Hindawi
Complexity
Volume 2022, Article ID 3652441, 15 pages
https://doi.org/10.1155/2022/3652441

Research Article

Reinforcement Learning with Probabilistic Boolean Network Models of Smart Grid Devices

Pedro Juan Rivera Torres,[1,2] **Carlos Gershenson García**,[1,3,4] **María Fernanda Sánchez Puig**,[1,5] **and Samir Kanaan Izquierdo**[2,6]

[1]*Centro de Ciencias de La Complejidad (C3), Universidad Nacional Autónoma de México, Circuito Mario de La Cueva S/N, Cd. Universitaria, Coyoacán 04510, Ciudad de México, Mexico*
[2]*Bioinformatics and Biomedical Signals Laboratory, Centre de Recerca en Enginyeria Biomèdica, Universitat Politècnica de Catalunya, Facultat de Matemàtiques I Estadística, Edifici U, C/Pau Gargallo 5 08028, Barcelona, Spain*
[3]*Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, Universidad Nacional Autónoma de México, 04510 Ciudad de México, Mexico*
[4]*Lakeside Labs GmbH, Klagenfurt Am Wörthersee, Austria*
[5]*Facultad de Ciencias, Universidad Nacional Autónoma de México, Av. Universidad 3000, Circuito Exterior S/N Coyoacán, 04510, Ciudad de México, Mexico*
[6]*Institut de Recerca Sant Joan de Déu, Esplugues de Llobregat, Barcelona, Spain*

Correspondence should be addressed to Pedro Juan Rivera Torres; pedro.rivera@c3.unam.mx

Received 14 April 2021; Revised 28 November 2021; Accepted 23 May 2022; Published 13 June 2022

Academic Editor: Hiroki Sayama

The area of smart power grids needs to constantly improve its efficiency and resilience, to provide high quality electrical power in a resilient grid, while managing faults and avoiding failures. Achieving this requires high component reliability, adequate maintenance, and a studied failure occurrence. Correct system operation involves those activities and novel methodologies to detect, classify, and isolate faults and failures and model and simulate processes with predictive algorithms and analytics (using data analysis and asset condition to plan and perform activities). In this paper, we showcase the application of a complex-adaptive, self-organizing modeling method, and Probabilistic Boolean Networks (PBNs), as a way towards the understanding of the dynamics of smart grid devices, and to model and characterize their behavior. This work demonstrates that PBNs are equivalent to the standard Reinforcement Learning Cycle, in which the agent/model has an interaction with its environment and receives feedback from it in the form of a reward signal. Different reward structures were created to characterize preferred behavior. This information can be used to guide the PBN to avoid fault conditions and failures.

1. Introduction

There is not a picture of the present that is complete without electrical power; it has become essential to our civilization. Electrical power has been a constant in our lives for almost two centuries since Faraday's discovery and the first alternating current power grid in 1886. Generating, transmitting, and distributing electrical power has evolved from a commodity to a basic need during this time. This process has not changed much for a long time. Electricity is produced in different ways, but the basic cycle is essentially the same: it is generated (via electromechanical generators, geothermal power, nuclear fission, solar, and other means) and then it is delivered to clients via a transmission-distribution network.

Most modern systems are still like the first ones: centralized, unidirectional electrical power transmission systems with demand-driven control. In the latter decades of the 20th century, local grids began to arise, and since the early 21st, the industry has attempted to take advantage of telecommunication improvements to solve the limitations imposed by centralization and the challenges brought with the use of renewable sources and new technology like photovoltaic panels and wind turbines [1]. Decentralized systems provide benefits and cause significant challenges, boosting efficient techniques in modeling and controlling smart grid systems [2]. The European Union Commission Task Force on Smart Grids has defined these as an "electricity network that can cost efficiently integrate the behavior and actions of all users connected to it–generators, consumers, and those that do both–in order to ensure economically efficient, sustainable power system with low losses and high levels of quality and security of supply and safety" [3]. Applying Signal Processing and Communications to the power grid has allowed a flow of data that is one of the defining elements of the smart grid. This includes the use of "Smart Devices," such as the Intelligent Power Router (IPR). This device [4] was inspired on Internet routers, and it has a degree of "intelligence" that allows it to switch lines and shed loads. Devices such as the former allow the Electrical Power Distribution System (EPDS) to become reliable, resilient, flexible, and efficient. With them, decisions can be made in the event of power failures or component malfunctions, coordinating with other devices in their vicinity to react to load, demands, faults, and emergencies. It represents a multidisciplinary issue being faced in the last decade [5]. The basic elements of an IPR are shown in Figure 1.

EPDS that has incorporated IPRs (EPDS-IPR) has also the capacity of automatic service restoration if a network of IPRs is deployed strategically throughout the power grid, and if they are programmed for the exchange of information to manage and reconfigure the network following a rule set, any time a perturbation occurs. This allows survivability and better use of system resources.

Designing these EPDS-IPR networks is a very complex task. There is no specific model that can guide the designer. The devices must be configured with preset instructions on how to react when a particular set of conditions has occurred. Another challenging task is to make these grids adaptive [6] and not just follow a hard-wired set of instructions. A much more favorable situation is that the network can act autonomously and self-reconfigure in the event of a perturbation, i.e., loss of a power source, higher demand in critical loads, or sabotage. IPR devices when interconnected can be modeled as an intelligent Probabilistic Boolean Network, which is a complex-adaptive system that can learn from its steady state behavior and exhibits self-organization and resilience. Methodologies for modeling based on a Probabilistic Boolean Network (PBN) have been presented in [7–16], validating the use of PBNs as a modeling mechanism for industrial processes and Smart Grids using IPRs and enabling the simulation of several scenarios. This has the potential to allow designers to better program the devices and design a more robust network. We would like to imbue EPDS-IPRs with the intelligence that allows them to survive a wide set of perturbation events that are practically impossible to predict.

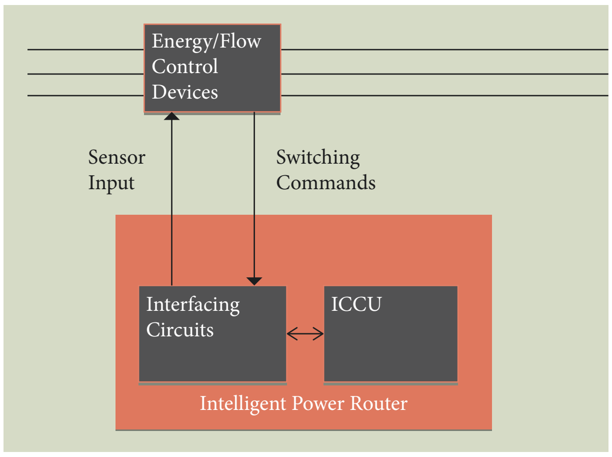

Figure 1: Basic elements of an IPR.

Biomimetic approaches have been used to analyze and solve complex problems in general and for EPDS design in particular. Frameworks that are qualitative in nature, such as PBNs, permit the description of biological system networks, with no property losses that are relevant to the system, and allow the representation of complex-adaptive behavior, such as self-healing and self-organization. Probabilistic Boolean Networks are used in bioinformatics for Gene Regulatory Network (GRN) modeling. GRNs are DNA segments in a cell that influence other segments and substances in it indirectly, to rule the level of expression of a gene or a set of genes. They are used to estimate the main rules that command the regulation of genes in genomic DNA. These PBNs are state-transition systems satisfying the Markov Property; they have no memory, so they are not reliant on previous states of the system). Proposed by I. Shmulevich and E. Dougherty [16] extending Stuart Kauffman's N-K or Boolean Network (BN) concept [17, 18], they mix the rule-based modeling richness of BNs and introduce probabilistic behavior. These PBNs are built upon a collection of constituent BNs which are assigned selection weights or probabilities, in which every BN can be considered a "context." Information for every one of the cells comes from antithetical sources; each represents a cell context. For every point in time *t*, a particular system can be commanded by a single BN, and the PBN will change to another context or constituent BN at a different time, based on a particular switching probability. The methodology for using PBNs in manufacturing engineering systems was proposed in [12, 16], with continued development in [7–11, 13–15].

In genomic research, the focus is to discern the way cells exercise control and perform extensive numbers of operations that are needed for their operation and function. They are massively parallel and highly cohesive systems, and a path that considers a perspective supreme to that of a single gene is needed so we can understand these biological processes better. Bioinformatics tools and algorithms are in high demand and have proven to be useful in solving these tasks [19]. However, novel computational approaches, digital medicine technologies, and networks and metabolic pathways analysis are needed to fully understand biological systems. Genes, cells, and molecules are networked systems that require a deeper understanding, to manufacture improved medicines and delivery mechanisms for treating and eradicating human disease. A mechanism for treating and processing massive quantities of data using computational methods and model checking can be used to understand the rules that govern them and make more accurate predictions about how these systems behave. EPDSs are akin to GRNs because to understand the main rules that control them and to make accurate forecasts on how they will behave, endure, or decline under a collection of prospects, models that correctly describe the system and its behavior are essential. The harmonized synergy, interaction, and governance between genes and their products form these chains, in which gene expression is an important factor.

In this research, the use of Probabilistic Boolean Networks, already applied auspiciously in manufacturing engineering systems, will be broadened to analyze IPR reliability and trust and the scrutiny of faults that may lead to catastrophe. As our main contribution, we explored the PBN's model capacity for performing a basic Reinforcement Learning (RL) cycle, and we explored RL as a means for directing the network's evolution to increase the network's resilience, working towards achieving automatic learning and control of itself using RL.

2. Preliminaries and Theoretical Background

A review of Boolean and Probabilistic Boolean Networks, Reinforcement Learning, and a basic understanding of Electrical Power Distribution Systems and Intelligent Power Routers is presented in the following subsections.

2.1. Probabilistic Boolean Networks in System Modeling and Simulation. Kauffman N-K or Boolean Networks (BNs) [17, 18] and PBNs [20] have been studied for biological systems modeling and their dynamics and to infer [21] their behaviors with statistical data analysis and [22] simulation. This application is very well documented in bioinformatics for biological systems modeling [23–27] and for GRNs description [28–33]. The mechanism of intervention [34] is used to conduct the evolution of the PBN away from unfavorable conditions or states as are those associated with illness.

Kauffman's BNs are a finite grouping of Boolean nodes [35, 36], in which states quantize to 0 or 1 (although in PBNs, alternative quantizations are possible). A state is determined by the present state of other nodes/genes in the network. The set of entry/input nodes in a BN is known as regulatory nodes, with a collection of Boolean functions (known as predictors), that dictate the future values of the different nodes. When the set of genes and their respective predictors are defined, the network is defined as well. PBNs are, in essence, a tree of BNs for which, at any particular time period, the node state vector transitions are established by one of the rules of the constituent BNs. Formally, a Kauffman Network is a graph G(V, F) defined by the set

$$V = \{x_1, x_2, \ldots, x_n\}. \tag{1}$$

that contains all the network's nodes, and the set

$$F = \left\{f_1^{(1)}, f_1^{(2)}, \ldots, f_j^{(i)} f_n\right\}, f_j^{(i)}\colon \{0,1\}^n \Delta \{0,1\}. \tag{2}$$

of sets of predictor functions, where the subindex j denotes the realization or constituent network and the subindex i denotes the predictor number, e.g., $f_2^{(1)}$ is the first predictor of the second constituent network of the PBN. Instead of a single predictor per node, we have one or more predictors, that can be selected to determine the future state of node x_j. The probability of selecting $f_j^{(i)}$ as the predictor for the node is given by $c_j^{(i)}$, where

$$\begin{gathered} 0 < c_j^{(i)} \leq 1, \sum_{i=1}^{l(j)} c_j^{(i)} = 1, \\ \forall j = 1, 2, \ldots, n. \end{gathered} \tag{3}$$

A useful metaphor is to think of the PBN as a tree of BNs, and each BN is selected with a particular probability.

Let f_i denote the ith possible realization of the network, with

$$f_i = \left(f_{i_1}^{(1)}, f_{i_2}^{(2)}, \ldots, f_{i_n}^{(n)}\right), \tag{4}$$

for every

$$1 \leq i_j \leq l(j), j = 1, 2, \ldots, n. \tag{5}$$

A realization of a PBN is one of its constituent BNs. The maximum number of realizations is given by

$$D = \prod_{j=1}^{n} l(j). \tag{6}$$

In [12], the authors validated that PBNs are appropriate for modeling engineering systems through a system model that was verified using model checking and the simulation results compared with real machine data. In [11], this methodology was applied to a manufacturing process, to gather quantitative occurrence data for DFMEA. In [10], the methods were further expanded including the application of PBNs in industrial manufacturing processes, using intervention (guided perturbations) as guide to move a system away from fault conditions and catastrophe, thus postponing its failure. A formal and thorough description of BN and PBN is presented in [21].

2.2. Reinforcement Learning. Artificial intelligence techniques are developing and growing rapidly. Methods like Deep Learning and Reinforcement Learning are helping with the complexities and uncertainty of power systems [37]. In this sense, there is a correlation between power systems and machine learning in order to predict the consumption [38], low prices [39], and energy optimization [40].

Born in the field of Behavioral Psychology, Reinforcement Learning (RL) [41, 42] is considered an area of Machine Learning (which can be defined as the design and analysis of algorithms that can improve on the base of experience) in the field of Computer Science. It is concerned with how agents should perform actions in a given environment such that they maximize a cumulative reward signal. In Reinforcement Learning, a learner, or agent, is not told what to do or which set of actions to take, rather it must discover the sequence of actions that achieve an optimal reward by trying them. The use of trial-and-error and delayed rewards are the two characteristic features of this approach. RL is studied in many other disciplines, such as control theory, operations research, statistics, and game theory. RL allows the software agent to learn a correct behavior based only on feedback from the environment, automating the learning scheme and extinguishing the need for human expertise and cutting the time needed to devise a solution. There are multiple solutions to a RL problem, but the most common approach is to allow the agent to select actions that yield a maximum reward in the long run, by using algorithms with an infinite horizon. One of the most used approaches is to make the agent learn to estimate the expected future rewards of *(action*, *states)* pairs. The estimates are adjusted through time by propagating part of the future state's reward, and if all states and all actions are tried numerous times, an optimal policy can be learned. Improved RL has been used for hybrid energy system management and optimization, e.g., SAC-based RL [43] and DDPG-based RL [44].

A RL agent learns by interacting with its environment. The RL agent acquires knowledge from the result of its interactions with the environment, instead of being taught explicitly, and selects its actions based on past interactions (called exploitation) or by making new choices (exploration). The reinforcement signal (mostly numerical in nature) it receives is a reward that encodes the success (or failure) of a given action's outcome, and it seeks to acquire knowledge by selecting actions that maximize the cumulative reward over time. Figure 2 illustrates the standard Reinforcement Learning Cycle.

In a standard Reinforcement Learning model, the learner is known as the agent, who makes decisions and is connected to its environment through perception and action. Agent and environment interact at a sequence of steps in time, t, and at each interaction step, our agent senses the environment for information and determines the state of its "world." Based on this information, the agent chooses and takes an action. The information of the state of the environment constitutes the input of our agent, and the action chosen by our agent becomes its output. The actions taken by the agent in each step change the state of its environment and its own state. A time step later, the state transition's value following the action taken is given to our agent by its environment as a numerical value, called reward.

Reinforcement Learning differs from supervised learning, another form of learning studied in machine learning where the agent learns from examples that are provided by a supervisor, which is external to it. A challenge that is present in RL and not in other learning methods is that we have to choose and/or balance exploration and exploitation. An agent that uses exploration discovers and tries new actions to see if they produce a greater or lesser reward, but an agent that uses exploitation uses preferred and tried actions that in the past have been successful at producing reward. It also considers the whole problem; in uncertain environments, the agent does not consider subproblems and sees how everything fits into the whole picture, starting with an agent that is complete, interactive, and with explicit goals, sensing aspects of its environment and choosing actions that influence it.

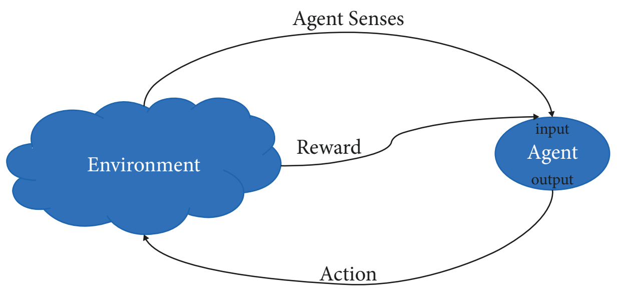

Figure 2: Standard Reinforcement Learning Cycle.

2.3. Supervised and Unsupervised Learning. A representative set of pairs of states and actions is provided by a teacher to the agent in supervised learning. The agent must modify its strategy for selecting actions so that its actions get closer every time to the selected target actions. Therefore, the main problem in supervised learning is then the approximation of a functional mapping from states to actions that is an unknown to the agent and known to the teacher, which can be done with neural networks, fuzzy systems, or other learning models. This is impractical for complex problems because of the inability to specify a representative set of pairs of states and actions, making finding optimal solutions unknown in some instances.

An agent that performs unsupervised learning in its purest form is perceiving the states of the process it has under control but will not get any information about the actions, and the control strategy is not evaluated. Unsupervised learning cannot be used to learn control strategies. A typical application of it is the identification of structure in data, as in data clustering.

2.4. Reinforcement Learning versus Purely Unsupervised Learning. Just as in unsupervised learning, our agent performs Reinforcement Learning and receives no information about an optimal strategy for control, but in RL the agent gets rewards or reinforcement signals provide feedback about its control strategy. With these signals, the agent can improve the strategy, giving intelligence to the trial-and-error process.

The problem faced by our agent in RL is that it must learn its behavior through trial-and-error interactions with its environment. Two main strategies are used for RL problem solving: an agent can choose to search in the behavior space to find a behavior that is appropriate to its environment (the approach used in genetic algorithms and genetic programming) or it can use statistics and dynamic programming to estimate a utility function of taking actions in states of its environment.

2.5. Main Components of a Reinforcement Learning System. In addition to the agent and the environment, the principal components of a Reinforcement Learning system are as follows:

(i) The **policy** dictates the way that our agent will behave at any given time. It maps states of the environment to a set of actions that are going to be taken when the states are reached. This policy is central to our agent since it is the only thing needed to determine the agent's behavior.

(ii) The **reward function** is the reinforcement signal and in our RL problem defines the goal of the agent, by mapping the perceived state or states of our world to a numerical value. This way we know which state is more desirable. In RL, our agent's only purpose in "life" is to maximize the total reward it will receive, and our agent must choose which actions contribute to that goal. Reward functions may be stochastic and are the basis for changing the policy of the agent. A strong assumption of the RL framework is that the reward signal can be unequivocally and directly observed, as the feedback the framework receives is part of the environment in which the agent is working on. However, rewards are often delayed as the effective reward is obtained several steps after the action leading to it has been executed. This fact notably increases the difficulty of training a RL agent.

(iii) There exist many RL algorithms, with different features and properties. One family of algorithms learn a **value function** that estimates the expected cumulative reward of a given state or an (action, state) pair. This way, it specifies what is better for our agent in the long run whether a given state is desirable considering the states that follow this one and the rewards available in them. They are secondary to rewards and serve as predictions of them. Action choices are based on values. Values are much harder to determine than rewards. Rewards are essentially given by the environment to the agent, whereas values are estimated and then re-estimated from the observations that an agent makes over time. Another family of algorithms simply tries to optimize the policy directly to achieve the maximum cumulative reward. Finally, there are hybrid methods that combine both approaches.

(iv) The **model** of the environment is a simulation of the behavior of the problem's environment. It is required by some RL algorithms although in practice it is only used in relatively simple problems.

2.6. Markov Decision Processes. Reinforcement learning problems are well modeled as Markov Decision Processes, or MDPs. Named after Russian scientist Andrey Markov, MDPs can be viewed as RL tasks that satisfy the Markov Property. When a stochastic process satisfies the Markov Property, it is memoryless. In other words, the conditional probability distribution of its future states only depends on the present state and not on the past. MDPs are discrete time stochastic control processes that are useful for studying and solving optimization problems through Dynamic Programming and RL. MDPs consist of the following:

(i) A set of states, **S**: the states are the inputs to our learning system. They represent all the information necessary to perform optimally.

(ii) A set of actions, **A.**

(iii) A reward function R: $S \times A \longrightarrow R$, R: $S \times A \longrightarrow R$: our learning system will execute an action in each state, and each action causes a transition. Because of the Markov Property, rewards are only dependent on the current and the successor state and do not depend on past information.

(iv) State-transition function T: $S \times A \longrightarrow \Pi(S) \vee \Pi(S)$ T: $S \times A \longrightarrow \Pi(S) \vee \Pi(S)$ is a probability distribution over **S.**

(v) Policies are sequences of mappings in the form Π: $\{\pi_0, \pi_1, \ldots\}$, Π: $\{\pi_0, \pi_1, \ldots\}$, where π_k, π_k maps the state $s_k \in \boldsymbol{S}$ to an action $a_k = \pi_k(s_k) \in \boldsymbol{A}(s_k)$. When both state and action spaces are finite, MDPs are said to be finite.

The Value Function, $V^{\pi}(s)$, in Reinforcement Learning is of extreme importance. It estimates the expected cumulative reward of state s. In MDPs, the value function can be defined as

$$\begin{aligned} V^{\pi}(s) &= E_{\pi}\{R_t \vee s_t = s\} \\ &= E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \vee s_t = s\right\}, \end{aligned} \tag{7}$$

where E_{π} { } defines the expected value when the agent follows the policy π, "R" is the reward, "s" is a state, and γ is the discount factor. The terminal state's value, if it exists, is zero. V^{π} is the state-value function for π. Most RL algorithms estimate value functions. A value function is a mapping of states that provide an estimate of how fit it is for the agent to be in a given state, defined in terms of the future rewards to be expected or the expected return. They are delineated with respect to a given policy. We also define $Q^{\pi}(s, a)$, the action-value function for π, as follows:

$$\begin{aligned} Q^{\pi}(s,a) &= E_{\pi}\{R_t \vee s_t = s, a_t = a\} \\ &= E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \vee s_t = s, a_t = a\right\}. \end{aligned} \tag{8}$$

where "a" represents action.

In many RL algorithms, the action-value function Q is used instead of the value function V because it easily lets the agent choose the action with higher expected rewards.

2.7. The Reinforcement Learning Problem. The RL agent interacts with its world in a series of time steps. With each discrete time step t, the agent receives an observation o_t and a reward r_t. An action at is chosen from the group of actions available to the agent and executed within the environment. This moves the environment to a new state s_{t+1}. A new reward for the transition and new state is now determined. The agent needs to accumulate as much rewards as possible.

The RL problem is defined as finding a policy for the agent that will specify the action that the agent will take when in a given state. Once an MDP combines with a policy as such, the action for each state is fixed and behaves like a Markov Chain. RL is not considered a technique for the solution but rather a way of formulating a problem [45]. In [46], the basic problem that is apt for the use of RL is formulated, where a system needs to interact with the environment to achieve a certain goal and based on the current state's feedback, what action should it perform next? RL is the way of learning the correct action to be taken in each situation based solely on feedback obtained from the environment [41]. For our purposes, feedback is a numerical reward that we assign to the actions that an agent takes. RL agents can learn offline or online. Offline learning is similar to the knowledge acquired by a student from a teacher; the agent is taught what is needed to know before venturing into the environment. Online learning is more spontaneous similar to the way a child learns how to walk, where knowledge is acquired in real-time. The agent explores its environment, and it is constantly adding experiences to make better future decisions.

3. Electrical Power Distribution Systems and Intelligent Power Routers

3.1. Electrical Power Distribution Systems. Electrical power generated at and transmitted from generation plants is the product of the transformation several energy sources (coal, natural gas, petroleum, nuclear, geothermal, solar, tidal, and so on) into electrical current [8]. Managing this vital resource is guided by the necessity to secure a stable and persistent supply of energy despite demand fluctuations. Electrical power plants have grown in capacity and size since they were first built over a hundred and fifty years ago. The generation of electrical power generally occurs distant from where it is ultimately consumed. Therefore, consumers are usually separated from electrical power plants by great distances.

Two distinguishable types of networks that interconnect consumers to generation sites exist. These types are as follows:

(i) Transmission networks: covering large areas, they make sure that most or all regions of a country are covered and provide the service. These high voltages (in the range of 230 kV or 138 kV) allow for the minimization of losses in its transmission. Different lines are bundled together at electrical power substations, and the networks eventually interconnect and feed power to distribution networks, that reach end-customers.

(ii) Distribution networks: these are engineered to provide power to smaller areas and have lower voltages than transmission grids but are denser because they are meant to serve electricity to the final customers. Lower voltages are used for safety and security reasons and due to installation costs. They are also able to provide several voltage levels to different end users, through transformers.

Industrial, commercial, and residential end-users must receive reliable electrical power at their facilities or homes. Several factors, natural and artificial, in the process of generating, transporting, and distributing electricity can damage equipment (wind, ice, storms (thunderstorms, typhoons, hurricanes), vegetation growth that can induce short circuits, and other nature-induced or human disasters, as well as malicious perturbations). Some factors cannot be predicted and must be taken care of in real time. Other events and factors can cause network unbalance. As an example, variations in temperature may cause changes in electrical loads, and overall demand for electrical power varies with time, season, weather, and so on. Some of these factors affect supplied power quality, while other factors cause emergency situations that force network operators to disconnect power to regions that cause problems to prevent chain reactions. Other severe situations may cause power outages or network power imbalance. Intentional power outages should be limited and minimized.

Electrical Power Grids are almost always managed from control rooms. Some can be telemetered, such that control engineers have accurate real-time information about their status. They can also have protection equipment that can be actioned from within the control room, so larger failures are prevented. There are instances in which telemetry may be cost ineffective, and aberrant network states can be reported by operators, engineers, workers, or customer communication. System repairs and maintenance may be performed manually by skilled workers.

Electrical substations [8] can have several busbars, and two of these may be interconnected via a switch or a conductor line. Both extremes of the power line are connected to a breaker. A breaker is a standard protection mechanism that has a relay that can automatically open in case of a short circuit, giving it the ability to disconnect all or a single circuit from the remaining network. Messages with alarms to control rooms can be generated as well, and with these, engineers can have the ability to control the state of the breakers. The main objective in fault management of an EPDS is to restore the power supply quickly to as many end users as possible. Since in an EPDS there may be different routes through which power can be served, the EPDS can be switched to select alternative routes through breakers and switches that can bypass the areas, lines, or devices that cause problems. The need arises to isolate and determine any malfunction in protective equipment, generate correct diagnoses from the alarm messages that are received, and continue to postulate a plan to restore electricity safely and efficiently to the largest number of end-users.

EPDSs [8] are a group of sources and power lines that operate under common supervision to provide electrical power to end-users. Systems for electrical power delivery are formed by joining Distribution and Transmission Systems. Transmission Networks are meant to transport high voltage electricity over longer distances. Their high voltage loads are reduced at major load centers and then distributed to customers, where distribution networks transport electricity from the Transmission Network to the customers. EPDSs are ubiquitous, from large ships to modern data centers. For our scope, we consider only Generation and Transmission Systems.

3.2. Intelligent Power Routers. IPRs [4] are the principal components of a smart grid that was developed as a distributed architecture for decentralized coordination, control, and communication between power system components. Intelligent control and planning of network operations are built into smart computing devices attached to sources, power lines, and other power network devices, thus allowing them to have a picture of current network conditions and assign resources to respond to failures, priority, or demand. They are configured on a Peer-to-Peer (P2P) network architecture, and in the event of a failure, they make local decisions and coordinate with other devices in their neighborhood to return the system into operation from an undesired state.

Currently, the control of electrical power generation and distribution, even if redundant generators and lines are present, is done in a centralized way. Future EPDS should be capable of distributing coordination and control of generation and distribution tasks throughout the network when contingencies or emergencies arise. IPRs were engineered for survivability, fault tolerance, scalability, cost-effectiveness, and continuous unattended operation. At its core, the IPR is a power flow controller with embedded software. An IPR has two principal components: Interfacing Circuits (ICKT) and an Integrated Control and Communications Unit (ICCU). The ICKTs operate power flow control and sensing devices, such as breakers, capacitors, and transformers. They can also receive network status information from sensors and dynamic system monitors. They have direct control of the ICCU, and with their logic and software, calculate how to route power, change loads, and take any corrective or preventive actions that enhance safety, stability, and security. The network architecture and communication protocols are similar to the Internet Protocol (IP) Local Area Networks. A load connected to an IPR can be assigned a priority, and contrary to nonsmart power networks, when a power source fails, the ICCU of an IPR reacts to this failure through reconfiguration of the network, so that the load with the highest priority may be served.

4. Materials and Methods

With the following methodology, faults and failures can be categorized for a single IPR's failure mode in an EPDS. We propose establishing the model using the Probabilistic Symbolic Model Checker (PRISM) [47], to verify its use and formal correctness using Probabilistic Computational Tree Logic (PCTL). The models were built in PRISM by constructing three modules: one for the environment in which the device operates, a module for the IPR's Probabilistic Boolean Network, and a reward structure. The actual state of the device PBN's nodes is in the second module, which uses the state of the variables available in the environment module and applies the corresponding Boolean Predictor Functions to transition to the next state. With the values of these variables as a base and the device's failure modes, the state of the IPR variables is changed, giving us the device's current state. In this manner, given the device's failure modes (which are based on the possible failure modes of its components), the model produces the failure modes corresponding to the system as an output.

To calculate individual IPR reliability, we have divided them into three principal subsystems: power hardware (power circuit breakers), computer hardware (used for IPR-to-IPR communications, routing, and CPU functions), and the software that manages the device. The reliability estimates of each of the subsystems that compose the IPR are provided in [4]. The reliability of a circuit breaker was obtained from data sheets as 0.99330. Each IPR has two circuit breakers, a main breaker and a redundant secondary. The reliability of data routers is estimated at 0.9009 (in a year). Lastly, software reliability is estimated at 0.99.

PBNs can precisely emulate an EPDS with IPRs since this has coincidences with GRNs that have been modeled with BNs and PBNs. As a first step, the PBN representing the EPDS is built. Each modeled component of the EPDS is equivalent to a gene (node) in a GRN, where a gene can assume one of two states; 0 means the IPR is ON and 1 means it is OFF (by the convention established in [4]). For each node, the Boolean functions that determine the state of its IPR in time $t+1$ are applied, given the state of the EPDS's nodes in time t.

In the next step, a matrix for every node is built, to construct its Predictor Function. When calculating the predictors, only relevant nodes, those directly affecting the status or state of the node under study, are considered. All nodes that do not directly affect the current node's state are ignored. As per the connections between relevant nodes and the observed node, the equation or set of equations (constructed with the basic Boolean operators) that determine the state of each node are presented. For every node, there exists a set of equations (one or more per node). These Boolean Functions are solved from the examination of the relationships between each node and its relevant nodes. All possible states of all relevant nodes are analyzed, and an evaluation is made about the next state of the node in time $t+1$, given the state of all relevant nodes in time t. This method proposed adapts the Fault Detection and Isolation (FDI) scheme described in [48] and shown in Figure 3, where a model is used for describing the normal operation of the process and another model is used to describe each of the faults or failure modes.

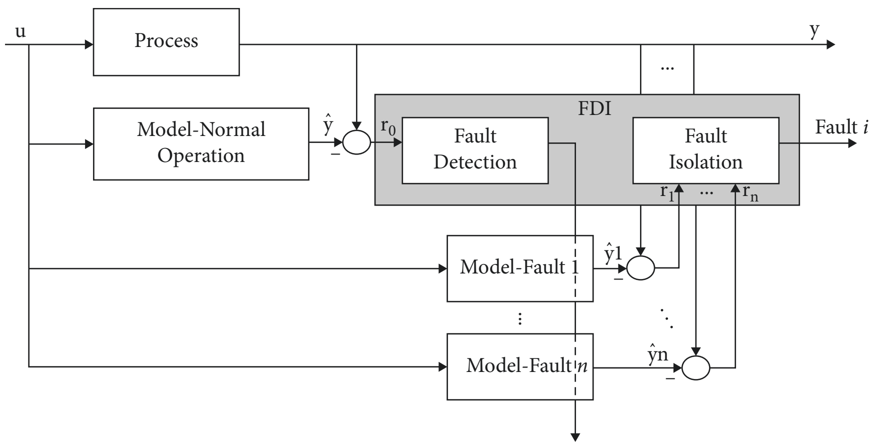

Figure 3: Fault detection and isolation (FDI) method from [48].

PBNs possess a characteristic called self-organization, and they do so into attractor states [36]. Attractors are sets of repeating states that, in the case of the models under study, are related to the failure modes that the system exhibits. There are similarities between the construction and semantics of the models we present and those in [11]. By characterizing the failure modes of the device under study, the models can, with model checking, characterize the state of their nodes to determine the faults and failures correlated to the device's fault conditions. This methodology is flexible, and the design of the network model and its state transitions depends on how much resolution the experts need, based on design specifications. Complexity and expression are scalable in this method, depending on the needs of the experts.

Device operation has been modeled by simulation of the network's components, taking into consideration the reliability analysis in [4]. These simulations were performed for the IPR by modeling of its relevant components based on the data of their Mean Time between Failures (MTBFs). The model can detect and isolate single and multiple IPR failure modes.

4.1. Classification. We begin by classifying the different states of the IPR's components, to properly create models of the device. Following the methodology in [48], we begin describing the system's failure modes, starting from its normal operation and continuing into modeling the different types of faults in the system.

Each subsystem is perceived as being in one of two different states:

Breakers:

0: the breakers can close/switch properly

1: the breakers do not close/switch properly

Router:

0: the data router communicates/sends-receives information in the network properly

1: the data router does not communicate/send-receive information in the network

Software:

0: the software makes correct decisions

1: the software makes incorrect decisions

The state of the device is, therefore, a set of states of its subsystems. There is redundancy in the breakers because all configurations of the IPR break into a series system, and the reliability of a series system is below the reliability of its lowest component. Therefore, the only way to increase the reliability of the IPR would be to provide a redundant path to the breaker. The device can be in 16 states (Router, Software, Breaker1, and Breaker2) that go from all subsystems operational (0000) to all subsystems in failure (1111). Some of these states, such as the failure of a single breaker, are identical, and after merging, there are 12 unique states. Table 1 summarizes the Categories, Types, and states that constitute these categories in the IPR.

Failure probability for each component is assumed to be independent of each other. Reliability estimates for each of the device's components was detailed in "Electrical Power Distribution Systems and Intelligent Power Routers" Section.

The relevant genes of the IPR's PBN are its data router, software, and the main and secondary breakers [8]. For these, the state of their components determines the failure mode they are currently on, as per the categories. Category 1 is a type of fault, where the IPR acts appropriately and changes the state of the breakers on an Active Signal (AS) but may change them when switching is unnecessary. Category 2 describes the normal operation mode of the IPR. Category 3 describes a failure (catastrophic) of this device. Lastly, Category 4 describes a fault condition on which the device does not act upon an AS and may also switch the breakers unnecessarily when there is no AS. Table 2 presents the predictor Boolean functions for every IPRs subsystems, based on their configuration.

This permits the prognosis of fault conditions those that do not cause a total failure but rather failure modes that will lead to instances where the device continues its operation but does not perform the required task to specifications. These are unhealthy states of the device, and they should be

Table 1: IPR failure mode classification and their corresponding states [4].

Category	1	2	3	4
Type	Fault	Normal operation	Failure	Fault
Description	On active signal (AS, switching event), the IPR works as intended. On inactive signal (IS, nonswitching event), IPR does not work as intended.	On AS, the IPR works as intended. On IS, the IPR works as intended.	On AS, the IPR does not work as intended. On IS, the IPR does not work as intended.	On AS, the IPR does not work as intended. On IS, the IPR works as intended.
States	S9	S0	S3, S4, S10	S1-2, S5-8, S11

Table 2: Predictors and selection probability, IPR PBN.

Component	Predictor	Selection probability $c_j^{(i)}$
x_1, software	$x_1(t+1) = x_1(t)$	1
x_2, router	$x_2(t+1) = x_1(t) \land x_2(t)$	0.9611
	$x_2(t+1) = x_1(t) \lor x_2(t)$	0.0389
x_3, main breaker	$x_3(t+1) = x_1(t) \land x_3(t)$	0.9611
	$x_3(t+1) = x_1(t) \lor x_3(t)$	0.0389
x_4, secondary breaker	$x_4(t+1) = x_1(t) \land x_4(t)$	0.9611
	$x_4(t+1) = x_1(t) \lor x_4(t)$	0.0389

treated, or they will otherwise lead to failure. For the device under study, the failure modes described in [4] were used, and an expert determination was made as to which device components and failure modes produce a failure or a fault.

PRISM's Property verification in PCTL was used to determine the maximum probability of occurrence of the failure modes that could evolve to a fault or a failure. From an initial state for the IPR, such as Category 2, a determination is made about the maximum probability of reaching one of the different identified failure modes. Property verification in PRISM permits the verification the models, and they also permit, through experiments, to reach an estimate regarding when in time a fault is certain.

5. Results and Discussion

PRISM [47] was used to validate the model quantitatively. These experiments were performed using a PBN representation of the IPR. Its main components (router, software, and breakers) were modeled, and their interrelationships are expressed as Boolean Functions or predictors. These components are considered the PBN's nodes, which give as output the overall state of the device. In the experiments, time is expressed in hours (h). A reward is assigned to the interaction of the PBN agent with its environment. A reward of '1' has been assigned to the state in which all components of the IPR are operating correctly. In this way, the agent can obtain a feedback signal, based on its actions. The main objective of the PBN-RL agent is to remain in a normal operating state through its operation.

We performed reward-based property experiments to test the model's capacity to emulate the standard RL cycle. We studied the agents' combined actions in the environment, and we also studied the actions individually. The experiments assess the model's capacity to perform the standard Reinforcement Learning cycle. In the standard RL cycle, an agent interacts with its environment and receives feedback upon performing actions in the form of a reward or cost.

Thus, a PBN-based model was established in PRISM using an MDP, with a module for the environment and a module for the PBN, with its predictors. PBNs used as GRN models have been expressed and developed as MDP [1, 2, 5, 19, 37, 38]. The actions of the model correspond to the different states in which the model can assume, which are correlated to the classifications previously presented. These classifications correspond to the device's different failure modes, as per the reliability analysis. In this scheme, the only missing element would be to assign a reward for the actions that are to be reinforced, so that the agent can receive feedback from its environment. PRISM has a rewards structure that can be used within the model's specification to assign rewards to states or sets of states. Currently, all assigned rewards in PRISM must be positive, and therefore we do not assign penalties or costs to states or sets of states. It is possible, however, to create multiple reward structures within the same model. With these multiple reward structures, we can analyze the effect of the different actions in the model. We are also able to conduct reward-based experiments that can provide information about the maximum cumulative reward over time, for a particular action or state. We have studied the effect of these rewards separately because although we value the benefits of model checking, we are unable to assess the effect of assigning costs and rewards at the same time with this tool. We understand that the current benefits of the use of model checking outweigh its limitations.

The first experiment conducted was performed to determine the maximum expected reward *Rmax* for the agent interacting in its environment and executing any of its actions. The rewards structure assigns a reward of '5' to the normal operation mode, a reward of '1' to any of the fault modes, and a reward of '0' to the failure of the IPR (a higher reward for the action that we would like the device to reinforce more). This was executed through verification of the following property:

$$''R\text{max} = ?\,[c < \,= \text{time}]'', \tag{9}$$

where Rmax is the maximum reward property operator and 'C' is the operator for Cumulative Reward in PRISM.

Figure 4 presents the results of this experiment to assess the maximum expected reward of the agent when interacting with its environment in the standard RL cycle.

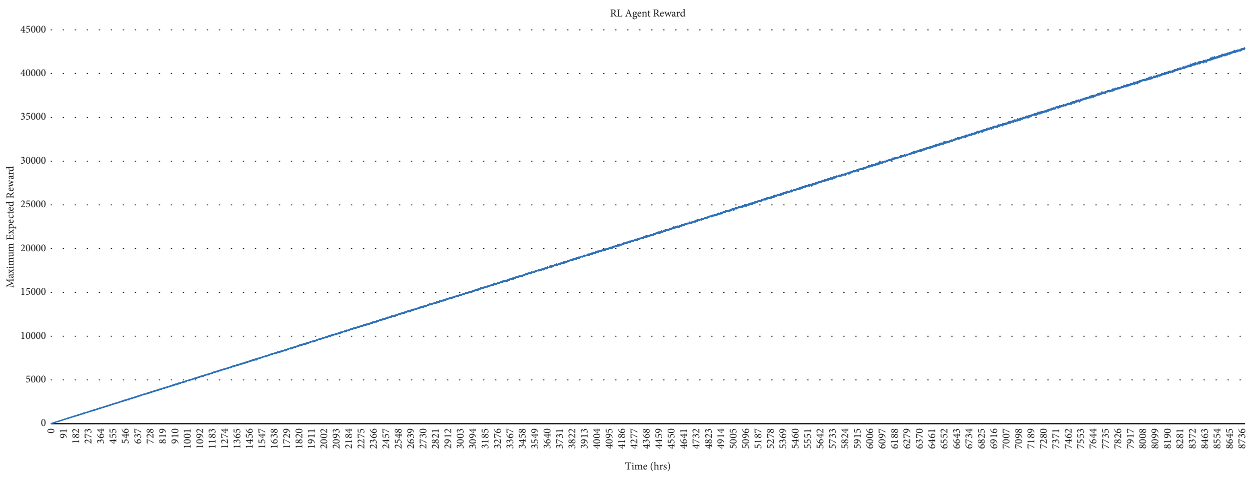

Figure 4: Maximum expected reward for the RL agent interacting with the environment.

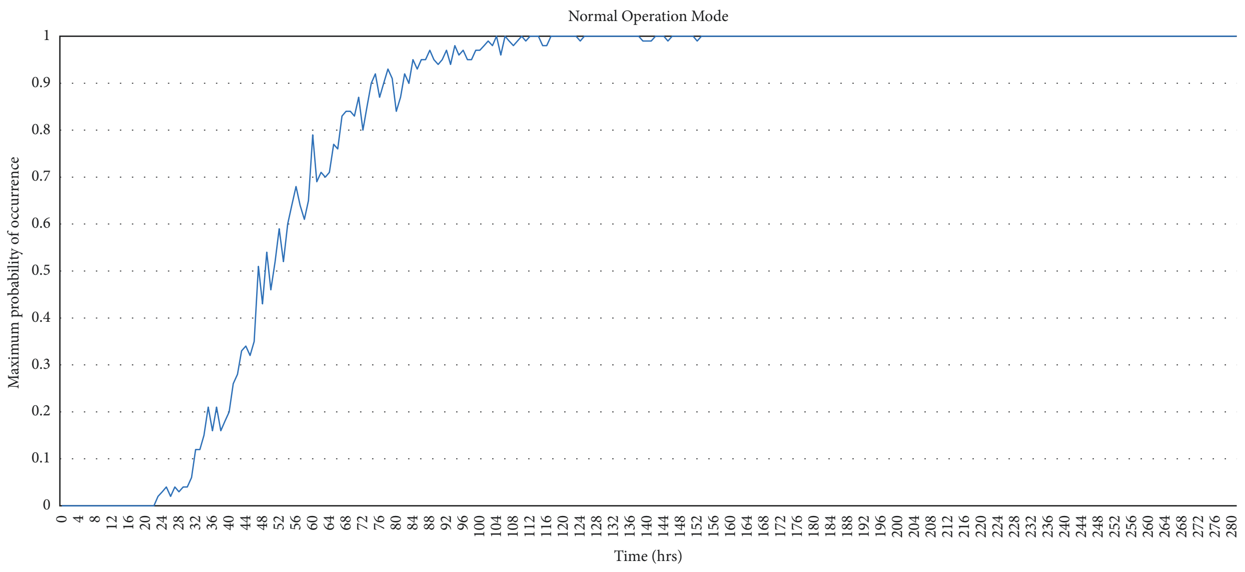

Figure 5: Maximum probability of occurrence of the normal operation mode.

In this experiment, the final value of the reward is 42946. The agent performs its actions and receives feedback in the form of a reward from each action from the environment, resulting in the linear plot from Figure 4. To recall, Figure 5 presents a graph of the maximum probability of occurrence of the normal operation action that can be seemingly approximated by a sigmoid curve. Since this property reaches 100% probability quickly, a year of operation is not plotted.

Figure 5 can also be approximated by a sigmoid curve. Figure 6 presents the maximum expected reward for the normal operation action. It shows a plot of the maximum reward obtained for the normal operation mode of the IPR in time, over a period of one year of operation. The final value of the maximum cumulative reward for this action is 8620, which translates into the device receiving a reward of ‘1’ for every hour of normal operation or 8,620 hours of normal device operation in the simulation.

This is the action with the largest cumulative reward, as the set of states that are part of this classification have the highest probability of occurrence. The rewards for all the other actions presented have a smaller cumulative reward as these actions also have a lower probability of occurrence. We can adjust the scale of the maximum occurrence experiment in Figure 6 to match the time axis to Figure 5. Figure 7 shows this adjustment.

The reward initially is low in the very early hours of operation, corresponding to the early failures (or infant mortality) period, and as the device enters a steady state, its expected reward rate increases almost linearly.

As PRISM supports multiple reward structures within a single model, each reward structure needs to be identified when running an experiment, as

$$''R\{''\text{normop}''\} = ?[c< \ = \text{time}]'', \tag{10}$$

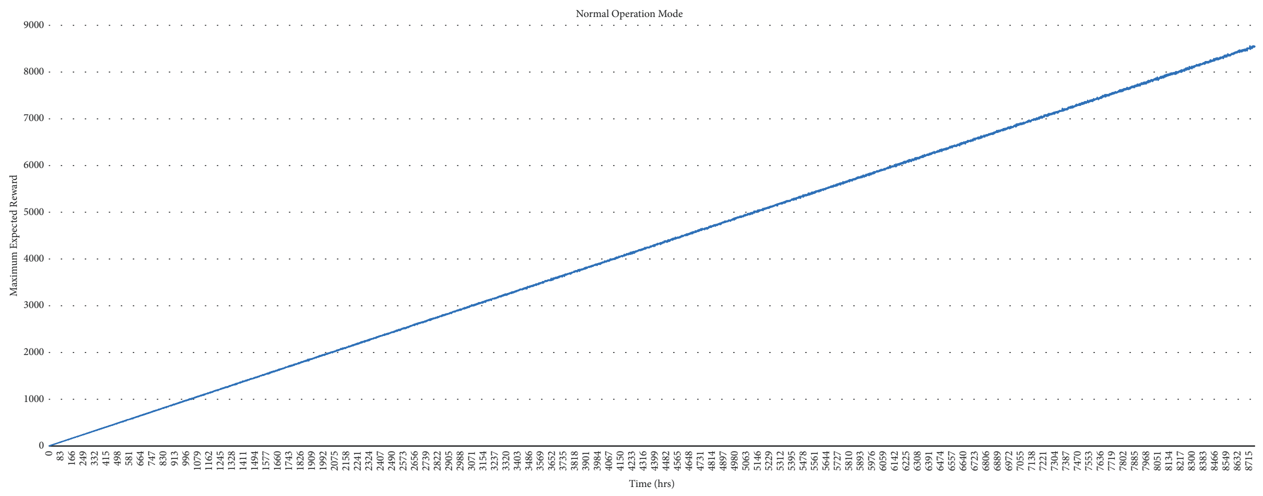

Figure 6: Maximum reward obtained for the normal operation mode of the IPR in one year of operation.

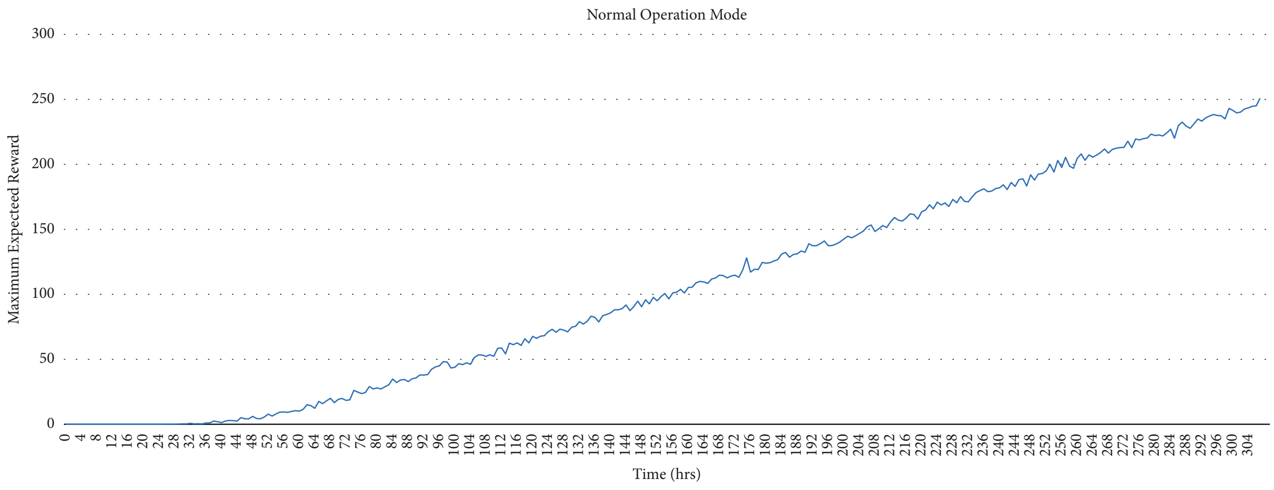

Figure 7: Maximum reward for normal operation, adjusted scale.

where the property used the “normop” (for normal operation) reward structure of the model. The rest of the experiments was executed with similar properties.

Figure 8 presents a plot with the results of a maximum cumulative reward for the failure of the IPR over a year of operation. The final value for the maximum cumulative reward is 73, which reflects a total of 73 hours for a period of one year in which the device was in a catastrophic failure in the simulation.

In Figure 9, the results of a maximum expected reward experiment for the Fault 1 operating mode of the IPR over a year of operation are presented in a plot. The maximum cumulative reward was 6, which indicates a total of 6 hours in which the device was in a type 1 fault in the simulation.

Figure 10 presents the result of the maximum cumulative reward for the Fault 2 operation mode of the IPR over a year of operation, and this was found to be 118, reflecting 118 hours in the simulation that the device spent in a Type 2 fault.

These results demonstrate that the PBN model of the IPR can correctly emulate the standard RL cycle; as for every iteration in time, the variables’ states are assessed and compared with the failure modes of the device, and a reward is assigned to the actions that the user wants to reinforce. The results of these experiments are exportable in PRISM as a file that can later be used to analyze the model’s behavior using statistical packages or machine learning tools.

In these systems, modeled with complex systems tools, learning occurs in the fundamental sense of adaptation to changes; the system adapts to survive. Complex systems self-organize into steady states, which are the long-term behavior of the system, and this self-organization in the most basic sense is a form of learning (considering learning as a type of adaptation and self-organization as an adaptive mechanism). The evolution of these complex systems can be controlled externally through interventions [11] so that the system can avoid “unhealthy” states (failures or faults).

In a standard RL model, the agent makes decisions and is connected to its environment through perception and action. Agent and environment interact during a sequence of time steps, and at each interaction step, our agent senses the environment for information and determines the state of its

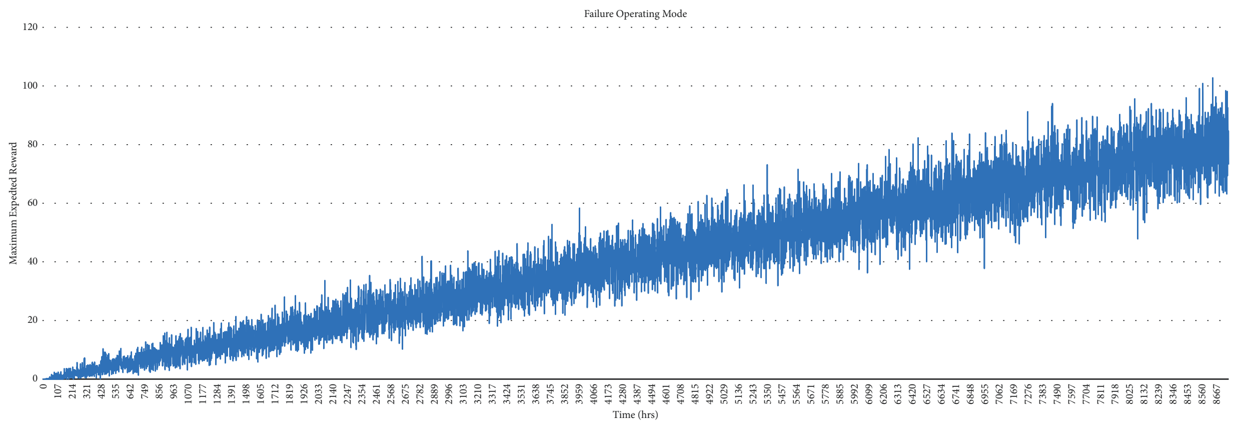

Figure 8: Maximum expected reward obtained for the failure operation mode of the IPR in one year of operation.

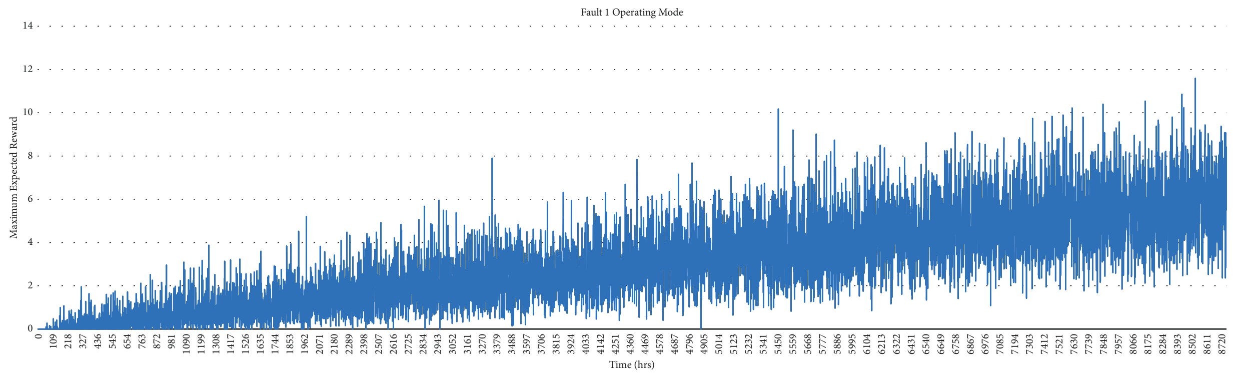

Figure 9: Maximum reward obtained for the Fault 1 operation mode of the IPR in one year of operation.

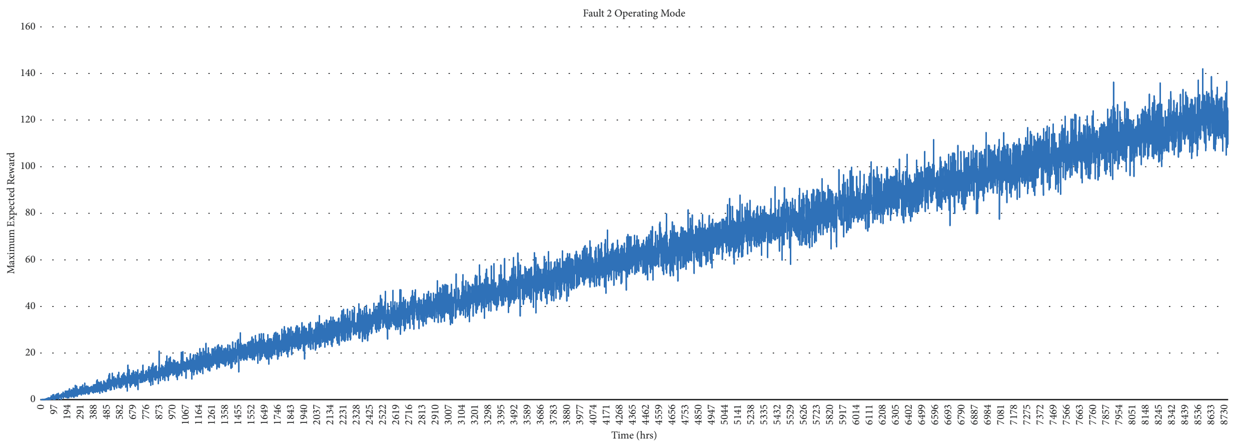

Figure 10: Maximum reward obtained for the Fault 2 operation mode of the IPR in one year of operation.

"world." Based on this information, the agent takes an action. The information of the state of the environment constitutes the input of our agent, and the action chosen by our agent becomes its output. The actions taken in each step change the state of its environment and its own state. A time step later, the state transition's value following the action taken is given to our agent by its environment as a reward. The model is an RL agent acting on its environment and receiving a reward that reinforces a certain behavior over others. Rewards were assigned to the IPR RL Agent acting upon the information on its environment. A reward of '1' was assigned to the state in which all of the IPR components are operating correctly (Cat. 2) and no rewards for other failure modes. A rewards-based property in PRISM was used to run an experiment in which an IPR is operating continuously for a year, and we obtained the maximum reward assigned to that action. In Figure 4, the RL agent's goal is to remain in a normal operating mode. If instead of positively rewarding the correct operation of the IPR, we rewarded one of its failure modes, such as a Category 1 Failure mode, the resulting experiment would yield a maximum reward as per Figure 9. The maximum rewards obtained are directly correlated with the estimated reliability of the IPR in [4]. Therefore, this reward result is related to the occurrence of the Category 1 failure mode. Validation through experiments that PBN modeled systems can perform the standard RL cycle is an important step towards an automatic way of system control through Machine Learning, where control is not external but intrinsic to the system.

6. Conclusions and Future Work

In this work, we have studied a smart grid management device, the Intelligent Power Router, assessed the device's reliability, and presented a bioinspired modeling technique that uses Probabilistic Boolean Networks to create simple and logical models that exhibit complex behavior. These models self-organize into constituent Boolean networks with attractor cycles that can describe long-term system behavior. In this case, this behavior is equivalent to the different states in which the device's components can assume, and, therefore, to the different failure modes of the device. We used PBNs to model Intelligent Power Routers, a smart grid component, with Reinforcement Learning, and we studied the model's evolution in time and how they may learn to avoid undesirable states in an autonomous way, increasing their reliability and resilience. We proposed PBNs as a building block with a novel analysis technique for problem solving in smart grid modeling, through RL. We validated and verified through model checking (MC) the viability of this methodology.

We believe that many areas of future work are available for the unanswered questions in this research. Particularly, we have only explored modeling the standard RL cycle, but we infer it is possible to use other RL techniques, such as Q-Learning and deep Q-Learning, to endow the system with automatic machine learning-based control of its evolution. There is a more fundamental question that can be answered through the study of RL in PBN modeling that an alternative to artificial neural networks can be achieved using PBNs as the building block for this structure. To achieve this, artificial neural network neurons need to be proven equivalent to the set of nodes of a PBN, which have input and output states. The set of input nodes have a set of predictor functions that define the output state (like the threshold function). The learning task would be to change the transition probabilities to select a context (constituent BN) that represents the state in which the network must be steady state, where states are the goals to be achieved in the system.

Data Availability

Access to data is restricted due to 3rd party rights.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] A. Grigoriev, V. V. Skorlygin, S. A. Grigoriev, D. A. Melnik, and M. N. Filimonov, "A hybrid power plant based on renewables and electrochemical energy storage and generation systems for decentralized electricity supply of the northern territories," *Int. J. Electrochem. Sci*, vol. 13, pp. 1822–1830, 2018.

[2] N. El Assri, S. Chabaa, K. Lmesri, M. A. Jallal, and A. Zeroual, "Modeling techniques for decentralized energy systems applied in smart grids," in *E3S Web of Conferences*vol. 297, EDP Sciences, Article ID 01068, 2021.

[3] European Union Commission Task Force on Smart Grids, "European Union Commission Task Force on Smart Grids," 2009, https://www.usef.energy/implementations/smart-grids-task-force-eg3.

[4] A. A. Irizarry-rivera, M. Rodríguez-Martínez, B. Vélez et al., "Intelligent power routers: distributed coordination for electric energy processing networks," in *Operation and Control of Electric Energy Processing Systems" James Momoh and Lamine Mili*, pp. 47–85, John Wiley and Sons, Hoboken, NJ, USA, 2010.

[5] M. Liserre, G. Buticchi, J. I. Leon et al., "Power routing: a new paradigm for maintenance scheduling," *IEEE Industrial Electronics Magazine*, vol. 14, no. 3, pp. 33–45, 2020.

[6] C. Gershenson, "Facing complexity: prediction vs. adaptation," in *Complexity Perspectives on Language, Communications and Society*, A. Massip and A. Bastardas, Eds., Springer-Verlag, Berlin, Germany, pp. 3–14, 2013.

[7] R. Torres, J. Pedro, and S. Kanaan Izquierdo, "Contributions to Reinforcement Learning through Probabilistic Boolean Networks," in *Proceedings of the a Poster Presented in the Conference on Complex Systems*, Thessaloniki, Greece, October 2020.

[8] P. J. Rivera Torres and O. Llanes Santiago, "Fault Detection and isolation in smart grid devices using probabilistic boolean networks," in *Computational Intelligence in Emerging Technologies for Engineering Applications*, O. Llanes Santiago, C. Cruz Corona, A. J. Silva Neto, and J. L. Verdegay, Eds., vol. 872, pp. 165–185, Springer-Nature, London, UK, 2020.

[9] R. Torres, A. J. S. Neto, J. Pedro, Silva Neto, J. Antônio, and O. Llanes Santiago, "Multiple fault diagnosis in manufacturing processes and machines using probabilistic boolean networks," in *Advances in Intelligent Systems and Computing*, F. Martínez Álvarez, A. Troncoso Lora, J. Sáez Muñoz, H. Quintián, and E. Corchado, Eds., vol. 950, pp. 355–365, Springer, Berlin, Germany, 2019.

[10] Rivera-Torres, J. Pedro, E. I. Serrano Mercado, O. Llanes Santiago, and L. Anido Rifon, "Modeling preventive maintenance of manufacturing processes with probabilistic boolean networks with interventions," *Journal of Intelligent Manufacturing*, Springer, vol. 29, no. 8, pp. 1941–1952, 2018.

[11] Rivera-Torres, J. Pedro, E. I. Serrano Mercado, and L. Anido Rifon, "Probabilistic Boolean network modeling and model checking as an approach for DFMEA for manufacturing systems," *Journal of Intelligent Manufacturing*, Springer, vol. 29, no. 6, pp. 1393–1413, 2018.

[12] Rivera-Torres, J. Pedro, E. I. Serrano Mercado, and L. Anido Rifon, "Probabilistic boolean network modeling of an

industrial machine," *Journal of Intelligent Manufacturing*, Springer, vol. 29, no. 4, pp. 875–890, 2018.

[13] R. Torres, J. Pedro, and O. Llanes Santiago, "Model-based Fault Diagnosis of Manufacturing Processes and Machines Using Probabilistic Boolean Networks," in *Proceedings of the 19th Scientific Conference of Engineering and Architecture, ISP-CUJAE*, La Habana, Cuba, November 2018d.

[14] R. Torres and J. Pedro, *Contribuciones al modelado teórico de sistemas de fabricación mediante Redes Booleanas Probabilísticas*, Doctoral Dissertation, Universidade de Vigo, Spain, March, 2017.

[15] R. Torres, J. Pedro, and E. I. Serrano Mercado, "Probabilistic Boolean Network Modeling as an Aid for DFMEA in Manufacturing Systems," in *Proceedings of the 18th Scientific Conference of Engineering and Architecture, ISP-CUJAE*, La Habana, Cuba, November 2016.

[16] Rivera-Torres, J. Pedro, J. Seguel, M. Rodríguez-Martínez, Irizarry-Rivera, and Agustín, "Formal Methods for the Design and Analysis of Electrical Power Distribution Systems Endowed with Intelligent Power Routers," Final Report for PO 4100307844, Lockheed-Martin, Bethesda, MD, USA, 2012.

[17] S. A. Kauffman, "Metabolic stability and epigenesis in randomly constructed genetic nets," *Journal of Theoretical Biology*, vol. 22, no. 3, pp. 437–467, 1969.

[18] S. A. Kauffman, "Homeostasis and differentiation in random genetic control networks," *Nature*, no. 5215, pp. 177-178, 1969.

[19] Y. L. Orlov and A. V. Baranova, "Editorial: bioinformatics of genome regulation and systems biology," *Frontiers in Genetics*, vol. 11, p. 625, 2020.

[20] I. Shmulevich, E. Dougherty, and S. Kim, "Probabilistic Boolean Networks: A Rule-Based Uncertainty Model for Gene Regulatory Networks," *Bioinformatics*, vol. 18, 2002.

[21] I. Shmulevich and E. R. Dougherty, *Probabilistic Boolean Networks: Modeling and Control of Gene Regulatory Networks*, SIAM, Philadelphia, PA, USA, 2010.

[22] I. Shmulevich, E. R. Dougherty, S. Kim, and W. Zhang, "From boolean to probabilistic boolean networks as models of genetic regulatory networks," *Proceedings of the IEEE*, vol. 90, no. 11, pp. 1778–1792, 2002b.

[23] D. N. Arnosti and A. Ay, "Boolean modeling of gene regulatory networks: driesch redux," *Proceedings of the National Academy of Sciences*, vol. 109, no. 45, pp. 18239-18240, 2012.

[24] B. Vasic, V. Ravanmehr, and A. R. Krishnan, "An information theoretic approach to constructing robust Boolean gene regulatory networks," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 9, no. 1, pp. 52–65, 2012.

[25] G. Didier and E. Remy, "Relations between gene regulatory networks and cell dynamics in Boolean models," *Discrete Applied Mathematics*, vol. 160, no. 15, pp. 2147–2157, 2012.

[26] F. Ghanbarnejad, *Perturbations in Boolean Networks as Model of Gene Regulatory Dynamics (Doctoral Thesis)*, University of Leipzig, Leipzig, Germany, 2012.

[27] C. Chaouiya, O. Ourrad, and R. Lima, "Majority rules with random tie-breaking in boolean gene regulatory networks," *PLoS ONE*, vol. 8, no. 7, Article ID e69626, 2013.

[28] W.-K. Ching, X. Chen, and N.-K. Tsing, "Generating probabilistic boolean networks from a prescribed transition probability matrix," *IET Systems Biology*, vol. 3, no. 6, pp. 453–464, 2009.

[29] K. Kobayashi and K. Hiraishi, "Reachability analysis of probabilistic boolean networks using model checking," in *Proceedings of the SICE Annual Conference 2010*, pp. 829–832, Taipei, Taiwan, August 2010.

[30] X. Chen, H. Jiang, and W.-K. Ching, "On construction of sparse probabilistic boolean networks," *East Asian Journal on Applied Mathematics*, vol. 2, no. 1, pp. 1–18, 2012.

[31] Y. m. Gao, P. Xu, X. h. Wang, and W. b. Liu, "The complex fluctuations of probabilistic boolean networks," *BioSystems*, vol. 114, no. 1, pp. 78–84, 2013.

[32] P. Trairatphisan, A. Mizera, J. Pang, A. A. Tantar, J. Schneider, and T. Sauter, "Recent development and biomedical applications of probabilistic boolean networks," *Cell Communication and Signaling*, vol. 11, no. 1, p. 46, 2013.

[33] H. Chen and J. Sun, "Stability and stabilisation of context-sensitive probabilistic boolean networks," *IET Control Theory & Applications*, vol. 8, no. 17, pp. 2115–2121, 2014.

[34] I. Shmulevich and E. R. Dougherty, *Genomic Signal Processing*, Princeton University Press, vol. 1p. 1, 1st edition, Princeton, NJ, USA, 2007.

[35] C. Gershenson, "Introduction to random boolean networks," in *Proceedings of the Workshop and Tutorial Proceedings, Ninth International Conference on the Simulation and Synthesis of Living Systems (Alife IX)*, M. Bedau, T. Hutton, S. Kumar, and H. Suzuki, Eds., pp. 160–173pp. 160–, Boston, MA, USA, September2004.

[36] C. Gershenson, "Guiding the self-organization of random Boolean networks," *Theory in Biosciences*, vol. 131, no. 3, pp. 181–191, 2012.

[37] D. Zhang, X. Han, and C. Deng, "Review on the research and practice of deep learning and reinforcement learning in smart grids," *CSEE Journal of Power and Energy Systems*, vol. 4, no. 3, pp. 362–370, 2018.

[38] H. T. Zhang, F. Y. Xu, and L. Zhou, "Artificial neural network for load forecasting in smart grid,"vol. 6, pp. 3200–3205, in *Proceedings of the 2010 International Conference on Machine Learning and Cybernetics*, vol. 6, IEEE, Qingdao, China, (July2010.

[39] X. He, T. Huang, C. Li, H. Che, and Z. Dong, "A recurrent neural network for optimal real-time price in smart grid," *Neurocomputing*, vol. 149, pp. 608–612, 2015.

[40] F. Mountassir, R. Mali, and M. Bousmah, "Machine learning au service de la prédiction de la demande d'énergie dans les smart grid," *Mediterranean Telecommunications Journal*, vol. 8, no. 2, 2018.

[41] R. S. Sutton and A. G. Barto, *Macro-Actions in Reinforcement Learning: An Empirical Analysis' by Amy McGovern and Richard S*, University of Massachusetts Amherst, Amherst, MA, USA, 1998.

[42] R. S. Sutton and A. G. Barto, "IEEE Transactions on Neural Networks," *Reinforcement Learning: An introduction*, MIT Press, vol. 9, no. 5, , p. 1054, Cambridge, MA, USA, 1998.

[43] J. Wu, Z. Wei, W. Li, Y. Wang, Y. Li, and D. U. Sauer, "Battery thermal- and health-constrained energy management for hybrid electric bus based on soft actor-critic DRL algorithm," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 6, pp. 3751–3761, 2021.

[44] J. Wu, Z. Wei, K. Liu, Z. Quan, and Y. Li, "Battery-involved energy management for hybrid electric bus based on expert-assistance deep deterministic policy gradient algorithm," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 12786–12796, 2020.

[45] A. K. McCallum, *Reinforcement Learning with Selective Perception and Hidden State*, Ph.D. Dissertation, University of Rochester, Rochester, NY, USA, 1996.

[46] G. A. Rummery, “Problem Solving with Reinforcement Learning,” Ph.D. Dissertation, Cambridge University, Cambridge, UK, 1995.

[47] M. Z. Kwiatkowska, G. Norman, and D. Parker, “Prism 4.0: verification of probabilistic real-time systems,” *Computer Aided Verification*, vol. 6806, pp. 585–591, 2011.

[48] L. F. Mendonça, J. M. Sousa, and J. M. Sá da Costa, “An architecture for fault detection and isolation based on fuzzy methods,” *Expert Systems with Applications*, no. 2, pp. 1092–1104, 2009.