Residual Entropy-based Graph Generative Algorithms

Wencong Liu
Beijing Institute of Technology
Beijing, China
Southeast Institute of Information Technology
Fujian, China

Jiamou Liu
The University of Auckland
Auckland, New Zealand

Zijian Zhang
Beijing Institute of Technology
Beijing, China
Southeast Institute of Information Technology
Fujian, China

Yiwei Liu
Defence Industry Secrecy Examination and Certification Center
China

Liehuang Zhu
Beijing Institute of Technology
Beijing, China

ABSTRACT

Classification and clustering are crucial tasks that recognize the identities and the communities of nodes in a graph. Several methods have been proposed to reduce the accuracy of node classification and clustering through graph neural networks (GNN). Existing defense methods usually modify the model architecture and adopt countermeasure training to enhance the robustness of the node classification and clustering. However, these defense methods are model-oriented and not robust. To alleviate the problem, this paper first proposes a robust node classification metric based on residual entropy. More concretely, we prove that maximizing the residual entropy helps to improve the robustness of the classification accuracy. We them propose two graph generative algorithms to resist against two kinds of GNN-based attacks, the *untargeted* and the *targeted attacks*. Finally, experimental analysis show that the proposed algorithms outperform the existing defense works under five classic datasets. [1]

KEYWORDS

Graph adversarial learning; Graph generative algorithm; Node classification; Residual entropy; Robustness

ACM Reference Format:
Wencong Liu, Jiamou Liu, Zijian Zhang, Yiwei Liu, and Liehuang Zhu. 2022. Residual Entropy-based Graph Generative Algorithms. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 9 pages.

1 INTRODUCTION

Graph mining are crucial techniques to reveal insights about the social structures of multi-agent systems [2, 8, 20, 27–29]. One of the key insights involve subsets of agents who have dense internal and sparse external connections, which are naturally expressed as agent communities [14, 31, 37]. A large number of community detection algorithms have emerged in the last two decades. The classical methods focus on optimizing the detection metrics [1, 10, 34]. Nowadays, researchers usually treat the task of community detection as that of node classification. More specifically, agents in the same community are regarded as having the same label and graph-neural classification models, e.g., graph neural networks (GNN), are trained. Compared with the classical methods, graph-neural methods not only consider the topological relationship but also node information of the network [3, 9, 15, 21, 38].

Although node classification has achieved impressive performance in various fields such as malwares detection [36], analyzing chromosomal domains [25], and inferring node identity [21], existing node classification algorithms are vulnerable and their performance can be easily affected by well-designed attacks [46, 47]. Indeed, a number of recent adversarial algorithms have been designed and shown to be effective [4, 5, 13, 26, 43, 46]. These adversarial algorithms can change the label distribution of the nodes in the graph [13, 30] by inserting or removing several edges [13, 30, 42], once the graph structure is leaked. Since these attacks have caused huge damage to the node classification algorithms, it is useful to study the corresponding defense methods to maintain the robustness of node classification [7].

In response to the attack methods, various defense methods were also proposed. Many of these methods focus on rebuilding the node classification models [6, 12, 35] or adopting adversarial training models [16, 41, 45] to enhance the robustness of classification. Others methods try to design data generative methods [11, 43]. For example, Wu el al. attempted to drop edges among nodes with low similarity to reduce the mis-classification risk for the edges of being attacked [43]. However, most of the data generative methods are model-driven and do not exhibit sufficient robustness, meaning that they become invalid when the adversarial model changes or the attackers modifies the graphs structure later.

Our motivation in this paper is to design a general and robust graph generative metric. More clearly, such a metric is expected to preserve the accuracy of node classification under two constraints: (1) The metric is independent from the adversarial model, and (2) the attackers may change the graph structure later arbitrarily. With such a metric, we may address problems with existing defence methods to enhance robustness in a general setting.

There are several challenges that we must overcome. On one hand, there is a lack of a commonly-agreed and mature metric to guide the designing of defense algorithms. In terms of the adversarial goal, there exist two kinds of attacks, *untargeted attack* and *targeted attack* [18]. The former attempts to mis-classify the entire

[1]Corresponding author: Zijian Zhang. This paper is supported by National Natural Science Foundation of China No. 62172040, No.61872041, No.U1836212.

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online.

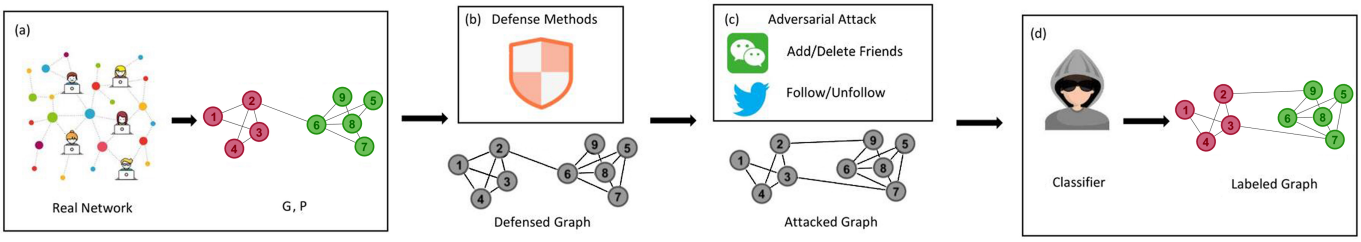

Figure 1: The process of designing graph generative algorithms to enhance the robustness of a community structure. (a) Model a real-world network to a graph G with community structure P. (b) Apply the graph generative algorithms to make the community structure P more robust, i.e. add edges $\{1, 4\}$ and $\{5, 7\}$. (c) When G is attacked, attackers arbitrarily modify the real-world relationship by adding or deleting edges, i.e. add edges $\{2, 9\}$, $\{3, 7\}$, and delete edges $\{1, 2\}$, $\{2, 6\}$. (d) Classifiers are used to classify nodes on the attacked graph. The accuracy of the node classification is similar with that of G.

set of nodes in a graph, while the latter aims to mis-classify only a specific set of nodes. Further, the potential strategies to execute either goal on graph vary with the graph structures. It seems impossible to explore a general defense metric by passing through all the concrete attacks. On the other hand, the designed concrete defense algorithms have to be used to support both traditional machine learning-based and deep learning-based node classification models like graph neural network.

In order to solve the aforementioned challenges, it is important to learn the amount of information for community structure. Li and Pan proposed the concept of structural entropy as a general tool in structural information theory [24]. Li et al. subsequently used this notion in community detection [22]. Liu el al. further applied residual entropy to measure the information of community structure, and asserted that higher residual entropy lead to more stable community structure [30] is. These work inspired us to apply residual entropy to explore the defense metric. In order to keep the existing topological relationship of the original graph, we only add new edges among nodes. The idea is to insert new edges with the maximal residual entropy. We verify this idea using two tasks. The primary task is to verify whether maximizing residual entropy can improve the performance of node classification. The secondary task is to check whether it can defend both untargeted and targeted attacks. The contributions are summarized as follows:

- We propose a new residual entropy-based graph generative metric. We demonstrate that maximizing the residual entropy can improve the robustness of community structure, thereby defending against various adversarial attacks.
- We design two graph generative algorithms REGG_UA and REGG_TA. Both of them are independent from adversarial attacks. Further, we optimize the execution efficiency of those algorithms.
- We verify our algorithms against several adversarial attacks on five datasets and seven classifiers. Experimental results reveal the effectiveness of our algorithms. Specifically, the algorithm REGG_UA improves the performance of node classification and can effectively defense untargeted attacks, and REGG_TA is effective in resisting targeted attacks.

2 RELATED WORKS

Node classification has been studied for several years. The initial research focused on community detection in social networks. These researches usually defined a metric, and achieved node clustering by optimizing this metric. For example, the algorithm Louvain is a traditional unsupervised method by optimizing modularity [1], and the algorithm Scalable Community Detection (SCD) is designed by maximizing a Weighted Community Clustering (WCC) [33]. After that, Perozzi el al. [32] proposed a node embedding method named DeepWalk, which can capture connectivity patterns in networks by representation learning. Moreover, this method can also be combined with machine learning algorithms, such as logistic regression [32]. Recently, the artificial neural network has been used for node classification. Kipf el al. [21] proposed a representative Graph Convolutional Network (GCN) model. Petar el al. [39] proposed a neural network architectures called Graph Attention Network (GAT) following a self-attention strategy to compute the hidden states of each node by attending over its neighbors in the graph. Wang el al.[40] combined Label Propagation (LPA) with GCN to propose the GCN-LPA model. Mohammand el al. [17] proposed Semi-Supervised Preconditioning (SSP) to employ information-geometric tools to optimize graph neural networks, which achieved excellent performance on node classification.

With the wide application of node classification in reality, researches of adversarial attacks on graph have been paid more attention [46]. Based on whether the adversary's target is the whole graph or parts of the graph, these attacks can be divided into untargeted attack and targeted attack. As a method of untargeted attack, Liu el al. [30] proposed an efficient attack strategy Residual Entropy Minimization (REM) from the view of information theory. Disconnect Internally Connect Externally (DICE) [42] was another method of untargeted attack, which worked by randomly deleting internal edge and adding external edge for each community. For targeted attack, Fionda el al. [13] devised Community Deception via Safeness(FDA) for a targeted set of nodes by rewiring a certain number of edges related with the targeted set through the metric safeness. Li el al. [26] proposed an iterative learning framework Community Detection ATTACKer(CD-attack) to modify the edges of graph, which took turns to update two modules: one working as the constrained graph generator and the other as the surrogate

community detection model. Zügner el al. [46] designed an algorithm named Nettack to generate unnoticeable structure and feature perturbations on attributed graphs by preserving graph's degree distribution and features co-occurrences. RND is another bench mark attack by randomly sample nodes and connect them with other nodes of different labels [46].

Corresponding to the graph adversarial attacks, there are also some studies on graph defense. One type of defense is to improve the robustness of classification by improving the classifier, such as, directly modifying the classifier [16, 41, 45] and adopting adversarial training to the classification model [6, 12, 35, 44]. Another type of defense is to improve the robustness of graph data, such as, considering the graph generative defense algorithms [11, 43]. Specifically, Wu el al. designed a defense algorithm by dropping edges among nodes with low similarity score to reduce the attacked risk of edges [43]. They tested the performance of the algorithm on a limited number of nodes and a specific classifier GCN. Entezari el al. [11] observed that Nettack [46] results in changes in high-rank spectrum of the graph. They propose to preprocess the graph with its low-rank approximations. Their algorithm depends on linearly modifying the adjacency matrix of the graph instead of directly add or drop edges in graph and may not be able to counteract attacks other than Nettack. However, most of the existing works have not mainly studied how to generate universal and robust graphs. Thus, in this paper, we focus on designing graph generative algorithms to enable graphs to prevent the potential attacks by pre-protecting the structure information of the graph data. In order to provide theoretical support, we introduce the relevant knowledge of information theory. In information theory, quantifying structural information is essential [29]. Li el al. first introduced the structural entropy of networks, and they used it to detect community structure in the network [23]. As a well-defined measure for the security of networks, Li el al. also proposed the notion of resistance based on the one-dimensional and two-dimensional structural entropy of networks, respectively [22]. Following by those definitions, Liu el al. introduced the notion of residual entropy [30]. A higher value of residual entropy means more information of the community structure. This motivates us to introduce residual entropy to learn the universal and robust graph generative metric.

3 PROBLEM FORMULATION

A network of agents can be described by a graph $G = (V, E)$, where the node set $V = \{v_1, v_2, ..., v_n\}$ is the set of agents, the edge set E denotes the relationships among agents in V. These nodes spontaneously form a community (or label set) according to the closeness of the connections and the agents' attributes. Suppose $P = \{X_1, X_2, ..., X_L\}$ is the community structure of $G = (V, E)$. This structure can be used to study sub-structural properties of agents V, such as, common preferences or intentions. Therefore, it is necessary for the network publisher to protect this structure P from being attacked. As we described above, in this paper, we focus on a preprocessing-based defense on raw graph. That is, given a undirected graph $G = (V, E)$ and its community structure P, our goal is to look for a defense strategy $\mathcal{D}$ such that $\mathcal{D}(G) = \hat{G}$ is robust in detecting P, where $\mathcal{D}(G)$ is a generated graph from raw graph. Suppose $\mathcal{A}$ is the attack strategy and $\mathcal{F}$ is the classifier, the preprocessing-based defense means $\mathcal{F}(\mathcal{A}(\hat{G}))$ resembles P as much as possible. Figure 1 shows a workflow of the preprocessing-based defense on G. It can be see that the node classification result $\mathcal{F}(\mathcal{A}(\hat{G}))$ after the attack is consistent with the initial community structure P.

Given the graph $G = (V, E)$, initial community structure P, attack strategy $\mathcal{A}$ and classifier $\mathcal{F}$, assuming that $\mathcal{S}$ is the similarity function between two community structure, then the preprocessing-based defense problem can be defined by

$$\arg\max_{\mathcal{D}} \mathcal{S}(\mathcal{F}(\mathcal{A}(\mathcal{D}(G))), C) \tag{1}$$

where C is the initial community structure P if we protect all communities $\{X_1, X_2, ..., X_L\}$ and C is X_t if we protect the target community X_t. The operation of defense strategy $\mathcal{D}$ includes adding or deleting nodes and edges. For simplicity, in this paper, we only consider the case of adding edges. Nevertheless, it still takes exponential time to greedily find the optimal defense strategy for edge addition, so we introduce residual entropy in the next section to guide the edge addition to achieve an approximate effect.

4 RESIDUAL ENTROPY

Li el al. first defined the k-dimensional structure entropy for a graph G [24]. Along with the idea, Liu el al. further investigated the notion of residual entropy to normalize the difference between one-dimensional and two-dimensional structure entropy for community deception [30]. They pointed out that a higher value of residual entropy can represent more information of the community structure. In this paper, we try to extend this concept to create a new general and robust metric to defense both of the untargeted attacks and the targeted attacks. Before giving the concrete definition of this metric, we first recall 1-dimensional and 2-dimensional structure entropy.

Let $G = (V, E)$ be an undirected graph. Denote by d_i the degree of node $v_i \in V$, and $|E|$ the number of edges in E. Each node can be naturally considered as an agent which interacts with others by passing random events through edges. The distribution of agents' interactions means that $\frac{d_i}{2|E|}$ is the probability that agent v_i is involved in a random interaction. Thus the agent can be encoded by a codeword with $-\log_2 \frac{d_i}{2|E|}$ bits. In this sense, 1-dimensional structure entropy is defined by the average codeword length of all interactions from nodes in V [30]:

$$\mathcal{H}(G) = -\sum_{v_i \in V} \frac{d_i}{2|E|} \log_2 \frac{d_i}{2|E|}. \tag{2}$$

The value of $\mathcal{H}(G)$ expresses the average information of a call in G without assuming any community structure. Now assume that $P = \{X_1, X_2, ..., X_L\}$ is the community partition or label partition of V, where L is the number of communities or labels. Write v_j for the volume of X_j (i.e., the sum degree of nodes in X_j), and g_j as the number of edges with only one endpoint in X_j. Then there are two cases to encode an interaction from v_i: 1) we encode the interaction as $-\log_2 \frac{d_i}{v_j}$ if it is from $v_i \in X_j$ to another node in X_j; 2) we encode the interaction as $-\log_2 \frac{d_i}{2|E|} = -\log_2 \frac{d_i}{v_j} - \log_2 \frac{v_j}{2|E|}$ if the call is from $v_i \in X_j$ to a different community. Compared with case 2), case 1) omits the information of the community $-\log_2 \frac{v_j}{2|E|}$.

Then the 2-dimensional structure entropy is defined by the average encode length of all calls with community partition P [30]:

$$\mathcal{H}_P(G) = \sum_{j=1}^{L} \left(-\frac{v_j}{2|E|} \mathcal{H}(G|X_j) - \frac{g_j}{2|E|} \log_2 \frac{v_j}{2|E|} \right), \tag{3}$$

where $\mathcal{H}(G|_{X_j}) = -\sum_{v_i \in X_j} \frac{d_i}{v_j} \log_2 \frac{d_i}{v_j}$. The value of $\mathcal{H}_P(G)$ reflects the average information of a call in G if we know G's community structure P. Therefore, their difference $\mathcal{H}(G) - \mathcal{H}_P(G)$ measures the gained information as the communities in P take shape. Then the normalization of this difference is defined as residual entropy.

Definition 1. *[30] The* normalized residual entropy of $\mathcal{P}$ *is*

$$\rho_{\mathcal{P}}(G) \coloneqq R_P / \mathcal{H}(G), \tag{4}$$

where $R_P = \mathcal{H}(G) - \mathcal{H}_P(G) = \frac{v_j - g_j}{2|E|} \log_2 \frac{v_j}{2|E|}$.

The normalized residual entropy of G [30] indicates the amount of information about the community structure of the graph G. In theory, a relatively high $\rho_P(G)$ represents that the partition P contains more information, which makes the community structure of G easier to be detected. Hence, the defense strategy is to modify the graph structure to generate a new graph with higher residual entropy. The new graph takes an explicit community structure to achieve the effect of defensing adversarial attacks.

We first start to discuss what strategy can be adopted to increase the residual entropy. From the definition, $\rho_P(G)$ increases as long as $\mathcal{H}(G)$ decreases and $R_P(G)$ increases. In our defense strategy, we enhance the graph by adding edges. Specifically, we denote a pair of nodes (u_i, u_j) as a *non-edge* where the node u_i and u_j are disconnected. From the definition of resistance [22], we prove the next lemma.

Lemma 1. *For two non-edges* $(u_i, u_j)\&(u_i, u_w)$*, if* $u_i \in C_m$ *and* $u_j \in C_m$ *and* $u_w \in C_n$ *where* $m \neq n$*, then* $R_P(G \oplus (u_i, u_j)) > R_P(G \oplus (u_i, u_w))$.

It can be observed that adding a non-edge inside a single sub-community can make the value of $R_P(G)$ higher, when comparing with adding a non-edge between two different sub-communities. Similarly, from (4), adding any edge inside a single sub-community has the same effect to the value of R_P. Hence, we focus on how to add edges inside a single sub-community to make $\mathcal{H}(G)$ smaller.

From the formula of structure entropy, another lemma can be derived as below.

Lemma 2. *[[30]] For two non-edges* (u_i, u_j), $\&(u_x, u_y)$*, if* $\min(d_i, d_j) \leq \min(d_x, d_y)$ *and* $d_i + d_j \leq d_x + d_y$ *then* $\mathcal{H}(G \oplus (u_i, u_j)) \geq \mathcal{H}(G \oplus (u_x, u_y))$.

In addition, two corollaries can be derived as below.

Corollary 1. *If* $d_i + d_j < d_x + d_y$*, then* $\mathcal{H}(G \oplus (u_i, u_j)) \geq \mathcal{H}(G \oplus (u_x, u_y))$.

Corollary 2. *If* $d_i + d_j = d_x + d_y$ *and* $|d_i - d_j| \geq |d_x - d_y|$*, then* $\mathcal{H}(G \oplus (u_i, u_j)) \geq \mathcal{H}(G \oplus (u_x, u_y))$.

We can deduce from Lemma 2 and the corollaries above that if we adopt edge addition to make the structural entropy of the graph smaller, we should tend to choose a non-edge with a larger sum of degrees at both ends and a smaller difference between the degrees.

Through the previous analysis, we can conclude that adding edges in sub-communities of the graph can affect $R_P(G)$ and $\mathcal{H}(G)$. Meanwhile, it is able to increase $\rho_P(G)$. Therefore, we can make the residual entropy larger than the original graph through adding edges. Consequently, the key question is changed to designing an efficient edge choice algorithm. In the next section, we will discuss the algorithm in detail.

5 RESIDUAL ENTROPY BASED DEFENSE ALGORITHMS

In this section, we adopt the strategy of directly adding the edges in the graph instead of modifying node feature. As mentioned earlier, the key point is to adding as few as edges to the graph and get a more robust graph structure. We will first give the definition of the candidate edge:

Definition 2. *For a sub-community* C_i *of the graph and a non-edge* (u_x, u_y) *in* C_i*, we call* (u_x, u_y) *an* aim edge *if* $d_x + d_y$ *is the maximum among all non edges in* C_i*. In all aim edges of* C_i*,* (u_x, u_y) *is the* candidate edge *if* $|d_x - d_y|$ *is the minimum.*

From the previous lemmas and the definitions about candidate edge, we can infer that comparing with randomly choose a non-edge in a single sub-community, choosing to add the candidate edge to the graph can better increase the residual entropy.

Next we will introduce the robust graph generative algorithms. Since there are two categories of attacks on graph: untargeted attack and targeted attack, we design two defense algorithms based on residual entropy, respectively. The specifications of both algorithms are listed as follows.

The *residual entropy-based graph generative algorithm for untargeted attack* (REGG_UA) is proposed below. This algorithm contains three steps. The first step is to select the candidate edge in each sub-community to form a candidate edge set. The second step is to select a best candidate edge from the candidate edge set. In this step, we traverse the entire set in line 10 of the procedure, and calculate the change of the residual entropy when adding each edge to the graph, and finally choose the one that maximizes the residual entropy. The third step is to add this edge into the graph and update the candidate edge set. We suppose the best candidate edge is in the sub-community C_i. In the step of updating the candidate edge set in line 18, we first remove the best candidate edge from the set and then add a new candidate edge to the set from the sub-community C_i. After repeating the whole three steps for K times (K is a hyper-parameter), a robust graph to resist against the untargeted attack is generated. When implement this algorithm, we sort the non edges inside each sub-community by the sum of degrees of both nodes and save the results in advance. This is used to enable the algorithm to quickly calculate the candidate edge in each sub-community. Let $|P|$ be the number of sub-communities in the graph, then the compute complexity of the REGG_UA is $O(|P||V| \log_2 |V|)$, which is faster than greedy strategy to select candidate edges in the entire graph with the compute complexity of $O(|V|^2)$.

The *residual entropy-based graph generative algorithm for targeted attack* (REGG_TA) is defined similarly to the REGG_UA. The REGG_TA algorithm can be divided into two steps. The first step in

line 25 is to calculate the candidate edge in the aim sub-community(aim sub-community is known). The second step in line 26 is to add this edge to the graph. The two steps are repeated for T times (T is a hyper-parameter) as well.

Algorithm REGG_UA / REGG_TA

Input: Graph G, Partition P, K, T, Aim Community X_c
Output: Defensed Graph G

1: **function** REGG_UA(Graph G, Partition P, K)
2: Initialize *candidate edge set* as {}
3: **for** $X_i \in P = \{X_1, X_2, ..., X_L\}$ **do**
4: find a candidate edge (u_x, u_y) from X_i
5: add (u_x, u_y) to *candidate edge set*
6: **end for**
7: **while** $K > 0$ **do**
8: max residual entropy = $-\infty$
9: best candidate edge = (None, None);
10: **for** $(u_i, u_j) \in$ *candidate edge set* **do**
11: re = $CalculateResidualEntropy(G \oplus (u_i, u_j))$
12: **if** re > max residual entropy **then**
13: max residual entropy = re
14: best candidate edge = (u_i, u_j)
15: **end if**
16: **end for**
17: add best candidate edge to G
18: update *candidate edge set*
19: $K = K - 1$
20: **end while**
21: **return** G
22: **end function**
23: **function** REGG_TA(Graph G, Partition P, T, Aim Community X_c)
24: **while** $T > 0$ **do**
25: calculate the candidate edge (u_x, u_y) from X_c
26: add the candidate edge to G
27: $T = T - 1$
28: **end while**
29: **return** G
30: **end function**

6 EXPERIMENTAL EVALUATION

In this section, we conduct experiments to evaluate the effectiveness of REGG_UA and REGG_TA on the following three tasks mentioned as before.

- First, we test the performance of REGG_UA on the task of improving the performance of node classification through different classifiers.
- Second, we test the performance of REGG_UA on the task of defending different untargeted attacks.
- Finally, we test the performance of REGG_TA on the task of defending different targeted attacks.

Here, we first describe the experimental settings.

6.1 Experimental Settings

Datasets: Similar with [19, 46], we adopt five benchmark datasets, including one blog graph Polblogs and one email data graph Email, and three citation graphs, i.e., Cora, Citeseer, Pubmed. We show the statistics of the datesets in Table 1. Node features are not available both in Email and Polblogs. In this case, we test them on classifiers that do not require node features.

Table 1: Statistics of datasets. For each graph, $|V|$ is the number of nodes and $|E|$ is the number of edges.

	$\|V\|$	$\|E\|$	Classes	Features
Cora	2485	5278	7	1433
Citeseer	3312	4536	6	3703
Pubmed	19717	44324	3	500
Polblogs	1491	16715	2	/
Email	1134	5451	12	/

Classifiers: We validate our proposed algorithms on Email, Polblogs and Cora through three node classification classifiers as following. All of them do not require node features in the graph.

- **Louvain** [1]: One of the fastest and most effective unsupervised community detection algorithms. It has been shown to perform very well in comparative benchmark tests.
- **DeepWalk** [32]: A classic node representation method which can be combined with traditional machine learning models such as Logistic Regression on the node classification methods.
- **SCD**(Scalable Community Detection) [33]: A novel disjoint community detection algorithm by maximizing the Weighted Community Clustering(WCC).

We then test our proposed algorithms on Cora, Citeseer, Pubmed through the following four graph neural network models. All of them require node features in the process of training.

- **GCN** [21]: The most representative one in the existing Graph Convolutional Networks (GCN) models.
- **GAT** [39]: Comparing with GCN, Graph Attention Network (GAT) is composed of attention layers which can learn different weights for different nodes in the neighborhood.
- **SSP**(Semi-Supervised Preconditioning) [17]: The author proposed SSP to develop optimization algorithms for the graph-based semi-supervised learning by employing the natural gradient information in the optimization process, which achieves good performance in node classification task.
- **GCN-LPA** [40]: GCN-LPA uses Label Propagation (LPA) to assist the GCN in learning proper edge weights that lead to improved classification performance.

Parameter settings: For the first and second task, we set hyper-parameters K as $p|E|$ for each dataset, where p is called the *defense ratio*. The budget of updates for untargeted attack methods is the same value K for each graph. For the third task, we randomly choose one as the attacked sub-community for each dataset and set hyper parameters T as $p|X_c|$, where $|X_c|$ is the number of edges on the nodes of the target sub-community. The budget of updates for untargeted attack on the target sub-community is also the same

value T for each graph. For all node classification model, we adopt the default parameter setting in the author's implementation. For the algorithms with random results like Louvain, we report the average performance of 10 runs.

Baselines: We first investigate the existing graph generative algorithms. Wu el al. [43] proposed a defense method based on deleting edges where nodes at both nodes of an edge have a low feature similarity. To validate the defense performance of this strategy, we delete all edges that feature similarity is 0 in the dataset Cora according to the proposed defense strategy. Then we test the classification performance on GCN [21] with the default parameter setting and show the result in Table 1. From Table 2 we can observe that the performance of node classification on the defensed graph decreases comparing with the original graph. Because the author only test this strategy on 40 nodes, the results show that this defense method is not suitable to be extended on the whole graph.

Table 2: Accuracy (%) of GCN on dataset Cora with/without the defense[43].

Dataset	Original	Defensed
Cora	83.71	82.75

Entezari el al. [11] proposed a defense algorithm that keeps only the high-rank components of the adjacency matrix of the graph during model training. Dropping the rank of the adjacency matrix cannot be mapped to the modification of graph edges. The method also does not work for some classifiers that do not require an adjacency matrix. Therefore, the above two existing preprocessing methods are not suitable for baselines. In order to compare our defense algorithms REGG_UA and REGG_TA with other baselines, we replaced the process of calculating candidate edge in REGG_UA and REGG_TA with randomly selecting an edge. Then we get two variant algorithms REGG_UA_r and REGG_TA_r for the comparative experiments. We will keep parameter K consistent in REGG_UA and REGG_UA_r, and parameter T is also the same in REGG_TA and REGG_TA_r.

6.2 The Performance of Node Classification

In this subsection, we show the results of REGG_UA and REGG_TA on the task of improving the performance of node classification. We use the Normalized Mutual Information(NMI) to evaluate the similarity between the results of classifiers and the true community structure P. When the community structure $P' = \{X'_1, X'_2, ..., X'_J\}$ detected by classifiers is known, we can calculate the NMI between the P and P' according to the following formula

$$NMI(P, P') = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J} P(i,j)\log\frac{P(i,j)}{P(i)P(j)}}{\sqrt{H(P)H(P')}}. \quad (5)$$

Assume the partition P and P' have I and J clusters respectively. $P(i)$ is the probability that a randomly selected node from the graph falls into the sub-community X_i in partition P. $P(i, j)$ denotes the probability that an node belongs to the sub-community X_i in P and the sub-community X'_1 in P'. $H(P)$ is the entropy associated with all probabilities $P(i)$ in partition P $(1 < i < I)$. And $H(P)$ can be represented by

$$H(P) = \sum_{i=1}^{I} |X_i| \log \frac{|X_i|}{|V|} \quad (6)$$

As a result of the meaning of mutual information, larger NMI reflects better performance of the classifier.

We first evaluate how REGG_UA and REGG_UA_r behaves under different defense ratios from 0% to 50% with a step size of 10%. On the classifiers that do not require node features, we show the performance of two algorithms REGG_UA and REGG_UA_r on Email, Polblogs, and Cora in Table 3. We then show the performance of the two algorithms on the three datasets of Cora, Citeseer, and Pubmed for the four graph neural network models GCN, GAT, SSP, and GCN-LPA in Table 4.

It can be seen from Table 3 that on the three datasets, REGG_UA improves the effect of node classification on classifiers that do not require features, while REGG_UA_r maintains almost the same effect or even decreases in most cases. In Table 4, REGG_UA still performs better than REGG_UA_r on the graph neural network

Table 3: The performance (NMI * 100) for the graph defensed by REGG_UA and REGG_UA_r on Louvain, DeepWalk, SCD.

Dataset	p(%)	Louvain		DeepWalk		SCD	
		REGG_UA	REGG_UA_r	REGG_UA	REGG_UA_r	REGG_UA	REGG_UA_r
Email	0	42.23	42.23	66.11	66.11	49.86	49.86
	10	**47.49**	34.39	**80.90**	72.90	**51.53**	49.09
	20	**51.28**	34.40	**82.62**	76.97	**53.43**	50.03
	30	**52.22**	31.41	**86.20**	77.41	**53.90**	50.20
	40	**54.11**	31.88	**87.30**	80.16	**55.52**	51.09
	50	**55.37**	31.12	**90.77**	81.55	**56.06**	51.36
Polblogs	0	89.50	89.50	56.99	56.99	57.02	57.02
	10	**92.79**	87.94	**63.83**	55.43	**63.42**	58.11
	20	**93.06**	87.72	**63.85**	55.27	**64.08**	58.10
	30	**93.34**	87.86	**64.15**	55.49	**65.57**	58.47
	40	**94.79**	88.05	**64.26**	55.10	**67.93**	58.39
	50	**96.57**	87.86	**64.36**	55.21	**78.85**	58.78
Cora	0	15.16	15.16	51.14	51.14	38.94	38.94
	10	**23.44**	13.87	**71.80**	52.45	**39.81**	39.48
	20	**27.11**	13.69	**77.49**	52.80	**40.56**	40.08
	30	**31.80**	13.48	**84.17**	53.41	**41.26**	40.31
	40	**34.27**	13.28	**87.40**	53.46	**42.13**	40.49
	50	**40.74**	12.56	**92.84**	53.39	**43.50**	40.51

Table 4: The performance (NMI * 100) for the graph defensed by REGG_UA and REGG_UA_r on GCN, GAT, SSP, GCN-LPA.

Dataset	p(%)	GCN		GAT		SSP		GCN-LPA	
		REGG_UA	REGG_UA_r	REGG_UA	REGG_UA_r	REGG_UA	REGG_UA_r	REGG_UA	REGG_UA_r
Cora	0	68.10	68.10	66.11	66.11	65.44	65.44	73.90	73.90
	10	**72.05**	66.26	**72.67**	69.49	**72.28**	66.79	77.67	78.45
	20	**74.70**	66.22	**74.03**	70.01	**77.71**	66.68	76.48	77.45
	30	**76.62**	66.09	**77.09**	69.40	**79.67**	66.75	**77.20**	76.83
	40	**78.73**	66.13	**77.38**	69.76	**82.44**	66.62	76.53	76.82
	50	**81.00**	66.01	**80.24**	69.42	**86.25**	66.62	**79.73**	77.64
Citeseer	0	50.65	50.65	47.99	47.99	55.62	55.62	56.49	56.49
	10	**55.13**	50.94	**55.59**	50.44	**56.16**	50.21	**60.56**	50.25
	20	**60.20**	54.31	**62.12**	53.61	**62.88**	55.40	**61.09**	49.62
	30	**65.70**	57.25	**67.92**	56.25	**69.55**	58.62	**64.57**	63.83
	40	**70.40**	58.12	**73.56**	57.55	**71.81**	58.42	**65.27**	63.60
	50	**74.35**	57.95	**78.91**	58.69	**75.46**	58.37	**67.14**	63.35
Pubmed	0	48.17	48.17	49.25	49.25	51.97	51.97	37.59	37.59
	10	**55.13**	50.94	**55.59**	50.44	51.99	52.45	**37.93**	37.51
	20	**58.35**	52.60	**59.64**	52.18	52.52	53.96	**39.71**	38.39
	30	**60.20**	54.31	**62.12**	53.61	**58.40**	57.70	**42.92**	27.52
	40	**63.55**	55.89	**66.90**	55.70	**58.46**	58.39	**41.83**	29.21
	50	**65.70**	57.25	**67.92**	56.25	**60.40**	59.59	**42.72**	28.87

models. The results show that the graph after defensed basing on REGG_UA has more information about the community structure, which makes it easier for classifiers to detect the community structure.

6.3 Defending Untargeted Attack

In this section, we compare the performance of node classification on the original graph with the defended graph when the two graphs are attacked by untargeted attack methods. We set defense ratio as 50% then $K = 0.5|E|$. For untargeted attack, we choose the two following methods samely based on modifying edges in graph.

- **REM** [30]: An efficient attack strategy through adding edges to the graph, which is designed by residual entropy from the view of information theory.
- **DICE** [42]: It's a attack method which randomly choose whether to insert or remove an edge. Edges are only removed between nodes from the same classes, and only inserted between nodes from different classes.

First, we studied the defensive effects of REGG_UA and REGG_UA_r on the attack algorithms REM and DICE. For each dataset, we use REM and DICE to attack three corresponding graphs: the original graph, the graph defensed by REGG_UA, and the graph defensed by REGG_UA_r. Then we get six attacked graphs, and we compare the final classification results of them and the unattacked orginal graph together through different classifiers. We also use NMI as an evaluation indicator. Table 5 shows the results of the classifier that does not require node features, and Table 6 shows the results of the graph neural network models.

We can see from Table 5 that the performance of orginal graph has dropped significantly when attacked by REM or DICE. Compared with the attacked original graph, the performance of the graph defensed by REGG_UA still maintain a higher value. It even performs better than the unattacked orginal graphs in almost all cases. And we also notice that REGG_UA_r also has a certain defensive effect, but the performance is unstable due to its random stratedy. For the graph neural network models in Table 6, we can see that REGG_UA and REGG_UA_r still maintains the same conclusion as before. The performance of the graph defensed by REGG_UA performs worse than the unattacked orginal graphs on the dataset cora in some cases. We can infer that the classifiers uses node features as the input of the model, so its performance is not only affected by the graph structure.

6.4 Defending Target Attack

Next we will concentrate on the effect of REGG_TA and REGG_TA_r for defending targeted attack with the same budget T. Similar to the previous section, we set defense ratio as 50% then $T = 0.5|X_c|$. For targeted attack, we also choose the two following methods which only attack a certain sub-community of the graph.

- **FDA**[13]: It introduce a measure to quantify the level of hiding of a target sub-community in the graph and present an efficient algorithm to hide the target sub-community.
- **RND** [46]: It is an attack based on modifying the structure of the graph by a random strategy. Given a target node v,in each step it randomly sample nodes u whose label is different from v and add the edge u, v to the graph structure.

Since the classifiers Louvain, Deepwalk, and SCD are all unsupervised algorithms, in order to measure the classification effect of unsupervised classifiers on a specific sub-community, we first define a new classification evaluation parameter NMI_part for a sub-community based on the theory of information. When the specific sub-community X_c of the original graph and the community structure P' detected by classifiers are given, the definition of NMI_part can be represented by

Table 5: Node classification performance ($NMI \times 100$) for the original graph, and three graphs attacked by REM or DICE: the original graph, the graph defensed by REGG_UA and the graph defensed by REGG_UA_r on Louvain, DeepWalk, SCD.

Dataset	Attack	Graph	Louvain	DeepWalk	SCD
Email	NO	original	42.23	66.11	49.86
	REM	original	17.52	63.48	47.98
		REGG_UA_r	27.40	80.86	52.10
		REGG_UA	**44.13**	**89.84**	**56.68**
	DICE	original	21.52	49.91	48.32
		REGG_UA_r	25.50	64.35	51.62
		REGG_UA	**42.58**	**81.66**	**55.29**
Polblogs	NO	original	89.50	56.99	57.02
	REM	original	68.04	55.68	67.34
		REGG_UA_r	67.30	46.98	70.70
		REGG_UA	**74.41**	**69.68**	**77.87**
	DICE	original	67.39	65.77	51.76
		REGG_UA_r	69.09	66.85	44.65
		REGG_UA	**82.21**	**78.15**	**60.37**
Cora	NO	original	15.16	51.14	38.94
	REM	original	5.30	42.58	23.97
		REGG_UA_r	8.84	48.84	35.99
		REGG_UA	**27.97**	**89.82**	**48.88**
	DICE	original	16.44	38.74	30.52
		REGG_UA_r	15.85	39.51	33.17
		REGG_UA	**25.11**	**86.91**	**51.32**

Table 6: Node classification performance ($NMI \times 100$) for the unattacked original graph, and three graphs attacked by REM or DICE: the original graph, the graph defensed by REGG_UA and the graph defensed by REGG_UA_r on GCN, GAT, SSP, GCN-LPA.

Dataset	Attack	Graph	GCN	GAT	SSP	GCN-LPA
Cora	NO	original	68.10	66.11	65.44	73.90
	REM	original	42.31	44.19	35.32	66.89
		REGG_UA_r	48.41	54.30	47.50	73.04
		REGG_UA	**58.58**	**63.91**	**61.61**	**77.04**
	DICE	original	41.09	43.08	21.42	70.73
		REGG_UA_r	42.23	46.00	22.33	73.63
		REGG_UA	**63.88**	**64.72**	**62.59**	**79.63**
Citeseer	NO	original	50.65	47.99	55.62	56.49
	REM	original	31.95	30.88	28.78	47.05
		REGG_UA_r	48.52	46.74	53.73	57.48
		REGG_UA	**53.25**	**48.74**	**59.72**	**64.50**
	DICE	original	31.81	31.50	22.26	56.06
		REGG_UA_r	45.59	42.83	32.97	52.02
		REGG_UA	**50.52**	**49.90**	**52.47**	**59.71**
Pubmed	NO	original	48.17	49.25	51.97	37.59
	REM	original	38.92	37.85	42.60	29.94
		REGG_UA_r	45.91	34.85	44.66	20.15
		REGG_UA	**52.43**	**57.85**	**48.44**	**33.95**
	DICE	original	41.11	40.11	43.93	40.50
		REGG_UA_r	38.21	39.21	50.05	17.43
		REGG_UA	**50.18**	**54.18**	**51.92**	**38.60**

$$H(X_c) = -\frac{|X_c|}{|V|}\log\frac{|X_c|}{|V|} \tag{7a}$$

$$H(X_c|P') = -\sum_{j=1}^{J}\frac{|X_c \cap X'_j|}{|V|}\log\frac{|X_c \cap X'_j|/|V|}{|X_j|/|V|} \tag{7b}$$

$$NMI\_part = (H(X_c) - H(X_c|P'))/H(X_c) \tag{7c}$$

$H(X_c)$ can represent the amount of uncertainty to encode sub-community X_c in the original graph. And $H(X_c|P')$ can represent the residual uncertainty if the community structure P' detected by classifiers is known. Then the metric NMI_part can be defined as $(H(X_c) - H(X_c|P'))/H(X_c)$, which reflects the amount of existed information for X_c in P'. A higher NMI_part means a better classification effect for X_c.

For each dataset we first randomly choose a sub-community as the aim sub-community. Similar to the untargeted attack, we use FDA and RND to attack the aim sub-community in three corresponding graphs: the original graph, the graph defensed by REGG_TA, and the graph defensed by REGG_TA_r. Then we calculate NMI_part for the aim sub-community of the three attacked graphs above and the aim sub-community of the unattacked original graph. We show the results for the classifiers that does not require node features in the Table 7 and the results of the graph neural network models in the Table 8.

From the results of the two tables, we can see that the performance on the attacked aim sub-community decreases a lot comparing with the aim sub-communitiy in the original graph. For unsupervised classifiers, the aim sub-community becomes more robust when defensed by REGG_TA and its NMI_part only has a small drop. For the graph neural network models, the performance of the aim sub-community defensed by REGG_TA even performs better than the original aim sub-community in many cases. In contrast, REGG_TA_r performs worse than the original aim sub-community or only shows a very weak defense effect, which indicated that the strategy of REGG_TA is more effective.

7 CONCLUSION

In this paper, we first explore a new graph generative metric to reduce the effect of graph adversarial attacks when the information of the graph is leaked. We then proposed two graph generative algorithms based on the concept of residual entropy for untargeted attack and targeted attack. We also analyze the performance of proposed algorithms with the existing works on different classifiers and datasets.

In the future, we will attempt to consider the node features and design more robust defense algorithms through pre-processing on the raw graph. We wish the features could further improve the performance of the defense algorithms.

Table 8: Node classification performance ($NMI\_part\times100$) for the unattacked original aim sub-community, and the aim sub-community in three graphs attacked by FDA or RND: the original graph, the graph defensed by REGG_TA and the graph defensed by REGG_TA_r on GCN, GAT, SSP, GCN-LPA.

Dataset	Attack	Graph	GCN	GAT	SSP	GCN-LPA
Cora	NO	original	57.90	62.37	62.68	69.35
	FDA	original	42.19	49.03	49.39	22.08
		REGG_TA_r	50.65	60.73	53.15	31.16
		REGG_TA	**59.05**	**63.39**	**57.47**	**43.99**
	RND	original	47.85	52.36	50.92	24.05
		REGG_TA_r	55.10	60.48	52.75	30.81
		REGG_TA	**64.03**	**62.36**	**65.60**	**38.60**
Citeseer	NO	original	51.37	46.84	59.14	65.68
	FDA	original	39.45	37.28	44.17	52.13
		REGG_TA_r	54.18	54.94	56.39	57.79
		REGG_TA	**55.98**	**56.93**	**59.63**	**69.02**
	RND	original	38.64	37.91	49.19	67.52
		REGG_TA_r	55.61	53.39	57.64	67.63
		REGG_TA	**56.51**	**58.03**	**61.72**	**85.21**
Pubmed	NO	original	42.70	42.86	46.30	23.70
	FDA	original	22.02	23.02	24.51	12.25
		REGG_TA_r	36.92	33.92	39.71	4.72
		REGG_TA	**42.83**	**51.83**	**45.02**	**14.94**
	RND	original	25.95	27.95	29.37	9.70
		REGG_TA_r	41.15	36.15	39.37	6.73
		REGG_TA	**43.94**	**52.94**	**44.14**	**13.49**

Table 7: Node classification performance ($NMI\_part\times100$) for the unattacked original aim sub-community, and the aim sub-community in three graphs attacked by FDA or RND: the original graph, the graph defensed by REGG_TA and the graph defensed by REGG_TA_r on Louvain, DeepWalk, SCD

Dataset	Attack	Graph	Louvain	DeepWalk	SCD
Email	NO	original	72.23	83.90	84.87
	FDA	original	24.56	39.72	75.43
		REGG_TA_r	18.19	45.97	80.62
		REGG_TA	**32.56**	**50.24**	**84.32**
	RND	original	11.85	42.05	77.67
		REGG_TA_r	17.15	41.15	76.08
		REGG_TA	**29.73**	**52.96**	**80.73**
Polblogs	NO	original	96.65	99.09	98.34
	FDA	original	31.13	60.89	85.33
		REGG_TA_r	47.27	71.67	89.47
		REGG_TA	**80.07**	**93.08**	**93.39**
	RND	original	20.27	64.42	57.14
		REGG_TA_r	10.02	67.14	67.41
		REGG_TA	**33.54**	**88.72**	**73.67**
Cora	NO	original	13.81	40.40	92.12
	FDA	original	7.386	19.85	83.56
		REGG_TA_r	7.01	31.15	85.34
		REGG_TA	**17.41**	**54.21**	**87.84**
	RND	original	10.04	23.99	85.54
		REGG_TA_r	8.19	36.68	86.80
		REGG_TA	**15.53**	**53.05**	**88.06**

REFERENCES

[1] Vincent D Blondel, Jeanloup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment* 2008, 10 (2008), 10008.

[2] Yijin Cai, Hong Zheng, Jiamou Liu, Bo Yan, Hongyi Su, and Yiping Liu. 2018. Balancing the pain and gain of hobnobbing: Utility-based network building over atributed social networks. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. 193–201.

[3] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2016. Deep neural networks for learning graph representations. (2016), 1145–1152.

[4] Jinyin Chen, Lihong Chen, Yixian Chen, Minghao Zhao, Shanqing Yu, Qi Xuan, and Xiaoniu Yang. 2019. GA-based Q-attack on community detection. *IEEE Transactions on Computational Social Systems* 6, 3 (2019), 491–503.

[5] Jinyin Chen, Yixian Chen, Lihong Chen, Minghao Zhao, and Qi Xuan. 2019. Multiscale Evolutionary Perturbation Attack on Community Detection. *arXiv preprint arXiv:1910.09741* (2019).

[6] Jinyin Chen, Yangyang Wu, Xiang Lin, and Qi Xuan. 2019. Can Adversarial Network Attack be Defended. *arXiv: Social and Information Networks* (2019).

[7] Liang Chen, Jintang Li, Jiaying Peng, Tao Xie, Zengxu Cao, Kun Xu, Xiangnan He, and Zibin Zheng. 2020. A Survey of Adversarial Learning on Graphs. *arXiv preprint arXiv:2003.05730* (2020).

[8] Yang Chen, Jiamou Liu, He Zhao, and Hongyi Su. 2020. Social structure emergence: A multi-agent reinforcement learning framework for relationship building. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. 1807–1809.

[9] Zhengdao Chen, Xiang Li, and Joan Bruna. 2017. Supervised community detection with line graph neural networks. *arXiv preprint arXiv:1705.08415* (2017).

[10] Aaron Clauset, M E J Newman, and Cristopher Moore. 2004. Finding community structure in very large networks. *Physical Review E* 70, 6 (2004), 066111.

[11] Negin Entezari, Saba A Alsayouri, Amirali Darvishzadeh, and Evangelos E Papalexakis. 2020. All You Need is Low (Rank): Defending Against Adversarial Attacks on Graphs. (2020), 169–177.

[12] Fuli Feng, Xiangnan He, Jie Tang, and Tatseng Chua. 2019. Graph Adversarial Training: Dynamically Regularizing Based on Graph Structure. *arXiv: Learning* (2019).

[13] Valeria Fionda and Giuseppe Pirro. 2018. Community Deception or: How to Stop Fearing Community Detection Algorithms. *IEEE Transactions on Knowledge and Data Engineering* 30, 4 (2018), 660–673.

[14] Santo Fortunato. 2010. Community detection in graphs. *Physics Reports* 486, 3 (2010), 75–174.

[15] Mikael Henaff, Joan Bruna, and Yann Lecun. 2015. Deep Convolutional Networks on Graph-Structured Data. *arXiv: Learning* (2015).

[16] Vassilis N Ioannidis and Georgios B Giannakis. 2019. Edge Dithering for Robust Adaptive Graph Convolutional Networks. *arXiv: Learning* (2019).

[17] Mohammad Rasool Izadi, Yihao Fang, Robert Stevenson, and Lizhen Lin. 2020. Optimization of Graph Neural Networks with Natural Gradient Descent. *CoRR* (2020).

[18] Wei Jin, Yaxin Li, Han Xu, Yiqi Wang, and Jiliang Tang. 2020. Adversarial Attacks and Defenses on Graphs: A Review and Empirical Study. *CoRR* (2020). https://arxiv.org/abs/2003.00653

[19] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020. Graph Structure Learning for Robust Graph Neural Networks. (2020), 66–74.

[20] Bakhadyr Khoussainov, Jiamou Liu, and Imran Khaliq. 2009. A dynamic algorithm for reachability games played on trees. In *International Symposium on Mathematical Foundations of Computer Science*. Springer, 477–488.

[21] Thomas Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv: Learning* (2016).

[22] Angsheng Li, Qifu Hu, Jun Liu, and Yicheng Pan. 2016. Resistance and Security Index of Networks: Structural Information Perspective of Network Security. *Scientific Reports* 6, 1 (2016), 26810–26810.

[23] Angsheng Li, Jiankou Li, and Yicheng Pan. [n.d.]. Discovering natural communities in networks. *Physica A Statistical Mechanics & Its Applications* 436 ([n. d.]), 878–896.

[24] Angsheng Li and Yicheng Pan. 2016. Structural Information and Dynamical Complexity of Networks. *IEEE Transactions on Information Theory* 62, 6 (2016), 3290–3339.

[25] Angsheng Li, Xianchen Yin, Bingxiang Xu, Danyang Wang, Jimin Han, Yi Wei, Yun Deng, Ying Xiong, and Zhihua Zhang. 2018. Decoding topologically associating domains with ultra-low resolution Hi-C data by graph structural entropy. *Nature communications* 9, 1 (2018), 3265.

[26] Jia Li, Honglei Zhang, Zhichao Han, Yu Rong, Hong Cheng, and Junzhou Huang. 2020. Adversarial Attack on Community Detection by Hiding Individuals. *arXiv: Social and Information Networks* (2020).

[27] Jiamou Liu and Ziheng Wei. 2014. Community detection based on graph dynamical systems with asynchronous runs. In *2014 Second International Symposium on Computing and Networking*. IEEE, 463–469.

[28] Jiamou Liu and Ziheng Wei. 2017. Network, popularity and social cohesion: a game-theoretic approach. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[29] Yiwei Liu, Jiamou Liu, Kaibin Wan, Zhan Qin, Zijian Zhang, Bakhadyr Khoussainov, and Liehuang Zhu. 2021. From Local to Global Norm Emergence: Dissolving Self-reinforcing Substructures with Incremental Social Instruments. In *International Conference on Machine Learning*. PMLR, 6871–6881.

[30] Yiwei Liu, Jiamou Liu, Zijian Zhang, Liehuang Zhu, and Angsheng Li. 2019. REM: From Structural Entropy to Community Structure Deception. (2019), 12918–12928.

[31] Symeon Papadopoulos, Yiannis Kompatsiaris, Athena Vakali, and Ploutarchos Spyridonos. 2012. Community detection in social media. *Data Mining and Knowledge Discovery* 24, 3 (2012), 515–554.

[32] Bryan Perozzi, Rami Alrfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. (2014), 701–710.

[33] Arnau Pratperez, David Dominguezsal, and Josepllluis Larribapey. 2014. High quality, scalable and parallel community detection for large real graphs. (2014), 225–236.

[34] Jorg Reichardt and Stefan Bornholdt. 2006. Statistical mechanics of community detection. *Physical Review E* 74, 1 (2006), 016110.

[35] Ke Sun, Hantao Guo, Zhanxing Zhu, and Zhouchen Lin. 2019. Virtual Adversarial Training on Graph Convolutional Networks in Node Classification. *arXiv: Learning* (2019).

[36] Acar Tamersoy, Kevin Roundy, and Duen Horng Chau. 2014. Guilt by association: large scale malware detection by mining file-relation graphs. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1524–1533.

[37] Twan Van Laarhoven and Elena Marchiori. 2012. Robust community detection methods with resolution parameter for complex detection in protein protein interaction networks. In *IAPR International Conference on Pattern Recognition in Bioinformatics*. Springer, 1–13.

[38] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[39] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph Attention Networks. *arXiv: Machine Learning* (2017).

[40] Hongwei Wang and Jure Leskovec. 2020. Unifying Graph Convolutional Neural Networks and Label Propagation. *CoRR* (2020).

[41] Shen Wang, Zhengzhang Chen, Jingchao Ni, Xiao Yu, Zhichun Li, Haifeng Chen, and Philip S Yu. 2019. Adversarial Defense Framework for Graph Neural Network. *arXiv: Learning* (2019).

[42] Marcin Waniek, Tomasz P Michalak, Talal Rahwan, and Michael Wooldridge. 2018. Hiding individuals and communities in a social network. *Nature Human Behaviour* 2, 2 (2018), 139–147.

[43] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. 2019. Adversarial Examples for Graph Data: Deep Insights into Attack and Defense. (2019), 4816–4823.

[44] Kaidi Xu, Hongge Chen, Sijia Liu, Pinyu Chen, Tsuiwei Weng, Mingyi Hong, and Xue Lin. 2019. Topology Attack and Defense for Graph Neural Networks: An Optimization Perspective. *arXiv: Learning* (2019).

[45] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2019. Robust Graph Convolutional Networks Against Adversarial Attacks. (2019), 1399–1407.

[46] Daniel Zugner, Amir Akbarnejad, and Stephan Gunnemann. 2018. Adversarial Attacks on Neural Networks for Graph Data. (2018), 2847–2856.

[47] Daniel Zugner and Stephan Gunnemann. 2019. Certifiable Robustness and Robust Training for Graph Convolutional Networks. (2019), 246–256.