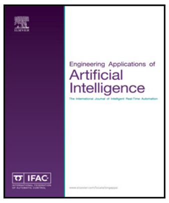
Engineering Applications of Artificial Intelligence 125 (2023) 106657

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



Causal reinforcement learning based on Bayesian networks applied to industrial settings

Gabriel Valverde [*], David Quesada, Pedro Larrañaga, Concha Bielza

Computational Intelligence Group, Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid, Campus de Montegancedo s/n, Boadilla del Monte, Madrid 28660, Spain

A R T I C L E I N F O

Keywords:
Reinforcement learning
Bayesian networks
Causality
Parameter learning
Dynamic simulators
Ordinary differential equations

A B S T R A C T

The increasing amount of real-time data collected from sensors in industrial environments has accelerated the application of machine learning in decision-making. Reinforcement learning (RL) is a powerful tool to find optimal policies for achieving a given goal. However, RL's typical application is risky and insufficient in environments where actions can have irreversible consequences and require interpretability and fairness. While new trends in RL may provide guidance based on expert knowledge, they do not often consider uncertainty or include prior knowledge in the learning process. We propose a causal reinforcement learning alternative based on Bayesian networks (RLBNs) to address this challenge. The RLBN simultaneously models a policy and takes advantage of the joint distribution of the state and action space, reducing uncertainty in unknown situations. We propose a training algorithm for the network's parameters and structure based on the reward function and likelihood of the effects and measurements taken. Our experiment with the CartPole benchmark and industrial fouling using ordinary differential equations (ODEs) demonstrates that RLBNs are interpretable, secure, flexible, and more robust than their competitors. Our contributions include a novel method that incorporates expert knowledge into the decision-making engine. It uses Bayesian networks with a predefined structure as a causal graph and a hybrid learning strategy that considers both likelihood and reward. This would avoid losing the virtues of the Bayesian network.

1. Introduction

The number of machine learning (ML) methods proposed for different business environments is increasing, and these methods are now being applied in industrial systems. The availability of cheaper sensors for information acquisition and the systematization and industrialization of real-time data collection and storage systems have contributed to the increase in these methods (Boyes et al., 2018). ML has proven to be effective in supporting decision-making based on predictions or pattern detection (Larrañaga et al., 2018). For example, tomato detection using the modified YOLOv3 framework (Lawal, 2021) showcases the application of ML in agricultural settings. Additionally, a multiscale convolutional network architecture is integrated into neuroscience research for EEG data (Roy, 2022). Once this milestone has been reached, adaptive tools that are functional in dynamic environments and prescriptive solutions that allow the automation of decision-making become issues of interest (Lepenioti et al., 2020).

In this context, critical factors for industrial processes, such as safety, efficiency, and reliability, should be considered. Often, they involve large-scale operations that can be dangerous if not handled properly. Ensuring safety is of the utmost importance to protect workers' health and well-being, while improving efficiency can result in increased productivity, reduced operating costs and higher profitability. Additionally, reliability is crucial as any unexpected downtime or failure in industrial processes can bring about substantial financial losses, disrupt supply chains, and damage customer relationships. Therefore, reliable processes and equipment to minimize the risk of breakdowns, avoid costly repairs, and maintain consistent performance.

In this regard, reinforcement learning (RL) and its base, dynamic stochastic programming, seem to be good tools. RL (Sutton and Barto, 1998) builds agents that are capable of intervening in a system through state-adjusted actions. Moreover, it can dynamically learn the optimal policy that maximizes an expected reward. Its suitability has been demonstrated in systems with high replication and low impact of incorrect actions (Mnih et al., 2013). It has also proven adaptable to previously intractable decision-making problems, such as those with high-dimensional states and action spaces (Mnih et al., 2015; Silver et al., 2018). However, RL application has substantial limitations in those situations, such as industry, where uncertainty is relevant (Li and Quiu, 2020), erroneous actions can have serious consequences, causal justification of the applied action is necessary or it is essential to understand the policy that is ultimately defined. One of the causes

* Corresponding author.
E-mail address: gabrielvalverdecastilla@gmail.com (G. Valverde).

https://doi.org/10.1016/j.engappai.2023.106657
Received 1 December 2022; Received in revised form 27 May 2023; Accepted 14 June 2023
Available online 10 July 2023

of these limitations is that the exploratory phase becomes unacceptable. In this phase, the agent selects and evaluates random actions (without limitations) to obtain a more general system vision. This carries significant risks in some scenarios. Furthermore, the lack of interpretability of neural network-based RL systems (Bai et al., 2020) is also a limitation when determining whether agent decisions are appropriate in a physical or scientific sense or when changes to the system can improve results.

Some authors have proposed alternative solutions to overcome these limitations. Causal reinforcement learning (CRL) (Gershman, 2017) addresses some of these problems. Learning is more controlled (limiting the exploratory phase) and more interpretability is achieved by introducing causality (Pearl, 1995). However, the causal alternative generally does not offer the possibility of including expert knowledge and does not use uncertainty prior to decision making. Thus, it does not identify situations in the environment or in the relationship of actions that may be too atypical to make a decision. Reinforcement learning from demonstration (RLfD) (Schaal, 1996) extracts values from test cases performed by other systems or agents to add them as a guide, either with supervised pretraining of the model that represents the policy or with a simulator (learned from the history) that functions as a reference for the agent. However, most existing methods only regard demonstrations as low-level knowledge instances, usually resulting in poor generalization capability and weak robustness performance. Interpretable reinforcement learning (IRL) (Silva et al., 2020) is particularly valuable because it helps reduce the RL search space, limiting the possibility of applying a bizarre action that can be harmful. Above all, it allows a better understanding of decision making. One natural way to include IRL is by using decision trees to represent the policy or the value function. However, decision trees used in IRL may be less effective in adjusting an optimal policy and may not limit the possibility of making decisions with insufficient information.

Several alternative approaches to RL for industrial decision-making problems have advantages and disadvantages. One alternative is expert knowledge-based RL, which involves using expert knowledge to define a Estimation of Distribution Algorithms (EDA) or a fuzzy-rule network which is then fine-tuned using RL (Treesatayapun, 2020; Du et al., 2022). This method has the advantage of leveraging the domain expertise of the decision-maker, resulting in a more interpretable and robust model. However, it can be difficult to obtain expert knowledge, and the resulting model may not generalize well to new situations. Data-driven probabilistic graphical models, in contrast, leverage RL algorithms to learn the structure and parameters of the graphical models from data. Nonetheless, their primary goal is to represent the underlying data distribution rather than to address a decision-making problem and derive a policy. Zhang et al. (2021). This method has the advantage of learning from experience and generalizing well to new situations. However, the resulting model may be less interpretable than one based on expert knowledge, and the quality of the model depends on the quality and quantity of available data. Another alternative is iterative using Bayesian methods for RL (Ghavamzadeh et al., 2015). This approach can result in a more flexible and adaptive model, but it may require more computational resources and may be more difficult to interpret. Other alternative, model-based RL involves building a model of the environment and using that model to make decisions (Sutton and Barto, 1998). In some situations, this approach can be more efficient than model-free RL, as the model can be used to plan ahead and avoid unnecessary exploration. However, building an accurate model can be difficult, and the resulting decisions are only as good as the model. Finally, hybrid RL methods combine model-based and model-free RL elements and other machine learning techniques to provide a more flexible and robust approach to decision-making (Song et al., 2022). For example, a hybrid RL method might use a data-driven Bayesian network for state estimation, combined with a model-based planner for decision-making. These methods can be more challenging than traditional RL approaches but can provide better performance and scalability in complex industrial environments.

Bayesian networks (Koller and Friedman, 2009) are an excellent complement to these last strategies. A Bayesian network is a probabilistic graphical model that compactly and intuitively represents probabilistic dependencies between random variables in a domain. Probabilistic graphical models maintain the virtues of other causal proposals and allow expert intervention, as well as the management and measurement of uncertainty (Dawid, 2020).

To apply the RL strategy to industrial decision-making problems, we propose a Bayesian network as a way to express a stochastic policy by which the agent makes decisions (Tedrake et al., 2004). The ability to represent its structure as a graph, the option of evaluating uncertainty, and its capacity to generate samples allow a better response to the challenges posed. Making decisions based on the optimization criterion of the problem (the reward) and discarding the actions with high uncertainty reduces the probability of failure. In addition, the explainability provided by Bayesian networks leaves scope to consider us to consider structural changes in the system to minimize the dynamic optimization task. Furthermore, the possibility of including previous knowledge, either by prior learning based on historical data or by expert knowledge of the network structure, minimizes the initial problem of taking risks by exploring the action space.

We propose different forms of application. The first alternative is to define the network structure based on expert knowledge. Once we have the network structure, we learn its parameters through RL based on the reward and the likelihood so that it does not lose representative and generative capacity; hence, the Bayesian network does not cease to be a representation of joint probability. In the second alternative, we learn the Bayesian network from samples of the state space. Given this Bayesian network, using RL, we learn the relationship between the state variables, and action variable and parameters. Another alternative is to learn both the structure and parameters iteratively. Following a standard learning process centred around rewards could lead to unrepresentative solutions. This can undermine the virtues of Bayesian networks as an inferential approximation of the optimal policy (Pearl, 1986). Therefore, it is essential to include likelihood as a goal of reinforcement learning algorithms to ensure success.

We verify the objectives have been achieved, applying the proposed alternatives to two examples: (1) a standard benchmark in the field of RL, CartPole (Barto et al., 1983), and (2) a simulated model of ordinary differential equations (ODE) (Harper, 1976) for the problem of industrial fouling (Bott, 1995). We propose a simplified representation of this industrial problem through a version of the ODEs defined by Quesada in Quesada et al. (2022). This will help us in the definition of the Bayesian network's causal structure in the first and second-developed alternatives.

For each of the examples, we will establish some initial parameters that define the environment behaviour. To determine the generalization and adaptation capacity after convergence, we consider other examples with different initial environment definitions. Noise is added in both cases to simulate an information collection system. Several simulations are carried out to ensure a more robust evaluation through different statistical tests.

Our contributions include the following: (1) A hybrid learning strategy that considers likelihood and reward for maintaining the representative and generative capacity of the Bayesian network. This strategy is applied to all our proposals. (2) A novel method that includes expert knowledge on the agent's decision-making engine based on using Bayesian networks with a predefined structure as a causal graph. (3) A comparative study of our proposals with different state-of-the-art alternatives, including a version in which everything is learned through RL (without expert knowledge) on the common benchmark CartPole. (4) An alternative approach, that involves using historical information to study the probability distribution of the states and connect the actions with RL. (5) A simulated test with differential equations that demonstrates the improved reward in the long term, avoidance of system failures in the initial steps, and adaptiveness of our proposed method. Additionally, we can measure the uncertainty of the decisions.

RLBNs offer several advantages over other RL approaches. They provide interpretability through their transparent structure, making them easily understandable to domain experts, while also being more secure than black-box models by handling uncertainty and noise in input data. RLBNs are also flexible and robust, allowing them to incorporate new knowledge and adapt to changing conditions in dynamic environments. The hybrid learning strategy used in RLBNs, which combines likelihood and reward, reduces the number of trials and failures needed to reach a solution. Therefore, decision-making becomes more efficient and effective.

Compared to CRL, which introduces causality for more controlled learning and higher interpretability, RLBNs can incorporate expert knowledge and use uncertainty before decision-making to identify atypical situations and relationships of actions. RLBNs also outperform both RLfD by having better generalization capability and robustness performance, and IRL by limiting the possibility of applying harmful actions, while allowing for a better understanding of decision-making.

This paper is organized as follows: In Section 1, we provide an overview of the topic and introduce the main focus of the paper, which is the combination of Bayesian networks and RL. Section 2 covers the preliminaries, providing a foundation of theoretical concepts and tools necessary to understand the content of the paper. This includes a discussion of Bayesian network learning (Section 2.1) and RL (Section 2.2). Section 3 presents the proposed approach, detailing the algorithm and its implementation. In Section 4, we describe the experimental validation of the solution, including testing on the CartPole problem and a simplified industrial fouling problem. Finally, Section 5 provides a conclusion that summarizes the key findings of the work and suggests future research directions.

2. Preliminaries

In this section, we provide an overview of the learning process of a Bayesian network from data and introduce basic RL concepts. Then, we introduce the novel approach, Reinforcement Learning based on Bayesian Networks (RLBN). RLBN is our contribution to the field. It involves incorporating expert knowledge into the decision-making engine using Bayesian networks with a predefined structure as a causal graph. Additionally, a hybrid learning strategy is proposed, considering both the likelihood and reward of avoiding losing the virtues of the Bayesian network. Overall, the section provides a comprehensive overview of the concepts related to BNRL, with an introduction to the contribution.

2.1. *Bayesian network learning*

A BN is a graphical representation of a probability distribution that consists of a directed acyclic graph and parameters (Koller and Friedman, 2009; Pearl, 1995). In recent years, BNs have gained significant importance due to their explanatory, prescriptive and inferential capacity as a tool in decision-making aid systems (McLachlan et al., 2020; Larrañaga et al., 2018). This has accelerated their theoretical development.

A BN represents the joint probability distribution of a set of random variables as $(\mathcal{G},\boldsymbol{\theta})$. $\mathcal{G}$ is a directed acyclic graph (DAG) defined as $\mathcal{G} = (\mathcal{V},\mathcal{E})$, where $\mathcal{V} = \boldsymbol{X} = \{X_1,\dots,X_n\}$ is a set of random variable, and $\mathcal{E} \subset \mathcal{V}\times\mathcal{V}$ is a set of arcs between variables. Each arc, from X_i to X_j, denotes a probabilistic dependence between the nodes. In this case, $X_i \in \boldsymbol{\Pi}_j$ is the set of parents of X_j. $\boldsymbol{\theta} = (\theta_1,\dots,\theta_n)$ is the set of parameters of the conditional probability distributions (CPDs) of each variable X_j given its parents, $\theta_j = P(X_j|\boldsymbol{\Pi}_j)$. If the variables are discrete, the parameters to be estimated are conditional probability tables (CPTs). The joint probability distribution factorizes to: $P(\boldsymbol{X}) = \Pi_{j\in\mathcal{I}_\mathcal{V}} P(X_j|\boldsymbol{\Pi}_j)$ where $\mathcal{I}_\mathcal{V} = \{1,\dots,n\}$.

The interpretation of a DAG as carrying conditional independence assumptions does not necessarily imply causality; in fact, a valid graph set can be constructed from independent variables with any ordering, not necessarily causal or chronological. However, interpreting DAG models built from data can lead to false assumptions about causality when assessing evidence or generating data (this sequencing of events does not have to represent the causal relationship of the variables). This interpretation explains why DAGs are sometimes used as solutions that respect the direction of time and causality. Therefore, in the first two contributions of this paper, we assume that the expert-defined Bayesian network structure is known and it is a causal graph. Indeed, it has been built bearing in mind cause–effect relationships between triplets of variables and not based only on conditional independence.

The growing emphasis of statistical approaches on vast amounts of data has prompted novel methods that enhance the learning of Bayesian networks (BNs) from data. Although the network structure and parameters can be estimated using data or domain knowledge, this research assumes that, in two out of three alternatives presented, a portion of the Bayesian network structure is already known and follows a causal graph.

2.1.1. *The parameter estimation*

The parameter estimation process of BNs (Ramoni and Sebastiani, 2001) aims to learn the CDP of each node (variable) given its parents (Koller and Friedman, 2009).

We assume that the structure $\mathcal{G}$ of the BN is given by an expert or that the conditional independence relationship it represents has been learned from data. Assume that we have $\mathcal{D} = \{\mathbf{x}^{(1)},\dots,\mathbf{x}^{(N)}\}$ a dataset of size N where $\mathbf{x}^{(i)} = (x_1^{(i)},\dots,x_n^{(i)})$ is the ith sample of the joint distribution (Ji et al., 2015). Given these assumptions, the parameters can be estimated using the maximum likelihood estimation (MLE) method: $\boldsymbol{\theta} = (\theta_{X_1|\Pi_1},\dots,\theta_{X_n|\Pi_n})$.

The likelihood function is defined as the conditional probability of the data $\mathcal{D}$ given the parameter $\boldsymbol{\theta}$. Assuming that there is global and local parameter independence (Spiegelhalter and Lauritzen, 1990): the samples are independent, and the parameter sets are disjoint (global decomposition), we have:

$$\begin{aligned} L(\theta|\mathcal{D}) = P(\mathcal{D}|\theta) = \prod_i^N L(\theta|x^{(i)}) &= \prod_i^N \prod_j^n L(\theta_{X_j|\Pi_j}|\mathbf{x}^{(i)}) \\ &= \prod_j^n L(\theta_{X_j|\Pi_j}|\mathcal{D}) \end{aligned} \tag{1}$$

Thus, we can estimate the parameters associated with each node independently: $\hat{\theta} = (\hat{\theta}_{X_1|\Pi_1},\dots,\hat{\theta}_{X_n|\Pi_n})$. This is relevant for one of our proposals. Considering Gaussian BNs (Heckerman et al., 1995; Shachter and Kenley, 1989), each node X_m with parents $\Pi_m = \{U_1,\dots,U_k\}$ follows a linear Gaussian CPD $P(X_m|\boldsymbol{u}) = N(\beta_0^{(i)} + \beta_1 u_1^{(i)} + \dots + \beta_k u_k^{(i)}; \sigma^{(i)2})$. Let us define each of the factors of Eq. (1) under the assumption that the data are complete:

$$\begin{aligned} l(\theta_m|\mathcal{D}) &= \sum_i^N log(L(\theta_m|\mathbf{x}_\mathbf{m}^{(i)})) \\ = \sum_i^N &\left[-\frac{1}{2}log(2\pi\sigma^{(i)2}) - \frac{1}{2\sigma^{(i)2}}(\beta_0 + \beta_1 u_1^{(i)} + \dots + \beta_k u_k^{(i)} - x^{(i)})^2\right] \end{aligned} \tag{2}$$

To apply the MLE method, we consider the gradient of the log-likelihood function for each parameter $\boldsymbol{\theta}$ and equating to 0, we arrive at a system of linear equations that is solvable by standard techniques (Koller and Friedman, 2009). As a result, we obtain the following expressions:

$$\begin{aligned} \beta_0 &= \mu_{X_m} - \Sigma_{X_m,\boldsymbol{\Pi}_m}\Sigma^{-1}_{\boldsymbol{\Pi}_m,\boldsymbol{\Pi}_m}\mu_{\boldsymbol{\Pi}_m} \\ \boldsymbol{\beta} &= \Sigma^{-1}_{\boldsymbol{\Pi}_m,\boldsymbol{\Pi}_m}\Sigma_{X_m,\boldsymbol{\Pi}_m} \\ \sigma^2 &= \Sigma_{X_m,X_m} - \Sigma_{X_m,\boldsymbol{\Pi}_m}\Sigma^{-1}_{\boldsymbol{\Pi}_m,\boldsymbol{\Pi}_m}\Sigma_{\boldsymbol{\Pi}_m,X_m} \end{aligned} \tag{3}$$

Recent works propose alternative learning algorithms to work with limited data (Benjumeda et al., 2019; Scanagatta et al., 2018). From the fact that we have a deterministic simulator of the effects of actions (ODEs or CartPole), MLE is efficient and precise in its estimations. The

Bayesian methodology with maximum a posteriori probability (MAP) can also be an excellent alternative to direct the search for optimal parameters. It gives flexibility in case of multicriteria objectives, for example, prediction in addition to adjustment.

In Bayesian statistics, parameters are assumed to be samples from a probability distribution. To identify the distribution, we start from the prior distribution over the parameters; the uncertainty, which does not measure MLE, will be reflected in the posterior distribution that takes into account this initial distribution and the data samples. The posterior of the parameter θ given the data $\mathcal{D}$ is as follows:

$$P(\theta|\mathcal{D}) = \frac{P(\mathcal{D}|\theta)P(\theta)}{P(\mathcal{D})} \tag{4}$$

where $P(\theta)$ is the a priori distribution, and $P(\mathcal{D}|\theta)$ is the probability of the data given the set of likelihood parameters. Finally, $P(\mathcal{D})$ is the probability of the data acting as the original data generation process.

The a priori distribution reflects the a priori knowledge of the problem, so it must be chosen appropriately. This distribution usually depends on the form of the CPD; in discrete cases, the Dirichlet distribution is commonly used because it facilitates the calculation of the posterior. In the linear Gaussian case, Gaussian inverse-gamma priors play a very similar role.

In the Bayesian case, we also introduce the assumption of the global independence of the parameters which allows a decomposition that facilitates the calculations (Spiegelhalter and Lauritzen, 1990):

$$P(\theta) = \prod_m P(\theta_{x_m}|\pi_m). \tag{5}$$

Therefore, the posterior is as follows:

$$\begin{aligned} P(\theta|\mathcal{D}) &= \frac{1}{P(\mathcal{D})} \prod_m \left[L_m(\theta_{x_m|\pi_m}|\mathcal{D}) P(\theta_{x_m|\pi_m}) \right] \\ &= \prod_m P(\theta_{x_m|\pi_m}|\mathcal{D}). \end{aligned} \tag{6}$$

Thus, the chosen parameters are the most likely ones according to this posterior distribution, known as the MAP; we can also use the distribution to sample or even generate the predictions and results considering the uncertainty at all times (Bishop and Nasrabadi, 2006).

2.1.2. Structure learning

Parameter learning starts from the assumption that the Bayesian network structure has been defined. However, in the alternatives proposed in this paper, this may be false or only partially true. Therefore, we briefly comment on the process of learning the structure, which is an NP-hard problem (Chickering, 1996a).

Given a dataset $\mathcal{D}$, we can define three main BN structure learning approaches: (1) the constraint-based method, which induces the graph from the results of conditional independence tests on data over triplets of variables, (2) the score-based method, which maximizes a score function relative to data, measuring the goodness of each structure, and (3) the hybrid method that combines both.

We focus on the second method, the score-based approach. There are many scores, among which the decomposable ones stand out. The Bayesian network structure quality can be calculated as the sum of the qualities of the subgraphs formed by each variable and its parents. Thus, decomposability is a convenient property for our proposal, allowing us to evaluate only local changes. The optimization problem can be stated as follows:

$$\mathcal{G}^* = \arg\max_{\mathcal{G}}(f(\mathcal{G},\mathcal{D})) = \arg\max_{\mathcal{G}}(\sum_{i=1}^{n} f(\mathcal{G}_i|\mathcal{D})) \tag{7}$$

where $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$ is the subgraph involving $\mathcal{V}_i = X_i \cup \Pi_i$, f is a decomposable score such as the K2 score, the Bayesian information criterion (BIC) or the Bayesian Dirichlet equivalent uniform (BDeu) score.

To solve the optimization problem, one of the local search methods called hill climbing is applied (Gámez et al., 2011). It goes through the search space starting from an initial solution and performs a finite number of steps. At each step, the algorithm only considers local changes, i.e., neighbouring DAGs, and chooses the one that results in the largest improvement in $f(\mathcal{G},\mathcal{D})$. The algorithm stops when there is no local change that produces a score improvement.

2.1.3. Probabilistic inference

Probabilistic inference estimates the probability distribution of unknown variables given the Bayesian network and the evidence (the value of the observed variables). This enables the inference of different variable values through probabilistic reasoning. In the Gaussian case, which is a closed family with a defined formulation, it is easy to do. There are exact methods that work well on simple structures and approximate ones that are more efficient on complex Bayesian networks. Since the examples presented in this work do not have a very complex structure, we will use the exact inference method without losing efficiency.

2.2. Reinforcement learning

Reinforcement learning (RL) is an artificial intelligence method linked directly to stochastic dynamic programming and decision theory. It can be seen as a paradigm that lies between supervised learning (because we have a reward that guides us) and unsupervised learning (because the learning process does not need labels against which to evaluate the result).

RL approach is to train an intelligent agent to make sequential decisions through trial and error in a system that returns a reward for each decision (each action applied, in RL terminology). The agent's goal is to build a strategy and determine the best course of action to achieve the objective set by the reward.

RL operation involves a sequential process, where the agent's actions on the system affect the system itself, creating feedback. Since our process involves sequential decision-making tasks, early actions may have long-term consequences for the overall goal. Sometimes, it may be better to sacrifice an immediate reward to gain a long-term one. The objective is to find a function, called a policy, that determines the best course of action at any time depending on the system's evolution.

The agent optimizes an action policy based on its interaction with the environment. This is done by means of stationary policies, which determine the action to be applied at time t by the state at t, generating what is called a Markov decision process (MDP). An MDP, $M = \{S, A, R, P, \gamma\}$, is defined by a state space S, an action space A, a reward function $r(s_t, a_t)$ at an instant t with state values s_t and action value a_t, a transition matrix $P = (P^a_{s\hat{s}})$ between states s and $\hat{s}$, where $P^a_{s\hat{s}} = P(s|\hat{s}, a)$ and a corresponding discount factor $\gamma \in [0,1]$ that gradually reduces the effect of previous actions. The policy $\pi : S \to A$ is defined as a stochastic distribution $\pi(a_t|s_t)$ for action a_t and state s_t. The history or trajectory is $\tau_t = s_0, a_0, s_1, a_1, \ldots, s_t, a_t$ where at is the sequence of states and actions over time. The environment is the setting where all the actions occur, and the goal is to select actions to maximize the total future rewards, which is equivalent by assumption to maximizing the discounted return $R_t = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$.

We can calculate the expectation of the discounted return under policy π:

$$\eta(\pi) = E_{s_0,a_0,\ldots}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t)\right] \tag{8}$$

where $a_t \sim \pi(a_t|s_t)$ and $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$.

To estimate the potential return of state s_t, the value function $V_p i(s_t)$ is defined as:

$$V_\pi(s_t) = E_{a_t,s_{t+1},\ldots}\left[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l})\right] \tag{9}$$

The value function is the expectation of the accumulated reward, penalized over time passes, given the current state, without knowing the course of action but applying a certain policy.

The quality, also called the state–action value, is expressed as:

$$Q_\pi(s_t, a_t) = E_{s_{t+1}, a_{t+1}, \ldots} \left[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l}) \right] \tag{10}$$

This is the value function, but it starts immediately with a current given action.

The advantage function $A_\pi(s_t, a_t)$ is the expected advantage of applying action a_t instead of using the action selected by π:

$$A_\pi(s_t, a_t) = Q_\pi(s_t, a_t) - V_\pi(s_t) \tag{11}$$

State types. The states can be divided into three main types:

- **Environment state** (S_t^e) - The environment state has a private representation and may not be visible to the agent. It is used to choose the next observation.
- **Agent state** (S_t^a) - The agent's internal representation, used by the agent to choose the next action.
- **Information state/Markov state** (S_t) - The useful information from history. This state will offer sufficient information to model the future. Then, history can be disregarded.

Environment. Environments are divided into fully observable environments and partially observable environments.

- **Fully observable environments (MDP)**: The agent directly observes the environment state: $S_t = S_t^a = S_t^e$.
- **Partially observable environments (Partially observable MDP)**: The agent indirectly observes the environment: $S_t^a \neq S_t^e$.

Reinforcement learning agent. The agent has several learning methods at its disposal, all of them acting on the system. The most interesting one form our perspective is learning through policies, either deterministic or stochastic.

RL agents can be categorized into the following types:

- **Value-based**: With no policy, the agent chooses actions greedily, based on state values.
- **Policy-based**: With no value function, uses a policy function to choose actions.
- **Actor-critic**: The agent uses both value and policy functions.

As regards the use of models to define the policies, two categories are defined: model free, if agents do not use a model and model-based, if they do.

We focus on agents with full observation capabilities that apply stochastic policies based on models. Our objective is to make decisions based on probability, defining the probabilistic space of the set of actions according to the state, selecting the most likely actions, and discarding those that are very risky and those with high uncertainty. This probability distribution will be determined by a BN that will house expert information.

2.2.1. Policy-based models

The main goal is to determine which action maximizes the reward at a state s. The policy parameterized by θ is indicated as $\pi(a|s, \theta) = P(A_t = a|S_t = s, \theta_t = \theta)$, which means that policy π is the probability of taking action a when the system is in state s and the parameters are $\boldsymbol{\theta}$.

Advantage. Policy-based alternatives entail evaluating actions based on states with different value or quality functions finally defining a policy based on those values. Policy-based models have better convergence properties because they are estimated directly and not through the intermediate step of the value function. They are also effective in large-scale continuous action spaces and yield stochastic policies that are more flexible and less predictable than deterministic ones.

Disadvantage. Policy-based methods generally converge to a local rather than a global optimum. Policy evaluation is often ineffective and varies widely. Our Bayesian network-based solution mitigates this by introducing, together with the reward, the likelihood in the policy learning process. This will restrict the space of actions to be explored.

Algorithms. Before dwelling on the different algorithms, we present a small sketch of what we want to achieve and the optimization problem to be solved by focusing the process on policies.

We will consider four different steps: 1. Define an objective function that determines how good a policy is. In our case, this function includes a likelihood. 2. Define the policies, that is, set the search space for distribution functions. 3. Choose a method that makes direct use of the policy in the environment to update the parameters. 4. Update and improve the parameters to maximize policy effectiveness, generating better returns and leading to lower risks.

One of the most interesting ways to deal with this problem would be to use gradient descent. For that purpose, we need a policy π_θ defined by parameters. This policy will be used to derive and build a function $J(\theta)$ to optimize the reward that depends on those parameters and is derivable.

$$J(\theta) = V_{\pi_\theta}(S_0) \tag{12}$$

where $V_{\pi_\theta}(S_0)$ represents the value of π_θ and S_0 is the first state. Maximizing $J(\theta)$ means maximizing $V_{\pi_\theta}(S_0)$. Using the policy gradient theorem (Sutton et al., 1999) with discrete spaces S and A, we have:

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a Q_\pi(s, a) \nabla \pi(a|s, \theta) \tag{13}$$

where $\mu(s)$ is the distribution function of the states s in the spaces where the policy used is π, and $Q_\pi(s, a)$ is the action value function given the policy. With continuous S and A:

$$\nabla J(\theta) = E[\nabla_\theta log(\pi_\theta(a|s)) A_{\pi_\theta}(s, a)] \tag{14}$$

The parameter update function is:

$$\theta = \theta + \alpha \nabla_\theta J(\theta) \tag{15}$$

where $\alpha \in [0, 1]$ is the learning rate.

There are different distribution spaces in which to find the optimal policy. Two very popular spaces are defined using the softmax function and the Gaussian distribution:

- Softmax policy: The softmax policy consists of a softmax function that converts the final policy obtained into a probability distribution. This means that it returns a probability for each possible action given the state as if it was a multiple classification problem.
- Gaussian policy: The Gaussian policy is used in the case of a continuous action space.

2.2.2. Trust region policy optimization

Trust region policy optimization (TRPO) (Schulman et al., 2015) is a policy-based algorithm that, instead of optimizing $J(\theta)$, defines its own optimization problem to minimize the drawbacks of policy-based algorithms by smoothing policy changes during learning. To that end, it defines an objective function $\mathcal{L}^{TRPO}_{\pi_{\theta_{t-1}}}(\pi_{\theta_t})$ in which the expectation of the advantage obtained by updating π_{θ_t} versus $\pi_{\theta_{t-1}}$ is measured. Moreover, a restriction is added so that the new π_{θ_t} does not move too far from the previous one:

$$\max_{\pi_{\theta_t}} \mathcal{L}^{TRPO}_{\pi_{\theta_{t-1}}}(\pi_{\theta_t}) = E\left[\frac{\pi_{\theta_t}(a|s)}{\pi_{\theta_{t-1}}(a|s)} A_{\pi_{\theta_{t-1}}}(s, a)\right] \tag{16}$$
$$\text{subject to } E\left[D_{KL}(\pi_{\theta_{t-1}}(\cdot|s), \pi_{\theta_t}(\cdot|s))\right] \leq \delta$$

where D_{KL} refers to the Kullback–Leibler divergence (Kullback and Leibler, 1951) and expresses the differences between two probability distributions. It measures the possible error from using distribution Q, when the original is P:

$$\begin{aligned} D_{KL}(Q \parallel P) &= \iint Q(\mathcal{H}, \mathcal{I}) log \frac{Q(\mathcal{H}, \mathcal{I})}{P(\mathcal{H}, \mathcal{I}|\mathcal{D})} d\mathcal{H} d\mathcal{I} \\ &= E_Q\left[log \frac{Q(\mathcal{H}, \mathcal{I})}{P(\mathcal{H}, \mathcal{I}|\mathcal{D})}\right] \end{aligned} \tag{17}$$
$$= E_Q[log Q(\mathcal{H}, \mathcal{I})] - E_Q[log P(\mathcal{H}, \mathcal{I}, \mathcal{D})] + log(P(\mathcal{D}))$$

Eq. (16) is equivalent to optimizing the following function by appropriately choosing the weight parameter β:

$$\max_{\pi_{\theta_t}} \mathcal{L}^{KL}_{\pi_{\theta_{t-1}}}(\pi_{\theta_t}) = E\left[\frac{\pi_{\theta_t}(a|s)}{\pi_{\theta_{t-1}}(a|s)} A_{\pi_{\theta_{t-1}}}(s,a) - \beta D_{KL}(\pi_{\theta_{t-1}}(\cdot|s), \pi_{\theta_t}(\cdot|s))\right] \tag{18}$$

Our solution is inspired by this strategy.

3. Bayesian network reinforcement learning

Our proposal stands out from the rest of the approaches since it leverages the combination of expert knowledge and Bayesian networks with a predefined causal graph, using a hybrid learning strategy that considers both likelihood and reward. This approach not only enhances the interpretability and security of the model but also provides a more robust and efficient decision-making process. In contrast, other RL algorithms have higher variability in their decision-making, often requiring more trials and displaying outliers. This is especially evident in industrial cases where safety, efficiency, and reliability are critical factors. Overall, our proposed approach showcases the advantages of RLBNs in solving industrial problems, such as fouling, with standard parameters, and highlights the importance of incorporating expert knowledge into the learning process.

In our approach, we work with policy-based models. Specifically, we will use Monte Carlo policy gradient (Williams, 1992) alternative, where the policy function is the BN itself. We believe that the use of a BN will significantly reduce the limitations of the Monte Carlo alternative: the risk of high variance of the gradient and the need for many interactions with the environment.

By introducing the likelihood L, defined from the history of trajectories τ in the policy evaluation function, we estimate the parameters of the BN, as well as the structure in the two proposals in which it is not fixed. This does not affect the theory that guarantees convergence and helps to stabilize the gradient.

However, to ensure rapid coverage and few jumps when working with the gradient, we establish a selection criterion for actions applied based on probability, discarding those actions that do not fit probabilistically to the sample of the environment (the evidence).

Thus, our proposal is to define the policy, employing a BN that is trained to find the actions with the highest expected reward but also aims to represent the collected data with the highest plausibility. Therefore, the agent will use the BN in two ways when making decisions: as a policy and as a knowledge filter that allows to discard low probability actions in the probabilistic space represented by the BN generated by the samples. Next, we introduce the agent's decision process when receiving information of the state S_t. Then, the policy (the BN) is updated based on these decisions, introducing the three different versions of learning and updating the BNs that we have defined. Finally both phases are combined in a single algorithm.

The agent's decision process starts from a given BN model that represents the environment's probability distribution, including the action space. In the initial stages, where the exploratory phase has not been carried out, we can start from a BN with a practically random architecture and parameters. Under this assumption, the agent (1) receives information S_t from the environment, (2) evaluates the probability of the evidence received using the BN, and (3) checks that it is a known event, meaning the probability is below a certain preset threshold δ, (4.1). If the probability is too far below the threshold, the agent does not act, as there is not enough knowledge for making a decision. (4.2) Otherwise, the agent uses the BN, taking S_t as evidence and estimating the most likely a_t. Then, (5) the agent evaluates the uncertainty regarding the action. (5.1) If the uncertainty is too high, the agent does not apply the action; (5.2) otherwise, it takes the most likely action.

The fact that the model does not initially represent the probabilistic space may lead to several iterations in which the agent does not make decisions because the probabilities are too low for steps 3 and 5. The agent makes exploratory decisions in which it performs actions with a high risk of being wrong. Therefore, we propose three novel ways of using Bayesian networks as a policy definition engine with different levels of prior knowledge regarding the probability distribution of the environment variables and the action space. Prior knowledge reduces the training time of the network before convergence to a good policy, as well as the time of the initial exploratory phase. Moreover, it makes the uncertainty assessment process more valuable. Here below, we present each of three novel proposals from the highest to the lowest level of knowledge.

In the first option, we have an overview of the system that shows the conditional independence structure between the state variables and the action variable; that is, we know the graph structure of the Bayesian network. We call this alternative RLBN expert ($RLBN_{EXP}$) because this information is assumed to come from expert knowledge, in some cases even from systems of equations or physical models representing the system. Alternatively, there are situations where the expert information is partial, such as when the conditional independence relationship between the state variables is well known but the effects that different actions can have on them are not, i.e.; we have only a part of the BN network structure. We call this RLBN semiexpert ($RLBN_{SEMIEXP}$). Finally, we may face a situation where there is no prior information, we know neither the network structure nor its parameters. For this alternative in which we learn everything from the sequence of actions and events, wholly based on data, we call it RLBN based in data ($RLBN_{BDATA}$).

The learning of both the structure and parameters is based on the evaluation function of the policies that contains the likelihood of the stored history. The parameters are learned by the descending gradient of the error function Eq. (15), and the structure is learned by algorithms that optimize a measure.

The learning process is based on that of the TRPO performs, replacing the KL of Eq. (17) with probability. Recall that we are working with π_θ which is the policy and, in turn, the factored probability distribution that defines the Bayesian network $(\mathcal{G}, \theta)$. We consider that we want to find a new θ. Our objective function to optimize is:

$$\max_{\pi_{\theta_t}} \mathcal{L}^{RLBN}_{\pi_{\theta_{t-1}}}(\pi_{\theta_t}) = E\left[\frac{\pi_{\theta_t(a|s)}}{\pi_{\theta_{t-1}}(a|s)} A_{\pi_{\theta_{t-1}}}(s,a)\right] \tag{19}$$

subject to $L(\theta_t|\tau) = Pr(\tau|\theta_t) \geq \delta$

where L is the likelihood function, $\mathcal{L}^{RLBN}_{\pi_{\theta_{t-1}}}(\pi_{\theta_t})$ the function to optimize, and δ is the minimum likelihood bound on the history of actions and states covered thus far. In this case, τ represents the data sample to emphasize the sequential character and the trajectory form of the data.

By setting a proper β, solving Eq. (19) is equivalent to solving:

$$\max_{\pi_{\theta_t}} \mathcal{L}^{RLBN}_{\pi_{\theta_{t-1}}}(\pi_{\theta_t}) = E\left[\frac{\pi_{\theta_t}(a|s)}{\pi_{\theta t-1}(a|s)} A_{\pi_\theta}(s,a) + \beta log L(\theta_t|\tau)\right] \tag{20}$$

This means that when faced with policies with similar reward estimates, we select those that have a greater likelihood. The beta parameter plays a crucial role as it influences the structure of the objective function in the RLBN algorithm. Particularly in uncertain scenarios, we meticulously determine the value of the beta parameter to strike a balance between exploration and exploitation. The specific choice of beta depends on the problem domain and the desired behaviour of the RLBN. For the $RLBN_{EXP}$ case, the process is already complete since we consider the structure fixed.

In the case of $RLBN_{SEMIEXP}$ and $RLBN_{BDATA}$, the careful selection of beta is crucial to ensure the desired behaviour of the RLBN. The search domain for beta is defined by the expert knowledge that contributes its experience in determining the importance of each criterion in the optimization problem. We consider the specific problem domain and adopt a systematic approach through experimentation and analysis. Multiple runs of the RLBN algorithm are conducted with varying beta values, allowing for an evaluation of performance and robustness across different scenarios. After that, the process remains the same until reaching the final point before updating the parameters. Once the parameter are updated, (1) we compare the performance of the distributions $\pi_{\theta t-1}$ and π_{θ_t} on the dataset by calculating their likelihood. (2) If the difference between the distributions based on likelihood is very high, we can assume that the BN structure has also changed and we apply a structure learning algorithm, in our case, hill-climbing using the BIC as score. Algorithm 1 shows the pseudocode of the process.

Algorithm 1: RLBN: Estimation of policy with BN

```
 1: Input:
 2: $\pi_\theta \leftarrow$ Initial probability model. A Bayesian network.
 3: $\mathcal{G} \leftarrow$ BN graph structure.
 4: $f \leftarrow$ structural score function.
 5: $u_{\mathcal{G}} = \begin{cases} \text{True} & \text{if the graph is updated} \\ \text{False} & \text{otherwise} \end{cases}$
 6: $\gamma \leftarrow$ threshold parameters
 7: Upgrade $\tau_t$ trajectories with new state $s_t$
 8: Output: $\tau_{\pi_\theta}$ and $\pi_\theta$ modelled by $(\mathcal{G}, \theta)$ Bayesian network
 9: for $l = 1, 2, \ldots N$ do
10:   while $s_t$ does not terminate do
11:     Collect $\tau_{\pi_{\theta_t}}$
12:     Estimate $A_{\pi_{\theta_t}}(s_t, a_t)$ based on Eq. (11) using $\tau_{\pi_{\theta_t}}$
13:     Calculate $L(\theta_t | \tau_{\pi_{\theta_t}})$
14:     Update $\theta_t \leftarrow \arg\max(L^{RLBN}_{\pi_{\theta_{t-1}}}(\pi_{\theta_t}))$ Eq. (20)
15:     if $(u_{\mathcal{G}} = True \ \& \ \frac{L(\theta_{t-1}|\tau_{\pi_{\theta_t}})}{L(\theta_t|\tau_{\pi_{\theta_t}})} \leq \gamma)$ then
16:       Apply hill climbing based on $f$ to upgrade $\mathcal{G}$
          Clear buffer $\tau_{\pi_{\theta_t}}$
17:     end if
18:     $a_t = \arg\max(\pi_{\theta_t})$
19:     $s_t \leftarrow s_{t+1}$
20:   end while
21: end for
```

In Algorithm 1, we see the step-by-step process to build the proposed solutions. We start with an initial policy as a Bayesian network, π_{θ_0}, and we also explicitly need its structure in the form of a graph $\mathcal{G}$. If we do not have prior information on the estimate and therefore cannot provide an initial probabilistic model, we start by learning a BN from a set of historical data stored thus far for the trajectory τ_0. If this information does not exist or is insufficient, we start with an initial BN defined randomly, taking into account the domain of the different variables that are part of the problem. We choose the score function f that we will use when evaluating the structure of the network since we use the hill climbing algorithm. Later, we define two parameters: N, the number of iterations and $u_{\mathcal{G}}$, a flag that indicates whether we want to update the structure defined by the network (it is false, for example, when we work with the expert alternative $RLBN_{EXP}$, where we already know the structure in advance). On Line 5, γ is the bound used to determine whether the variation in likelihood $L(\theta_t|\tau_{\pi_{\theta_t}})$ due to the modification of the parameters is high enough to propose an update of the structure on Line 6.

Once the input parameters are defined, we collect the state of the system s_t which we add to the memory or data buffer in which we have stored τ_t (at the initial times, it could be empty). After this step, Line 11 of Algorithm 1, begins the process of learning the optimal policy and selecting the optimal action to apply. We start by estimating the advantage function $A_{\pi_{\theta_t}}(s_t, a_t)$ based on Eq. (11) using $\tau_{\pi_{\theta_t}}$ and the computational methods defined in Schulman et al. (2015) on Line 12 of Algorithm 1. We next compute the likelihood $L(\theta_t|\tau_{\pi_{\theta_t}})$ of the BN parameters θ_t on the newly updated trajectory data. Once A and L are calculated, we can identify the maximum argument of $L^{RLBN}_{\pi_{\theta_{t-1}}}(\pi_{\theta_t})$ and update the parameter θ_t by Eq. (20). This involves maximizing the objective function in Eq. (20), using the TRPO optimization algorithm. By considering the advantage estimates and the likelihood of the updated parameters given the observed data, the optimization process adjusts the policy parameters to enhance the expected return.

Now we have two parameters θ_t and θ_{t-1}, two probability distributions π_{θ_t} and $\pi_{\theta_{t-1}}$, two policies, and two BNs with the same structure in Line 14 of Algorithm 1. If the structure update parameter is $u_{\mathcal{G}} = True$, we compare their likelihoods, and if the difference is large enough given the trajectory, we update the structure using the chosen score function f and the hill climbing algorithm on Line 15 of Algorithm 1. Next, we eliminate the initial historical values of the trajectory so that it does not distort the next evaluations generated by the new network structure. We apply the optimal action a_t, defined as the most likely action determined by the BN given the state s_t as evidence, and we update the next state. This process is repeated until a terminal state is reached or the maximum number of iterations is reached.

3.1. Setting the beta parameter in uncertain scenarios

The beta parameter plays a crucial role as it influences the structure of the objective function in the RLBN algorithm. Particularly in uncertain scenarios, we meticulously determine the value of the beta parameter to strike a balance between exploration and exploitation. The specific choice of beta depends on the problem domain and the desired behaviour of the RLBN.

To address uncertainty, we adopt a systematic approach through careful experimentation and analysis. Multiple runs of the RLBN algorithm are conducted with different beta values, allowing us to evaluate performance and robustness across various scenarios. By comparing results and analysing statistical metrics such as mean performance, variance, and confidence intervals, we gain insights into the impact of different beta values on the behaviour of RLBN.

Moreover, sensitivity analyses are performed to understand the influence of the beta parameter on convergence speed, exploration–exploitation trade-off, and the algorithm's ability to handle uncertainties during the learning process. These analyses assist in identifying an appropriate range or specific value of beta that leads to optimal performance, stability, and adaptability in the presence of uncertainty.

By carefully selecting and tuning the beta parameter, we ensure that the RLBN approach can effectively adapt to uncertain environments, providing reliable decision-making capabilities, and achieving desired learning outcomes.

4. Experimental validation

In this section, we propose different experiments to evaluate the solution performance. We will include one of the standard examples in the literature: CartPole. It presents a simplification of an industrial problem to validate the success of introducing a set of expert knowledge as a network structure. This is relevant for industrial problems where a deterministic ODEs model represents the observable stochastic system.

The experiments were performed on a PC with a 2.60 GHz Intel(R) Core(TM) i7-9750H processor, 32 GB of RAM, and 2024 GB of hard disk space, running Ubuntu 16.04.

4.1. CartPole

The CartPole problem (Fig. 1) is well-known in the RL and optimal control literature. It is a cart with a pole connected at one of its ends, similar to a pendulum (parameters such as the weight and length of the pole are configurable). The cart can be moved from left to right, and the pole rotates based on the movement. The goal is to keep the pole upright for as long as possible. A +1 reward is earned for each

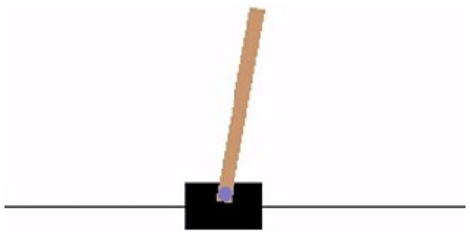


Fig. 1. The CartPole task (Nagendra et al., 2017).

time the pole remains upright, and the process stops if the pendulum is more than 15 degrees from vertical or the cart moves too far from the centre.

The state vector of the CartPole task is $[x, \tilde{x}, \phi, \tilde{\phi}]$, which represents the position and velocity of the cart, and the angle and angular velocity of the pole. The space of actions is continuous, since the actions represent the forces exerted to move the cart to one side or the other; the force is in the real interval $(-1, 1)$.

4.2. *Fouling problem definition*

The chosen experiment is a simple example of the fouling process in an industrial oven. This problem occurs in different industrial areas, such as the steel industry. Formally, it corresponds to a nonstationary time series that, by including human interventions, in the form of shutdowns or cleaning, due to anthropogenic factors such as shutdowns or cleaning, generates a seasonal structure and separate nonhomogeneous cycles. Our goal is to determine the optimal intervention strategy to improve furnace performance (Bott, 1995; Brusakov, 1971; Copisarow, 1945).

More specifically, we have analysed the problem of fouling that occurs in some smelting furnaces in the metallurgical industry. These furnaces consist of a container heated to raise the temperature, generally by gas heaters. Inside, the raw material is melted to separate it into cast iron and scrap. The chemical reaction generates carbonized residues that accumulate on the walls of the container, creating an insulating layer. This insulating layer becomes thicker and forces more energy and heat to be applied to the outside walls. Raising the outdoor temperature becomes more expensive and can cause damage when approaching the melting temperature of the container material. A diagram of the process is illustrated in Fig. 2 and the steps of the process, as it approaches possible breakdown, are shown in Fig. 3.

In the oven, the reaction receives liquid pig iron, scrap, additives, fluxes (Fe-Si), and oxygen and returns the final material together with sedimentary compounds and accumulated fouling as insulation. The operators can intervene by varying the temperature, changing the mixing percentages or stopping to clean up the accumulated fouling.

The optimization problem is to find the best dynamic intervention strategy to achieve the highest system performance. A system of differential equations is used as a simplified deterministic representation of this dynamic chemical reaction. If the operation of the oven obeyed in the standard conditions that are axiomatic of this representation, the problem would simply be solved by a system of equations generating the trajectories derived from an action at a given moment. However, uncertainty and probability are found in real systems. For us the system of differential equations will suffice as the representation of the expert knowledge of the problem and as the basis of the stochastic simulation performed by the agent to validate decisions. To make this case more straightforward, we propose a simplification (explained in Section 4.2.1), defined with differential equations.

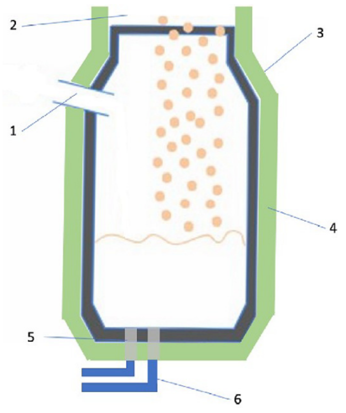


Fig. 2. Mixing oven. 1. Loading mouth: place where the reagents are introduced. 2. Filter mouth: area through which particles that can accumulate in the form of dirt can be cleaned from the surface. 3. Case: protective surface against external agents. This zone has low melting temperature. 4. Refractory: insulation layer, heat preserving. This material has high melting temperature. 5. Porous zone: becomes denser due to the effect of fouling and does not allow heat to penetrate. 6. Heaters: elements that generate heat to raise the internal temperature.

4.2.1. *Experiment: Simplification of the fouling problem*

The simplification in Fig. 4 illustrates a tube-shaped furnace that simplifies the input mix and does not consider the ability to intervene in the mix. Once again, the goal is to find the policy that defines the best strategy in terms of long-term performance.

The deterministic formal representation of the simplified version is a set of ODEs. In this case we will use the system of ODEs introduced by Quesada in Quesada et al. (2022) and described below. The following variables make up the ODEs that represent the environment: fouling level or thickness of the insulating layer $S_c(t)$, temperature of the fluid material inside $T_1(t)$, outside temperature in the tube walls $T_2(t)$, burner temperature $T_3(t)$, flow of new fluids entering the system $Q_{in}(t)$, sedimentary capacity (as a proportion of particles that can form sediments) $C_a(t)$, and product value generated $P(t) = e^{(\alpha T_1(t))}$, where α is a correction factor of the temperature measurement unit to properly represent the value of the product in monetary units, T_{min} and T_{max} establish the physical temperature limits of the burners through which the material flows.

Note that the outside and inside temperatures directly affect the sediment accumulation rate, in the same way as the amount of C_a included in the input compound:

$$\begin{aligned}\frac{\partial S_c}{\partial t} &= A_1 k_1 C_a \\ \frac{\partial C_a}{\partial t} &= -A_4 k_2 C_a\end{aligned} \tag{21}$$

$$\begin{aligned}k_1 &= A_2 e^{\frac{-A_3}{RT_1}} \\ k_2 &= A_2' e^{\frac{-A_3}{RT_1}}\end{aligned} \tag{22}$$

$$\rho_1 C_{p_1}\left(\frac{\partial T_1}{\partial t} - A_5 Q_{in} \nabla T\right) = k_1 \nabla^2 T_1 + f_1(T_1, T_2) \tag{23}$$

$$Q_{in} = vol\,\frac{\pi(2r)^2}{4} \tag{24}$$

$$f_1(T_1, T_2) = \frac{A_6}{S_c}(T_2 - T_1) \tag{25}$$

$$\rho_2 C_{p_2}\left(\frac{\partial T_2}{\partial t}\right) = k_2 \nabla^2 T_2 + f_2(T_1, T_2, T_3) \tag{26}$$

$$f_2(T_1, T_2, T_3) = \frac{A_6}{S_c}(T_1 - T_2) + A_7(T_3^4 - T_2^4) \tag{27}$$

$$T_3 = T_{min} + \frac{1}{1 + e^{-m_c}} T_{max} \tag{28}$$

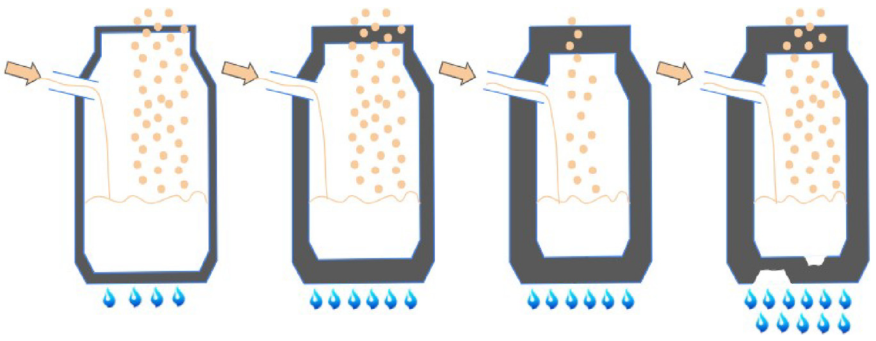


Fig. 3. Process outline. Step by step accumulation of fouling (outer grey layer) as reactions occur without intervention. It is necessary to increase the temperature at the risk of breaking the casing.

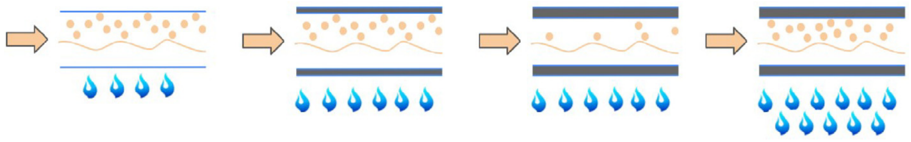


Fig. 4. Simplification of fouling.

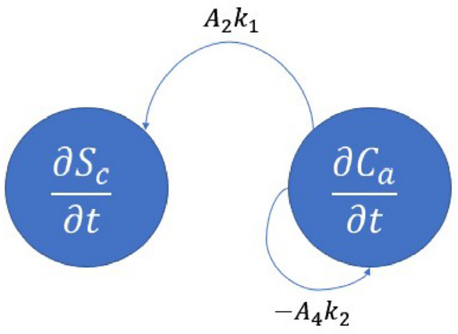

A_2k_1
$\frac{\partial S_c}{\partial t}$
$\frac{\partial C_a}{\partial t}$
$-A_4k_2$

Fig. 5. Relationship between S_c and C_a.

Table 1
The configuration of parameters for the different cases under study. We denote by $param_i$ the ith configuration of the parameters that fix the operation of the environment.

Environment	$param_1$	$param_2$	$param_3$	$param_4$
A_1	0.01	0.02	0.02	0.02
A_2, A_2'	52	32	32	32
A_4	0.080	0.080	0.080	0.001
ρ_1	847	747	647	647
C_{p1}	0.12	0.12	0.12	0.12
ρ_2	8050	8050	5050	5050

Eq. (21) shows that the fouling layer thickens linearly as a function of the sedimentation capacity C_a of the material at the reaction rate at which fouling particles precipitate k_1, and $A_1 > 0$ is a control constant. In parallel, C_a decreases over time according to the control constant A_4 at the reaction rate set by k_2. In Eq. (23) we can see the expression defining k_1 and k_2 where $A_2, A_2' > 0$ is a constant preexponential factor, $A_3 > 0$ is the activation energy constant, R is the ideal gas constant and T_1 is the fluid temperature. The constants A_1, A_2, A_2', A_3 and A_4 serve to generate different systems but not to produce variations that the agent must be able to identify and anticipate. Each environment is tightly defined by these parameters and constants, which establish the state space. The graph in Fig. 5 illustrates this relationship.

In Eqs. (23), (24) and (25), ρ_1 is the density of the fluid, C_{p_1} is the thermal capacity of the fluid, Q_{in} is the flow of new fluids entering the system, ∇T is the variation in temperature between the material inside the tube and the fresh material that is entering, A_5 and A_6 are the control constant and the volume of the fluid in the tube, and r is tube radius. These equations represent the fact that the convective capacity of the tube degrades as the embedding layer S_c increases; furthermore, the temperature can decrease as new fluids are introduced into the system at lower temperatures, as the causal Q_in increases, which we measure by means of the vol (volume) of fluid and the capacity. Otherwise, the temperature would grow steadily even if T_2 did not continue to grow.

Eqs. (26) and (27) show how T_2 absorbs the energy generated by the heaters in T_3 and loses it more slowly when heating internal fluid at temperature T_1; the velocity decreases if fouling increases. The transmission capacity of the tube depends on its thickness ρ_2, and C_{p_2} is the thermal capacity.

Looking at the temperature of the oven, T_3 is calculated in Eq. (28) and is bounded above and below by the temperatures T_{min} and T_{max}, depending on the amount of fuel input m_c, which represents the valve opening size using a sigmoid to calculate the exact volume of fuel associated.

The variables that can change the state of the system are S_c and C_a. ρ_2 and C_{p_2} are not considered among these variables because we do not aim to vary the composition of the tube; m_c and vol are the variables that allow operator actions, with the former increasing the temperature and the latter decreasing it.

We consider that the conditions of the environment vary, both the initial values of the different variables (ρ_1, ρ_2) measured, and the constants (A_1, A_2, A_2', A_3, A_4, A_5, A_6) that establish the relationships. These modifications do not make physical sense but they will help us outline different scenarios and validate the results in each of them.

Table 1 lists the parameters assigned to each of the examples of defined environments. The trajectories they generate are shown in Fig. 6. For the last example we add white noise to the trajectories to simulate the information collection process.

All other variables take the following fixed values for all tested configurations: $A_3 = 10800$, $R = 8.31$, $A_6 = 0.3$, $C_{p_2} = 43$, $A_7 = 1e-05$, $m_c = 3$, $T_{min} = 800$, $T_{max} = 1300$, $C_{p2} = 43$, $A_5 = 0.1$.

Some actions have an impact on the system by varying the other variables. They are:

- Type 1, Stop $T_2(t) = T_{min}$ and $\frac{\partial S_c}{\partial t} = -\beta$; this action involves stopping to clean the fouling tubes with a cleaning speed of β.

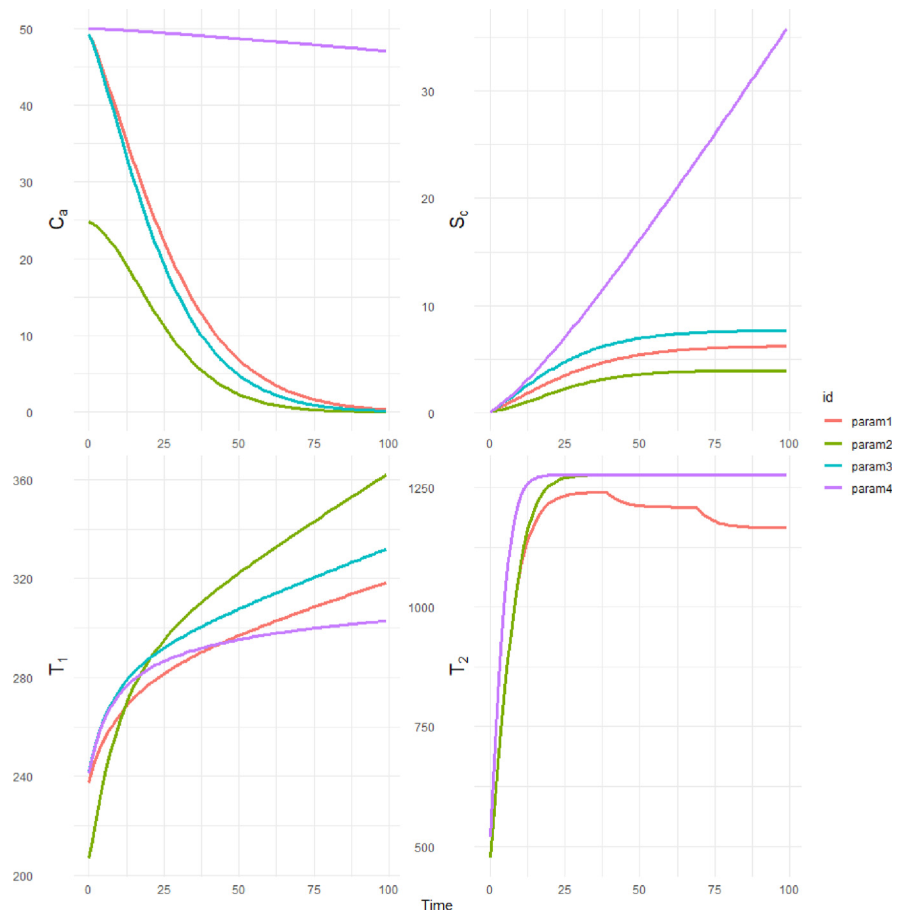


Fig. 6. Trajectory of the different variables measured in the environment. The colour represents each of the settings in the environment (see 1).

T_2 is the temperature of the heater which, when this action is applied, is brought to the minimum temperature T_{min}. Note that we can decide at any instant of time, to stop until the fouling is removed or resume operations sooner.

- Type 2, varies the temperature $T_2(t)$, applied to the walls, in a continuous range to increase the internal temperature and the reaction rate. With this action space, the optimal policies found by the agents gradually increase the temperature to accelerate the reaction without the risk of exceeding the melting temperature.
- Type 3, varies the flow rate $Q_{in}(t)$. This causes the temperature $T_1(t)$ to drop as more liquid is introduced at a lower temperature and the fouling capacity C_a is replenished as the new product contains more soil. $\partial Q(t) > 0 \Rightarrow \partial T_1(t) < 0$ y $\partial C_a(t) > 0$

The causal representation provided by the system of ODEs is illustrated in Fig. 7; this is the structure of the Bayesian network, where node A represents a distribution of actions.

4.2.2. Definition of the problem in RL

The basic structure of an RL model consists of three main parts: the environment, the reward and the actions to be applied. The environment is described by the values that the variables take along the trajectory of 100 temporary evaluations that we generate from the ODE system between decision making steps, in addition to the last applied action. Additionally, we define the action space in the simplest way: we only have to decide whether to continue or stop to clean by removing dirt and reducing the temperature. Finally, the reward is the estimated value of the product obtained in the reaction. However, this is an exponential function of the temperature T_1 reached inside the tube. To simplify and compare the results of the different experiments, we define

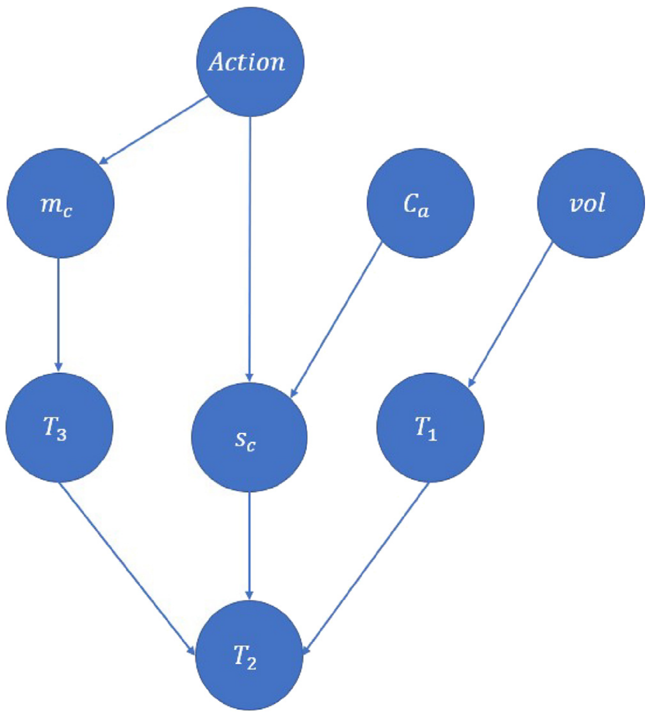


Fig. 7. Graph derived from a system of ordinary differential equations.

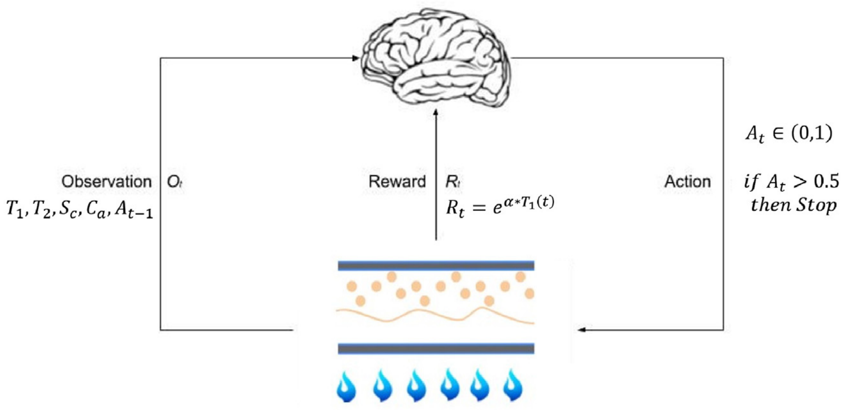


Fig. 8. RL graph representing the main components of an RL system in the particular case of the tube fouling problem.

instead temperature T_1 as the reward as long as T_2 does not exceed the threshold of 1500 in which case we have a cost of $-T_1$:

$$r(s) = \begin{cases} -T_1 & \text{if } T_2 \geq 1550 \\ T_1 & \text{otherwise} \end{cases} \tag{29}$$

In Fig. 8 offers a summary representing the main components of an RL system in the particular case of the tube fouling problem. We have the observations obtained from the sensors of the system that measure the variables described in constructing the ODEs, T_1, T_2, S_c, and C_a as well as the action applied in the previous instant A_{t-1}. The action space depends on the type of alternative chosen among those in Section 4.2.1; in this case, we have selected $type_1$. The agent selects in each situation a numerical value within the interval $(0, 1)$; if it exceeds a certain threshold the plant will stop. The agent determines values according to the observations of which action increases the reward expectation, where the values are determined by means of an exponential function, taking T_1 as the main objective.

4.2.3. Sample size considerations

In the design of a RL experiment, the approach to determining the sample size differs from other ML problems. Instead of requiring a static training sample size, RL needs a dynamic environment that generates responses to interventions or a continuous simulation that can generate uninterrupted responses until a failure or cessation of the generative process occurs. In our case, we addressed this issue by setting a sufficiently large time horizon to allow for potential failures and achieve convergence in those cases where it occurred. The length of the horizon depends on the specific problem, as it is conditioned by the probability of failure or breakdown. It can be defined based on expert criteria. Hence, we have been able to evaluate the performance and effectiveness of our approach in different scenarios, ensuring the validity of our results in relation to the sample size. It is worth noting that we have taken into account some theoretical results regarding the minimum sample size for training Bayesian networks (Chickering, 1996b; Madigan et al., 1995; Neapolitan et al., 2004), although in two of our cases, the network structure was pre-defined by experts, which reduces this requirement.

Furthermore, we have considered the inherent uncertainty in the proposed generative systems and the selected algorithms. We conducted multiple experiments to account for the stochastic factors present in the data generation systems. We used descriptive statistics and hypothesis testing that take into account these sources of variability to obtain solid and statistically supported conclusions. Thus, the reliability of our results is guaranteed.

4.3. Baseline models

Our proposals are an alternative to value and policy-based hybrid strategies, so-called actor-critic strategies, which also try to solve the two main problems of the high variance of the gradients and the high number of iterations with the environment. Among them, we have chosen the methods that also use machine learning algorithms (neural networks) internally:

- A2C: A policy based alternative, a synchronous, deterministic variant of A3C (Mnih et al., 2016).
- DDPG: Deep deterministic policy gradient. Deep Q-learning (value based) is applied to the continuous action domain (Lillicrap et al., 2016).
- PPO: Proximal policy optimization, a policy based alternative. The main idea is that, after an update, the new policy should be not too far from the old policy. To ensure this, PPO uses clipping to avoid updates that are too large (Wang et al., 2020).
- SAC: Soft actor critic (SAC) off-policy maximum entropy deep reinforcement learning with a stochastic actor (Haarnoja et al., 2018).

The proposals defined to solve the considered problem of optimal industrial control seek to include expert information. Expert information comes from the system's dynamics described by ODEs and the professionals who currently carry out this task. Therefore, it is essential to compare our method with current decision-making strategies.

Considering that a deterministic ODE is a good fit for the system evolution, the problem becomes one of finding the optimal design and can be solved with rules. Technicians use two rules: the first proposes stopping when the fouling level S_c exceeds 5, the level at which isolation makes the process very inefficient. The latter proposes stopping when the temperature level in T_2 exceeds a threshold that is considered risky, 1550 °C.

The following figures present the results of applying the first strategies in the system. We can see in Fig. 9, which represents the most common configuration, that the strategy is valid and seems to give good results since the furnace keeps running for a long time at high temperatures without exceeding the security thresholds.

The case in Fig. 10 shows one of the weaknesses of these strategies: they tend to over-adjust to common situations and can even be harmful in situations with minor variations. In this case, we observe how the rule is inefficient, and the temperature remains at very high levels even though fouling does not exceed the established level. This entails high risks and reduces profit. In Figs. 11 and 12, where the configuration is more favourable, the results are more similar to the case represented in Fig. 9.

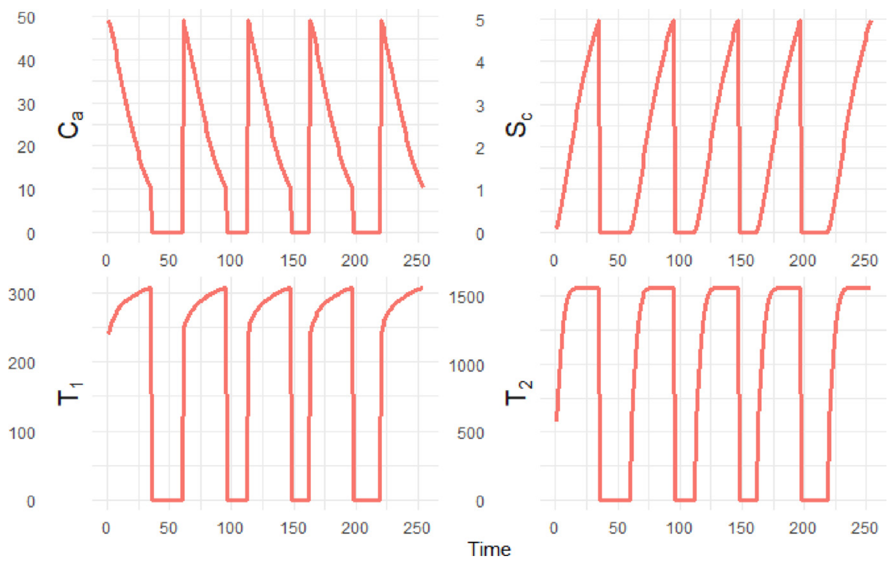


Fig. 9. Trajectory of the different variables measured in the environment for the case of fouling applying human strategies. Rule "if $S_c \geq 5$ then cleanup and stop" is applied. Parameter setting $param_1$ in Table 1.

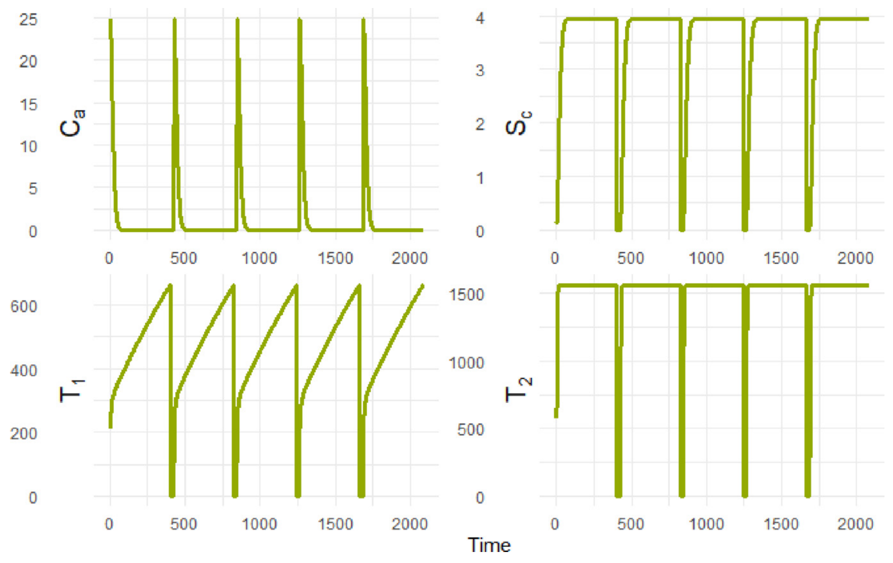


Fig. 10. Trajectories of the different variables measured in the environment for the case of fouling by applying typical strategies. The rule "if $S_c \geq 5$, then stop and clean" is applied. Parameter setting $param_2$ in Table 1 is used.

4.4. Results

The experiments were designed to validate the hypotheses raised for the built solutions and to verify that information speeds up learning and produces more generic agents. These agents have the capacity to adapt to different configurations of the environment, avoiding penalties and crashes. To achieve this goal, we designed different variants for the same industrial example, modifying the parameters so that the first set serves as a training set, and the rest are used for evaluation. In addition, we compare solutions using the CartPole example as a benchmark.

The algorithms are trained with 100 simulations of each of the trajectories that are lengthened by a total of 100 time units. We consider the average reward (AR) and the average failure rate (FR) in these 100 simulations. We make the first evaluation in the training environment of the algorithms in (Table 2) and then assess the decision-making of the agents trained in new scenarios for the same problem in Table 3.

Our experiments verified that models with a causal structure lead to fewer failures in the training system. They are capable of a quicker response to variations in the parameters, with a lower number of failures before a sure policy is defined.

We observed that the solution with neural networks as the engine (DDPG) shows the most significant difference between the results in training and testing. This is because they are too conservative; the results with rules either do not lead to failure at any time or lead to negative sums. In contrast, the solutions obtained by the causal engines ($RLBN_{EXP}$, $RLBN_{SEMIEXP}$, $RLBN_{BDATA}$) have better results when varying the environment and when working with different parameters. In addition, they seem to find a better policy than the operator' rules, optimizing downtimes and adjusting them with more precision.

The results presented in both tables highlight the performance differences between the various models tested. Table 4 shows the results of the paired hypothesis tests, which reveal that the expert model $RLBN_{EXP}$ consistently outperforms all other models in terms of mean reward, with statistical significance against all of them, except for

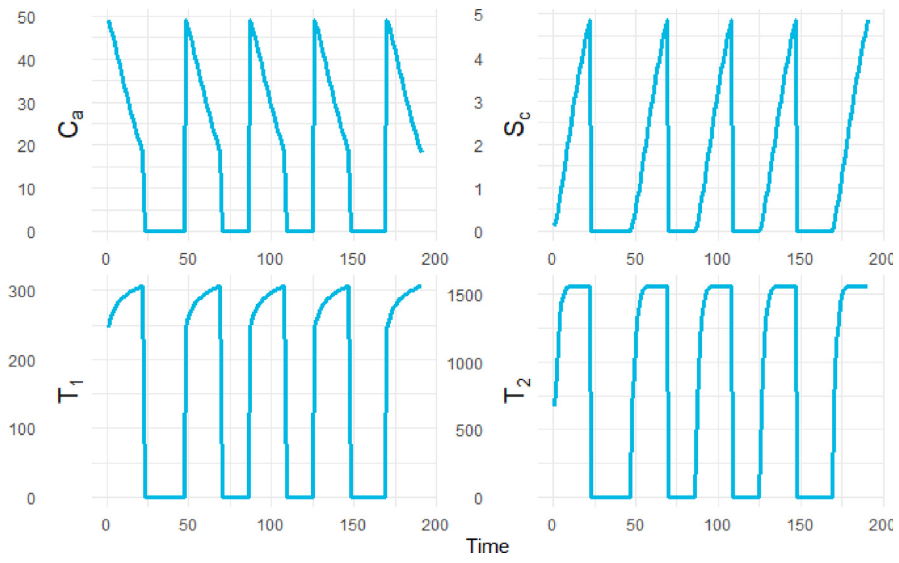


Fig. 11. Trajectories of the different variables measured in the environment for the case of fouling with typical strategies. The rule "if $S_c \geq 5$, then stop and clean" is applied. Parameter setting $param_3$ in Table 1 is used.

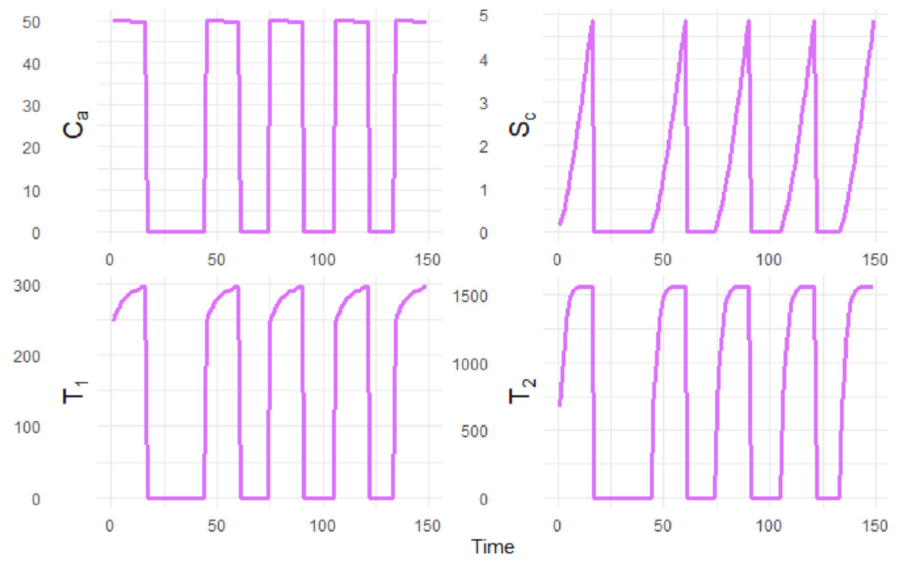


Fig. 12. Trajectories of the different variables measured in the environment for the case of fouling with typical strategies. The rule "if $S_c \geq 5$ then stop and clean" is applied. Parameter setting $param_4$ in Table 1 is used.

Table 2
Average reward and average failure rate, obtained in 100 simulations of trajectories of 100 time units for the CartPole and fouling ($param_1$) problems.

Environment	CartPole		Fouling $param_1$	
Algorithms	AR	FR	AR	FR
DDPG	131	0.25	−62	0.4
A2C	147	0.25	205	0.15
PPO	120	0.10	225	0.32
SAC	170	0.10	257	0.10
$Rule_1$ ($T_2 \leq 1200$)	–	–	283	0
$Rule_2$ ($S_c \leq 5$)	–	–	263	0
$RBLN_{EXP}$	190	0.02	**293**	**0.01**
$RBLN_{SEMIEXP}$	196	0.01	289	0.05
$RLBN_{BDATA}$	175	0.1	261	0.15

$RLBN_{SEMIEXP}$, which has a very similar network structure and a p value of 0.433. Although $RLBN_{SEMIEXP}$ also achieves good results, the failure rate is similar to those of DDPG and PPO. In Table 3, we can observe the different performance levels of each model when facing new scenarios.

Regarding the CartPole environment, $RLBN_{EXP}$ achieves an average reward improvement of more than 40 points compared to the other models, while maintaining a failure rate that is less than half of those in DDPG and PPO. On the other hand, $RLBN_{SEMIEXP}$ performs better than $RLBN_{EXP}$ with an average improvement of up to 50 points. However, its runs more risks, with a failure rate similar to DDPG and PPO. Nevertheless, without expert references on the system's operation to manage the uncertainty in the exploratory phase, the failure rate of $RLBN_{SEMIEXP}$ is similar to other models.

The data-based model differs from the other two proposals and is similar to the results obtained with the PPO algorithm and $Rule_2$. $Rule_2$ is conservative and it is common for historical data. Reducing the application of actions with high uncertainty makes $RLBN_{BDATA}$ more conservative, applying the actions with the highest historical probabilities. PPO is similar to TRPO but simpler, and it can lead to bad decisions; however, it introduces a concise restriction on the

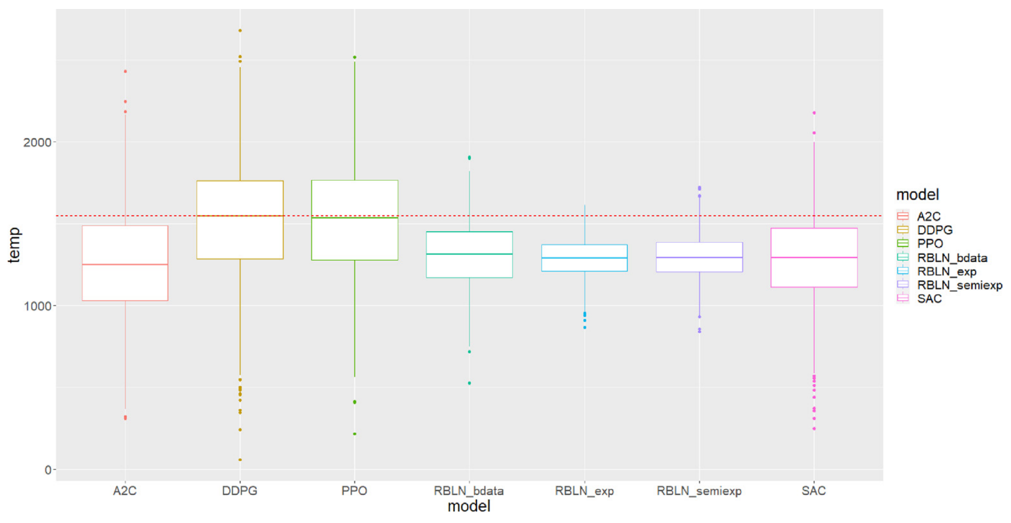


Fig. 13. The distribution of the temperatures at which the stopping decisions were made for each of the tested RL algorithms in the fouling case. The horizontal line indicates the initial limit temperature that can lead to breakage.

Table 3
Results obtained by the pretrained agents in the basic case ($param_1$ ODEs and CartPole) when facing new scenarios. The average reward and average break rate, obtained in 100 simulations of trajectories of 100 time units are shown.

Environment	CartPole		Fouling $param_2$		Fouling $param_3$		Fouling $param_4$	
Algorithms	AR	FR	AR	FR	AR	FR	AR	FR
DDPG	127	0.26	−185	0.45	−92	0.41	240	0.39
A2C	125	0.12	240	0.25	220	0.17	**290**	0.03
PPO	146	0.25	−56.5	0.3	131	0.3	255	0.34
SAC	165	0.085	186	0.2	193	0.1	245	0.08
$Rule_1$ ($T_2 \leq 1250$)	–	–	266	**0**	255	**0**	244	**0**
$Rule_2$ ($S_c \leq 5$)	–	–	−58.3	0.3	123	0.2	227	0.1
$RLBN_{EXP}$	175	**0.1**	**306**	**0**	275	**0**	**284**	**0**
$RLBN_{SEMIEXP}$	**195**	0.2	296	**0**	**278**	**0**	281	**0**
$RLBN_{BDATA}$	175	**0.1**	−57.2	0.3	133	0.2	223	0.1

Table 4
Results of the paired hypothesis t tests carried out to validate the statistically significant difference (5%) between the AR results of the models taking into account the parameterization as a factor.

Our algorithms	$RLBN_{EXP}$		$RLBN_{SEMIEXP}$		$RLBN_{BDATA}$	
Other algorithms	Result	*p*-value	Result	*p*-value	Result	*p*-value
DDPG	reject	0.0166	reject	0.028	reject	0.0043
A2C	reject	0.0102	reject	0.0094	reject	0.0085
PPO	reject	0.0201	reject	0.03	fail	0.26
SAC	reject	0.035	reject	0.0232	reject	0.0164
$Rule_1$ ($T_2 \leq 1200$)	reject	0.031	reject	0.018	reject	0.00616
$Rule_2$ ($S_c \leq 5$)	reject	0.0027	reject	0.009	fail	0.1299
$RLBN_{EXP}$	–	–	fail	0.4	reject	0.04
$RLBN_{SEMIEX}$	fail	0.433	–	–	reject	0.032
$RLBN_{BDATA}$	reject	0.04	reject	0.032	–	–

optimization problem, thus reducing the failure rate and favouring more conservative strategies.

In Fig. 13, the boxplot shows the distribution of the temperatures at which decisions to stop the process were made for each of the RL algorithms tested on the fouling problem with standard parameters. The horizontal line indicates the initial limit temperature that can lead to rupture. Our proposed alternatives, including $RLBN_{SEMIEXP}$ and $RLBN_{BDATA}$, demonstrate greater robustness, making stable and uniform decisions with less failures than DDPG, PPO, A2C and SAC. These other four exhibit higher variability, requiring testing of more distinct cases and with more outliers. Our proposed alternatives make decisions based on multiple variables, leading to a more accurate representation of the system and reducing the likelihood of making a decision based on insufficient information.

In some cases, we observed negative rewards; the reward is negative when the temperature T_2 exceeds a limit value. We penalize accepting the risk of approaching the physical limit of the material. If the melting temperature is reached, it results in complete equipment failure, which requires a shutdown of the system until the damaged furnace parts are replaced, resulting in additional repair costs. The algorithms that have negative values are those that either are not able to compensate for the losses due to excess temperature in the exploratory phase with the benefits obtained once a policy has been found or need more iterations to find an optimal policy that avoids these drops.

The Bayesian network structures obtained by the proposed algorithms exhibit similarities, thus confirming the effectiveness of the architecture score-based learning process.

The $RLBN_{SEMIEXP}$ strategy only needs to consider the relationship between state variables and their corresponding action, unlike the expert-defined policies. As a result, the only difference from the original structure (Fig. 7) is the lack of connection between m_c and *action* in Fig. 14. Since using multiple variables may result in conflicts arising from data reading errors, decision-making based on a single node connected to *Action* seems reasonable. However, it is worth noting that this approach may not accurately fit the connections with the action node. The expert strategies, $Rule_1$ and $Rule_2$, are conservative, but the solutions provided by $RLBN_{SEMIEXP}$ tend to resemble each other and adjust the cut-off values more precisely than the expert method. It is also possible that the relationship structure between state variables is accurate and fits the data correctly but does not fit the connections with the action node. The method of making decisions constantly changes, which affects the connections. Reducing the number of variables connected to *Action* may affect the reward in the case of modifications but

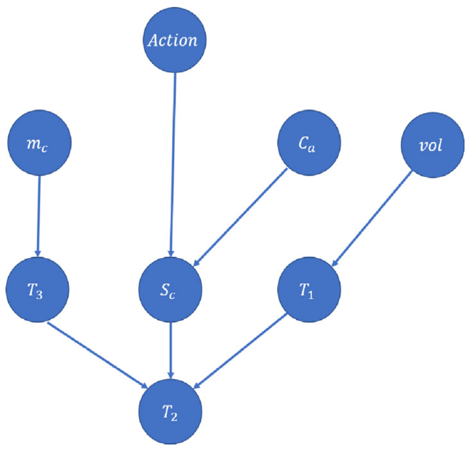


Fig. 14. Structure learned by $RLBN_{SEMIEXP}$.

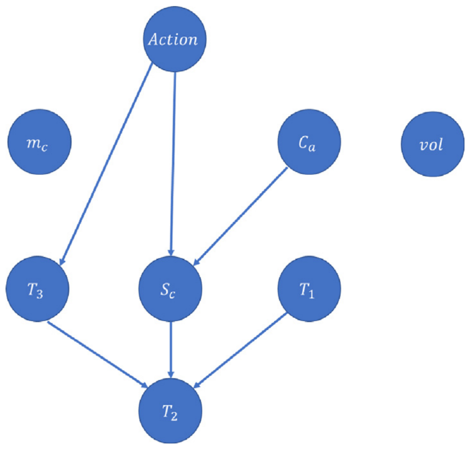


Fig. 15. Structure learned by $RLBN_{BDATA}$.

not the likelihood as much. For example, Fig. 14 shows that decisions are made based on fouling S_c, the only node directly connected to *Action*. If there is a weight change, it may impact the reward but not the likelihood, as most of the system's structure remains unaffected.

In contrast, the $RLBN_{BDATA}$ algorithm's output graph has no prior causal information provided in graph form. Therefore, some connections may be lost due to data collection problems or redundant information. Focusing decision making on a few variables may simplify the method of decisions, reduce uncertainty and bound the likelihood. The way in which the level of C_a affects fouling and how T_3 is calculated from m_c also play a role in the cause–effect relationship between the variables. Moreover, choosing an action based on the cause variables can have negative effects on the reward by being too far ahead of possible consequences. Therefore, neither the *Action* variable in Figs. 14 and 15, nor its "effect" variables in Fig. 15, are connected, unlike in the original structure (Fig. 7).

The proposed models' improvement over the rest of the alternatives in Table 3 may be due to their ability to reduce the search space and the number of trials according to the context and uncertainty using the likelihood. This reduces the number of failures necessary to reach the solution, as illustrated in Table 3 (FR column), where models such as PPO and DDPG have up to 4 times more failures than the proposed alternatives.

Furthermore, the proposed models show a higher level of adaptability to changes in the environment and parameters, which is essential in industrial settings where the production process can be subject to multiple variations. This is demonstrated by the results in Table 3, where the RLBN models outperform in most cases the other alternatives in terms of average reward and failure rate in most cases.

In conclusion, the proposed RLBNs for industrial settings present a novel approach to address the optimization of decision-making processes in dynamic and complex environments. The simulation results indicate that the proposed methodology outperforms other alternatives in terms of adaptability and effectiveness. The use of knowledge-based models represents more accurately the relationship between variables, reducing uncertainty and improving decision-making. The experiments validate the hypothesis that the use of our alternatives leads to fewer failures and faster learning in the training phase, as well as a more efficient response to changes in the environment and parameters.

5. Conclusion

This paper presents an innovative solution to overcome the limitations of RL solutions in environments where risky exploration and interpretability are fundamental. Our approach proposes the use of probabilistic graphical models with causal structures to infer the agent's optimal policy. Additionally, we develop three algorithms of dynamic learning: of parameters alone; parameters and a simplified structure; and the entire structure of the joint probability distribution of states and actions as the policy.

To validate our solution, we compared our technique with other existing ones and measured the variance reduction, robustness, decrease in the number of crashes, increase in reward, and adaptability. Additionally, we controlled the iterations on the architecture during the learning process and took advantage of the power of uncertainty measurements to limit high-risk and low-knowledge actions. We verified our approach on a simple, generic benchmark, where the relationship between the variables gives rise to a very simple Bayesian network structure. However, in a simulated industrial case of pipeline fouling, we noted the advantages of introducing expert information in the process by comparing the different RLBN alternatives. This was reflected in the better average and accumulated results obtained. Indeed, the agent begins to apply more successful actions earlier. It is also reflected in the lower number of failures reduced by using uncertainty as another criterion for selecting interventions. Additionally, as we fixed more Bayesian network structures, we further reduced exploration errors. Meanwhile, we also increased the agent's inflexibility in the face of changes in structural relationships. In general, our solutions are more robust, run fewer risks, and lead to fewer accidents than other models.

In addition, it is important to discuss the practical implications of our results. Our proposed RLBN approach has practical advantages in various industrial applications, such as process control and optimization, fault diagnosis, and predictive maintenance. For instance, in the case of pipeline fouling, our approach can effectively predict and prevent fouling occurrences, leading to significant cost savings and increased equipment lifetime. RLBNs can provide interpretable decision-making solutions in process control and optimization, leading to more efficient and sustainable operations. In fault diagnosis, RLBNs could identify the root cause of system malfunctions, leading to faster repairs and reduced downtime. In predictive maintenance, RLBNs could forecast equipment failure and schedule maintenance activities, reducing the likelihood of unplanned downtime and improving overall equipment effectiveness.

In the future, we plan to explore additional applications of RLBN and further refine our approach to address the challenges and limitations identified. Specifically, to explore alternative proposals to RL and BNs, such as dynamic Bayesian networks to better model the dynamic structure of differentiable equations or the use of nonlinear RB models of the kernel type (Atienza et al., 2022). We also agree that insufficient knowledge may be available to define the network structure. We would like to explore different forms of causal learning of network structures from historical data or recent works in which RL is used to learn cause-and-effect relationships. We would represent them in Bayesian network forms (Méndez-Molina et al., 2022).

In our future research, we plan to incorporate dynamic Bayesian networks as the modelling engine for policy, address uncertainties in model parameters, and explore alternative approaches to causal learning. By doing so, we aim to enhance the capabilities of our RLBN framework and provide more robust and accurate predictions for fouling test cases. These advancements will enable us to better understand the influence of various factors on the resulting profiles and improve performance and reliability in real-world applications. Furthermore, these enhancements will allow our system to model second-order kinetics or delayed effects in the generation term of the sedimentary capacity, as well as address the dynamic uncertainty associated with this process.

CRediT authorship contribution statement

Gabriel Valverde: Conceptualization, Methodology, Investigation, Software, Writing – original draft. **David Quesada:** Conceptualization, Investigation. **Pedro Larrañaga:** Supervision, Writing – review & editing. **Concha Bielza:** Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Gabriel Valverde reports financial support was provided by Spanish Ministry of Science, Innovation and Universities.

Data availability

No data was used for the research described in the article.

Acknowledgements

This study was partially supported by the Spanish Ministry of Science, Innovation and Universities through the PID2019-109247GB-I00 project and TED2021-131310B-I00 projects and by the grant to the ELLIS Unit Madrid by the Autonomous Region of Madrid.

References

Atienza, D., Larrañaga, P., Bielza, C., 2022. Hybrid semiparametric Bayesian networks. TEST 31 (2), 299–327.

Bai, W., Li, T., Tong, S., 2020. NN reinforcement learning adaptive control for a class of nonstrict-feedback discrete-time systems. IEEE Trans. Cybern. 50 (11), 4573–4584.

Barto, A.G., Sutton, R.S., Anderson, C.W., 1983. Neuronlike adaptive elements that can solve difficult learning control problems. IEEE Trans. Syst. Man Cybern. SMC-13 (5), 834–846.

Benjumeda, M., Luengo-Sanchez, S., Larrañaga, P., Bielza, C., 2019. Tractable learning of Bayesian networks from partially observed data. Pattern Recognit. 91, 190–199.

Bishop, C.M., Nasrabadi, N.M., 2006. Pattern Recognition and Machine Learning, Vol. 4. Springer.

Bott, T.R., 1995. Fouling of Heat Exchangers. Elsevier.

Boyes, H., Hallaq, B., Cunningham, J., Watson, T., 2018. The industrial internet of things (IIoT): An analysis framework. Comput. Ind. 101, 1–12.

Brusakov, V., 1971. Law for the deposition of materials on heat-transmitting surfaces under the action of thermoelectric effects. Atomnaya Energiyae 30, 10–14.

Chickering, D.M., 1996a. Learning Bayesian networks is NP-complete. Networks 121–130.

Chickering, D.M., 1996b. Learning Bayesian networks is NP-complete. In: Learning from Data: Artificial Intelligence and Statistics V. Springer, pp. 121–130.

Copisarow, M., 1945. Marine fouling and its prevention. Science 101 (2625), 406–407.

Dawid, P., 2020. Decision-theoretic foundations for statistical causality. J. Causal Inference 9, 39–77.

Du, Y., Li, J.-q., Chen, X.-l., Duan, P.-y., Pan, Q.-k., 2022. Knowledge-based reinforcement learning and estimation of distribution algorithm for flexible job shop scheduling problem. IEEE Trans. Emerg. Top. Comput. Intell.

Gámez, J., Mateo, J., Puerta, J., 2011. Learning Bayesian networks by hill climbing: efficient methods based on progressive restriction of the neighborhood. Data Min. Knowl. Discov. 22, 106–148.

Gershman, S.J., 2017. Reinforcement learning and causal models. Oxf. Handb. Causal Reason. 1, 295.

Ghavamzadeh, M., Mannor, S., Pineau, J., Tamar, A., et al., 2015. Bayesian reinforcement learning: A survey. Found. Trends Mach. Learn. 8 (5–6), 359–483.

Haarnoja, T., Zhou, A., Abbeel, P., Levine, S., 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. arXiv:1801.01290.

Harper, C., 1976. Introduction to Mathematical Physics. Prentice hall.

Heckerman, D., Geiger, D., Chickering, D., 1995. Learning Bayesian networks: The combination of knowledge and statistical data. Mach. Learn. 20, 197–243.

Ji, Z., Xia, Q., Meng, G., 2015. A review of parameter learning methods in Bayesian network. In: Advanced Intelligent Computing Theories and Applications. Springer, pp. 3–12.

Koller, D., Friedman, N., 2009. Probabilistic Graphical Models: Principles and Techniques. The MIT Press.

Kullback, S., Leibler, R.A., 1951. On information and sufficiency. Ann. Math. Stat. 22 (1), 79–86.

Larrañaga, P., Atienza, D., Diaz-Rozo, J., Ogbechie, A., Puerto-Santana, C., Bielza, C., 2018. Industrial Applications of Machine Learning. CRC Press.

Lawal, M.O., 2021. Tomato detection based on modified YOLOv3 framework. Sci. Rep. 11 (1), 1–11.

Lepenioti, K., Pertselakis, M., Bousdekis, A., Louca, A., Lampathaki, F., Apostolou, D., Mentzas, G., Anastasiou, S., 2020. Machine learning for predictive and prescriptive analytics of operational data in smart manufacturing. In: Advanced Information Systems Engineering Workshops. Springer, pp. 5–16.

Li, C., Quiu, M., 2020. Reinforcement Learning for Cyber-Physical Systems. Chapman and Hall CRC.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D., 2016. Continuous control with deep reinforcement learning. In: International Conference on Learning Representations. pp. 10–15.

Madigan, D., York, J., Allard, D., 1995. Bayesian graphical models for discrete data. Int. Stat. Rev./Revue Int. Stat. 215–232.

McLachlan, S., Dube, K., Hitman, G.A., Fenton, N.E., Kyrimi, E., 2020. Bayesian networks in healthcare: Distribution by medical condition. Artif. Intell. Med. 107, 101912.

Méndez-Molina, A., Morales, E.F., Sucar, L.E., 2022. Causal discovery and reinforcement learning: A synergistic integration. In: International Conference on Probabilistic Graphical Models. PMLR, pp. 421–432.

Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K., 2016. Asynchronous methods for deep reinforcement learning. In: Proceedings of the 33rd International Conference on Machine Learning. Vol. 48. pp. 1928–1937.

Mnih, V., Kavukcuoglu, K., Silver, D., 2015. Human-level control through deep reinforcement learning. Nature 518, 529–533.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M., 2013. Playing atari with deep reinforcement learning. arXiv preprint.

Nagendra, S., Podila, N., Ugarakhod, R., George, K., 2017. Comparison of reinforcement learning algorithms applied to the cart-pole problem. In: 2017 International Conference on Advances in Computing, Communications and Informatics. ICACCI, IEEE, pp. 26–32.

Neapolitan, R.E., et al., 2004. Learning Bayesian Networks, Vol. 38. Pearson Prentice Hall Upper Saddle River.

Pearl, J., 1986. Fusion, propagation, and structuring in belief networks. Artif. Intell. 29, 241–288.

Pearl, J., 1995. Causal diagrams for empirical research. Biometrika 82 (4), 669–688.

Quesada, D., Bielza, C., Fontán, P., Larrañaga, P., 2022. Piecewise forecasting of nonlinear time series with model tree dynamic Bayesian networks. Int. J. Intell. Syst.

Ramoni, M., Sebastiani, P., 2001. Robust learning with missing data. Mach. Learn. 45, 147–170.

Roy, A.M., 2022. Adaptive transfer learning-based multiscale feature fused deep convolutional neural network for EEG MI multiclassification in brain–computer interface. Eng. Appl. Artif. Intell. 116, 105347.

Scanagatta, M., Corani, G., Zaffalon, M., Yoo, J., Kang, U., 2018. Efficient learning of bounded-treewidth Bayesian networks from complete and incomplete data sets. Internat. J. Approx. Reason. 95, 152–166.

Schaal, S., 1996. Learning from demonstration. In: Advances in Neural Information Processing Systems, Vol. 9. The MIT Press, pp. 10–15.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P., 2015. Trust region policy optimization. In: Proceedings of the 32nd International Conference on Machine Learning, Vol. 37. pp. 1889–1897.

Shachter, R.D., Kenley, C.R., 1989. Gaussian influence diagrams. Manage. Sci. 35 (5), 527–550.

Silva, A., Gombolay, M., Killian, T., Jimenez, I., Son, S.-H., 2020. Optimization methods for interpretable differentiable decision trees applied to reinforcement learning. In: Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics. In: Proceedings of Machine Learning Research, vol. 108, pp. 1855–1865.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al., 2018. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science 362 (6419), 1140–1144.

Song, Y., Zhou, Y., Sekhari, A., Bagnell, J.A., Krishnamurthy, A., Sun, W., 2022. Hybrid RL: Using both offline and online data can make RL efficient. arXiv preprint arXiv:2210.06718.

Spiegelhalter, D.J., Lauritzen, S.L., 1990. Sequential updating of conditional probabilities on directed graphical structures. Networks 20 (5), 579–605.

Sutton, R., Barto, A., 1998. Reinforcement Learning: An Introduction. The MIT Press.

Sutton, R.S., McAllester, D., Singh, S., Mansour, Y., 1999. Policy gradient methods for reinforcement learning with function approximation. In: Solla, S., Leen, T., Müller, K. (Eds.), Advances in Neural Information Processing Systems, Vol. 12. The MIT Press, p. 1.

Tedrake, R., Zhang, T., Seung, H., 2004. Stochastic policy gradient reinforcement learning on a simple 3D biped. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, Vol. 3. pp. 2849–2854.

Treesatayapun, C., 2020. Knowledge-based reinforcement learning controller with fuzzy-rule network: experimental validation. Neural Comput. Appl. 32 (13), 9761–9775.

Wang, Y., He, H., Tan, X., 2020. Truly proximal policy optimization. In: Proceedings of the 35th Uncertainty in Artificial Intelligence Conference, Vol. 115. pp. 113–122.

Williams, R.J., 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach. Learn. 8 (3), 229–256.

Zhang, Y., Lan, Y., Fang, Q., Xu, X., Li, J., Zeng, Y., 2021. Efficient reinforcement learning from demonstration via Bayesian network-based knowledge extraction. Comput. Intell. Neurosci. 2021.