Attacking Community Detectors: Mislead Detectors via Manipulating the Graph Structure

Kaibin Wan[1], Jiamou Liu[3], Yiwei Liu[1], Zijian Zhang[2,3(✉)], and Bakhadyr Khoussainov[4]

[1] School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China
{kaibinwan,yiweiliu}@bit.edu.cn
[2] School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, China
zhangzijian@bit.edu.cn
[3] School of Computer Science, The University of Auckland, Auckland, New Zealand
jiamou.liu@auckland.ac.nz
[4] School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China
bmk@uestc.edu.cn

Abstract. Community detection has been widely studied from many different perspectives, which include heuristic approaches in the past and graph neural network in recent years. With increasing security and privacy concerns, community detectors have been demonstrated to be vulnerable. A slight perturbation to the graph data can greatly change the detection results. In this paper, we focus on dealing with a kind of attack on one of the communities by manipulating the graph structure. We formulate this case as target community problem. The big challenge to solve this problem is the universality on different detectors. For this, we define structural information gain (SIG) to guide the manipulation and design an attack algorithm named SIGM. We compare SIGM with some recent attacks on five graph datasets. Results show that our attack is effective on misleading community detector.

Keywords: Adversarial community detection · Graph neural network · Structural entropy

1 Introduction

Real-world activities can usually be abstracted into networks, where entities are abstracted into nodes, and relationships or interactions between them are abstracted into edges. Entities naturally form communities because of their interaction. In general, a community is usually defined as a group of nodes with close internal interactions and sparse external interactions [11,21]. Analyzing communities in networks is one of the most active research directions due to its wide applications in the study of

This paper is supported by National Natural Science Foundation of China No. 62172040, No. U1836212, No. 61872041.

S. Deng et al. (Eds.): MobiCASE 2021, LNICST 434, pp. 112–128, 2022.
https://doi.org/10.1007/978-3-030-99203-3_8

social structures, biological interactions, collaboration networks, etc. [5,6,20,22,32]. The main task of identifying communities is community detection. Classical approaches mainly focus on optimizing a community-quality scores [1,8,26], random walk [25,28], and graph embedding [29,30]. The methods usually follow a unsupervised learning paradigm and only concern the topological structure of a graph. Modern approaches recast community detection as a node-wise classification task. The nodes with the same label can be consider as that they are in the same community. The main difference is that traditional community detection only concerns topology information, while the node-wise classification task will concern both node attributes and topology information. The representative work is graph neural network [7,13,33], where it could be unsupervised, semi-supervised and supervised.

Recent years, graph-based community detection algorithms have been widely applied to real world such as topic classification [27], malware detection [31], finding criminal organization [9], identifying auction fraud [24] and analyzing chromosomal domains [17], and they have achieved remarkable performance in those task. However, the community detection algorithms has been proved unreliable and vulnerable against perturbation [39]. For example, the node classification algorithm will be fooled if the attackers control several fake nodes and insert or delete some edges between fake nodes and other benign nodes. Zügner et al. [39] first study this attack by slightly changing the topology structure and node feature.

With the concern about security of graph-based classification, a surge of works have been done on adversarial model [10,18,23,35,36,39]. Based on the adversary's knowledge of target model, the attack can be characterized as three threatening levels, that is, white-box attack, gray-box attack and black-box attack [4]. Specifically, the attackers able to process all information of the target model in the white-box attack, process limited knowledge in the gray-box attack, and know nothing but do limited queries in the black-box attack. In terms of the adversary's goal, there exists targeted and untargeted attack. In targeted attack, the attacker aims to let the train model misclassify a small set of nodes, while, in the untargeted attack, the attacker aims to let the train model misclassify nodes as many as possible. In terms of the attack strategy, it contains topology attack and feature attack. For topology attack, the attackers are allowed to add or remove some edges in the graph to mislead the classifier, and for feature attack, the attackers can change the features of some specified nodes [3,38].

The goal of this paper is to perform community detection attacks from the perspective of information theory. Liu et al. have introduced residual entropy to solve this problem [23]. They develop an algorithm named REM to do the non-targeted attack. However, in most cases, the attackers do not have the ability to modify the edges of all the nodes and only a few nodes' misclassification are important to the attacker but not for all the nodes. On the other hand, REM's topology attack only consider adding edges. But, in fact, deleting edges is equally important. At last, attackers generally know nothing about the targeted classifier, that is, they do not have access to the classifier's parameter or training labels. Therefore, in this paper, we focus on doing black-box and targeted attack. The attackers have the ability to add or delete edges link nodes in target community. To solve this problem has three challenges: The first is the lack of universally-agreeable definition to model the structure of targeted community, which can view as an index to guide the targeted attack. The second is the black-box problem.

There are many models to do community detection including the classical and modern methods which make it hard to pinpoint a single objective metric to cheat all the models. The third is to achieve an efficient attack with a low complexity.

In this work, we defined structural information gain (SIG) as a new metric to guide the black-box and targeted attack. By minimizing SIG, we propose a SIGM-ATTACK algorithm to cheat the classifiers to do the correct classification by adding or removing edges. Our contribution are summarized as follow:

- We define structural information gain from a view of information theory which aims to reflect the information revealed from community structure.
- We propose an effective model to mislead the node classification of the target community by minimizing SIG. We further give an efficient algorithm SIGM-ATTACK with a low compute complexity $O(|V| + |C_t|)$.
- We experimentally validate the performance of our SIG-ATTACK on 5 community detection algorithm including the classical and modern ones.

2 Related Work

Community Detection. Communities occur in many network systems from biology, computer science, politics etc. In recent years, community detection has attracted many works due to a huge amount of data and computational resource [11]. Traditional community detection algorithms aim to classify nodes in a graph into subgraphs with tight internal connections and sparse external connections based on the topology of the graph. Some of them focus on optimizing a metric, like Modularity [1,8], Spinglass [26], and structure entropy [16]. Some of them are based on random walk, for example, the idea that random walks are more likely to stay in the same community [25] and the shortest description length for a random walk [28]. Some of them are graph embedding, like spectral clustering algorithm [34], sequence-based embedding [29] and probabilistic generative model [30]. Modern algorithms recast community detection as a node-wise classification task [37]. They not only pay attention to the topology of the complex network but also the features of each node. In general, every node has a label which can be considered as its community's name. The goal of these algorithms is to predict the labels of the remaining nodes. Motivated by convolutional neural network (CNN) [15] and recurrent neural network (RNN) [12], Graph neural network (GNN) appears as a new generalization to deal with graph data [37]. Bruna et al. firstly proposed a CNN-type architecture on graphs by formulated convolution-like operations in the spectral domain [2]. After that, many work emerged along this direction [7,13,33].

Attacks on Community Detection. Many studies of successful attacks indicate that community detection algorithms are vulnerable against perturbation [39]. Chen et al. provide taxonomies for various attacks [4]. Based on the attacker's knowledge of the target community detection model, they can be divided into white-box, gray-box and black-box attack. Based on attackers' goal, there are targeted attack and untargeted attacks. As for attackers' strategy, there are topology attack, feature attack and their hybrid. Based on different manipulation methods, the attacker can perform add, remove or rewiring operation. In addition, from the perspective of the attacker, there

are gradient-based and non-gradient-based algorithms. In our work, we focus on *black-box*, *targeted*, *non-gradient-based topology attack* with the add and remove manipulation. There are work on similar research with ours. Waniek et al. first came up with an algorithm named DICE to hide the target community [35]. DICE is inspired by modularity and worked by randomly deleting internal edge and adding external edge for the target community. Fionda et al. further extend the idea and propose the concept of community deception [10]. They devised two attack methods: one is based on maximizing safeness and the other is based on minimizing modularity. Later, based on GNN, Li et al. proposed an iterative attack model which learns to select the suitable candidate edges [18]. However, none of them do attack on the modern community detection algorithms. [35] and [10] only did attack on traditional detection algorithms and did not consider the graph data with features. [18] directly replaces all detection algorithms with a surrogate model and only did experiments on two data sets. In this paper, we will verify our attack algorithm on ten community detection algorithms and eight data sets. The community detection algorithms contain four optimized graph metric algorithms, two random-walk-based algorithms, two graph neural network algorithms, and heuristic algorithms. The data set also contains two forms, namely, with node features and without node features.

3 Problem Formulation

In this section, we will firstly give a general perspective of attackers' model and then give a formulation of our targeted attack.

3.1 Attacker's Model

The attacker's model includes the adversary's goal, knowledge, and abilities. We have roughly introduced it in the related work section, and below we will discuss these aspects in detail.

Attacker's Goal: In general, the attacker's goal can be divided in two categories:

- **Untargeted Attack:** In this attack, the attacker's purpose is to fool the detection algorithm so that the system has poor overall performance on the input graph data.
- **Targeted Attack:** In this attack, the attacker only focuses on a small set of nodes in the graph data and its purpose is to fool the detection algorithm to have the correct classification on those nodes.

In this paper, we consider the targeted attack on a set of nodes in the same community, since the set of nodes in different communities can be regarded as the union of sets of nodes in the same community.

Attacker's Knowledge: A complete community detection task includes datasets and the community detection algorithms on those datasets. Therefore, the background knowledge of an attacker can be summarized along two dimensions:

- **Graph Dataset:** This dimension characterized the attacker's knowledge about the dataset, including the global data, partial data, and labeled data. For graphs with features on nodes, the attacker's knowledge also includes whether they know the features of the nodes.

– **Algorithm or Classifier:** This dimension characterized the attacker's knowledge about the community detection algorithm, including algorithm settings and optimization methods. For node-wise classifier, it also includes whether there are supervision and classifier parameters, etc.

In this paper, we focus on black-box attack, that is, for the knowledge of the algorithm, the attacker knows nothing about the community detection algorithm besides a few limited queries. In addition, for the knowledge of graph dataset, the attacker knows the topological structure information and the node features of the target community.

Attacker's Capability: The capability of an attacker is reflected in the use of the attacker's knowledge. Therefore, the attacker's capabilities can be divided into three categories:

– **Modifying structure:** In this case, the attacker can make some perturbations on the topology of the graph data. Some common perturbations include adding edges, deleting edges, injecting nodes, and so on.
– **Modifying feature:** In this case, the attacker can slightly change some of the node features.
– **Modifying classifier:** In this case, the attacker has a certain ability to modify the community detection algorithm, for example, modifying the model parameters and gradient information.

In this paper, the attacker only has the ability to modify the graph structure, i.e., the attacker can insert fake edges or delete existing edges. Although the attacker knows features of some nodes, the attacker has no ability to modify these features. At the same time, we are conducting a black-box attack, where the attacker cannot perform any operations on the community detection algorithm.

3.2 A Form of Topology Attack

Generally, a social network is modelled as an undirected graph $G = (V, E)$ that includes a set of nodes V represents users and a set of edges E represents the relationship among those users. Traditionally, communities in G is characterized by high internal density and low external density. Non-overlapping communities structure in G refer to a partition $\mathcal{P} = \{C_1, C_2, \ldots, C_p\}$ of V where $p \in \mathbb{N}$ is the number of communities. Traditional community detection algorithm $\mathscr{F}$ is used to mining this partition based on the topology of G. However, in reality, each node usually has its own feature. Therefore, modern community detection algorithms take the features of nodes into consideration. For supervised tasks, nodes usually have their own labels, and the algorithm will classify nodes into communities with the same labels. For unsupervised tasks, as just mentioned, the algorithm divides the nodes into partition without labels. For convenience, in this situation, we take the real partition $\mathcal{P} = \{label_1, label_2, \ldots, label_p\}$ as their label set, i.e., we denote the label of $u \in C_j$ as $label_j$. After we get a new partition $\mathcal{P}' = \{C_1', C_2', \ldots, C_q'\}$ by unsupervised community detection algorithm, we match C_i' with C_j by solving a linear sum assignment problem to find a complete assignment of

$\mathcal{P}^{'}$ to $\mathcal{P}$ of $\max \sum_i \sum_j |C_i^{'} \cap C_j|$ [14]. If $C_i^{'}$ is matched with C_j, we denote the label of $u \in C_i^{'}$ as $label_j$.

For a graph $G = (V, E)$, the goal of the attacker is to rewire some edges such that the detection algorithm can not precisely identify members in the target community. We denote an *edge rewiring* for graph $G = (V, E)$ as

$$G \odot \{u, v\} := \begin{cases} G \oplus \{u, v\} = (V, E \cup \{u, v\}) & \{u, v\} \notin E, \\ G \ominus \{u, v\} = (V, E \setminus \{u, v\}) & \{u, v\} \in E. \end{cases}$$

Given an edge set $E^{'}$, for all $\{u, v\} \in E^{'}$, iteratively execute $G \odot \{u, v\}$, we get $G' = (V, E \odot E')$.

Definition 1 (Targeted community attack problem). *Given $G = (V, E)$ with feature X and a targeted community C_t. The goal of the attacker is to find a perturbed graph $G' = (V, E \odot E')$ so that any community detection algorithm $\mathscr{F}$ can obtain minimal information about C_t without changing X, i.e.,*

$$\min M(C_t | \mathscr{F}(G')), \ \textit{subject to,} \ |E'| \leq \Delta,$$

where M is the function to measure the information of C_t from $\mathscr{F}(G')$ and Δ is the perturbation budget.

4 Targeted Community Attack (TCA) Based on Entropy

In this section, we will study the structure of a graph based on information theory. Assume an undirected graph $G = (V, E, X)$ is composed of vertex set V, edge set E and the feature X, where a vertex is an entity, e.g., a user or computer, an edge is the relationship among two different vertexes, and the feature X is the attribute information of V. To solute TCA problem from modifying the structure, it makes sense to study the graph structure. Generally, the set V is known in the social network. Then the edges (the relationships among vertexes V) determine the structural information of a graph. How much information does an edge have? In fact, if we have already known the degrees of all the vertices, then for an edge uv, uv happens with probability $(d_u/2|E|)(d_v/2|E|)$ since u and v are independent before they form an edge. The information associated with the edge uv in G is $\log_2[(d_u/2|E|)(d_v/2|E|)]$. We define the average information of an edge in G as the edge entropy:

$$\mathcal{H}(G) := -\frac{1}{|E|} \sum_{uv \in E} \log_2[(d_u/2|E|)(d_v/2|E|)]. \tag{1}$$

The edge entropy $\mathcal{H}(G)$ of G captures the average number of bits needed to encode a relationship in G. Since choosing u and v are independent in edge uv, the average number of bits needed to encode an edge is twice the number of bits needed to encode a vertex:

$$\begin{aligned} \mathcal{H}(G) &= -\frac{1}{|E|} \sum_{uv \in E} \log_2[(d_u/2|E|)(d_v/2|E|)] \\ &= -2 \cdot \sum_{u \in V} \frac{d_u}{2|E|} \log_2 \frac{d_u}{2|E|}. \end{aligned} \tag{2}$$

We then define the edge entropy related to the target community C_t as the average number of bits to encode a vertex $u \in C_t$.

Definition 2. *The* edge entropy related to C_t in G *is*

$$\mathcal{H}(C_t|G) := -\sum_{u\in C_t} \frac{d_u}{|E|} \log_2 \frac{d_u}{2|E|}. \tag{3}$$

$\mathcal{H}(G)$ expresses the average amount of information of all edges, while $\mathcal{H}(C_t|G)$ is the part of these information that relates to community C_t. Neither $\mathcal{H}(G)$ nor $\mathcal{H}(C_t|G)$ considers the community structure. Let $\mathcal{P} = \{C_1, C_2, \ldots, C_p\}$ be the community partition of V. Then the members in G are divided into p communities $\{C_1, C_2, \ldots, C_p\}$. For the edge $uv \in E$, we have two independent steps to select an edge. The first step is identifying the community, and then the second step is selecting a member from this community. Then we have two cases: (1) $u, v \in C_j$ for some j, let ν_j be the sum degree of vertices in C_j, then we select the edge uv with probability $(d_u/\nu_j)(d_v/\nu_j)$ since we only select u and v from the vertex set C_j; (2) $uv \in C_i \times C_j$ and $i \neq j$, then we first identify C_i and C_j with probability $\nu_i/2|E|$ and $\nu_j/2|E|$, and select u and v with probability d_u/ν_i and d_v/ν_j from the corresponding part X_i and X_j. Therefore, the edge information of uv is $-\log_2[(d_u/\nu_j)(d_v/\nu_j)]$ for case (1) and $-\log_2[(\nu_i/2|E|)(\nu_j/2|E|)(d_u/\nu_j)(d_v/\nu_j)]$ for case (2), where the second information can be reduced to $-\log_2[(d_u/2|E|)(d_v/2|E|)]$. Therefore, if we know the community partition, the average information of an edge is

$$\mathcal{H}_{\mathcal{P}}(G) := \frac{1}{|E|}\{\mathcal{H}_1(G) + \mathcal{H}_2(G)\} \tag{4}$$

where

$$\mathcal{H}_1(G) = -\sum_{j=1}^{p} \sum_{uv\in E\&u,v\in X_j} \log_2[(d_u/\nu_j)(d_v/\nu_j)],$$

$$\mathcal{H}_2(G) = -\sum_{j=1}^{p} \sum_{uv\in E,u\in X_j\&v\notin X_j} \log_2[(d_u/2|E|)(d_v/2|E|)].$$

$\mathcal{H}_1(G)$ and $\mathcal{H}_2(G)$ correspond to case (1) and case (2), respectively. Simplify the equation above, we have

$$\mathcal{H}_{\mathcal{P}}(G) = \sum_{j=1}^{L} \sum_{u\in C_j} \left(-\frac{d_u^{in}}{|E|} \log_2 \frac{d_u}{\nu_j} - \frac{d_u^{out}}{|E|} \log_2 \frac{d_u}{2|E|} \right) \tag{5}$$

where d_u^{in} is the number of nodes $v \in C_j$ links u and d_u^{out} is the number of nodes $v \in V \setminus C_j$ links u.

Definition 3. *The* structural entropy related to C_t in G *is*

$$\mathcal{H}_{\mathcal{P}}(C_t|G) := \sum_{u\in C_t} \left(-\frac{d_u^{in}}{|E|} \log_2 \frac{d_u}{\nu_t} - \frac{d_u^{out}}{|E|} \log_2 \frac{d_u}{2|E|} \right) \tag{6}$$

and ν_t is the volume of the community C_t.

The distinction between $\mathcal{H}(C_t|G)$ and $\mathcal{H}_{\mathcal{P}}(C_t|G)$ is that whether the vertices in C_t know their community members. We define their difference as structural information gain revealed from community C_t:

Definition 4. *The* structural information gain related to the community C_t in G *is*

$$\mathcal{C}_{\mathcal{P}}(C_t|G) := \mathcal{H}(C_t|G) - \mathcal{H}_{\mathcal{P}}(C_t|G). \tag{7}$$

Let ν_t is the volume of C_t and g_t is the number of edges linked outside from C_t, then

$$\begin{aligned}\mathcal{C}_{\mathcal{P}}(C_t|G) &= -\sum_{u\in C_t} -\frac{d_u^{in}}{|E|}\log_2\frac{\nu_t}{2|E|} \\ &= -\frac{\nu_t - g_t}{|E|}\log_2\frac{\nu_t}{2|E|} \qquad (8)\\ &= -\sum_{uv\in E\&u,v\in C_t}\log_2(\frac{\nu_t}{2|E|})^2 \qquad (9)\end{aligned}$$

The probability that an edge's endpoints happen in C_t is $(\nu_t/2|E|)^2$. Then from Eq. (9), $\mathcal{C}_{\mathcal{P}}(C_t|G)$ is the average information that the endpoints of an edge uv are both in C_t. It reflects the information revealed by community C_t. A high value of $\mathcal{C}_{\mathcal{P}}(C_t|G)$ expresses the closed connection within C_t, and a low value means the sparse connections within C_t. In particular, if there is no edge within community C_t, $\mathcal{C}_{\mathcal{P}}(C_t|G) = 0$. Therefore, if we say $\mathcal{H}(C_t|G)$ is the whole information of C_t in G, $\mathcal{C}_{\mathcal{P}}(C_t|G)$ would be the community information that community C_t takes shape because of the known members in C_t.

Lemma 1. *The maximum decrease of $\mathcal{C}_{\mathcal{P}}(C_t|G \oplus \{u,v\})$ happens only when the are exactly one endpoint of $\{u,v\}$ belongs to C_t. Similarly, the maximum decrease of $\mathcal{C}_{\mathcal{P}}(C_t|G \ominus \{u,v\})$ happens only when both u and v belong to C_t.*

Proof. Denote $\mathcal{C}_{\mathcal{P}}(C_t|G \odot \{u,v\}) = g(u,v)$ for the rewire edge $\{u,v\}$. Let $k = 1$ if $\odot = \oplus$, $k = -1$ if $\odot = \ominus$, and $|E| = m$. Then if $u_1, v_1, u_2 \in C_t$, $v_2, u_3, v_3 \notin C_t$, from (8), we have

$$\begin{cases} g(u_1,v_1) = -\frac{\nu_t - g_t + 2k}{2m+2k}\log_2\frac{\nu_t+2k}{2m+2k}, \\ g(u_2,v_2) = -\frac{\nu_t - g_t}{2m+2k}\log_2\frac{\nu_t+k}{2m+2k}, \\ g(u_3,v_3) = -\frac{\nu_t - g_t}{2m+2k}\log_2\frac{\nu_t}{2m+2k}. \end{cases}$$

Let $\gamma = -1/(2m+2k)$, then

$$\begin{cases} g(u_2,v_2) - g(u_3,v_3) = \gamma(\nu_t - g_t)\log_2\frac{\nu_t+k}{\nu_t} \\ g(u_3,v_3) - g(u_1,v_1) = \gamma\log_2(\frac{\nu_t}{\nu_t+2k})^{\nu_t-g_t}(\frac{2m+2k}{\nu_t+2k})^{2k} \end{cases}$$

Since $(1+1/x)^x$ is monotonically increasing, $\lim_{x\to+\infty}(1+1/x)^x = e$ and $\frac{2m+2k}{\nu_t+2k} \geq \sqrt{e}$ where e is natural logarithm, we have $g(u_2,v_2) \leq g(u_3,v_3) \leq g(u_1,v_1)$ if $k = 1$ and $g(u_1,v_1) \leq g(u_3,v_3) \leq g(u_2,v_2)$ if $k = -1$.

Remark 1. Lemma 1 shows the general principle for hiding a community is adding external and deleting internal (AEDI).

Since the value of $\mathcal{C}_{\mathcal{P}}(C_t|G)$ is related to the size of G, then a natural definition of the normalized structural information gain is:

$$\rho_t(G) := \mathcal{C}_{\mathcal{P}}(C_t|G)/\mathcal{H}(C_t|G). \tag{10}$$

So, if we want to attack a community C_t, we would like to minimize $\rho_t(G)$. Denote a structural information gain minimization (SIGM) attack as an algorithm that outputs a rewiring edge e such that $\rho_t(G \odot e)$ is minimized. A crude implementation of a SIGM attack according for the principle AEDI to examine each potential edge e that connects external for $\rho_t(G \oplus e)$ and disconnect internal $\rho_t(G \ominus e)$. This implementation runs in $O(|C_t||V|)$ time since it takes $|C_t|(|V| - |C_t|)$ for adding and $|C_t|^2$ for deleting, but it is still inapplicable for large graphs. We instead present an $O(\log|V| + |C_t|)$-implementation in Algorithm 1 according to the following Lemmas.

Lemma 2. *For any edge $\{u, v\} \notin E$, $u \in C_t$ and $v \notin C_t$ where C_t is the target community. The maximum decrease of $\rho_t(G \oplus \{u, v\})$ happens only when d_u is minimum.*

Proof. For any $u_1, u_2 \in C_t$, $v \notin C_t$, we have $\mathcal{C}_{\mathcal{P}}(C_t|G \oplus \{u_1, v\}) = \mathcal{C}_{\mathcal{P}}(C_t|G \oplus \{u_2, v\})$ from (8). Then, assume $d_{u_1} < d_{u_2}$, from (10), we only need to prove $\mathcal{H}(C_t|G \oplus \{u_1, v\}) > \mathcal{H}(C_t|G \oplus \{u_2, v\})$. Define the function $F\colon \mathbb{R} \to \mathbb{R}$ by $F(x) = (x+1)\log_2(x+1) - x\log_2(x)$. The function F is *monotonically increasing*, as $F'(x) = \log_2(x+1) - \log_2(x) > 0$. Then, we finish the proof since $\mathcal{H}(C_t|G \oplus \{u_1, v\}) - \mathcal{H}(C_t|G \oplus \{u_2, v\}) = \frac{1}{2(|E|+1)}(F(d_{u_2}) - F(d_{u_1})) > 0$.

For $\{x, y\} \in E$, we call $\{x, y\} \in C_t \times C_t$ is critical if d_y is maximum among all x's neighbors in C_t.

Lemma 3. *There exists a critical edge $\{u, v\}$ such that $\rho_t(G \ominus \{u, v\})$ is minimum.*

Proof. Assume $\{u, v\}$ is the best edge such that $\rho_X(G \ominus \{u, v\})$ is minimum among all edges in $C_t \times C_t$. If $\{u, v\}$ is not a critical edge, then either u or v is not the other one's neighbor with the maximum degree. Without loss of generality, we assume v is not u's neighbor with the maximum degree and w is. Define $F(x)$ as described in Lemma 2. Since the function F is *monotonically increasing* for $x > 0$, then $\mathcal{H}(C_t|G \ominus \{u, w\}) - \mathcal{H}(C_t|G \ominus \{u, v\}) = \frac{1}{2(|E|-1)}(F(d_w - 1) - F(d_v - 1)) > 0$. We finish the proof since $\mathcal{C}_{\mathcal{P}}(C_t|G \ominus \{u, v\}) = \mathcal{C}^{\mathcal{P}}(C_t|G \ominus \{u, w\})$ and (10).

Theorem 1. *Algorithm 1 implements* SIGM *in* $O(\log|V| + |C_t|)$.

Proof. For adding edge, the Algorithm 1 will take $O(1)$ operations to find minimum degree and $O(\log|V|)$ operations to update the degree sequence $SeqV$. For deleting edge, Algorithm 1 will take $O(|C_t|)$ operations to go over all critical edges and at most $O(|C_t|)$ operations to update the critical edges $CriC$.

Algorithm 1. An efficient SIGM attack

Input: Graph $G = (V, E)$, $\mathcal{P} = \{C_1, C_2, \ldots, C_p\}$, $1 \leq t \leq p$
Output: A non-edge $\{u^*, v^*\}$

Precalculate the degree sequence $SeqV$ from small to large for V and the critical edge set $CriC$ for C_t
Choose $u \in C_t$ from $SeqV$ s.t. d_u is minimum
Randomly choose $v \notin C_t$ from $SeqV$ limited in $V \backslash C_t$ s.t. $\{u, v\} \notin E$ and d_v is minimum
Go over all critical edges in $CriC$ to select an edge $\{x, y\}$ s.t. $\rho_t(G \ominus \{x, y\})$ is minimum
if $\rho_t(G \oplus \{u, v\}) \leq \rho_t(G \ominus \{x, y\})$ **then**
    Set $u^* \leftarrow u$, $v^* \leftarrow v$
else
    Set $u^* \leftarrow x$, $v^* \leftarrow y$
end if
Update the sequence $SeqV$ and the edge set $CriC$
return $\{u^*, v^*\}$

5 Experiments

In this section, we will evaluate the performance of our algorithm SIGM on five real-world datasets where three datasets contain the node features and the others do not contain node features. In our experimental settings, we choose both traditional and modern community detection algorithms as the targets of the attack, and three attack algorithms as the benchmark. Finally, we use experimental results to verify that our algorithm can achieve good performance on both traditional and modern community detection algorithms.

Table 1. Specifics of the datasets, where C_t is the target community and $E(C_t)$ is the edge set in C_t.

Dataset	V	E	Targeted community	
			$(\|C_1\|, \|E_1\|)$	$(\|C_2\|, \|E_2\|)$
Email	1133	5451	(214, 784)	(303, 1053)
DBLP	317080	1049866	(746, 1990)	(463, 1262)
Amazon	334863	925872	(351, 810)	(399, 1139)
Cora	2708	5429	(818, 1175)	(180, 253)
Citeseer	3312	4732	(668, 1041)	(701, 1016)

5.1 Data

We will evaluate the performance of our algorithm over two types of datasets:

– one type is the real social network without node features, including the email communication network between members of the Univeristy Rovira i Virgili (**Email**[1]),

[1] https://deim.urv.cat/~alexandre.arenas/data/welcome.htm.

the co-authorship network in DBLP (**DBLP**[2]) and the co-purchased network in Amazon (**Ama**[2]).
– the other type is the real social network with node features, including three citation networks of 2708 and 3312 scientific publications respectively (**Cora**[3] and Citeseer[3]).

Table 1 gives an overview of the datasets. In each dataset, we choose two communities as our target community. In our experiments, we will rewire a certain ratio edges of $E(C_t)$, where $E(C_t)$ is the edge set within the target community C_t. It should be noted that because the data of **DBLP** and **Amazon** is too large, here we only extracted 3095 nodes of data DBLP and 10328 nodes of data Amazon.

5.2 Benchmark, Community Detectors and Metrics

Benchmark. Given a social network $G = (V, E)$ and the target community C_t, we use the following attack algorithms as our benchmark:

– Random Attack **(RND)**, which follows the algorithms RND in [39]. At each rewiring step, randomly sample a node v in V. If v is not directly connected to any node in C_t, we randomly choose a node u in in C_t and add an edge between u and v; otherwise we delete one of the directly connected edge between v and C_t.
– **DICE**, which follows the heuristic attack idea that **D**isconnect **I**nternally and **C**onnect **E**xternally in [35]. It works via the following steps given a budget Δ: (1) randomly delete $\Delta_d \leq \Delta$ edges from within the community C_t; and (2) randomly insert $\Delta - \Delta_d$ edges between C_t and the rest nodes $V \backslash C_t$.
– Modularity Based Attack **(MBA)** [10], which weaken the target community by minimizing modularity. At each rewiring step, the algorithm will add or delete an edge that links to C_t and has the minimal modularity.

Community Detectors. We consider five well-known community detection algorithms as our object to verify our attack performance. The detectors contain both traditional and modern community detection algorithms.

(1) Louvain [1]: a heuristic multi-level modularity maximizing algorithm.
(2) InfoMap [28]: an optimization algorithm by providing the shortest description length for a random walk.
(3) EdMot [19]: an Edge enhancement approach for Motif-aware community detection.
(4) GCN [13]: a scalable approach that is based on an efficient variant of convolutional neural networks.
(5) GAT [33]: a novel neural network architectures following a self-attention strategy to compute the hidden representations of each node by attending over its neighbors in the graph.

[2] http://snap.stanford.edu/data/.
[3] https://paperswithcode.com/datasets?mod=graphs.

Metrics. We introduce two measures to quantify the degree of attack. Given a network $G = (V, E)$ and the target community C_t, the goal is let C_t escape detection from community detector $\mathscr{F}$, which means that we cannot refer C_t from $\mathscr{F}(G') = \mathcal{P}^{'}$, where G' is the perturbed graph after attack. Therefore, how to measure the extent that we can refer C_t from $\mathcal{P}^{'}$ is the key to evaluate the effect of an attack. Here we introduce information entropy to give an explainable definition. Let $\mathcal{P}^{'} = \{C_1^{'}, C_2^{'}, \ldots, C_q^{'}\}$.

$$H(C_t) = -\frac{|C_t|}{|V|}\log\frac{|C_t|}{|V|},$$
$$H(C_t|\mathcal{P}^{'}) = -\sum_{j=1}^{q}\frac{|C_t \cap C_j^{'}|}{|V|}\log\frac{|C_t \cap C_j^{'}|/|V|}{|C_j^{'}|/|V|}.$$

$H(C_t)$ is the amount of uncertain information to encode the target community C_t in G. $H(C_t|\mathcal{P}^{'})$ is the residual uncertain information if we have the partition $\mathcal{P}^{'}$. Thus, normalized mutual information for C_t is defined as $I(C_t, \mathcal{P}^{'}) = (H(C_t) - H(C_t|\mathcal{P}^{'}))/H(C_t)$. Of course, a low $I(X, \mathcal{P})$ means a good attack.

On the other hand, we consider a measure widely used in machine learning, F1-score. Let C_t is the target community, the F1 score for C_t is defined as:

$$F_1 = \frac{2 \times \mathcal{R}(C_t, C_s') \times \mathcal{P}(C_t, C_s')}{\mathcal{R}(C_t, C_s') + \mathcal{P}(C_t, C_s')}$$

where

$$\mathcal{R}(C_t, C_s') = \frac{|C_t \cap C_s'|}{|C_t|},$$
$$\mathcal{P}(C_t, C_s') = \frac{|C_t \cap C_s'|}{|C_s'|},$$

where C_s' has the same label with C_t. Similarly, the smaller the value of F1, the better the attack effect.

5.3 Experimental Settings and Results

Experimental Settings. We compare SIGM with three other attack algorithms, including RND, DICE and MBA. We conduct experiments on the four algorithms. The setup and process of the experiment are described as follows: (1) Randomly choose a community C_t as the target community from the ground truth network $G = (V, E)$; (2) Calculate a certain ratio edges of $|E(C_t)|$ by applying an attack on the initial network G, and output the edge set E'; (3) Calculate the new community partition $\mathcal{P}'$ by applying community detector $\mathscr{F}$ on the perturbed network $G \odot E'$, i.e., $\mathcal{P}' = \mathscr{F}(G \odot E')$; (4) Calculate the metrics $\{I, F1\}$ for the target community C_t based on the new community partition $\mathcal{P}'$.

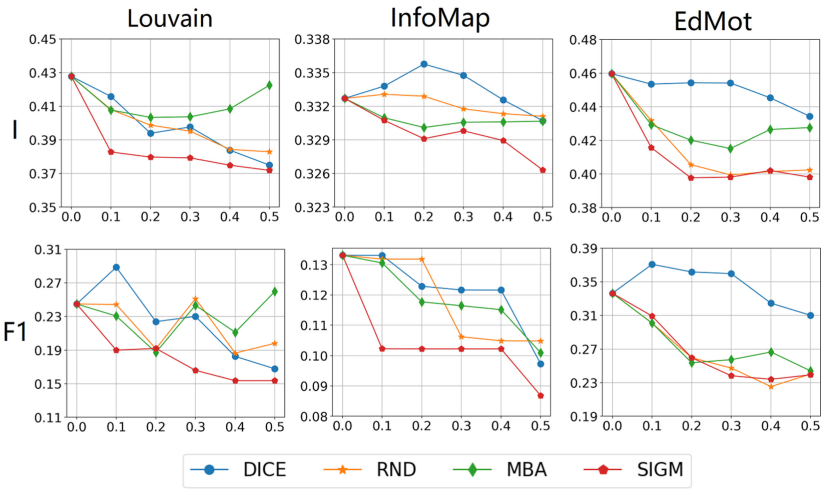

Fig. 1. The effect after applying attack RND, MBA, DICE and SIGM for a target community of dataset DBLP.

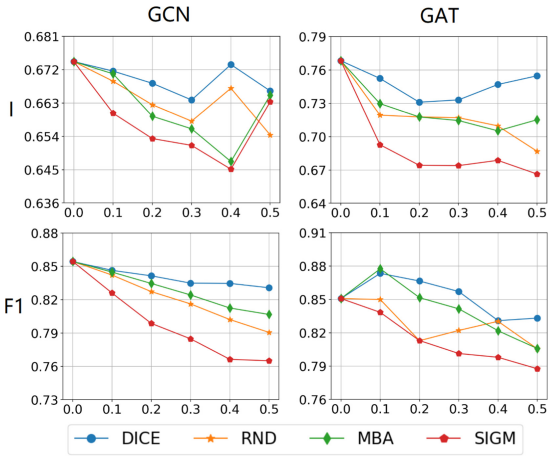

Fig. 2. The effect after applying attack RND, MBA, DICE and SIGM for a community of dataset Cora.

Experimental Results. We examine the performance of the four attacks over five datasets by modifying a certain ratio edges of $E(C_t)$. In Table 2, the ratio is 30%. Table 2 compares the attack effect score $I, F1$ with the other three attacks. Clearly, SIGM performs better than RND, DICE and MBA in most scenarios. In Fig. 1 and Fig. 2, we show the detailed trend of $I, \mathscr{F}1$ scores with the ratio of budget edges for the data DBLP and Cora. The indicators $\{I, F1\}$, which used to evaluate the effectiveness of detection algorithm, decrease significantly as the cost of perturbing the graph increasing. Obviously, all attacks are effective, but the performance is still different.

Table 2. The value of I and $F1$ for different attacks on 5 kinds of community detectors after modifying 30% edges of $E(C_t)$.

Data	Detector	NMI (I)				F1-score (F1)			
		RND	DICE	MBA	SIGM	RND	DICE	MBA	SIGM
Email	Louvain	0.527	0.542	0.516	**0.501**	0.563	0.522	0.517	**0.449**
		0.445	0.447	0.465	**0.435**	0.578	0.549	0.574	**0.440**
	InfoMap	0.351	0.353	0.352	**0.338**	0.273	0.268	0.267	**0.233**
		0.276	0.277	0.277	**0.274**	0.170	0.187	**0.169**	0.186
	EdMot	0.476	0.495	0.488	**0.457**	0.447	0.472	0.447	**0.416**
		0.391	0.392	0.404	**0.386**	0.305	0.309	0.369	**0.297**
DBLP	Louvain	0.326	0.337	0.341	**0.318**	**0.278**	0.290	0.305	0.306
		0.383	0.390	0.413	**0.370**	0.193	0.209	0.208	**0.183**
	InfoMap	0.258	0.258	0.257	**0.254**	0.112	0.111	**0.105**	0.118
		0.333	0.330	0.330	**0.326**	0.118	0.109	0.120	**0.087**
	EdMot	0.339	0.369	0.360	**0.332**	0.286	0.345	0.324	**0.257**
		0.403	0.440	0.427	**0.403**	**0.250**	0.330	0.311	0.253
Amazon	Louvain	0.617	0.715	0.630	**0.559**	0.330	0.432	0.351	**0.313**
		0.695	0.776	0.721	**0.601**	0.674	0.735	0.556	**0.463**
	InfoMap	0.539	0.548	0.540	**0.529**	0.296	0.324	0.303	**0.27**
		0.607	0.664	0.623	**0.568**	0.676	0.763	0.686	**0.617**
	EdMot	0.710	0.910	0.736	**0.709**	0.727	0.951	**0.681**	0.739
		0.775	0.910	0.824	**0.710**	0.873	0.962	0.773	**0.766**
Cora	GCN	0.666	0.662	**0.651**	0.656	0.813	0.837	0.823	**0.783**
		0.698	0.721	0.710	**0.680**	0.723	0.771	0.694	**0.692**
	GAT	0.701	0.730	0.708	**0.693**	0.835	0.856	0.838	**0.792**
		0.696	0.771	0.732	**0.687**	**0.588**	0.804	0.664	0.599
Citeseer	GCN	0.682	0.688	0.679	**0.667**	0.727	0.754	0.728	**0.711**
		0.656	0.654	0.655	**0.641**	0.773	0.792	0.773	**0.760**
	GAT	0.667	0.669	**0.649**	0.657	0.716	0.749	0.716	**0.685**
		0.646	0.650	0.657	**0.646**	0.751	0.789	0.743	**0.736**

SIGM performs better than RND, DICE and MBA in most of sampling point and much more stable than other three benchmarks. Thus the result validates SIGM's effectiveness in attacking the target community.

Although all the four attacks are effective, there are still fluctuations in the downward trend of indicators $I, F1$. This can be explained in two perspectives. On the one hand, the fluctuation may be caused by the attack algorithm. For example, the DICE is a heuristic algorithm, which controls the proportion of adding and deleting operations equals 0.5 on average. But this ratio is affected by the community structure, so some of the perturbations taken by DICE are invalid or even counterproductive. On the

other hand, the robustness of the community detectors will also cause such fluctuations. In general, the graph embedding algorithms are more robust than traditional community detectors. That's why the attacks over EdMot, GCN and GAT are more stable than Louvain and InfoMap in Fig. 1.

6 Conclusion

In this paper, we introduce the target community attack problem, utilize community based structural information to the this problem, and propose a structural information gain minimization (SIGM) algorithm. We optimize SIGM to make it more efficient. Experiments in the real world network show that our algorithm SIGM performs better than RND, DICE and MBA in most of attack scenarios. Some future works include (1) attacking some important nodes, e.g., influential nodes, hierarchies, etc. (2) attacking the communities in weighted and directed graphs; (3) exploring some new metric as evaluation score (4) Combining the information with differential privacy on graphs.

References

1. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Stat. Mech. **2008**(10), P10008 (2008)
2. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203 (2013)
3. Cai, Y., Zheng, H., Liu, J., Yan, B., Su, H., Liu, Y.: 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, pp. 193–201 (2018)
4. Chen, L., et al.: A survey of adversarial learning on graphs. arXiv preprint arXiv:2003.05730 (2020)
5. Chen, Q., Su, H., Liu, J., Yan, B., Zheng, H., Zhao, H.: In pursuit of social capital: upgrading social circle through edge rewiring. In: Shao, J., Yiu, M.L., Toyoda, M., Zhang, D., Wang, W., Cui, B. (eds.) APWeb-WAIM 2019. LNCS, vol. 11641, pp. 207–222. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26072-9_15
6. Chen, Y., Liu, J.: Distributed community detection over blockchain networks based on structural entropy. In: Proceedings of the 2019 ACM International Symposium on Blockchain and Secure Critical Infrastructure, pp. 3–12 (2019)
7. Chen, Z., Li, X., Bruna, J.: Supervised community detection with line graph neural networks. arXiv preprint arXiv:1705.08415 (2017)
8. Clauset, A., Newman, M.E., Moore, C.: Finding community structure in very large networks. Phys. Rev. E **70**(6), 066111 (2004)
9. Ferrara, E., De Meo, P., Catanese, S., Fiumara, G.: Detecting criminal organizations in mobile phone networks. Expert Syst. Appl. **41**(13), 5733–5750 (2014)
10. Fionda, V., Pirro, G.: Community deception or: how to stop fearing community detection algorithms. IEEE Trans. Knowl. Data Eng. **30**(4), 660–673 (2017)
11. Fortunato, S.: Community detection in graphs. Phys. Rep. **486**(3–5), 75–174 (2010)
12. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)

13. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
14. Kuhn, H.W.: The hungarian method for the assignment problem. Naval Res. Logist. Q. **2**(1–2), 83–97 (1955)
15. LeCun, Y., Bengio, Y., et al.: Convolutional networks for images, speech, and time series. Handb. Brain Theory Neural Netw. **3361**(10), 1995 (1995)
16. Li, A., Li, J., Pan, Y.: Discovering natural communities in networks. Phys. A **436**, 878–896 (2015)
17. Li, A., et al.: Decoding topologically associating domains with ultra-low resolution hi-c data by graph structural entropy. Nat. Commun. **9**(1), 3265 (2018)
18. Li, J., Zhang, H., Han, Z., Rong, Y., Cheng, H., Huang, J.: Adversarial attack on community detection by hiding individuals. In: Proceedings of The Web Conference 2020, pp. 917–927 (2020)
19. Li, P.Z., Huang, L., Wang, C.D., Lai, J.H.: Edmot: an edge enhancement approach for motif-aware community detection. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 479–487 (2019)
20. Liu, J., Minnes, M.: Deciding the isomorphism problem in classes of unary automatic structures. Theoret. Comput. Sci. **412**(18), 1705–1717 (2011)
21. Liu, J., Wei, Z.: Community detection based on graph dynamical systems with asynchronous runs. In: 2014 Second International Symposium on Computing and Networking, pp. 463–469. IEEE (2014)
22. Liu, Y., et al.: From local to global norm emergence: Dissolving self-reinforcing substructures with incremental social instruments. In: International Conference on Machine Learning, pp. 6871–6881. PMLR (2021)
23. Liu, Y., Liu, J., Zhang, Z., Zhu, L., Li, A.: Rem: from structural entropy to community structure deception. In: Advances in Neural Information Processing Systems, pp. 12938–12948 (2019)
24. Pandit, S., Chau, D.H., Wang, S., Faloutsos, C.: Netprobe: a fast and scalable system for fraud detection in online auction networks. In: Proceedings of the 16th International Conference on World Wide Web, pp. 201–210 (2007)
25. Pons, P., Latapy, M.: Computing communities in large networks using random walks. In: Yolum, I., Güngör, T., Gürgen, F., Özturan, C. (eds.) ISCIS 2005. LNCS, vol. 3733, pp. 284–293. Springer, Heidelberg (2005). https://doi.org/10.1007/11569596_31
26. Reichardt, J., Bornholdt, S.: Statistical mechanics of community detection. Phys. Rev. E **74**(1), 016110 (2006)
27. Revelle, M., Domeniconi, C., Sweeney, M., Johri, A.: Finding community topics and membership in graphs. In: Appice, A., Rodrigues, P.P., Santos Costa, V., Gama, J., Jorge, A., Soares, C. (eds.) ECML PKDD 2015. LNCS (LNAI), vol. 9285, pp. 625–640. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23525-7_38
28. Rosvall, M., Bergstrom, C.T.: Maps of random walks on complex networks reveal community structure. Proc. Natl. Acad. Sci. **105**(4), 1118–1123 (2008)
29. Rozemberczki, B., Davies, R., Sarkar, R., Sutton, C.: Gemsec: graph embedding with self clustering. In: Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pp. 65–72 (2019)
30. Sun, F.Y., Qu, M., Hoffmann, J., Huang, C.W., Tang, J.: vgraph: a generative model for joint community detection and node representation learning. In: Advances in Neural Information Processing Systems, pp. 514–524 (2019)
31. Tamersoy, A., Roundy, K., Chau, D.H.: 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, pp. 1524–1533 (2014)

32. van Laarhoven, T., Marchiori, E.: Robust community detection methods with resolution parameter for complex detection in protein protein interaction networks. In: Shibuya, T., Kashima, H., Sese, J., Ahmad, S. (eds.) PRIB 2012. LNCS, vol. 7632, pp. 1–13. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34123-6_1
33. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
34. Von Luxburg, U.: A tutorial on spectral clustering. Stat. Comput. **17**(4), 395–416 (2007)
35. Waniek, M., Michalak, T.P., Wooldridge, M.J., Rahwan, T.: Hiding individuals and communities in a social network. Nat. Hum. Behav. **2**(2), 139–147 (2018)
36. Wu, H., Wang, C., Tyshetskiy, Y., Docherty, A., Lu, K., Zhu, L.: Adversarial examples on graph data: Deep insights into attack and defense. arXiv preprint arXiv:1903.01610 (2019)
37. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S.Y.: A comprehensive survey on graph neural networks. IEEE Trans. Neural Netw. Learn. Syst. **32**(1), 4–24 (2021)
38. Yan, B., Liu, Y., Liu, J., Cai, Y., Su, H., Zheng, H.: From the periphery to the center: information brokerage in an evolving network. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence, pp. 3912–3918 (2018)
39. Zügner, D., Akbarnejad, A., Günnemann, S.: Adversarial attacks on neural networks for graph data. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2847–2856 (2018)