2020 American Control Conference
Denver, CO, USA, July 1-3, 2020

An Influence Model Approach to Failure Cascade Prediction in Large Scale Power Systems

Xinyu Wu, Dan Wu, and Eytan Modiano

Abstract— **Power system failures are often accompanied by failure cascades which are difficult to model and predict. The ability to predict the failure cascade is important for contingency analysis and corrective control designs to prevent large blackouts. In this paper, we study an influence model framework to predict failure cascades. A hybrid learning scheme is proposed to train the influence model from simulated failure cascade sample pools. The learning scheme firstly applies a Monte Carlo approach to quickly acquire the pairwise influences in the influence model. Then, a convex quadratic programming formulation is implemented to obtain the weight of each pairwise influence. Finally, an adaptive selection of threshold for each link is proposed to tailor the influence model to better fit different initial contingencies. We test our framework on a number of large scale power networks and verify its performance through numerical simulations. The proposed framework is capable of predicting the final state of links within 10% error rate, the link failure frequency within 0.08 absolute error, and the failure cascade size within 7% error rate expectedly. Our numerical results further show that the influence model framework can predict failure cascade two magnitudes faster than the power flow based prediction approach with a limited compromise of accuracy, making it very attractive for online monitoring and screening.**

I. Introduction

Modern power systems frequently experience unpredictable component failures which are caused by falling tree branches, storms, lightening strikes, aged devices, wrong protective actions, etc. These random failures, if not treated properly, can propagate to other system components and eventually incur blackouts. For example, in Northeast America on Aug. 14, 2003, a 345kV line tripped off after touching a tree limb, which, within 1 hour, led to expansion of failures from several links to over 500 generating units in the US and Canada [1]. Recently, Midtown Manhattan in New York sank into darkness due to a disabled transformer [2], and London underwent a two-hour power interruption when a lightning strike hit a transmission link and caused a simultaneous loss of two power plants [3].

Past events have shown that large scale blackouts are usually accompanied by a rapid propagation of failures among many system components. This "rolling snowball" phenomenon is referred to as a *failure cascade* in the power system literature. Modeling and predicting failure cascades is very difficult because a large scale power grid can have hundreds of thousands of components whose dynamic interactions are nonlinear and whose parameters are varying with time.

Many efforts have been devoted to constructing reasonable models of failure cascades at different time scales. A commonly used failure cascade model attempts to solve the static power flow problem step-by-step to determine the sequence of (quasi) static transmission link overflows that occur before the system finally satisfies all the engineering constraints [**?**], [4]. In this failure cascade model, solving the AC power flow problem suffers from a heavy computational burden. It is also difficult to find an AC power flow solution when the network configuration changes drastically [5]. Thus, most works applied the simplified DC power flow model in the failure cascade analysis. Although the DC power flow is a linearized version of the AC model, it still bears some computational efforts when solving it repeatedly in the failure cascade screening process. A few works have been done to derive a closed form solution for the link flows when some links are tripped off [6], [7]. This approach has to take the pseudo-inverse of the system admittance matrix, and has to reformulate the solution when islanding occurs.

In order to further simplify the network model while preserving the basic cascade dynamics, a number of existing works completely ignore the power flow constraints and concentrate on the contagious cascade process in which new failures only happen over adjacent components. It is referred to as the "contagion model" which originates from the percolation model [8]–[10]. This model can capture a large part of the cascade dynamics, but overlooks the non-adjacent correlations among distant failures. Moreover, this abstract model ignores power system dynamics, which renders it rather inaccurate.

To evaluate the nonadjacent failure propagation, Savathiratham *et. al.* proposed the Influence Model (IM) [11]. It is a special graphic model that considers the influence of all the network components on each individual component. It first establishes the pairwise correlation between any two network components; then, summarizes all the pairwise correlations associated with each single component. This aggregated correlation value of each component, which is called the *influence*, is used to determine the component's state in the failure cascade prediction. The influence model is essentially a Markovian model which is easy to construct and implement for large scale applications. Thus, it is a powerful tool for analyzing and predicting failure cascades in power networks [12]–[14].

In order to construct the influence model that can be

This work was supported by DTRA grants HDTRA1-13-1-0021 and HDTRA1-14-1-0058, and NSF grant CNS-1735463.

The authors are with the Laboratory of Information and Decision Systems (LIDS), Massachusetts Institute of Technology, Cambridge, MA 02139, USA. Email: {xinyuwu1,danwumit,modiano}@mit.edu.

used for power system failure cascade predictions, Hines *et. al.* applied the Monte Carlo method to learn the influences from historical data [15]. They focused on generating the appropriate distribution of failure cascade sizes that best match existing records. Zhou *et. al.* further extended the pairwise influences to capture more complex influences [16] for failure size prediction.

In this paper, we propose a hybrid learning framework that can be used to train the influence model from either historical data or synthetic data. Then, we apply our trained influence model to predict the failure cascade sequences for a few large scale power system test cases. Numerical simulations show that our model results in relatively accurate prediction and significant speed advantage as compared to brute force computation of the power dynamics. The major contributions are summarized below.

1) We proposed a hybrid learning framework that can efficiently train the influence model for very large systems. The proposed learning framework integrates Monte Carlo method with quadratic programming, and an adaptive threshold selection scheme to quickly train the model for making good predictions.
2) We applied the influence model to a few large scale power system test cases to predict their failure cascade sequences. It is the first time that such large systems are investigated through the influence model. The prediction performances are thoroughly evaluated at different levels of granularity.
3) We showed that the influence model can predict the cascade sequence two orders of magnitude faster than simulation based on power flow calculation, with small compromise in accuracy. Such efficiency makes the influence model very attractive for online contingency analysis for large scale applications.

II. Influence Model

The IM is a Markovian model whose dynamics are described by the state variable transitions. In power system failure cascade analysis, the state variable of the IM is typically chosen to be the binary operational state of each transmission link[1], which takes on values of either 0 (failed) or 1 (normal) respectively [17]. Given the i-th link, we use $s_i[t]$ to denote its state at time step t. The collection $s[t] := [s_1[t], \dots, s_M[t]]' \in \{0,1\}^{M\times 1}$ of all the M link state variables represents the *network state* at time t, and we define $s := \{s[t]\}_{t=0}^{T}$ as the state (failure cascade) sequence where T is the termination time of the cascade.

According to the IM, the transition of a state variable $s_i[t]$ from the current time t to the next time $t+1$ is described by

$$\tilde{s}_i[t+1] = \sum_{j=1}^{M} d_{ij}\big(\mathbf{A}^{11}_{ji} s_j[t] + \mathbf{A}^{01}_{ji}(1-s_j[t])\big), \quad (1)$$

where

$$\mathbf{A}^{11}_{ji} := P\big(s_i[t+1]=1 \mid s_j[t]=1\big) \quad (2)$$
$$\mathbf{A}^{01}_{ji} := P\big(s_i[t+1]=1 \mid s_j[t]=0\big) \quad (3)$$

are the transition probabilities from the state of link j at t to the state of link i at $t+1$; $\tilde{s}_i[t+1]$ is the estimated value of $s_i[t+1]$[2]. We collect all the $\mathbf{A}^{11}_{ji}$'s into an $M \times M$ matrix $\mathbf{A}^{11}$, and similarly $\mathbf{A}^{01}_{ji}$'s to obtain $\mathbf{A}^{01}$. We refer to each of $\mathbf{A}^{11}$ and $\mathbf{A}^{01}$ as the "pairwise influence matrix".

The weight d_{ij} in (1) represents the proportional affect from the link pair-(i,j). If we further require

$$d_{ij} \geq 0, \quad (4a)$$
$$\textstyle\sum_{j=1}^{M} d_{ij} = 1, \quad (4b)$$

then with given link i, d_{ij} can be regarded as a probability mass function for the pairwise influence of any link j on i. We collect all the d_{ij}'s into an $M \times M$ matrix $\mathbf{D}$, referred to as the "weighted influence matrix".

When $\tilde{s}_i[t+1]$ is obtained, the IM assigns 1 to $s_i[t+1]$ with a probability $\tilde{s}_i[t+1]$, and 0 to $s_i[t+1]$ with a probability $1-\tilde{s}_i[t+1]$. This is a randomized mapping from the unit interval $[0,1]$ to the binary set $\{0,1\}$

$$\hat{s}_i[t+1] = \begin{cases} 1 & \text{w.p.} \quad \tilde{s}_i[t+1] \\ 0 & \text{w.p.} \quad 1-\tilde{s}_i[t+1] \end{cases}, \quad (5)$$

where $\hat{s}_i[t+1]$ is the predicted binary value of $s_i[t+1]$ after the randomized mapping.

This process can yield different prediction results from the same initial contingencies. To make predictions consistent, we reduce the randomized mapping to a deterministic bisection scheme with a threshold ϵ_i for each link i:

$$\hat{s}_i[t+1] = \begin{cases} 1 & \text{if } \tilde{s}_i[t+1] \geq \epsilon_i \\ 0 & \text{if } \tilde{s}_i[t+1] < \epsilon_i \end{cases}, \quad (6)$$

We collect $\hat{s}_i[t]$ in $\hat{s}[t]$ for all the links and refer to it as the prediction of the network state at time t. Then, the time sequence of $\hat{s}[t]$, denoted as $\hat{s}$, is a predicted failure cascade sequence.

An illustrative example of the IM can be found in Fig. 1.

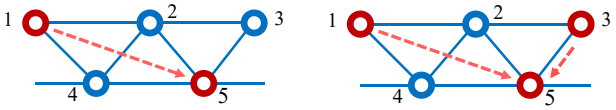


Fig. 1: We consider a 5-component network and take the influence on component 5 as an example. The left subfigure depicts the pairwise influence of component 1 on 5, concerning $\mathbf{A}^{11}_{15}$ and $\mathbf{A}^{01}_{15}$ in (1). The right subfigure reflects that component 5 is mutually influenced by component 1 and 3, where $\mathbf{A}^{11}_{15}$ and $\mathbf{A}^{01}_{15}$ take weight d_{51}, while $\mathbf{A}^{11}_{35}$ and $\mathbf{A}^{01}_{35}$ take weight d_{53} in (1).

To apply the IM in failure cascade analysis, the pairwise influence matrices $\{\mathbf{A}^{11}, \mathbf{A}^{01}\}$, the weighted influence matrix $\mathbf{D}$, and the bisection thresholds ϵ_i's need to be specified,

[1]Other component can also be considered, for example, the generators and transformers.

[2]$\tilde{s}_i[t+1]$ can be interpreted as the expectation of $s_i[t+1]$, provided equation (4).

which serves as the main theme for the following two sections.

III. Generating Sample Cascade Sequences

The IM parameters $\mathbf{A}^{11}$, $\mathbf{A}^{01}$, $\mathbf{D}$ and ϵ_i's are learned from known failure cascade data. The historical failure data of a real power grid is usually inaccessible because the power grid is a critical infrastructure whose data should be kept confidential, and because in a particular power grid large scale failure cascades rarely happen. To get enough data for training the IM, we generate synthetic failure cascades by the power flow based simulation approach [15]. Specifically in this paper, we solve the DC power flow problem using the MATPOWER Toolbox[3][18]. The procedures of generating synthetic failure cascades are summarized below.

1) Given a loading condition, compute the initial link flows to ensure no overflow.
2) Randomly initiate an $M-k$ contingency (k initially failed links) where $k=2$ or 3.
3) Detect if islands (disconnected sub-graphs) appear.
4) If true, re-balance the power in each island by either generation curtailment or load shedding depending on whether the supply exceeds the demand.
5) Recompute the link flows in each island by solving the DC power flow problem.
6) Detect new overflowed links.
7) If true, remove the overflowed links and return to Step 3). Otherwise, terminate.

Repeating this procedure we can build up the training sample pool $\mathcal{S}_{train}$. We denote the k-th cascade sequence in $\mathcal{S}_{train}$ as

$$s^k := \{s^k[t]\}_{t=0}^{T_k}, \quad k=1,2,\ldots,K,$$

where superscript k is the sample index; K is the total number of training samples; T_k denotes the final time that the k-th cascade terminates; $s^k[t]$ is the network state at time t; and s^k records the k-th cascade sequence. We will use this sample pool to train our IM parameters.

IV. Learning Influence Model Parameters

In this section, we explore how to learn the best values of $\mathbf{A}^{11}$, $\mathbf{A}^{01}$, $\mathbf{D}$ and ϵ_i's from $\mathcal{S}_{train}$. Recall (1) that $\tilde{s}_i[t+1]$ estimates $s_i[t+1]$ from the IM parameters $\mathbf{A}^{11}$, $\mathbf{A}^{01}$, $\mathbf{D}$ and the network state $s[t]$ at time t. Hence, the objective is to identify the values of $\mathbf{A}^{11}$, $\mathbf{A}^{01}$, $\mathbf{D}$ such that the estimation $\tilde{s}_i[t+1]$ can best fit the existing sample $s_i[t+1]$ for every link at all the time steps. We achieve this goal by formulating a constrained optimization problem as follows.

[3]AC power flow model will be one of our future research directions.

$$\begin{aligned}\min_{\mathbf{D},\mathbf{A}^{11},\mathbf{A}^{01}} \quad & \frac{1}{K}\sum_{k=1}^{K}\sum_{t=1}^{T_k} f(s^k[t],\tilde{s}^k[t]) \\ s.t. \quad & \tilde{s}_i^k[t+1]=\sum_{j=1}^{M} d_{ij}\big(\mathbf{A}_{ji}^{11}s_j^k[t]+\mathbf{A}_{ji}^{01}(1-s_j^k[t])\big), \forall i,k; \\ & \sum_{j=1}^{M} d_{ij}=1, \forall i; \quad d_{ij},\mathbf{A}_{ji}^{11},\mathbf{A}_{ji}^{01}\geq 0, \forall i,j,\end{aligned} \tag{7}$$

where $f(s^k[t],\tilde{s}^k[t])$ is the cost function that quantifies the distance between $s^k[t]$ and $\tilde{s}^k[t]$; M is the number of links; K is the size of $\mathcal{S}_{train}$; and T_k is the termination time step in the k-th cascade sample.

The problem size of (7) is very large because for each link pair (i,j) there exist two independent pairwise influence values $\mathbf{A}_{ji}^{11}$ and $\mathbf{A}_{ji}^{01}$. For example, in a system with $1,000$ links, we have altogether 3×10^6 variables concerning $\{\mathbf{A}^{11},\mathbf{A}^{01},\mathbf{D}\}$. In order to improve the computational efficiency, we train the pairwise influence $\{\mathbf{A}^{11},\mathbf{A}^{01}\}$ and relative influence $\mathbf{D}$ separately.

A. Learning Pairwise Influence Matrices $\mathbf{A}^{11}$ and $\mathbf{A}^{01}$

We apply the Monte Carlo method to learn the pairwise influence matrices $\mathbf{A}^{11}$ and $\mathbf{A}^{01}$ from the sample pool $\mathcal{S}_{train}$ with size K.

Let τ_i^k be the time step that link i changes to failure state in the k-th cascade sequence s^k. If link i does not fail in s^k, we set τ_i^k to be the termination time of s^k. Then, the value of $\mathbf{A}_{ji}^{11}$ for any link i and j is computed by

$$\mathbf{A}_{ji}^{11} := \frac{\sum_{k=1}^{K} C_{ji}^{11}(s^k,\tau_i^k)}{\sum_{k=1}^{K} C_j^{1}(s^k,\tau_i^k)} \tag{8}$$

where $C_j^1(s^k,\tau_i^k)$ is the number of time steps before τ_i^k in s^k such that link j is normal; $C_{ji}^{11}(s^k,\tau_i^k)$ is the number of time steps before τ_i^k in s^k such that link i is normal, given link j is normal on the adjacent upstream time step.

Similarly, we can estimate $\mathbf{A}_{ji}^{01}$ via

$$\mathbf{A}_{ji}^{01} := \frac{\sum_{k=1}^{K} C_{ji}^{01}(s^k,\tau_i^k)}{\sum_{k=1}^{K} C_j^{0}(s^k,\tau_i^k)} \tag{9}$$

where $C_j^0(s^k,\tau_i^k)$ is the number of time steps before τ_i^k in s^k such that link j is failed; $C_{ji}^{01}(s^k,\tau_i^k)$ is the number of time steps before τ_i^k in s^k such that link i is normal, given link j is failed on the adjacent upstream time step. In this counting process, we should exclude the samples where link i fails initially because it results from external factors independent of flow dynamics.

We take the following toy example to gain a more intuitive view of (8) and (9). We consider two cascade sequences, s^1 and s^2, over 2 links, where

$$s^1=\begin{bmatrix}1&1&1&1&1&0\\1&1&0&0&0&0\end{bmatrix}, \quad s^2=\begin{bmatrix}1&1&1&0&0&0\\1&1&1&1&1&0\end{bmatrix}.$$

We can observe that $\tau_1^1=6$ and $\tau_1^2=4$. For $\mathbf{A}_{21}^{11}$, we have $C_2^1(s^1,\tau_1^1)=2$ and $C_{21}^{11}(s^1,\tau_1^1)=2$ in s^1, and meanwhile

$C_2^1(s^2,\tau_1^2)=3$ and $C_{21}^{11}(s^2,\tau_1^2)=2$ in s^2. According to (8) and (9),

$$\mathbf{A}_{21}^{11}=\frac{C_{21}^{11}(s^1,\tau_1^1)+C_{21}^{11}(s^2,\tau_1^2)}{C_2^1(s^1,\tau_1^1)+C_2^1(s^2,\tau_1^2)}=\frac{2+2}{3+2}=\frac{4}{5},$$

$$\mathbf{A}_{21}^{01}=\frac{C_{21}^{01}(s^1,\tau_1^1)+C_{21}^{01}(s^2,\tau_1^2)}{C_2^0(s^1,\tau_1^1)+C_2^0(s^2,\tau_1^2)}=\frac{2+0}{3+0}=\frac{2}{3}.$$

B. Learning Weighted Influence Matrix $\mathbf{D}$

Once $\{\mathbf{A}^{11},\mathbf{A}^{01}\}$ have been obtained, their values can be substituted into (7) to form a reduced optimization problem whose decision variables are only from $\mathbf{D}$. We choose the objective function $f(\cdot)$ to be the least square error function, which induces a convex quadratic programming problem as follows.

$$\min_{\mathbf{D}} \quad \frac{1}{K}\sum_{k=1}^{K}\sum_{t=1}^{T_k}\sum_{i=1}^{M}\Bigg(s_i^k[t+1] - \sum_{j=1}^{M} d_{ij}(\mathbf{A}_{ji}^{11}s_j^k[t]+\mathbf{A}_{ji}^{01}(1-s_j^k[t]))\Bigg)^2 \tag{11}$$

$$s.t. \quad \sum_{j=1}^{M} d_{ij}=1,\forall i; \qquad d_{ij}\geq 0,\forall i,j.$$

The formulation (11) can be solved efficiently by numerical methods such as the Frank-Wolfe algorithm [19]. The optimal solution to this problem serves as the Bayes least-squares estimator when $\tilde{s}_i[t+1]=\mathbf{E}[s_i[t+1]\mid s[t]]$. Moreover, the weighted influences on each link i are independent of the influences on any other link j, which further supports a problem decomposition into M sub-problems. We solve these small optimization problems in parallel in practice to further reduce the training time.

C. Learning Bisection Threshold ϵ_i

To make a deterministic prediction of a failure cascade, the value of ϵ_i in (6) should be provided. A naive way is to set an universal $\epsilon_i=0.5$ for every link i. However, this undifferentiated threshold value can easily incur wrong predictions. For example in Fig. 2, the third row shows that link 2 fails at the fourth time step. However, the fourth row indicates that the predicted state value at the fourth time step is 0.63 which is greater than 0.5. Thus, by (6) the state of link 2 will be assigned to 1 instead of 0, misidentifying the failure. Therefore, the threshold value ϵ_i should be selected adaptively according to differential initial contingencies. We summarize our adaptive threshold selection scheme for a specific link i as follows.

1) **Identifying a threshold value of link i in a sample sequence s^k.** Three situations can happen. 1) Link i fails initially in the sample sequence s^k. In this situation, there is no way to know the threshold value. 2) Link i fails but not from the beginning of the sample sequence s^k. Then, we recursively compute the estimated state variable $\tilde{s}_i^k[t+1]$ by assigning $\tilde{s}_j^k[t]$ to $s_j^k[t]$ on the right hand side of (1). The critical time step where link i fails is determined. We choose the threshold value for this sequence to be the intermediate value of the estimated state at the critical time step and the estimated state at its upstream adjacent time step. 3) Link i never fails in the sample sequence s^k. In this situation, we recursively compute the estimated state variable $\tilde{s}_i^k[t]$, and choose the threshold value to be $\alpha\times\tilde{s}_i^k[T_k]$ at the final time T_k, where $\alpha\in(0,1)$ can be selected arbitrarily. Fig. 2 shows the identification of ϵ_i among these three situations by a toy example.

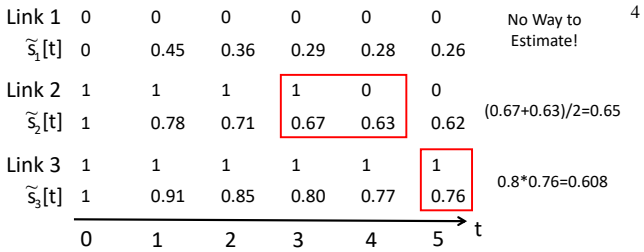


Fig. 2: This is an example of determining thresholds on all 3 link categories in a cascade sample. Each link representative has two rows of records: the first row denotes the real state value at each time step, while the second row denotes the iterative values of $\tilde{s}_i[t]$ for every link i based on (1). For link 2 we set ϵ_2^k as the average of $\tilde{s}_2[4]$ and $\tilde{s}_2[5]$, while for link 3 we set ϵ_3^k to be $0.8\times\tilde{s}_3[T_k]=0.8\times0.76=0.608$, where 0.8 can be replaced by any real value within $(0,1)$.

2) **Forming the threshold pool of link i from all sample sequences.** For each link i, we compute the threshold value ϵ_i^k for every sample sequence s^k and collect them in a set Ω_i.
3) **Selecting an appropriate threshold value of link i for a new contingency.** The basic idea is to select the threshold value ϵ_i from the known threshold set Ω_i such that the associated known contingency is "closest" to the new contingency denoted as $s^{new}[0]$. The closest known contingency to $s^{new}[0]$ is defined by[4]

$$k^*=\arg\min_{k=1,2,\cdots,K}||s^{new}[0]-s^k[0]||_1 \tag{12}$$

where k is the index of the known contingency; $s^k[0]$ is the known contingency; $s^{new}[0]$ is the new contingency; and $||s^{new}[0]-s^k[0]||_1$ is the L_1-norm, denoting the number of links that have different initial states in $s^{new}[0]$ and $s^k[0]$. Then, we select the threshold value $\hat{\epsilon}_i$ to be

$$\hat{\epsilon}_i=\epsilon_i^{k^\star}. \tag{13}$$

Sometimes multiple solutions for $k^\star$ exist for (12). We choose $\hat{\epsilon}_i$ to be the median value among multiple options.

$$\hat{\epsilon}_i=\text{median}\{\epsilon_i^k\}_{k\in K^*} \tag{14}$$

where $K^\star$ is the optimal solution set of (12).

D. Overall Procedure

The overall procedure of learning the IM and using it for failure cascade predictions is presented in Algorithm 1. In the learning modular, finding the weighted influence matrix

[4]Other ways to select the closest contingency is also possible.

$\mathbf{D}$ is the most computationally expensive part because it requires solving a convex quadratic programming problem. After that, the remaining computational efforts are much less demanding. The prediction modular in Algorithm 1 is computationally inexpensive since it only requires multiplication and addition manipulations. Therefore, using the IM to predict failure cascades is potentially much faster than using flow equations to make predictions. This claim will be numerically verified later in the performance evaluations in Section V.

Algorithm 1: Learning Approach and Failure Cascade Prediction based on Influence Model

Input: Training Sample Pool $\mathcal{S}_{train} = \{s^k[t]\}_{t=0,\dots,T_k}^{k=1,\dots,K}$; New Initial State $s^{new}[0]$.

Output: Weighted Influence Matrix $\mathbf{D}$; Pairwise Influence Matrices $\{\mathbf{A}^{11}, \mathbf{A}^{01}\}$; Sequence Prediction $\hat{s}^{new}$.

// Learning Influence Model Parameters

Estimate $\mathbf{A}^{11}$ and $\mathbf{A}^{01}$ based on Monte-Carlo Method;

Learn $\mathbf{D}$ from the quadratic optimization (11);

Build the threshold set Ω_i for each link i;

Find k^* based on equation (12) and form the set K^* containing all k^*s;

Obtain $\hat{\epsilon}_i$ by equation (14) for each link i;

// Failure Cascade Prediction

$t \leftarrow 0$;

while *there are new links predicted failed at time* t **do**

  $\tilde{s}_i^{new}[t+1] \leftarrow \sum_{j=1}^{M} d_{ij}(\mathbf{A}_{ji}^{11} s_j^{new}[t] + \mathbf{A}_{ji}^{01}(1 - s_j^{new}[t]))$ for each link i;

  $\hat{s}_i^{new}[t+1] \leftarrow 0$ if $\tilde{s}_i^{new}[t+1] < \hat{\epsilon}_i$, $\hat{s}_i^{new}[t+1] \leftarrow 1$ otherwise;

  $s^{new}[t+1] \leftarrow \tilde{s}^{new}[t+1]$, $t \leftarrow t+1$;

$T \leftarrow t-1$;

return $\mathbf{D}$, $\mathbf{A}^{11}$, $\mathbf{A}^{01}$, $\hat{s}^{new}$.

V. Performance Evaluation

In this section, we present a comprehensive numerical study of the proposed method for failure cascade prediction.

A. Dataset Information

We consider three large scale power systems, namely, 1354-bus, 2383-bus, and 3012-bus, which can be found in Matlab MATPOWER toolbox [18]. These systems are mostly equipped with given transmission link capacities, with only a few not given[5]. We assume that these unrated links are free from overloading in simulations. We further exclude links that never fail in the training sample pool $\mathcal{S}_{train}$, and term the rest as *effective links*. In our experiments, $\mathcal{S}_{train}$ contains $50,000$ cascade samples under $M-2$ contingencies, out of a total of $\binom{M}{2}$ where M is the number of links in a system.

[5]The `rateA` in MATPOWER toolbox denotes given capacity value [18].

Table I displays a brief summary of cascade samples for all test systems under default loadings in MATPOWER, where 'Eff. Rate' is the portion of effective links, and 'Fail Size' is the number of link outages in a cascade. Note that the initial outages for training in each system accounts for very small portion: $50,000/\binom{1710}{2} = 3.4\%$ in 1354-bus system and even lower in larger systems. We will show that our approach can capture most of the cascade patterns with these samples.

TABLE I: Default Cascade Sample Information

System	1354-Bus	2383-Bus	3012-Bus
#Generators	260	327	297
#Links	1710	2886	3566
#Eff. Links	762	2088	2083
Eff. Rate	44.6%	72.4%	58.4%
Avg. Fail Size	179	598	263
Max Fail Size	314	862	792
Min Fail Size	2	110	11

B. Performance Metrics

To evaluate prediction performance, we consider 4 metrics.

- **Avg. Failure Size Error Rate** l_{size}: $l_{size} = \frac{1}{K}\sum_{k=1}^{K} l_{size}^k$, where l_{size}^k is the failure size prediction error, relative to real failure size, in the k-th test sample.
- **Avg. Failure Frequency Error** l_{freq}: $l_{freq} = \frac{1}{M}\sum_{i=1}^{M} l_{freq}^i$, where l_{freq}^i is the absolute difference between real and predicted failure frequency of link i among all test samples.
- **Avg. Final State Error Rate** l_f: $l_f = \frac{1}{K}\sum_{k=1}^{K} l_f^k$, where l_f^k is the ratio of links whose final states are mistakenly predicted, in the k-th test sample.
- **Avg. Failure Time Error** l_t: $l_t = \frac{1}{K}\sum_{k=1}^{K} l_t^k$, where l_t^k is the failure time prediction error among all links that fail eventually in the k-th test sample.

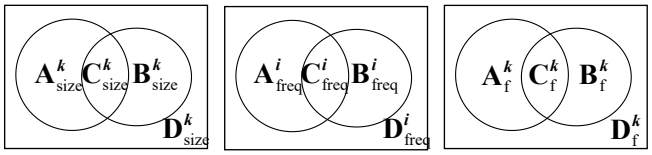
Fig. 3 illustrates the way we calculate l_{size}^k, l_{freq}^i, l_f^k, and l_t^k for each test sample s_{test}^k and each link i. For l_{size}^k, l_{freq}^i, and l_f^k, we use Venn graph, divided into four disjoint subsets A, B, C, D, to show the calculation. Inside, $A \cup C$ denotes the set of real failures, $B \cup C$ denotes the set of predicted failures, C is the set of correctly predicted failures, while D represents links not failed and meanwhile not predicted to be failed, under each metric.

- For l_{size}^k, $|A_{size}^k \cup C_{size}^k|$ is the real failure size while $|B_{size}^k \cup C_{size}^k|$ is the predicted failure size in s_{test}^k, and
$$l_{size}^k = \frac{\big||B_{size}^k \cup C_{size}^k| - |A_{size}^k \cup C_{size}^k|\big|}{|A_{size}^k \cup C_{size}^k|} \times 100\%.$$
- For l_{freq}^i, $|A_{freq}^i \cup C_{freq}^i|$ is the total number of test samples in which link i fails but not from the beginning, while $|B_{freq}^i \cup C_{freq}^i|$ is our prediction towards it, and
$$l_{freq}^i = \big||B_{freq}^i \cup C_{freq}^i| - |A_{freq}^i \cup C_{freq}^i|\big|.$$
- For l_f^k, $|A_f^k \cup B_f^k|$ is the number of links whose final states we mistakenly predict in s_{test}^k, while $|A_f^k \cup B_f^k \cup$

$C_f^k \cup D_f^k|$ denotes the number of effective links, and

$$l_f^k = \frac{|A_f^k \cup B_f^k|}{|A_f^k \cup B_f^k \cup C_f^k \cup D_f^k|} \times 100\%.$$

For l_t^k, we take an example on a cascade sequence over 6 links that fail during the cascade. Each number in the first row denotes the time step a link changes to failure, while each number in the second row denotes our prediction on the time step[6]. We ignore counting in initially failed links in all 4 metrics.



	Link 1	Link 2	Link 3	Link 4	Link 5	Link 6
Real	4	3	5	4	2	1
Pred.	2	5	5	1	5	3

$$\boldsymbol{l}_{\text{t}}^{\boldsymbol{k}} = \frac{1}{6}\big(|4-2|+|3-5|+|5-5|+|4-1|+|2-5|+|1-3|\big) = 2$$

Fig. 3: An illustrative example of all metrics

These metrics represent four levels of granularity shown in Fig. 4. l_{size} focuses on the number of failures but does not care about the accuracy on each link. l_{freq} casts light on failure risk of each link, but does not specify in which case a link will fail. l_f reflects the binary prediction accuracy on each link in each cascade, while l_t further unveils the prediction performance at each time step. To the best of our knowledge, this is the first attempt to evaluate these four levels of metrics.

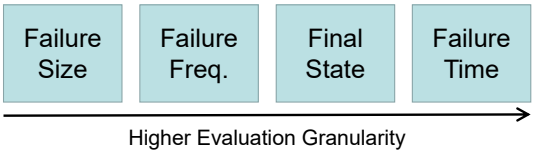


Fig. 4: Four levels of granularity in performance evaluation

C. Test Results

In this part, we summarize our main results for failure cascade prediction in order of granularity. To study the performance under different loading conditions, we proportionally increase power generation and loading. In the 1354-bus system, we take 1, 1.5 and 2 times of original condition, while in the 2383-bus and the 3012-bus system we take 1, 1.25, and 1.5 times, where the lower loading increment is because the data for these two systems are measured at peak power in winter.

Under all settings, we evaluate the prediction performance on $2,000$ test samples (different from any training sequence in $\mathcal{S}_{train}$). For each metric, we present two categories of results. The first is how the performance varies from $5,000$ to $50,000$ training samples. The second is the cumulative distribution function (CDF) of the corresponding error metric on each test sample when learning with $50,000$ training samples.

[6]If we predict a link to be normal, then the corresponding number in the second row is our predicted termination time of this cascade.

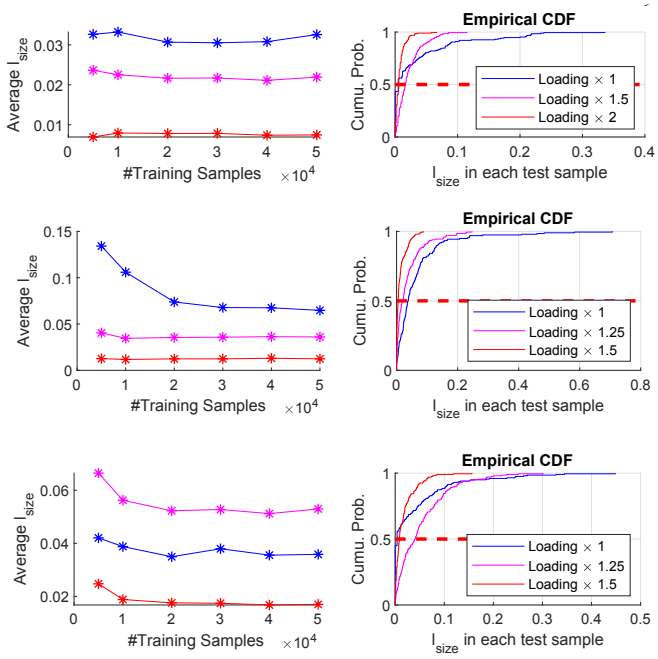


Fig. 5: Results on l_{size}: 1353, 2383, 3012-bus systems in order

1) Level 1–Failure Size l_{size}: Fig. 5 shows the results on l_{size}. Generally, by incorporating more training samples, l_{size} becomes lower and drops below 7%, indicating reasonable prediction. For example, if 200 links fail eventually, then our predicted failure size will be within $[186, 214]$ on average. Note that the decreasing trend diminishes after $20,000$ training samples, indicating that our approach can offer good enough failure size prediction with limited cascade records. From CDF plots, on the other hand, we can find that in each case, l_{size} is less than 10% in more than 80% of test samples, and almost all predictions will not make l_{size} larger than 30%.

2) Level 2–Failure Frequency l_{freq}: Fig. 6 presents the results on l_{freq}. Generally, $l_{freq} \leq 0.08$ in almost all cases, which indicates that the absolute frequency prediction error will not deviate much. For example, if a link fails with frequency 0.3, then our prediction on it will lie within $[0.22, 0.38]$ in expectation. Unlike l_{size}, however, l_{freq} does not decrease monotonically in our simulations. Fig. 6 shows some fluctuations of l_{freq} when we tested on different sample sets. It may require more sample data to stabilize. From the CDF plots, we can observe that under most conditions around 75% of the links have $l_{freq} < 0.1$. However, the medium loading conditions (1.5 and 1.25 times loading) in 1354-bus and 3012-bus systems yield the worst CDF results. It is caused by the fact that a light loading condition or a heavy loading condition induces a rather simple failure

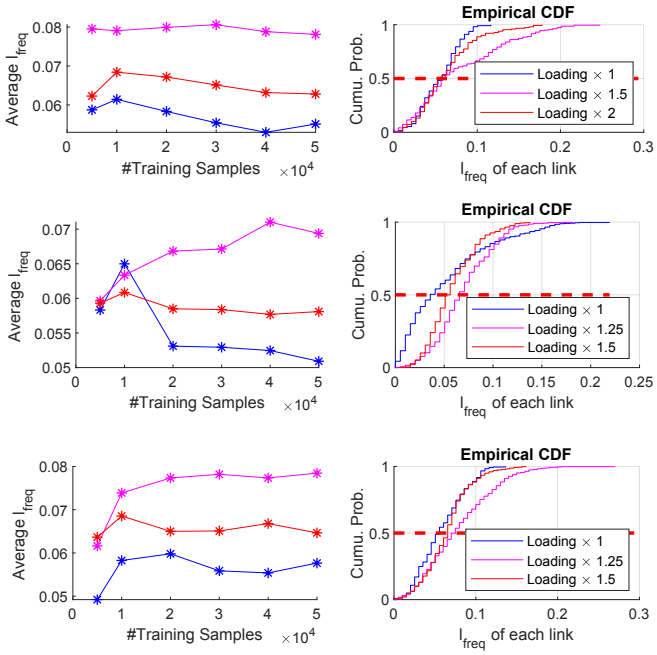


Fig. 6: Results on l_{freq}: 1353, 2383, 3012-bus systems in order

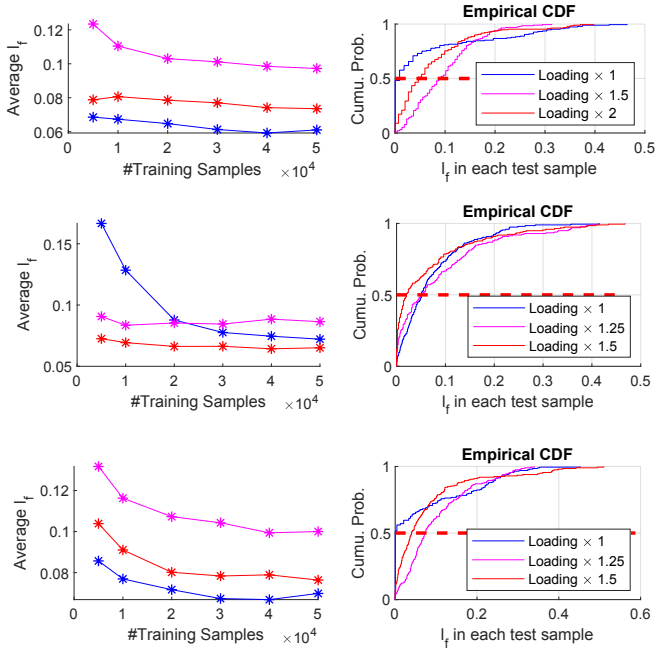


Fig. 7: Results on l_f: 1353, 2383, 3012-bus systems in order

cascade situation (either mostly in small scale or mostly in large scale), however, a medium loading can introduce a more complicated failure cascade situation, sometimes in small scale and sometimes in large scale.

3) Level 3–Final State l_f: Fig. 7 presents the results on l_f. We can observe that l_f declines generally as more training samples are involved, and l_f can be lower than 10% in most cases. This means that among $1,000$ effective links, we can predict more than 900 of them correctly under different initial outages. The CDF plots further demonstrate that for at least half of the test samples, l_f will be smaller than 10% among all tested conditions, and altogether more than 95% of all the predictions cause l_f smaller than 30%.

4) Level 4–Failure Time l_t: Fig. 8 illustrates the results on l_t. l_t mostly decreases under more training samples, while in other settings the error keeps stable at a low level. In most of cases, l_t is within 1 time step, showing that failure time prediction is valid. For example, in 1354-bus system the cascade generally lasts for 10 time steps, if it lasts for 20 minutes in reality, then our prediction can cause error within 2 minutes in most situations. In perspective of CDF, we can further draw that in most cases around 90% of test samples can achieve time error within 2 steps, where medium loading conditions are generally harder to predict. This may stem from longer cascade duration under medium loading, as failure propagates rapidly and terminates in few steps under high loading, and also stops early under low loading.

5) Prediction Time Cost: We show the superiority of the influence model based prediction to power flow calculation by MATPOWER Toolbox in time cost reduction. Specifically, we test both methods in MATLAB 2019a on Intel(R) Core(TM) i9-7920X CPU@2.90GHz Processor with 128GB installed memory. In each case, we run $1,000$ test sample via each method and compute its total time cost.

We summarize the results in Table II. We use '$a \mid b \mid \mathbf{c}$' structure to present our results, where a denotes total seconds by DC flow calculation, b denotes total seconds by our method, and $\mathbf{c} = a/b$, reflecting how many times faster the IM is compared to the traditional power flow method. Results show that our method works better in larger systems and under higher loading conditions that tend to cause large failures, which demonstrates its effectiveness and scalability. For example, the time cost by our method is $1/136$ of that by flow calculation under medium loading in the 3012-bus system. The main reason is that more islands appears under these conditions, which requires to solve DC flow equations for more times, while the prediction by influence model discards all such calculations.

TABLE II: Prediction Time Cost on 1,000 Samples

	Low Load	Medium Load	High Load
1354-bus	808 | 21.3 | **38**	1930 | 19.8 | **97**	1740 | 19.6 | **89**
2383-bus	2597 | 43.3 | **60**	3490 | 37.8 | **92**	3603 | 34.7 | **104**
3012-bus	3891 | 59.4 | **66**	8020 | 58.9 | **136**	5864 | 46.9 | **125**

VI. Conclusion and Future Work

In this paper, we build an influence model framework to study and predict failure cascades in large scale power systems. We propose a hybrid learning scheme to train the influence model based on simulated failure cascade samples. The scheme consists of three steps in sequel. Firstly, it adopts a Monte-Carlo approach to learn pairwise influence between any two transmission links. Then it formulates a convex quadratic programming problem to learn the weight of each pairwise influence in determining network state

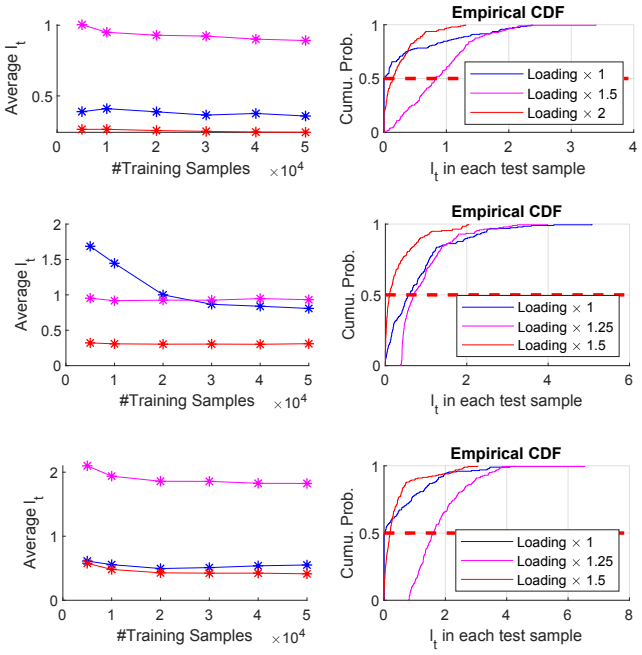


Fig. 8: Results on l_t: 1353, 2383, 3012-bus systems in order

transitions. Finally, it selects the bisection threshold for each transmission link in an adaptive manner to better capture different cascades. Experimental results on large scale power systems demonstrate acceptable prediction performance over 4 levels of evaluation granularity. Moreover, we show that our approach can be two orders of magnitude faster than power flow simulation based approach, which is promising for online screening and $M-k$ contingency analysis.

One of the future research directions is to predict failure cascades under the AC power flow model. The AC model will not change the underpinning of the proposed hybrid learning scheme. A potential faster prediction speed (compared to the power flow based prediction) is expected, since solving the AC power flow problem is much slower than solving the DC one. Another potential research direction will be exploring failure cascade features for different power systems from their $\mathbf{D}$ matrices. From our observations, the weighted influence matrix $\mathbf{D}$ has shown certain clustering patterns and sparsity structure, which may reveal some intrinsic information about the system.

REFERENCES

[1] "NorthEast US Failure Cascade," https://www.bostonglobe.com/magazine/2012/02/03/anatomy-blackout-august/mAsrr41nLAjGFIU3IF440O/story.html.

[2] "Manhattan, New York Failure Cascade," https://www.theatlantic.com/technology/archive/2019/07/manhattan-blackout-reveals-infrastructure-risk/594025/.

[3] "London Failure Cascade," https://www.bloomberg.com/news/articles/2019-08-09/london-blackout-occurred-amid-drop-in-wind-and-natural-gas-power.

[4] A. Bernstein, D. Bienstock, D. Hay, M. Uzunoglu, and G. Zussman, "Power grid vulnerability to geographically correlated failuresłanalysis and control implications," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 2634–2642.

[5] H. Cetinay, S. Soltan, F. A. Kuipers, G. Zussman, and P. Van 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, 2017.

[6] 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, 2015.

[7] S. Soltan, A. Loh, and G. Zussman, "Analyzing and quantifying the effect of k-line failures in power grids," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 3, pp. 1424–1433, 2017.

[8] Z. Kong and E. M. Yeh, "Correlated and cascading node failures in random geometric networks: A percolation view," in *2012 Fourth International Conference on Ubiquitous and Future Networks (ICUFN)*. IEEE, 2012, pp. 520–525.

[9] H. Xiao and E. M. Yeh, "Cascading link failure in the power grid: A percolation-based analysis," in *2011 IEEE International Conference on Communications Workshops (ICC)*. IEEE, 2011, pp. 1–6.

[10] Z. Wang, D. Zhou, and Y. Hu, "Group percolation in interdependent networks," *Physical Review E*, vol. 97, no. 3, p. 032306, 2018.

[11] C. Asavathiratham, S. Roy, B. Lesieutre, and G. Verghese, "The influence model," *IEEE Control Systems Magazine*, vol. 21, no. 6, pp. 52–64, 2001.

[12] M. Rahnamay-Naeini, Z. Wang, N. Ghani, A. Mammoli, and M. M. Hayat, "Stochastic analysis of cascading-failure dynamics in power grids," *IEEE Transactions on Power Systems*, vol. 29, no. 4, pp. 1767–1779, 2014.

[13] P. Das, R. A. Shuvro, Z. Wang, M. R. Naeini, N. Ghani, and M. M. Hayat, "Stochastic failure dynamics in communication network under the influence of power failure," in *2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 2017, pp. 1–8.

[14] R. A. Shuvro, Z. Wang, P. Das, M. R. Naeini, and M. M. Hayat, "Modeling impact of communication network failures on power grid reliability," in *2017 North American Power Symposium (NAPS)*. IEEE, 2017, pp. 1–6.

[15] P. D. 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, 2016.

[16] K. Zhou, I. Dobson, Z. Wang, A. Roitershtein, and A. P. Ghosh, "A markovian influence graph formed from utility line outage data to mitigate cascading," *arXiv preprint arXiv:1902.00686*, 2019.

[17] I. Dobson, "Estimating the propagation and extent of cascading line outages from utility data with a branching process," *IEEE Transactions on Power Systems*, vol. 27, no. 4, pp. 2146–2155, 2012.

[18] "IEEE Power Systems," https://github.com/MATPOWER/matpower.

[19] M. Jaggi, "Revisiting frank-wolfe: Projection-free sparse convex optimization." in *ICML (1)*, 2013, pp. 427–435.