GRNN-based Real-time Fault Chain Prediction

Anmol Dwivedi and Ali Tajer, *Senior Member, IEEE*

***Abstract*—This paper proposes a data-driven graphical framework for the real-time search of risky cascading fault chains (FCs). While identifying risky FCs is pivotal to alleviating cascading failures, the complex spatio-temporal dependencies among the components of the power system render challenges to modeling and analyzing FCs. Furthermore, the real-time search of risky FCs faces an inherent combinatorial complexity that grows exponentially with the size of the system. The proposed framework leverages the recent advances in graph recurrent neural networks to circumvent the computational complexities of the real-time search of FCs. The search process is formalized as a partially observable Markov decision process (POMDP), which is subsequently solved via a time-varying graph recurrent neural network (GRNN) that judiciously accounts for the inherent temporal and spatial structures of the data generated by the system. The key features of this structure include (i) leveraging the spatial structure of the data induced by the system topology, (ii) leveraging the temporal structure of data induced by system dynamics, and (iii) efficiently summarizing the system's history in the latent space of the GRNN. The proposed framework's efficiency is compared to the relevant literature on the IEEE 39-bus New England system and the IEEE 118-bus system.**

I. Introduction

Large-scale disruptions in power systems generally follow a sequence of less severe anomalous events that gradually stress the system over time. Various reports on the events preceding blackouts indicate that system operators are either unaware of these gradual changes or oblivious to the contingencies following these changes. Specifically, such small-scale anomalies can lead to hidden failures that propagate and eventually result in large-scale monitoring, and control disruptions. For instance, in a report by the North American Electric Reliability Corporation [1], it was concluded that inadvertent tripping of a power line led to a series of failures causing the 2003 blackout in North America. Therefore, forming *real-time and accurate* situational awareness in power systems has a pivotal role in ensuring a secure and reliable power system operation.

In this paper, we formalize a graphical framework for dynamically predicting the chains of risky faults a power system faces. A fault chain (FC) is a sequence of consecutive component outages that captures the temporal evolution of a cascading outage process. Due to the combinatorially extensive number of possible failures, which grows with the system size and failure horizon, finding critical failure sequences is computationally challenging. Since a FC captures the high-impact and rear-occurrence characteristic of cascading failures in any dynamically-changing power system partly, timely identification of the riskiest FCs to system operators for any given system state is instrumental to predicting failures and preventing cascading failures.

Predicting FCs and assessing their risks can be formalized by (i) creating an exhaustive list of all possible failure scenarios up to a specific time horizon, (ii) evaluating the disruption (e.g., load loss) caused by each, and (iii) evaluating the likelihood of each scenario. Accomplishing these three tasks faces the following two key challenges. First, the space of scenarios grows exponentially fast with the power system size and time horizon, rendering listing all scenarios computationally prohibitive even for moderate network sizes and target horizons. Secondly, the system changes dynamically with high unpredictability, which necessitates constantly updating the scenario space. Before specifying our approach, we review the existing literature relevant to this paper's scope.

A. Literature Review

In this subsection, we provide an overview of the literature most closely related to the scope of this paper. The study in [2] aims to identify and quantify all the vulnerable sections of the power system that can potentially lead to cascading failures. Specifically, it develops a FC framework for risky FC identification to address this. In [3], a rapid stochastic procedure is proposed to yield large collections of high-risk FCs. Despite the effectiveness of [3], such a method faces a computational bottleneck addressed by the study in [4], which proposes eliminating a large number of redundant constraints to make the contingency screening more efficient. With similar motivation, [5] formulates a bi-level optimization problem to gain a higher evaluation efficiency for risky FC search and [6] employs a sequential importance sampling algorithm to acquire critical FCs from cascading outage simulations. Due to a variety of other benefits associated with identifying a collection of risky FCs, FC search algorithms are employed for risk-assessment [7]–[9], risk mitigation [10], and vulnerable component identification [11], [12] and others [13].

There exist a number of machine learning (ML) approaches that aim to enhance the efficiency of risky FC search. Broadly, these models quantify the vulnerability of each power system component from simulated power system operational data by learning data-driven models. For instance, the study in [14] formulates the search for risky FCs in a Markov decision process (MDP) environment and employs reinforcement learning (RL) algorithms to find risky FCs. The investigation in [15] employs deep neural networks to obtain critical states during cascading outages, and [16] employs convolutional neural networks for faster contingency screening. The study in [12] proposes a transition-extension approach that builds upon the RL approach in [14] to make it amenable to real-time implementation. This is facilitated by exploiting the similarity between adjacent power-flow snapshots. Finally, [17] proposes

The authors are with the Electrical, Computer, and Systems Engineering Department, Rensselaer Polytechnic Institute, Troy, NY.

This research was supported in part by the U. S. National Science Foundation under the CAREER Award ECCS-1554482 and the Rensselaer-IBM Artificial Intelligence Research Collaboration program.

to employ graph convolutional neural networks that leverage the grid's topology to identify cascading failure paths.

In parallel to ML-based approaches, there also exist model-based approaches that quantitatively and qualitatively model, analyze and simulate large-scale cascading outage processes. This is done by developing simulation models that capture the power system physics. The first category of studies aims to develop high-level statistical models to facilitate faster computation and quick inference. Generally, these are data-driven models such as CASCADE [18], the branching process model [19], the interaction model [20], and models based on influence graphs [21], [22]. While these approaches are quick in revealing important quantitative properties of cascading outages and often lead to interpretable conclusions due to their ability to capture the non-local behavior of the cascading outage process, such models cannot accommodate the time-varying interactions among components at different stages of a cascade. To tackle this challenge, detailed failure models such as the OPA [23], improved OPA [24], random chemistry model [3] and others [25], [26] are studied that precisely model the AC power-flow and power dispatch constraints of the components in the grid while simulating the cascading outage process. Despite their detailed modeling and effectiveness, these models are computationally intensive, a major impediment to their adoption for real-time implementation.

B. Contribution

Despite the effectiveness of the aforementioned models, these models broadly face the following challenges: (i) The models fail to capture the concurrent spatio-temporal dependencies across the time horizon among the components of the power system under dynamically changing network topologies. (ii) The cascading outage process is assumed to follow Markov property. While this simplification can render reasonable approximations during the earlier stages of the cascading failure, the later stages of the failure process typically exhibit temporal dependencies beyond the previous stage, making the Markovian assumptions inadequate. (iii) These models become prohibitive even for moderate grid sizes due to the combinatorial growth in either the computational or storage requirements with the number of components in the grid.

We propose a data-driven graphical framework for efficiently identifying risky cascading FCs to address the aforementioned challenges associated with the ML approaches. The proposed framework designs a graph recurrent neural network (GRNN) to circumvent the computational complexities of the real-time search of FCs. The search process is formalized as a partially observable Markov decision process (POMDP), which is subsequently solved via a time-varying GRNN that judiciously accounts for the inherent temporal and spatial structures of the data generated by the system. The key features of this structure include (i) leveraging the spatial structure of the data induced by the system topology, (ii) leveraging the temporal structure of data induced by system dynamics, and (iii) efficiently summarizing the system's history in the latent space of the GRNN, rendering the modeling assumptions realistic and the approach amenable to real-time implementation.

Finally, we highlight the difference between our proposed approach and the graph neural network (GNN)-based approach investigated in [17]. The goal in [17] is to detect all cascading failure paths that lead to load-shedding in a limited number of search attempts. Such a decision is *binary* since a cascading failure path may either lead to load shedding or not. In contrast, our approach aims to identify cascading failure paths with *maximum risk* (FCs with maximum load-shed) in a limited number of search attempts where, ideally, the most critical cascading failure paths should be identified **earlier** than the relatively less critical ones. While related, the problem descriptions and the attendant solutions are distinct.

II. Problem Formulation

A. System Model

Consider a power system consisting of N buses. To capture the interconnectivity of the system, we represent it by an undirected graph $\mathcal{G} \triangleq (V, E)$, where the set of vertices $V \triangleq [N] \triangleq \{1, \ldots, N\}$ represents the buses and the edge set $E \subseteq V \times V$ represents the transmission lines. We denote the binary adjacency matrix of $\mathcal{G}$ by $\mathbf{B} \in \{0,1\}^{N \times N}$ such that $b_{u,v} \triangleq [\mathbf{B}]_{u,v} = 1$ indicates there exists at least one transmission line between buses u and v (e.g., in the case of parallel transmission lines). We define $\mathbf{X} \in \mathbb{R}^{N \times F}$ as the system state matrix of the grid, which compactly represents F system state parameters (e.g., voltage angles, injected real power) for all the N buses and we refer to any system state parameter $f \in \{1, \ldots, F\}$ of bus $u \in [N]$ by $x_{u,f} \triangleq [\mathbf{X}]_{u,f}$.

B. Modeling Fault Chains

Depending on an array of internal (e.g., system instabilities) and external (e.g., weather) conditions, the system faces a degree of risk in disruptions that may lead to component outages. We focus on transformers and transmission lines as the components of interest. When the outages are substantial enough, they can lead to more outages, resulting in a FC. Our objective is to dynamically identify the FCs that the system faces and assess their associated risks (e.g., load losses). In this subsection, we formalize a model for FCs and their risks based on an objective formalized in the next subsection.

Consider the topology of a generic outage-free system that precedes an FC given by $\mathcal{G}_0$, and denote the associated system state by $\mathbf{X}_0$. A generic FC that the system might be facing is specified as a sequence of consecutive component outages that represents a cascading outage process. Consider a FC model that consists of at most P stages, where P can be selected based on the horizon of interest for risk assessment, and denote the set of all components in the system by $\mathcal{U}$. It is noteworthy that the number of stages in different FCs can be distinct. In cases that a fault chain terminates before $\tilde{P} < P$ stages, it means that the sets $\{\mathcal{U}_{\tilde{P}+1}, \ldots, \mathcal{U}_P\}$ will be empty sets.

In each stage $i \in [P]$, a number of additional components fail. We define $\mathcal{U}_i$ as the set of components that fail in stage $i \in [P]$. We note that the set $\mathcal{U}_i$ could consist of more than one component failures in any stage i, i.e., $|\mathcal{U}_i| \geq 1$. We also note that when a transmission line fails and it has parallel channels,

only the failed line will be included in $\mathcal{U}_i$, and its parallel lines will be retained as functioning components. Clearly, $\mathcal{U}_i \subseteq \mathcal{U}\backslash\{\cup_{j=1}^{i-1}\mathcal{U}_j\}$, and the sequence of components that fail in P stages is specified by a FC sequence $\mathcal{V}$

$$\mathcal{V} \triangleq \langle \mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_P \rangle \ . \tag{1}$$

Additionally, we denote the healthy components in any stage i of the FC by $\ell_i \in \mathcal{U}\backslash\{\cup_{j=1}^{i-1}\mathcal{U}_j\}$. We note that when the failure process is slow (e.g., in the earlier stages of a cascading failure), the sets $\mathcal{U}_i$ have fewer components, and when the process is fast (e.g., last stages of a cascading failure), these sets are more populated since component outages are usually grouped depending on their time of occurrence [8], [9], [21].

Due to component outages in each stage i, the system's topology alters to $\mathcal{G}_i \triangleq (V_i, E_i)$ with an associated adjacency matrix $\mathbf{B}_i$ and an underlying system state $\mathbf{X}_i$. Consecutive failures lead to compounding stress on the remaining components, which need to ensure minimal load shedding in the network. Nevertheless, when the failures are severe enough, they can lead to load losses. We denote the load loss (LL) imposed by the component failures in $\mathcal{U}_i$ in stage i by $\mathsf{LL}(\mathcal{U}_i)$, i.e.,

$$\mathsf{LL}(\mathcal{U}_i) \triangleq \mathsf{load}(\mathcal{G}_{i-1}) - \mathsf{load}(\mathcal{G}_i) \ , \tag{2}$$

where $\mathsf{load}(\mathcal{G}_i)$ is the total load (in MWs) when the system's state in stage i is associated with the topology $\mathcal{G}_i$. Accordingly, we define the total load loss (TLL) imposed by the FC $\mathcal{V}$ by

$$\mathsf{TLL}(\mathcal{V}) \triangleq \sum_{i=1}^{P} \mathsf{LL}(\mathcal{U}_i) \ . \tag{3}$$

C. Problem Statement

Due to the re-distribution of power across transmission lines after each stage of the FC, some FC sequences particularly lead to substantial risks and owing to the continuously time-varying system's state, different loading and topological conditions face different risks. Hence, it is important for system operators to find, efficiently and in real-time, the set of FCs with the largest TLL associated with any given initial system state $\mathbf{X}_0$.

Our objective is to identify S number of FC sequences, each consisting of P stages, that impose the largest TLL. To formalize this, we define $\mathcal{F}$ as the set of all possible FCs with a target horizon of P, and our objective is to identify S members of $\mathcal{F}$ with the largest associated losses. We denote these S members by $\{\mathcal{V}_1^*, \ldots, \mathcal{V}_S^*\}$. Identifying the sets of interest can be formally cast as solving

$$\mathcal{P}: \quad \{\mathcal{V}_1^*, \ldots, \mathcal{V}_S^*\} \triangleq \underset{\{\mathcal{V}_1, \ldots, \mathcal{V}_S\} \,:\mathcal{V}_i \in \mathcal{F}}{\arg\max} \sum_{i=1}^{S} \mathsf{TLL}(\mathcal{V}_i) \ . \tag{4}$$

Problem $\mathcal{P}$ aims to maximize the accumulated TLL due to the S number of FC sequences. Without loss of generality, we assume that the TLLs of the set of sequences $\{\mathcal{V}_1^*, \ldots, \mathcal{V}_S^*\}$ are in the descending order, i.e., $\mathsf{TLL}(\mathcal{V}_1^*) \geq \mathsf{TLL}(\mathcal{V}_2^*) \geq \cdots \geq \mathsf{TLL}(\mathcal{V}_S^*)$. Solving $\mathcal{P}$ faces a significant computational challenge since the cardinality of $\mathcal{F}$ grows exponentially with the system size N, the number of components $|\mathcal{U}|$, and the risk assessment horizon P.

III. Sequential Search via POMDPs

A. Sequential Search

To circumvent the complexity of solving $\mathcal{P}$ in (4), we design an agent-based learning algorithm that sequentially constructs the set of FCs $\{\mathcal{V}_1^*, \ldots, \mathcal{V}_S^*\}$. Specifically, the agent starts with constructing $\mathcal{V}_1^* \triangleq \langle \mathcal{U}_{1,1}, \ldots, \mathcal{U}_{1,P} \rangle$ such that it sequentially identifies the sets $\{\mathcal{U}_{1,1}, \ldots, \mathcal{U}_{1,P}\}$ in each stage $i \in [P]$ as follows. The agent admits the topology and the system state as its initial baseline inputs, denoted by $\mathcal{G}_0$ and $\mathbf{X}_0$, respectively. In the first stage, the agent identifies *the components* in $\mathcal{U}$ removing which is expected to impose the most intense TLL. To control the complexity and reflect the reality of FCs, in which failures occur component-by-component, we are interested in identifying only one component in each stage. Nevertheless, due to the physical constraints, removing one component can possibly cause outages in one or more other components in the same stage. We denote the set of all components to be removed in the first stage by the set $\mathcal{U}_{1,1}$.

The risk associated with each candidate set $\mathcal{U}_{1,1}$ has two, possibility opposing, impacts. The first pertains to the immediate loss due to component failures in $\mathcal{U}_{1,1}$, and the second captures the losses associated with the future possible failures driven by the failures in $\mathcal{U}_{1,1}$. Hence, identifying the sets $\mathcal{U}_{1,1}$ involves look-ahead decision-making and cannot be carried out greedily based on only the immediate LLs. Once the set $\mathcal{U}_{1,1}$ is identified (via our proposed Algorithm 1 the details of which we discuss in Section IV-C), the agent removes all the components in this set to update the grid topology to $\mathcal{G}_1$, and uses simulations (by solving a power-flow or an optimal power-flow, if necessary) to determine the associated system state $\mathbf{X}_1$.

Subsequently, $\mathcal{G}_1$ and $\mathbf{X}_1$ are leveraged to identify the set $\mathcal{U}_{1,2}$ by removing *the components* from the set $\mathcal{U}\backslash\{\mathcal{U}_{1,1}\}$, and this process continues recursively for a total of P stages, at the end of which the set $\mathcal{V}_1^*$ is constructed. Subsequently, a similar process is repeated to construct $\mathcal{V}_2^*$ and the algorithm repeats this process S times to identify S sequences of interest. While repeating the search process, it is important that the agent avoids finding the same FC sequences over and over again that were discovered previously. Accordingly, as discussed in detail in Section IV-D, we alter the agent decision process to take into account the number of times any given component was removed. Fig. 1 illustrates a search process where an agent constructs a FC sequence $\mathcal{V}_s^* = \langle \ell_1^1, \ell_2^2, \ell_3^1 \rangle$ by leveraging the current system state $(\mathcal{G}_i, \mathbf{X}_i)$ in each stage $i \in [3]$.

B. Modeling Search as a POMDP

The cascading outage process renders temporal dependencies across outage stages, typically spanning more than two stages. The full extent of such dependencies might be hidden in the observed (time) domain. We leverage the hidden dependencies in the latent (hidden) space of the fault chain generation process. Since the observation at each stage provides only *partial* information for decision making, we formalize the agent decision process at every stage of the search process by a partially observed Markov decision process (POMDP). In our search process, to control the computational complexity, we

determine the system state in stage $(i+1)$, i.e., $(\mathcal{G}_{i+1}, \mathbf{X}_{i+1})$ by leveraging the system state $(\mathcal{G}_i, \mathbf{X}_i)$ in stage i. Nevertheless, the load loss at stage $(i+1)$ depends on all the past i stages and the set of components removed in those system states. In this subsection, we characterize the POMDP of interest, and address solving it in Section IV.

We denote the partial observation that the agent uses at stage i to determine the system state at stage $i+1$ by $\mathbf{O}_i \triangleq (\mathcal{G}_i, \mathbf{X}_i)$. Accordingly, we define the sequence

$$\mathbf{S}_i \triangleq \langle \mathbf{O}_0, \ldots, \mathbf{O}_i \rangle \ , \tag{5}$$

which we refer to as the POMDP state at stage i, and it characterizes the entire past sequence of observations that render $\mathbf{O}_{i+1}$. As stated earlier, at stage i only $\mathbf{O}_i$ is known to the agent. At stage i, upon receiving the observation $\mathbf{O}_i$, the agent aims to choose a component from the set of *available* components to be removed in the next stage. To formalize this process, we define the agents action as its choice of the component of interest. We denote the action at stage i by a_i. Accordingly, we define the action space $\mathcal{A}_i$ as the set of all remaining components, i.e., $\mathcal{A}_i \triangleq \mathcal{U} \backslash \{\cup_{j=1}^{i-1} \mathcal{U}_j\}$. Once the agent takes an action $a_i \in \mathcal{A}_i$ in stage i, the underlying POMDP state in next stage is randomly drawn from a transition probability distribution $\mathbb{P}$

$$\mathbf{S}_{i+1} \sim \mathbb{P}(\mathbf{S} \mid \mathbf{S}_i, a_i) \ . \tag{6}$$

Probability distribution $\mathbb{P}$ captures the randomness due to the power system dynamics, and it is determined by the generator re-dispatch strategy in each stage of the FC. To quantify the risk associated with taking action a_i in POMDP state $\mathbf{S}_i$ when transitioning to $\mathbf{S}_{i+1}$, we define an instant reward r_i

$$r_i \triangleq r(\mathbf{S}_{i+1} \mid \mathbf{S}_i, a_i) \triangleq \mathsf{load}(\mathcal{G}_i) - \mathsf{load}(\mathcal{G}_{i+1}) \ . \tag{7}$$

Hence, for any generic action selection strategy π, the aggregate reward collected by the agent starting from the baseline POMDP state can be characterized by a value function

$$V_\pi(\mathbf{S}_0) \triangleq \sum_{i=0}^{P-1} \gamma^i \cdot r(\mathbf{S}_{i+1} \mid \mathbf{S}_i, \pi(\mathbf{O}_i)) \ , \tag{8}$$

where the discount factor $\gamma \in \mathbb{R}_+$ decides how much future rewards are favored over instant rewards, and $\pi(\mathbf{O}_i)$ denotes the action selected by the agent given an observation $\mathbf{O}_i$ in stage $i \in [P]$. Therefore, finding an optimal action selection strategy π^* for the agent can be formally cast as solving

$$\mathcal{Q}: \quad \pi^* \triangleq \arg\max_{\pi} \ \mathbb{E}\left[V_\pi(\mathbf{S}_0)\right] \ . \tag{9}$$

IV. GRNN-based Approach to Solving POMDPs

A. Motivation

An optimal strategy π^* (9) in a POMDP environment can be found by leveraging classical dynamic programming algorithms. Such traditional solutions, however, not only require the knowledge of the transition probability model $\mathbb{P}$ and the reward function dynamics but also pose a significant computational challenge when solving for π^*. Under such unknown model settings, Q-learning serves as an effective *model-free value-based* RL algorithm [27] that can find optimal strategies, although only in an MDP environment. In particular, the algorithm focuses on implicitly *estimating* the MDP state value function (8), from each state $\mathbf{S}_i$, associated with every action $a_i \in \mathcal{A}_i$. These *estimates* are widely referred to as Q-values, denoted by $Q(\mathbf{S}_i, a_i) \in \mathbb{R}$, where a higher $Q(\mathbf{S}_i, a_i)$ indicates that taking an action a_i from a MDP state $\mathbf{S}_i$ yields better aggregate future rewards [27].

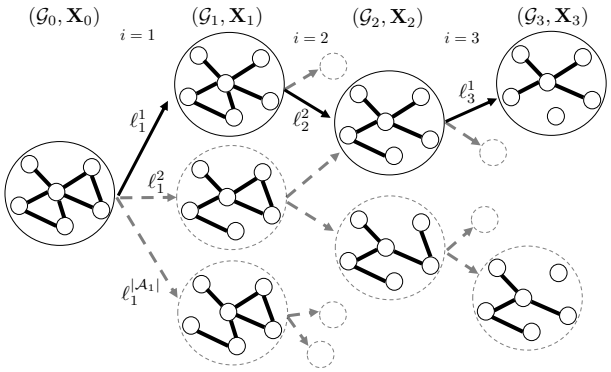

Fig. 1: An agent decision process rendering a FC $\mathcal{V}_s^* = \langle \ell_1^1, \ell_2^2, \ell_3^1 \rangle$.

Note that, however, when solving $\mathcal{Q}$ in a POMDP environment, the ordinary Q-learning algorithm is ineffective as partial observations $\mathbf{O}_i$ are typically not reflective of the underlying POMDP state $\mathbf{S}_i$. As a result, $Q(\mathbf{O}_i, a_i) \neq Q(\mathbf{S}_i, a_i)$. Therefore, designing real-time FC search algorithms critically hinges on finding an *accurate* and *efficient* solution to $\mathcal{Q}$ in (9). To this end, we aim to design an approach that judiciously exploits the underlying structure of each POMDP state $\mathbf{S}_i$. In particular, we develop a graph recurrent Q-network (GRQN) architecture that exploits the following three key elements:

1) Neural Networks for Q-value Prediction: Implementing the ordinary Q-learning algorithm relies on building Q-tables that store Q-values for each POMDP state-action pair. However, storing such a Q-table imposes a significant storage challenge even for moderate system sizes since the number of state-action pairs scales combinatorially with the system size N. Furthermore, such an algorithm fails to incorporate continuous POMDP state spaces $\mathbf{S}_i$. As a remedy, we employ universal function approximators such as NNs to *predict* $Q(\mathbf{S}_i, a_i)$ values as approximate surrogates for each *estimated* POMDP state-action element in the Q-Table [28].

2) Spatial Correlation: The data streams generated by measurement units across the network have strong spatial correlation induced due to the inherent meshed topology of transmission networks. Specifically, the coordinates of $\mathbf{X}_i$, in each stage i, are statistically correlated, and hence, data streams are typically jointly modeled via probabilistic graphical models [29], [30]. Therefore, it is important to judiciously leverage the spatial structure of $\mathbf{X}_i$ when solving $\mathcal{Q}$.

3) Temporal Correlation: Besides the spatial correlation, there exists a strong temporal structure induced due to the observational dependencies across the successive stages of any FC sequence. In particular, the load loss incurred due to a component failure in stage i depends on the set of components that have failed in *all* the preceding $(i-1)$ stages.

To this end, in order to exploit the spatio-temporal correlation among the coordinates of a sequence of system states $\mathbf{X}_i$ to *predict* $Q(\mathbf{S}_i, a_i)$ values accurately, we leverage the output of a time-varying GRNN to incorporate the spatial structure and leverage its recurrency to effectively integrate observational dependencies through time to account for the temporal structure. Subsequently, the GRNN output then acts as an input to a NN that *predicts* Q-values for each POMDP state-action pair, altogether assembling into an GRQN architecture. Next, we discuss the various components of the time-varying GRNN, and discuss designing a learning algorithm to search for FC sequences of interest in Section IV-C.

B. Development of a Time-Varying GRNN *Model*

GRNNs are a family of GNN architectures specialized for processing sequential graph structured data streams [31]. These architectures exploit the *local* connectivity structure of the underlying network topology $\mathcal{G}_i$ to efficiently extract features from each bus by sequentially processing time-varying system states $\mathbf{X}_i$ that evolve on a sequence of graphs $\mathcal{G}_i$. More broadly, these architectures generalize recurrent neural networks [32] to graphs. To lay the context for discussions, we first discuss time-varying graph convolutional neural networks (GCNNs), the components of which serve as an essential building block to motivate time-varying GRNNs.

1) Time-Varying GCNNs*:* Consider the i^{th} stage of a FC sequence $\mathcal{V}_s$ where an agent receives an observation $\mathbf{O}_i \triangleq (\mathcal{G}_i, \mathbf{X}_i)$ on graph $\mathcal{G}_i$ associated with an adjacency matrix $\mathbf{B}_i$. Central to the development of GCNNs is the concept of a *graph-shift* operation that relates an input system state $\mathbf{X}_i \in \mathbb{R}^{N\times F}$ to an output system state $\mathbf{F}_i \in \mathbb{R}^{N\times F}$

$$\mathbf{F}_i \triangleq \mathbf{B}_i \cdot \mathbf{X}_i \ . \tag{10}$$

Clearly, the output system state $\mathbf{F}_i$ is a locally shifted version of the input system state $\mathbf{X}_i$ since each element $[\mathbf{F}_i]_{u,f}$, for any bus $u \in V_i$ and parameter f, is a linear combination of input system states in its 1-hop neighborhood. Local operations, such as (10) capture the 1-hop structural information from $\mathbf{X}_i$ by *estimating* another system state $\mathbf{F}_i$ and are important because of the strong spatial correlation that exists between $[\mathbf{X}_i]_{u,f}$ and its network neighborhood determined by $\mathcal{G}_i$. Alternative transformations such as that employed in [33] quantify the merits of estimating system states that capture the spatial structure of $\mathbf{X}_i$. In order to capture the structural information from a broader K-hop neighborhood instead, (10) can be readily extended by defining a time-varying *graph convolutional filter* function $\mathsf{H} : \mathbb{R}^{N\times F} \to \mathbb{R}^{N\times H}$ that operates on an input system state $\mathbf{X}_i$ to *estimate* an output system state $\mathsf{H}(\mathcal{G}_i, \mathbf{X}_i \ ; \ \mathcal{H})$

$$\mathsf{H}(\mathcal{G}_i, \mathbf{X}_i \ ; \ \mathcal{H}) \triangleq \sum_{k=1}^{K} \left[\mathbf{B}_i^{k-1} \cdot \mathbf{X}_i\right] \cdot \mathbf{H}_k \ , \tag{11}$$

where H denotes the number of output features *estimated* on each bus and $\mathcal{H} \triangleq \{\mathbf{H}_k \in \mathbb{R}^{F\times H} : k \in [K]\}$ denotes the set of filter coefficients parameterized by matrices $\mathbf{H}_k$ *learned* from simulations where each coordinate of $\mathbf{H}_k$ suitably weighs the aggregated system state obtained after k repeated 1-hop graph-shift operations performed on $\mathbf{X}_i$. Note that (11) belongs to a broad family of *graph-time filters* [34] that are polynomials in time-varying adjacency matrices $\mathbf{B}_i$. There exists many types of *graph-time filters* [35]. We employ (11) due to its simplicity. Nevertheless, (11) only captures simple linear dependencies within $\mathbf{X}_i$. To capture non-linear relationships within $\mathbf{X}_i$, time-varying GCNNs *compose* multiple layers of graph-time filters (11) and non-linearities such that the output system state of each GCNN layer is given by

$$\Phi(\mathcal{G}_i, \mathbf{X}_i \ ; \ \mathcal{H}) \triangleq \sigma\left(\ \mathsf{H}(\mathcal{G}_i, \mathbf{X}_i \ ; \ \mathcal{H})\ \right) \ , \tag{12}$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is commonly known as the activation function (applied element-wise) such that $\Phi(\mathcal{G}_i, \mathbf{X}_i \ ; \ \mathcal{H}) \in \mathbb{R}^{N\times H}$.

2) Time-Varying GRNNs*:* GCNNs can only extract spatial features from each system state $\mathbf{X}_i$ independently (12). For this reason, we add recurrency to our GCNN model (12) in order to capture the temporal observational dependencies across the various stages of a FC sequence to construct a GRNN. A time-varying GRNN extracts temporal features from an input sequence $\langle \mathbf{O}_i : i \in [P] \rangle$ by *estimating* a sequence of hidden system states $\langle \mathbf{Z}_i : i \in [P] \rangle$ where each system state $\mathbf{Z}_i \in \mathbb{R}^{N\times H}$ is latent that facilitates in summarizing the entire past observational history, that is both redundant and difficult to store, until stage i. This is done by judiciously parameterizing each hidden system state $\mathbf{Z}_i$ that is a function of the graph-time filter output (11) operated on both the current input system state $\mathbf{X}_i$ and previous hidden system state $\mathbf{Z}_{i-1}$ independently to obtain

$$\mathbf{Z}_i \triangleq \sigma\left(\ \mathsf{H}_1(\mathcal{G}_i, \mathbf{X}_i \ ; \ \mathcal{H}_1) \ + \ \mathsf{H}_2(\mathcal{G}_{i-1}, \mathbf{Z}_{i-1} \ ; \ \mathcal{H}_2)\ \right) \ , \tag{13}$$

where $\mathsf{H}_1 : \mathbb{R}^{N\times F} \to \mathbb{R}^{N\times H}$ and $\mathsf{H}_2 : \mathbb{R}^{N\times H} \to \mathbb{R}^{N\times H}$ are filters (11) each parameterized by a distinct set of filter coefficients $\mathcal{H}_1 = \{\mathbf{H}_k^1 \in \mathbb{R}^{F\times H} \ \forall \ k \in [K]\}$ and $\mathcal{H}_2 = \{\mathbf{H}_k^2 \in \mathbb{R}^{H\times H} \ \forall \ k \in [K]\}$, respectively. Subsequently, similar to (12), to capture the non-linear relationships from each hidden system state $\mathbf{Z}_i$ to facilitate dynamic decision-making on graphs $\mathcal{G}_i$, the *estimated* output system state $\mathbf{Y}_i \in \mathbb{R}^{N\times G}$

$$\mathbf{Y}_i \triangleq \rho\left(\ \mathsf{H}_3(\mathcal{G}_i, \mathbf{Z}_i \ ; \ \mathcal{H}_3)\ \right) \quad \forall i \in [P] \ , \tag{14}$$

where the graph-time filter $\mathsf{H}_3 : \mathbb{R}^{N\times H} \to \mathbb{R}^{N\times G}$ in (11) is parameterized by the filter coefficient set $\mathcal{H}_3$, G denotes the number of *estimated* output features on each bus $u \in V_i$, and ρ is a pointwise non-linearity applied element-wise to output H_3. Note that H, K, G, σ and ρ are hyper-parameters for a GRNN. Additionally, the number of *learnable* parameters $\mathcal{H}_i \ \forall i \in [3]$ is independent of the system size N and the horizon P of the FC sequence due to parameter sharing across the stages of the FC, providing the model with flexibility to *learn* from input sequences $\langle \mathbf{O}_i : i \in [P] \rangle$ of different and long risk assessment horizons without a combinatorial growth in the number of *learnable* parameters, ensuring tractability. Next, we leverage the sequence of GRNN output system states $\langle \mathbf{Y}_i : i \in [P] \rangle$ to learn a strategy $\tilde{\pi} \approx \pi^*$ (9) to efficiently solve (4).

This article has been accepted for publication in IEEE Transactions on Power Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPWRS.2023.3258740

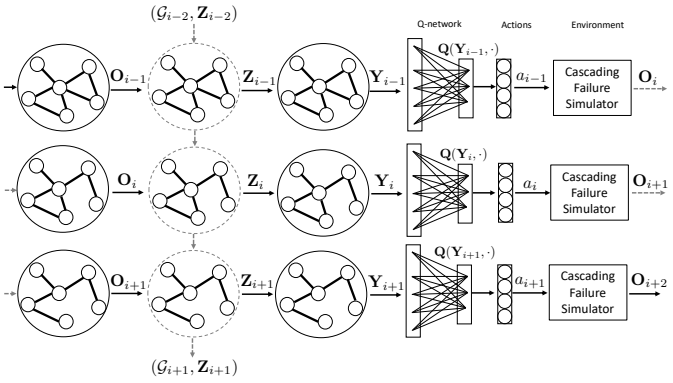

Fig. 2: Graph recurrent Q-network (GRQN) architecture.

C. Finding a Strategy via Graph Recurrent Q-learning

As discussed in Section IV, estimating $Q(\mathbf{S}_i, a_i)$ via traditional RL algorithms is intractable. Hence, we choose to *predict* $Q(\mathbf{S}_i, a_i)$ via NNs. However, since the agent is no longer privy to the true underlying POMDP state $\mathbf{S}_i$, inspired by the approach of [36], we develop a GRQN architecture to *estimate* $\mathbf{Y}_i$ (14) in order to *predict* $Q(\mathbf{Y}_i, a_i)$ as a proxy to approximate $Q(\mathbf{S}_i, a_i)$. Additionally, since for any given initial system state $(\mathcal{G}_0, \mathbf{X}_0)$, it is of interest to find S number of FC sequences (4) with the maximum TLL, it is not enough to *predict* $Q(\mathbf{Y}_i, a_i)$ but rather to guide the search of subsequent FCs with the next highest TLL by leveraging $Q(\mathbf{Y}_i, a_i)$. Therefore, we design a graph recurrent Q-learning algorithm to discover the S number of FC sequences of interest $\{\mathcal{V}_1^*, \ldots, \mathcal{V}_S^*\}$ one after the other while concurrently training the GRQN. Next, we discuss the GRQN architecture and the designed training algorithm.

1) GRQN *Architecture:* The architecture consists of a time-varying GRNN, parameterized by $\boldsymbol{\theta}_{\mathsf{GRNN}} \triangleq \{\mathcal{H}_i : i \in [3]\}$, that sequentially processes observations at each stage of the FC and a fully-connected NN, parameterized by $\boldsymbol{\theta}_{\mathsf{NN}}$, to *predict* $Q(\mathbf{Y}_i, a_i)$ for each action $a_i \in \mathcal{A}_i$. Accordingly, we parameterize the GRQN by $\boldsymbol{\theta} \triangleq \{\boldsymbol{\theta}_{\mathsf{GRNN}}, \boldsymbol{\theta}_{\mathsf{NN}}\}$. For any FC sequence $\mathcal{V}_s$, an input data stream of observations $\mathbf{O}_i \in \mathbb{R}^{N \times F}$ obtained at each stage i of the FC acts as an input to the GRNN using which an output system state $\mathbf{Y}_i \in \mathbb{R}^{N \times G}$ is estimated. Subsequently, a fully connected NN of input and output dimension $N \times G$ and $|\mathcal{U}|$, respectively, is leveraged to output a $\mathbf{Q}(\mathbf{Y}_i, \cdot|\boldsymbol{\theta}) \in \mathbb{R}^{|\mathcal{U}|}$ vector consisting of Q-values for each action. Overall, at each stage i, a GRQN takes $\mathbf{O}_i$ and $\mathbf{Z}_{i-1}$ as it's baseline inputs, concisely denoted by $\mathsf{GRQN}(\mathbf{O}_i, \mathbf{Z}_{i-1}|\boldsymbol{\theta})$, and outputs a vector $\mathbf{Q}(\mathbf{Y}_i, \cdot|\boldsymbol{\theta})$ and the next hidden system state $\mathbf{Z}_i$ (13), crucial to carry forward to guide the search of FCs during the training of the GRQN. Note that we initialize $\mathbf{Z}_0$ by $\mathbf{0} \in \mathbb{R}^{N \times H}$. Fig. 2 illustrates the end-to-end GRQN architecture.

2) Sequential Experience Buffer: In order to deal with the issue of *catastrophic forgetting* [28], we employ a *sequential* experience buffer that stores FC sequences discovered during training of the GRQN from which batches of random sequences are sampled to facilitate the learning of parameters $\boldsymbol{\theta}$. While there exists various ways to implement such a buffer, we employ an ordered list that stores all the visited transition tuples as a sequence $\langle(\mathbf{O}_i, a_i, r_i, \mathbf{O}_{i+1}, \mathsf{end}(\mathbf{O}_{i+1})) : i \in [P]\rangle$ where we have defined $\mathsf{end}(\mathbf{O}_{i+1})$ as a boolean value, if true, indicating that the observation $\mathbf{O}_{i+1}$ is associated with a last stage of the risk assessment horizon P.

3) Training the GRQN*:* For stability in the training of the GRQN, we employ the standard trick [28] of splitting the task of predicting and evaluating Q-values via two separate GRQNs, a target network $\mathsf{GRQN}(\cdot, \cdot|\boldsymbol{\theta}^-)$ and a behavior network $\mathsf{GRQN}(\cdot, \cdot|\boldsymbol{\theta})$ each parameterized by a distinct set of parameters $\boldsymbol{\theta}^-$ and $\boldsymbol{\theta}$, respectively. For every training iteration n of the graph recurrent Q-learning algorithm, the agent samples B random batches of FC sequences, each of type $\langle(\mathbf{O}_j, a_j, r_j, \mathbf{O}_{j+1}, \mathsf{end}(\mathbf{O}_{j+1})) : j \in [P]\rangle$, from the sequential experience buffer on which the target GRQN is *unrolled* to estimate $\mathbf{Y}_j$ (14) using which B batches of $\mathbf{Q}(\mathbf{Y}_j, \cdot|\boldsymbol{\theta}^-)$ are predicted with respect to the target network. Subsequently, the agent computes a *look-ahead* target output for each batch where each target $t_j \; \forall j \in [P]$ is given by

$$t_j = r_j + \gamma \cdot (1 - \mathsf{end}(\mathbf{O}_{j+1})) \cdot \max_a Q(\mathbf{Y}_{j+1}, a|\boldsymbol{\theta}^-) \; . \quad (15)$$

Accordingly, the parameters of the behavior network is updated via gradient descent with respect to a quadratic loss

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - \alpha \cdot \nabla_{\boldsymbol{\theta}} (t_j - Q(\mathbf{Y}_j, a_j|\boldsymbol{\theta}))^2 \; , \quad (16)$$

where α denotes the learning rate and n denotes the current training iteration. The update (16) is preformed κ times for every action taken by the agent in any stage $i \in [P]$ of the FC and additionally, serves as a means to control the *computational complexity* of our learning algorithm. In this paper, we employ the Adam optimizer [37] to perform the gradient update (16) and update the target network parameters $\boldsymbol{\theta}^- = \boldsymbol{\theta}$ at the end of every FC sequence discovered.

4) Graph Recurrent Hidden System State Updates: As the agent gains more experience and continues to store visited transition sequences of tuples in the experience buffer, the hidden system state $\mathbf{Z}_i$ of the GRNN may either zeroed or carried forward after every newly discovered FC. Our experiments suggest that sequential updates where the hidden system state $\mathbf{Z}_i$ (13) is carried forward from the previous stages throughout the parameter update (16) leads to learning of better FC search strategies. Hence, we choose to carry forward the previously learned hidden system state during the training of our GRQN.

5) Outline of Algorithm 1: Algorithm 1 outlines the graph recurrent Q-learning algorithm used to *learn* the parameters $\boldsymbol{\theta}$ of the behaviour $\mathsf{GRQN}(\cdot, \cdot|\boldsymbol{\theta})$ to facilitate the discovery of the S number of FC sequences of interest. During the initial few iterations, since the sequential experience buffer is empty, we allow the agent to *explore* and fill the buffer with FC sequences for Explore number of iterations *offline* (more details in Section V-A1). Subsequently, the real-time algorithm is initiated. Initially, the agent lacks any information about the cascading failure dynamics. Hence, it relies on the *prior knowledge* to construct $\mathcal{U}_{j,i}$, for any fault chain j in each stage $i \in [P]$, as the agent is unable to evaluate the LL associated with removing an arbitrary component $\ell_i \in \mathcal{U}_j \backslash \{\cup_{k=1}^{i-1} \mathcal{U}_{j,k}\}$ in any stage of the FC $\mathcal{V}_j$. Gradually, Algorithm 1 learns to construct the sets $\mathcal{U}_{j,i}$ by leveraging the learned latent

Algorithm 1 Graph Recurrent Q-learning (GRQN)

procedure GRAPH RECURRENT Q-LEARNING
  Initialize behaviour network with random $\boldsymbol{\theta}$ $\mathsf{GRQN}(\cdot,\cdot|\boldsymbol{\theta})$
  Initialize target network with weights $\boldsymbol{\theta}^- = \boldsymbol{\theta}$ $\mathsf{GRQN}(\cdot,\cdot|\boldsymbol{\theta}^-)$
  Initialize $\mathsf{Buffer} \leftarrow \langle\rangle$
  Initialize $\mathbf{Z}_0 \leftarrow \mathbf{0} \in \mathbb{R}^{N\times H}$
  for Episode $s = 1,\ldots,S$ **do**
    $\mathsf{Episode} \leftarrow \langle\rangle$
    $\mathcal{V}_s \leftarrow \langle\rangle$
    Reset power-flow according to initial state $\mathbf{S}_0, \mathbf{O}_0$
    for Stage $i = 1,\ldots,P$ **do**
      $\_, \mathbf{Z}_i \leftarrow \mathsf{GRQN}(\mathbf{O}_i, \mathbf{Z}_{i-1}|\boldsymbol{\theta})$
      $a_i \leftarrow \begin{cases}\text{explore} & \text{if rand}(0,1) \le \epsilon \\ \text{exploit} & \text{otherwise}\end{cases}$
      Take action a_i and determine $\mathcal{U}_i$
      $\mathcal{V}_s \leftarrow \mathcal{V}_s \cup \mathcal{U}_i$
      Update power-flow and obtain $\mathbf{O}_{i+1}$
      Calculate load loss r_i from (7)
      $\mathsf{Episode} \leftarrow \mathsf{Episode} \cup (\mathbf{O}_i, a_i, r_i, \mathbf{O}_{i+1}, \mathsf{end}(\mathbf{O}_{i+1}))$
      if $s \ge \mathsf{Explore}$ **then**
        $\mathsf{count}(\mathbf{S}_i, a_i) \leftarrow \mathsf{count}(\mathbf{S}_i, a_i) + 1$
        for training $n = 1,\ldots,\kappa$ **do**
          Sample B FC sequences from Buffer
          $t_j, \_ \leftarrow \mathsf{GRQN}(\mathbf{O}_{j+1}, \mathbf{0}|\boldsymbol{\theta}^-)$ from (15)
          $Q(\mathbf{Y}_j, a_j|\boldsymbol{\theta}), \_ \leftarrow \mathsf{GRQN}(\mathbf{O}_j, \mathbf{0}|\boldsymbol{\theta})$
          Calculate $\nabla_{\boldsymbol{\theta}}(t_j - Q(\mathbf{Y}_j, a_j|\boldsymbol{\theta}))^2$
          Update $\boldsymbol{\theta}$ as in (16)
          Update ϵ as in (19)
        end for
      end if
      Update availability of actions backwards
    end for
    $\mathsf{Buffer} \leftarrow \mathsf{Buffer} \cup \mathsf{Episode}$
    $\mathbf{Z}_0 \leftarrow \mathbf{Z}_P$
    if $\mathsf{TLL}(\mathcal{V}_s) \ge M$ **then**
      Store risky FC
    end if
    $\boldsymbol{\theta}^- = \boldsymbol{\theta}$
  end for
end procedure
Find the accumulated risk due to all the S FCs $\{\mathcal{V}_1,\ldots,\mathcal{V}_S\}$.

graphical feature representations. These representations are obtained by optimizing the *look-ahead* target function (15) characterized by the behavior network $\mathsf{GRQN}(\cdot,\cdot|\boldsymbol{\theta})$ since the parameters $\boldsymbol{\theta}$ of this network determine the Q-values influencing the actions $a_i \in \mathcal{A}_i$ taken by the agent. Therefore, when choosing actions $a_i \in \mathcal{A}_i$, the agent should make a trade-off between *exploration* and *exploitation* throughout the training of the GRQNs. Next, we discuss an *exploration-exploitation* search strategy that the agent employs to make the real-time search of FCs of interest more efficient.

D. Fault Chain Search Strategy

We employ the standard ϵ-greedy search strategy with an adaptive exploration schedule. Initially, the agent is compelled to take actions based on prior knowledge to find FCs with maximum expected TLL. Typically, since an outage of a component carrying higher power makes the remaining components vulnerable to overloading, a reasonable exploration strategy of the agent would be to remove components carrying maximum power-flow. Therefore, we follow a power-flow weighted (PFW) exploration strategy (also adopted in [12]) such that, in any stage i of the FC, the agent chooses the j^{th} available component $\ell_i^j \in \mathcal{A}_i$ according to the rule

$$a_i = \underset{\ell_i^j}{\arg\max}\ \frac{\mathsf{PF}(\ell_i^j)/\sqrt{\mathsf{count}(\mathbf{S}_i, \ell_i^j)+1}}{\sum_{k=1}^{|\mathcal{A}_i|}\ \mathsf{PF}(\ell_i^k)/\sqrt{\mathsf{count}(\mathbf{S}_i, \ell_i^k)+1}}\ , \quad (17)$$

with probability ϵ where we denote $\mathsf{PF}(\ell_i^j)$ as the *absolute* value of the power flowing through component ℓ_i^j and denote $\mathsf{count}(\mathbf{S}_i, \ell_i^j)$ as the number of times the component ℓ_i^j was chosen when the agent was in POMDP state $\mathbf{S}_i$ in the past. On the other hand, as the agent gains more experience, the agent should choose actions based on the Q-values learned via the behaviour network $\mathsf{GRQN}(\cdot,\cdot|\boldsymbol{\theta})$. Accordingly, a strategy based on Q-values learned by the agent is designed. Specifically, actions are chosen proportional to the Q-values normalized by each POMDP state-action visit count to *avoid repetitions* of FC sequences discovered earlier. Accordingly, the agent chooses action a_i

$$a_i = \underset{\ell_i^j}{\arg\max}\ \frac{Q(\mathbf{Y}_i, \ell_i^j|\boldsymbol{\theta})}{\sqrt{\mathsf{count}(\mathbf{S}_i, \ell_i^j)+1}}\ , \quad (18)$$

with probability $1-\epsilon$.

In order to balance the exploration-exploitation trade-off between (17) and (18) during the training of the GRQN, it is important to dynamically alter the probability ϵ so that, with more experience, the agent chooses actions based on (18). Appropriately, we follow the exploration schedule given by

$$\epsilon = \max\left(\frac{\sum_{j=1}^{|\mathcal{A}_1|}\ \mathsf{PF}(\ell_1^j)/\sqrt{\mathsf{count}(\mathbf{S}_0, \ell_1^j)+1}}{\sum_{k=1}^{|\mathcal{A}_1|}\ \mathsf{PF}(\ell_1^k)},\ \epsilon_0\right)\ , \quad (19)$$

where ϵ_0 ensures a minimum level of exploration.

V. Case Studies and Discussion

In this section, extensive simulations are performed to validate the proposed graphical framework to find FCs that incur large TLLs. While we employ the DC power-flow model to simulate FCs, other models such as the AC power-flow can be easily integrated within our framework. In order to generate FCs, we employ PYPOWER a port of MATPOWER [38] to Python and leverage PyTorch [39] to train the behavior and target GRQNs.

A. Algorithm Initialization and Evaluation Criteria

1) Offline Sequential Buffer Initialization: To initiate the parameter update of the behavior GRQN via gradient descent (16), there must exist at least B FC sequences in the experience buffer. However, unlike in the case of (17), the buffer can be populated *offline* for any loading condition. Therefore, the agent can afford to take actions greedily with respect to components conducting maximum power and, accordingly, backtrack to update the availability of actions to avoid repeating FCs discovered previously. Hence, prior to the start of our real-time FC search Algorithm 1, we let the agent explore *offline* for Explore iterations where the agent, in any stage i, chooses actions according to the rule

$$a_i = \underset{\ell_i^j}{\arg\max}\ \frac{\mathsf{PF}(\ell_i^j)}{\sum_{k=1}^{|\mathcal{A}_i|}\ \mathsf{PF}(\ell_i^k)}\ , \quad (20)$$

with probability 1 to fill the sequential experience buffer. Note that this needs to be done only once offline for any loading condition. This is important since the quality of the sequences in the buffer greatly affects the efficiency of the search.

	Evaluation Metrics		Accuracy Metrics	
Algorithm	**Range for Accumulative** TLL $\sum_{i=1}^{S}\mathsf{TLL}(\mathcal{V}_i)$ **(in** MWs**)**	**Range for the No. of Risky** FCs $\sum_{i=1}^{S}\mathbb{1}(\mathsf{TLL}(\mathcal{V}_i)\geq M)$	**Range for** Regret(S) (in MWs)	**Range for** Precision(S)
Algorithm 1 ($\kappa=3$)	$110.42\times10^3\ \pm\ 37\%$	$199\ \pm\ 71$	$765.33\times10^3\ \pm\ 5.3\%$	$0.169\ \pm\ 35\%$
Algorithm 1 ($\kappa=2$)	$96.18\times10^3\ \pm\ 38\%$	$173\ \pm\ 59$	$779.57\times10^3\ \pm\ 4.7\%$	$0.144\ \pm\ 34\%$
Algorithm 1 ($\kappa=1$)	$86.43\times10^3\ \pm\ 35\%$	$161\ \pm\ 48$	$789.32\times10^3\ \pm\ 3.87\%$	$0.134\ \pm\ 29\%$
PFW + RL + TE [12]	$60.63\times10^3\ \pm\ 3.4\%$	$109\ \pm\ 5$	$815.12\times10^3\ \pm\ 0.26\%$	$0.0909\ \pm\ 4.8\%$
PFW + RL [12]	$57.18\times10^3\ \pm\ 3.1\%$	$101\ \pm\ 9$	$818.57\times10^3\ \pm\ 0.22\%$	$0.084\ \pm\ 3.7\%$

TABLE I: Performance comparison for the IEEE-39 New England test system.

2) Accuracy Metrics and Evaluation Criteria: We aim to identify FCs with the largest TLLs. Hence, one natural evaluation metric is the accumulated TLL due to the set of FCs $\{\mathcal{V}_1,\ldots,\mathcal{V}_S\}$. Besides that, we also consider the total number of *risky* FCs discovered as a function of FC sequence iterations $s\in[S]$, as considered in [12], to further perform comparisons. A FC sequence $\mathcal{V}_s$ is deemed *risky* if it's associated TLL exceeds a pre-specified level M, i.e., $\mathsf{TLL}(\mathcal{V}_s)\geq M$. These two metrics are useful for evaluating the relative performance of Algorithm 1 compared to alternative approaches. For evaluating the accuracy of Algorithm 1, we adopt the following two metrics that quantify the accuracy in load loss and risky fault chain discovery rates.

a) Load Loss Accuracy: We define a regret term that quantifies the gap between the accumulated TLL discovered by Algorithm 1 and the *optimal* accumulated TLLs of the ground truth FCs with the maximum TLL given by the set $\{\mathcal{V}_1^*,\ldots,\mathcal{V}_S^*\}$. Specifically, for $s\in[S]$ we define

$$\mathsf{Regret}(s)\ \triangleq\ \sum_{i=1}^{S}\mathsf{TLL}(\mathcal{V}_i^*)-\sum_{i=1}^{s}\mathsf{TLL}(\mathcal{V}_i)\ . \tag{21}$$

A lower regret value indicates a higher accuracy.

b) Risky FC *Discovery Rate:* We define a precision metric that quantifies the fraction of FCs that are deemed risky in the set of discovered FC. Specifically, for $s\in[S]$ we define

$$\mathsf{Precision}(s)\ \triangleq\ \frac{1}{s}\sum_{i=1}^{s}\mathbb{1}(\mathsf{TLL}(\mathcal{V}_i)\geq M)\ , \tag{22}$$

where $\mathbb{1}(\cdot)$ denotes the indicator function. A higher precision value indicates higher accuracy.

B. IEEE-39 New England Test System

This test system comprises of $N=39$ buses and $|\mathcal{U}|=46$ components, including 12 transformers and 34 lines. We consider a loading condition of $0.55\times$base_load, where base_load denotes the standard load data for the New England test case in PYPOWER *after* generation-load balance to quantify the performance of our approach. These loading conditions were chosen since it is relatively difficult to discover FCs with large TLLs in a lightly loaded power system as there are fewer such FCs in comparison to the space of all FC sequences $|\mathcal{F}|$.

1) Parameters and Hyper-parameters: The hyper-parameters of the GRQNs are chosen by performing hyper-parameter tuning. Accordingly, we choose $H=G=12$ hidden and output number of features when computing both the hidden system state (13) and the output system state (14). We use $K=3$ graph-shift operations for the graph-filter (11) that is used to compute (13) and (14). We use both ρ and σ as the hyperbolic tangent non-linearity $\sigma=\rho=\tanh$ and the ReLU non-linearity for the fully-connected NN that approximates the Q-values. For other parameters, we choose $F=1$ since we employ voltage phase angles as the only input system state parameter f, choose $\gamma=0.99$ since large LLs mostly occur in the last few stages of the FC sequence, an $\epsilon_0=0.01$ to ensure a minimum level of exploration during the FC search process, a batch size $B=32$, Explore $=250$, a risk assessment horizon $P=3$, FC sequence iteration $S=1200$ (*excluding* the initial Explore iterations), learning rate $\alpha=0.005$, and $\kappa\in[3]$ that controls the frequency of the learning update (16) and also governs the computational complexity of the graph recurrent Q-learning algorithm.

We consider M equal to 5% of total load (where the total load is $0.55\times$ base_load). To quantify the regret (21), we need to compute the TLL associated with the S most critical FC sequences (ground truth) $\mathcal{V}_s^*$, $\forall s\in[S]$. This is carried out by generating *all possible* FC sequences (i.e., set $\mathcal{F}$) with a target horizon of $P=3$ for the considered total load of $0.55\times$ base_load. By leveraging the pre-computed set $\mathcal{F}$, we observe a total of 3738 risky FCs for the loading condition $0.55\times$ base_load using our developed FC simulator.

2) Accuracy and Efficiency - Performance Results: Prior to the start of Algorithm 1, we let the agent fill the experience buffer for Explore iterations following the strategy (20) and subsequently, initiate Algorithm 1. Table I illustrates the results obtained. The first column specifies the algorithm employed for evaluation. The second and third columns show the mean and standard deviation of the evaluation metrics and the forth and fifth columns show the mean and standard deviation of the accuracy metrics defined in (21) and (22) for $S=1200$. It is observed that Algorithm 1 with a greater κ discovers FC sequences that incur larger accumulated TLL and also discovers more number of risky FCs, on average. This indicates that the accuracy metrics improve as κ increases. This is expected since the weights of the behaviour GRQN are updated more frequently resulting in a more accurate prediction of the Q-values associated with each POMDP state $\mathbf{S}_i$. For instance, the average regret of Algorithm 1 for $\kappa=3$ is 765.33×10^3 MWs, which is 3.04% lower than the average regret when $\kappa=1$. Similarly, the average precision for $\kappa=3$ is 0.169, which is 26% higher than the average precision when $\kappa=1$. To further assess how the accuracy metrics scale with $s\in[S]$, figures 3 and 4 illustrate the accuracy versus $s\in[S]$. The observations are consistent with Table I, where it is observed that a higher κ

This article has been accepted for publication in IEEE Transactions on Power Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPWRS.2023.3258740

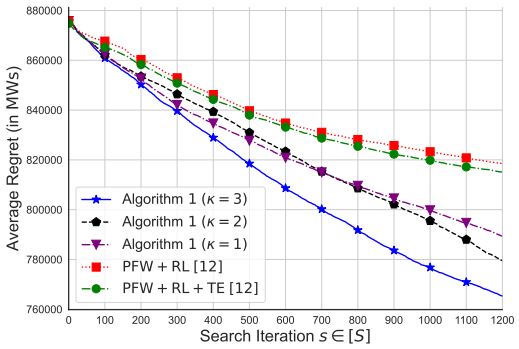

Fig. 3: Regret(s) versus s for the IEEE-39 bus system.

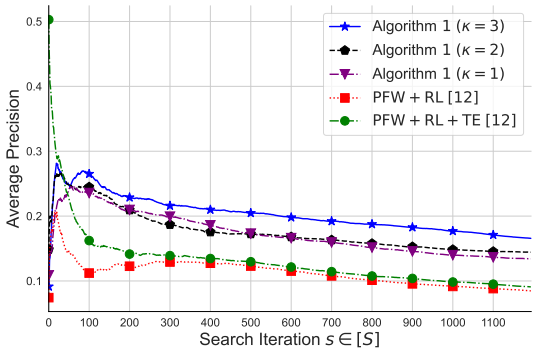

Fig. 4: Precision(s) versus s for the IEEE-39 bus system.

results in a lower average regret and greater average precision. It is noteworthy that the advantage of the setting $\kappa = 3$ is viable at the expense of incurring a higher computational cost. This is due to more frequent weight updates rendering an inevitable accuracy-complexity trade-off.

3) Comparison with Baselines: To further illustrate the merits of the proposed graphical framework, we compare it with two state-of-the-art baseline approaches proposed in [12]. We label the first approach in [12] based on the ordinary Q-learning algorithm without prior knowledge as PFW + RL and label their best performing approach based on transition and extension of prior knowledge from other power system snapshots by PFW + RL + TE. To ensure a fair comparison, we employ the *same* exploration schedule for ϵ discussed in Section IV-D with the same parameters and the same discount factor γ for all the approaches. Note that, in the PFW + RL + TE approach, we first run their proposed Q-learning based approach offline, for a loading condition of $0.6 \times$ base_load (bringing in the prior knowledge). This is run for $S = 5000$ iterations to ensure the convergence of their Q-learning algorithm. Subsequently, we store its extensive Q-table to run its PFW + RL + TE approach in real-time for the considered loading condition of $0.55 \times$ base_load, signifying a transition from the power system snapshot loaded at $0.6 \times$ base_load and an extension to the current power system snapshot loaded at $0.55 \times$ base_load. Note that, when performing comparisons, we set the parameters and hyper-parameters associated with Algorithm 1 the same as that described in Section V-B1.

a) Comparison under Unbounded Computational Budget: In parallel to Algorithm 1, we simultaneously run the Q-learning update discussed in [12] for both the PFW + RL and PFW + RL + TE approaches to perform comparisons. Table I compares the accuracy metrics for $S = 1200$, showing that Algorithm 1 consistently outperforms both the other baseline approaches by a wide margin. For instance, Algorithm 1 with $\kappa = 3$ renders an average regret that is 6.2% smaller than the regret associated with the best preforming baseline PFW + RL + TE approach. Furthermore, we have two more key observations. First, Algorithm 1 with $\kappa = 3$ finds FC sequences whose accumulated TLL is almost double than that of the two baseline approaches.

Secondly, even though our approach is designed to optimize the accumulated TLLs (4) that is quantified via regret (21), it also finds a larger number of risky FCs (on average). This is reflected in Table I and Fig. 4. It is noteworthy that the baseline approach PFW + RL + TE *outperforms* Algorithm 1 for the first fifty search iterations as shown in Fig. 4. This is expected since our approach does not assume any prior knowledge of the failure dynamics while the PFW + RL + TE approach brings in prior knowledge via the extensive Q-table computed offline for the power system snapshot loaded at $0.6\times$base_load. However, as S increases, Algorithm 1 learns the failure dynamics more accurately, resulting in improved accuracy metrics than that of baseline approaches. For example, the average precision of Algorithm 1 with $\kappa = 3$ is roughly 86% more than that of PFW + RL + TE and 99% more than PFW + RL.

b) Comparison under Bounded Computational Budget: The previous subsection focused on the performance, sans the computational complexity of performing each FC sequence iteration $s \in [S]$. For real-time implementation, the computational complexity of the FC search should be within the period of a dispatch cycle. Hence, we evaluate all the above approaches considering a given computational budget. Specifically, we consider the same evaluation and accuracy metrics as discussed in Section V-A2 and evaluate them considering a strict run-time of five minutes for the algorithms, averaged over 50 Monte Carlo iterations. Table II illustrates the relative performance of different algorithms under budget. There are three main observations. First, within the 5 minute computational time budget, for $\kappa = 3$, the number of FC sequences discovered is considerably smaller than other algorithms. This is reflected in the second column of Table II. This observation is expected since a large value of κ necessitates a larger computation time per FC search iteration due to the gradient update (16), and hence, affords fewer search iterations. Secondly, when comparing the evaluation metrics, Algorithm 1 with $\kappa = 2$ finds the greatest accumulated TLL and also the largest number of risky FCs, on average. Although both PFW + RL and PFW + RL + TE approaches find the most number of FC sequences, $S = 1608$ and $S = 1611$, respectively, the quality of the FCs found are inferior compare to Algorithm 1 with $\kappa = 2$ since their accumulative TLL and the number of risky FCs are smaller. Third, when comparing the accuracy metrics, the average regret and precision for Algorithm 1 with $\kappa = 3$ outperform other algorithms. Although this approach only finds $S = 575$ FCs on average, it yields the most quality FC sequences since the frequent update of the behavior GRQN

	Evaluation Metrics			Accuracy Metrics	
Algorithm	**Average No. of FC Sequences S Discovered**	**Range for Accumulative TLL $\sum_{i=1}^{S}$ TLL$(\mathcal{V}_i)$ (in MWs)**	**Range for the No. of Risky FCs $\sum_{i=1}^{S} \mathbb{1}(\text{TLL}(\mathcal{V}_i) \geq M)$**	**Range for Regret(S) (in MWs)**	**Range for Precision(S)**
Algorithm 1 ($\kappa = 3$)	575	$63.484 \times 10^3 \pm 36.7\%$	104 ± 33	$457.09 \times 10^3 \pm 5.9\%$	$0.1791 \pm 31\%$
Algorithm 1 ($\kappa = 2$)	700	$79.792 \times 10^3 \pm 39\%$	123 ± 38	$521.47 \times 10^3 \pm 7.5\%$	$0.1752 \pm 31.5\%$
Algorithm 1 ($\kappa = 1$)	937	$70.848 \times 10^3 \pm 33.3\%$	117 ± 34	$673.56 \times 10^3 \pm 4.76\%$	$0.1249 \pm 29.1\%$
PFW + RL + TE [12]	1608	$68.74 \times 10^3 \pm 3.4\%$	112 ± 4	$965.41 \times 10^3 \pm 1.61\%$	$0.0698 \pm 2.91\%$
PFW + RL [12]	1611	$65.35 \times 10^3 \pm 3.9\%$	105 ± 5	$969.78 \times 10^3 \pm 2.27\%$	$0.0653 \pm 5.07\%$

TABLE II: Performance comparison for a computational time of 5 minutes for the IEEE-39 New England test system.

weights results in a more accurate Q-value prediction that optimize (4) better *per* search iteration $s \in [S]$. Compared to non-graphical algorithm counterparts, although it can find fewer FC sequences S, it has been able to identify the more relevant sequences of interest.

It is noteworthy that we have evaluated all the algorithms on a standard computer with no Graphics Processing Units (GPUs). By leveraging GPUs, our approach can accelerate the risky FC search process even further as GPUs are designed to facilitate the operations involving matrix and vectors for efficient training of the GRQNs, as opposed to other two baselines approaches that, cannot be accelerated for a given computation time.

C. IEEE-118 Test System

This test system consists of $N = 118$ buses and $|\mathcal{U}| = 179$ components. We consider a loading condition of $0.6 \times$ base_load, where base_load denotes the standard load data for the IEEE-118 test case in PYPOWER after generation-load balance to quantify the performance of our approach.

1) Parameters and Hyper-parameters: We choose $H = G = 48$ hidden and output number of features when computing both the hidden system state (13) and the output system state (14). We use $K = 3$ graph-shift operations for the graph-filter (11), use both ρ and σ as the hyperbolic tangent non-linearity $\sigma = \rho = \tanh$ and the ReLU non-linearity for the fully-connected NN to approximate Q-values. For other parameters, we choose $F = 1$ since we employ voltage phase angles as the only input system state parameter f, choose $\gamma = 0.99$, $\epsilon_0 = 0.01$, a batch size $B = 32$, Explore $= 250$, a risk assessment horizon $P = 3$, FC sequence iteration $S = 1600$ (*excluding* the initial Explore iterations), learning rate $\alpha = 0.0005$, and $\kappa \in [3]$. We set M to 5% of total load (where the total load is $0.6 \times$ base_load).

2) Accuracy and Efficiency – Performance Results: Algorithm 1 is initiated after the experience buffer is filled for Explore search iterations. Table III illustrates the results obtained. Similar to 39-bus system, we observe that Algorithm 1 with a greater κ discovers FC sequences with larger accumulated TLLs and discovers more number of risky FCs leading to superior accuracy metrics for larger κ. For instance the average regret of Algorithm 1 with $\kappa = 3$ is 321.90×10^3 MWs and it is 0.4% lower the average regret when $\kappa = 1$. Similarly, the average precision when $\kappa = 3$ is 11.7% higher that of $\kappa = 1$. Figures 5 and 6 illustrate the average accuracy metrics for Algorithm 1 as a function of $s \in [S]$.

3) Comparison with Baselines: We perform comparisons with the two baselines approaches in [12], i.e., the PFW + RL and PFW + RL + TE. For the PFW + RL + TE approach, we first run their proposed Q-learning-based approach offline, for a loading condition of $1.0 \times$ base_load (bringing in the prior knowledge) for $S = 5000$ iterations and store its extensive Q-table to run the PFW + RL + TE approach in real-time for the considered loading condition of $0.6 \times$ base_load, signifying a transition from $1.0 \times$ base_load to an extension to the current system loaded at $0.6 \times$ base_load. We set the parameters and hyper-parameters as described in Section V-C1.

a) Comparison under Unbounded Computational Budget: Table III compares the accuracy metrics for $S = 1600$, showing that Algorithm 1 consistently outperforms both the other baseline approaches. Figure 5 shows how the average regret scales as a function of search iterations $s \in [S]$. Furthermore, even though our approach is designed to optimize the accumulated TLLs (4) that is quantified via regret (21), it also finds a larger number of risky FCs.

b) Comparison under Bounded Computational Budget: We next evaluate all the above approaches considering a runtime computational budget of five minutes, averaged over 25 MC iterations, and Table IV illustrates the relative performance. All the observations corroborate those observed for the 39-bus system.

D. Discussion

1) Performance Comparisons: The ordinary Q-learning algorithm performs only one Q-value update for every action taken by the agent due to the intrinsic design of the algorithm illustrated in [12]. On the other hand, the graph recurrent Q-learning algorithm discussed in section IV-C can perform multiple gradient updates (κ in the inner loop in Algorithm 1) that directly influence the Q-values learned by the agent, via the GRQN. This is possible due to the availability of a sequential experience buffer. This, in turn, facilitates learning more efficient strategies in fewer search trials. We also emphasize the approach in [12] models each permutation of component outages as a unique MDP state, and as a result, it stores the Q-values for a combinatorial number of resulting MDP state-action pairs in an extensive Q-table, rendering it not scalable. However, by judiciously leveraging the graphical structure of each POMDP state and appropriately modeling the dependencies across the various stages of the FC, we have bypassed the storage challenge with fewer modeling assumptions while, at the same time, achieving better performance.

This article has been accepted for publication in IEEE Transactions on Power Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPWRS.2023.3258740

	Evaluation Metrics		Accuracy Metrics	
Algorithm	**Range for Accumulative** TLL $\sum_{i=1}^{S}$ TLL$(\mathcal{V}_i)$ **(in MWs)**	**Range for the No. of Risky** FCs $\sum_{i=1}^{S} \mathbb{1}(\text{TLL}(\mathcal{V}_i) \geq M)$	**Range for** Regret(S) (in MWs)	**Range for** Precision(S)
Algorithm 1 ($\kappa = 3$)	$45.73 \times 10^3 \pm 16\%$	302 ± 49	$321.90 \times 10^3 \pm 2.3\%$	$0.19 \pm 17\%$
Algorithm 1 ($\kappa = 2$)	$40.42 \times 10^3 \pm 17\%$	254 ± 39	$327.22 \times 10^3 \pm 2.4\%$	$0.16 \pm 15\%$
Algorithm 1 ($\kappa = 1$)	$44.62 \times 10^3 \pm 12\%$	274 ± 28	$323.02 \times 10^3 \pm 1.7\%$	$0.17 \pm 10\%$
PFW + RL + TE [12]	$34.23 \times 10^3 \pm 2.7\%$	208 ± 6	$333.40 \times 10^3 \pm 0.28\%$	$0.13 \pm 3\%$
PFW + RL [12]	$35.50 \times 10^3 \pm 2.4\%$	174 ± 8	$332.14 \times 10^3 \pm 0.25\%$	$0.11 \pm 5\%$

TABLE III: Performance comparison for the IEEE-118 test system.

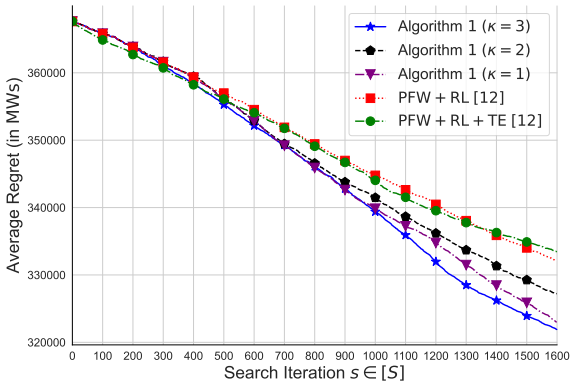

Fig. 5: Regret(s) versus s for the IEEE-118 bus system.

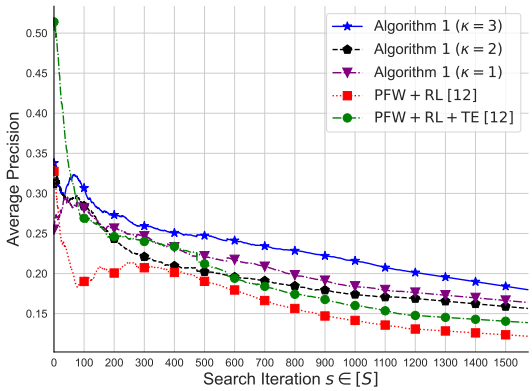

Fig. 6: Precision(s) versus s for the IEEE-118 bus system.

2) Scalability: For large networks, we can further leverage the structure of our GRNN by following an approach based on distributed processing. Specifically, we partition the system into smaller subsystems following the conventional approaches for other monitoring purposes (e.g., estate estimation). The size of subsystems can be decided based on the computational complexity the system operator can afford. The FCs are subsequently identified within each subsystem. The identified risky FCs can be subsequently concatenated to form the FCs for the entire system. This approach, of course, might induce suboptimality in the overall performance of identifying the risky fault chains. Nevertheless, the level of suboptimality induced is expected to be negligible by noting that a fault with a high probability will lead to other faults in its locality.

3) Model Adaptation: The computational complexity of learning the failure dynamics can vary across different cascading failure models. The agent's role is to *learn* the underlying failure dynamics via repeated interactions with the cascading failure simulator. Under different models (e.g., DC power flow model, AC power flow model, transient stability model), the complexity of the learning environment changes. As expected, the transient stability-related models are more challenging to learn than DC power flow-based models under the same number of search iterations S. When S is small, the difference in accuracies can be considerable, and the simpler models (e.g., DC power flow-based) will exhibit better performance. Nevertheless, the performance gap diminishes as the S increases, and the complex models also get a chance to be learned accurately. When prediction accuracies are compared over an arbitrary number of search iterations S (each model can have different search iterations S), then we expect all the models to render similar performance since the latent feature representation of the GRNN will be able to better learn the underlying failure dynamics.

4) Robustness to Different Cascade Triggers: This paper focuses on triggering mechanisms that fall within the general framework of topological changes, i.e., component failures and their subsequent failures. However, other types of triggering mechanisms such as inappropriate power system control decisions and hidden failures in protection systems, to name a couple, also influence the final load loss. As a result, the type of triggering mechanism directly influences the complexity of the learning problem and the agent decision process. One commonality, however, across the different types of triggering mechanisms is that the underlying topology of the network is bound to change as the cascading failure evolves. Therefore, our framework (which explicitly takes into account the topological changes) can be readily customized to accommodate other triggers. For instance, in cases where power system control decisions trigger the initial failures, the actions space $\mathcal{A}_i$ can be modeled as continuous, resulting in a more complex agent learning problem. In such cases, an obvious modification would be to augment the input system state $\mathbf{X}_i$ to include more nodal features (e.g., net power injection and voltage magnitudes). This results in a greater amount of information propagated across the hidden layers $\mathbf{Z}_i$. Subsequently, this results in a more complex objective that is a function of the output $\mathbf{Y}_i$ of the time-varying GRNN architecture, and the action space $\mathcal{A}_i$. Optimizing this objective can potentially incur failure paths resulting in maximum load shed. This way, the proposed framework can be robust to the initial fault event type by appropriate re-formulation.

	Evaluation Metrics			Accuracy Metrics	
Algorithm	**Average No. of FC Sequences S Discovered**	**Range for Accumulative TLL $\sum_{i=1}^{S} \mathsf{TLL}(\mathcal{V}_i)$ (in MWs)**	**Range for the No. of Risky FCs $\sum_{i=1}^{S} \mathbb{1}(\mathsf{TLL}(\mathcal{V}_i) \geq M)$**	**Range for Regret(S) (in MWs)**	**Range for Precision(S)**
Algorithm 1 ($\kappa = 3$)	186	$11.46 \times 10^3 \pm 16.2\%$	92 ± 10	$175.63 \times 10^3 \pm 3.4\%$	$0.201 \pm 15\%$
Algorithm 1 ($\kappa = 2$)	239	$18.32 \times 10^3 \pm 18\%$	101 ± 12	$212.22 \times 10^3 \pm 4.1\%$	$0.17 \pm 14\%$
Algorithm 1 ($\kappa = 1$)	301	$19.86 \times 10^3 \pm 13\%$	115 ± 7	$256.75 \times 10^3 \pm 2.6\%$	$0.18 \pm 13\%$
PFW + RL + TE [12]	527	$18.56 \times 10^3 \pm 3\%$	102 ± 3	$436.22 \times 10^3 \pm 0.98\%$	$0.096 \pm 5\%$
PFW + RL [12]	522	$17.98 \times 10^3 \pm 3\%$	97 ± 4	$439.13 \times 10^3 \pm 1.1\%$	$0.092 \pm 6\%$

TABLE IV: Performance comparison for a computational time of 5 minutes for the IEEE-118 bus test system.

VI. Conclusion

In this paper, we have considered the problem of real-time risky fault chain identification in a limited number of search trials. We have proposed a data-driven graphical framework that can dynamically predict the chains of risky faults a power system faces. First, the search for risky fault chains is modeled as a partially observed Markov decision process. Then a graph recurrent Q-learning algorithm is designed to leverage the grid's topology to discover new risky fault chains efficiently. Test results on the IEEE standard systems demonstrate the effectiveness and efficiency of the proposed approach.

References

[1] "August 14, 2003 blackout: NERC actions to prevent and mitigate the impacts of future cascading blackouts," February 2014, https://www.nerc.com/docs/docs/blackout/NERC_Final_Blackout_Report_07_13_04.pdf.

[2] A. Wang, Y. Luo, G. Tu, and P. Liu, "Vulnerability assessment scheme for power system transmission networks based on the fault chain theory," *IEEE Transactions on Power Systems*, vol. 26, no. 1, pp. 442–450, 2011.

[3] Margaret J. Eppstein and Paul. D. H. Hines, "A "random chemistry" algorithm for identifying collections of multiple contingencies that initiate cascading failure," *IEEE Transactions on Power Systems*, vol. 27, no. 3, pp. 1698–1705, 2012.

[4] Y. Yang, X. Guan, and Q. Zhai, "Fast grid security assessment with N - k contingencies," *IEEE Transactions on Power Systems*, vol. 32, no. 3, pp. 2193–2203, 2017.

[5] T. Ding, C. Li, C. Yan, F. Li, and Z. Bie, "A bilevel optimization model for risk assessment and contingency ranking in transmission system reliability evaluation," *IEEE Transactions on Power Systems*, vol. 32, no. 5, pp. 3803–3813, 2017.

[6] J. Guo, F. Liu, J. Wang, J. Lin, and S. Mei, "Toward efficient cascading outage simulation and probability analysis in power systems," *IEEE Transactions on Power Systems*, vol. 33, no. 3, pp. 2370–2382, 2018.

[7] P. Rezaei, P. D. H. Hines, and M. J. Eppstein, "Estimating cascading failure risk with random chemistry," *IEEE Transactions on Power Systems*, vol. 30, no. 5, pp. 2726–2735, 2015.

[8] P. Henneaux, P.-E. Labeau, J.-C. Maun, and L. Haarla, "A two-level probabilistic risk assessment of cascading outages," *IEEE Transactions on Power Systems*, vol. 31, no. 3, pp. 2393–2403, 2016.

[9] R. Yao, S. Huang, K. Sun, F. Liu, X. Zhang, S. Mei, W. Wei, and L. Ding, "Risk assessment of multi-timescale cascading outages based on Markovian tree search," *IEEE Transactions on Power Systems*, vol. 32, no. 4, pp. 2887–2900, 2017.

[10] R. Yao, K. Sun, F. Liu, and S. Mei, "Management of cascading outage risk based on risk gradient and Markovian tree search," *IEEE Transactions on Power Systems*, vol. 33, no. 4, pp. 4050–4060, 2018.

[11] X. Wei, J. Zhao, T. Huang, and E. Bompard, "A novel cascading faults graph based transmission network vulnerability assessment method," *IEEE Transactions on Power Systems*, vol. 33, no. 3, pp. 2995–3000, 2018.

[12] Z. Zhang, R. Yao, S. Huang, Y. Chen, S. Mei, and K. Sun, "An online search method for representative risky fault chains based on reinforcement learning and knowledge transfer," *IEEE Transactions on Power Systems*, vol. 35, no. 3, pp. 1856–1867, 2020.

[13] L. Liu, H. Wu, L. Li, D. Shen, F. Qian, and J. Liu, "Cascading failure pattern identification in power systems based on sequential pattern mining," *IEEE Transactions on Power Systems*, vol. 36, no. 3, pp. 1856–1866, 2021.

[14] J. Yan, H. He, X. Zhong, and Y. Tang, "Q-learning-based vulnerability analysis of smart grid against sequential topology attacks," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 1, pp. 200–210, 2017.

[15] F. Li and Y. Du, "From AlphaGo to power system AI: What engineers can learn from solving the most complex board game," *IEEE Power and Energy Magazine*, vol. 16, no. 2, pp. 76–84, 2018.

[16] Y. Du, F. Li, J. Li, and T. Zheng, "Achieving 100x acceleration for $N-1$ contingency screening with uncertain scenarios using deep convolutional neural network," *IEEE Transactions on Power Systems*, vol. 34, no. 4, pp. 3303–3305, 2019.

[17] Y. Liu, N. Zhang, D. Wu, A. Botterud, R. Yao, and C. Kang, "Searching for critical power system cascading failures with graph convolutional network," *IEEE Transactions on Control of Network Systems*, vol. 8, no. 3, pp. 1304–1313, 2021.

[18] I. Dobson, B. Carreras, and D. Newman, "A probabilistic loading-dependent model of cascading failure and possible implications for blackouts," in *Proc. Annual Hawaii International Conference on System Sciences*, Big Island, HI, Jan 2003.

[19] ——, "A branching process approximation to cascading load-dependent system failure," in *Proc. Annual Hawaii International Conference on System Sciences*, Big Island, HI, Jan 2004.

[20] J. Qi, K. Sun, and S. Mei, "An interaction model for simulation and mitigation of cascading failures," *IEEE Transactions on Power Systems*, vol. 30, no. 2, pp. 804–819, 2015.

[21] Paul. D. H. Hines, I. Dobson, and P. Rezaei, "Cascading power outages propagate locally in an influence graph that is not the actual grid topology," *IEEE Transactions on Power Systems*, vol. 32, no. 2, pp. 958–967, 2017.

[22] X. Wu, D. Wu, and E. Modiano, "Predicting failure cascades in large scale power systems via the influence model framework," *IEEE Transactions on Power Systems*, vol. 36, no. 5, pp. 4778–4790, 2021.

[23] I. Dobson, B. Carreras, V. Lynch, and D. Newman, "An initial model for complex dynamics in electric power system blackouts," in *Proc. Annual Hawaii International Conference on System Sciences*, Maui, HI, Jan 2001.

[24] S. Mei, F. He, X. Zhang, S. Wu, and G. Wang, "An improved OPA model and blackout risk assessment," *IEEE Transactions on Power Systems*, vol. 24, no. 2, pp. 814–823, 2009.

[25] S. Soltan, D. Mazauric, and G. Zussman, "Analysis of failures in power grids," *IEEE Transactions on Control of Network Systems*, vol. 4, no. 2, pp. 288–300, 2017.

[26] H. Cetinay, S. Soltan, F. A. Kuipers, G. Zussman, and P. V. Mieghem, "Comparing the effects of failures in power grids under the AC and DC power flow models," *IEEE Transactions on Network Science and Engineering*, vol. 5, no. 4, pp. 301–312, 2018.

[27] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[28] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[29] M. He and J. Zhang, "A dependency graph approach for fault detection and localization towards secure smart grid," *IEEE Transactions on Smart Grid*, vol. 2, no. 2, pp. 342–351, 2011.

[30] J. Heydari and A. Tajer, "Quickest localization of anomalies in power grids: A stochastic graphical framework," *IEEE Transactions on Smart Grid*, vol. 9, no. 5, pp. 4679–4688, 2018.

[31] L. Ruiz, F. Gama, and A. Ribeiro, "Gated graph recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 68, pp. 6303–6318, 2020.
[32] D. Mandic and J. Chambers, *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability*. Wiley, 2001.
[33] A. Dwivedi and A. Tajer, "Scalable quickest line outage detection and localization via graph spectral analysis," *IEEE Transactions on Power Systems*, vol. 37, no. 1, pp. 590–602, 2022.
[34] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, "Filtering random graph processes over random time-varying graphs," *IEEE Transactions on Signal Processing*, vol. 65, no. 16, pp. 4406–4421, 2017.
[35] F. Gama, Q. Li, E. Tolstaya, A. Prorok, and A. R. Ribeiro, "Synthesizing decentralized controllers with graph neural networks and imitation learning," *IEEE Transactions on Signal Processing*, 2022.
[36] M. Hausknecht and P. Stone, "Deep recurrent Q-learning for partially observable MDPs," in *Proc. AAAI Fall Symposium Series*, Arlington, VA, July 2015.
[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. International Conference on Learning Representations*, San Diego, CA, May 2015.
[38] R. D. Zimmerman, C. E. Murillo-Snchez, and R. J. Thomas, "MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Transactions on Power Systems*, vol. 26, no. 1, pp. 12–19, 2011.
[39] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Advances in Neural Information Processing Systems*, Vancouver, Canada, Dec 2019.

Anmol Dwivedi received the B. Tech. degree in Electrical Engineering from National Institute of Technology, Tiruchirappalli, India, in 2019. He is currently working towards the Ph.D. degree in Electrical Engineering at Rensselaer Polytechnic Institute, NY, USA.

Ali Tajer (S'05, M'10, SM'15) received the B.Sc. and M.Sc. degrees in Electrical Engineering from Sharif University of Technology and the M.A degree in Statistics and the Ph.D. degree in Electrical Engineering from Columbia University. He is currently an Associate Professor of Electrical, Computer, and Systems Engineering at Rensselaer Polytechnic Institute. His research interests include mathematical statistics, statistical signal processing, and network information theory, with applications in wireless communications and power grids. His recent publications include an edited book entitled Advanced Data Analytics for Power Systems (Cambridge University Press, 2021). He has received an NSF CAREER award in 2016 and AFRL Faculty Fellowship in 2019. He is currently serving as an Associate Editor for the IEEE Transactions on Information Theory and for the IEEE Transactions on Signal Processing. In the past he has served as an an Editor for the IEEE Transactions on Communications, a Guest Editor for the IEEE Signal Processing Magazine, an Editor for the IEEE Transactions on Smart Grid, an Editor for the IET Transactions on Smart Grid, and as the Guest Editor-in-Chief for the IEEE Transactions on Smart Grid Special Issue on Theory of Complex Systems with Applications to Smart Grid Operations.