EDGE REWIRING GOES NEURAL: BOOSTING NETWORK RESILIENCE VIA POLICY GRADIENT

Anonymous authors
Paper under double-blind review

ABSTRACT

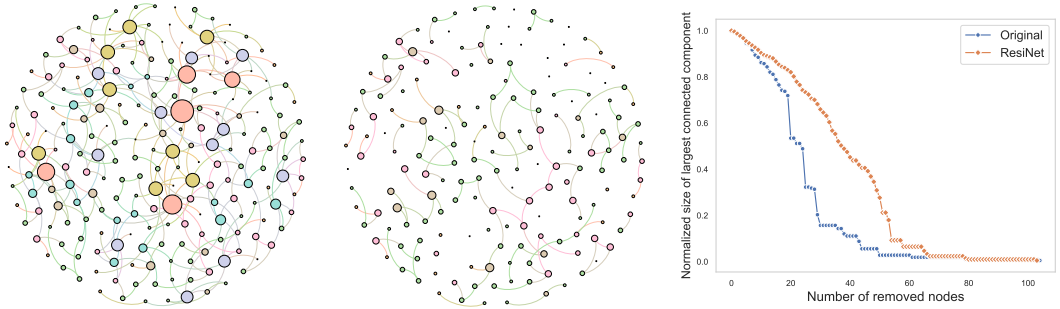
Improving the resilience of a network protects the system from natural disasters and malicious attacks. This is typically achieved by introducing new edges, which however may reach beyond the maximum number of connections a node could sustain. Many studies then resort to the degree-preserving operation of rewiring, which swaps existing edges AC, BD to new edges AB, CD. A significant line of studies focuses on this technique for theoretical and practical results while leaving three limitations: network utility loss, local optimality, and transductivity. In this paper, we propose **ResiNet**, a reinforcement learning (RL)-based framework to discover **Resi**lient **Net**work topologies against various disasters and attacks. ResiNet is objective agnostic which allows the utility to be balanced by incorporating it into the objective function. The local optimality, typically seen in greedy algorithms, is addressed by casting the cumulative resilience gain into a sequential decision process of step-wise rewiring. The transductivity, which refers to the necessity to run a computationally intensive optimization for each input graph, is lifted by our variant of RL with auto-regressive permutation-invariant variable action space. ResiNet is armed by our technical innovation, **Fil**t**r**ation **e**nhanced GNN (**FireGNN**), which distinguishes graphs with minor differences. It is thus possible for ResiNet to capture local structure changes and adapt its decision among consecutive graphs, which is known to be infeasible for GNN. Extensive experiments demonstrate that with a small number of rewiring operations, ResiNet achieves a near-optimal resilience gain on multiple graphs while balancing the utility, with a large margin compared to existing approaches.

1 INTRODUCTION

Modern infrastructure systems, such as computer routing and electric power networks, are vulnerable to natural disasters and malicious attacks. Consider the scenario where the abnormality of one power supply station causes other power supply stations to overload, which cascades more power supply to fail, resulting in a region-wide power outage. Figure 1 visualizes this scenario that the failures of a dozen of nodes could jeopardize the connectivity and utility of the EU power network with 217 nodes. The ability for a system to defend itself from such failures and attacks is characterized by the *network resilience*. A resilient network should continue to function and maintain an acceptable level of utility when part of the network fails.

A network becomes more resilient if some connections could backup the others. It is seemingly straightforward to achieve resilience by adding redundant edges. However, it will not be practically feasible as the nodes are usually already running at their full capacity. For example, a power supply facility can only support a certain number of connections and cannot afford additional loads. In such a regime, degree-preserving operations, such as edge rewiring, are desired (Schneider et al., 2011; Rong & Liu, 2018; Yazıcıoğlu et al., 2015; Zhou & Liu, 2014). Let $G = (V, E)$ be a graph. An edge rewiring operation alters the graph structure by removing AC and BD and adding AB and CD, where $AC, BD \in E$ and $AB, CD, AD, BC \notin E$.[1] Edge rewiring is empirically effective to maximize network resilience and becomes the prevailing atomic operation in the literature. Despite this, methods based on edge rewiring share several limitations: utility loss, local optimality, and transduction.

[1] A few works have a different definition for the edge rewiring operation.



(a) Original EU network (b) EU network after attacks (c) Remaining LCC size under attack

Figure 1: The EU power network under the adaptive degree-based attack which removes the most critical node recursively with (a) original EU network with 217 nodes, (b) remaining EU network after a series of attacks on 40 nodes, and (c) the change of the normalized size of the largest connected component (LCC). The node size is proportional to its degree and the node color is given by DBSCAN (Ester et al., 1996).

- Network utility loss. Improving network resilience aims to protect the network utility from degrading under attack and make the network continue functioning. Optimizing network resilience without considering the network utility could jeopardize the functioning without the presence of an attack.
- Local optimality. Combinatorially choosing the edges to rewire to optimize the resilience is proved to be NP-hard (Mosk-Aoyama, 2008). Previous studies predominantly seek approximate optimality through greedy-like algorithms, which yields local optimality in practice.
- Transductivity. Traditional resilience optimization methods are transductive since they search the resilience topology on a given graph. This search process has to be executed for every graph and does not transfer between graphs, even if the graphs are only up to a minor structure difference.

To tackle the challenges above, in this work, we present ResiNet, the first objective-agnostic learning-based method for inductively discovering resilient network topologies. Frist, by formulating the cumulative gain of the objective into RL, ResiNet is agnostic to arbitrary objective function, which makes resilience optimization practical in real applications by incorporating the utility into the resilience. The nature of delayed gratification in RL also allows the agent to bypass local optimums caused by step-wise optimal actions. Second, the fact that graphs have different sizes and arbitrary node permutations makes it challenging to solve transductivity with deep learning. In this scenario, nodes can no longer be represented by their ID since it is computationally infeasible to train a model to handle all isomorphic graphs. Using an attention-based mechanism (Vinyals et al., 2015) to identify edges and an auto-regressive space (Trivedi et al., 2020) to formulate edge selection, an agent trained by ResiNet works on a wide range of graphs with different sizes and permutations.

The implementation of ResiNet is armed with Filtration enhanced GNN (FireGNN), our technical innovation, to solve the following dilemma: If the representation power of the GNN is too weak (e.g., the regular GNN), then it could not distinguish graphs with minor differences and will choose similar actions in consecutive steps of RL. This causes the process to alternate between two graphs, forming an infinite loop. If the representation power is too strong (e.g., SMP, IDGNN), it tends to give similar representations to connected vertices, known as the *oversmoothing phenomenon* (Vignac et al., 2020; You et al., 2021). It is then difficult for the agent to distinguish vertices in a connected component, causing large randomness in the output of the policy. FireGNN creates a series of subgraphs (the filtration) by successively removing nodes from the graph, and aggregates the node representations from the subgraphs. It makes the GNN powerful when representing graphs while avoiding oversmoothing as the node information are adequately acquired from a multi-step filtration. This technical innovation is inspired by persistent homology and the approximation of the persistence diagram (Edelsbrunner & Harer, 2008; Aktas et al., 2019; Hofer et al., 2020).

The main contributions of this paper are summarized as follows:

1) We propose ResiNet, a data-driven framework to boost network resilience in a degree-preserving way with moderate loss of the network utility by forming resilience optimization into a objective-agnostic sequential decision process of edge rewiring. Extensive experiments show that with a small number of rewiring operations, ResiNet achieves a near-optimal resilience gain on multiple graphs while balancing network utilities. Existing approaches are outperformed by a large margin.
2) ResiNet automatically learns various strategies for defending against different attack strategies, balancing multiple resilience and utility metrics, and optimizing different graphs. Furthermore, our framework can be easily extended to resilience optimization on weighted graphs by including the weights as an edge feature. The framework can also incorporate a budget for adding new edges, such that both edge rewiring and edge adding are involved.
3) ResiNet is the first to improve network resilience in an inductive way, with the specialized auto-regressive permutation-invariant size-variable policy network. Once an agent is trained, it works on a wide range of graphs. The policy network is general enough to be extended to other problems in constrained graph generation.
4) FireGNN, our technical innovation, balances the expressive power and the oversmoothing issue through the graph filtration augmentation. FireGNN could distinguish very similar graphs and distinguish connected vertices at the same time, making it a viable choice for RL algorithms. It is general enough to be applied to a broader range of tasks, such as link prediction and link recommendation.

2 Related works

Network resilience and utility Network utility refers to the system's quality to provide a certain service, for example transmitting electricity in power networks and transmitting packages in routing networks. A popular metric for network utility is the network efficiency (Latora & Marchiori, 2003; Boccaletti et al., 2006). Network resilience measures the ability of preventing the utility loss under failures and attacks. Our goal is to improve both network resilience and utility by network structure manipulations.

In many previous works, despite that the resilience could be improved, the utility may dramatically drop at the same time (Li et al., 2019; Carchiolo et al., 2019; Wang et al., 2014; Schneider et al., 2011; Chan & Akoglu, 2016; Buesser et al., 2011). This is contradicting with the idea behind improving network resilience and will be infeasible in real-world applications.

Network attacks and defenses The problem of attacking a network is characterized by the *target resilience measurement* and the *manipulation type*. Depending on the application, network resilience is defined as a corresponding measure to quantify the network functionality, such as graph connectivity (Grassia et al., 2021; Fan et al., 2020). For particular networks (e.g., scale-free networks), attacks appear in destroying critical nodes and critical connections (Grassia et al., 2021; Fan et al., 2020; Zhao et al., 2021; Zhang et al., 2017; Medya et al., 2020). Heuristic and learning-based attack methods (Holme et al., 2002; Iyer et al., 2013; Grassia et al., 2021; Fan et al., 2020) have been proposed to target the critical subset of the network.

To defend against potential attacks, various defense strategies have been proposed to protect the network functionality from crashing and preserve some of its topologies. The commonly used defense manipulations include adding additional edges (Li et al., 2019; Carchiolo et al., 2019), protecting vulnerable edges (Wang et al., 2014) and rewiring two edges (Schneider et al., 2011; Chan & Akoglu, 2016; Buesser et al., 2011). Among these manipulations, edge rewiring fits well to real-world applications as it induces less functionality changes (e.g. degree preserving) to the original network and does not impose additional loads to the vertices. Network defense is a more challenging problem if rewiring is the only allowed operation (Schneider et al., 2011; Rong & Liu, 2018; Yazıcıoğlu et al., 2015; Zhou & Liu, 2014). It is also important to point out that there does not exist a "silver graph" that is resilient against all kinds of attacks. For example, the random scale-free network with power-law degree distribution is resilient to random attacks but vulnerable to targeted attacks on nodes or edges (Albert et al., 2000). Therefore, a unified framework to defend against different kinds of attacks is desired.

Positioning of this work In this paper, we use only the edge rewiring operation to defend against degree-based and centrality-based attacks while preserving the network utility.

Extended related works The related works on *GNNs for combinatorial optimization problems*, *multi-views graph augmentation for GNNs*, and *deep graph generation models* are deferred to Appendix A.

3 PROPOSED APPROACH: RESINET

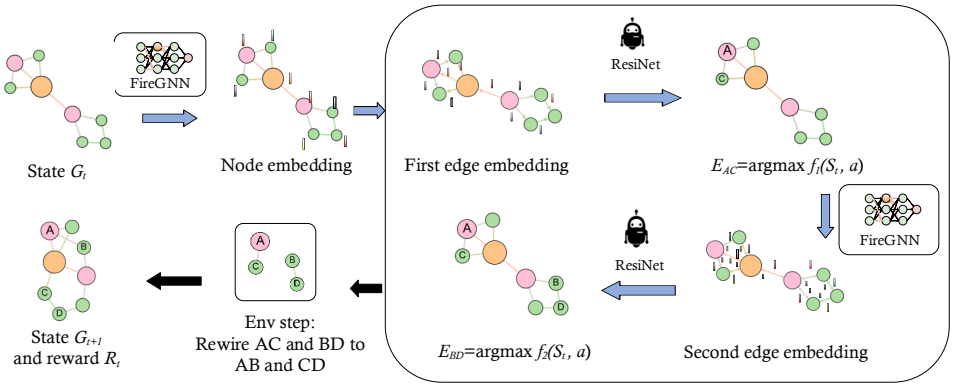


Figure 2: Overview of the policy architecture of ResiNet to select two rewiring edges.

In this section, we first present the problem formulation of maximizing network resilience while preserving utility performance. We then discuss the graph resilience-aware environment design and give the graph policy network which guides the process of edge rewiring. The problem of boosting network resilience is formulated as a reinforcement learning task by iteratively rewiring the edges.

3.1 PROBLEM DEFINITION

An undirected graph is defined as $G = (V, E)$, where $V = \{1, 2, \ldots, N\}$ is the set of N nodes, E is the set of M edges, $A \in \{0, 1\}^{N \times N}$ is the adjacency matrix, and $F \in \mathbb{R}^{N \times d}$ is the d-dimensional node feature matrix.[2] The degree of a node is defined as $d_i = \sum_{i=1}^{N} A_{ij}$, and a node with degree 0 is called an isolated node.

Let $\mathbb{G}_G$ denote the set of graphs with the same node degrees as G. Given the network resilience function $\mathcal{R}(G)$ and the utility function $\mathcal{U}(G)$, the objective of boosting the resilience of G is to find a target graph $G^\star \in \mathbb{G}_G$, which maximizes the network resilience while preserving the network utility. Formally, the problem of maximizing network resilience is formulated as

$$G^\star = \underset{G' \in \mathbb{G}_G}{\arg\max}\ \alpha \cdot \mathcal{R}(G') + (1 - \alpha) \cdot \mathcal{U}(G'),$$

where $\alpha \in \mathbb{R}$ is the scalar weight that balances the resilience and the utility.

Two examples of resilience functions, including the graph connectivity-based measurement (Schneider et al., 2011) and the spectrum-based measurement (e.g., adjacency matrix spectrum and Laplacian matrix spectrum), and two examples of utility functions, including global efficiency and local efficiency, are given in Appendix B. Our formulation could generalize to other definitions of resilience and utility.

3.2 ROBUST GRAPH GENERATION VIA EDGE REWIRING AS MARKOV DECISION PROCESS

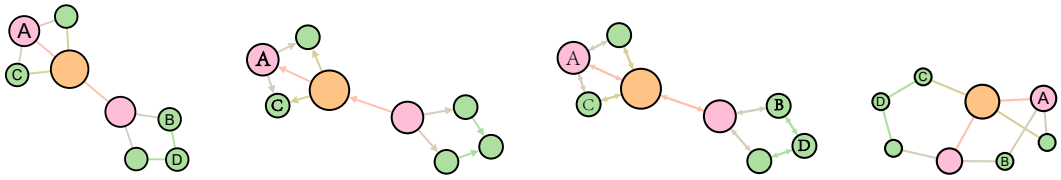
To satisfy the constraint of preserving the node degree, the resilience optimization of a given graph is based on edge rewiring. We formulate this process into the MDP framework. The Markov property

[2]Without loss of generality, unweighted graphs are considered. Our framework can be extended to weighted graphs with the weight as an additional edge feature.

denotes the fact that the graph obtained at time step $t+1$ relies only on the graph at time step t and the rewire operation. As is illustrated in Figure 2, the environment performs the resilience optimization in an auto-regressive step-wise way through a sequence of edge rewiring actions guided by ResiNet. Given an input graph, the agent first decides whether to terminate if or not. If it chooses not to terminate, it selects one edge from the graph to remove, receives the very edge it just selected as the regression signal, and then selects another edge to remove. Four nodes of these two removed edges are re-combined, forming two new edges to be added to the graph. The optimization process repeats until the agent decides to terminate. The details of the design of the state, the action, the transition dynamics, and the reward are presented as follows.

State The state of the environment is formulated as $S_t = G_t$, where G_t is the current input graph at time step t. The initialization of the node feature matrix is described in Appendix C.3. The state is fully observable.

Action ResiNet is equipped with a node permutation-invariant, variable-dimensional action space. Given a graph G_t, the action a_t is to select two edges and the rewiring order. The agent first chooses an edge $e_1 = AC$ and a direction $A \to C$. Then conditioning on the state, e_1, and the direction the agents chooses an edge $e_2 = BD$ such that $AB, CD, AD, BC \notin E$ and a direction $B \to D$. The heads of the two edges reconnect as a new edge AB and so does the tail CD. As $A \to C$, $B \to D$ and $C \to A$, $D \to B$ refer to the same rewiring operation, the choice of the direction of e_1 is randomized (this randomized bit is still an input of choosing e_2), as is shown in Figure 3. This effectively reduces the size of the action space by half. In this way, The action space is the set of all feasible pairs of $(e_1, e_2) \in E^2$, which has a variable size no larger than $2|E|(|E|-1)$.



(a) Example graph (b) First edge selection (c) Second edge selection (d) Graph after the edge rewiring

Figure 3: Illustration of the edge rewiring and the rewiring order. (a) The undirected example graph as the state; (b) Edges with random directions are candidates when selecting the first edge; (c) All $2M$ directed edges can be candidates when selecting the second edge, subject to that it forms a rewiring with the first edge; (d) The resultant graph after the rewiring, with the removal of AC, BD and the introduction of AB, CD.

Transition dynamics By the formulation of the action space, if the agent does not terminate at step t, the selected action must form an edge rewiring. This edge rewiring is executed by the environment and the graph transits to the new graph.

Note that in some other works, infeasible operations are also included in the action space (to make the action space constant through the process) (You et al., 2018; Trivedi et al., 2020). In these works, if the operation is infeasible, it is not executed and the state is not changed. This reduces the sample efficiency, causes biased gradient estimation (Huang & Ontañón, 2020), and makes the process to be prone to stuck at the state (which requires manually disabling the specific action in the next step). ResiNet takes the advantage of the action space composed of only feasible operations.

Reward ResiNet desires to optimize the resilience while preserving the utility, forming a complicated and possibly unknown objective function. Despite this, by Wakuta (1995), an MDP that maximizes a complicated objective is up to an MDP that maximizes the linear combination of resilience and utility for some coefficient factor. This fact motivates us to design the reward as the step-wise gain of such a linear combination as

$$R_t = \alpha \cdot \mathcal{R}(G_{t+1}) + (1-\alpha) \cdot \mathcal{U}(G_{t+1}) - (\alpha \cdot \mathcal{R}(G_t) + (1-\alpha) \cdot \mathcal{U}(G_t)),$$

where $\mathcal{R}(G)$ and $\mathcal{U}(G)$ are the resilience and the utility functions, respectively. The cumulative reward $\sum_{t=0}^{T-1} R_t$ up to time T is then the total gain of such a linear combination.

3.3 EDGE REWIRING POLICY

Having presented the details of the resilient graph generation environment, in this section, we describes the architecture of ResiNet in details. ResiNet is a policy network trained to maximize the cumulative reward of the agent in the graph resilience-aware environment. At time step t, the policy network takes the current graph G_t as input and outputs an action a_t which represents the two rewired edges, leading to the new state G_{t+1} and the reward R_t.

Inductively optimizing network resilience requires solving three challenges: the 1-WL test limitation of existing GNN models, the variable graph size and node permutation, and the complexity of sampling from a large action space that grows as $O(N^4)$. Our ResiNet is the first to achieve this by first proposing FireGNN via the graph filtration argumentation to enhance the expressivity of standard GNNs' expressivity and then design a specialized action network to guide the edge selection process.

3.3.1 FIREGNN VIA GRAPH FILTRATION ARGUMENTATION

Motivated by the graph filtration in persistent homology (Edelsbrunner & Harer, 2008), we propose to enhance the expressivity of GNN via the graph filtration process by augmenting an input graph G. To avoid the computational cost of calculating the persistence diagram at each step, the similarity between the filtration process and resultant subgraphs after each attack motivates us to construct a sequence of nested subgraphs of G as an approximate filtration, such that

$$(V, \emptyset) = G^{(0)} \subset G^{(1)} \subset \cdots \subset G^{(N)} = G\,,$$

where $G^{(k)}$ denotes the remaining graph after removing $N-k$ nodes in a particular task-dependent order, $G^{(N)}$ is the original graph, and $G^{(0)}$ is an empty graph with no edge.

The sequence of nested subgraphs of G is termed the filtrated graph G. Regular GNN only operates on the original graph G to obtain the node embedding for each node v_i as

$$h(v_i) = \phi(G^{(N)} = G)_i\,,$$

where $\phi(\cdot)$ denotes a standard GNN model.

Node embedding In FireGNN, by using the top $K+1$ subgraphs in a graph filtration, the final node embedding $h(v_i)$ of v_i is obtained by

$$h(v_i) = \text{Agg}_N\left(h^{(N-K)}(v_i), h^{(N-K+1)}(v_i), \ldots, h^{(N-1)}(v_i), h^{(N)}(v_i)\right)\,,$$

where $\text{Agg}_N(\cdot)$ denotes a node-level aggregation function, $h^{(k)}(v_i)$ is the node embedding of i in the k-th subgraph $G^{(k)}$, and $K \in [N]$. In practice, $h^{(k)}(v_i)$ is discarded when calculating $h(v_i)$ if v_i is isolated or not included in $G^{(k)}$.

Edge embedding The directed edge embedding $h^{(k)}(e_{ij})$ of the edge from node i to node j in each subgraph is obtained by combining the embeddings of the two end vertices in $G^{(k)}$ as

$$h^{(k)}(e_{ij}) = m_f\left(\text{Agg}_{N\to E}\left(h^{(k)}(v_i), h^{(k)}(v_j)\right)\right)\,,$$

where $\text{Agg}_{N\to E}(\cdot)$ denotes an aggregation function for obtaining edge embedding from two end vertices (typically chosen from `min`, `max`, `sum`, `difference`, and `multiplication`). $m_f(\cdot)$ is a multilayer perceptron (MLP) model that ensures the consistent dimension of edge embedding and graph embedding.

The final embedding of the directed edge e_{ij} of the filtrated graph G is given by,

$$h(e_{ij}) = \text{Agg}_E\left(h^{(N-K)}(e_{ij}), h^{(N-K+1)}(e_{ij}), \ldots, h^{(N-1)}(e_{ij}), h^{(N)}(e_{ij})\right)\,,$$

where $\text{Agg}_E(\cdot)$ denotes an edge-level aggregation function.

Graph embedding With the node embedding $h^{(k)}(v_i)$ of each subgraph available, the graph embedding $h^{(k)}(G)$ of each subgraph is calculated by a readout functions (e.g., `mean`, `sum`) on all non-isolated nodes in $G^{(k)}$

$$h^{(k)}(G) = \text{READOUT}\left(h^{(k)}(v_i)\right) \ \forall v_i \in G^{(k)} \text{ and } d_i^{(k)} \geq 0\,.$$

The final graph embedding of the filtrated graph G is given by,

$$h(G) = \text{Agg}_G\left(h^{(N-K)}(G), h^{(N-K+1)}(G), \ldots, h^{(N-1)}(G), h^{(N)}(G)\right)\,,$$

where $\text{Agg}_G(\cdot)$ denotes a graph-level aggregation function.

By observing a sequence of nested subgraphs of G, a filtration process transforms an input graph from the static graph to a sequences of subgraphs, which grants GNNs the capability to observe how G evolves and improve the expressivity in node, edge, and graph levels. We term the **Filt**rated graph **e**nhanced **GNN** as FireGNN.

3.3.2 POLICY NETWORK

After obtaining the directed edge embedding $h(e_{ij}) \in \mathbb{R}^{2|E|\times d}$ and the graph embedding $h(G) \in \mathbb{R}^d$ from the proposed FireGNN, armed with our specially designed autoregressive node permutation-invariant dimension-variable action space, we can select the edge rewiring operation for graphs with arbitrary sizes and node-permutations. The detailed architecture of ResiNet and the mechanism of obtaining the action a_t are presented as follows.

Auto-regressive latent action selection An edge rewiring action a_t at time step t involves the prediction of the termination probability $a_t^{(0)}$ and the selection of two edges ($a_t^{(1)}$ and $a_t^{(2)}$) and the rewiring order. The action space of $a_t^{(0)}$ is binary, however, the selection of two edges imposes a huge action space in $O(E^2)$, which is too expensive to sample from even for a small graph. Instead of selecting two edges simultaneously, we decompose the joint action a_t into $a_t = (a_t^{(0)}, a_t^{(1)}, a_t^{(2)})$, where $a_t^{(1)}$ and $a_t^{(2)}$ are two existing edges which do not share any common node (recall that $a_t^{(1)}$ and $a_t^{(2)}$ are directed edges for an undirected graph). Thus the probability of the action a_t is formulated as

$$\mathbb{P}(a_t|s_t) = \mathbb{P}(a_t^{(0)}|s_t)\mathbb{P}(a_t^{(1)}|s_t, a_t^{(0)})\mathbb{P}(a_t^{(2)}|s_t, a_t^{(0)}, a_t^{(1)})\,.$$

Predicting the termination probability The first policy network $\pi_0(\cdot)$ takes the graph embedding as input and outputs the probability distribution of the first action that decides to terminate or not.

$$\mathbb{P}(a_t^{(0)}|s_t) = \pi_0(h(G))\,,$$

where $\pi_0(\cdot)$ is implemented by a two layer MLP. Then $a_t^{(0)} \sim \text{Ber}(\mathbb{P}(a_t^{(0)}|s_t)) \in \{0, 1\}$.

Selecting two edges If the signal $a_t^{(0)}$ given by the agent decides to continue to rewire, two edges are then selected in an auto-regressive way. The signal of continuing to rewire $a_t^{(0)}$ is input to the selection of two edges as a one-hot encoding vector l_c. The second policy network $\pi_1(\cdot)$ takes the graph embedding and l_c as input and outputs a latent vector $l_1 \in \mathbb{R}^d$. The pointer network (Vinyals et al., 2015) is used to measure the proximity between l_1 and each edge embedding $h(e_{ij})$ in G to obtain the first edge selection probability distribution. Then, to select the second edge, the graph embedding $h(G)$ and the first selected edge embedding $h(e_t^{(1)})$ and l_c are concatenated and fed into the third policy network $\pi_2(\cdot)$. $\pi_2(\cdot)$ obtains the latent vector l_2 for selecting the second edge using a respective pointer network. The process can be formulated as

$$\begin{aligned}
l_1 &= \pi_1([h(G), l_c]) \\
\mathbb{P}(a_t^{(1)}|s_t, a_t^{(0)}) &= f_1(l_1, h(e_{ij}))), \ \forall e_{ij} \in E \\
l_2 &= \pi_2([h(G), h_{e_t^{(1)}}, l_c]) \\
\mathbb{P}(a_t^{(2)}|s_t, a_t^{(1)}, a_t^{(0)}) &= f_2(l_2, h(e_{ij}))), \ \forall e_{ij} \in E\,,
\end{aligned}$$

where $\pi_i(\cdot)$ is implemented by a two layer MLP model. $[\cdot,\cdot]$ denotes the concatenation operator, $h_{e_t^{(1)}}$ is the embedding of the first selected edge at time step t, and $f_i(\cdot)$ is a pointer network.

Table 1: Resilience optimization algorithm. We report the weighted combination of the graph connectivity-based resilience and the network efficiency improvement (in percentage) with the weighted coefficient $\alpha \in \{0, 0.5\}$. Results are averaged over 3 runs.

Method	α	BA-15	BA-50	BA-100	EU
HC	0	32.7	63.4	**107.3**	71.1
	0.5	26.8	46.9	74.2	**62.1**
Greedy	0	20.9	**72.5**	101.3	**71.3**
	0.5	14.2	41.7	72.8	57.3
SA	0	32.0	63.2	104.8	69.0
	0.5	23.1	48.2	**75.6**	60.7
EA	0	**35.3**	50.2	61.9	66.2
	0.5	**27.1**	38.3	46.6	58.4
ResiNet	0	**35.3**	61.5	70.0	54.2
	0.5	26.9	**53.9**	53.1	51.8

4 EXPERIMENTS

In this section, we demonstrate the advantage of ResiNet in optimizing network resilience. We show that the optimization is achieved with fewer edge rewiring operations, the inductive ability to generalize to unseen graphs, and the ability to accommodating multiple resilience metrics and utility metrics. Moreover, ResiNet provides insights to better understand and interpret the evolving network structure to improve resilience.

4.1 EXPERIMENTAL SETTINGS

Datasets Synthetic datasets and the real EU power network (Zhou & Bialek, 2005) are used to demonstrate the performance of ResiNet in both transductive and inductive settings. The details of data generation and the statistics of the datasets are presented in Appendix C.1.

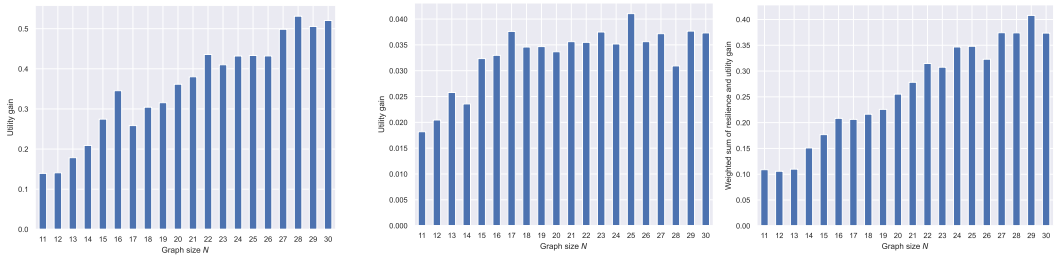
Baselines We compare ResiNet with existing graph resilience optimization algorithms, including hill climbing (HC) (Schneider et al., 2011), the greedy algorithm (Chan & Akoglu, 2016), simulated annealing (SA) (Buesser et al., 2011), and the evolutionary algorithm (EA) (Zhou & Liu, 2014). All algorithms are repeated for three random seeds using their recommended hyper-parameters to obtain the averaged performance. The parameters and training strategies of ResiNet are given in Appendix C.2.

4.2 COMPARISONS TO THE BASELINES

In this set of experiments, we first compare ResiNet to the baselines in optimizing the combination of resilience and utility with weight coefficient $\alpha \in \{0, 0.5\}$. The graph connectivity-based metric is used as the resilience metric and the global efficiency is used as the utility metric. The test is conducted on 4 transductive settings and 3 inductive settings.

Table 1 records the metric gain of different methods. The table indicates that ResiNet achieves the best performance on BA-15 and is comparable to baselines on large datasets. Note that this comparable performance is under a much fewer number of rewiring operations. In general, the metric gain is larger for larger graphs, which implies that it is sensible and cost-effective to improve the resilience of large networks.

Compared to other algorithms, ResiNet outperforms them on small networks and achieves solid improvements on graphs of different sizes with a single set of parameters. In contrast, the greedy algorithm is trapped into a local optimum by maximizing the one-step resilience gain. For SA, the initial temperature and the temperature decay rate need to be carefully tuned for each network. The population-based EA performs well when $\alpha = 0$, as it uses the "onion structure" prior information to encourage the nodes to connect to other nodes that are with similar degrees. EA will only work on tasks that specifically desire the onion structure, however.



(a) Inductivity on resilience (b) Inductivity on utility (c) Inductivity on both metrics

Figure 4: The inductive ability of ResiNet on the test dataset (BA-10-30) when optimizing (a) network resilience, (b) network utility, and (c) their combination.

4.3 Generalization

In this section, we conduct extensive experiments to show that ResiNet generalizes to different utility and resilience metrics, various attack strategies, and unseen graphs.

To explore the complicated objective of resilience and utility, the BA-15 network is taken as an example to be optimized by ResiNet to obtain the approximate Pareto frontier. The Pareto points are shown in Figure 6 to denote the optimum under different objectives. Surprisingly, the initial gain of resilience (from around 0.21 to around 0.24) is without loss of the utility, which incentivizes almost every network to conduct such optimization at least to some extent when feasible.

To demonstrate the inductivity of ResiNet, we first train ResiNet on two different datasets and the data setting is recorded in Table 2 and we evaluate its performance on an individual test dataset. These datasets are unknown to ResiNet and have not been observed during the training process. Fine-tuning is not allowed. For each dataset we report the averaged resilience gain for the graphs with the same number of nodes. The result of BA-10-30 is shown in Figure 4 and the results of the other datasets are deferred to Appendix C.6. Figure 4 shows a nearly linear improvement of resilience with the increasing of graph size, which is also consistent with the results in the transductive setting that larger graphs usually have a larger room to improve. In contrast, the gain in the inductive setting for the baselines are neglectable.

To demonstrate that ResiNet can learn from the networks for accommodating different utility and resilience functions, we conduct experiments based on the BA-15 and EU datasets by using multiple different resilience functions and different utility functions (details are in Appendix B). The results are included in Appendix C.6. The network structures are visualized in Figure 8 and Figure 10.

5 Conclusion and Future Works

In this work, we propose a general, learning-based framework, ResiNet, for the inductive discovery of resilient network topologies. The framework is practically feasible as it also preserves the utility of the networks. ResiNet is the first to formulate the problem of boosting network resilience as an MDP, with a specialized auto-regressive permutation-invariant variable action space. This variant of MDP allows the agent to handle graphs of different sizes and isomorphisms. Once an agent is trained, it inductively works on a wide range of graphs and is ready to be deployed to real systems. Our technical innovation, FireGNN, is a powerful technique for node and graph representations, which is a suitable replacement to other GNNs, especially in RL. It balances the expressive power and the oversmoothing issue of GNNs and could distinguish graphs and connected nodes at the same time. Both ResiNet and FireGNN are general enough to be applied to a variety of problems in graph theory, network theory, combinatorial optimization, machine learning, and their applications.

An interesting direction of future work is to obtain a better approximation algorithm for the persistence diagram. The current implementation includes an average operation across the entire filtration, which is computationally expensive when the graph is large. This forces us to truncate the filtration in our implementations. An efficient and accurate approximation algorithm will grant FireGNN better feasibility in real applications.

REFERENCES

Mehmet E Aktas, Esra Akbas, and Ahmed El Fatmaoui. Persistence homology of networks: methods and applications. *Applied Network Science*, 4(1):1–28, 2019.

Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47, 2002.

Réka Albert, Hawoong Jeong, and Albert-László Barabási. Error and attack tolerance of complex networks. *Nature*, 406(6794):378–382, 2000.

Marcin Andrychowicz, Anton Raichuk, Piotr Stanczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters in on-policy reinforcement learning? A large-scale empirical study. *CoRR*, abs/2006.05990, 2020. URL `https://arxiv.org/abs/2006.05990`.

Yunsheng Bai, Hao Ding, Song Bian, Ting Chen, Yizhou Sun, and Wei Wang. Graph edit distance computation via graph neural networks. *CoRR*, abs/1808.05689, 2018. URL `http://arxiv.org/abs/1808.05689`.

Sergey Bereg and Hiro Ito. Transforming graphs with the same graphic sequence. *Journal of Information Processing*, 25:627–633, 2017. doi: 10.2197/ipsjjip.25.627.

Stefano Boccaletti, Vito Latora, Yamir Moreno, Martin Chavez, and D-U Hwang. Complex networks: Structure and dynamics. *Physics Reports*, 424(4-5):175–308, 2006.

Béla Bollobás and Oliver Riordan. Robustness and vulnerability of scale-free random graphs. *Internet Mathematics*, 1(1):1–35, 2004.

Pierre Buesser, Fabio Daolio, and Marco Tomassini. Optimizing the robustness of scale-free networks with simulated annealing. In *International Conference on Adaptive and Natural Computing Algorithms*, pp. 167–176. Springer, 2011.

Tianle Cai, Shengjie Luo, Keyulu Xu, Di He, Tie-Yan Liu, and Liwei Wang. GraphNorm: A principled approach to accelerating graph neural network training. *CoRR*, abs/2009.03294, 2020. URL `https://arxiv.org/abs/2009.03294`.

Vincenza Carchiolo, Marco Grassia, Alessandro Longheu, Michele Malgeri, and Giuseppe Mangioni. Network robustness improvement via long-range links. *Computational Social Networks*, 6 (1):1–16, 2019.

Hau Chan and Leman Akoglu. Optimizing network robustness by edge rewiring: a general framework. *Data Mining and Knowledge Discovery*, 30(5):1395–1425, 2016.

Herbert Edelsbrunner and John Harer. Persistent homology-A survey. *Contemporary Mathematics*, 453:257–282, 2008.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pp. 226–231, 1996.

Changjun Fan, Li Zeng, Yizhou Sun, and Yang-Yu Liu. Finding key players in complex networks through deep reinforcement learning. *Nature Machine Intelligence*, pp. 1–8, 2020.

Zhang-Hua Fu, Kai-Bin Qiu, and Hongyuan Zha. Generalize a small pre-trained model to arbitrarily large TSP instances. *CoRR*, abs/2012.10658, 2020. URL `https://arxiv.org/abs/2012.10658`.

Marco Grassia, Manlio De Domenico, and Giuseppe Mangioni. Machine learning dismantling and early-warning signals of disintegration in complex systems. *arXiv preprint arXiv:2101.02453*, 2021.

Xiaojie Guo and Liang Zhao. A systematic survey on deep generative models for graph generation. *CoRR*, abs/2007.06686, 2020. URL `https://arxiv.org/abs/2007.06686`.

Xiaojie Guo, Lingfei Wu, and Liang Zhao. Deep graph translation. *CoRR*, abs/1805.09980, 2018. URL http://arxiv.org/abs/1805.09980.

Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 4116–4126. PMLR, 13–18 Jul 2020.

Christoph Hofer, Florian Graf, Bastian Rieck, Marc Niethammer, and Roland Kwitt. Graph filtration learning. In *International Conference on Machine Learning*, pp. 4314–4323. PMLR, 2020.

Petter Holme, Beom Jun Kim, Chang No Yoon, and Seung Kee Han. Attack vulnerability of complex networks. *Physical Review E*, 65(5):056109, 2002.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=HJlWWJSFDH.

Shengyi Huang and Santiago Ontañón. A closer look at invalid action masking in policy gradient algorithms. *CoRR*, abs/2006.14171, 2020. URL https://arxiv.org/abs/2006.14171.

Swami Iyer, Timothy Killingback, Bala Sundaram, and Zhen Wang. Attack robustness and centrality of complex networks. *PloS One*, 8(4):e59613, 2013.

Chaitanya K. Joshi, Thomas Laurent, and Xavier Bresson. An efficient graph convolutional network technique for the travelling salesman problem. *CoRR*, abs/1906.01227, 2019. URL http://arxiv.org/abs/1906.01227.

Nikolaos Karalias and Andreas Loukas. Erdos goes neural: an unsupervised learning framework for combinatorial optimization on graphs. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/49f85a9ed090b20c8bed85a5923c669f-Abstract.html.

Elias B. Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 6348–6358, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/d9896106ca98d3d05b8cbdf4fd8b13a1-Abstract.html.

P. Diederik Kingma and Lei Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 971–980, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/5d44ee6f2c3f71b73125876103c8f6c4-Abstract.html.

Vito Latora and Massimo Marchiori. Efficient behavior of small-world networks. *Physical Review Letters*, 87(19):198701, 2001.

Vito Latora and Massimo Marchiori. Economic small-world behavior in weighted networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 32(2):249–263, 2003.

Henrique Lemos, Marcelo O. R. Prates, Pedro H. C. Avelar, and Luís C. Lamb. Graph colouring meets deep learning: Effective graph neural network models for combinatorial problems. In *31st IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2019, Portland, OR, USA, November 4-6, 2019*, pp. 879–885. IEEE, 2019. doi: 10.1109/ICTAI.2019.00125. URL https://doi.org/10.1109/ICTAI.2019.00125.

Guohao Li, Chenxin Xiong, Ali K. Thabet, and Bernard Ghanem. DeeperGCN: All you need to train deeper gcns. *CoRR*, abs/2006.07739, 2020a. URL https://arxiv.org/abs/2006.07739.

Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding: Design provably more powerful neural networks for graph representation learning. *Advances in Neural Information Processing Systems*, 33, 2020b.

Wenguo Li, Yong Li, Yi Tan, Yijia Cao, Chun Chen, Ye Cai, Kwang Y Lee, and Michael Pecht. Maximizing network resilience against malicious attacks. *Scientific Reports*, 9(1):1–9, 2019.

Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Combinatorial optimization with graph convolutional networks and guided tree search. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 537–546, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/8d3bba7425e7c98c50f52ca1b52d3735-Abstract.html.

Sourav Medya, Tianyi Ma, Arlei Silva, and Ambuj Singh. A game theoretic approach for k-core minimization. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '20, pp. 1922–1924. International Foundation for Autonomous Agents and Multiagent Systems, 2020. ISBN 9781450375184.

Damon Mosk-Aoyama. Maximum algebraic connectivity augmentation is NP-hard. *Operations Research Letters*, 36(6):677–679, 2008.

MohammadReza Nazari, Afshin Oroojlooy, Lawrence V. Snyder, and Martin Takác. Reinforcement learning for solving the vehicle routing problem. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 9861–9871, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/9fb4651c05b2ed70fba5afe0b039a550-Abstract.html.

Bo Peng, Jiahai Wang, and Zizhen Zhang. A deep reinforcement learning algorithm using dynamic attention model for vehicle routing problems. *CoRR*, abs/2002.03282, 2020. URL https://arxiv.org/abs/2002.03282.

Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. GCC: graph contrastive coding for graph neural network pre-training. In Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash (eds.), *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pp. 1150–1160. ACM, 2020. doi: 10.1145/3394486.3403168. URL https://doi.org/10.1145/3394486.3403168.

Lei Rong and Jing Liu. A heuristic algorithm for enhancing the robustness of scale-free networks based on edge classification. *Physica A: Statistical Mechanics and its Applications*, 503:503–515, 2018.

Christian M Schneider, André A Moreira, José S Andrade, Shlomo Havlin, and Hans J Herrmann. Mitigation of malicious attacks on networks. *Proceedings of the National Academy of Sciences*, 108(10):3838–3841, 2011.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Rakshit Trivedi, Jiachen Yang, and Hongyuan Zha. Graphopt: Learning optimization models of graph formation. In *International Conference on Machine Learning*, pp. 9603–9613. PMLR, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. Deep graph infomax. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=rklz9iAcKQ.

Clement Vignac, Andreas Loukas, and Pascal Frossard. Building powerful and equivariant graph neural networks with structural message-passing. *arXiv preprint arXiv:2006.15107*, 2020.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/29921001f2f04bd3baee84a12e98098f-Paper.pdf.

Kazuyoshi Wakuta. Vector-valued markov decision processes and the systems of linear inequalities. *Stochastic Processes and Their Applications*, 56(1):159–169, 1995.

Po-Wei Wang, Priya L. Donti, Bryan Wilder, and J. Zico Kolter. SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6545–6554. PMLR, 2019. URL http://proceedings.mlr.press/v97/wang19e.html.

Xiangrong Wang, Evangelos Pournaras, Robert E Kooij, and Piet Van Mieghem. Improving robustness of complex networks via the effective graph resistance. *The European Physical Journal B*, 87(9):1–12, 2014.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

A Yasin Yazıcıoğlu, Magnus Egerstedt, and Jeff S Shamma. Formation of robust multi-agent networks through self-organizing random regular graphs. *IEEE Transactions on Network Science and Engineering*, 2(4):139–151, 2015.

Emre Yolcu and Barnabás Póczos. Learning local search heuristics for boolean satisfiability. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 7990–8001, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/12e59a33dea1bf0630f46edfe13d6ea2-Abstract.html.

Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay S. Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 6412–6422, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/d60678e8f2ba9c540798ebbde31177e8-Abstract.html.

Jiaxuan You, Jonathan Gomes-Selman, Rex Ying, and Jure Leskovec. Identity-aware graph neural networks. *CoRR*, abs/2101.10320, 2021. URL https://arxiv.org/abs/2101.10320.

Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/3fe230348e9a12c13120749e3f9fa4cd-Abstract.html.

James Jian Qiao Yu, Wen Yu, and Jiatao Gu. Online vehicle routing with neural combinatorial optimization and deep reinforcement learning. *IEEE Trans. Intell. Transp. Syst.*, 20(10):3806–3817, 2019. doi: 10.1109/TITS.2019.2909109. URL https://doi.org/10.1109/TITS.2019.2909109.

Fan Zhang, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. Finding critical users for social network engagement: The collapsed k-core problem. In Satinder P. Singh and Shaul Markovitch (eds.), *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pp. 245–251. AAAI Press, 2017. URL http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14349.

Kangfei Zhao, Zhiwei Zhang, Yu Rong, Jeffrey Xu Yu, and Junzhou Huang. Finding critical users in social communities via graph convolutions. *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2021. doi: 10.1109/TKDE.2021.3089763.

Mingxing Zhou and Jing Liu. A memetic algorithm for enhancing the robustness of scale-free networks against malicious attacks. *Physica A: Statistical Mechanics and its Applications*, 410: 131–143, 2014.

Qiong Zhou and Janusz W Bialek. Approximate model of european interconnected system as a benchmark system to study effects of cross-border trades. *IEEE Transactions on Power Systems*, 20(2):782–788, 2005.

A EXTENDED RELATED WORK

GNNs for combinatorial optimization problems.

The idea of applying GNNs to solve combinatorial optimization problems on graphs is becoming a promising yet challenging topic. Abundant algorithms have been developed and can be classified into three categories. ***Supervised learning approach*** combines GNNs with heuristics search procedures or solvers to solve classical combinatorial problems, such as graph matching (Bai et al., 2018), graph coloring (Lemos et al., 2019), TSP (Li et al., 2018; Joshi et al., 2019) and SAT (Wang et al., 2019). These methods require labeled instances and thus are difficult to generalize to large-scale instances directly. ***Unsupervised learning approach*** (Karalias & Loukas, 2020) trains GNNs to parametrize a probability distribution over sets. The probabilistic proof achieves the existence of feasible solutions. After that, the derandomized method is applied to decode the desired solutions. However, meeting complex constraints have not been supported yet. ***Reinforcement learning approach*** formulates a problem as a Markov decision process with a proper reward function to guide the search towards an optimal solution. Some classical CO problems have achieved remarkable results like TSP (Fu et al., 2020; Khalil et al., 2017), Vehicle Routing Problem (Nazari et al., 2018; Peng et al., 2020; Yu et al., 2019), SAT (Yolcu & Póczos, 2019) and max-cut (Khalil et al., 2017). In this paper, we adopt RL based approach but with a specialized policy network.

Multi-views graph augmentation for GNNs

Multi-views graph augmentation is one efficient way to improve expressive power or combine domain knowledge, which is adapted based on the task's prior (Hu et al., 2020). For example, GCC (Qiu et al., 2020) generates multiple subgraphs from the same ego network. DGI (Velickovic et al., 2019) maximizes the mutual information between global and local information. GCA (You et al., 2020) adaptively incorporates various priors for topological and semantic aspects of the graph. (Hassani & Khasahmadi, 2020) contrasts representations from first-order neighbors and a graph diffusion. In our problem, two desired properties are different edge representations and the capability to

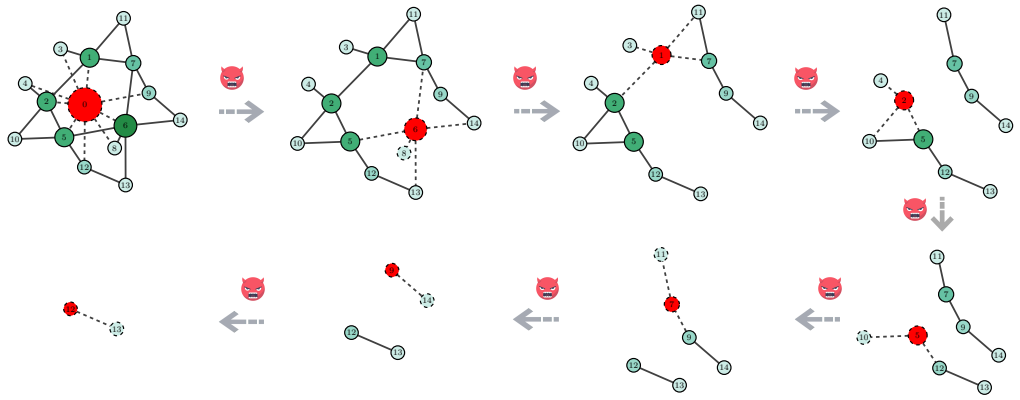


Figure 5: Illustration of the graph connectivity-based measurement calculation with a scale-free example graph under node degree-based malicious attack. Each snapshot demonstrates the attack on the red node, which removes the red node and its adjacent (dashed) edges, leading to a next remnant graph. The attacker recalculates the node degree and repeatedly seeks for the remaining node with a maximum degree as the next target. In the end, the entire graph becomes disconnected when each node cannot reach other nodes.

predict accurate network resilience gain. Motivated by the calculation process of persistent homology (Edelsbrunner & Harer, 2008), we apply the filtration process to enhance the expressive power of GNNs.

Deep graph generation models

Deep graph generation models learn the distribution of given graphs and generate more novel graphs, which could be classified as unconditional generation methods and conditional generation methods (Guo & Zhao, 2020). Specifically, edge transformation conditioned on another graph (a.k.a. graph translation) is more relevant to our work since the edge rewiring process can be regarded as operations on the previous graph via edge addition and deletion. Some works utilize the encoder-decoder framework by learning latent representation of the input graph through the encoder and then generating the target graph through the decoder. For example, GCPN (You et al., 2018) incorporates chemistry domain rules on molecular graph generation. GT-GAN (Guo et al., 2018) proposes a GAN-based model on malware cyber-network synthesis. GraphOpt (Trivedi et al., 2020) learns an implicit model to discover an underlying optimization mechanism of the graph generation using inverse reinforcement learning. Although exploiting a similar encoder-decoder framework, our approach differs from existing works by the rigid constraint of the studied task: node degree preservation, which imposes additional requirements for the expressive power of GNNs.

B Definitions of different objective functions

In this section, we present the resilience definitions and the utility definitions we used in our experiments.

B.1 Resilience definitions

The graph connectivity-based measurement (Schneider et al., 2011) is defined as :

$$\mathcal{R}(G) = \frac{1}{N}\sum_{q=1}^{N} s(q)\,, \tag{1}$$

where $s(q)$ is the fraction of nodes in the largest connected remaining graph after removing q nodes from graph G according to certain attack strategy, as shown in Figure 5. The range of possible values of $\mathcal{R}$ is $[1/N, 1/2]$, where these two extreme values correspond to a star network and a fully connected network, respectively.

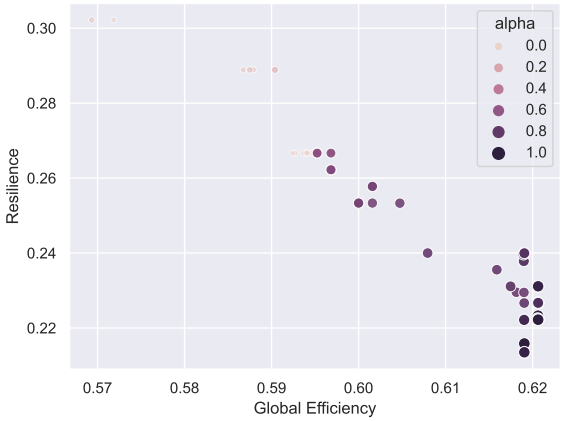


Figure 6: Pareto points obtained by ResiNet of balancing various combinations of resilience and utility on BA-15 dataset.

Other resilience measures include the spectral radius based on adjacency matrix spectrum and the algebraic connectivity based on Laplacian matrix spectrum. The largest eigenvalue λ_1 of an adjacency matrix is call the spectral radius and the algebraic connectivity is defined as the second smallest eigenvalue of the Laplacian matrix.

B.2 Utility definitions

Efficiency is widely used across diverse applications of network science, such as transportation and communication networks (Latora & Marchiori, 2003; Boccaletti et al., 2006). In this paper, the global and local communication efficiency are used as two measurements of the network utility.

The average efficiency of a network G is defined as inversely proportional to the distance (Latora & Marchiori, 2001):

$$E(G) = \frac{1}{N(N-1)} \sum_{i \neq j \in G} \frac{1}{d(i,j)}\,, \tag{2}$$

where N denotes the total nodes in a network and $d(i,j)$ denotes the length of the shortest path between a node i and another node j.

The global efficiency of a network G is defined as (Latora & Marchiori, 2001; 2003):

$$E_{glob}(G) = \frac{E(G)}{E(G^{ideal})}\,, \tag{3}$$

where G^{ideal} is the "ideal" fully-connected graph on N nodes, and the range of $E_{glob}(G)$ is [0, 1].

The local efficiency of a network G measures a local average of pairwise communication efficiencies and is defined as (Latora & Marchiori, 2001):

$$E_{loc}(G) = \frac{1}{N} \sum_{i \in G} E(G_i)\,, \tag{4}$$

where G_i is the local subgraph including only of a node i's one-hop neighbors, but not the node i itself. The range of $E_{local}(G)$ is [0, 1].

The pareto points obtained by ResiNet on the BA-15 dataset in Figure 6 implies that the maximization of the resilience and utility may not be satisfied simultaneously except for the poor-designed systems.

Table 2: Statistics of graphs used for resilience maximization. Both transductive and inductive settings ($\star$) are included. Consistent with our implementation, we report the number of edges by transforming undirected graphs to directed graph so that the edge rewiring has the fixed execution order. For the inductive setting, we report the maximum number of edges. The action space size of the edge rewiring is measured by $2E^2$.

Dataset	Node	Edge	Action Space Size	Train/Test	Setting
BA-15	15	54	5832	Null	Transductive
BA-50	50	192	73728	Null	Transductive
BA-100	100	196	76832	Null	Transductive
EU	217	392	307328	Null	Transductive
BA-10-30 ($\star$)	10-30	112	25088	1000/500	Inductive
BA-20-200 ($\star$)	40-200	792	1254528	4500/360	Inductive

C IMPLEMENTATION DETAILS OF RESINET

In this section, we provide the implementation details of ResiNet, including dataset, network structure training strategies and node feature construction. Moreover, we discussed the role of FireGNN in ResiNet and analyze current limitations of ResiNet and future work.

C.1 DATASET

In this section, we present the data generation strategies and report the statistics of each dataset in Table 2. Synthetic datasets are generated using the Barabasi-Albert (BA) model (known as scale-free graphs) (Albert & Barabási, 2002), with the node set size varying from $|N|$=10 to $|N|$=200. During the data generation process, each node is connected to two existing nodes. For the transductive setting, all algorithms are trained and tested on randomly generated BA networks with sizes 15, 50, and 100. BA graphs are chosen since they are vulnerable to malicious attacks on networks and are commonly used to test network resilience optimization algorithms (Bollobás & Riordan, 2004). For the inductive setting, the algorithm is still trained on BA graphs but the real EU power network is used to compare the performance of different methods. The statistics of graphs used for resilience maximization experiments are provided in Table 2, where BA-m means the a randomly generated BA network with m nodes for the transductive setting for evaluating all algorithms, and BA-m-n denotes the inductive setting for only evaluating ResiNet since other methods are all transductive.

C.2 RESINET SETUP

In this section, we provide detailed parameter setting and training strageise for ResiNet.

Our proposed FireGNN is used as the graph encoder in ResiNet, including a 5-layer defined GIN (Xu et al., 2019) as backbone. The hidden dimensions for node embedding and graph embedding in each hidden layer are set to 64 and the SeLU activation function (Klambauer et al., 2017) is used before each message passing propagate (Li et al., 2020a). The node feature for G consists of node degree and the attack-related information, such as attack order, the shortest path length, as presented in C.3. Graph normalization strategy is adopted to stabilize the training of GNN (Cai et al., 2020).

The overall policy is trained by using the highly tuned implementation of proximal policy optimization (PPO) algorithm (Schulman et al., 2017). Several critical strategies for stabilizing and accelerating training of ResiNet are used, including advantage normalization (Andrychowicz et al., 2020) and separate optimizers for policy network and value network. Additionally, since the stepwise reward range is small (around 0.01), we scale the reward by a factor of 10 to aim the training of ResiNet. The policy head model and value function model use two separated FireGNN encoder network with the same architecture. ResiNet is trained using two separate Adam optimizers (Kingma & Ba, 2015) with batch size 256 and the linearly decayed learning rate 0.0007 for the policy network and 0.0003 the for value network. The aggregation function of FireGNN is defined as an attention mechanism-based linear weighted combination.

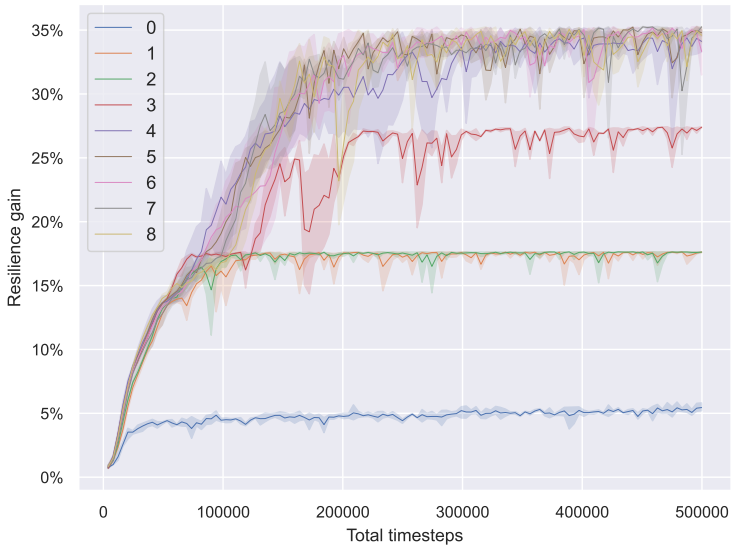


Figure 7: Experiments on the BA-15 dataset to study the effect of filtration order on FireGNN on maximizing resilience. Results are averaged over 3 runs.

C.3 NODE FEATURE CONSTRUCTION

The widely-used node degree features cannot significant benefit the optimization of a single graph due to the degree-preserving rewiring. Therefore, we construct node features for each input graph to aid the inductive learning as follows:

- The 8-dimensisonal position embedding originating from the Transformer (Vaswani et al., 2017) is adopted, as the measurement of the vulnerability of each node under attack. If the attack order is available, we can encode it into the position embedding. If the attack order is unknown, we can use the node degree or node betweenness to approximate it in practice.
- The distance encoding strategy (Li et al., 2020b).
- One-hot encoding to denote whether the node removal is caused by the attack itself, or just due to that fact that all adjacent nodes are removed.

C.4 THE ROLE OF FIREGNN

In this section, we provide a discuss about the role of FireGNN in aiding the network resilience optimization.

In FireGNN, the total number of subgraphs involved in calculating the final node embedding, edge embedding and graph embedding, are determined by the filtration order $K \in [0, N-1]$, where $K = 0$ means only the original graph is used and $K = N - 1$ means there are total N subgraphs used. We run a grid search to explore the effect of the filtration order on ResiNet's performance. As shown in Figure 7, ResiNet only achieves a minor gain of around 5% without FireGNN. For larger $K > 0$, the performance is improved significantly and ResiNet obtained the optimal resilience gain of around 35 % on BA-15 when $K = 5$. We only report the performance of $K \leq 8$ since the BA-15 loses all connections when more than 8 critical nodes are removed.

Besides the BA-15 dataset, We also conducted the ablation experiments of FireGNN on large datasets, including the BA-50, BA-100 and EU power network. We found that without FireGNN, ResiNet hardly learn to find a positive gain so it often chooses to stop at the first time step, which maybe further validates the effectiveness of FireGNN. More discussions of the importance and the possible reasons why FireGNN is critical for the network resilience task are provided in Appendix C.4

- Each rewire only change the graph by four edges, so the graph embedding and the node embedding may not change significantly at two steps to provide essential information for the RL agent to make new decisions. For example, we empirically found that the RL agent can be stuck in an action loop, which means that after the rewiring of A-C and B-D to obtain A-B and C-D at step t, the agent may choose to rewire the new forming edges back at $t + 1$, forming an infinite loop.
- The dominating nodes may potentially be regarded as outliers which can shade other nodes unique features, which is problematic for agent's learning. By removing the dominating nodes, the residual nodes in each subgraph have the freedom to express their less salient features.
- It might also potentially useful for other edge embedding-based tasks, such as graph generation, recommendation and link prediction.

C.5 Ablation study

C.5.1 FireGNN

In FireGNN, the total number of subgraphs involved in calculating the final node embedding, edge embedding and graph embedding, are determined by the filtration order $K \in [0, N-1]$, where $K = 0$ means only the original graph is used and $K = N - 1$ means there are total N subgraphs used. We run a grid search to explore the effect of the filtration order on ResiNet's performance. As shown in Figure 7, ResiNet only achieves a minor gain of around 5% without FireGNN. For larger $K > 0$, the performance is improved significantly and ResiNet obtained the optimal resilience gain of around 35 % on BA-15 when $K = 5$. We only report the performance of $K \leq 8$ since the BA-15 loses all connections when more than 8 critical nodes are removed.

Besides the BA-15 dataset, We also conducted the ablation experiments of FireGNN on large datasets, including the BA-50, BA-100 and EU power network. We found that without FireGNN, ResiNet hardly learn to find a positive gain so it often chooses to stop at the first time step, which maybe further validates the effectiveness of FireGNN. More discussions of the importance and the possible reasons why FireGNN is critical for the network resilience task are provided in Appendix C.4

C.6 More experimental results

C.6.1 Optimization with different resilience and utility metrics

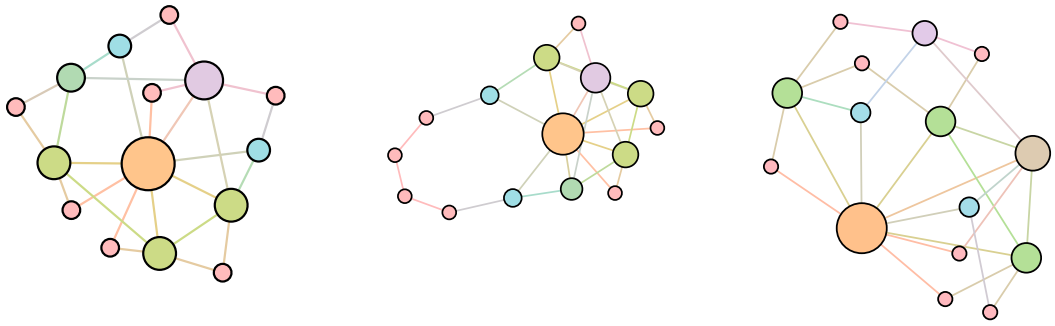
Defending betweenness based attacks We use BA-15 with 15 nodes and 27 edges, as visualized in Figure 8, to show that ResiNet is not limited to the node degree based attacks and can also learn to defend the betweeness based attacks.

The optimized network with a improvement about 3.6% for defending the betweeness based attacks also has a higher resilience (around 7.8%) against node degree attacks. This may be explained as the similarity of node degree and betweeness for a small network.

Meanwhile, the number of rewiring operations needed is much smaller for ResiNet compared to the baselines, especially for large networks. A large number of rewiring is undesired in practice, as transforming the original network to the target network could incur a significant cost. Given a target graph, obtaining the minimum rewiring operations is NP-hard (Bereg & Ito, 2017). For example, in the BA-15 dataset, ResiNet only takes 6 rewiring operations to obtain the graph with optimal resilience, while the baselines require a larger number from a dozen to a hundred. Different from the training strategies of other algorithms that there is no limit for the number of rewiring operations, ResiNet is trained using RL with a rewiring budget of no more than 20 times of edges rewiring. That is, if the agent has rewired for 20 times, it would return to the original graph position and start a new round of search.

C.6.2 Inductivity on more datasets

Even with limited computational resource, armed with the autoregressive action space and the power of FireGNN, ResiNet can be trained fully end-to-end on graphs with hundreds of nodes using re-

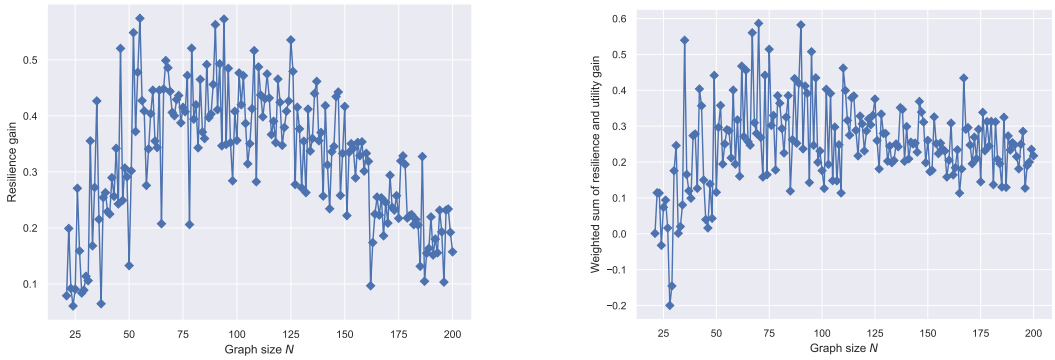


(a) Original network

(b) Optimization against degree attack

(c) Optimization against betweeness attack

Figure 8: The resilience maximization on the BA-15 dataset with 15 nodes and 27 edges with (a) original network, (b) optimization against node degree based attacks, and (c) optimization against node bewteeness based attacks.



(a) Inductivity on resilience

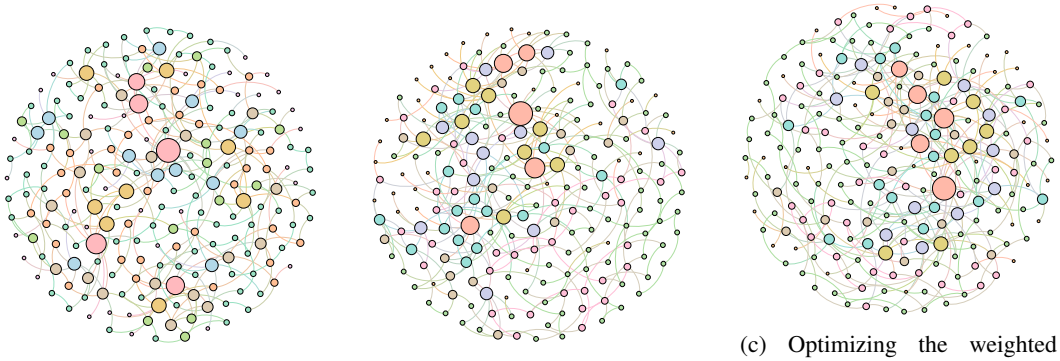
(b) Inductivity on resilience and utility

Figure 9: The inductive ability of ResiNet on the test dataset (BA-20-200) when optimizing (a) network resilience and (b) the combination of resilience and utility.

inforcement learning. We demonstrate the inductive ability of ResiNet on graphs of different size by training ResiNet on the BA-40-200 dataset, which consists of graphs with the size ranging from 40 to 200, and then report its performance on directly guiding the edges selections on unseen test graphs. The filtration order K is set to 1. As shown in Figure 9, we can see that ResiNet has the best performance for $N \in [70, 100]$. The degrading performance with the graph size may be explained by that the larger graphs requires a larger filtration order to work well. If ResiNet is trained to optimize network resilience and utility simultaneously, it has a more stable performance with the increment of graph size, and it possibly find the strategy to balance these two metrics.

C.6.3 LIMITATION

It should be noted that the weighted combination of resilience and utility setting can be a little cheating for other baselines due to the additional RL training process of ResiNet. Despite our implementation of ResiNet is highly tuned, the training of RL is known challenging, and two reward functions with different ranges poses another challenge. Moreover, due to limited computational resources, we only train ResiNet with all non-empty subgraphs in FireGNN on the BA-15 network as a transductive and BA-10-30 network as an inductive setting. For networks with more than 50 nodes, we only use the top-2 subgraphs in the filtration, as we observed to balance the computational



(a) Original EU network (b) Optimizing R (c) Optimizing the weighted combination of 0.5 R + 0.5 GE

Figure 10: Visualizations of the original EU network and optimized networks using ResiNet under different objectives: R means the connectivity-based resilience measurement and GE is the global efficiency.

effort and the satisfying performance. This may explain the performance of ResiNet is slightly weaker than several baselines on larger networks.

Another point is that for ResiNet the maximum episode length is set to 20 steps on all graphs, which means all solutions given by ResiNet require no longer than 20 edge rewiring operations. For example, the optimal solution given by ResiNet on BA-15 only requires a minimum of six steps to arrive at the target graph. A larger filtration order, a longer episode budget, or a powerful RL learning algorithm should be expected to boost the performance of ResiNet further.

Despite the success of ResiNet armed with our proposal FireGNN in both the transductive and inductive settings, we empirically observe that the trained agent only works well on the unseen datasets with the similar setting it is trained. How to further improve the inductivity of ResiNet is left as future work.

C.7 Inspection of optimized network samples

Moreover, to provide a deeper inspection into the optimized network structure, we take EU power network as an example to visualize its network structure and the optimized networks given by ResiNet in terms of different objectives.

Comparing to original EU network, Figure 10 (b) is the network structure obtained by only optimizing the graph connectivity-based resilience. We can observe a more crowded region on the left, which is consistent to the 'onion-like' structure concluded in previous studies.

If we consider the combination gain of both resilience and utility, we observe a more compact clustering "crescent moon"-like structure as shown in Figure 10 (C).

It is worth pointing out that the task-dependent order of node removal will affect the filtration. For example, the scale-free networks lost all connections under the degree-based attack on half nodes on average, while it maintains its topologies under some other attacks. For the network resilience task, we are inspired by the definition of the graph connectivity-based resilience metric, with the adaptive node degree centrality to determine the node removal order. Nodes with larger degrees are first removed and the ties are broken by arbitrarity.