Energy and AI 23 (2026) 100672

Contents lists available at ScienceDirect

Energy and AI

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

Evolving power system operator rules for real-time congestion management

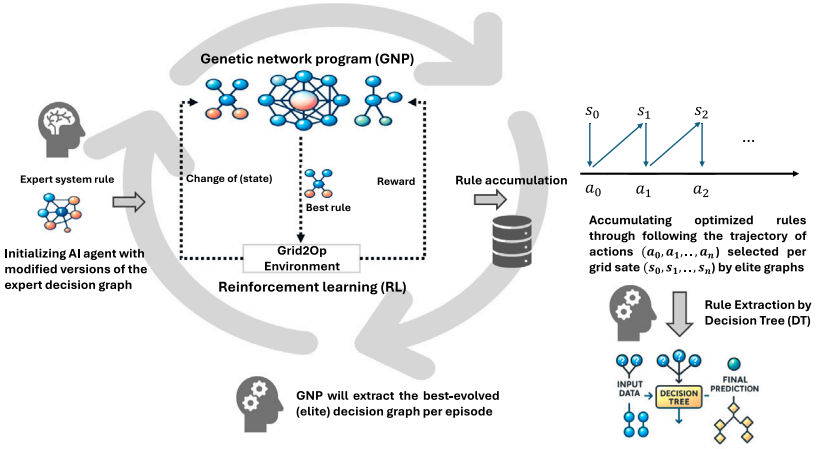
Ferinar Moaidi [a,b,*], Ricardo J. Bessa [b]

[a] Faculty of Engineering of the University of Porto (FEUP), Campus da FEUP, Rua Dr. Roberto Frias, Porto, 4200-465, Portugal
[b] Institute for Systems and Computer Engineering, Technology and Science (INESC TEC), Campus da FEUP, Rua Dr. Roberto Frias, Porto, 4200-465, Portugal

HIGHLIGHTS

- New genetic network programming variant that learns from data via reinforcement learning.
- Hybrid graph and decision tree method provides clear decision logic and interpretability.
- Time-efficient and scalable to large power grids with contingency scenarios and renewables.
- Outperforms expert systems and deep reinforcement learning agents on the IEEE 118-bus system.

GRAPHICAL ABSTRACT



ARTICLE INFO

Dataset link: GitHub repository of the AI4REAL NET European project

Keywords:
Congestion management
Expert systems
Genetic network programming
Interpretability
Reinforcement learning
Remedial actions

ABSTRACT

The growing integration of renewable energy sources and the widespread electrification of the energy demand have significantly reduced the capacity margin of the electrical grid. This demands a more flexible approach to grid operation, for instance, combining real-time topology optimization and redispatching. Traditional expert-driven decision-making rules may become insufficient to manage the increasing complexity of real-time grid operations and derive remedial actions under the N-1 contingency. This work proposes a novel hybrid AI framework for power grid topology control that integrates genetic network programming (GNP), reinforcement learning, and decision trees. A new variant of GNP is introduced that is capable of evolving the decision-making rules by learning from data in a reinforcement learning framework. The graph-based evolutionary structure of GNP and decision trees enables transparent, traceable reasoning. The proposed method outperforms both a baseline expert system and a state-of-the-art deep reinforcement learning agent on the IEEE 118-bus system, achieving up to an 28% improvement in a key performance metric used in the Learning to Run a Power Network (L2RPN) competition.

1. Introduction

Real-time congestion management in transmission networks, particularly under contingency scenarios, is becoming increasingly complex due to a convergence of emerging challenges such as the variability and uncertainty of renewable energy sources (RES), the rising frequency and intensity of extreme weather events, and the increasing risk of cyberattacks on critical infrastructures. In this demanding context,

* Corresponding author at: Institute for Systems and Computer Engineering, Technology and Science (INESC TEC), Campus da FEUP, Rua Dr. Roberto Frias, Porto, 4200-465, Portugal.
E-mail addresses: ferinar.moaidi@inesctec.pt (F. Moaidi), ricardo.j.bessa@inesctec.pt (R.J. Bessa).

https://doi.org/10.1016/j.egyai.2025.100672
Received 30 June 2025; Received in revised form 29 October 2025; Accepted 21 December 2025
Available online 6 January 2026

control room operators are required to rapidly assess system conditions and define effective remedial actions to maintain grid stability, prevent overloads, and mitigate the risk of cascading failures. These remedial actions may involve generation redispatch, RES curtailment, demand response, and topological reconfiguration of the network. The large number of possible action combinations, coupled with the need for real-time decision-making, makes it extremely challenging to identify optimal solutions, which often requires the use of approximate or heuristic methods to support timely and reliable operations [1].

This work builds upon the longstanding concept of expert systems (ES) in power systems, traditionally rule-based approaches that integrate expert domain knowledge with physics-based models, and introduces a data-driven methodology to enhance these systems. Specifically, focusing on the real-time congestion management problem, it adopts the formalism of Markov decision processes as established in the Learning to Run a Power Network (L2RPN) competition [2], created by RTE (the French transmission system operator), and proposes the augmentation of ES by using reinforcement learning (RL), enabling the ES to learn and adapt optimal policies from experience and interaction with the environment.

1.1. Literature review

Early research on network topology control used heuristics to reduce congestion. Mazi et al. relied on engineering judgment and sensitivity analysis for switching actions [3]. Bacher and Glavitsch modeled switching in optimal power flow (OPF) to minimize losses, though their method primarily focuses on static network snapshots rather than dynamic system behavior, limiting its adaptability to real-time operation [4]. Makram et al. introduced a Z-matrix strategy to enhance efficiency, but it lacked optimality guarantees and often omitted voltage constraints [5]. These early efforts, while foundational, were limited in scalability and real-time applicability, making them less suited to modern grids that demand dynamic, constraint-aware, and computationally efficient control strategies. During the 1990s, research increasingly addressed dynamic aspects of switching strategies. Chen and Glavitsch, for example, investigated switching actions aimed at enhancing system stability during emergencies, with a focus on transient dynamics [6]. While pioneering in considering dynamic response, their approach lacked integration within a formal OPF framework. In the early 2000s, optimization-based approaches regained attention. Granelli et al. [7] applied genetic algorithms for congestion management with topology reconfiguration under deterministic conditions (i.e., static load) conditions. However, the method lacked adaptability and contextual awareness, relying on fixed fitness evaluations without real-time feedback, limiting performance in dynamic scenarios. A key advancement came with the formalization of Optimal Transmission Switching (OTS). Fisher et al. [8] modeled OTS using mixed-integer programming within a DC-OPF framework, achieving notable cost reductions. Yet, the neglect of AC system characteristics constrained its practical applicability.

Recent studies have further advanced the topology optimization action into the OPF problem. Heidarifar et al. [9] proposed a heuristic method incorporating AC power flow and N-1 contingency constraints into line switching and bus splitting. While effective, the approach faces real-time limitations due to the computational burden of contingency analysis. Zhou et al. addressed the optimization of the grid topology at the substation level using bus splitting actions, with the aim of reducing congestion and improving grid efficiency [10]. However, the complexity of substation configurations and high computational demands pose challenges for real-time deployment. Zhang and Liu developed a multiperiod OPF model to manage voltage stability and security through transmission switching [11]. While it provides a comprehensive solution, the method's real-time application is limited by the need for extensive computations over multiple periods. Tavakkoli and Amjady extended the AC-OPF model by incorporating bus-bar switching actions to prevent overload conditions [12]. But its practical use in real-time is constrained by the intricacy of the overload relay dynamics and the computational time required. These developments highlight the need to balance optimization accuracy with computational tractability in large-scale systems.

In parallel to conventional mathematical optimization, different groups have explored RL as a promising technique to achieve the goal of real-time decision making in large-scale and uncertain dynamic environments [13]. For instance, unlike the classical OPF, which relies on mathematical programming and model-based optimization, RL offers a data-driven and adaptive approach capable, at inference time, to recommend remedial actions for a human operator. Building on these foundations, recent studies seek to enhance the practicality and robustness of RL in the power system operations context. For instance, Wang et al. integrated a physics-informed RL framework for AC-OPF considering future system conditions over a specified time horizon (e.g., several hours- or day-ahead) [14]. Wu et al. extended this concept by developing a safe RL method in which the learned policies respect operational limits during stochastic OPF [15]. Furthermore, RL has also been used to manage contingencies, (a) Awais proposed deep RL to handle generator failures [16], while Li et al. combined graph-based learning with a deep RL model (graph neural network) to capture the network topology in OPF problems involving renewable generation [17]. Despite their potential, these approaches often struggle with convergence, interpretability, and the demand for large-scale training data.

In 2019, RTE launched the L2RPN (Learning to Run a Power Network) challenge [18], focused on congestion management. The competition introduced Grid2Op [19], an RL environment designed to train and evaluate control algorithms using realistic grid dynamics and historical data. The challenge evolved in complexity with the introduction of robustness and adaptability tracks in the NeurIPS 2020 edition. The robustness track incorporated an adversarial component that simulated $N-1$ contingency scenarios by heuristically disabling transmission lines [20], thus testing agents' resilience to sudden network disruptions. Meanwhile, the adaptability track introduced a higher share of RES, ranging from 10% to 30%, to evaluate the flexibility and responsiveness of agents under variable generation conditions [21]. The winning agent in the NeurIPS 2020 L2RPN competition [22] introduced an action-set-based policy optimization technique that directly embeds safety constraints into the learning process – crucial for bridging experimental success and real-world applicability. Unlike other top submissions that used standard deep RL methods like Proximal Policy Optimization (PPO) and Double Deep Q-Network (DDQN), this agent employed a genetic algorithm for policy optimization. However, a subsequent analysis [2] showed that no submitted solution fully met the stringent safety and reliability standards required for deployment in operational power grids. A notable advancement beyond the L2RPN NeurIPS 2020 and WCCI 2020 agents is the PowRL agent [23], a PPO-based model designed to manage power networks under variable and adversarial conditions. While it showed improved resilience and adaptability, its effectiveness depends heavily on the simulation environment's fidelity and reward design. Similarly, CurriculumAgent (CAgent) [24] extended PPO by integrating an $N-1$ reliability strategy and systematically comparing rule-based and RL-driven topology control. Despite their promise, such RL models often lack interpretability, limiting trust in safety-critical grid operations.

These RL-based approaches represent significant progress toward intelligent data-driven grid management, demonstrating promising performance in tasks such as re-dispatch and topology optimization. However, they often suffer from two shortcomings. Firstly, the limited interpretability of the learned policies. These approaches rely heavily on deep neural networks to approximate value or policy functions, making it difficult to trace the rationale behind specific control actions, such as switching operations or dispatch decisions. For example, in [17], the agent learns to act based on graph-structured observations, but

it provides no explicit explanation of the grid conditions or topological features that triggered its decisions. Similarly, in [15], while the constraints are enforced via a safe-RL framework, the internal decision pathway remains opaque. Secondly, these methods often require large amounts of simulated data and prolonged training episodes to converge, which can be computationally expensive and hinder rapid adaptation to evolving grid conditions.

1.2. Contributions

Compared to the state-of-the-art, this paper introduces three novel contributions, which are discussed in detail below:

- Genetic Network Programming (GNP) framework that incorporates dynamic node behavior, enabling context-aware decision-making in uncertain power system environments. Unlike conventional approaches that rely on fixed function nodes, this method employs functional nodes that dynamically adapt their behavior based on real-time system states.
- A hybrid methodology that combines Genetic Network Programming with Decision Tree (GNP-DT), and due to its graph-based structures, enhances interpretability through providing human-understandable reasoning for each control action.
- This method outperformed both the baseline expert system [25] and CAgent [24] for the IEEE 118-bus system, achieving up to an 18% improvement in the mean L2RPN performance score and an 80% reduction in inference time.

In this work, a novel variant of GNP is proposed that diverges significantly from conventional formulations such as [26], particularly in its structural design and dynamic node functionality. Traditional GNP frameworks typically use fixed function nodes and update them in a sequential manner, resulting in rigid decision paths that struggle to adapt in high-dimensional, time-varying environments. In contrast, the proposed GNP architecture features adaptive nodes capable of modifying their behavior in response to evolving network states and accumulated experience. This flexibility allows the network to represent more sophisticated, state-dependent policies well-suited to the uncertain and non-linear nature of power systems, such as real-time fluctuations from RES or non-linearities inherent in AC power flow. Crucially, this design enables context-aware decision-making. For instance, an overloaded line may not always trigger the same switching action; instead, the node selects an appropriate control based on additional context, such as nearby contingencies, mimicking expert reasoning under diverse operational scenarios. As training progresses (i.e., learning from data and experience), these evolving heuristics capture expert-like judgment, resulting in a responsive and interpretable control mechanism that adapts in real-time to changing grid conditions. Furthermore, the possibility of seeding at initialization the GNP population with expert-derived decision graphs (e.g., from a pre-existing ES) can accelerate convergence in early learning phases by reducing the need for extensive trial-and-error.

Interpretability is a central contribution of the proposed method, achieved through the integration of graph-based structures. In particular: (a) unlike conventional deep RL approaches that operate as "black-boxes", the GNP framework structures decision-making within an explicit graph-based representation. Each node in the graph corresponds to a programmed decision heuristic, allowing the entire policy to be visualized and traced. This structure makes it possible to understand the rationale behind specific topology actions at any point during learning. In contrast, end-to-end deep RL methods, such as those in [22,24,27], often produce uninterpretable decisions encoded in high-dimensional neural parameters, limiting reasoning and reducing trust from grid operators; (b) a multistage decision tree (DT) was employed to extract human-readable rules from the trajectory of actions created by top-performing decision graphs in GNP, which were collected into a pool representing high-quality policy behavior. Separate DTs are then trained to capture key aspects of the control logic. This hierarchical rule extraction enables modular interpretation, where each stage informs and constrains the next.

1.3. Structure

The remainder of this paper is organized as follows. The framework of evolving system operation rules is described in Section 2, where the proposed AI agent is explained for real-time congestion management. The setup of the case study and the evaluation approach are detailed in Section 3. Section 4 provides a comparative discussion of the experimental results. The interpretability of the proposed framework is discussed in Section 5. Finally, the conclusions and future work are included in Section 6.

2. Evolving system operation rules

The proposed methodology targets real-time line congestion management in power systems. Its core concept involves evolving heuristic control policies, originally derived from operator expertise or pre-existing expert systems, encoded as a decision graph (see Section 2.1).

The congestion management task can be modeled as a Markov decision process [28], defined as a tuple $\mathcal{M} = (S, \mathcal{A}, P, R, \gamma)$, where:

- S is the state space. Each state $s_t \in S$ represents the full state of the grid at time t, including power line flows, bus voltages, switch configurations, and operational constraints of the AC power flow.
- $\mathcal{A}$ is the action space. Each action $a_t \in \mathcal{A}$ corresponds to a control action, such as line switching (disconnection/reconnection), bus reconfiguration, bus recovery, or active power flexibility (e.g., redispatch, RES curtailment, storage system management).
- $P(s_{t+1} \mid s_t, a_t)$ is the state transition function, encoding the system dynamics and physical laws that govern how the grid evolves in response to actions.
- $R(s_t, a_t)$ is the reward function, used to evaluate the safety and effectiveness of each action.
- $\gamma \in [0, 1]$ is the discount factor.

At each timestep t, the agent observes the system state s_t and selects an action a_t according to a pre-defined policy $\pi(a_t \mid s_t)$. The environment then transits to a new state s_{t+1} based on the dynamics:

$$s_{t+1} \sim P(s_{t+1} \mid s_t, a_t), \quad a_t \sim \pi(a_t \mid s_t) \tag{1}$$

In the original formulation, the RL framework relied on a sparse binary reward signal to encourage system survival. Let $r(s_t, a_t)$ denote the reward received at timestep t, where s_t is the system state and a_t the action taken. the agent receives a reward of 1 for each successful step t taken (i.e., system is able to fully meet the electricity demand with the available generation resources), and a reward of 0 upon reaching a terminal or gameover state (i.e., when transmission lines exceed their physical capacity and lead to cascading failure of other lines that ends with system collapse); this is similar to Grid2Op [19] Episode duration reward:

$$r(s_t, a_t) = \begin{cases} 1, & \text{if the agent successfully passes a step} \\ 0, & \text{if the agent reaches a gameover state} \end{cases} \tag{2}$$

The RL objective is to find the optimal policy π^* that maximizes the expected cumulative reward over an episode of length T:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{T} r(s_t, a_t) \right], \tag{3}$$

where $\tau = (s_0, a_0, s_1, \ldots, s_T)$ denotes a trajectory under policy π, and the expectation $\mathbb{E}_{\tau \sim \pi}$ is over trajectories sampled under the environment dynamics. As also noted in [2,27], the winning deep reinforcement learning (deep-RL) agent of the L2RPN WCCI 2020 Challenge adopted

the episode-duration reward formulation, reinforcing its validity as a baseline reward signal for grid control learning tasks. While this formulation encourages robust strategies for prolonging operation, it does not directly account for operational costs, which were only evaluated post hoc. To address this limitation, we propose a *cost-aware reward* that incorporates a formal operational cost model inspired by the reward proposed by Grid2Op contributors and closely aligned with the L2RPN scoring metric [29]. The per-timestep operational cost is:

$$C_{\text{op}}(t) = C_{\text{dispatch}}(t) + C_{\text{flex}}(t) + C_{\text{losses}}(t), \tag{4}$$

with

$$C_{\text{dispatch}}(t) := \sum_{g \in \mathcal{G}} c_g(t)\, p_g(t),$$
$$C_{\text{flex}}(t) := C_{\text{storage}}(t) + C_{\text{curtail}}(t) + C_{\text{other\_flex}}(t),$$
$$C_{\text{losses}}(t) := \text{cost-per-MWh}_{\text{loss}} \times \text{losses}(t),$$

and $\mathcal{G}$ is the set of generators. The blackout (unserved load) cost is:

$$C_{\text{blackout}}(t) = \text{USL}(t) \times \pi_m(t) \times \beta, \tag{5}$$

with $\text{USL}(t)$ the not-supplied load, $\pi_m(t)$ the marginal price, and $\beta > 1$ a penalty factor. The total episode cost is:

$$C_{\text{episode}} = \sum_{t=1}^{T} \big(C_{\text{op}}(t) + C_{\text{blackout}}(t)\big). \tag{6}$$

We formally define an upper bound for the episode as the worst-case cost:

$$C_{\max} := \max_{\text{worst scenario}} \sum_{t=1}^{T} C_{\text{blackout}}(t), \tag{7}$$

where the worst scenario assumes that the agent fails to supply all demand at the highest marginal price in each timestep. The normalized cost-aware reward is then:

$$R_{\text{cost}} = \frac{C_{\max} - C_{\text{episode}}}{C_{\max}} \times 100, \tag{8}$$

so that lower costs produce higher rewards on a 0–100 scale. Survival is naturally embedded, as blackouts contribute heavily to C_{episode}.

The survival-based reward emphasizes robustness alone, whereas the normalized cost-aware reward jointly captures system reliability and operational efficiency. The blackout penalties ensure that maintaining supply security remains a primary objective, aligning the reward with real-world operational goals.

In this work, an undiscounted reward formulation ($\gamma = 1$) was adopted to evaluate the cumulative performance of the control policy over a finite time horizon. This choice is motivated by the need to treat all future rewards with equal importance, as the objective is to optimize the agent's performance without biasing toward immediate gains. Since the task involves an episodic evaluation with a clear termination criterion, discounting is unnecessary and can introduce an unintended emphasis on short-term outcomes as explained in [30]. Moreover, the undiscounted reward simplifies the analysis and interpretation of the learned policies, ensuring that the optimization process directly targets the maximization of overall long-term effectiveness.

The learning process, including the evolution of expert knowledge, is detailed in Section 2.2. Moreover, a rule extraction method that predicts the best action with respect to evolved knowledge, considering contextual information, is described in Section 2.3.

2.1. Expert system knowledge representation

The baseline knowledge for this work and also for the GNP rule evolution in the next section is the ES developed by RTE [25], where the aim was to emulate the decision-making process of human operators. This ES was enhanced by considering improvements in the calculation method for determining the influence graph, using more flexibility options (e.g., line switching and redispatch, in addition to the bus splitting option), new ranking criteria, the possibility of decision revision, and the possibility of proposing multiple actions rather than a single action.

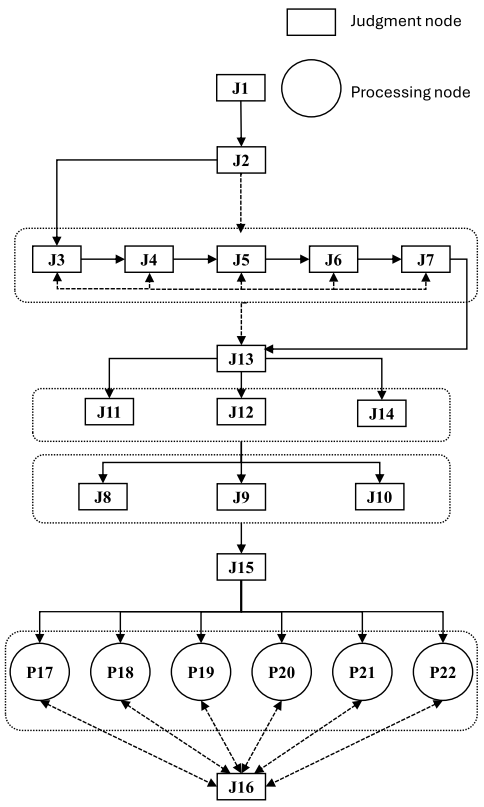


Fig. 1. ES decision-making graph.

The ES is formulated as a network graph for knowledge representation, as illustrated in Fig. 1, where ES does not include dashed connections; these connections represent other reformation of ES used for initializing the GNP algorithm in Section 2.2. Each judgment node (J) within this framework is implemented as an individual computer program, responsible for a specific decision-making task, as detailed in Table 1.

The sequence of these judgment programs guarantees convergence to one of the processing nodes: $P17$, $P18$, and $P19$ execute bus-splitting for hub, loop, and downstream buses, respectively; $P20$ handles line switching; $P21$ activates power flexibility; and $P22$ implements a recovery strategy to the reference topology. This ensures that the system reaches a unique, executable action path tailored to the grid's condition. For instance, in a specific case, two congestion events were detected on lines 20 and 12.

- For **line 20**: The flexibility nodes identified buses 76 and 81 as hub buses.
- For **line 12**: The responsible flexibility group identified buses 67 and 68 as hub buses. In addition, the bus set {67, 80, 79, 78} was recognized to form a looped path suitable for re-routing.

Since the congested lines were located in different zones, the judgment node J2, which handles inter-zone coordination, was not activated. In a specific topological configuration, bus 68 alone was sufficient to solve both congestion events. Consequently, the rest of the judgment nodes that contain additional conditional constraints were not activated.

Table 1
Functions of judgment nodes (J) in the ES decision graph for congestion management.

Node	Function	Details
J1	Identify and rank congested lines	Based on criticality
J2	Skip further analysis	Triggered if all issues are within a single critical zone
J3	Explore flexibility – hub bus splitting	Key locations enabling power flow rerouting through parallel uncongested paths
J4	Explore flexibility – loop bus splitting	Buses that build a local mesh
J5	Explore flexibility – downstream bus splitting	Buses that could be supplied from a different path
J6	Explore flexibility – line switching	Based on topological search [31]
J7	Explore flexibility – active power flexibility (redispatch, RES curtailment, storage management)	Based on sensitivity indices [31]
J8	Evaluate stop condition	Solution quality considering the first threshold
J9	Evaluate stop condition	Solution quality considering the second threshold
J10	Evaluate stop condition	Number of analyzed issues exceeded a threshold
J11	Recovery	Triggered when no valid action proposed
J12	Recovery	Triggered when solution quality falls below a threshold
J13	Multi-criteria hierarchical ranking	Uses tuple $\sigma(a) = (\sigma_{cp}(a), \sigma_t(a), \sigma_{op}(a))$: $\sigma_{cp}(a)$: Control priority (e.g., switching vs. redispatch) $\sigma_t(a)$: Topological effectiveness (e.g, issue disappeared/alleviated/worsened/created) $\sigma_{op}(a)$: Operational score (e.g., sum of squared marginal power flows)
J14	Ranking revision under thresholds	Revising the criteria order in J13 per issue
J15	Ranking revision under thresholds	Revising the criteria order in J13 after full analysis
J16	Add more actions	Uses ranked list if one action is insufficient

2.2. GNP rule evolution

To overcome the limitations of rigid decision-making in traditional GNP, the proposed method integrates an RL-enhanced GNP framework that supports dynamic and context-aware behavior. In this approach, each node is capable of altering its output depending on current grid conditions, such as line overloads, generation/load levels, or switching states. For example, a judgment node that initially selects the reconfiguration action at 'Bus A in zone 1' under moderate congestion may instead recommend reconfiguration at 'Bus C in zone 3' if RES fluctuations create a localized congestion between the two zones. The node's behavior is therefore not statically defined, but instead learns a mapping from state features to decision criteria (e.g., change in priority of flexible units or threshold of activating a flexible unit), enabling adaptive control that aligns with requirements for real-time management. Furthermore, the execution sequence of the decision graph is not fixed, in contrast to traditional GNP, which follows a hardcoded node traversal. Instead, the proposed method allows the policy graph to activate different substructures conditionally, based on current state inputs. These conditions include the level of congestion, fault locations, substation configurations, and the recent history of control actions. The adaptive traversal mechanism in the proposed GNP supports conditional connections between nodes, especially under time-varying network constraints.

This structural advancement is depicted in Fig. 2, which compares the traditional and proposed GNP formulations. On the left, the traditional GNP follows a linear structure in which each node is associated with a fixed function, and execution proceeds through a predefined sequential path, formalized in Eq. (9).

$$n_{t+1} = \text{Next}(n_t), \quad a_t = f_{n_t}(s_t) \tag{9}$$

where n_t denotes the node visited at time t, $\text{Next}(n_t)$ is the deterministic next node pointer, s_t is the observed grid state at time t, f_{n_t} is the decision rule implemented at node n_t, and a_t is the resulting action taken by the agent. Each node processes its input and passes control unconditionally to the next node, regardless of the evolving state of the grid. The system response is therefore insensitive to diverse operational contexts, limiting its effectiveness in dynamic environments.

In contrast, the proposed GNP framework (right) organizes the policy graph into an interconnected network of nodes, each with adaptive behavior. The nodes evaluate the state vector, which can include the power flow on the lines, AC power flow constraints, grid topology, and dynamically determine their output. The graph includes multiple

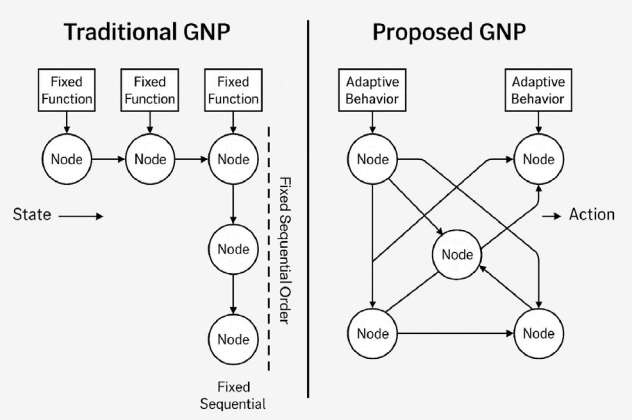


Fig. 2. Regular GNP vs. adaptive GNP. The regular GNP employs sequential node activation, whereas the adaptive GNP dynamically adjusts node behavior and execution paths based on environmental states.

possible transitions from each node, with links encoding condition-based execution paths learned through RL. For example, depending on whether the overload is localized or widespread, a node might route control to a sub-policy targeting either demand-side response or topological reconfiguration. The arrows between nodes represent these conditional transitions, learned from high-performance decision graphs during training. Node connectivity reflects logical dependencies and execution flexibility, rather than fixed ordering, as expressed in Eq. (10).

$$n_{t+1} \sim \pi_{\text{trans}}(n_{t+1} \mid n_t, s_t), \quad a_t = f_{n_t}(s_t) \tag{10}$$

where π_{trans} is a stochastic policy over node transitions, and all other symbols are as defined previously. The system thus constructs control policies by composing rule segments that are most relevant under the current grid state, forming context-specific decision pathways that align with expert behavior while remaining responsive to uncertainty. This structural difference is summarized in Table 2. Thus, in contrast to classic GNP, where node transition probabilities remain static, the adaptive GNP dynamically adjusts these transitions according to RL feedback, enabling the decision pathways to evolve in response to the agent's performance. In general, the proposed structure enables the GNP policy to behave as a state-driven control mechanism, capable of decomposing complex decisions into subgraphs, dynamically

Table 2
Comparison of traditional GNP vs. adaptive GNP execution.

Component	Traditional GNP	Adaptive GNP
Node execution	$a_t = f_{n_t}(s_t)$	$a_t = f_{n_t}(s_t)$
Node transition	$n_{t+1} = \text{Next}(n_t)$	$n_{t+1} \sim \pi_{\text{trans}}(n_{t+1} \mid n_t, s_t)$
Policy structure	Fixed	Conditional (learned)
Graph behavior	Deterministic	State-driven

coordinating local actions, and maintaining interpretability for human operators, as will be shown in Section 5. The interaction between the GNP evolutionary process and the RL-based adaptive learning is further illustrated in Fig. 3, which presents a flowchart summarizing the information flow, feedback loop, and optimization stages within the proposed GNP–RL collaborative framework. Let G_i denote the ith individual graph in the population, representing a policy μ_i that governs decision-making in a Markov decision process with state space $\mathcal{S}$, action space $\mathcal{A}$, and reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Each graph consists of *judgment nodes* and *processing nodes*; *judgment nodes* perform condition-based branching, while *processing nodes* issue actions $a_t \in \mathcal{A}$. During interaction with the environment, a graph follows its encoded logic to generate trajectories $\pi_i = (s_0, a_0, r_0, s_1, \dots)$. During execution, each node n maintains heuristic parameters θ_n, which are updated based on local feedback from the environment, typically as a function of the observed state, i.e., $\theta_n \leftarrow f_n(s_t)$. This adaptive mechanism refines node behavior across generations, supporting the learning dynamics. The process continues until a failure condition or episode termination is met. The cumulative reward along π_i defines the fitness F of G_i, Eq. (11), which reflects the operational lifespan of the graph under dynamic grid conditions. This formulation inherently prioritizes policies that maintain safe grid operation.

$$F(G_i) = \sum_{t=0}^{T-1} r_{t+1} \tag{11}$$

The evolution of the graph population $\mathcal{P}$ is governed by Algorithm 1, which integrates crossover and mutation operators as defined in Algorithm 2. A two-stage crossover mechanism is employed to balance exploitation and exploration:

1. **Exploitation-driven crossover**: selects both parents P_1, P_2 from the top-performing 50% of the population to preserve high-fitness substructures.
2. **Exploration-driven crossover**: pairs elite individuals (top e_i%) with non-elite ones to introduce novel combinations and maintain diversity.

Random selection from a finite set is denoted by $x \sim \mathcal{U}(S)$, where $\mathcal{U}$ represents the uniform distribution over set S. Concatenation of gene segments is expressed using the symbol $\|$, while sub-vectors are denoted by slicing notation, e.g., $P[1{:}x]$ refers to the first x elements of parent P, and $P[x{+}1{:}L]$ refers to the remaining portion. Individual candidates P_1, P_2 are removed from their selection pools via set subtraction, indicated by $S \setminus \{P\}$. The crossover logic uses inline conditional checks such as $\texttt{random}() < cp_1$ to decide probabilistic events directly, avoiding auxiliary variables. Mutation is applied by modifying a random subsequence of each individual with probability m_p, constrained by a maximum range $\lambda_{\max} \cdot L$, where L is the gene length. The modification could be defined by ignoring/adding conditional constraints, ignoring/adding a node, decreasing/increasing a threshold, or changing a criterion. This mutation operates directly on the graph structure by modifying node logic or interconnections.

After each generation, the elite graph G^* is updated by selecting the graph with the highest fitness – Eq. (12).

$$G^* = \arg\max_{G_i \in \mathcal{P}} F(G_i) \tag{12}$$

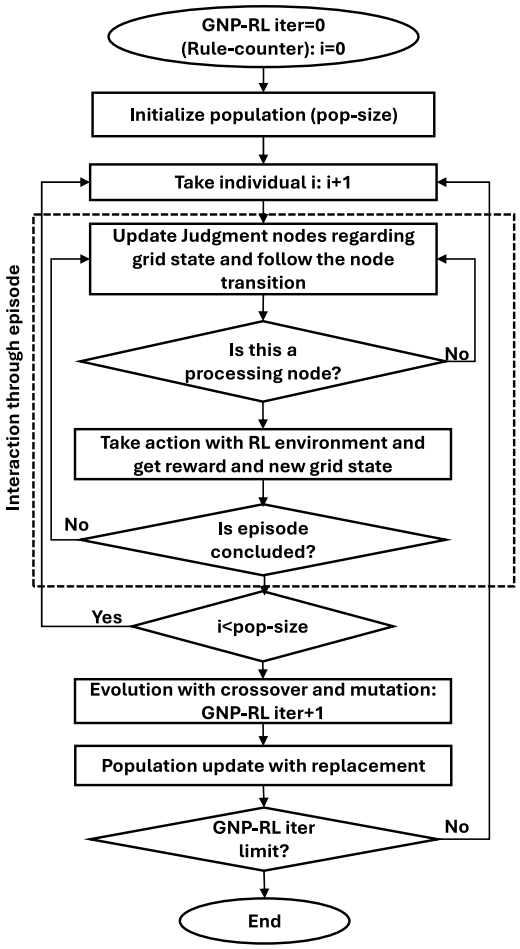


Fig. 3. Flowchart illustrating the fusion between GNP and RL, showing how the RL-based feedback updates guide the evolutionary process toward adaptive and optimal decision graph structures.

The GNP algorithm executes multiple decision-making agents (GNP graphs) in parallel, each interacting with the environment and collecting diverse experiences. While the entire population contributes to exploration through varied behaviors, learning and policy updates are guided by the elite G^* graph, identified as the best-performing individual based on the cumulative reward. This design enables efficient reuse of experience across generations and ensures that optimization is focused on high-quality policies. By decoupling exploration from exploitation, where population diversity drives exploration and the elite graph directs learning, the framework achieves both robustness and adaptability in evolving effective control strategies.

2.3. Decision tree rule extraction

It has been recognized that a single decision graph may not generalize optimally across diverse grid conditions due to varying fault locations, network topologies, and intertemporal dependencies. Therefore, the elite policies of the GNP algorithm are used to build a rule pool, consisting of the best-performing decision graph for each episode in the last generation. In particular, the elite graph represents an action selection trajectory that has the highest possible performance in an episodic evaluation; this is quite important, since it has optimized the impact of an action at the present grid condition (present timestep) and its post-impact in the next time steps. Therefore, elite-derived actions per timestep will be used to generate labeled datasets (i.e., consisting of grid state features and the corresponding actions), in which these

Algorithm 1: Adaptive GNP evolutionary algorithm

Data: Initial population of size N GNP graphs $\{G_1, G_2, \ldots, G_N\}$;
Crossover probabilities $cp_1, cp_2 \in (0,1)$;
Mutation probability $m_p \in (0,1)$;
Maximum mutation range $\lambda_{\max} \in \mathbb{R}_+$;
Total reward R;
Result: Elite graph G^* with highest fitness

Initialize population: $\mathcal{P} = \{G_1, G_2, \ldots, G_N\}$;
$best\_fitness \leftarrow -\infty$, $G^* \leftarrow$ None;
while *termination condition not met* **do**
  foreach *graph* $G_i \in \mathcal{P}$ **do**
    $R \leftarrow 0$;
    Initialize RL environment $\mathcal{E}$ at state s_0;
    while *episode not done* **do**
      $n_t \leftarrow n_{\text{start}}$ of G_i;
      while n_t *is a judgment node* **do**
        $n_{t+1} \leftarrow \pi_i(n_{t+1} \mid n_t, s_t)$;
      end
      if n_t *is a processing node* **then**
        $a_t \leftarrow f_{n_t}(s_t)$;
        Simulate a_t in environment: $\mathcal{E}(s_t, a_t) \rightarrow s_{t+1}$;
        $r_{t+1} \leftarrow$ reward from $\mathcal{E}$;
        $R \leftarrow R + r_{t+1}$;
        $s_t \leftarrow s_{t+1}$, $t \leftarrow t+1$;
      end
      Update heuristic parameters of node: $\theta_n \leftarrow f_n(s_{t+1})$;
    end
    $F(G_i) \leftarrow \sum_{t=0}^{T-1} r_{t+1} = R$;
    if $F(G_i) > best\_fitness$ **then**
      $best\_fitness \leftarrow F(G_i)$;
      $G^* \leftarrow$ copy of G_i
    end
  end
  $\mathcal{P} \leftarrow$ **GNP_Operators**$(\mathcal{P}, cp_1, cp_2, m_p, \lambda_{\max})$;
end
return G^*

Algorithm 2: GNP operators: crossover and mutation

Data: Population size N; crossover rates $cp_1, cp_2 \in (0,1)$; mutation rate m_p; max mutation span $\lambda_{\max}$;
Subgroups: top_half, elite, non_elite;
Result: New population P_{new}

$P_{\text{new}} \leftarrow \emptyset$;
for $i \leftarrow 1$ **to** $|top\_half|$ **do**
  if $|top\_half| = 0$ **then**
    break;
  end
  Select $P_1, P_2 \sim \mathcal{U}$(top_half), without replacement;
  top_half $\leftarrow$ top_half $\setminus \{P_1, P_2\}$;
  $x \sim \mathcal{U}(\{1, \ldots, L-1\})$;
  if *random()* $< cp_1$ **then**
    $c \leftarrow P_1[1:x] \parallel P_2[x+1:L]$;
  else if *random()* $< cp_2$ **then**
    $c \leftarrow P_2[1:x] \parallel P_1[x+1:L]$;
  else
    $c \leftarrow$ *random choice*(P_1, P_2);
  end
  $P_{\text{new}} \leftarrow P_{\text{new}} \cup \{c\}$;
end
for $i \leftarrow 1$ **to** $|non\_elite|$ **do**
  if $|non\_elite| = 0$ **then**
    break;
  end
  Select $P_1 \sim \mathcal{U}$(elite);
  Select $P_2 \sim \mathcal{U}$(non_elite), without replacement;
  non_elite $\leftarrow$ non_elite $\setminus \{P_2\}$;
  $x \sim \mathcal{U}(\{1, \ldots, L-1\})$;
  if *random()* $< cp_1$ **then**
    $c \leftarrow P_1[1:x] \parallel P_2[x+1:L]$;
  else if *random()* $< cp_2$ **then**
    $c \leftarrow P_2[1:x] \parallel P_1[x+1:L]$;
  else
    $c \leftarrow$ *random choice*(P_1, P_2);
  end
  $P_{\text{new}} \leftarrow P_{\text{new}} \cup \{c\}$;
end
foreach $c \in P_{new}$ **do**
  if *random()* $< m_p$ **then**
    Modify a subsequence of c (length $\leq \lambda_{\max} \cdot L$);
  end
end
return P_{new}

datasets are then used to train a multistage DT to capture different components of control logic. During training, the agent was initially allowed to explore a broad range of control actions, including redispatch, curtailment, and storage activation. However, as evolution progressed, the population of decision graphs consistently converged toward topological reconfiguration — particularly bus-splitting and line switching actions — as the dominant and most effective strategy. This behavior aligns with findings from the Learning to Run a Power Network (L2RPN) competitions framework, which emphasizes that topological operations are the most direct and cost-efficient means of alleviating congestion and maintaining grid stability, indicating that power flexibility actions were less effective, and it may also involve a CO_2 emissions increase (e.g., from RES curtailment) and higher operational cost. For instance, the Binbinchen team (Huawei) [32], which achieved second place in the L2RPN WCCI 2020 competition, developed a deep reinforcement learning Actor–Critic agent with PPO that was trained exclusively on topological actions, restricted to 200 safe configurations following an expert-guided reduction phase. Similarly, the RTE expert system [25] focused solely on bus-splitting operations. In particular, the final analysis of the L2RPN NeurIPS 2020 challenge in the robustness track [2] challenge demonstrated that several of the toughest episodes (Jan 28.1, Nov 34.1, Apr 42.2, Oct 21.1) remained feasible if the agent relied solely on topological actions. These findings justify the convergence of the agent's decision-space making onto topology control, aligning with both real-world operational practice and proven benchmark evidence. As a result, GNP-DT focused solely on topology reconfiguration, though it can accommodate redispatch actions without changing its formulation. The sequence of DT models is defined as follows:

(i) The first DT model (DT1) to classify the flexibility type (e.g., line reconnection vs. bus reconfiguration);
(ii) The second DT model (DT2) to estimate the required number of flexibility actions;
(iii) The third DT model (DT3) to estimate the specific line(s)/bus(es) to operate.
(iv) The fourth DT model (DT4) will be used only if the detected flexibility type involves bus reconfiguration (bus splitting) regarding DT1. The set of top-k most probable feasible topology reconfiguration actions for the bus identified by DT3 is extracted from DT4, based on the predicted class probabilities at the leaf nodes.

Table 3
Summary of the DTs and their corresponding state variables.

Variable	DT1	DT2	DT3	DT3	DT3	DT3	DT4
Action type	Any	Any	Bus reconfig.	Line disconn.	Bus recovery	Line reconnection	Bus reconfig.
Cong. line IDs	✓	✓	✓	✓	✗	✗	✓
Max overload	✗	✓	✓	✗	✗	✗	✗
Split bus IDs	✓	✓	✓	✓	✓	✗	✓
Disconn. line IDs	✓	✓	✓	✓	✓	✓	✓
Cong. line count	✓	✓	✓	✓	✗	✗	✓
Split bus count	✓	✓	✓	✓	✓	✗	✓
Disconn. line count	✓	✓	✓	✓	✓	✓	✓

Table 3 presents a summary of the relevant state variables considered for different decision trees (DT1-DT4) and actions in the system. Each row indicates whether a specific variable (for example, congested line IDs, maximum overload, split bus count) is used (✓) or not (✗) by the corresponding decision logic. The variables include:

- **Cong. Line IDs**: identifiers of congested lines;
- **Max Overload**: maximum overload value among congested lines;
- **Split Bus IDs**: identifiers of splitted buses;
- **Disconn. Line IDs**: identifiers of disconnected transmission lines;
- **Cong. Line Count**: total number of congested lines;
- **Split Bus Count**: number of splitted buses;
- **Disconn. Line Count**: number of disconnected lines.

DT3 and DT4 are dependent on the action type; in this regard, Bus reconfig. refers to bus splitting. Line disconn. means disconnecting a transmission line to isolate a fault or relieve congestion. Bus recovery restores a previously reconfigured bus (split bus) to the original topology (unaltered topology), and line reconnection involves re-energizing a previously disconnected line to resume normal operation. In this work, line identifiers (e.g., *Cong. Line ID_**) are not arbitrary indices but are mapped to physically meaningful locations in the power grid. Each identifier corresponds to a specific transmission line whose electrical location is associated with a predefined zone (area) or group of feeders based on the grid layout. This structured mapping allows the model to implicitly reason about localized congestion using indexed features aligned with the physical topology of the system.

This module is a hierarchical approach in which each stage constrains and informs the subsequent one. For instance, the flexibility type determined in the first stage conditions the search space for the next model, which predicts how many units must be activated, and in turn, this output influences the final model that selects which specific assets to operate. Formally, each stage is modeled as a conditional expectation:

$$\hat{y}_r = \mathbb{E}\left[y_r \mid X, \hat{y}_{r-1}, \hat{y}_{r-2}, \ldots, \hat{y}_{r-k}, \theta\right] + \epsilon_r \tag{13}$$

where $\hat{y}_r$ denotes the predicted decision component at rule stage r, conditioned on the input features X, preceding predicted components $\hat{y}_{r-1}, \hat{y}_{r-2}, \ldots, \hat{y}_{r-k}$, and model parameters θ. The parameter set θ represents the learnable weights of the conditional model, including decision thresholds, node weights, and any coefficients used to combine input features and prior stage predictions. These parameters are estimated during training by minimizing a suitable loss function (e.g., mean squared error or cross-entropy) over the training episodes, using the observed target outputs y_r. The residual term ϵ_r captures unmodeled variability at stage r. The residual term ϵ_r accounts for the prediction error. This layered rule extraction allows the DT ensemble to capture interdependencies among rule components, while enabling both interpretability and domain-relevant fidelity.

3. Test case

3.1. Description

The IEEE 118-bus system environment, featured in the L2RPN competition at the 2022 World Congress on Computational Intelligence (WCCI) [33], serves as the test case and is illustrated in Fig. 4. This case incorporates an adversarial module that heuristically simulates line outages to emulate the N-1 security criterion, while allocating 10% to 30% of the generation mix to RES. It comprises 118 substations, 186 lines, 91 loads, and 62 generators.

A set of 48 episodes was generated, using the methodology from [34], extracted from 12 months. Each scenario in the dataset represents a full operational episode. Scenarios were selected to cover all months of the year and a variety of operating conditions, including normal operation, high load, renewable variability, congestion-prone situations, and contingencies. This approach ensures that both the training and test sets are representative of the full range of realistic grid conditions, allowing for robust evaluation of the proposed GNP-DT agent. The episodes comprise 2018 time steps with a 5-minute resolution to feed the GNP method, in which the elite graphs generated 49,186 grid operating conditions and corresponding action set pairs for training DT models. All experiments use episode-level partitioning: each scenario corresponds to a full operational episode and is assigned entirely to either the training or test set to prevent temporal leakage. To ensure both coverage and separation, four episodes from each month were used for training and three episodes for testing, spanning the full year and capturing seasonal variations and diverse grid operating conditions. For robustness, the evaluation was conducted using 20 different random seeds for partitioning and training, ensuring that results are statistically representative and not dependent on a specific initialization. The proposed GNP algorithm interacted with the simulated power grid using the Grid2Op AI-friendly digital environment [19] as the RL environment. To ensure a robust evaluation of the proposed methodology, a distinct test set containing data from various months was employed during the testing phase.

3.2. Evaluation metric and benchmarks

In terms of performance indicator, the score adopted in the L2RPN competition [2] was used to allow a fair comparison with baseline agents that evaluates both the operational cost and the survivability of the agent. The score ranges from $[-100.0, 0.0, 80.0, 100.0]$, each value corresponding to a distinct level of performance by the agent during an episode. A score of -100.0 indicates that no steps were played, reflecting the maximum blackout penalty applied across all steps in all scenarios. A score of 0.0 represents the performance of a "Do-Nothing" baseline agent that takes no action and does not influence the scenario; if this agent completes the episode, the effective score range becomes $[-100.0, 0.0, 100.0]$. A score of 80.0 indicates that the agent successfully plays through all scenarios without correcting line disconnections caused by overloads; losses in this case are equal to the difference between total generation and total consumption in each scenario, and lines under maintenance are reconnected after their maintenance period. The maximum score of 100.0 is awarded when the agent both completes all scenarios and optimizes system performance by reducing losses to 80% of those previously computed. Thus, higher scores reflect more efficient decision-making and longer survival across scenarios.

The following agents from the Grid2Op baselines, which employ topological reconfiguration strategies for congestion management, were selected for benchmarking:

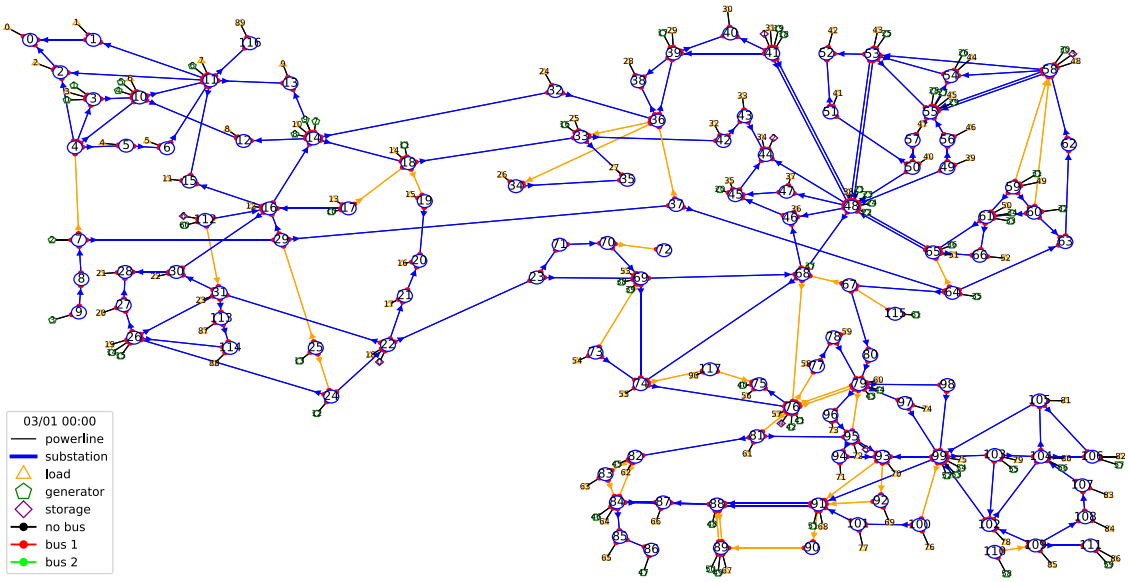


Fig. 4. IEEE 118-bus system of the L2RPN-WCCI-2022 competition. All the elements at each substation are connected to bus-1 in the reference topology, which could be switched to bus-2 through bus splitting actions.

- **ExpertOp4Grid**: An ES which tries to solve congestion issues within the grid [25] using an influence graph;
- **Enhanced ExpertOp4Grid**: An ES which is an enhanced version of *ExpertOp4Grid* agent explained in Section 2.1;
- **MILP agent**: A Mixed Integer Linear Programming (MILP) agent that seeks to minimize the overthermal lines based on DC approximation [35] with CBC-solver [36];
- **CAgent**: Curriculum Agent (Senior agent) that consists of a deep RL model based on PPO [24] with a senior threshold of $\rho_{senior} = 0.95$. Note that this agent is an enhanced version of a model originally developed for a past L2RPN competition [32], which inspired subsequent agents in later editions and also the winning solution of the 2023 Paris Region AI Challenge for Energy Transition [37].

The hyperparameters of the proposed GNP algorithm, including the crossover probabilities, mutation rate, and mutation range described in Section 2.2, were tuned to ensure robust convergence and satisfactory performance of the algorithm. The Bayesian optimization method was applied across 16 episodes covering the whole season of the year to ensure that the selected hyperparameters generalize beyond a single run. Table 4 presents the hyperparameters of the GNP algorithm. For clarity, the following nomenclature is adopted throughout the experiments: **GNP-DT-1** refers to the agent trained with the survival-duration reward function and experimental hyperparameters; **GNP-DT-2** corresponds to the agent trained with the proposed cost-aware reward function and optimized hyperparameters; and **GNP-DT-3** denotes the agent trained with the survival-duration reward function but using the optimized hyperparameters.

All experiments were conducted on a system running on a cloud-based virtual machine with an AMD EPYC CPU with 8 cores at 2.94 GHz, 32 GB of RAM, and Microsoft Windows 10 Pro.

4. Experimental results

This section presents the experimental results, evaluating the performance of the proposed agent across multiple random seeds and scenarios. The analysis includes a comparison with three state-of-the-art baseline models (available in the Grid2Op environment) and focuses on key metrics to assess the agent's effectiveness and consistency.

Table 4
Hyperparameters of the adaptive GNP algorithm.

Hyperparameter	Experimental	Optimized
Population size	30	38
Elite fraction	0.1	0.24
Crossover probability cp_1	0.5	0.41
Crossover probability cp_2	1	0.70
Mutation probability m_p	0.3	0.33
Maximum mutation range $\lambda_{\max}$	0.7	0.77

Median survival time comparison. Fig. 5 presents the median survival time across 36 test episodes for all evaluated agents. Among the proposed variants, *GNP-DT-2* and *GNP-DT-3* demonstrate consistently superior performance, reaching the maximum survival limit of 2018 steps in multiple episodes (e.g., *Jun_20*, *Sep_5_2*). Notably, the cost-aware reward used in *GNP-DT-2* results in comparable or even improved average survivability relative to the survival-based agents (*GNP-DT-1* and *GNP-DT-3*), suggesting that cost optimization implicitly encourages longer stable operation. In contrast, the optimization-based *MILP* and the deep RL-based *CAgent* exhibit greater variability across episodes, with early terminations in challenging conditions such as *Feb_28* and *Dec_12*. The expert-based *Enhanced ExpertOp4Grid* maintains steady performance and consistently surpasses the baseline *ExpertOp4Grid*, confirming the benefit of its refined heuristics. The passive *Do-Nothing* agent remains the weakest performer throughout all episodes. Overall, the GNP-DT family demonstrates the most resilient and adaptive behavior under diverse grid conditions, with *GNP-DT-2* offering the best trade-off between cost efficiency and operational stability.

Operational cost comparison. In terms of average operational cost, the proposed GNP-DT variants demonstrate a clear improvement in economic efficiency. Specifically, *GNP-DT-2*, which employs the cost-aware reward, achieved the lowest average cost of 35.12 M€, followed by *GNP-DT-3* at 35.6 M€ and *GNP-DT-1* at 38.22 M€. This indicates that integrating cost considerations into the reward function enhances both economic performance and overall stability. Among the comparative agents, the *Enhanced ExpertOp4Grid* and *CAgent* achieved moderate cost reductions with 40.15 M€ and 40.40 M€, respectively, outperforming the baseline *ExpertOp4Grid* (43.02 M€). Conversely, the optimization-based *MILP* agent exhibited the highest operational cost at 58.80 M€,

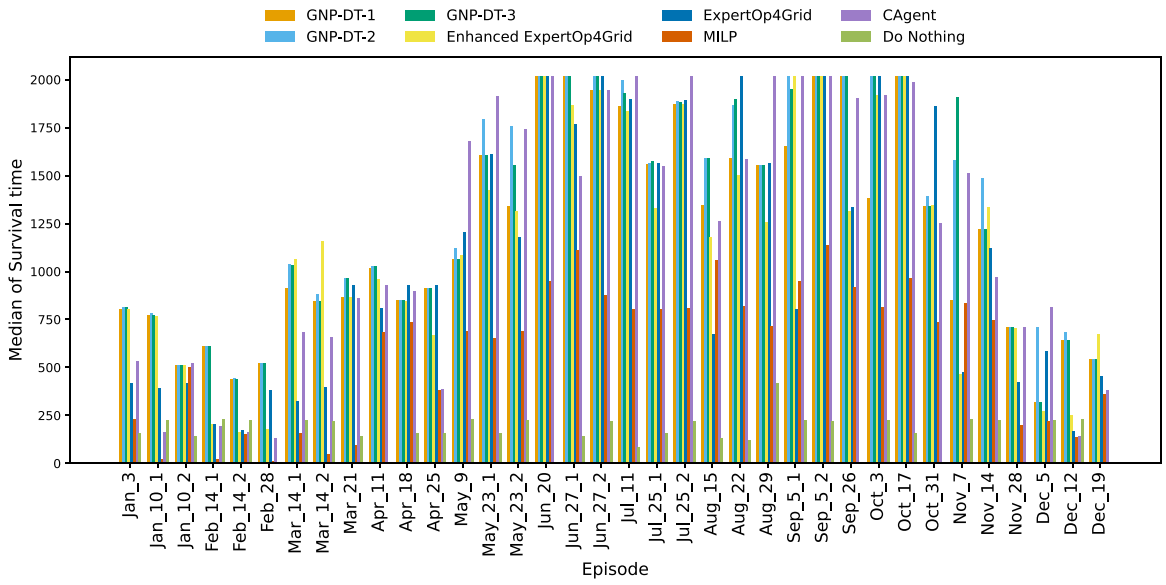


Fig. 5. Visualization of the median survival time per episode. Each bar corresponds to the median survival time in the respective episode across all seeds.

Table 5
Performance comparison between different agents on L2RPN score and survival time.

Agent	L2RPN Score							Survival time	
	Mean	Std	Median	1st quantile	3rd quantile	Min	Max	Median	MSTCM
Do nothing	0	0	0	0	0	0	0	158	158
ExpertOp4Grid [25]	37.5	**5.85**	36.96	33.48	42.08	24.93	50.52	804	930.75
Enhanced ExpertOp4Grid	39.04	7.95	36.63	34.09	44.93	26.3	56.97	981	1170
GNP-DT-1	44.16	7.37	41.94	38.82	49.72	33.64	59	1034	1143.25
GNP-DT-2	**48.1**	7.07	**46.01**	**43.73**	52.73	**36.04**	**62.33**	**1147**	**1439.5**
GNP-DT-3	46.99	7.43	45.68	41.97	**52.97**	35.08	62.12	1135	1282.25
MILP [35]	6.6	9.62	4.16	−0.2	10.5	−10.55	30.88	511	701.25
CAgent [24]	37.7	6.89	37.35	34.07	43.12	23.29	50.85	1032	1257.5

reflecting limited adaptability under dynamic grid conditions. Overall, the GNP-DT family effectively balances system survivability and operational cost, with *GNP-DT-2* offering the most cost-efficient control strategy.

L2RPN performance score. The L2RPN score quantifies the overall operational effectiveness and reliability of the agents across dynamic grid conditions. Fig. 6 illustrates the distribution of scores across 20 random seeds for all agents. Among the proposed variants, *GNP-DT-2* achieved the highest mean score of 48.10 with a median of 46.01, followed closely by *GNP-DT-3* (mean 46.99, median 45.68) and *GNP-DT-1* (mean 44.16, median 41.94). The higher median and tighter interquartile range of *GNP-DT-2* indicate both improved performance and enhanced stability, demonstrating the advantage of incorporating cost-awareness into the reward function. Among the comparative baselines, the *Enhanced ExpertOp4Grid* (mean 39.04) surpasses the original *ExpertOp4Grid* (mean 37.50) and the deep RL-based *CAgent* (mean 37.70), confirming the benefit of refined rule-based heuristics. The *CAgent* exhibits comparable spread to *ExpertOp4Grid*, ranging from 23.29 to 50.85, though with a slightly lower median. MILP, representing a DC optimization benchmark, demonstrates the widest variability and lowest scores overall, including negative values in some seeds, highlighting its instability and inefficiency in this context. Overall, the GNP-DT family—particularly *GNP-DT-2*—demonstrates superior and reliable performance across all tested configurations, balancing both resilience and cost efficiency.

Table 5 provides a comprehensive comparison of the agents in terms of their average L2RPN score, score distribution, and survivability metrics. The GNP-DT variants consistently outperform all benchmarks, with *GNP-DT-2* achieving the highest average score (48.1), reflecting stable and effective control. Its score distribution is particularly

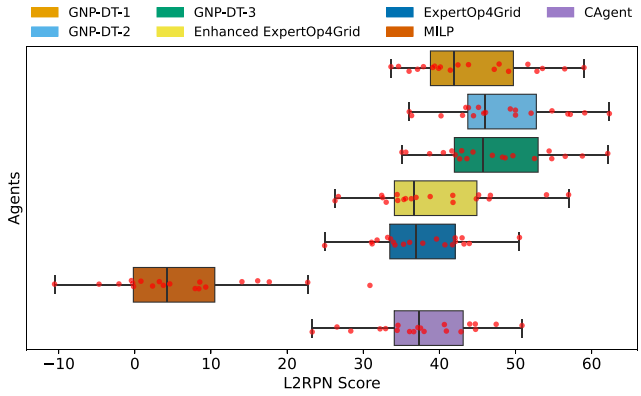


Fig. 6. Boxplot of the agent average score across all 20 seeds. Each point corresponds to the average score of all scenarios for one seed.

consistent, with a minimum of 36.04 and a third quartile of 52.73, while median survival (1147) and MSTCM (1439.5) indicate strong robustness across episodes (MSTCM is the median of these survival times across all episodes and all seeds, providing a measure of typical agent longevity). Compared to *CAgent* and *Enhanced ExpertOp4Grid*, *GNP-DT-2* improves the average L2RPN score by up to 28%, *GNP-DT-3* by 25%, and *GNP-DT-1* by 18%. The superior performance of *GNP-DT-2* agent trained with the cost-aware reward not only achieved a higher cumulative score but also demonstrated longer average survival times. This indicates that minimizing operational cost indirectly reinforces grid survivability, as blackout costs dominate the total episode

penalty. Consequently, the cost-aware reward yields both economically and operationally efficient control policies. *GNP-DT-3* also performs strongly (mean = 46.99, median survival = 1135, MSTCM = 1282.25), demonstrating the benefits of hyperparameter optimization even with the survival-duration reward. *GNP-DT-1* achieves a mean score of 44.16 and median survival of 1034, already surpassing conventional benchmarks and confirming the progressive improvement across the variants. Among the baseline agents, *CAgent* and *Enhanced ExpertOp4Grid* show competitive scores (37.7 and 39.04, respectively), with *CAgent* attaining a higher MSTCM (1257.5) than *ExpertOp4Grid* (930.75). In contrast, *MILP* exhibits low reliability (mean = 6.6, median survival = 511, MSTCM = 701.25), and the *Do Nothing* agent scores zero with minimal survival. Overall, GNP-DT-2 achieves the highest performance and robustness, demonstrating that cost-aware training effectively balances score maximization and operational longevity, while other agents either underperform or show greater variability in survival.

Inference time. The average time required to compute and apply an action varies significantly across agents. The proposed GNP-DT agent demonstrates the fastest inference time, requiring only 0.11 s on average, making it highly suitable for real-time deployment. In comparison, the *ExpertOp4Grid* baseline responds in 0.54 s, while both the *Enhanced ExpertOp4Grid* and *CAgent* take longer, averaging 2.23 and 2.33 s, respectively. The MILP method is the slowest, with an average inference time of 20.58 s, an expected outcome given that it requires solving an MILP problem at each timestep. These results highlight the advantage of GNP-DT in real-time decision making under operational time constraints.

5. Interpretability: A discussion

This section discusses the interpretability of the GNP-DT method during both the learning and operational phases.

5.1. Learning phase

To provide some insight into the reasoning window of the learning phase, Tables 6 and 7 together illustrate three decision graphs G_a, G_b, and G_c to address line congestion in a specific period of an episode. In Table 6, HB, LB, and DSB stand for hub bus, buses on the looped path, and downstream buses (Table 1 defines each type of these buses), respectively. In addition, the numbers in the detection rows represent the congested lines. The behavior of each graph is shaped by its internal judgment nodes and their relevant functions and parameters, as mentioned in Table 1 in Section 2.1. The key parameters that distinguish the graphs are presented in Table 7, which can cause changes in the functions of the graphs as follows:

- The threshold I_{th} defines the overload limit used to trigger the action proposal process; both G_a and G_c use a strict threshold (100%), while G_b lowers this to 90%, allowing for a proactive action.
- The parameter β_{th}, which sets the threshold coefficient for determining the influence path, is highest in G_b (0.6), indicating a more rigorous but potentially narrower focus in selecting effective paths. In contrast, G_c balances this (0.4), leading to more robust but flexible action pathways. For example, with the same detection criterion, when congestion starts on lines 12 and 20 and is followed by lines 41, 44 and 51, the flexible units activated by G_c were more effective in alleviating congestion at each timestep, the trajectory of actions (bus 67 → bus 81 → bus 93 → bus 95) outperformed the selection of actions (bus 68 → bus 76 → lines 24 and 146) by G_a.
- The σ^1_{cp} mentioned in Table 1 stands for the actions with the highest priority, where G_a and G_c give the highest focus to the buses at the hub points, and G_b has the most tendency to reconfigure the buses on the looped path, which was in some timesteps a less effective selection.
- The quality of the action in terms of the topological score is indicated by σ_t as mentioned in Table 1. In this regard, releasing a critical congestion is satisfying for G_a and G_b, while G_c demands full topological resolution of all congestion, critical or not, before terminating the search. This stricter criterion ensures that G_c systematically avoids the risk of creating new congestions and provides a safer solution that relieves all overloads.

The observed action trajectory confirms these architectural differences. G_a acts sequentially on hub buses but fails to prevent collapse due to late response and incomplete resolution. G_b detects earlier and applies diverse actions, including looped and downstream bus reconfigurations, surviving the scenario later because it raises more overload concerns with its action trajectory. G_c, however, mirrors G_a in the type of action but distinguishes itself by selecting actions that fully mitigate system risk, leading to survival without creating new overloads. These comparisons demonstrate the interpretability of the proposed method during the learning phase and support human operators in reasoning about and building trust in the resulting elite graphs. For example, this traceable possibility in GNP can provide the operator with such a sensitivity analysis on the graph parameters that justifies the superiority of G_c over G_a and G_b.

In addition, this example provides insight into the evolutionary phase of the GNP. The initial seed corresponds to the baseline expert system (Ga). Through evolutionary operators and reinforcement-learning-based fitness evaluation, the GNP progressively adapts, producing intermediate graphs (Gb) and ultimately an elite graph (Gc). Comparative results show that Gc consistently outperforms both Ga and Gb, confirming that the performance gains are attributable to the learning and adaptation process rather than to the expert initialization alone. This distinction highlights that while the expert provides a structured starting point, the final evolved agent embodies novel strategies that emerge through the learning process.

5.2. Operational phase

To demonstrate the interpretability of the proposed method during the operational phase, a representative portion of the complete DT is illustrated in Fig. 7. This selected partial tree is from the DT3 model (bus reconfiguration), which is relevant to determining the ID of the flexible unit (DT2) when the flexibility type (DT1) and the number of needed actions (DT2) have been previously predicted.

In the DT plot, *value* attribute represents the proportional distribution of these instances across classes, and the *class* attribute corresponds to the majority class, indicating the predicted outcome at that node. Each box's color represents the dominant class in that node. Nodes sharing the same color are mostly associated with the same class. The intensity of the color indicates the node's purity – darker shades mean the node contains mostly one class, while lighter shades suggest a mix of classes. This visual aid helps quickly identify which class is favored and how confidently the tree makes its splits. Moreover, left arrows correspond to *True*, and right arrows correspond to *False* for each split.

This example determines the next control action based on the grid congestion state and prior topology changes. *Cong. Line ID_1* represents the identifier of the line with the highest loading percentage (i.e., most congested) at the current timestep, followed by *Cong. Line ID_2*, *Cong. Line ID_3*, etc., sorted in descending order based on their loading levels. *Split Bus ID_i* indicates the ID of a bus that has already been reconfigured, sorted by ID. The *class* output of the tree corresponds to the ID of a candidate bus selected for reconfiguration. Due to limited space, the tree was plotted for only three classes using reconfiguration buses 16, 67, and 93. Some illustrative human-readable rules could be extracted from the DT as follows:

1. If *Cong. Line ID_1* is in $Z1$ (i.e., the line with the maximum overload is situated in zone $Z1$) and *Cong. Line ID_2* is in $Z1$ or $Z2$, the model selects **bus 67** from $Z1$ as the next reconfiguration candidate—that is, the bus proposed for splitting to alleviate congestion at the current timestep.

Table 6
Comparison of decision graph action trajectory corresponds to the line congestion detection across 9 illustrative timesteps.

Time step	1	2	3	4	5	6	7	8	9
G_a: Detection	–	–	{12, 20}	{20}	{41, 44, 51, 175, 185}	Collapse	Collapse	Collapse	Collapse
G_a: Action	–	–	Reconfig. HB 68	Reconfig. HB 76	Disconn. {24, 146}	–	–	–	–
G_b: Detection	{20}	{41, 51}	{41, 51}	{12}	{12}	{41, 44, 51}	{175}	{175}	–
G_b: Action	Reconfig. LB 81	Disconn. {24, 16}	Reconfig. LB 79	Disconn. {13, 122}	Reconfig. LB 79	Reconfig. DSB 95	Reconfig. DSB 67	Reconfig. DSB 79	–
G_c: Detection	–	–	{12, 20}	{20}	{41, 44, 51}	{41}	–	–	–
G_c: Action	–	–	Reconfig. HB 67	Reconfig. HB 81	Reconfig. HB 93	Reconfig. HB 95	–	–	–

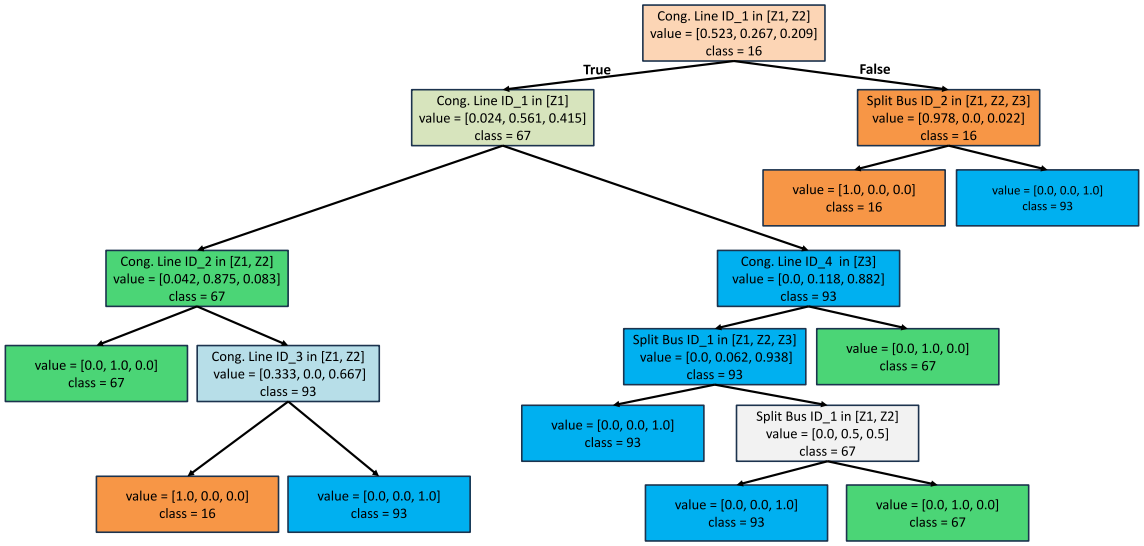


Fig. 7. Excerpt from DT3 (bus reconfig.) for flexible unit ID determination. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 7
Summary of functional differences between decision graphs.

Aspect	G_a	G_b	G_c
I_{th}	100%	90%	100%
β_{th}	0.2	0.6	0.4
σ^1_{cp}	Hub	Loop	Hub
σ_t	Critical issue	Critical issue	All issues

2. If *Cong. Line ID_1* is in $Z2$, and *Cong. Line ID_4* in $Z3$, and *Split Bus ID_1* in $\{Z1, Z2, Z3\}$, then the model returns **bus 93** from $Z3$ for splitting.
3. If *Cong. Line ID_1* out of $\{Z1, Z2\}$ and *Split Bus ID_2* is in $\{Z1, Z2, Z3\}$, **bus 16** from $Z2$ is the candidate for reconfiguration action (bus splitting).
4. If *Cong. Line ID_1* is in $Z2$, and *Cong. Line ID_4* is in $\{Z1, Z2, Z3\}$, then the model selects **bus 67** from $Z1$ for bus splitting.

The interpretability achieved through the DT extraction is primarily logical, as it expresses the agent's control policy in the form of explicit "if–then" decision rules. Although such interpretability does not constitute causal reasoning in the strict analytical sense, it provides a practical and meaningful explanation of the agent's actions that aligns with human operator expertise. Each path in the DT directly links observable grid states to corresponding control decisions, allowing operators to trace why a particular action is recommended under specific system conditions and another action is suitable under a different operating condition. For example, a rule such as "If congestion occurs in Zone-1 and an overload also exists in Zone-2, then split Bus-X and Bus-Y could be reconfigured only if the congestion is in Zone-1" demonstrates the underlying decision logic that connects system observations to a physically interpretable control measure. This transparency enables operators to validate, trust, and refine the automated control behavior, bridging the gap between data-driven intelligence and human operational understanding. Consequently, the proposed interpretability mechanism provides actionable insight into the decision-making process, even though it remains logically rather than causally explanatory.

6. Conclusions

This work proposed an interpretable and adaptive control framework based on Genetic Network Programming with Decision Trees (GNP-DT) for real-time congestion management in power systems. The agent integrates rule-based structures with RL to evolve effective control strategies, maintaining both interpretability and adaptability. Through extensive evaluation across multiple stochastic seeds and benchmark agents, GNP-DT demonstrated superior performance in terms of L2RPN score, survival time, and operational cost compared to baseline methods of different natures, including an expert system, MILP with DC approximation, and a deep RL-based agent (*CAgent*). In particular, GNP-DT achieved an improvement in the average score of up to 28% over *ExpertOp4Grid* and the deep RL agent and consistently ensured a higher median survival and a lower average operational cost.

Moreover, the framework maintained high computational efficiency, exhibiting significantly shorter inference times than optimization-based and deep learning approaches. The alignment of

GNP-DT performance with interpretable design principles supports its practical applicability in real-world scenarios that require transparent and reliable grid control. Future work could research the integration of multi-agent coordination and real-time memory sharing mechanisms to further enhance the adaptability and scalability of control strategies in large-scale power systems.

CRediT authorship contribution statement

Ferinar Moaidi: Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Data curation, Conceptualization. **Ricardo J. Bessa:** Writing – review & editing, Supervision, Resources, Methodology, Funding acquisition, Formal analysis, Conceptualization.

Code availability

The code is publicly available as open source through the GitHub repository of the AI4REALNET European project.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Ferinar Moaidi reports financial support was provided by Foundation for Science and Technology. Ferinar Moaidi reports financial support was provided by Horizon Europe. Ricardo Jorge Bessa reports financial support was provided by Horizon Europe. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is part of the AI4REALNET (*AI for REAL-world NETwork operation*) project, which received funding from European Union's Horizon Europe Research and Innovation programme under the Grant Agreement No 101119527, and from the Swiss State Secretariat for Education, Research and Innovation (SERI). This project is funded by the European Union and SERI. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union and SERI. Neither the European Union nor the granting authority can be held responsible for them. Ferinar Moaidi was supported by FCT (*Fundação para a Ciência e a Tecnologia*) within the Ph.D. Grant 2023.04869.BD.

Data availability

GitHub repository of the AI4REALNET European project.

References

[1] Marot A, Kelly A, Naglic M, Barbesant V, Cremer J, Stefanov A, Viebahn J. Perspectives on future power system control centers for energy transition. J Mod Power Syst Clean Energy 2022;10(2):328–44. http://dx.doi.org/10.35833/MPCE.2021.000673.

[2] Marot A, Donnot G, Kelly A, O'Sullivan A, Viebahn J, Awad M, Guyon I, Panciatici P, Romero C. Learning to run a power network challenge: a retrospective analysis. In: Proceedings of machine learning research. NeurIPS 2020 competition and demonstration track, vol. 133, 2021, p. 112–32. http://dx.doi.org/10.48550/arXiv.2103.03104.

[3] Mazi M, et al. Correction of overloads and voltage violations by corrective control. IEEE Trans Power Syst 1986;1(2):73–80. http://dx.doi.org/10.1109/TPWRS.1986.4334990.

[4] Bacher R, Glavitsch H. Network topology optimization with security constraints. IEEE Trans Power Syst 1986;1(4):103–11. http://dx.doi.org/10.1109/TPWRS.1986.4335024.

[5] Makram I, et al. Optimization of transmission line switching using the Z-matrix method. IEEE Trans Power Syst 1989;4(1):259–64. http://dx.doi.org/10.1109/59.193839.

[6] Chen S, Glavitsch H. Stabilizing switching. IEEE Trans Power Syst 1993;8(4):1511–7. http://dx.doi.org/10.1109/59.260953.

[7] Granelli G, et al. Optimal network reconfiguration for congestion management by deterministic and genetic algorithms. Electr Power Syst Res 2006;76(6–7):549–56. http://dx.doi.org/10.1016/j.epsr.2005.09.014.

[8] Fisher EB, et al. Optimal transmission switching. IEEE Trans Power Syst 2008;23(3):1346–55. http://dx.doi.org/10.1109/TPWRS.2008.922256.

[9] Heidarifar M, et al. An optimal transmission line switching and bus splitting heuristic incorporating AC and N-1 contingency constraints. Int J Electr Power Energy Syst 2021;133:107278. http://dx.doi.org/10.1016/j.ijepes.2021.107278.

[10] Zhou Y, et al. Substation-level grid topology optimization using bus splitting. 2020, http://dx.doi.org/10.48550/arXiv.2009.14418, arXiv.

[11] Zhang Y, Liu Y. Multi-period optimal transmission switching with voltage stability and security constraints. Sustainability 2024;16(18):8272. http://dx.doi.org/10.3390/su16188272.

[12] Tavakkoli MA, Amjady N. Incorporating bus-bar switching actions into AC optimal power flow to avoid over-current status. Sci Iran 2019;26(1):206–13. http://dx.doi.org/10.24200/sci.2019.54166.3625.

[13] Pei Z, Rojas-Arevalo AM, de Haan FJ, Lipovetzky N, Moallemi EA. Reinforcement learning for decision-making under deep uncertainty. J Environ Manag 2024;359:120968. http://dx.doi.org/10.1016/j.jenvman.2024.120968.

[14] Wang X, Zhong H, Zhang G, Ruan G, He Y, Yu Z. Look-ahead AC optimal power flow: A model-informed reinforcement learning approach. 2023, arXiv preprint arXiv:2303.02306. URL: https://doi.org/10.48550/arXiv.2303.02306.

[15] Wu T, Scaglione A, Arnold D. Constrained reinforcement learning for predictive control in real-time stochastic dynamic optimal power flow. 2023, http://dx.doi.org/10.48550/arXiv.2302.10382, arXiv preprint arXiv:2302.10382.

[16] Awais MU. Using deep reinforcement learning to solve optimal power flow problem with generator failures. 2022, http://dx.doi.org/10.48550/arXiv.2205.02108, arXiv preprint arXiv:2205.02108.

[17] Li J, Zhang R, Wang H, Liu Z, Lai H, Zhang Y. Deep reinforcement learning for optimal power flow with renewables using graph information. 2021, http://dx.doi.org/10.48550/arXiv.2112.11461, arXiv preprint arXiv:2112.11461.

[18] Marot A, Donon B, Guyon I, Donnot B. Learning to run a power network competition. In: CiML workshop. NeurIPS, Montréal, Canada; 2018, URL: https://hal.science/hal-01968295.

[19] Donnot B. Grid2op- A testbed platform to model sequential decision making in power systems. 2020, URL: https://GitHub.com/Grid2Op/grid2op.

[20] Omnes L, Marot A, Donnot B. Adversarial training for a continuous robustness control problem in power systems. In: 2021 IEEE madrid PowerTech. 2020, p. 1–6. http://dx.doi.org/10.48550/arXiv.2012.11390.

[21] Marot A, Guyon IM, Donnot B, Dulac-Arnold G, Panciatici P, Awad M, O'Sullivan A, Kelly A, Hampel-Arias Z. L2RPN: Learning to run a power network in a sustainable world NeurIPS2020 challenge design. 2020, URL: https://api.semanticscholar.org/CorpusID:243830891.

[22] Zhou B, Zeng H, Liu Y, Li K, Wang F, Tian H. Action set based policy optimization for safe power grid management. In: ECML/PKDD. 2021, http://dx.doi.org/10.48550/arXiv.2106.15200.

[23] Chauhan A, Baranwal M, Basumatary A. PowRL: A reinforcement learning framework for robust management of power networks. In: Proceedings of the AAAI conference on artificial intelligence. vol. 37, 2023, p. 14757–64. http://dx.doi.org/10.1609/aaai.v37i12.26724, 12.

[24] Lehna M, Viebahn J, Marot A, Tomforde S, Scholz C. Managing power grids through topology actions: A comparative study between advanced rule-based and reinforcement learning agents. Energy AI 2023;14:100276. http://dx.doi.org/10.1016/j.egyai.2023.100276.

[25] Marot A, Donnot B, Tazi S, Panciatici P. Expert system for topological remedial action discovery in smart grids. In: Mediterranean conference on power generation, transmission, distribution and energy conversion. MEDPOWER 2018, 2018, p. 1–6. http://dx.doi.org/10.1049/cp.2018.1875.

[26] Mabu S, Hirasawa K, Obayashi M, Kuremoto T. Enhanced decision making mechanism of rule-based genetic network programming for creating stock trading signals. Expert Syst Appl 2013;40(16):6311–20. http://dx.doi.org/10.1016/j.eswa.2013.05.037.

[27] Yoon D, Hong S, Lee B-J, Kim K-E. Winning the L2{RPN} challenge: Power grid management via semi-Markov afterstate actor-critic. In: International conference on learning representations. 2021, URL: https://openreview.net/forum?id=LmUJqB1Cz8.

[28] Puterman ML. Markov decision processes: Discrete stochastic dynamic programming. Hoboken, NJ: John Wiley & Sons; 2005, http://dx.doi.org/10.1002/9780470316887.

[29] Île-de-France Region and RTE. Paris region AI challenge for energy transition: Low-carbon grid operations. 2023, URL: https://www.iledefrance.fr/sites/default/files/medias/2023/05/Description_Challenge_RTE.pdf. [Accessed 03 October 2025].

[30] Sutton RS, Barto AG. Reinforcement learning: An introduction. 2nd ed.. MIT Press; 2018, URL: http://incompleteideas.net/book/the-book-2nd.html.

[31] Bessa RJ, Moaidi F, Viana J, Andrade JR. Uncertainty-aware procurement of flexibilities for electrical grid operational planning. IEEE Trans Sustain Energy 2024;15(2):789–802. http://dx.doi.org/10.1109/TSTE.2023.3305865.

[32] Binbin C. Teacher-tutor-junior student-senior student. 2025, URL: https://github.com/Aspirin96/L2RPN_NIPS_2020_a_PPO_Solution. GitHub.

[33] Grid2Op Documentation. Available environments — Grid2op 1.10.4 documentation. 2025, https://grid2op.readthedocs.io/en/latest/available_envs.html. [Accessed 28 May 2025].

[34] Paulos J, Silva P, Bessa R, Marot A, Dejaegher J, Donnot B. Generation of power network operating scenarios for an AI-friendly digital environment. In: IEEE PowerTech 2025 conference. Kiel, Germany; 2025.

[35] Grid2op Contributors. grid2op-milp-agent: A MILP-based grid topology control agent for the grid2op environment. 2025, https://github.com/Grid2op/grid2op-milp-agent. [Accessed 07 June 2025].

[36] Forrest J, et al. CBC (coin-or branch and cut). 2020, Version 2.10.5, COIN-OR. https://github.com/coin-or/Cbc.

[37] Sintes J. How we built the winning real time autonomous agent for power grid management in the L2RPN challenge 2023. 2024, URL: https://medium.com/@lajavaness/how-we-built-the-winning-real-time-autonomous-agent-for-power-grid-management-in-the-l2rpn-41ab3cfaddbd. Medium.