018). In the literature, a number of PN variants have been introduced to enhance the original PN approach with improved rules of inference and knowledge learning. Of the existing variants, fuzzy PNs (FPNs) (Looney, 1988) have received much attention due to their efficiency for reasoning in expert systems using fuzzy production rules based on imprecise and vague information (Chiang, Liu, & Lee, 2000; Flintsch & Chen, 2004; Lee, Liu, & Chiang, 1999; Zhou & Zain, 2016). In the past, some authors dealt with self-adaptation of FPNs by training an FPN model using a reference one taken as benchmark (Li & Lara-Rosano, 2000; Zhang, Wang, & Yuan, 2009). More recently, other PN variants have been introduced to deal with some sort of self-adaptivity, see for example (Hsieh & Lin, 2014; Vidal, Lama, & Bugarín, 2012; Vidal, Lama, Díaz-Hermida, & Bugarín, 2013), although most of them are domain or purpose-specific (Serral, De Smedt, Snoeck, & Vanthienen, 2015). However, to the best of the authors' knowledge, none of them are well suited to (a) embedding plausible information within their formulation, and (b) automatically reacting to that information by adaptive learning, while dealing with the hybrid nature of real-world dynamical systems, consisting of a combination of discrete and continuous processes whose evolution may be uncertain.

In this article, a new methodology is proposed to enable self-adaptation in expert systems using the Plausible Petri nets (PPNs). PPNs are a new class of models within the PN paradigm, originally developed by the authors in M. Chiachío et al. (2016, 2018), whereby discrete events (e.g., go/no-go decisions) can be jointly modeled together with continuous processes whose evolution may be uncertain (e.g., deterioration process). In PPNs, the uncertainty is modeled using *states of information* (Rus, Chiachío, & Chiachío, 2016), which provide a mapping between the possible numerical values of a state variable and their relative plausibility, hence giving greater versatility for representing uncertain knowledge in a more principled approach. As a key contribution, this article reveals how self-adaptation can be achieved naturally as a by-product of the evaluation of a PPN, because an inherent learning mechanism is implemented within the PPN execution semantics. More specifically, an instance of *Bayesian model updating* is seen to appear as a particular case of the *conjunction of states of information* (Tarantola & Valette, 1982), which is a basic operation within the PPN execution rules (Chiachío et al., 2016, 2018). Consequently, the resulting approach has the advantages of (a) being able to deal with uncertain information in expert systems combining discrete events and continuous state variables, and (b) enabling self-adaptation by Bayesian learning from external input data.

The proposed methodology is general, and as such, it can be applied to different applications dealing with self-adaptation and uncertain information in infrastructure asset monitoring. However, in this article, it is illustrated using an engineering case study of condition-based maintenance for a railway track. The interest of this engineering application resides in the need for artificial intelligence (AI) methodologies that allow automated and adaptive decisions about maintenance activities and inspection actions in railway networks based on monitoring data (Lajnef, Rhimi, Chatti, Mhamdi, & Faridazar, 2011; Wang, Liu, & Ni, 2018; Weston, Roberts, Yeo, & Stewart, 2015). To illustrate the efficiency of the proposed methodology in this application, a PPN-based computational model is developed to act as an expert system for railway track monitoring and inspection, incorporating information about a state variable along with a number of operational rules that provides the basis for triggering a number of control operations and inspection activities. The overall system is shown to be adaptable by sequentially updating the state variable as (noisy) monitoring data become available.

The remainder of the article is organized as follows. Section 2 briefly overviews basic concepts about Bayesian model updating and PNs before introducing the PPN methodology in Section 2.3. In Section 2.3.3, an algorithmic description of PPNs is provided. The mathematical basis and computational aspects of Bayesian learning of PPNs are provided in Section 3. Section 4 illustrates and discusses our approach in application to a case-base self-adaptive expert system for railway track inspection. Finally, Section 5 gives concluding remarks.

2 | THEORETICAL BACKGROUND

2.1 | Bayesian model updating

Let us consider a probability model described by the state vector $\mathbf{x}$ taking values in a space denoted by $\mathcal{X} \subset \mathbb{R}^d$, and $p(\mathbf{x})$ a *prior* probability density function (PDF) of $\mathbf{x}$ over $\mathcal{X}$, a Lebesgue integrable function that can be normalized such

that $\int_{\mathcal{X}} p(\mathbf{x})d\mathbf{x} = 1$. The focus of Bayesian model updating is to update the prior information about $\mathbf{x} \in \mathcal{X}$, based on the information given by the data $\mathbf{y} \in \mathcal{D} \subset \mathbb{R}^{\ell}$, where $\mathcal{D}$ is the *observation space* within the region $\mathbb{R}^{\ell}$. Following the Bayesian formulation, the solution is not a single value of $\mathbf{x}$; on the contrary, Bayes' theorem takes the initial quantification of the plausibility of $\mathbf{x}$, which is expressed by the *prior* PDF $p(\mathbf{x})$, and updates this plausibility using the information in the data set $\mathcal{D}$ to obtain the *posterior* PDF of the state variable $\mathbf{x}$, as:

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{\int_{\mathcal{X}} p(\mathbf{y}|\mathbf{x})p(\mathbf{x})d\mathbf{x}} \propto p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) \tag{1}$$

where $p(\mathbf{y}|\mathbf{x})$ is the *likelihood function*, which provides us with a measure of how well the model specified by $\mathbf{x}$ predicts the actual data $\mathbf{y}$ (Beck, 2010). The interested reader is referred to Beck (2010) and Rus et al. (2016) for further information about Bayesian model updating. In this work, we adopt a subjective interpretation of probability as a multi-valued logic (Cox, 1946; Jaynes, 1983) whereby a PDF over the uncertain variable $\mathbf{x}$ (e.g., $p(\mathbf{x}|\mathbf{y})$) represents a measure of the *relative plausibility* of the values of $\mathbf{x} \in \mathcal{X}$ conditional on the available information (e.g., $\mathbf{y} \in \mathcal{D}$). This interpretation of probability is not well known in engineering where there is a widespread belief that probability only applies to aleatory uncertainty (inherent randomness in nature), and not to epistemic uncertainty (lack of knowledge).

2.2 | Petri nets

A PN is a mathematical and graphical modeling tool first introduced by Carl Petri in 1962 (Petri, 1962) for analyzing the dynamic behavior of sequential asynchronous automatons. Since then, they have expanded to many areas of science and engineering for the modeling of complex distributed dynamical systems. The reader is referred to Murata (1989) for a comprehensive review and tutorial on PNs, but for the sake of clarity and readability, the main concepts are reproduced here under a unified notation.

A PN is a bipartite directed graph (digraph) consisting of two types of nodes, *places* (e.g., states, represented by circles) and *transitions* (e.g., actions, represented by bars or boxes), connected by *arcs* either from places to transitions or vice versa. See Figure 1 for an illustration of a simple PN consisting of three places (p_1, p_2, p_3), and one transition (t_1). The places contain *tokens* that travel through the net depending on the firing of the transitions. The presence of tokens in the places of the PN is interpreted as holding the truth of the condition or information about the states associated with those places, and defines the *marking* of the net. A transition t can fire only if all places leading to that transition have at least one token. Those places define the *preset* of transition t, denoted by ${}^{\bullet}t$. After the transition fires, one token is added to each of its output places, which define the *post set* of the transition, referred to as $t^{\bullet}$.

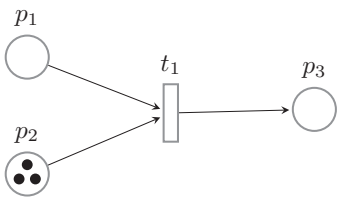


FIGURE 1 Example of a Petri net composed of three places and one transition. Three tokens are represented in p_2

From a mathematical perspective, a PN is defined as a tuple $\mathfrak{N} = \langle \mathbf{P}, \mathbf{T}, \mathbf{F}, \mathbf{W}, \mathbf{M}_0 \rangle$, where:

1. $\mathbf{P} \in \mathbb{N}^{n_p}$ is an n_p-dimensional set of places.
2. $\mathbf{T} \in \mathbb{N}^{n_t}$ is an n_t-dimensional set of transitions.
3. $\mathbf{F} \subseteq (\mathbf{P} \times \mathbf{T}) \cup (\mathbf{T} \times \mathbf{P})$ represents a set of directed arcs connecting places to transitions and vice versa.
4. $\mathbf{W} : \mathbf{F} \to \mathbb{N}_{>0}$ is a weight function, which assigns a value (1 by default) to each arc within $\mathbf{F}$.
5. $\mathbf{M}_0 : \mathbf{P} \to \mathbb{N}$ is a vector containing the initial distribution of tokens over the set of places (initial marking).

The state of the overall net is represented by the marking $\mathbf{M} \in \mathbb{N}^{n_p}$, which, at a certain state $k \in \mathbb{N}$, evolves dynamically according to the following state equation (Murata, 1989):

$$\mathbf{M}_{k+1} = \mathbf{M}_k + \mathbf{A}^T \mathbf{u}_k \tag{2}$$

where $\mathbf{A}$ is an $n_t \times n_p$ matrix typically referred to as the *incidence matrix*, which can be obtained as the result of subtracting the *backward incidence matrix* ($\mathbf{A}^-$) from the *forward incidence matrix* ($\mathbf{A}^+$), i.e., $\mathbf{A} = \mathbf{A}^+ - \mathbf{A}^-$, where $\mathbf{A}^+ = [a_{ij}^+]$, $\mathbf{A}^- = [a_{ij}^-]$, $i = 1, \dots, n_t$, $j = 1, \dots, n_p$. The element a_{ij}^+ is the weight of the arc from transition t_i to output place p_j, whereas a_{ij}^- is the weight of the arc to transition t_i from input place p_j. The term $\mathbf{u}_k = (u_{1,k}, u_{2,k}, \dots, u_{n_t,k})^T$ denotes the *firing vector*, a vector of binary values whose ith component takes 1 if transition t_i is fired, and 0 otherwise. In PNs, any transition t_i needs to be enabled as a condition to be fired, which occurs when each input place of t_i is marked with at least a_{ij}^- tokens. Mathematically:

$$M(j) \geqslant a_{ij}^- \qquad \forall p_j \in {}^{\bullet}t_i \tag{3}$$

where $M(j) \in \mathbb{N}$ is the marking for place p_j. Note that by means of PNs and their marking, the behavior of engineering systems can be described in terms of discrete system states and their changes over time. Also, it is worth mentioning that in practical applications of PNs, transitions are typically assigned with time delays that are useful for performance evaluation and scheduling problems of dynamical systems (Murata, 1989).

2.3 | Plausible Petri nets

PPNs (M. Chiachío et al., 2018, 2016) are a hybrid variant of PNs where the sets of nodes $\{\mathbf{P}, \mathbf{T}\}$ are partitioned into two disjoint subsets, namely, *numerical* and *symbolic*, which are denoted using superscripts $(\mathcal{N})$ and $(\mathcal{S})$, respectively. The symbolic subnet accounts for the discrete behavior of the system using regular tokens, as in classical PNs. In the *numerical subnet*, tokens are *states of information* about a state variable $\mathbf{x}_k \in \mathcal{X}$ (Rus et al., 2016; Tarantola & Valette, 1982), which accounts for the numerical behavior of the system. In practical terms, these states of information can be understood as PDFs about $\mathbf{x}_k$ (except for a normalizing constant; Mosegaard & Tarantola, 2002), which are referred to as $f^p(\mathbf{x}_k)$ and $f^t(\mathbf{x}_k)$ for numerical places and transitions, respectively. Hence, the marking $\mathbf{M}_k = (\mathbf{M}_k^{(\mathcal{N})}, \mathbf{M}_k^{(\mathcal{S})})$ consists of both types of information given by $\mathbf{M}_k^{(\mathcal{N})}$ for the numerical places, and $\mathbf{M}_k^{(\mathcal{S})}$ for the case of symbolic places, where $\mathbf{M}_k^{(\mathcal{N})}$ and $\mathbf{M}_k^{(\mathcal{S})}$ are column vectors of normalized PDFs and integer values, respectively.

From a mathematical perspective, a PPN can be described as a 9-tuple $\mathfrak{M} = \langle \mathbf{P}, \mathbf{T}, \mathbf{F}, \mathbf{W}, \mathbf{D}, \mathcal{X}, \mathcal{G}, \mathcal{H}, \mathbf{M}_0 \rangle$, where:

1. The set $\mathbf{P}$ is partitioned into $\mathbf{P}^{(\mathcal{N})} \in \mathbb{N}^{n_p}$ for numerical places, and $\mathbf{P}^{(\mathcal{S})} \in \mathbb{N}^{n'_p}$ for symbolic places, such that $\mathbf{P}^{(\mathcal{N})} \cup \mathbf{P}^{(\mathcal{S})} = \mathbf{P}$, and $\mathbf{P}^{(\mathcal{N})} \cap \mathbf{P}^{(\mathcal{S})} = \emptyset$. Superscripts n_p and n'_p represent the number of numerical and symbolic places, respectively.
2. Similarly, the set of transitions $\mathbf{T}$ is partitioned into subsets $\mathbf{T}^{(\mathcal{N})} \in \mathbb{N}^{n_t}$ and $\mathbf{T}^{(\mathcal{S})} \in \mathbb{N}^{n'_t}$ corresponding to numerical and symbolic transitions, respectively, where $\mathbf{T}^{(\mathcal{N})} \cup \mathbf{T}^{(\mathcal{S})} = \mathbf{T}$, $\mathbf{T}^{(\mathcal{N})} \cap \mathbf{T}^{(\mathcal{S})} \neq \emptyset$. Those transitions that belong to $\mathbf{T}^{(\mathcal{N})} \cap \mathbf{T}^{(\mathcal{S})}$ are referred to as *mixed transitions*.
3. The set of arcs $\mathbf{F}$ indicates the connections between transitions and places from the numerical and symbolic subnets, i.e., $\mathbf{F} \subseteq (\mathbf{P} \times \mathbf{T}) \cup (\mathbf{T} \times \mathbf{P})$, where $\mathbf{F} \subset \mathbb{N}^{n_p + n'_p} \times \mathbb{N}^{n_t + n'_t}$.
4. $\mathbf{W}$ is a set of nonnegative weights applied to each arc within $\mathbf{F}$ (1 by default). The set is partitioned into two subsets: $\mathbf{W}^{(\mathcal{N})}$ and $\mathbf{W}^{(\mathcal{S})}$, each one corresponding to the numerical and symbolic places, respectively, such that $\mathbf{W}^{(\mathcal{N})} \cup \mathbf{W}^{(\mathcal{S})} = \mathbf{W}$, $\mathbf{W}^{(\mathcal{N})} \cap \mathbf{W}^{(\mathcal{S})} = \emptyset$. Values from $\mathbf{W}^{(\mathcal{N})}$ are real numbers.
5. $\mathbf{D}$ is a set of switching delays for the symbolic (0 by default).
6. $\mathcal{X} \subset \mathbb{R}^d$ is the state space of a stochastic variable $\{\mathbf{x}_k\}_{k \in \mathbb{N}}$, which is representative of the numerical state of the net.
7. $\mathcal{G}$ is a set of density functions associated with the numerical places and transitions.
8. $\mathcal{H}$ is a set of equations representing the dynamics of the state variable $\mathbf{x}_k \in \mathcal{X}$.

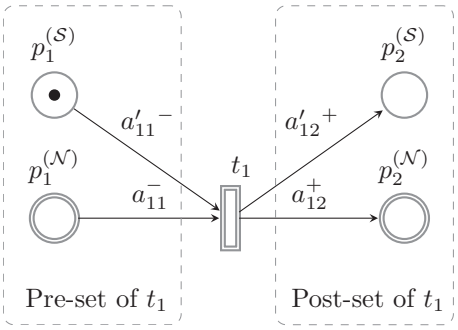


FIGURE 2 Illustration of a sample PPN with two numerical places ($p_1^{(\mathcal{N})}$, $p_2^{(\mathcal{N})}$), two symbolic places ($p_1^{(\mathcal{S})}$, $p_2^{(\mathcal{S})}$), and one transition (t_1). Note that numerical nodes are drawn using double lines

9. $\mathbf{M}_0$ is the initial marking of the net, which is given by the pair of vectors $\mathbf{M}_0^{(\mathcal{N})}$ and $\mathbf{M}_0^{(\mathcal{S})}$ for numerical and symbolic places, respectively.

In PPNs, the arc weights for the numerical subnet are denoted by $a_{ij}^+, a_{ij}^- \in \mathbf{W}^{(\mathcal{N})} \subset \mathbb{R}^+$. The *incidence matrix* for the numerical subnet is defined from $\mathbf{W}^{(\mathcal{N})}$ as $\mathbf{A}^{(\mathcal{N})} = [a_{ij}]$, $i = 1, \ldots, n_t$, $j = 1, \ldots, n_p$, where the (i, j)th element is obtained as $a_{ij} = a_{ij}^+ - a_{ij}^-$. Correspondingly, $a'^{+}_{ij}, a'^{-}_{ij} \in \mathbf{W}^{(\mathcal{S})} \subset \mathbb{N}$ denote the arc weights for the symbolic subnet, where the incidence matrix $\mathbf{A}^{(\mathcal{S})} = [a'_{ij}]$ is obtained as $a'_{ij} = a'^{+}_{ij} - a'^{-}_{ij}$, $i = 1, \ldots, n'_t$, $j = 1, \ldots, n'_p$. Note that the arc weights from the symbolic subnet, e.g., $(a'^{-}_{11}, a'^{+}_{12})$, are differentiated from the numerical ones by using an accent ($'$). A PPN model is shown in Figure 2 for illustration purposes.

2.3.1 | Execution semantics

As stated in the last section, the dynamics of PPNs can be described by the join evolution of the numerical and symbolic subnets through the marking given by $\mathbf{M}_k$. Equation (2) is used to model the evolution of the symbolic part of the net, as in ordinary PNs. However, the evolution of $\mathbf{M}_k^{(\mathcal{N})}$ relies on an *ad hoc* information flow based on two basic operations (M. Chiachío et al., 2016, 2018): the *conjunction* and *disjunction* of states of information (Rus et al., 2016; Tarantola & Valette, 1982). In these operations, the first principles of Boolean logic, in particular the logic operators AND ($\wedge$) and OR ($\vee$), are invoked to allow the information from the numerical subnet to be exchanged within a PPN. More specifically, they enable the combination and aggregation of states of information across the numerical subnet of a PPN. To confer the conceptual framework without repeating the literature, the conjunction and disjunction of states of information are briefly explained and illustrated in Figure 3. In Figure 3, the term $\mu(\mathbf{x})$ is the *homogeneous density function* (Mosegaard & Tarantola, 2002; Tarantola & Valette, 1982) that represents the state of complete ignorance about $\mathbf{x} \in \mathcal{X}$, hence providing a reference probability model for $\mathbf{x}$ in the absence of any other

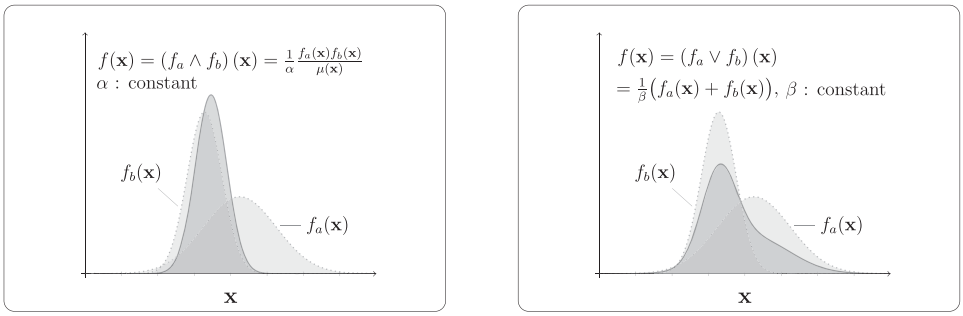


FIGURE 3 Illustration of the conjunction (left) and disjunction (right) of two arbitrary states of information, namely, $f_a(\mathbf{x})$ and $f_b(\mathbf{x})$

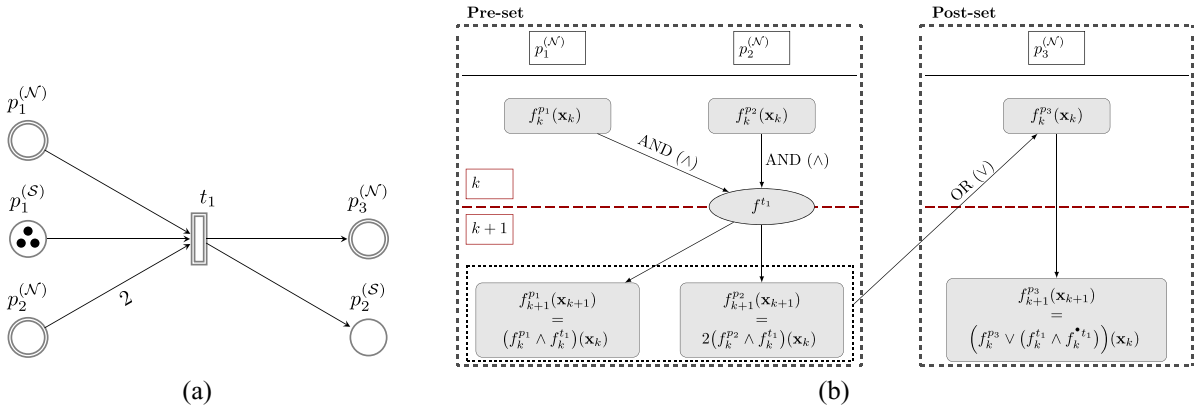


FIGURE 4 (a) PPN of Example 1. (b) Flowchart to illustrate rules (1) to (3) from the PPN execution semantics defined in Section 2.3.1

information. The interested reader is referred to Tarantola and Valette (1982) for further details. From this standpoint, the dynamics of PPNs are formulated under the adoption of the following rules (M. Chiachío et al., 2018):

1. An input arc from place $p_j^{(\mathcal{N})}$ to transition t_i conveys a state of information given by $a_{ij}^{-}(f^{p_j} \wedge f^{t_i})(\mathbf{x}_k)$, which remains in $p_j^{(\mathcal{N})}$ after transition t_i has fired.
2. An output arc from t_i to $p_j^{(\mathcal{N})}$ conveys a state of information given by $a_{ij}^{+}(f^{\bullet t_i} \wedge f^{t_i})(\mathbf{x}_k)$, where $f^{\bullet t_i}(\mathbf{x}_k)$ is the resulting density from the disjunction of the states of information of the preset of t_i.
3. After firing numerical transition t_i, the state of information resulting in the output place $p_j^{(\mathcal{N})}$ is the disjunction of the state of information $f^{p_j}(\mathbf{x}_k)$ (the previous state of information), and $a_{ij}^{+}(f^{t_i} \wedge f^{\bullet t_i})(\mathbf{x}_k)$ (the information produced after firing transition t_i).

Example 1. Figure 4 provides an example of the execution rules presented above, by using a PPN model of one transition, three numerical places, and two symbolic places, as depicted in Figure 4a. In Figure 4b, a conceptual scheme is provided to exemplify rules 1–3 and shows how to obtain the marking at times k and $k+1$. The numerical marking for places $p_1^{(\mathcal{N})}$ to $p_3^{(\mathcal{N})}$ are depicted in rectangular gray boxes. The red dashed line indicates the separation between the marking at k and $k+1$, respectively. Note that at $k+1$, the places $p_1^{(\mathcal{N})}$ and $p_2^{(\mathcal{N})}$ are updated with the information coming from transition t_1 through a conjunction of states of information weighted according to $(a_{11}^{-}=1, a_{12}^{-}=2)$, respectively. Also, observe that the state of information resulting in place $p_3^{(\mathcal{N})}$ after firing transition t_1 is the joint information between the information that existed in $p_3^{(\mathcal{N})}$ at k, and that produced by transition t_1 through the intersection with its preset, i.e., $a_{13}^{+}(f^{t_1} \wedge f^{\bullet t_1})(\mathbf{x}_k) = 1 \cdot (f^{t_1} \wedge (f^{p_1} \vee f^{p_2}))(\mathbf{x}_k)$.

The rules given above are enough to describe the dynamics of the numerical subnet of PPNs. However, they can be synthesized through an algebraic expression describing a dynamical state equation, as follows (M. Chiachío et al., 2018):

$$\mathbf{M}_{k+1}^{(\mathcal{N})} = \left[\mathbf{M}_k^{(\mathcal{N})} \circ \boldsymbol{\gamma}_k + \left(\sum_{i=1}^{n_t} (\mathbf{a}_i^{+})^T \otimes \mathbf{c}_i + (\mathbf{A}^{-})^T \circ \mathbf{B}\right) \cdot \mathbf{v}_k\right] \circ \boldsymbol{\beta}_k \quad (4)$$

where $\mathbf{v}_k = (v_{1,k}, v_{2,k}, \ldots, v_{n_t,k})^T$ is the firing vector for the numerical subnet (numerical and mixed transitions) at state k; $\mathbf{A}^{-}$ is the backward incidence matrix; and $\mathbf{a}_i^{+}$ is a column vector corresponding to the ith row of the forward incidence matrix $\mathbf{A}^{+}$. The term $\mathbf{B}$ is an $(n_p \times n_t)$ matrix whose (i,j)th element is given by the conjunction

of states of information between f^{p_j} and f^{t_i} (expressed by $f^{p_j} \wedge f^{t_i}$). Next, $\mathbf{c}_i$ is an n_t-dimensional row vector given by $\mathbf{c}_i = (f^{t_i} \wedge f^{*t_i}) \cdot \boldsymbol{\delta}_{i\ell}$, where $\boldsymbol{\delta}_{i\ell}$ is a vector whose elements are the Kronecker delta of variables i and ℓ, which makes all elements zero except for $i = \ell, \ell = 1, \ldots, n_t$. The term $\boldsymbol{\gamma}_k$ is an n_p-dimensional column vector of binary constants, i.e., $\boldsymbol{\gamma}_k = (\gamma_k^{(1)}, \ldots, \gamma_k^{(j)}, \ldots, \gamma_k^{(n_p)})^T$, where $\gamma_k^{(j)}$ is given by

$$\gamma_k^{(j)} = \begin{cases} 1, & \text{if} \quad \sum_{i=1}^{n_t} a_{ij}^{+} v_{i,k} > 0 \\ 1, & \text{if} \quad \left(\sum_{i=1}^{n_t} a_{ij}^{+} v_{i,k} = 0 \;\; \& \sum_{i=1}^{n_t} a_{ij}^{-} v_{i,k} = 0 \right) \\ 0, & \text{if} \quad \left(\sum_{i=1}^{n_t} a_{ij}^{+} v_{i,k} = 0 \;\; \& \sum_{i=1}^{n_t} a_{ij}^{-} v_{i,k} > 0 \right) \end{cases} \tag{5}$$

Finally, $\boldsymbol{\beta}_k$ is a vector of normalizing constants required for $\mathbf{M}_k^{(\mathcal{N})}$ to be a vector of bona fide densities. In Equation (4), the symbols $\langle \otimes, \circ, \cdot \rangle$ are used to denote the matrix outer product, Hadamard product, and inner product (Ando, 1995; Beezer, 2007), respectively. Also, the (i, j)th element from $(\mathbf{A}^-)^T \circ \mathbf{B}$ equals $a_{ij}^{-}(f_k^{p_j} \wedge f_k^{t_i})(\mathbf{x}_k)$ and corresponds to the state of information that remains in place in $p_j^{(\mathcal{N})}$ at $k+1$ after firing t_i, given that $p_j^{(\mathcal{N})} \in t_i^{\bullet}$ is isolated from output arcs. Finally, observe that the summation of outer products $\sum_{i=1}^{n_t} (\mathbf{a}_i^{+})^T \otimes \mathbf{c}_i$ in Equation (4) renders an $(n_p \times n_t)$ matrix whose (i, j)th element represents a weighted density function corresponding to the state of information added to postset place $p_j^{(\mathcal{N})}$ after transition t_i has been fired. The interested reader should refer to (M. Chiachío et al., 2018) for further details.

2.3.2 | Rule of transition firing

In PPNs, any transition $t_i \in \mathbf{T}$ is fired at time k if the delay time has passed and

1. every symbolic place from the preset of t_i has enough tokens according to their input arc weight, as in classical PNs.
2. each of the conjunction of states of information between f^{t_i} and $f_k^{p_j}$ is possible, where $p_j^{(\mathcal{N})}$ belongs to the preset of t_i.
3. conditions 1 and 2 are both satisfied when t_i is a mixed transition, i.e., $t_i \in (\mathbf{T}^{(S)} \cap \mathbf{T}^{(\mathcal{N})})$.

Note from condition 2 that a conjunction, e.g., $(f_k^{p_j} \wedge f_k^{t_i})(\mathbf{x}_k)$, is possible if $(f_k^{p_j} \wedge f_k^{t_i})(\mathbf{x}_k) \neq \emptyset$ (Tarantola & Valette, 1982). Note also that when any of the states of information involved in a conjunction is the *homogenous density* (also referred to as "noninformative density") $\mu(\mathbf{x}_k)$ of the state space of consideration $\mathcal{X}$, then the conjunction is always possible (Tarantola & Valette, 1982), thus condition 2 is automatically fulfilled. This argument is important in terms of using PPNs in practical applications, as will be demonstrated in the next section. Further details about this aspect can be found in M. Chiachío et al. (2018).

2.3.3 | PPN algorithm

The recursive scheme for PPNs explained above is, in general, difficult to evaluate analytically because the normalizing constants involved in the conjunction of states of information of Equation (4) are difficult to evaluate in practical cases. Furthermore, there are situations where the density functions are not completely known. To alleviate this drawback and confer the required versatility to the PPN methodology, particle methods (Arumlampalam, Maskell, Gordon, & Clapp, 2002; Doucet, De Freitas, & Gordon, 2001) can be used to circumvent the evaluation of the normalizing constants with a feasible computational cost. In this section, a pseudocode implementation of PPNs is provided, which combines the PPN algebra with the particle approximation for the conjunction and disjunction of states of information. Three main blocks comprise the pseudocode, namely, *transition firing, information exchange*, and *marking evolution*, which have been remarked for clarity. For better readability, the pseudocode has been provided as Algorithm 1 in Appendix A. In addition, some mathematical insights into the particle approximation of the conjunction and disjunction of states of information have been provided in Appendix B. Observe from Algorithm 1 that the normalizing constants from Equation (4) have been omitted because the particle approximation bypasses them through resampling.

Example 2. The suitability of the PPN algorithm to reproduce the execution semantics rules given in Section 2.3.1 is illustrated here using a numerical example. To this end, let us consider again the PPN given in Figure 4a; however, now the system states are considered as two-dimensional, i.e., $\mathbf{x}_k \in \mathcal{X} \subset \mathbb{R}^2$. The initial marking $\mathbf{M}_0 = (f_0^{p_1}, f_0^{p_2}, f_0^{p_3})^T$ is described using 2D Gaussians as follows: $f_0^{p_1}(\mathbf{x}) \sim \mathcal{N}([20, 15], \boldsymbol{\Sigma}_1)$, where $\boldsymbol{\Sigma}_1 = diag(5^2, 3^2)$; $f_0^{p_2}(\mathbf{x}) \sim \mathcal{N}([30, 30], \boldsymbol{\Sigma}_2)$, with $\boldsymbol{\Sigma}_2$ being $\boldsymbol{\Sigma}_2 = diag(4^2, 2^2)$; and finally $f_0^{p_3}(\mathbf{x}) = \emptyset$. In this example, the transition t_1 is defined using a Dirac delta density function, i.e., $f^{t_1} \sim \mathbb{I}_{\mathcal{B}}(\mathbf{x}_k)$; thus, its firing is prescribed for the state variable $\mathbf{x}_k$ on fulfilling the proposition $\mathbf{x}_k \in \mathcal{B}$, where $\mathcal{B} \subseteq \mathcal{X}$ is a region of the $\mathbf{x}$-space defined as: $\mathcal{B} = \{\mathbf{x} \in \mathcal{X} : (x_1 - 20)^2 + (x_2 - 20)^2 \leqslant 10\}$. Algorithm 1 has been applied using $N = 1{,}000$ particles to evaluate the system state evolution, described through the marking $\mathbf{M}_k$, $k > 0$. The results for the numerical places $p_1^{(\mathcal{N})}$ (left) to $p_3^{(\mathcal{N})}$ (right) are depicted in Figure 5 for $k = 0$ (upper panels) to $k = 1$ (lower panels). Each subplot represents a state of information about $\mathbf{x}_k$ using samples (gray circles) in the state space $\mathcal{X}$. The blue circle represents the region $\mathcal{B}$. Observe that initially, at $k = 0$, transition t_1 is enabled because $p_1^{(S)}$ is assigned one token and $(f_0^{p_i} \wedge f_0^{t_1})(\mathbf{x}_k) \neq \emptyset$, for $i = 1, 2$.

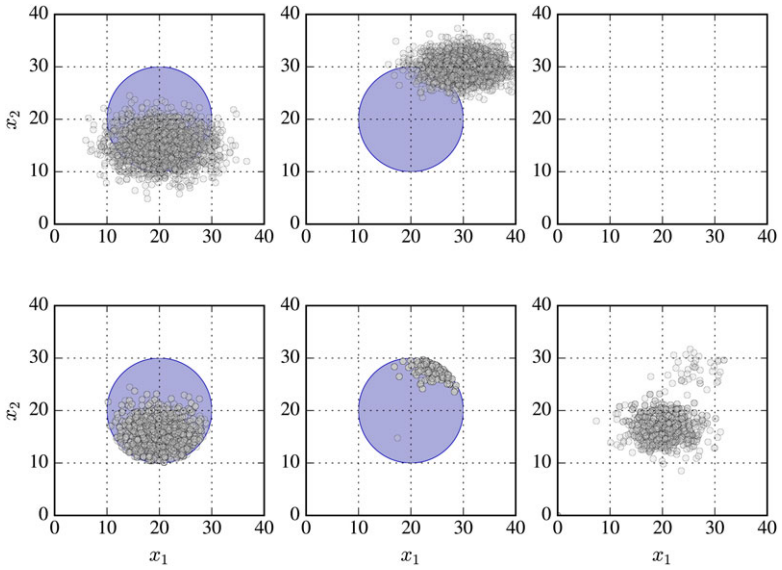


FIGURE 5 Output of the PPN from Figure 4a given for numerical places $p_1^{(\mathcal{N})}$ (leftmost panel) to $p_3^{(\mathcal{N})}$ (rightmost panel) at $k = 0$ (upper panels) and $k = 1$ (lower panels)

Once t_1 is fired, then the information coming from $p_1^{(\mathcal{N})}$ and $p_2^{(\mathcal{N})}$ is incorporated and the resulting state of information is subsequently transferred to $p_3^{(\mathcal{N})}$ (recall Figure 4b for a conceptual illustration of the information flow in PPNs). Note that the information coming from $p_2^{(\mathcal{N})}$ prevails in $p_3^{(\mathcal{N})}$ after firing t_1, precisely because the arc weight from $p_2^{(\mathcal{N})}$ to t_1 (a_{21}^-) equals two, while $a_{11}^- = 1$. Note also that the PDFs $f_1^{p_1}$ and $f_1^{p_2}$ are concentrated within the region $\mathcal{B}$ (depicted as a blue circle in Figure 5) because of the Dirac delta PDF in f^{t_1}, which acts as a filter canceling any information out of the region $\mathcal{B}$ once t_1 is fired. Finally, observe also that this example numerically shows how both the numerical and symbolic tokens interact in a synchronized manner in PPNs.

3 | SELF-ADAPTATION MODELING BY BAYESIAN LEARNING OF PPNs

Lemma 1. *In PNNs, any input arc from place $p_j^{(\mathcal{N})}$ to transition $t_i \in \mathbf{T}^{(\mathcal{N})}$ conveys a state of information given by the posterior density function of state variable $\mathbf{x}_k$, by adopting the following assumptions:*

1. *at time k, there are available data, denoted by $\mathbf{y}_k \in \mathcal{D}$;*
2. *$f^{t_i}(\mathbf{x}_k)$ acts as likelihood function for $\mathbf{y}_k$, hence t_i is named as data-transition; and*
3. *$\mathcal{X}$ is a linear space.*

Proof. Let us consider that the state of information $f^{p_j}(\mathbf{x}_k)$ represents a prior PDF $p(\mathbf{x}_k)$ for $\mathbf{x}_k$ within the state space $\mathcal{X}$. Let us now rewrite the likelihood function $f^{t_i}(\mathbf{x}_k)$ as $p(\mathbf{y}_k|\mathbf{x}_k)$. From Equation (4), the (i, j)th element of matrix $\mathbf{B}$ is given by $(f^{p_j} \wedge f^{t_i})(\mathbf{x}_k)$, which, by definition of conjunction of states of information, is given by Tarantola and Valette (1982):

$$\left(f^{p_j} \wedge f^{t_i}\right)(\mathbf{x}_k) \overset{\text{def}}{=} \alpha_{ij} \frac{f^{p_j}(\mathbf{x}_k) f^{t_i}(\mathbf{x}_k)}{\mu(\mathbf{x})} \propto p(\mathbf{y}_k|\mathbf{x}_k) p(\mathbf{x}_k) \tag{6}$$

where α_{ij} is a normalizing constant, and $\mu(\mathbf{x})$ is the homogeneous density function, which is also a constant because $\mathcal{X}$ is a linear space (Tarantola, 2005). By Bayes' theorem (recall Equation (1)), the resulting PDF can be interpreted as the posterior PDF of state vector $\mathbf{x}_k$ given data $\mathbf{y}_k \in \mathcal{D}$, except for a normalizing constant. □

Corollary 1. *Under the assumptions given above, and by considering that:*

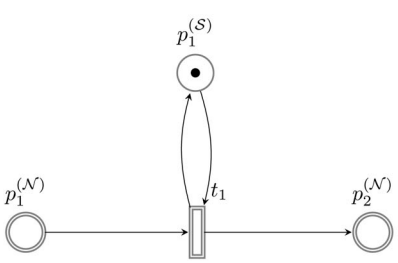
1. *$p_j^{(\mathcal{N})} \in {}^\bullet t_i$ is isolated from output arcs and*
2. *firing of transitions $t_i^{(\mathcal{N})} \in {}^\bullet\mathbf{T}_{p_j}$ is nonconcomitant, with ${}^\bullet\mathbf{T}_{p_j}$ being the set of transitions whose input arcs come from place $p_j^{(\mathcal{N})}$;*

then the marking $\mathbf{M}_{k+1}^{(\mathcal{N})}(j)$ can be just obtained by recurrence as a posterior Bayesian estimation from $\mathbf{M}_k^{(\mathcal{N})}(j)$ except for a normalizing constant, as follows:

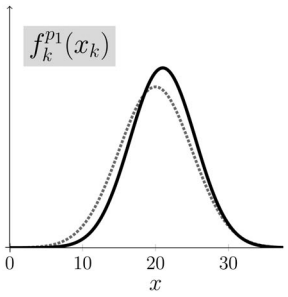
$$\mathbf{M}_{k+1}^{(\mathcal{N})}(j) \propto p(\mathbf{x}_k|\mathbf{y}_k) \tag{7}$$

where $\mathbf{x}_k \sim f^{p_j}(\mathbf{x}_k) = \mathbf{M}_k^{(\mathcal{N})}(j)$. In such case, $p_j^{(\mathcal{N})}$ is denoted as a learning-place.

The derivation of Corollary 1 can be straightforwardly obtained from Equation (4) by just considering that $a_{ij}^+ = 0, \forall i = 1, \ldots, n_t$ (from Assumption 1) and also that $\sum_{i=1}^{n_t} a_{ij}^+ v_{i,k} = 0$ and $\sum_{i=1}^{n_t} a_{ij}^- v_{i,k} > 0$ (from Assumption 2), then $\gamma_k^{(j)} = 0$ (recall Equation (5)).



(a)



(b)

FIGURE 6 (a) PPN of Example 3. (b) Representation of the resulting PDF (solid line) in place $p_1^{(\mathcal{N})}$ from the evaluation of Equation (7) at $k = 1$. Note that the prior $f_0^{p_1}$ is represented by the dotted gray line

Example 3. Figure 6a represents a simple PPN where f^{t_1} acts as a likelihood function assumed as Gaussian, given by $f^{t_1} = \mathcal{N}(25, 5)$, where units are dimensionless. The probability densities for initial marking $\mathbf{M}_0^{(\mathcal{N})}$ are Gaussians given by: $f_0^{p_1} = \mathcal{N}(20, 5)$, $f_0^{p_2} = \mathcal{N}(10, 5)$, where $f_0^{p_1}$ is the prior PDF about the unidimensional variable x_k. The initial marking $\mathbf{M}_0^{(\mathcal{S})}$ comprises one token in $p_1^{(\mathcal{S})}$. According to the graph given by Figure 6a, the incidence matrices used for the calculations of the numerical and symbolic subnets are:

$$\left.\begin{array}{l}\mathbf{A}^{+(\mathcal{N})} = (0\ 1)\\ \mathbf{A}^{-(\mathcal{N})} = (0\ 1)\end{array}\right\} \rightarrow \mathbf{A}^{(\mathcal{N})} = (-1\ 1) \tag{8a}$$

$$\left.\begin{array}{l}\mathbf{A}^{+(\mathcal{S})} = (1)\\ \mathbf{A}^{-(\mathcal{S})} = (1)\end{array}\right\} \rightarrow \mathbf{A}^{(\mathcal{S})} = (0) \tag{8b}$$

respectively. Observe that in this example, the subnet given by $\{p_1^{(\mathcal{N})}, t_1\}$ fulfills the assumptions given in Lemma 1 and also $a_{11}^{+} = 0$, $\gamma_k^{(1)} = 0$ (assuming that t_1 is fired). Henceforth, according to Corollary 1, the marking $M_1^{(\mathcal{N})}(1)$ can be straightforwardly obtained by Equation (7) as a Bayesian posterior from $M_0^{(\mathcal{N})}(1) = f_0^{p_1}$, i.e.,

$$M_1^{(\mathcal{N})}(1) = f^{p_1} \wedge f^{t_1} \propto \mathcal{N}(25, 5)\mathcal{N}(20, 5) \tag{9}$$

Note that the same result can be obtained using Equation (4). Indeed, observe that in this example, the matrix $\sum_{i=1}^{n_t} (\mathbf{a}_i^{+})^T \otimes \mathbf{c}_i = (\mathbf{a}_1^{+})^T \otimes \mathbf{c}_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \left(f^{t_1} \wedge f_0^{\bullet t_1} \right) = \begin{pmatrix} 0 \\ 1 \cdot f^{t_1} \wedge f_0^{\bullet t_1} \end{pmatrix}$, which gives 0 for the component corresponding to $p_1^{(\mathcal{N})}$. Note also that $\gamma_0^{(1)} = 0$, which is obtained using Equation (5). Henceforth, the resulting PDF in place $p_1^{(\mathcal{N})}$ at $k = 1$ is the (1,1)th element from matrix $(\mathbf{A}^{-})^T \circ \mathbf{B}$, i.e., $\underbrace{a_{11}^{-}}_{=1} f_0^{p_1} \wedge f_0^{t_1} \propto \mathcal{N}(25, 5)\mathcal{N}(20, 5)$, which coincides with the result from Equation (9). Figure 6b provides a graphical representation of marking $M_k^{(\mathcal{N})}(1)$ for $k = 0, 1$.

Remark 1. Let us denote by $t_{\hat{i}}^{(\mathcal{N})}$ a specific data-transition within a PPN and let k be a time instant when data $\mathbf{y}_k$ are available. A self-adaptive expert system can be modeled via PPNs by setting a number of learning places. Any learning place $p_{\hat{j}}^{(\mathcal{N})}$ can be set and updated within a PPN by adopting the following procedure:

1. Make $a_{i\hat{j}}^{+} = 0, \qquad \forall i = 1, \ldots, n_t.$
2. Set firing constants $v_{i,k}$ for $t_i^{(\mathcal{N})} \in {}^{\bullet}\mathbf{T}_{p_{\hat{j}}}$ such that $v_{i,k} = \mathbb{I}_{\hat{i}}(i)$, where $\mathbb{I}_{\hat{i}}(i) = 1$ if $i = \hat{i}$, and 0 otherwise.
3. Adopt $\gamma_k^{(\hat{j})} = 0$.
4. Set $M_{k+1}^{(\mathcal{N})}(j) \propto p(\mathbf{x}_k | \mathbf{y}_k)$.

Remark 2. The methodology presented above reveals that PPNs are useful for modeling adaptive expert systems because they can deal with uncertain information such as the one coming from noisy condition monitoring data and expert opinion (Chiachío et al., 2018), and automatically update it for decision making. The resulting computational framework allows building computational models acting as self-adaptive expert systems, as shown in the next section within the context of a case study.

4 | CASE STUDY: APPLICATION OF METHODOLOGY TO RAILWAY TRACK INSPECTION

In this section, the computational framework presented above is demonstrated in a case study for railway track inspection management. One of the main interests of this exercise is to provide evidence about the self-adaptation of PPNs using monitoring data. To this end, a PPN-based expert system is provided to generate autonomous and adaptive management decisions based on monitoring data about the state of geometry deterioration of a railway track, as depicted by Figure 7.

In this case study, the deterioration of the track is assumed to occur due to traffic loading expressed in *cycles*, which represent an integer amount of train axles that have passed

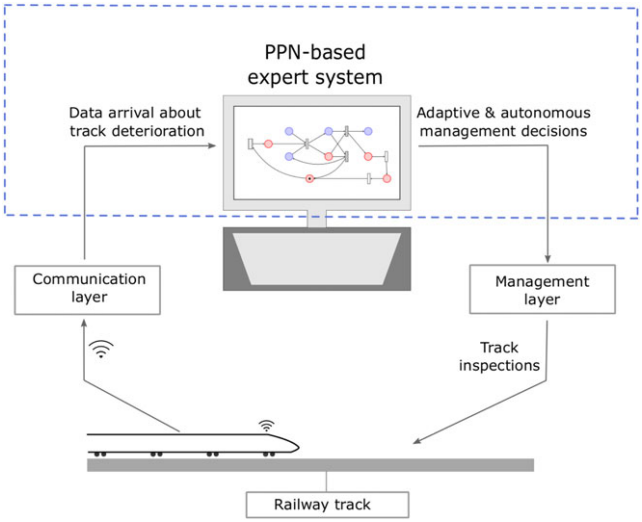


FIGURE 7 Schematic diagram about a self-adaptive expert system using PPNs applied to a railway track maintenance problem

through a particular track section during an operation period. Also, it is assumed that track measurement trains are used to provide measurements of the track state (in particular, the track settlement) at a set of regularly or nonregularly scheduled cycles. Observe from Figure 7 that every time the track measurement train provides new data about track settlement, the PPN-expert system uses that data to generate information to support decisions related to the scheduling of inspections by track engineers, as depicted by the dashed-blue rectangle. The track engineers' inspections do not provide new data points and are undertaken to confirm the requirement for a particular type of maintenance, such as *tamping* or *stoneblowing* (Esveld, 1989; Selig & Waters, 1994), or for speed restrictions or line closures to be imposed. The PPN-based expert system uses the available measurement data to ensure that an alarm is raised about the track condition if necessary or that track engineers' inspections are scheduled according to the latest knowledge of the track condition. This condition-based inspection (CBI) therefore acts to ensure that any disruption to operations caused by these inspections is minimized and that the required resources are used efficiently.

4.1 | Problem description

Railway track geometry deterioration due to traffic loading is a critical railway operation and maintenance problem with important implications in safety and cost. When measured track irregularities exceed allowable limits, either traffic speed restrictions have to be prescribed, or corrective maintenance interventions like tamping have to be performed to restore the track to an acceptable geometry. The modeling of the track geometry degradation is a core element in the railway maintenance problem (Andrews, 2012; Andrews, Prescott, & De Rozières, 2014). In general terms, the temporal evolution of the track degradation can be assumed to be given by a state-space model, as

$$x_k = h_k(x_{k-1}) + v_k \tag{10a}$$

$$y_k = x_k + w_k \tag{10b}$$

where x_k denotes the latent state of degradation at time or cycle $k \in \mathbb{N}$, y_k is a measurement of the degradation at time k, and v_k and w_k are random variables that represent the process noise and the measurement error, respectively. Supported by the principle of maximum information entropy (Jaynes, 1983), v_k and w_k can be conservatively modeled as zero-mean Gaussian distributions; thus, the state-space model in Equation (10) can be rewritten probabilistically as:

$$p(x_k|x_{k-1}) = \mathcal{N}(h_k(x_{k-1}), \sigma_{v_k}) \tag{11a}$$

$$p(y_k|x_k) = \mathcal{N}\left(x_k, \sigma_{w_k}\right) \tag{11b}$$

where σ_{v_k} and σ_{w_k} are the standard deviation of the model error and the measurement error, respectively. For this case study, a cycle-to-cycle incremental model for railway track degradation is adopted; thus, the *state transition function* h_k in Equation (11a) can be expressed as $h_k = x_{k-1} + \Delta x_k$, where Δx_k is the increment of track degradation in load cycle k. Here, the (latent) state of track degradation at loading cycle k is assumed to be given by the permanent settlement of the track in a particular track section. To calculate the cyclic increment of track settlement Δx_k, the semiempirical cyclic densification model from Indraratna, Thakur, Vinod, and Salim (2012) is adopted. This model is based on the theory of plasticity of soils (Yu, 2007) and the postulates of *critical state soil mechanics* (Schofield & Wroth, 1968), but for the sake of simplicity, it is not reproduced here. The interested reader is referred to J. Chiachío, M. Chiachío, Prescott, and Andrews (2017) and Indraratna et al. (2012) for further modeling details, and to Dahlberg (2001), Soleimanmeigouni and Ahmadi (2016), and Higgins and Liu (2018) for a comprehensive overview of track degradation models.

The data in this case study consist of a set of nonregularly scheduled (noisy) measurements $Y = (y_1, y_2, \dots, y_k)$ of track settlement taken from Aursudkij et al. (2009), which are sequentially introduced to the system at a set of discrete loading cycles. This data set corresponds to a laboratory simulation of traffic loadings for a 20-tonne axle-load train over a ballasted track section composed of 0.9 (m) (depth) subgrade material and 0.3 (m) (depth) ballast material, carried out in the Railway Test Facility (RTF) of the University of Nottingham (UK). The data set is reproduced in Table 1. The reader is referred to Aursudkij et al. (2009) for further information about the experimental setup whereby the data were collected, and to Brown, Brodrick, Thom, and McDowell (2007) for a detailed description on the Nottingham RTF.

TABLE 1 Experimental sequence of permanent unitary settlement (strain) data used for calculations, taken from Aursudkij et al. (2009)

Loading cycles $k(\times 10^3)$	0	0.625	1.25	2.5	5	10	20	30	50	75
Unitary settlement (dimensionless)	0	0.0017	0.0045	0.0058	0.0075	0.0087	0.0104	0.011	0.012	0.01275

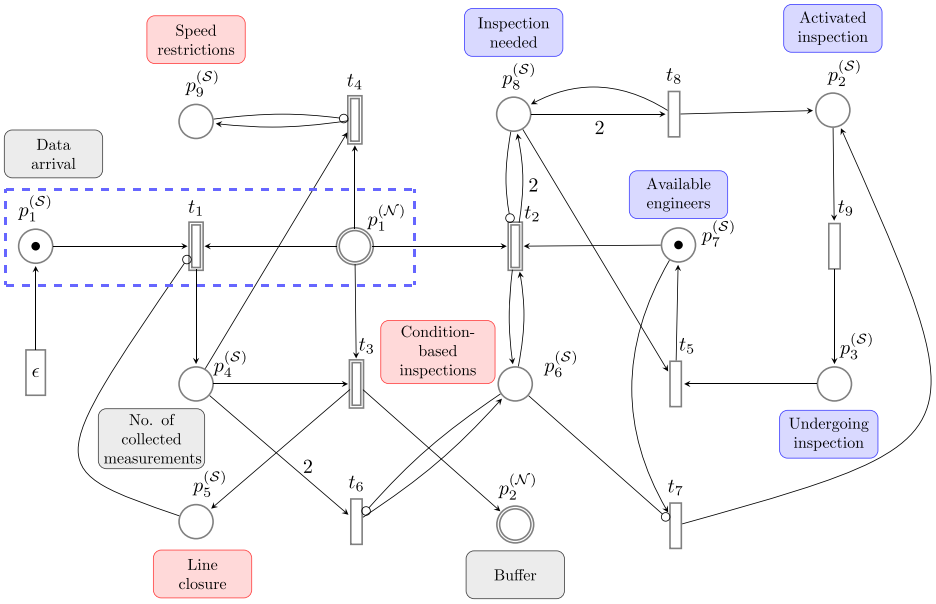


FIGURE 8 PPN of the case study presented in Section 4. The dashed rectangle indicates the nodes that enable the Bayesian updating of the numerical variable x_k

4.2 | Plausible Petri net model

The PPN-based expert system for railway track maintenance considered in this case study is depicted in Figure 8. The system represents a number of rules that autonomously raise an alarm (e.g., "line closure") or trigger inspection activities of a particular railway track section subjected to traffic loading degradation. As shown in Figure 8, the PPN is composed of two numerical places ($p_1^{(\mathcal{N})}$, $p_2^{(\mathcal{N})}$), nine symbolic places ($p_1^{(\mathcal{S})}$ to $p_9^{(\mathcal{S})}$), four mixed transitions (t_1 to t_4), and five symbolic transitions (t_5 to t_9). The stochastic model for track degradation represented in Equation (11a) is embedded within the numerical place $p_1^{(\mathcal{N})}$. Thus, the state of information in $p_1^{(\mathcal{N})}$ is given by:

$$f^{p_1}(x_k) = p(x_k|x_{k-1}) = \left(2\pi\sigma_{v_k}^2\right)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\left(\frac{h_k(x_{k-1}) - x_k}{\sigma_{v_k}}\right)^2\right) \quad (12)$$

Numerical place $p_2^{(\mathcal{N})}$ represents a buffer of information (initially empty), which collects the posterior values of track settlement leading to the "line closure" state, as will be explained further below. Discrete-event states such as "inspection needed," "activated inspection," etc., as specified by the colored text labels in Figure 8, are modeled by the symbolic places $p_1^{(\mathcal{S})}$ to $p_9^{(\mathcal{S})}$, respectively. Places $p_5^{(\mathcal{S})}, p_6^{(\mathcal{S})}, p_9^{(\mathcal{S})}$ have red labels and represent irreversible discrete states, i.e., nonnumerical discrete states of the track degradation that permanently remain in the same condition once their corresponding places (e.g., $p_9^{(\mathcal{S})}$) have been marked. The gray text labels provide explanatory information about places. Observe also from Figure 8 that a number of *inhibitor arcs* (Murata, 1989; Schneeweiss, 2001) (those ending with a small circle) are used to produce the opposite effect of the rule described in Equation (3), i.e., they prevent a transition from enabling once its preset places are marked.

The system is fed during runtime using measurements coming from monitoring data. A *cold transition* (ϵ) is used to represent the data arrival, which are assumed to be available at a set of nonregularly scheduled time instants, as shown in Table 1. For this case study, the measurements $y_1, y_2, \ldots, y_k$ are assumed to come with a 5% white-noise-type error; so, they are considered as random realizations of a Gaussian PDF centered in the latent state x_k with standard deviation given by $\sigma_{w_k} = 0.05\|y_k\|$, taking it as known, i.e., $y_k \sim \mathcal{N}(x_k, \sigma_{w_k})$ (recall Equation (11b)). This PDF represents the state of information within transition t_1, given by:

$$f^{t_1}(x_k) = p(y_k|x_k) = \left(2\pi\sigma_{w_k}^2\right)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\left(\frac{y_k - x_k}{\sigma_{w_k}}\right)^2\right) \quad (13)$$

TABLE 2 Description of the transitions shown in Figure 8. In the third column (rules), the delays are expressed in cycles. The last column provides a description of the action taken by the PPN-expert system when the rules are met

ID	Type	Rule	State of information	Action
t_1	Mixed	–	$f^{t_1} \sim p(y_k \mid x_k)$ (Likelihood)	Update predictions
t_2	Mixed	$H(x_k) \geqslant -4.8$	$f^{t_2} \sim \mathbb{I}_{\mathcal{C}_2}(x_k)$	Activates CBI
t_3	Mixed	$\mathbb{E}_{f^{p_1}}[x_k] \geqslant 0.014$ [m]	$f^{t_3} \sim \mathbb{I}_{\mathcal{C}_3}(x_k)$	Switches to LC
t_4	Mixed	$\mathbb{E}_{f^{p_1}}[x_k] \geqslant 0.012$ [m]	$f^{t_4} \sim \mathbb{I}_{\mathcal{C}_4}(x_k)$	Switches to SR
t_5	Symbolic	$\tau_5 \sim \mathcal{N}(20, 1)$ (delay)	–	Concludes inspections
t_6	Symbolic	$\tau_6 = 0$ (delay)	–	Switches to CBI
t_7	Symbolic	$\tau_7 = 0$ (delay)	–	Switches to PI
t_8	Symbolic	$\tau_8 \sim \mathcal{N}(1, 1)$ (delay)	–	Activates CBI
t_9	Symbolic	$\tau_9 = 4.0$ (delay)	–	Activates inspections

PI: periodic inspections, CBI: condition-based inspections, SR: speed restrictions, LC: line closure.

Note that each time a new measurement arrives, transition t_1 is enabled, which, by the PPN execution rules explained in Section 2.3.1, leads to the conjunction of the states of information of $p_1^{(\mathcal{N})}$ and t_1 (Equations 12 and 13, respectively). By Lemma 1, this conjunction leads to the posterior PDF and therefore to the update of the degradation variable x_k in place $p_1^{(\mathcal{N})}$, except for a normalizing constant. It follows that self-adaptation for this particular example is enabled by means of the subnet $\{p_1^{(S)}, p_1^{(\mathcal{N})}, t_1\}$.

By evaluating the proposed PPN-based system, changes in the numerical and discrete track states are obtained with reference to a number of automated actions that are activated through firing transitions t_1–t_9. An overview of the complete set of transitions is provided in Table 2. Observe from this table that the mixed transitions t_2, t_3, and t_4 are defined based on condition, and henceforth, their activation is prescribed for the state variable x_k on fulfilling the condition $x_k \in \mathcal{C}_i$, where subspaces $\mathcal{C}_i$, $i = 2, 3, 4$, are specified in the third column of Table 2. These transitions are driven by states of information that are expressed by Dirac delta density functions (Chiachío et al., 2018), i.e., $f^{t_i} = \mathbb{I}_{\mathcal{C}_i}(x_k), i = 2, 3, 4$. In Table 2, function $H : \mathcal{X} \to \mathbb{R}$ denotes the differential entropy (DE) of the degradation variable x_k, as a measure of the degree of belief about the values taken by x_k, which is given by $^1/_2 \ln[(2\pi e)\text{var}(x_k)]$. Also, $\mathbb{E}_{f^{p_1}}[x_k]$ denotes the expectation of x_k with respect to f^{p_1}. From a computational point of view, the conditions in the mixed transitions t_2–t_4 specify numerical rules used by the expert system to raise an alarm or trigger an inspection action.

The dynamics of the PPN-based expert system are briefly described next. Initially, at $k = 0$, the system starts in the dual discrete state "Data arrived" and "available engineers," represented by one token at $p_1^{(S)}$ and $p_7^{(S)}$, respectively, thus $\mathbf{M}_0^{(S)} = (1, 0, 0, 0, 0, 0, 1, 0, 0)^T$. In practice, this symbolizes an initial stage of the track where a first measurement is taken, almost no track degradation is observed, and engineers are available in case inspections are required. The initial marking of the numerical places is given by $f_0^{p_1} = \mathcal{N}(0, 1)$, $f_0^{p_2} = \emptyset$ (recall that $p_2^{(\mathcal{N})}$ represents a buffer of information that is initially empty). Subsequently, the numerical state variable x_k in $p_1^{(\mathcal{N})}$ starts evolving over time following Equation (10a), which is updated online by means of the subnet $\{p_1^{(S)}, p_1^{(\mathcal{N})}, t_1\}$ each time new data point arrives. In parallel, a sequence of *periodic inspection* (PI) activities takes place through subnet $\{p_2^{(S)}, p_3^{(S)}, p_7^{(S)}, p_8^{(S)}, t_5, t_7, t_8, t_9\}$, which is enabled because $p_7^{(S)}$ is initially assigned with one token, i.e., $M_0^{(S)}(7) = 1$. Note that the system keeps running in PI mode until place $p_6^{(S)}$ receives a token, whereupon it switches to *CBI* mode. The CBI mode allows the expert system to trigger an inspection only when the condition $x_k \in \mathcal{C}_2$ is met, i.e., when the uncertainty observed in the predicted track settlement is higher than a certain value.

Note also from Figure 8 that a *concurrency* takes place through the subnet $\{p_1^{(\mathcal{N})}, t_2, t_3\}$, hence once any of the numerical rules represented by t_2 and t_3 are met, then the referred transitions (not necessarily both nor simultaneously) are enabled whereupon two possible sequences of actions are activated, namely: (a) a set of inspection activities represented by the transitions t_5, t_8, t_9 (when t_2 is fired) and (b) the closure of the line by firing t_3 (when the mean predicted track settlement passes 0.014 m), which subsequently disables t_1 and makes the self-adaptive expert system stop. In that case, the buffer in place $p_2^{(\mathcal{N})}$ will collect and store the state of information taking place at $p_1^{(\mathcal{N})}$ when t_3 is fired. Finally, note that $p_4^{(S)}$ collects a number of tokens equivalent to the amount of collected measurements, and allows t_6 to be fired, provided that at least two measurements are available, whereupon $p_6^{(S)}$ is marked and the system automatically switches from PI to CBI mode by disabling transition t_7. The two tokens required in $p_4^{(S)}$ to fire t_6 are a prerequisite imposed on the system to avoid triggering CBIs based on predictions of the track settlement x_k with little or no learning from the data. By this means, at least two data points are assured to train the model

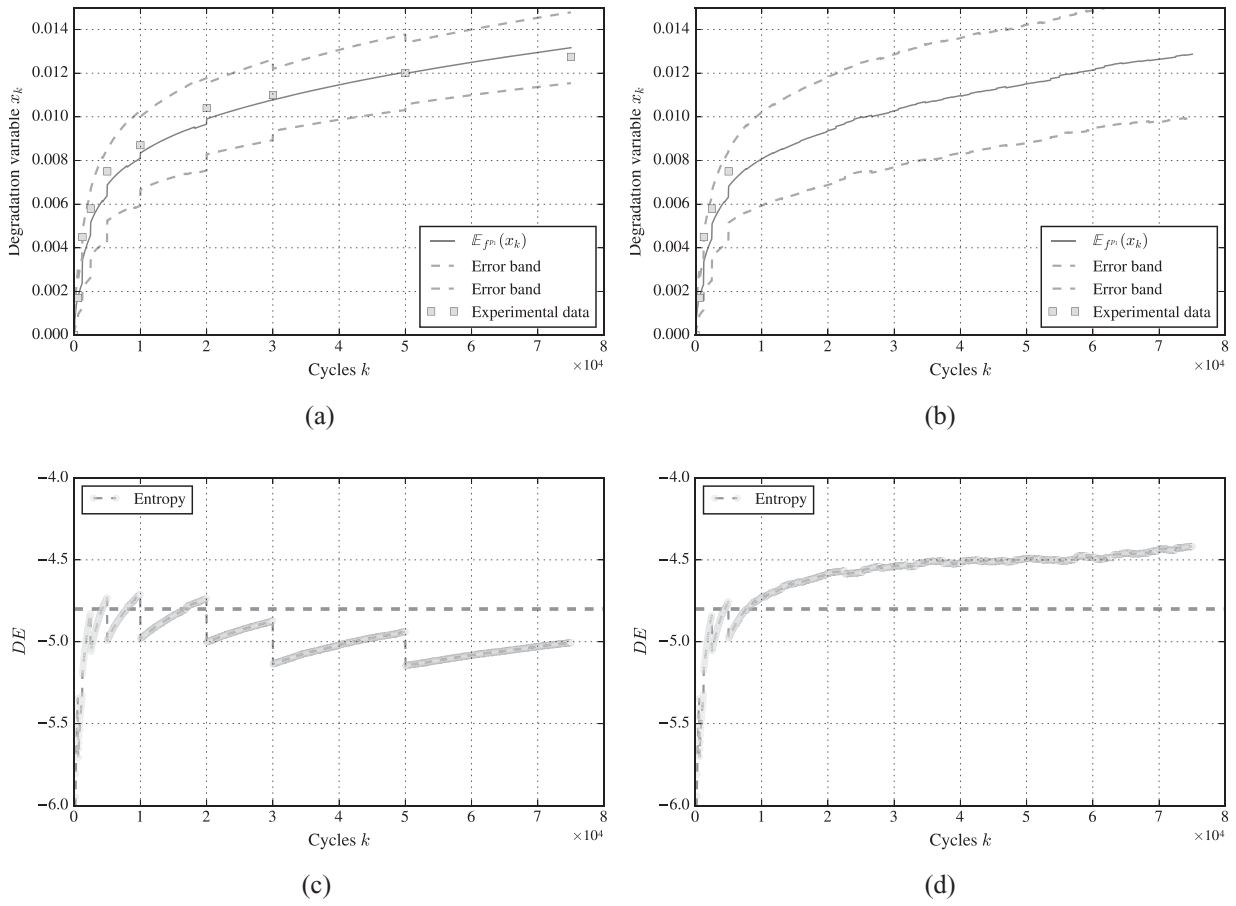


FIGURE 9 Upper panels: Plot of mean values and probability bands of $f_k^{p_1}$ for $k = 0$ to $k = 75{,}000$ considering (a) the complete set of measurement data arrived and (b) only the first four data points. Lower panels: History plot of the differential entropy of $f_k^{p_1}$ for (c) the complete and (d) the reduced data set as mentioned before. The dashed-horizontal line represents the threshold value (-4.8) given to activate transition t_2

so that the track settlement predictions are sufficiently reliable to be used for decision making.

4.3 | Results

The simulation of the PPN-based expert system shown in Figure 8 yields predicted information about the state variable x_k, along with the sequence of discrete events, such as activation of inspection, data arrival, etc. Algorithm 1 is applied to obtain the overall system evolution described through the marking $\mathbf{M}_k$, $k > 0$, using $N = 5{,}000$ samples. The results for the estimated degradation variable in place $p_1^{(\mathcal{N})}$ along with its $5\% - 95\%$ probability bands are depicted in Figure 9 for $k = 0 \rightarrow 75 \times 10^3$ cycles (see the leftmost panels). Figure 9c illustrates the temporal evolution of the uncertainty in the estimation of x_k within place $p_1^{(\mathcal{N})}$, with indication of the reference level when inspections are needed. This uncertainty is expressed and quantified through the DE. The observed drops in the sequence of DE values in Figure 9c correspond to the uncertainty reduction due to Bayesian learning when new measurements become available.

Observe from these results that there is a period required by the PPN model to learn from the data, which corresponds to the loading cycles in the interval $(0, 5 \times 10^3]$. After this learning period, not only does the precision of the prediction of x_k clearly improve with time (predicted values of x_k closer to data y_k), but also the uncertainty of the prediction gradually tends to diminish, which is numerical evidence of the Bayesian learning taking place in $p_1^{(\mathcal{N})}$. Figure 10a provides a plot of the history of the tokens visiting place $p_2^{(S)}$ during the overall period of evaluation $k = 0 \rightarrow 75 \times 10^3$, and indicates the sequence of activated inspections within that period. Note that at the beginning of the process (specifically the first 2,500 cycles), inspections are activated even when the uncertainty (DE) about x_k in $p_1^{(\mathcal{N})}$ is below the threshold value, as can be observed in Figure 9c. According to the PPN graph in Figure 8, these correspond to PIs that must be carried out until at least two measurements are available, whereupon $p_6^{(S)}$ is marked and the system switches from PI to CBI mode, as explained above. Note also from Figure 10a that a

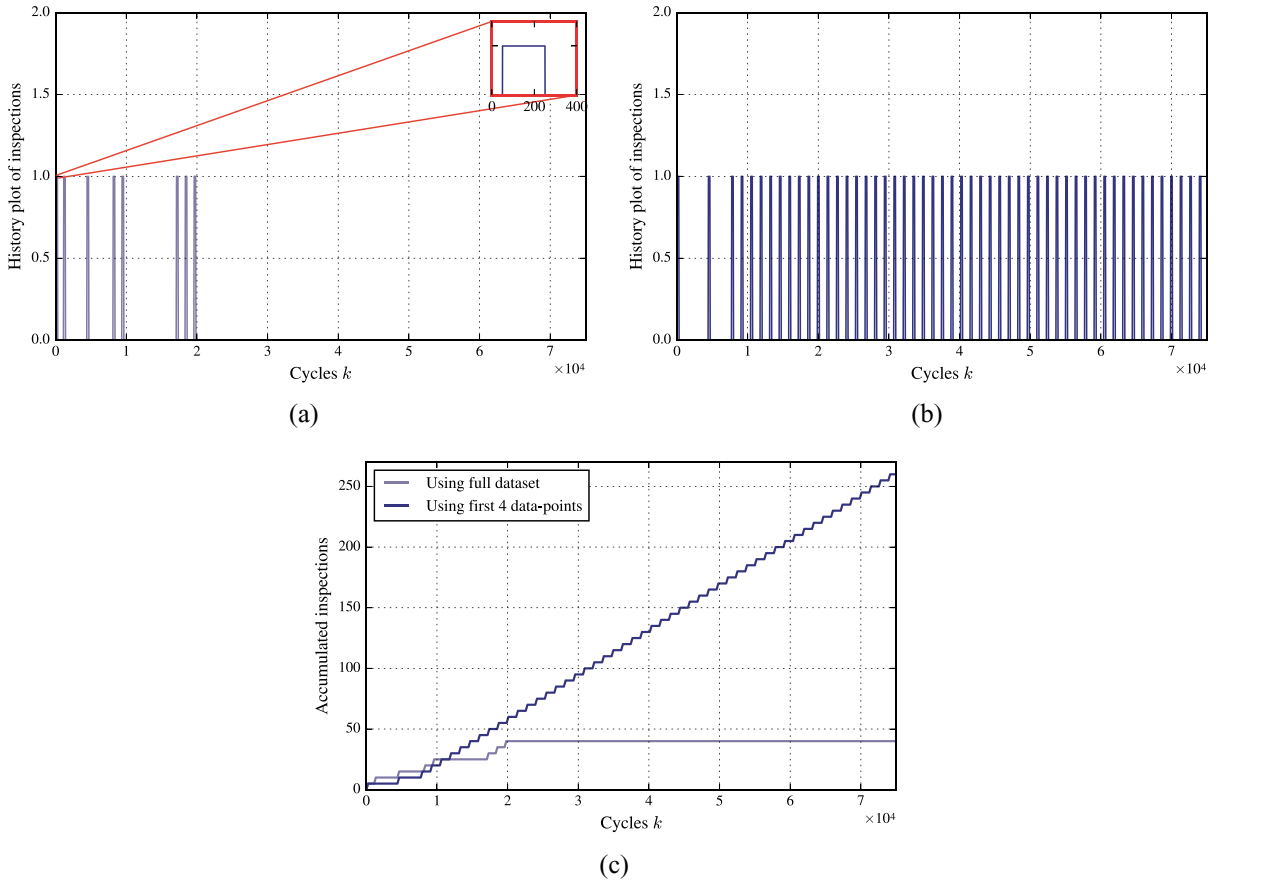


FIGURE 10 Plot of visiting tokens in place $p_2^{(S)}$ as a response of the PPN from Figure 8 using online data from (a) the data set (Y) shown in Table 1 and (b) only the first four data points arrived to the system. Panel (c) shows the accumulated number of tokens visiting place $p_2^{(S)}$ in both situations described before

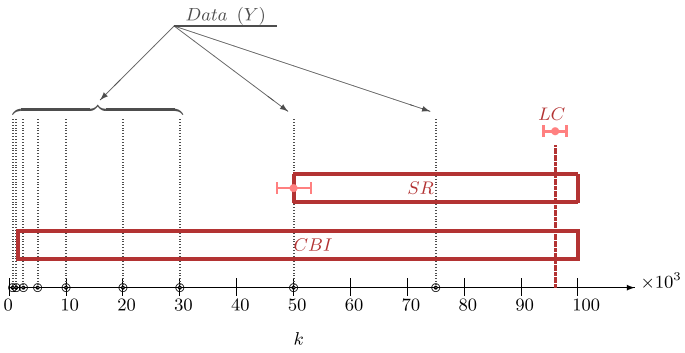


FIGURE 11 Intervals (expressed in cycles) for which the irreversible discrete-event states of the PPN shown in Figure 8 remain active. CBI: condition-based inspections, SR: speed restrictions, LC: Line closure

number of tokens visit place $p_2^{(S)}$ from cycle $k = 2.5 \times 10^3$ to about $k = 2 \times 10^4$, corresponding to inspection activities triggered because the uncertainty of the degradation variable x_k in this initial period passes the threshold several times, i.e. (DE $\geqslant -4.8$), activating t_2. After this initial period, the system identifies that no more inspections are needed. Observe that these results reveal that the PPN autonomously responds to the arrival of data through adaptation so that the sequence of discrete states is altered in response to the most up-to-date information from data Y. Further insight about the response of the PPN of this case study is provided through a schematic diagram of the sequence of irreversible discrete states activated during a period of evaluation $k = 0 \rightarrow 100 \times 10^4$, as shown in Figure 11. The results are provided after 200 independent runs of the PPN algorithm. The circled points in Figure 11 represent the cycles when measurement data become available. The error bars are to indicate the 25th–75th probability bands corresponding to the variation in activation of certain discrete states due to the independent runs of the PPN algorithm. This diagram gives the intervals of the cycles (represented using bars) for which each discrete state remains active within the period of evaluation.

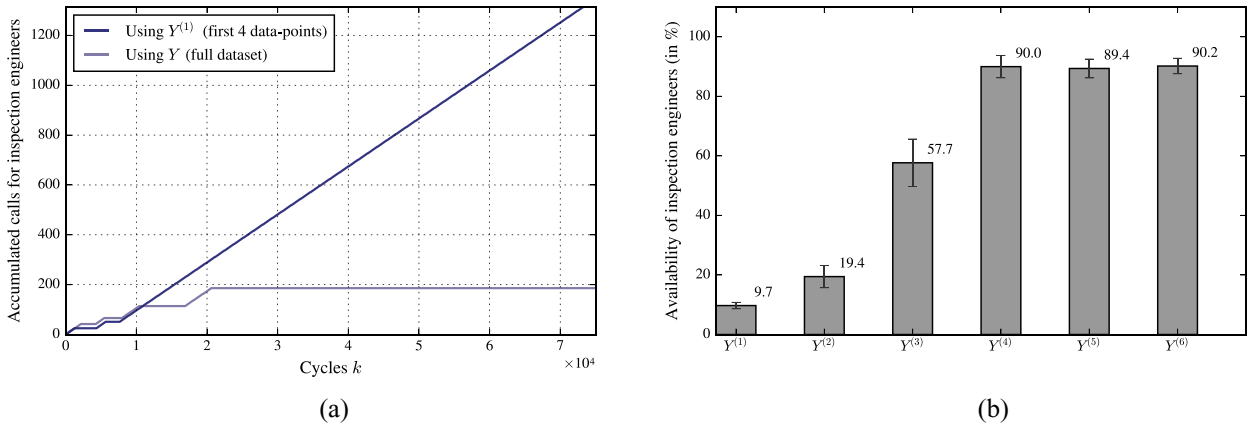


FIGURE 12 (a) Plot of total number of times where $p_7^{(S)}$ is empty. (b) Resulting average availability of track inspection engineers as a function of the number of data points used to train the PPN of Figure 8

4.4 | Discussion

In Section 4.3, a case study has been provided to illustrate the dynamics of PPNs, the different types of information that can be managed (uncertain information in confluence with information from discrete events), and how PPNs can sequentially learn from monitoring data. This section contains a discussion about the effectiveness of the proposed computational methodology in the context of the case study results. To this end, the PPN from the case study is comparatively evaluated by considering different subsets of data extracted from the data set in Table 1.

As a first exercise, the PPN is evaluated using Algorithm 1 by considering a data set $Y^{(1)}$ based on the first four data points shown in Table 1, i.e., the PPN only receives data up to 5×10^3 cycles. The results for the estimated degradation variable $x_k \sim f_k^{p_1}$ and its DE are provided in Figure 9 (see right-hand panels). Observe that, in this case, the uncertainty increases from $k = 5 \times 10^3$ loading cycles, when new measurements are no longer available, in comparison with the case shown in Figures 9a and c, respectively, where the overall data set is available. As a consequence, the PPN responds by continuously triggering CBIs from $k = 8 \times 10^3$, which is the load cycle when DE exceeds the threshold -4.8 (see Figures 10b and c). These results corroborate that the proposed model is able to adapt to the available monitoring information in such a way that the more uncertainty in the degradation estimation (e.g., due to low quality of the monitoring system), the more inspections required, and hence the higher the maintenance costs, provided that there exists a cost associated with each inspection.

In addition, a comparison between accumulated calls for inspection engineers over time is carried out by considering the reduced data set, $Y^{(1)}$, and Y, the complete data set. The interest in the accumulated amount of calls of engineers is because it is indicative of the availability of engineers for this particular example, such that the lower the calls for inspections, the higher the availability of engineers, hence lower inspection costs. To count the calls for engineers in each case, place $p_7^{(S)}$ in Figure 8 has been monitored over time of execution. The results, as shown in Figure 12b, are given in terms of total number of times that $p_7^{(S)}$ is empty. Note that during the first cycles (say $k \leqslant 1 \times 10^4$), the response of the PPN under both data sets is similar (if not equal) due to the PIs that are activated for the first cycles (recall that PIs are initially triggered until two data points arrive). After this initial period, the total number of triggered CBIs is clearly lower as a response of the PPN when using data set Y (recall Figure 10c), which implies lower calls for engineers, hence higher availability revealed through higher amount of tokens visiting $p_7^{(S)}$ during the runtime.

Finally, note that, from this standpoint, a natural research question arises about the optimum amount of data the self-adaptive expert system would require to minimize inspection costs. To answer such a question, the PPN shown in Figure 8 has been evaluated using six sets of data that have been named as $\{Y^{(j)}\}_{j=1}^{6}$, such that $Y^{(1)} \subset Y^{(2)} \ldots \subset Y^{(6)} = Y$, where $Y^{(1)}, Y^{(2)}, \ldots, Y^{(6)}$ correspond to data sets obtained by successively considering the first four, five, and finally, all data points from Table 1, respectively. Figure 12b shows the results obtained from 200 independent runs of the PPN algorithm for each data set considered. Observe that, in general, the more measurement data become available, the lower number of calls for inspections, and thus the higher the availability of engineers. This is a consequence of the Bayesian learning of the PPN from the data, which serves to control the uncertainty of the track settlement predictions so as to avoid triggering unnecessary inspections. Note also that there is no significant difference in terms of maximum availability of engineers when using data sets $Y^{(4)}$ to $Y^{(6)}$. This means that, in this particular case, taking $Y^{(4)}$ is optimal because the PPN

response is virtually the same in terms of triggered inspection actions, while avoiding unnecessary future track settlement measurements.

5 | CONCLUSIONS

This article presented a novel methodology to model self-adaptive expert systems by Bayesian learning of PPNs. The mathematical aspects behind PPNs along with the aspects relating to the learning procedure of PPNs have been provided using illustrative examples. As an application scenario, an engineering case study has been presented, which uses experimental data of railway track degradation to demonstrate how monitoring data and model-based knowledge about track degradation can be integrated within a self-adaptive expert system modeled using a PPN. The following are concluding remarks:

- Expert systems modeled using PPNs can manage uncertainty and also autonomously respond to the arrival of noisy data through adaptation.
- Bayesian model updating is shown to appear naturally as a particular case of the conjunction of states of information, which is one of the intrinsic operations of PPN execution semantics.
- A PPN-based railway expert system can operate autonomously or as a decision support tool allowing the appropriate managers and railway engineers to make better decisions. Future research steps in the context of this specific application include the consideration of other subsystems within the expert system model, such as signaling, electrification, switchings and crossings, etc.
- Building on this work, one desirable future research direction to enhance PPNs as self-adaptive expert systems for complex civil infrastructures is to explore PPN architectures that allow the modelng of foreseen intervention scenarios. Also, an additional challenge would be how to incorporate massive and heterogenous data (Thaduri, Galar, & Kumar, 2015; Thöns, 2018) from infrastructure monitoring within the PPN methodology.

ACKNOWLEDGMENTS

This work was supported by the Engineering and Physical Sciences Research Council (grant number EP/M023028/1), and also by the Lloyd's Register Foundation (LRF), a charitable foundation in the United Kingdom helping to protect life and property by supporting engineering-related education, public engagement, and the application of research. John Andrews is the LRF Director of the Resilience Engineering Research Group and also the Network Rail Professor of Infrastructure Asset Management at the University of Nottingham. The authors gratefully acknowledge the support of these organizations.

REFERENCES

Adeli, H., & Balasubramanyam, K. V. (1988). *Expert systems for structural design: A new generation*. Englewood Cliffs, NJ: Prentice Hall.

Ando, T. (1995). Majorization relations for Hadamard products. *Linear Algebra and Its Applications*, *223*, 57–64.

Andrews, J. (2012). A modelling approach to railway track asset management. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, *227*, 1–8.

Andrews, J., Prescott, D., & De Rozières, F. (2014). A stochastic model for railway track asset management. *Reliability Engineering & System Safety*, *130*, 76–84.

Arumlampalam, M., Maskell, S., Gordon, N., & Clapp, T. (2002). A tutorial on particle filters for on-line nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, *50*(2), 174–188.

Aursudkij, B., McDowell, G., & Collop, A. (2009). Cyclic loading of railway ballast under triaxial conditions and in a railway test facility. *Granular Matter*, *11*(6), 391.

Beck, J. (2010). Bayesian system identification based on probability logic. *Structural Control and Health Monitoring*, *17*(7), 825–847.

Beezer, R. A. (2007). *A first course in linear algebra*. Gainesville, FL: University Press of Florida.

Bencomo, N., & Belaggoun, A. (2013). Supporting decision-making for self-adaptive systems: From goal models to dynamic decision networks. In *Proceedings of International Working Conference on Requirements Engineering: Foundation for Software Quality*, pp. 221–236. Springer.

Brown, S., Brodrick, B., Thom, N., & McDowell, G. (2007). The Nottingham railway test facility, UK. In *Proceedings of the Institution of Civil Engineers-Transport*, Vol. *160*, pp. 59–65. Thomas Telford Ltd.

Chiachío, J., Chiachío, M., Prescott, D., & Andrews, J. (2017). A reliability-based prognostics framework for railway track management. In *Proceedings of the Annual Conference of the Prognostics and Health Management Society (PHM, 2017)*, pp. 396–406. PHM.

Chiachío, M., Chiachío, J., Prescott, D., & Andrews, J. (2016). An information theoretic approach for knowledge representation using Petri nets. In *Proceedings of the Future Technologies Conference 2016*, San Francisco, 6–7 December, pp. 165–172. IEEE.

Chiachío, M., Chiachío, J., Prescott, D., & Andrews, J. (2018). A new paradigm for uncertain knowledge representation by Plausible Petri nets. *Information Sciences*, *453*, 323–345.

Chiachío, M., Chiachío, J., Sankararaman, S., & Andrews, J. (2017). Integration of prognostics at a system level: A Petri net approach. In *Proceedings of the Annual Conference of the Prognostics and Health Management Society (PHM, 2017)*, pp. 376–388. PHM.

Chiang, W., Liu, K. F., & Lee, J. (2000). Bridge damage assessment through fuzzy Petri net based expert system. *Journal of Computing in Civil Engineering*, *14*(2), 141–149.

Cox, R. T. (1946). Probability, frequency and reasonable expectation. *American Journal of Physics*, *14*, 1.

Dahlberg, T. (2001). Some railroad settlement models - A critical review. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, *215*(4), 289–300.

Doucet, A., De Freitas, N., & Gordon, N. (2001). An introduction to sequential Monte Carlo methods. In A. Doucet, N. De Freitas, & N. Gordon (Eds.), *Sequential Monte Carlo methods in practice* (pp. 3–14). Berlin/New York: Springer.

Esveld, C. (1989). *Modern railway track*. Zaltbommel: MRT-Productions.

Flintsch, G. W., & Chen, C. (2004). Soft computing applications in infrastructure management. *Journal of Infrastructure Systems*, *10*(4), 157–166.

Higgins, C., & Liu, X. (2018). Modeling of track geometry degradation and decisions on safety and maintenance: A literature review and possible future research directions. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, *232*, 0954409717721870.

Hsieh, F.-S., & Lin, J.-B. (2014). Context-aware workflow management for virtual enterprises based on coordination of agents. *Journal of Intelligent Manufacturing*, *25*(3), 393–412.

Indraratna, B., Thakur, P. K., Vinod, J. S., & Salim, W. (2012). Semiempirical cyclic densification model for ballast incorporating particle breakage. *International Journal of Geomechanics*, *12*(3), 260–271.

Jaynes, E. (1983). *Papers on probability, statistics and statistical physics*. R. D. Rosenkrantz (Ed.). Dordrecht/Boston: Kluwer Academic Publishers.

Kastner, J., & Hong, S. (1984). A review of expert systems. *European Journal of Operational Research*, *18*(3), 285–292.

Krupitzer, C., Roth, F. M., VanSyckel, S., Schiele, G., & Becker, C. (2015). A survey on engineering approaches for self-adaptive systems. *Pervasive and Mobile Computing*, *17*, 184–206.

Lajnef, N., Rhimi, M., Chatti, K., Mhamdi, L., & Faridazar, F. (2011). Toward an integrated smart sensing system and data interpretation techniques for pavement fatigue monitoring. *Computer-Aided Civil and Infrastructure Engineering*, *26*(7), 513–523.

Lee, J., Liu, K. F., & Chiang, W. (1999). A fuzzy Petri net-based expert system and its application to damage assessment of bridges. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *29*(3), 350–370.

Li, X., & Lara-Rosano, F. (2000). Adaptive fuzzy Petri nets for dynamic knowledge representation and inference. *Expert Systems with Applications*, *19*(3), 235–241.

Looney, C. G. (1988). Fuzzy Petri nets for rule-based decision making. *IEEE Transactions on Systems, Man, and Cybernetics*, *18*(1), 178–183.

Mosegaard, K., & Tarantola, A. (2002). Probabilistic approach to inverse problems. In W. Lee, P. Jennings, C. Kisslinger, & H. Kanamori (Eds.), *International handbook of earthquake and engineering seismology, Part A*, Volume 81 of *International geophysics* (pp. 237–265). Amsterdam/Boston: Academics Press Ltd.

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

Paek, Y., & Adeli, H. (1990). Structural design language for coupled knowledge-based systems. *Advances in Engineering Software*, *12*(4), 154–166.

Petri, C. A. (1962). *Kommunikation mit Automaten* (PhD thesis). Institut für Instrumentelle Mathematik an der Universität Bonn.

Rus, G., Chiachío, J., & Chiachío, M. (2016). Logical inference for inverse problems. *Inverse Problems in Science and Engineering*, *24*(3), 448–464.

Schneeweiss, W. G. (2001). Tutorial: Petri nets as a graphical description medium for many reliability scenarios. *IEEE Transactions on Reliability*, *50*(2), 159–164.

Schofield, A., & Wroth, P. (1968). *Critical state soil mechanics*. New York: McGraw-Hill London.

Selig, E. T., & Waters, J. M. (1994). *Track geotechnology and substructure management*. London, UK: Thomas Telford.

Serral, E., De Smedt, J., Snoeck, M., & Vanthienen, J. (2015). Context-adaptive Petri nets: Supporting adaptation for the execution context. *Expert Systems with Applications*, *42*(23), 9307–9317.

Soleimanmeigouni, I., & Ahmadi, A. (2016). A survey on track geometry degradation modelling. In U. Kumar, A. Ahmadi, A. K. Verma & P. Varde (Eds.), *Current trends in reliability, availability, maintainability and safety* (pp. 3–12). Berlin/New York: Springer.

Tarantola, A. (2005). *Inverse problem theory and methods for model parameters estimation*. Philadelphia, PA: SIAM.

Tarantola, A., & Valette, B. (1982). Inverse problems = quest for information. *Journal of Geophysics*, *50*(3), 159–170.

Thaduri, A., Galar, D., & Kumar, U. (2015). Railway assets: A potential domain for big data analytics. *Procedia Computer Science*, *53*, 457–467.

Thöns, S. (2018). On the value of monitoring information for the structural integrity and risk management. *Computer-Aided Civil and Infrastructure Engineering*, *33*(1), 79–94.

Vidal, J. C., Lama, M., & Bugarín, A. (2012). Petri net-based engine for adaptive learning. *Expert Systems with Applications*, *39*(17), 12799–12813.

Vidal, J. C., Lama, M., Díaz-Hermida, F., & Bugarín, A. (2013). A Petri net model for changing units of learning in runtime. *Knowledge-Based Systems*, *41*, 26–42.

Wagner, W. P. (2017). Trends in expert system development: A longitudinal content analysis of over thirty years of expert system case studies. *Expert Systems with Applications*, *76*, 85–96.

Wang, J., Liu, X.-Z., & Ni, Y.-Q. (2018). A Bayesian probabilistic approach for acoustic emission-based rail condition assessment. *Computer-Aided Civil and Infrastructure Engineering*, *33*(1), 21–34.

Weston, P., Roberts, C., Yeo, G., & Stewart, E. (2015). Perspectives on railway track geometry condition monitoring from in-service railway vehicles. *Vehicle System Dynamics*, *53*(7), 1063–1091.

Yu, H.-S. (2007). *Plasticity and geotechnics*. New York/London: Springer Science & Business Media.

Zhang, Z., Wang, S., & Yuan, X. (2009). Advanced self-adaptation learning and inference techniques for fuzzy Petri net expert system units. In *International Conference on Artificial Intelligence and Computational Intelligence*, pp. 487–496. Springer.

Zhou, K.-Q., & Zain, A. M. (2016). Fuzzy Petri nets and industrial applications: a review. *Artificial Intelligence Review*, *45*(4), 405–446.

How to cite this article: Chiachío M, Chiachío J, Prescott D, Andrews J. Plausible Petri nets as self-adaptive expert systems: A tool for infrastructure asset monitoring. *Comput Aided Civ Inf*. 2018;1–18. https://doi.org/10.1111/mice.12427

APPENDIX A: PPN ALGORITHM

A pseudocode implementation of the PPNs is provided below as Algorithm 1.

Algorithm 1: PPN Algorithm

Inputs: N, ▷ {Number of particles} , $\mathfrak{M} = \langle \mathbf{P}, \mathbf{T}, \mathbf{F}, \mathbf{W}, \mathbf{D}, \mathscr{X}, \mathscr{G}, \mathscr{H}, \mathbf{M}_0 \rangle$, ▷ {Tuple of defining elements for the PPN}, $\left[\left(\mathbf{x}_0^{(n),j}, \omega_0^{(n),j}\right)\right]_{n=1}^{N}$ ▷ {Sample of initial marking $M(j)_0^{(\mathscr{M})}$, $j = 1, \ldots, n_p$ }

Outputs: $\mathbf{M}_k$

begin ($k \geqslant 0$):

for all $t_i \in \mathbf{T}$ **do**
  ▷ Transition firing
  Run the enabling rule (recall §2.3.2) and evaluate $u_{i,k}$, $v_{i,k}$
  ▷ Information exchange
  if $t_i \in \mathbf{T}^{(\mathscr{M})}$ **then**
    Obtain $f^{\bullet t_i}$ ▷ {Use Algorithm 3}
    Sample $\left[\left(\mathbf{x}_k^{(n),ii}, \omega_k^{(n),ii}\right)\right]_{n=1}^{N}$ from $f^{\bullet t_i} \wedge f^{t_i}$ ▷ {Use Algorithm 2 }
    for all $p_j^{(\mathscr{M})} \in \mathbf{P}^{(\mathscr{M})}$ **do**
      Sample $\left[\left(\mathbf{x}_k^{(n),ij}, \omega_k^{(n),ij}\right)\right]_{n=1}^{N}$ from $f^{t_i} \wedge f^{p_j}$ ▷ {Use Algorithm 2}
      Modify particle weights by $a_{ij}^- \in \mathbf{W}^{(\mathscr{M})}$: $\left[\left(\hat{\omega}_k^{(n),ij}\right) \leftarrow \left(v_{i,k} a_{ij}^- \omega_k^{(n),ij}\right)\right]_{n=1}^{N}$
      Modify particle weights by $a_{ij}^+ \in \mathbf{W}^{(\mathscr{M})}$: $\left[\left(\hat{\omega}_k^{(n),ii}\right) \leftarrow \left(v_{i,k} a_{ij}^+ \omega_k^{(n),ii}\right)\right]_{n=1}^{N}$
      Concatenate $\left[\left(\mathbf{x}_k^{(n),ii}, \mathbf{x}_k^{(n),ij}\right)\right]_{n=1}^{N}$ and renumber as $\left(\mathbf{x}_k^{(1),ij}, \ldots, \mathbf{x}_k^{(N),ij}, \ldots, \mathbf{x}_k^{(2N),ij}\right)$
      Concatenate $\left[\left(\hat{\omega}_k^{(n),ii}, \hat{\omega}_k^{(n),ij}\right)\right]_{n=1}^{N}$ and renumber as $\left(\hat{\omega}_k^{(1),ij}, \ldots, \hat{\omega}_k^{(N),ij}, \ldots, \hat{\omega}_k^{(2N),ij}\right)$
      Require $v_{i,k} = 1$
      Normalize weights: $\left[\omega_k^{(n),ij} \leftarrow \frac{\hat{\omega}_k^{(n),ij}}{\sum_{n'=1}^{N} \hat{\omega}_k^{(n'),ij}}\right]_{n=1}^{2N}$
      Resampling of N samples from $\left[\left(\mathbf{x}_k^{(n),ij}, \omega_k^{(n),ij}\right)\right]_{n=1}^{2N}$
      Set $\left[\omega_k^{(n),ij} \leftarrow 1/N\right]_{n=1}^{N}$
    end for
  end if
end for
▷ Marking evolution
for all $p_j^{(\mathscr{M})} \in \mathbf{P}^{(\mathscr{M})}$ **do**
  Evaluate $\gamma_k^{(j)}$ from Eq. (7)
  Concatenate $\left[\left(\mathbf{x}_k^{(n),j}, \mathbf{x}_k^{(n),ij}\right)\right]_{n=1,i=1}^{N,n_t}$ and renumber as $\left(\mathbf{x}_{k+1}^{(1),j}, \ldots, \mathbf{x}_{k+1}^{(N),j}, \ldots, \mathbf{x}_{k+1}^{N(n_t+1),j}\right)$
  Concatenate $\left[\left(\gamma_k^{(j)} \omega_k^{(n),j}, \omega_k^{(n),ij}\right)\right]_{n=1,i=1}^{N,n_t}$ and renumber as $\left(\hat{\omega}_{k+1}^{(1),j}, \ldots, \hat{\omega}_{k+1}^{(N),j}, \ldots, \hat{\omega}_{k+1}^{n_t(N+1),j}\right)$
  Normalize weights: $\left[\omega_{k+1}^{(m),j} \leftarrow \frac{\hat{\omega}_{k+1}^{(m),j}}{\sum_{m'=1}^{N(n_t+1)} \hat{\omega}_{k+1}^{(m'),j}}\right]_{m=1}^{N(n_t+1)}$
  Resampling of N samples from $\left[\left(\mathbf{x}_{k+1}^{(m),j}, \omega_{k+1}^{(m),j}\right)\right]_{m=1}^{N(n_t+1)}$
  Set $\left[\omega_{k+1}^{(n),j} \leftarrow 1/N\right]_{n=1}^{N}$
  Set $M_{k+1}^{(\mathscr{M})}(j) \leftarrow \left[\left(\mathbf{x}_{k+1}^{(n),j}, \omega_{k+1}^{(n),j}\right)\right]_{n=1}^{N}$
end for
for all $p_j^{(\mathscr{S})} \in \mathbf{P}^{(\mathscr{S})}$ **do**
  Set $M_{k+1}^{(\mathscr{S})}(j) \leftarrow M_k^{(\mathscr{S})}(j) + \sum_{i=1}^{n'_t} a'_{ji} u_{i,k}$, $a'_{ji} \in \mathbf{W}^{(\mathscr{S})}$
end for
Set $\mathbf{M}_{k+1}^{(\mathscr{M})} = \left(M_{k+1}^{(\mathscr{M})}(1), \ldots, M_{k+1}^{(\mathscr{M})}(n_p)\right)^T$
Set $\mathbf{M}_{k+1}^{(\mathscr{S})} = \left(M_{k+1}^{(\mathscr{S})}(1), \ldots, M_{k+1}^{(\mathscr{S})}(n_{p'})\right)^T$
$\mathbf{M}_{k+1} = \left(\mathbf{M}_{k+1}^{(\mathscr{M})}, \mathbf{M}_{k+1}^{(\mathscr{S})}\right)$
Set $\mathbf{M}_k \leftarrow \mathbf{M}_{k+1}$

APPENDIX B: PARTICLE APPROXIMATION OF CONJUNCTION AND DISJUNCTION OF STATES OF INFORMATION

In particle methods, a set of N samples $[\mathbf{x}^{(n)}]_{n=1}^{N}$ with associated weights $[\omega^{(n)}]_{n=1}^{N}$ is used to obtain an approximation for the required density function (e.g., $(f_a \wedge f_b)(\mathbf{x})$), as follows:

$$(f_a \wedge f_b)(\mathbf{x}) \approx \sum_{n=1}^{N} \omega^{(n)} \delta(\mathbf{x} - \mathbf{x}^{(n)}) \tag{B1}$$

where δ is the Dirac delta and $\mathbf{x}^{(n)} \sim (f_a \wedge f_b)(\mathbf{x})$. The particle weight $\omega^{(n)}$ represents the likelihood value of $\mathbf{x}^{(n)}$, and is representative of the plausibility of $\mathbf{x}^{(n)}$ when it is distributed according to $(f_a \wedge f_b)(\mathbf{x})$. It can be evaluated for the case of $\mathscr{X}$ being a linear space as follows (Chiachío et al., 2018):

$$\omega^{(n)} = \frac{f_a(\mathbf{x}^{(n)}) f_b(\mathbf{x}^{(n)})}{\sum_{n=1}^{N} f_a(\mathbf{x}^{(n)}) f_b(\mathbf{x}^{(n)})} \tag{B2}$$

A pseudocode implementation to obtain particles from the conjunction $(f_a \wedge f_b)(\mathbf{x})$ is provided as Algorithm 2.

Algorithm 2: Particle approximation of conjunction of states of information

Inputs: $N, f_a(\mathbf{x}), f_b(\mathbf{x})$ ▷ {number of particles and states of information}

Outputs: $\left[\left(\mathbf{x}^{(n)}, \omega^{(n)}\right)\right]_{n=1}^{N}$, where $\mathbf{x}^{(n)} \sim (f_a \wedge f_b)(\mathbf{x})$

Begin

Sample $\left[\left(\tilde{\mathbf{x}}_a^{(n)}, \tilde{\omega}_a^{(n)}\right)\right]_{n=1}^{N}$ from $f_a(\mathbf{x})$
Set $\mathbf{x}^{(n)} \leftarrow \tilde{\mathbf{x}}_a^{(n)}$, $n = 1, \ldots, N$
Set $\hat{\omega}^{(n)} \leftarrow f_b\left(\mathbf{x}^{(n)}\right)$, $n = 1, \ldots, N$ ▷ {unnormalised weights}
Normalise weights $\omega^{(n)} \leftarrow \frac{\hat{\omega}^{(n)}}{\sum_{n=1}^{N} \hat{\omega}^{(n)}}$, $n = 1, \ldots, N$

The particle approximation of disjunction of states of information can be evaluated by simply joining the particles from the component-wise density functions and affecting their particle weights using an appropriate normalizing constant so

as to obtain a bone fide density, as described in Algorithm 3. Note that the disjunction operation can be easily extended to the case of multiple states of information (e.g., $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})$), as follows (Mosegaard & Tarantola, 2002):

$$(f_1 \vee \cdots \vee f_m)(\mathbf{x}) = \frac{1}{\beta} \sum_{i=1}^{m} f_i(\mathbf{x}) \tag{B3}$$

The last expression can be approximated using particles through Algorithm 3 by aggregating samples from $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})$ and considering $1/m$ as the modifying constant for particle weights in steps 3 and 4.

Algorithm 3: Particle approximation of disjunction of states of information

Inputs: $N, f_a(\mathbf{x}), f_b(\mathbf{x})$ ▷ {number of particles and states of information}
Outputs: $\left[\left(\mathbf{x}^{(n)}, \omega^{(n)}\right)\right]_{n=1}^{N}$, where $\mathbf{x}^{(n)} \sim (f_a \vee f_b)(\mathbf{x})$

Begin

1: Sample $\left[\left(\tilde{\mathbf{x}}_a^{(n)}, \tilde{\omega}_a^{(n)}\right)\right]_{n=1}^{N}$ from $f_a(\mathbf{x})$
2: Sample $\left[\left(\tilde{\mathbf{x}}_b^{(n)}, \tilde{\omega}_b^{(n)}\right)\right]_{n=1}^{N}$ from $f_b(\mathbf{x})$
3: $\left[1/2\, \tilde{\omega}_a^{(n)} \leftarrow \tilde{\omega}_a^{(n)}\right]_{n=1}^{N}$ ▷ {Modify particle weights}
4: $\left[1/2\, \tilde{\omega}_b^{(n)} \leftarrow \tilde{\omega}_b^{(n)}\right]_{n=1}^{N}$ ▷ {Modify particle weights}
5: Concatenate $\left[\left(\tilde{\mathbf{x}}_a^{(n)}, \tilde{\mathbf{x}}_b^{(n)}\right)\right]_{n=1}^{N}$ and renumber as $\left(\tilde{\mathbf{x}}^{(1)}, \tilde{\mathbf{x}}^{(2)}, \ldots, \tilde{\mathbf{x}}^{(2N)}\right)$
6: Concatenate $\left[\left(\tilde{\omega}_a^{(n)}, \tilde{\omega}_b^{(n)}\right)\right]_{n=1}^{N}$ and renumber as $\left(\tilde{\omega}^{(1)}, \tilde{\omega}^{(2)}, \ldots, \tilde{\omega}^{(2N)}\right)$
7: $\left[\mathbf{x}^{(n)}, \omega^{(n)}\right]_{n=1}^{N} \leftarrow$ `resampling` $\left[\tilde{\mathbf{x}}^{(n)}, \tilde{\omega}^{(n)}\right]_{n=1}^{2N}$