Theoretical Computer Science 607 (2015) 306–319

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Catastrophic cascading failures in power networks ☆

Jungtaek Seo [a], Subhankar Mishra [b], Xiang Li [b], My T. Thai [c,b,*]

[a] The Attached Institute of ETRI, Republic of Korea
[b] Dept. of Comp. and Info. Sci. and Eng., University of Florida, Gainesville, 32611, USA
[c] Division of Algorithms and Technologies for Networks Analysis, Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Viet Nam

ARTICLE INFO

Article history:
Received 17 July 2015
Accepted 24 August 2015
Available online 24 September 2015

Keywords:
Cascading failure
Power network
Optimization

ABSTRACT

The high demand of electricity makes power networks more vulnerable under cascading failures. Because of the operational dependencies between nodes, the failure of a small set of nodes can cause a large cascade of failures which results in the breakdown of the network. Thus, it is crucial to study the vulnerability of the power network under the cascading failures.

In this paper, we study the cascading critical node (CasCN) problem which asks to find a set of nodes whose failure maximizes the number of failed nodes under the effect of cascading failures. We first show that the problem is NP-hard to approximate within the factor of $O(n^{1-\epsilon})$. We then design a new metric to evaluate the importance of nodes in the network and use it as the base to design the Fully Adaptive Cascading Potential algorithm. In the case where the network is robust, we propose an alternative algorithm, the Cooperating Attack algorithm, which includes several novel properties to solve the problem. Simulation results demonstrate the efficiency of proposed algorithms and provide more insight into the vulnerability of the power network.

1. Introduction

The complex network systems are now the essential parts of a modern society. Many complex systems are extremely vulnerable to attacks, that is, the failures of a few key nodes that play a vital role in maintaining the network's connectivity can break down their operation. In addition, this vulnerability may be propagated, leading to a much more devastating consequence. A failure within a single component may have a profound impact on large parts of the system. The nodes' failures may change the balance of flows and lead to a global redistribution of loads over the entire network. This can trigger a cascade of overload failures. Cascades can therefore be regarded as a specific manifestation of the robust yet fragile nature of many complex systems: a system may appear stable for long periods of time and withstand many external shocks (robust), then suddenly and apparently inexplicably exhibit a large cascade (fragile). These phenomena are all examples of what economists call information cascades (Ref. [4]; but which are herein called simply cascades), during which individuals in a population exhibit herd-like behavior because they are making decisions based on the actions of other individuals

☆ The first two authors contribute equally to this work.

* Corresponding author at: Division of Algorithms and Technologies for Networks Analysis, Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Viet Nam and Dept. of Comp. and Info. Sci. and Eng., University of Florida, Gainesville, 32611, USA.

E-mail addresses: seojt@ensec.re.kr (J. Seo), mishra@cise.ufl.com (S. Mishra), xixiang@cise.ufl.com (X. Li), thaitramy@tdt.edu.vn, mythai@cise.ufl.com (M.T. Thai).

http://dx.doi.org/10.1016/j.tcs.2015.08.021
0304-3975/

rather than relying on their own information about the problem. Although they are generated by quite different mechanisms, cascades in social and economic systems are similar to cascading failures in physical infrastructure networks and complex organizations in that initial failures increase the likelihood of subsequent failures, leading to eventual outcomes, are extremely difficult to predict, even when the properties of the individual components are well understood.

The common denominator of large blackouts is that the failures of components happened according to the cascading manner. It often starts with the failure of one or a few components, then some other components are failed due to the dependencies with previous failed components. The failure of these components continue to cause other components fail. The process continues until there is no more failed component. In power networks, generator, distribution, and consumption stations can only work well if the load is under the maximum capacity they can handle. When a station is overloaded, it cannot work with the best performance or even fails. During the operation, the power network is designed such that all stations work under their capacity. But when some stations fails, other stations which are directly or indirectly connected with failed ones may have bear more load. If the load of a station surpasses its capacity, it will fail and continue to shred their load to other stations. As a result of the load redistribution process, a large number of failed stations may be failed at the end. The dependencies between stations in the network make the outcome difficult to be predicted. Thus, it is necessary to develop efficient tools to analyze the sophisticated cascading process of failures in power networks.

The cascading failure has attracted a lot of attention and been studied in various perspective [1–8]. The structural vulnerability of power networks was studied in [2]. The authors showed that removing small fraction of highest degree nodes significantly reduces the connectivity of the network. After that, Hines et al. [4] studied the network vulnerability of different classes of scale-free networks including Erdos–Renyi, preferential-attachment, and small-world networks. They showed that different types of networks behave differently under node failures. Various models of cascading failures were later proposed to study the vulnerability of networks under the targeted attack [6,9,10,7]. However, these works mainly present different ranking methods for nodes and select most the highest ranked nodes as the critical ones. These methods fail to address the effect of the cascading process.

In this paper, we study the vulnerability of power grid under the targeted attack with the effect of the cascading failures. More specifically, we aim to find a set of k nodes whose failures maximize the number of failed nodes when the cascading of failures stops. It is challenging to identify such set of nodes due to the complicated interaction between nodes in the network. Thus, it is impossible to solve the problem optimally, especially on large networks. Our approach is to design a new measure for the importance of nodes considering the cascading effect, then develop efficient algorithms from that.

Our main contributions are summarized as follows:

- Evaluate the vulnerability of the power grid under a new kind of attack in which nodes are attacked one by one to increase the damage. Since the cascading failures happen fast, an attacker can choose the next attacked node based on the status of the network after the effect of previous attacks. By this way, the attacker can gain more failed nodes.
- Show that the proposed problem is NP-hard to approximate within the ratio of $O(n^{1-\epsilon})$.
- Introduce a new metric called *cascading potential* to measure the importance of nodes when the cascading effect is considered.
- Propose various algorithms which can work well under the variety of network settings.
- Validate the efficiency of proposed algorithms in a wide range of network configurations and provide new insights into the vulnerability of the power grid.

The rest of the paper is organized as follows. In Section 2, we present the failure cascading model and formulate the problem. Then, we show the hardness result in Section 3. After that, we propose the cascading potential metric and design various algorithms in Section 4. We next introduce the cooperating algorithm which is efficient on robust networks in Section 5. Section 6 shows the experimental evaluation. Finally, we conclude the paper in Section 7.

2. Network model and problem formulation

2.1. Graph notations

The network is modeled by a weighted directed graph $G = (V, E)$ with the node set V of $|V| = n$ nodes and the edge set E of $|E| = m$ oriented connections between nodes. Each edge (u, v) is associated with a weight $w(u, v)$ representing the operating parameter of the network. The higher $w(u, v)$ is, the more load is distributed from u to v. In addition, each node u has a current load $L(u)$ and a capacity $C(u)$. The capacity $C(u)$ is the maximum load that node u can accept. Finally, we denote the set of incoming neighbors, outgoing neighbors of u by N_u^- and N_u^+, respectively.

2.2. Cascading failure model

In this paper, we adopt the Load Redistribution model (LR-model) which was widely used in the research community [11,6]. In this model, nodes are failed in the cascading manner due to the load redistribution of failed nodes. Initially, a set of nodes S are failed, then the failures are propagated to other nodes in time steps. When node u fails, its load is redistributed to its neighbors as illustrated in Fig. 1. Each alive neighbor v will received an additional load which is proportional to weight $w(u, v)$ of edge from u to v. Precisely, v will receive additional load:

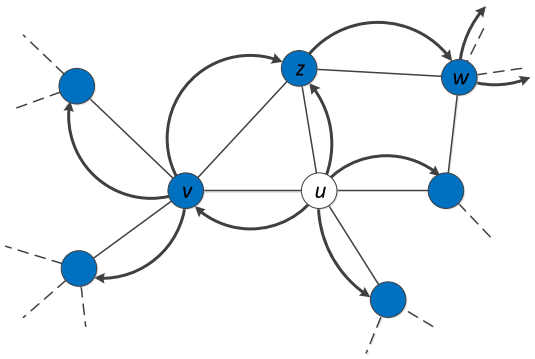


Fig. 1. When node u fails, its load is redistributed to the neighbor nodes. Among these nodes, v receives a high portion of load from u and becomes overloaded. The load of v is redistributed to its neighbors which makes z fail. Finally, the load from z continues to cause w fail and the cascading process stops.

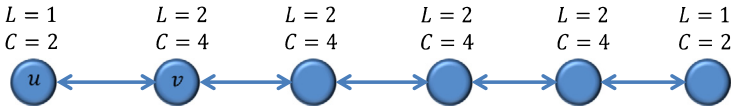


Fig. 2. An instance about the efficiency of the serial attack. Simultaneously attacking any two nodes causes no cascading failures. If u is attacked first, then its load is redistributed to v. After that, attacking v leads to the failure of all remaining nodes.

$$\Delta L(v) = L(u) \times \frac{w(u,v)}{\sum_{z \in N_u^+} w(u,z)}$$

Due to the load redistribution, the load of some nodes are exceeding their capacities, hence fail in the next time step. The process of load redistributing and node failing will stop when there are no more failed nodes. The set of failed nodes caused by the initial failure of S is denoted by $F(S)$.

2.3. Problem definition

Due to the cascading failures, the failures of a small set of nodes S can result in a catastrophic number of failed nodes. These nodes become the target to attack the network. Additionally, given the same set of attacked nodes, different attacking orders lead to different outcomes. With the same attacking cost, the attacker can choose the best order with suitable time for each attacked node. However, the cascading failures happen very fast, it is almost impossible to schedule the failure of each node with specific time steps. We consider a more practical strategy in which targeted nodes are attacked one by one. An example of the efficiency of this strategy is illustrated in Fig. 2. The next node is taken down when the cascading process stops. In particular, given an order set $S = \{s_1, s_2, \ldots, s_k\}$, node s_i is attacked after the cascading process caused by attacking s_{i-1} stops. Denote $F^+(S)$ as the number of failed nodes when nodes in S are attacked serially. We formally define the problem as follows.

Definition 1 *(Cascading Critical Node (CasCN) problem).* Given a network $G = (V, E)$ and a positive integer k, the problem asks to find a ordered subset $S \subseteq V$ of size $|S| = k$ such that the serial failures of nodes in S maximizes the number of failed nodes $F^+(S)$ under the LR-model.

3. Inapproximability result

In this section, we show the algorithmic hardness of the CasCN problem. We expect to design an algorithm that can identify the optimal seed set in an acceptable time. However, it may take the time as an exponential function of the number of nodes to compute even a set whose impact is close to the optimal set's. The hardness result is shown in Theorem 1.

Theorem 1. *It is NP-hard to approximate the CasCN problem within ratio of $O(n^{1-\epsilon})$ for any constant $1 > \epsilon > 0$.*

Proof. We use the gap-introduction reduction [12] to prove the inapproximability of the CasCN problem. Using a polynomial time reduction from Set Cover, to the CasCN problem, we show that if there exists a polynomial time algorithm that approximates the later problem within $O(n^{1-\epsilon})$, then there exists a polynomial time algorithm to solve the former problem.

Definition 2 *(Set Cover problem).* Given a universe $\mathcal{U} = \{e_1, e_2, \ldots, e_n\}$, a collection of subsets $\mathcal{S} = \{S_1, S_2, \ldots, S_m\} \subseteq 2^{\mathcal{U}}$, and an integer k, the Set Cover problem asks whether or not there are k subsets whose union is $\mathcal{U}$.

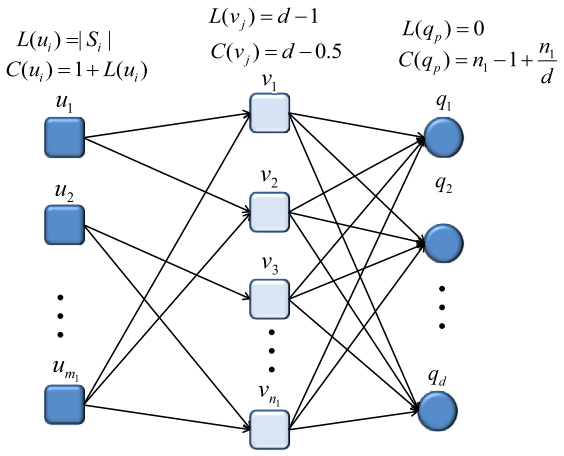


Fig. 3. Reduction form MIN3SC2 to CasCN.

Instead of using the hardness result of the general Set Cover problem, we use the result on a restricted variant MIN3SC2 of the Set Cover problem where the sizes of subsets are at most 3 and each element appears in exactly two subsets.

Theorem 2. *(See [13].) The Set Cover problem is NP-hard even when the sizes of subsets are bounded by 3 and each element appears in exactly two subsets.*

Reduction. Given an instance of the Set Cover problem $\mathcal{I} = (\mathcal{U}, \mathcal{S}, k)$ where each element appears in exactly two subsets, $n_1 = |\mathcal{U}|$ and $m_1 = |\mathcal{S}|$, we construct an instance $\mathcal{I}'$ of the CasCN problem as illustrated in Fig. 3.

The node set V. Add a *set node* u_i for each set $S_i \in \mathcal{S}$, an *element node* v_j for each element $e_j \in \mathcal{U}$, and $d = (n_1 + m_1)^{\frac{2}{\epsilon}}$ *extra nodes* $q_1, q_2, \ldots, q_d$.

The edge set E. Add edge (u_i, v_j) if the element v_j in the set S_i. In addition, there is an edge from v_j to q_p $\forall 1 \le j \le n_1$, $1 \le p \le d$. All edges have the weight of 1.

Node load and capacity. The load and capacity of the set node u_i are $L(u_i) = |S_i|$ and $C(u_i) = 1 + L(u_i)$. The load and capacity of the element node v_j are $L(v_j) = d - 1$ and $C(v_j) = d - 0.5$. All extra nodes have the load of 0 and capacity of $n_1 - 1 + \frac{n_1}{d}$.

Next, we prove that if $\mathcal{I}$ has a set cover of size k then there exist a seeding set $A \subset V$ such that $|F(A)| > d$. Otherwise, for all $A \subset V$ and $|A| \le k$, $|F(A)| < n_1 + m_1$.

Assume that $\mathcal{I}$ has a set cover $\mathcal{SC}$ of size k, then we will select the set $A = \{u_i | S_i \in \mathcal{SC}\}$ as the seeding set. Initially, each node $u_i \in S$ redistributes 1 unit of load to each of its $|S_i|$ neighbors. Since each element is covered by at least one set in $\mathcal{SC}$, each node v_j receives the additional load of at least 1. At the next round, v_j has the load at least $L(v_j) \ge (d-1) + 1)$, which is higher than its capacity, and is failed. When v_j fails, it equally redistributes $L(v_j)/d \ge 1$ load to its d extra neighbor nodes. The load of each extra node q_p after receiving load from all failed element nodes is $L(q_p) \ge n > C(q_p)$, hence all extra nodes are failed. The cascading process stops with $|F(A)| = d + n_1 + k$ failed nodes.

In the case $\mathcal{I}$ has no set cover of size k, we will show the optimal seeding set can cause at most $n_1 + k$ nodes fail in $\mathcal{I}'$. Let A be an arbitrary optimal seeding set. We observe that a set node only fails when it is selected in the seeding set since it has no incoming edge. Thus, there are at least $m_1 - k$ set nodes which are not in A. We can replace any extra node $q_p \in A$, if there are any, by a unselected set node without decreasing the number of failed nodes. Next, suppose that there exists an element node $v_j \in S$, we can also replace it by a set node. If v_j is adjacent to some node $u_i \in A$, we can remove v_j from A while maintaining the same number of failed nodes. If v_j is not adjacent to any node in A, we just replace v_j by one of its neighborhood set node u_i. u_i will make v_j fail, so the number of failed nodes caused by A is not decreased. So, we can replace extra and element nodes in A such that A contains only set nodes. Since, there is no set cover of size k, there at least one node is not adjacent to any node in A, i.e., the number of failed element nodes is at most $n_1 - 1$. Each failed element node v_j is adjacent to at most 2 set nodes, hence its load is at most $L(v_j) \le d + 1$. Each extra node q_p receives at most $(d+1)/d$ redistributed load from failed element nodes which are accumulated to at most:

$$\frac{(n_1 - 1)(d+1)}{d} = n_1 - 1 + \frac{n_1 - 1}{d} < n_1 - 1 + \frac{n_1}{d} = C(q_p)$$

Thus, there is no extra node fails. The total failed nodes caused by A is at most $n_1 + k < n_1 + m_1$.

Now suppose that we have polynomial algorithm $\mathcal{A}$ which approximates CasCN problem within $n^{1-\epsilon}$, we can decide the set cover problem as follows. For any instance $\mathcal{I}$ of the Set Cover problem, we construct the instance $\mathcal{I}'$ as above in polynomial time as d is a polynomial function of n_1 and m_1. Now, if $\mathcal{I}$ has a set cover of size k, the optimal A_{opt} seeding

set causes at least d nodes fail in $\mathcal{I}'$. The algorithm $\mathcal{A}$ approximate the optimal solution within $n^{1-\epsilon}$ ($n = n_1 + m_1 + d$, the number of nodes in $\mathcal{I}'$), so it finds a seeding set $\mathcal{A}(\mathcal{I}')$ whose causes at least $(m_1 + n_1)$ nodes fail:

$$\begin{aligned} |F(\mathcal{A}(\mathcal{I}')| \geq \frac{|F(A_{opt})|}{n^{1-\epsilon}} &> \frac{d}{(d+m+n)^{1-\epsilon}} \\ &> \frac{d}{(2d)^{1-\epsilon}} > \frac{d^{\epsilon}}{2} \\ &= \frac{(m_1+n_2)^2}{2} > m_1 + n_1 \end{aligned}$$

On the other hand, if $\mathcal{I}$ has no set cover of size k, then the optimal seeding set A_{opt} of $\mathcal{I}'$ causes less than $(m_1 + n_1)$ nodes fail. We have:

$$|F(\mathcal{A}(\mathcal{I}')| \geq |F(A_{opt})| < (m_1 + n_1)$$

It implies the $\mathcal{I}$ has a set cover of size k if and only if $|F(\mathcal{A}(\mathcal{I}')| > (m_1 + n_1)$. Hence, we can use $\mathcal{A}$ to decide the Set Cover problem in polynomial time i.e. $P = NP$. □

4. Cascading potential and derived algorithms

In this section, we introduce a new metric to measure the node importance under the cascading failure, and then use it as a criterion to design efficient algorithms for CasCN. To evaluate the vulnerability of networks under the load redistribution, previous works in the literature propose various ranking methods and measure the effect of attacking top k nodes. However, these methods consider very limited topological information, hence may miss the most critical nodes. In [6,9,10], the authors solely use the load as the criterion to rank nodes. The failure of a high load node intuitively tends to cause a large number of nodes to fail as it redistributes a large amount of load to its neighbors, but cascading failures starting from a small load node at the right position may result in a larger number of failed nodes [7]. Although Wang et al. [7] improved the ranking by counting the number of failed nodes, the mutual impact of multiple nodes is still excluded. When multiple nodes are attacked in the network, the effect of the early nodes is strengthened by the later nodes. Next, we introduce a new metric which considers both direct and mutual impact of the node.

4.1. Cascading potential

The cascading potential of a node is defined as combination of all possible impacts a node causes in the network under the cascading effect. Let's consider the failure of node u. For any other node v, there are two possible impacts that u can induce on v:

- *Failure impact*. The failure of u leads to the failure of v.
- *Load impact*. The failure of u makes the load of v increase but not enough to fail.

The overall *failure impact* and *load impact* of u in the network are defined as the number of failed nodes and the total increase in load of the unfailed nodes, respectively. The *cascading potential* of u is the linear combination of these factors:

$$\mathcal{C}(u) = \frac{|F(\{u\})|}{n} + \frac{\sum_{v \in V - F(\{u\})} \Delta L_u(v)}{\sum_{v \in V - F(\{u\})} (C(v) - L(v))}$$

where $F(\{u\})$ is the set of failed nodes when u fails and $\Delta L_u(v)$ is the additional load that v receives due to the failure of u.

In this formula, we normalize both the failure and load impacts to avoid the unit difference. The failure impact is divided by the number of nodes, hence is at most 1 when all other nodes fail. Similarly, the load impact is divided by the total of capacity-load difference of unfailed nodes and achieves the maximum value 1 when all remained nodes are at the edge of failure, i.e., the most vulnerable state of the network.

The role of the load impact. In the formulation of the cascading potential, the load impact plays an important role in providing a better assessment of the network vulnerability comparing to the metric in [7]. If only one node is attacked, it is obvious to choose the node which maximizes the number of failed nodes, i.e., to use Wang et al.'s metric. However, when multiple nodes are attacked, we need to consider the co-impact of attacked nodes to trigger a large size cascading failure. The load impact is bridge connecting the impact of these nodes. If the total load impact of earlier attacked nodes is high, the load of remaining nodes is close to the capacity. As a result, later attacked nodes can make more nodes to fail easier. For example, if u has the maximum load impact of 1 and the network is strongly connected, then attacking any node after u can take down the whole network. Thus, the cascading potential evaluate the importance of nodes more comprehensively.

Algorithm 1 Cascading Potential (CP) algorithm.

Input: A network $G = (V, E)$, an integer k.
Output: A set S of k attacked nodes.
Compute the cascading potential of all nodes
Sort nodes in non-increasing order of the cascading potential $\mathcal{C}(u_1) \geq \mathcal{C}(u_2) \geq \ldots \geq \mathcal{C}(u_n)$
Initialize $S \leftarrow \emptyset$
$j \leftarrow 1$
for $i = 1$ to k **do**
  $S \leftarrow S \cup \{u_i\}$
end for
Return S

4.2. Cascading potential algorithm

Intuitively, we can use cascading potential directly to design an algorithm for CasCN. We first compute the cascading potential of all nodes, then select top k as attacked nodes. The algorithm is described in Algorithm 1.

Time complexity. It takes at most $O(m)$ to compute the cascading potential of each node. Thus, the total running time is $O(nm + n \log n)$.

4.3. Partially adaptive cascading potential algorithm

The Cascading Potential algorithm runs fast, but it neglects an important property of the cascading failure: the overlapped impact of selected nodes. Let consider two nodes u and v which both have failure impact on node z. If u is selected before v, then v has no impact on z as z is already failed. As a consequence, some nodes that have a high impact initially may have a little impact at the later stage. To handle this situation, we propose to update the impact of remained nodes on the fly. More specifically, at the ith iteration, the impact (failure or load impact) of node u on failed nodes (due to the selection of first $i-1$ attacked nodes) will be subtracted from the initial impact of u. After that, the node with the highest remained impact will be selected.

The crucial problem is how to update the impact of nodes efficiently. A naive way of keeping the impacted nodes list at each node u will result in $\Omega(n^3)$ running time for each iteration, which is very time consuming. We reduce the updating time by reversing the process. Each node v will keep two lists of nodes: the list $FI[v]$ contains nodes which have failure impact on v and the list $LI(v)$ contains nodes which have load impact on v. Since the load impact of other nodes on v are different, we use $LI[v][u]$ to store the load impact of u on v after the normalization. When v is failed, the impact of nodes in its lists will be updated. The crucial point is that each node only fails once, thus the running time is reduced significantly. The algorithm with the adaptive cascading potential is described in Algorithm 2.

Time complexity. Since each node has impact on at most n nodes, the total size of all FI and LI lists are at most n^2. The number of updates is bounded by the total size of FI and LI lists. Therefore the total running time is $O(nm + n^2 + kn)$.

4.4. Fully adaptive cascading potential algorithm

On the line of cascading potential based algorithms, we continue to improve the solution's quality by spending more time to calibrate the cascading potential of nodes. After a node u is attacked, the network state is changed with new failed nodes and load updates; and this may decrease (as discussed in the proceeding part) or increase the impact of a node. The failure of u adds load to many nodes and makes them more vulnerable. Although the impact of a remained node v is deducted by the impact on failed nodes, it can still increase since other nodes are easier to be failed. We can fully update the cascading potential of each node as follows. After selecting a new node, we simulate the cascading failure triggered by it and obtain a new graph of remaining nodes. In this graph, the load of a node is the load when the cascading process stops. We then evaluate the cascading potential of all nodes in the updated graph and select one with the highest value. We present the algorithm in Algorithm 3.

Time complexity. We need to compute the cascading potential of all nodes to select a new one with time $O(nm)$. Thus the total running time is $O(kmn)$. However, the algorithm may run much faster than the worst case time since the size of the updated graph decreases when a new node is selected.

5. Cooperating attack algorithm

The key to connect the impact of multiple nodes in the above algorithms is the load impact which is a connection link when the network is robust. In this case where nodes have high failure tolerance, i.e., the gap between the capacity and load is big, the failure impact of each node is small. Thus, nodes with high load impact tend to be selected. If the load of these nodes are scattered to many nodes, they are not linked together to make other nodes fail. As a consequence, there are a large number of nodes whose loads are increased, but there are only a few failed nodes. We incidentally try to maximize the total load impact instead of the failure impact – the objective function. We need a better strategy which builds a strong connection between selected nodes to increase the number of failed nodes. To fulfill this goal, the new strategy should satisfy following features:

Algorithm 2 Partially Adaptive Cascading Potential (PACP) algorithm.

Input: A network $G = (V, E)$, an integer k
Output: A set S of k attacked nodes.

```
for each v ∈ V do
    Initialize Fl[v] ← ∅, Ll[v] ← ∅
end for
for each u ∈ V do
    Compute C(u)
    for each v ∈ F({u}) do
        Fl[v] ← Fl[v] ∪ {u}
    end for
    for each v: ΔL_u(v) > 0 and v ∉ F({u}) do
        Ll[v][u] ← ΔL_u(v) / Σ_{z∈V−F({u})} (C(z)−L(z))
    end for
end for
Initialize S ← ∅
for each u ∈ V do
    Mark[u] ← False
end for
for i = 1 to k do
    u ← arg max_{v∈V\F+(S)} {C(v)}
    S ← S ∪ {u}
    for each v ∈ F(S) do
        if Mark[v] == False then
            Mark[v] ← True
            for each u ∈ Fl[v] do
                C(u) ← C(u) − 1/|V|
            end for
            for each u ∈ Ll[v] do
                C(u) ← C(u) − CL[v][u]
            end for
        end if
    end for
end for
Return S
```

Algorithm 3 Fully Adaptive Centrality.

Input: A network $G = (V, E)$ and an integer k.
Output: A set S of k attacked nodes.

```
Initialize S ← ∅
for i = 1 to k do
    Compute the cascading potential of all nodes in G
    Select u as the node with highest cascading potential
    S ← S ∪ {u}
    Update node loads and remove all failed nodes in G with the failure of u
end for
Return S
```

- The redistributed load of selected nodes should be concentrated on certain nodes to fail them. If early selected nodes redistributed load to a set of nodes, then later selected nodes should also redistribute load to this set. It is said that selected nodes are cooperating in redistributing load to make more nodes to fail.
- Selected nodes should cooperate to make high load nodes fail. The failure of high load nodes can expand the cascading failures further. However, if high load node preference reduces the number of failed nodes, the new strategy should not blindly favoring to fail high load nodes.

We design a new evaluation function, the efficiency, of nodes with properties that tailor the selection process to embrace both desired features. Firstly, we give higher evaluation to nodes which redistributes its load to load-increased nodes. If the failure of u pushes an additional load $\Delta L_u(v)$ on v, then the impact of u on v is defined by:

$$\gamma(u, v) = \frac{\Delta L_u(v)}{C(v) - L(v)}$$

when $\Delta L_u(v) + L(v) \leq C(v)$. Since it requires $C(v) - L(v)$ additional load to make v fail, we can interpret that u makes a fraction $\frac{\Delta L_u(v)}{C(v)-L(v)}$ of v fail.

The new impact function implies that if the more load a node has already received, the more impact it received under the same additional load. On the other hand, the evaluation of u is higher if the loads of its neighbors are increased. This implication is stated in Proposition 1.

Proposition 1. *For any node v at two points of time, if v receives more load at the second time point, i.e., $L_2(v) > L_1(v)$, then the impact of other node u with the same redistributed load ΔL is higher at the second time point: $\gamma_2(u, v) \geq \gamma_1(u, v)$.*

Proof. We have:

$$\gamma_2(u, v) = \frac{\Delta L}{C(u) - L_2(u)} > \frac{\Delta L}{C(u) - L_1(u)} = \gamma_1(u, v) \qquad \square$$

We assume that u redistributes the same load on v in the Proposition 1, i.e. the load of u is the same at two points of time. In fact, the load of u may increase due to the selection of previous nodes, thus the evaluation of u increases even more at the second point of time.

To fulfill the second feature, we assign higher values to high load nodes which are impacted. The value of a node with load L is:

$$\sigma(L) = \frac{e^L}{1 + e^L}$$

The function $\sigma(L)$ is monotone increasing and in the range $0.5 \leq \sigma(L) < 1$. The monotone increasing of the function shows the preference toward high load nodes. Recall that the main goal is to increase the number of failed nodes, so even nodes with the lowest load have the value at least half of the highest value nodes.

Next, we will define the efficiency of selecting u via the impact on v. Intuitively, u makes $\gamma(u, v)$ fraction of v fail and v has value of $\sigma(L(v))$, thus the efficiency of u represented on v is:

$$\lambda(u, v) = \gamma(u, v)\sigma(L(v))$$

Finally, we obviously should take into account the number of failed nodes when evaluating node u. The overall efficiency of u is the total of the number of failed and the efficiency on unfailed nodes:

$$\lambda(u) = |F(\{u\})| + \sum_{v \in V \setminus F(\{u\})} \lambda(u, v)$$

The efficiency evaluation shows several notable properties which serves our design goal as followings:

Increase the number of failed nodes first. If u makes z fail and has efficiency $\lambda(u, v)$ on the unfailed node v, then the contribution of z to the overall efficiency of u is always higher than v since $1 \geq \gamma(u, v)\sigma(L(v))$.

Avoid redistributing load to impossible-to-fail nodes. If node v needs too much additional load before failing, it will be ignored in efficiency evaluation of nodes as stated in the Proposition 2.

Proposition 2. *Given two nodes u and v with fixed load $L(v)$, the efficiency of u on v is monotone decreasing and goes to 0 when the capacity $C(v)$ of v increases and goes to infinity.*

Proof. It is easy to see that $\gamma(u, v)$ is monotone decreasing and goes to 0 when $C(v)$ increases and goes to infinity. In addition, $\sigma(L(u))$ is a constant, so the efficiency $\lambda(u, v) = \gamma(u, v)\sigma(L(v))$ decreases and goes to 0. $\square$

Not favoring high load nodes with all cost. We consider the case the capacity is linear to the load, a common setting in the reality to guarantee the safety of nodes. In this case, even the load of node v is extremely large, it is still ignored as shown in Proposition 3.

Proposition 3. *Suppose that the capacity $C(v)$ is linear to the load $C(v) = T * L(v)$ with constant factor T. Then, the efficiency of any node u on v goes to 0 when the load $L(v)$ goes to infinity.*

Proof. We have:

$$\begin{aligned} \gamma(u, v)\sigma(L(u)) &= \frac{\Delta L_u(v)}{C(v) - L(v)} \frac{e^{L(v)}}{1 + e^{L(v)}} \\ &< \frac{\Delta L_u(v)}{(T - 1)L(v)} \end{aligned}$$

The function goes to 0 when $L(u)$ goes to infinity. $\square$

Based on the efficiency evaluation, we propose the Cooperating Attack (CA) algorithm with the same manner of the Fully Adaptive Cascading Potential algorithm. The algorithm also selects nodes one by one. After updating the state of the network, the node with the highest efficiency is selected. The whole algorithm is described in Algorithm 4.

Time complexity. Similarly to the Fully Adaptive Cascading Potential algorithm, the total running time is $O(kmn)$.

Algorithm 4 Cooperating Attack (CA) algorithm.

Input: A network $G = (V, E)$ and an integer k.
Output: A set S of k seed nodes.
Initialize $S \leftarrow \emptyset$
for $i = 1$ to k **do**
  Evaluate the efficiency of all nodes in G
  Select u as the node with the highest efficiency
  $S \leftarrow S \cup \{u\}$
  Update node loads and remove all failed nodes G with the failure of u
end for
Return S

6. Experimental evaluation

In this section, we demonstrate the experimental results on both synthesized and real power networks. We first test the performance of the proposed algorithms in the comparison with current attacking strategies in the literature [7,6]. These strategies sort nodes based on some criterion and select top k nodes as attacked nodes. The sorting criteria are:

- Highest load (HL).
- Lowest load (LL).
- Highest percentage of failure (POF). The percentage of failure of a node u is the fraction of nodes is failed when u fails.
- Highest risk if failure (RIF). RIF of a node u is the ration between its load and the total load of its neighbor nodes.

We omit the required redundancy of Wang et al. [7] since it provides the same order of nodes as RIF. After that, we evaluate the robustness of networks under different failure tolerance schemes to identify the suitable network design considering the effect of cascading failures. Finally, we test the effect of the failure tolerance value and the topology on the vulnerability.

6.1. Datasets

Real network. We use the Western North American (WNA) power grid network [14] with 4941 stations and 6594 transmission lines to run experiments. However, the dataset is lacking of load and capacity information of nodes, thus we use the similar method in [11] to assign the load and capacity for each node. The initial load of node u is given by $L(u) = d(u)^{\beta}$, where $d(u)$ is the total of incoming and outgoing degrees of u and β is a constant parameter. This assignment method is reasonable as the load of a node is shown to have high correlation with its degree [15]. In all experiments, the default value of β is 1, unless otherwise mentioned. Node capacities are assigned based on three different schemes:

Normal networks. In normal networks, the capacity $C(u)$ of each node u is proportional to its initial load $L(u)$:

$$C(u) = T * L(u)$$

where T is a constant representing the *system tolerance*. The larger T is, the more robust the network is under the cascading failure.

Safe networks. In safe networks, the node capacities are assigned in two phases. First, the capacity $C(u)$ of each node u is scaled as the normal network, i.e.:

$$C(u) = T * L(u)$$

Then, capacities of all nodes are raised to satisfy the $N-1$ failure tolerance criterion in which the failure of any node will cause no additional failed nodes. It means that any node u will not fail when it receives the redistributed load from any of its neighbor. The capacity of u will be:

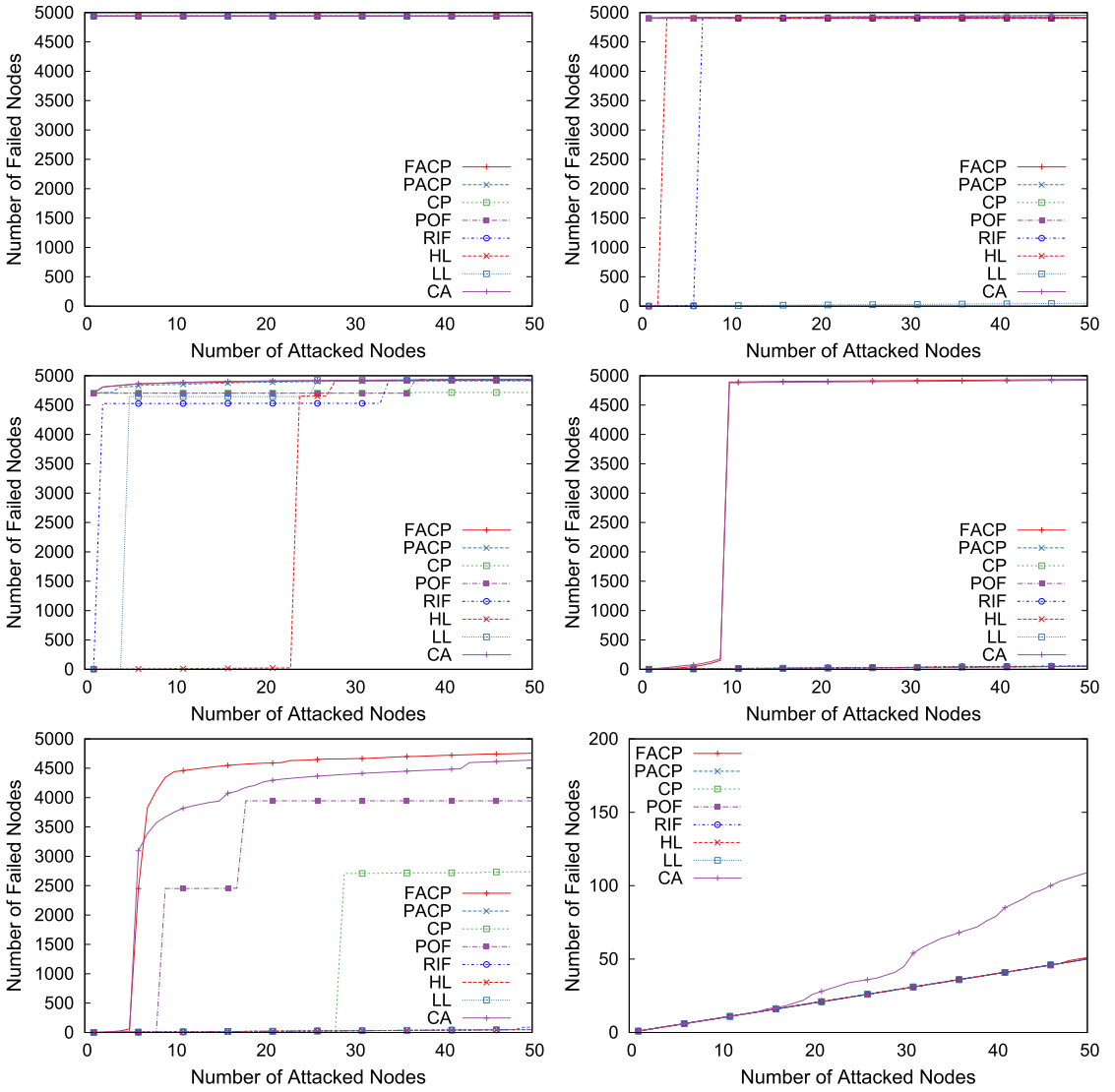
$$C(u) = \max\{C(u), \max_{v \in N_u^-}\{L(u) + L(v)\frac{w(v,u)}{\sum_{z \in N_v^+} w(v,z)}\}\}$$

Scaled safe networks. In contrast to safe networks, scaled safe networks are formed by raising the node capacities to satisfy $N-1$ failure tolerance criterion first, then be scaled up later. In particular, the network is made safe by assigning the capacity of each node u as:

$$C(u) = \max_{v \in N_u^-}\{L(u) + L(v)\frac{w(v,u)}{\sum_{z \in N_v^+} w(v,z)}\}$$

Then the capacity of u is scaled up to $C(u) = T * C(u)$.

Synthesized networks. We also run the experiments on synthesized networks generated by Erdos–Renyi random network model [16]. Each network has 5000 nodes with the average degree of 4. The other parameters of the network are generated similarly to the above schemes.



(a) WNA network (b) Synthesized network

Fig. 4. Vulnerability of networks under the normal setting.

6.2. The performance of different algorithms

We first evaluate the performance of all algorithms with various network settings and system tolerance values. We carry experiments with different system tolerance values: low value $T = 1.2$, medium value $T = 1.6$, and high value $T = 2.0$. The comparison is shown in Figs. 4, 5, and 6.

Overall, the FACP and Cooperating algorithms are competitive with each other and outperform other algorithms. When the network is easy to attack, the performance of the FACP algorithm is better but is not too far away from that of the CA algorithm. On WNA (synthesized) network with normal and safe settings and $T \leq 1.6$, the number of failed nodes provided by two algorithms is almost the same. On WNA network with normal and safe settings with $T = 2$, the FACP algorithm starts to surpass CA algorithm when $k = 5$, achieves peak when $k = 10$ with 18% better. When $k > 10$, the number of failed nodes is close to the total number of nodes, thus the gap between two algorithms is reduced. Under the scaled safe setting, the setting with highest failure tolerance, the CA algorithm takes advantage over the FACP algorithm when nodes are attacked. In WNA network with $T = 2$, the CA algorithm makes a considerable fraction of nodes fail when the number of attacked nodes exceeds 11 while the FACP algorithm almost makes no effect until $k = 19$. However, the number of failed nodes is the same when $k > 30$. The reason is that when a large number of nodes are failed, the load of remaining nodes are much higher which is the favor scenario for the FACP algorithm. In the synthesized network with the scaled safe setting, $T = 1.6$, it is more difficult to attack the network. Since the CA algorithm aims to increase the number of failed nodes as soon as

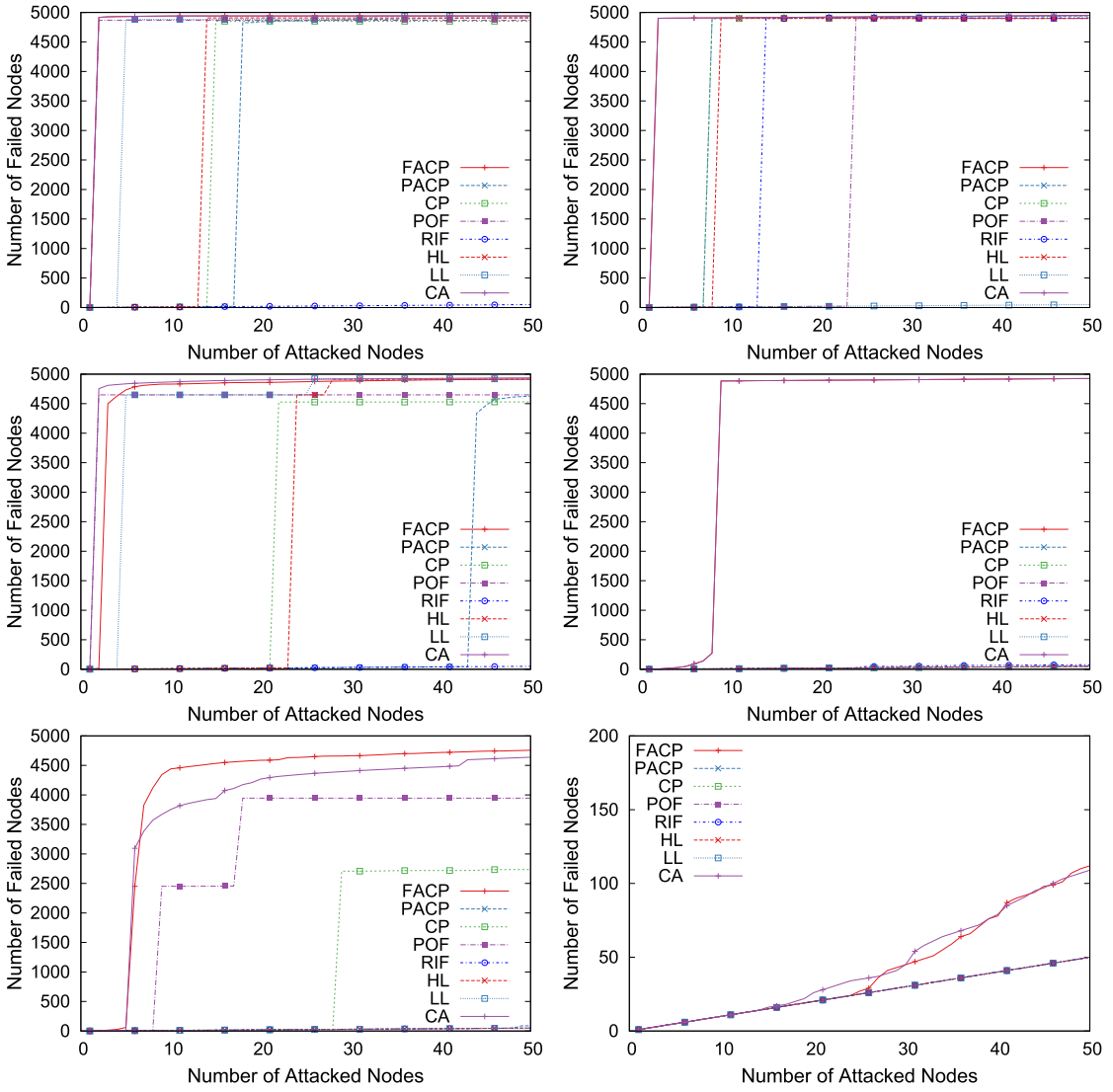
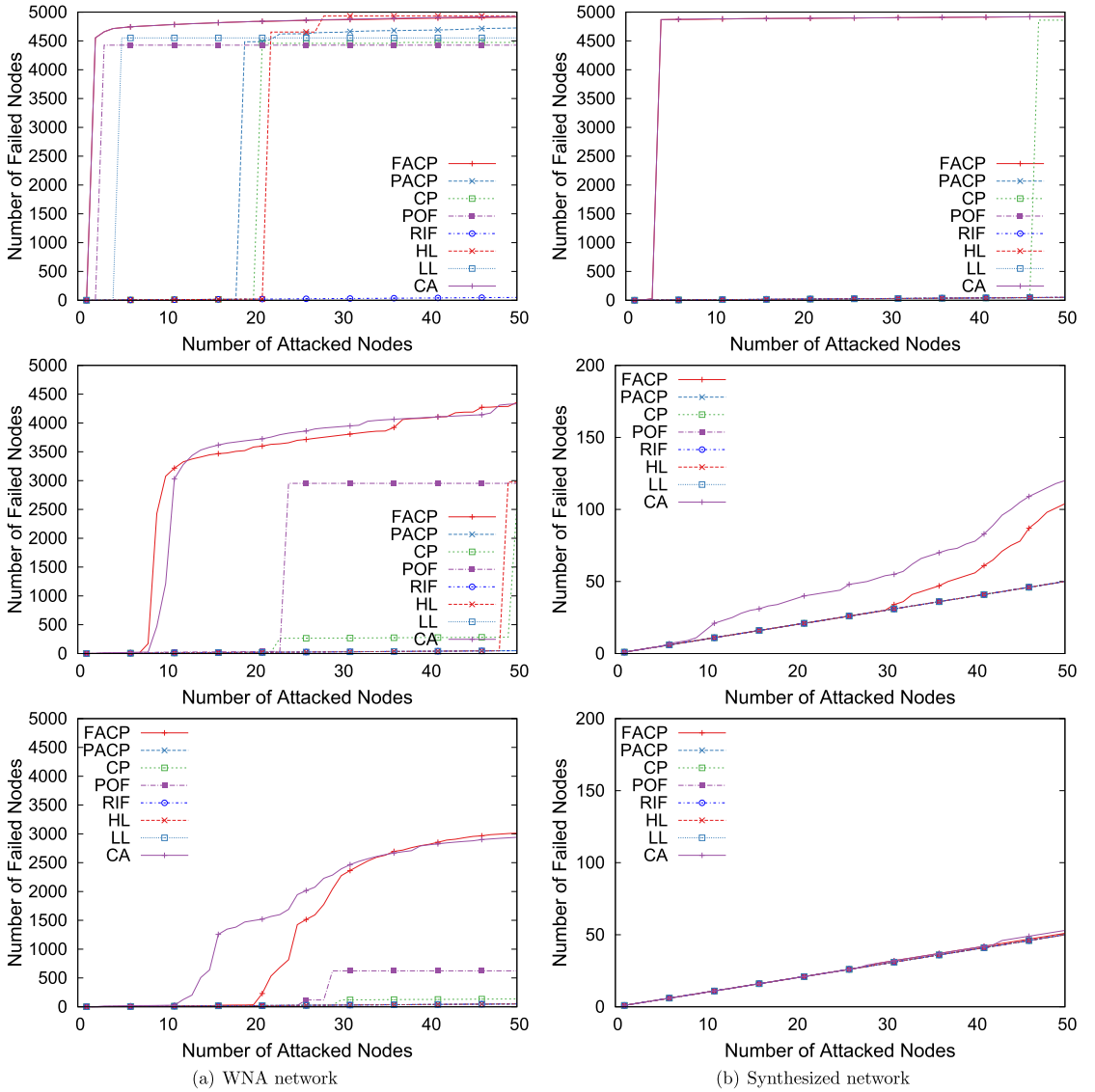


Fig. 5. Vulnerability of networks under the safety setting.

possible, it produces better solution with about 20% better than the FACP algorithm. We can conclude that the better choice is the FACP algorithm if the network is vulnerable and the CA algorithm if the network is more robust.

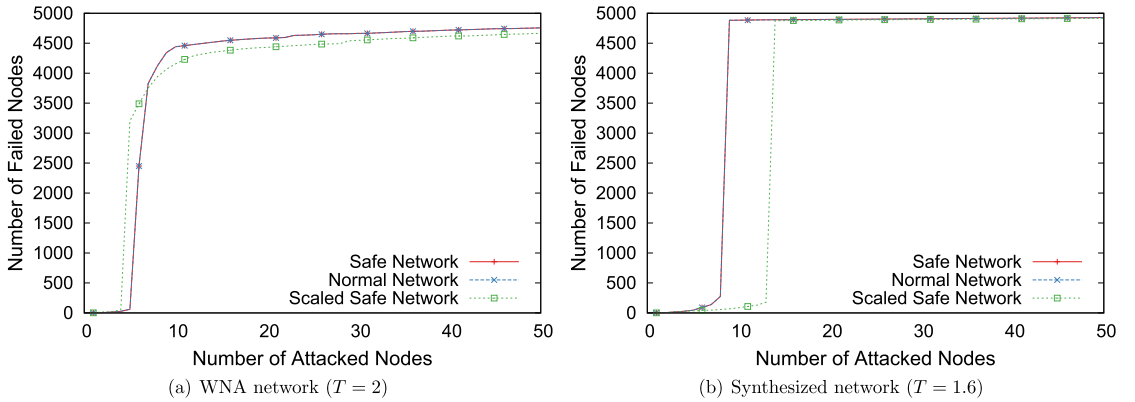
6.3. Network robustness under different settings

From Figs. 4, 5, and 6, we observe that scaled safe networks are more robust than safe networks and safe networks are stronger than normal networks. However, there is a bias since with the same value of T, the same node in the scaled safe network has higher capacity than one in the safe and normal networks. Thus, we set up the experiment such that all types of networks have the same total capacity to verify the network robustness of each setting. We first generate the safe network, then we choose a suitable T value to generate normal and scaled safe networks such that the total capacity is the same. Note that the normal network has the larger value of T while the scaled safe network has the smaller value. Fig. 7 shows that the scaled safe network is the most robust one. Normal and safe networks have the same robustness although the safe network can avoid the cascading failure after the first attack. Under the safe setting, a node can embrace the impact created by the failure of any neighbor (with the initial load) which is not true if its neighbors receive additional load. The failure of first nodes redistributed a certain load to other nodes which is enough to disqualify the $N-1$ safety setting. Hence, the failure of later nodes triggers the same cascading failures in both normal and safe networks. Therefore, it recommends us to use the scaled safe setting to improve the robustness of the power grid instead of the $N-1$ safety criterion.



(a) WNA network (b) Synthesized network

Fig. 6. Vulnerability of WSN network under the scaled safety setting.



(a) WNA network ($T = 2$) (b) Synthesized network ($T = 1.6$)

Fig. 7. Network robustness with different failure tolerance settings.

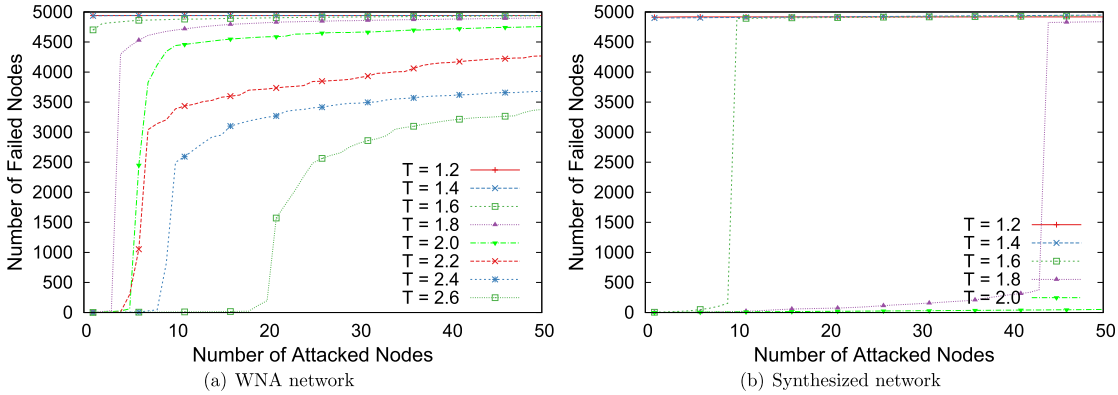


Fig. 8. Network robustness with different system tolerance values.

6.4. The impact of system tolerance value

We next evaluate the impact of the system tolerance value on the network robustness. Fig. 8 shows the results when the node capacities are set up based on the normal setting. Both WNA and synthesized networks show a noticeable property of the cascading attack: the number of failed nodes significantly jumps up when the number of attacked nodes reaches to a certain value called the disruptor threshold. Each system tolerance value has a corresponding disruptor threshold at which a considerable fraction of network is failed. The higher T is, the higher the disruptor threshold is. In WNA network, the disruptor threshold is increased gradually with T: from 3 nodes with $T = 1.8$ to 20 nodes with $T = 2.6$. In the synthesized network, the disruptor threshold increases faster from 1 nodes with $T = 1.4$ to 9 nodes with $T = 1.6$ and to 42 nodes with $T = 1.8$. The reason is that the synthesized network has higher density, so the load of failed nodes is distributed to many nodes. When T increases, each node can receive more load without failing, hence the cascading process is quickly stopped.

7. Conclusions

In this paper, we studied the critical node detection problem under the LR-model. Based on the new node evaluation function, cascading potential, which considers the cascading effect when evaluating the importance of nodes in networks, we develop various algorithms to identify critical nodes: one is purely based on the cascading potential, one is based on partially adaptive cascading potential, and one is based on fully adaptive cascading potential. Among them, the fully adaptive algorithm continuously updates the cascading potential of nodes and select the best one, hence achieves the highest performance among the three. However, this algorithm suffers when the network is highly tolerant to the cascading failures. We propose the Cooperating Attack algorithm which cooperates selected nodes to take down protected nodes with high capacity. The performance guarantee of the Cooperating Attack algorithm is supported by both theoretical and experimental results.

In addition, we use proposed algorithms to study the vulnerability of different safety settings. We find that networks with low density are extremely vulnerable under the cascading failures. In this kind of networks, the load of a failed node is redistributed to a small number of neighbors and can fails them easily. On the other hand, the load is shred to smaller portions in networks with high density. We also discover that even with networks of the same topology, node load, and total capacity, the network safety depends a lot on the distribution of the protection cost (the gap between the capacity and the load). In the future work, we would like to study defending strategies to limit the impact the cascading failures.

References

[1] Adilson E. Motter, Ying-Cheng Lai, Cascade-based attacks on complex networks, Phys. Rev. E 66 (6) (2002) 065102.
[2] R. Albert, H. Jeong, A.L. Barabasi, Error and attack tolerance of complex networks, Nature 406 (6794) (2000) 378–382.
[3] Paolo Crucitti, Vito Latora, Massimo Marchiori, Andrea Rapisarda, Error and attack tolerance of complex networks, Phys. A 340 (1) (2004) 388–394.
[4] Paul Hines, Seth Blumsack, E. Cotilla Sanchez, Clayton Barrows, The topological and electrical structure of power grids, in: 43rd Hawaii International Conference on System Sciences, HICSS, 2010, IEEE, 2010, pp. 1–10.
[5] Marti Rosas-Casals, Sergi Valverde, Ricard V. Solé, Topological vulnerability of the European power grid under errors and attacks, Internat. J. Bifur. Chaos 17 (07) (2007) 2465–2475.
[6] Jian-Wei Wang, Li-Li Rong, Cascade-based attack vulnerability on the us power grid, Saf. Sci. 47 (10) (2009) 1332–1336.
[7] Wenkai Wang, Qiao Cai, Yan Sun, Haibo He, Risk-aware attacks and catastrophic cascading failures in U.S. power grid, in: Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE, IEEE, 2011, pp. 1–6.
[8] Liang Zhao, Kwangho Park, Ying-Cheng Lai, Nong Ye, Tolerance of scale-free networks against attack-induced cascades, Phys. Rev. E 72 (2) (2005) 025104.
[9] Reka Kinney, Paolo Crucitti, Reka Albert, Vito Latora, Modeling cascading failures in the North American power grid, Eur. Phys. J. B 46 (1) (2005) 101–107.
[10] Réka Albert, István Albert, Gary L. Nakarado, Structural vulnerability of the North American power grid, Phys. Rev. E 69 (2) (2004) 025103.

[11] Zhi-Xi Wu, Gang Peng, Wen-Xu Wang, Sammy Chan, Eric Wing-Ming Wong, Cascading failure spreading on weighted heterogeneous networks, J. Stat. Mech. Theory Exp. 2008 (05) (2008) P05013.
[12] Vijay V. Vazirani, Approximation Algorithms, Springer-Verlag New York, Inc., New York, NY, USA, 2001.
[13] Miroslav Chlebík, Janka Chlebíková, Complexity of approximating bounded variants of optimization problems, Theoret. Comput. Sci. 354 (3) (2006) 320–338.
[14] Duncan J. Watts, Steven H. Strogatz, Collective dynamics of 'small-world' networks, Nature 393 (6684) (1998) 440–442.
[15] Liang Zhao, Kwangho Park, Ying-Cheng Lai, Attack vulnerability of scale-free networks due to cascading breakdown, Phys. Rev. E 70 (3) (2004) 035101.
[16] Paul Erdos, Alfréd Rényi, On the evolution of random graphs, Bull. Inst. Int. Stat. 38 (4) (1961) 343–347.