Multiscale Evolutionary Perturbation Attack on Community Detection

Jinyin Chen, Yixian Chen, Lihong Chen, Minghao Zhao, and Qi Xuan

Abstract—Community detection, aiming to group nodes based on their connections, plays an important role in network analysis, since communities, treated as meta-nodes, allow us to create a large-scale map of a network to simplify its analysis. However, for privacy reasons, we may want to prevent communities from being discovered in certain cases, leading to the topics on community deception. In this paper, we formalize this community detection attack problem in three scales, including global attack (macroscale), target community attack (mesoscale) and target node attack (microscale). We treat this as an optimization problem and further propose a novel Evolutionary Perturbation Attack (EPA) method, where we generate adversarial networks to realize the community detection attack. Numerical experiments validate that our EPA can successfully attack network community algorithms in all three scales, i.e., hide target nodes or communities and further disturb the community structure of the whole network by only changing a small fraction of links. By comparison, our EPA behaves better than a number of baseline attack methods on six synthetic networks and three real-world networks. More interestingly, although our EPA is based on the louvain algorithm, it is also effective on attacking other community detection algorithms, validating its good transferability.

Index Terms—Social network, community detection, community deception, privacy protection, genetic algorithm.

◆

1 INTRODUCTION

NATURAL network consists of many nodes and links, which captures certain relationship in real world and is often used as a mathematical representation for a variety of complex systems, such as social systems [1]–[4], transportation systems [5], [6] and supply chain systems [7], [8], etc. Many real-world networks can be divided into communities, with the nodes within the same communities connected densely, while those across different communities connected sparsely [9]. The revealed community structure can not only present the close relationship among the nodes inside a certain community, but may also indicate that these nodes tend to share common properties or play similar roles in the respective fields [10]. Analyzing community structure in a network thus can help to better understand the interactions inter- and intra- close groups of nodes in the network, which is the exact reason why new community detection algorithms are continuously proposed and widely used in a large number of areas.

Community detection algorithms are designed to divide the network into partitions of dense regions which correspond to strong related entities. Evaluating the effectiveness of these algorithms has been a controversial problem until Newman proposed the modularity Q [11], [12]. The proposal of modularity makes the problem of non-overlapping community detection unprecedented developed. A bunch of modularity optimization algorithms were subsequently proposed, some of which are based on splitting or aggregation [13]–[16], while many others use

optimization algorithms to maximizes the modular Q, such as annealing [17], Particle Swarm Optimization (PSO) [18], external optimization [19] and spectral optimization [20]. These algorithms translate community detection into an optimization problem and try to find the optimal community division by maximizing certain fitness. In addition, there are also community detection algorithms based on information theory [21]–[24] and label propagation [25]–[27]. The former believes that data flow can be compressed with regular code, such as random walk model. The main idea is that the probability of wandering to the nodes in the same group is much greater than those in different groups; while the latter updates each node label to its most frequent neighbor label through iteration, which was widely used due to its simplicity and high efficiency.

However, on one hand, people may not want their social information, such as communities, as a part of privacy to be discovered by certain algorithms; on the other hand, a lot of graph-based models, e.g., graph-based recommenders, need to integrate community detection to improve their efficiency, especially for those relatively large systems. The normal operation of such systems thus may rely on the robustness of community detection algorithm. These bring up a problem: how to attack community detection algorithms or defense against such attacks? Since different links play different roles in keeping the community structure of a network, the community detection algorithms could be significantly disturbed by only changing a small fraction of links. In this paper, we focus on the attack part, and seek the maximum community change by rewiring minimal number of links, which can help to judge which links are most vulnerable and thus also provides insights for the defense part in order to keep the community structure.

There are some attack strategies to disturb network algorithms [31]–[34], but few studies related to community

- J. Chen, Y. Chen, L. Chen, and Q. Xuan are with the Institute of Cyberspace Security, College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China. E-mail: {chenjinyin, 201407760108, 2111803032, xuanqi}@zjut.edu.cn.
  M. Zhao is with the Fuxi AI Lab, NetEase Inc, Hangzhou 310052, China. E-mail: zhaominghao@corp.netease.com.
  (Corresponding author: Qi Xuan.)

TABLE 1: Comparison with the existent approaches.

Approach	Scale of Attack	Way to Change Network	Budget Required
Nagaraja [28]	Mesoscale	Addition	Yes
Waniek et al. [29]	Mesoscale	Rewiring	Yes
Fionda et al. [30]	Mesoscale	Rewiring	Yes
Our EPA	Macroscale & Mesoscale & Microscale	Rewiring & Addition	No

detection attack. Nagaraja [28] first introduced a community hiding problem, where they added links based on centrality values. Waniek et al. [29] proposed a heuristic rewiring method to hide a community, by deleting the links within the same communities while adding the links across different communities. Because the links are random selected, the method is of relatively low effectiveness. And they also propose the concept of individual hiding, as a way to avoid an influential node being highlighted by three centrality measures (i.e., degree, closeness, betweenness). Valeria et al. [30] proposed a deception score to evaluate the effect of community deception, which takes the accuracy and recall of community detection into account, as well as the reachability of target community. The essence of this score is that the target community is expected to be divided into more new communities and each new community contains less percentage of target nodes. However, this may lead to the following two problems: first, it encourages more communities which might make the attack less concealed; second, when the number of communities is set to be constant, it can't guide the nodes from the target community to the optimal community to achieve better attack capacity. Quite recently, Chen et al. [35] proposed Q-Attack based on Genetic Algorithm (GA) and modularity, for the first time, to disturb the overall modular structure of a network.

Generally, we think there are following three types of community detection attacks of different scales.

- **Global attack (macroscale):** This attack is to achieve maximum community changes of the whole network by rewiring minimal number of links.
- **Target community attack (mesoscale):** This attack is also known as community deception, which is to hide one specific community by rewiring minimal number of links that are connected to at least one node in the community.
- **Target node attack (microscale):** This attack is to make target node belong to different communities by rewiring the minimal number of links around it. Note that our target node attack is different from the individual hiding proposed by Waniek et al. [29], since the latter has nothing to do with community detection.

In this paper, we formalize the community detection attack problem and present an Evolutionary Perturbation Attack (EPA) algorithm to attack community detection algorithms in three scales, i.e., macroscale (network), mesoscale (community) and microscale (node). The main differences between our method and the previous attack algorithms are summarized in TABLE 1. Moreover, we use a series of metrics to evaluate the attack results obtained by different attack methods on several real datasets. In particular, we make the following main contributions.

- To the best of our knowledge, this is the first time to

formalize the problem of community detection attack into three scales, i.e., macroscale (network), mesoscale (community) and microscale (node).
- We propose a novel EPA algorithm to attack community detection, which is capable of generating approximate optimal adversarial network with the minimal number of rewired links to launch all the three scales of attacks. We compare our EPA with other baseline attack methods on several synthetic networks and real-world networks, and find that EPA behaves the best in most cases, achieving the state-of-the-art results.
- We use GA to solve the optimization problem in EPA algorithm, and meanwhile introduce network structural properties including betweenness and the shortest path lengths between pairwise nodes into the mutation process to accelerate the convergence rate, making it faster to obtain the approximate optimal solution.
- We find that our EPA has outstanding transferability, i.e., the adversarial network generated by EPA against the louvain algorithm (LOU) can also be used to successfully fool other community detection algorithms.

The rest of paper is organized as follows. In Sec. 2, we formalize the problem of community detection attack and introduce our EPA method in details. Then, we compare the attack effects by utilizing EPA and other attack methods on a number of synthetic networks and real-world networks in Sec. 3, where we further use a variety of community detection algorithms to verify the transferability of EPA. Finally, we conclude the paper and highlight future research directions in Sec. 4.

2 COMMUNITY DETECTION ATTACK

2.1 Problem Formulation

First, we state and formalize the problem of community detection attack. Given a network $G=(V, E)$ as an undirected graph including a set of nodes V and a set of links E, suppose it can be divided into communities $C=\{C_1, C_2, \cdots, C_p\}$, with $C_i \subseteq V$ and $C_i \cap C_j=\varnothing$, for all $i,j \in \{1,2,\cdots,p\}$ and $i \neq j$. Then, we propose the concept of community detection attack, which is launched by adversarial network to fool community detection methods. We study this problem in three scales: global attack (macroscale), target community attacks (mesoscale) and target node attack (microscale).

Definition 1 (Adversarial network). Denoting the original network $G=(V, E)$ as the target, the adversarial attack selects some key links to construct an adversarial perturbation network $\hat{G}=(V, \hat{E}, \omega)$, where $\omega_{uv} \in \{-1,0,1\}$ is the weight on the $\hat{E}_{uv}$. Adversarial network $\bar{G}=(V, \bar{E})$,
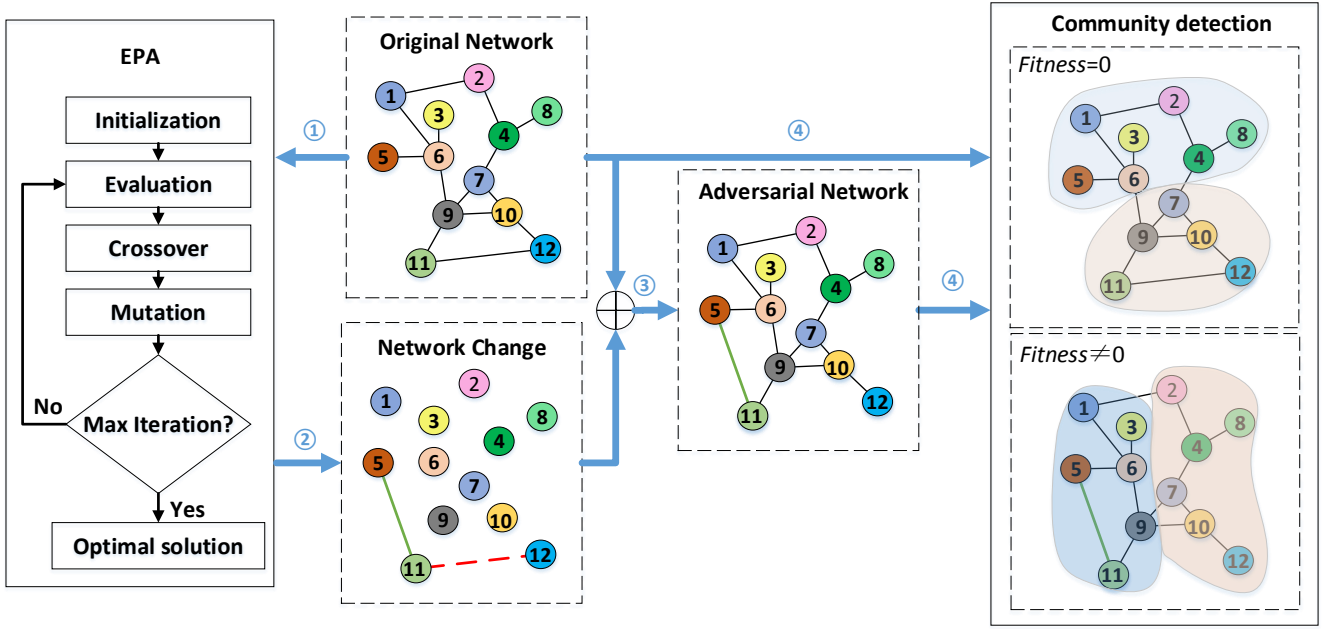
Fig. 1: The framework of EPA, which consists of four steps. First, an original network is input to EPA. Second, GA is used to generate the approximate optimal network change, denoted by the sets of added and deleted links. Third, an adversarial network is established by changing the links in the original network. Finally, various community detection algorithms are performed on the adversarial network to validate the attack effect of EPA.

generated by adding adversarial perturbation on the original network, is defined as

$$\bar{E}_{uv} = E_{uv} + \omega_{uv}\hat{E}_{uv}. \tag{1}$$

Problem 1 (Global Attack). Given a network $G=(V,E)$, generate an adversarial network $\bar{G}=(V,\bar{E})$ via attack strategy to make community detection method φ fail with budget β. $\bar{G}$ is divided into communities $\bar{C}=\{\bar{C}_1, \bar{C}_2, \cdots, \bar{C}_q\}$ by using the same community detection algorithm. Furthermore, E^+ (resp., E^-) denotes a set of link additions (resp., deletions) and the global attack is realized by solving the optimization problem:

$$\arg\max \{\phi(C, \bar{C}, E, \bar{E}, E^+, E^-, \beta)\}, \tag{2}$$

where $\beta = |E^+| = |E^-|$, meaning that we only consider the rewiring operation for global attack, and $\bar{E} = (E \cup E^+)\backslash E^-$. Since the links are rewired globally, we have

$$E^+ \subseteq \{(u,v) : u \in V \wedge v \in V, (u,v) \notin E\}, \tag{3}$$

$$E^- \subseteq \{(u,v) : u \in V \wedge v \in V, (u,v) \in E\}. \tag{4}$$

Problem 2 (Target Community Attack). Given a target community $C_i \subseteq C$, which is obtained by community detection method φ on the original network G. After target community attack, C_i cannot be detected by φ, i.e., in the adversarial network $\bar{G}$, the nodes in C_i belong to a set of new communities, denoted by $\dot{C}=\{\dot{C}_1, \dot{C}_2, \cdots, \dot{C}_r\} \subseteq \bar{C}$, with $\dot{C}_j \cap C_i \neq \varnothing, \forall j \in [1, r]$, which is realized by solving the optimization problem:

$$\arg\max \{\phi(C_i, \dot{C}, E, \bar{E}, E^+, E^-, \beta)\}. \tag{5}$$

Here, we still consider the rewiring process and thus have $\beta = |E^+| = |E^-|$, $\bar{E} = (E \cup E^+)\backslash E^-$. In order to hide community C_i, it would be best to disconnect the links inside the community while add connections between the nodes in C_i and those in other communities, thus we have

$$E^+ \subseteq \{(u,v) : u \in C_i \oplus v \in C_i, (u,v) \notin E\}, \tag{6}$$

$$E^- \subseteq \{(u,v) : u \in C_i \wedge v \in C_i, (u,v) \in E\}. \tag{7}$$

Problem 3 (Target Node Attack). Given a target node t, suppose it belongs to community C_i in the original network G, while it belongs to $\bar{C}_j$ in the adversarial network $\bar{G}$, then the target node attack is realized by solving the optimization problem:

$$\arg\max \{\delta \times \mu(E, \bar{E}, E^+, E^-, \beta)\}, \tag{8}$$

$$\delta = \begin{cases} 1 & \frac{|\bar{C}_j \cap C_i|}{|\bar{C}_j|} < \epsilon \\ 0 & else \end{cases}, \tag{9}$$

where μ is the function to measure the amount of network change. $\epsilon \in [0, 1]$ is a predefined threshold, based on which we determine whether communities C_i and $\bar{C}_j$ are close to each other or not. We think the attack is successful only when these two communities are relatively different from each other. Similarly, we still have $\beta = |E^+| = |E^-|$ and $\bar{E} = (E \cup E^+)\backslash E^-$. To make the attack more effective, at this time, we rewire links around the target node t, thus we have

$$E^+ \subseteq \{(u,t) : u \in V, (u,t) \notin E\}, \tag{10}$$

$$E^- \subseteq \{(u,t) : u \in V, (u,t) \in E\}. \tag{11}$$

TABLE 2: The definitions of main symbols.

Symbol	Definition
n	The number of nodes in network
m/m_t	The number of links in network/target community
β	The budget which is the number of rewired links
θ	The maximum number of rewired links
$C/\bar{C}$	The communities before/after the attack
M	The confusion matrix whose element is $m_{i,j}$
$M_{i.}/M_{.i}$	The number of nodes in new/original community C_i
$m_{i,j}/\tilde{m}_{i,j}$	The number of community C_i's nodes which originally belong/not belong to the target community C_j
E_{M_r}/E_{M_c}	The entropy of new/target communities
N_a/N_b	The number of communities in the control/test group
O_i	The i-th chromosome of the population
$(.)\|O_i$	The value under the attack represented by O_i

The function ϕ in Eqs. (2) and (5) is used to evaluate the attack gain, which will be specialized in Sec. 2.3. Note that the attack gain consists of two parts: the budget β, defined as the number of rewired links, and the attack effect. As we can see, the overall attack effect generally increases as the budget β grows, as a result, it is our focus to improve the attack gain with only limited budget β.

These indicate that community detection attack can always be represented as an optimization problem, which could be well solved by evolutionary computing methods, such as Genetic Algorithm (GA). In this paper, we propose a GA based Evolutionary Perturbation Attack (EPA) for community detection. In order to make the GA more suitable for finding appropriate attack strategy, we use rewired link ID as genes on the chromosome instead of binary coding, which can effectively reduce the storage space of the population. Considering that the number of rewired links is also a variable here, we adopt a strategy called *unequal crossover* to make the length of chromosome changeable during the crossover process. Moreover, a novel search mechanism based on network structural properties, including betweenness and the shortest path lengths between pairwise nodes, is introduced in the mutation process to accelerate the convergence of GA, making it faster to obtain the approximate optimal solution.

In particular, our EPA is established in four stages, including initialization, evaluation, crossover and mutation, which will be introduced one by one in the following. The flowchart of EPA is shown in Fig. 1, and the main symbols used in this paper are listed in TABLE 2 for convenience.

2.2 Initialization

First, we directly use the ID of rewired link as the gene of chromosome to facilitate the evolving. Specifically, we create the indexes for existent links and nonexistent links and treat them as link deletion and addition genes, respectively. To make the attack more concealed, we set the number of deleted links equal to that of added links, so that the total number of links in the network keeps constant. Denote the threshold of the number of rewired links by θ and the chromosome by O_i with the budget $\beta \in [1, \theta]$. We thus have

$$
\begin{aligned}
O_i &= \{A_i, B_i\} \\
&= \{a_i^1, a_i^2, \cdots, a_i^\beta, b_i^1, b_i^2, \cdots, b_i^\beta\},
\end{aligned} \quad (12)
$$
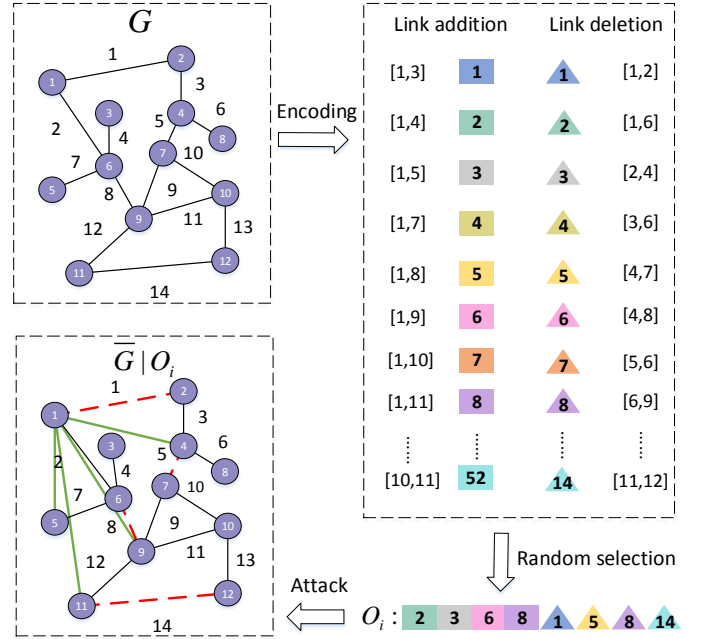
Fig. 2: An illustration of chromosome encoding and initialization. $\bar{G}|O_i$ is the adversarial network after the attack represented by chromosome O_i. The original network G contains 14 links and 52 disconnected pairs of nodes that have potential to be connected in the attack. The red dashed lines and the green solid lines represent the links that are deleted and added in the attack, respectively. In the left, the rectangles represent the link addition genes while the triangles denote the link deletion genes. Then, we initialize chromosome O_i by randomly selecting $\beta = 4$ link addition and deletion genes, respectively.

where O_i, with the index $i \in [1, P]$, represents the i-th chromosome in population and P is the population size of GA. A_i and B_i represent the sets of link addition and deletion links, respectively, in chromosome O_i, with the element a_i^k (or b_i^k) being the index of a node pair which are disconnected (or connected). The link change ω_{uv} between two nodes u and v, represented by chromosome O_i, is thus calculated by

$$
\omega_{uv} = \begin{cases} 1 & Index(u,v) \in A_i \\ -1 & Index(u,v) \in B_i \\ 0 & \text{otherwise} \end{cases}, \quad (13)
$$

where $Index(u,v)$ is the function to obtain the index of a node pair (u,v). An illustration to explain how chromosomes are coded and initialized is shown in Fig. 2.

Note that, to make the target node attack more effectively, instead of initializing randomly, we first select a target community randomly, then we preferentially delete the links between the target node and those not belonging to the target community, while establish links between the target node and those in the target community. If all nodes in the target community are connected to the target node, then we randomly select another target community and repeat the above steps.

2.3 Evaluation

2.3.1 Fitness Function

Each chromosome corresponds to an attack strategy. After encoding the attack strategies, we then need to evaluate the attack effect of each strategy using some fitness function. Note that the entropy is maximized when each new community consists of nodes uniformly from many different *real* communities. Since the uniform distribution emphasized by entropy makes the overall accuracy and recall as low as possible, we thus think it's an appropriate metric to evaluate the attack effect. For the global attack or target community attack, their fitness functions follow the same general form, represented by

$$\phi_i = \Psi(d'|O_i) \times X(C, \bar{C}|O_i), \qquad (14)$$

which consists of two parts: the attenuation function $\Psi \in [0,1]$ and the attack effect evaluation function X. Ψ is a monotonic decreasing function of $d'|O_i$, with d' being the normalization of d which represents the degree change after the attack represented by chromosome O_i. X is the function to evaluate the attack effect based on entropy, and the terms C and $\bar{C}|O_i$ denote the community detection results before and after the attack, respectively. For the target node attack, however, since the attack effect is binary, i.e., either success or failure, the fitness function only contains the first part, i.e., the change of target node degree.

2.3.2 Attenuation Function

In order to perform the attack with limited budget β, we use the exponential decay function as our attenuation function, described as follows:

$$\Psi(d'|O_i) = \exp(c \times d')|O_i, \qquad (15)$$

where $d' = d/m$ (m is the number of links in the whole network) for global attack while $d' = d/m_t$ (m_t is the number of links in the target community) for target community attack; and c is a constant that controls the decay speed. For a network of n nodes, the degrees of all nodes before and after the attack are denoted by $D=\{d_1, d_2, \cdots, d_n\}$ and $\bar{D}=\{\bar{d}_1, \bar{d}_2, \cdots, \bar{d}_n\}$, respectively. Then, the distance d between them is calculated by

$$d = \frac{1}{4} \sum_{i=1}^{n} |d_i - \bar{d}_i|, \qquad (16)$$

where d_i and $\bar{d}_i$ are the degrees of node i before and after the attack, respectively. Note that when adding and deleting links on totally different nodes, the degree distance d tends to be equal to the number of rewired links. However, if we delete a link around a node whenever we add a link to it, its degree will not change, and thus the degree distance must be equal to 0 all the time. In this study, the latter is considered more concealed and thus is preferred according to Eq. (15).

2.3.3 Attack Effect

Given a confusion matrix M with each element m_{ij} representing the number of the shared nodes between the original community C_i and the new community $\bar{C}_j$, we define the function X to evaluate the attack effect:

$$X(C, \bar{C}|O_i) = (E'_{M_r} + E'_{M_c})|O_i, \qquad (17)$$

where E'_{M_r} and E'_{M_c} are the normalized entropy of M_r and M_c, and thus X must be in the range of $[0,2]$. Suppose E_{M_r} and E_{M_c} are the entropy of new communities and target communities, which are obtained by considering the row vectors and column vectors of M, respectively. Suppose the matrix M dimension is $N_a \times N_b$, with N_a and N_b being the numbers of communities in control group (i.e., real community number) and test group (i.e., community number detected by certain community detection algorithm after the attack), respectively, the maximum values of E_{M_r} (resp., E_{M_c}) E_{M_c} are obtained when the values of each row (resp., column) are equal. In this case, the calculation of E_{M_r} and E_{M_c} is simplified to

$$\max E_{M_r} = -\sum_{i=1}^{N_a} \frac{M_{i.}}{n} \left(\frac{1}{N_b} \log_2 \frac{1}{N_b} \times N_b \right) = \log_2 N_b, \quad (18)$$

$$\max E_{M_c} = -\sum_{i=1}^{N_b} \frac{M_{.i}}{n} \left(\frac{1}{N_a} \log_2 \frac{1}{N_a} \times N_a \right) = \log_2 N_a, \quad (19)$$

where n is the number of nodes in network. $M_{i.}$ and $M_{.i}$ represent the numbers of nodes in new and original community C_i, respectively.

For global attack, the entropy for new communities E_{M_r} and that for target communities E_{M_c} are calculated by

$$E_{M_r} = -\sum_{i=1}^{N_a} \sum_{j=1}^{N_b} \frac{M_{i.}}{n} \left(\frac{m_{i,j}}{M_{i.}} \log_2 \frac{m_{i,j}}{M_{i.}} \right), \qquad (20)$$

$$E_{M_c} = -\sum_{i=1}^{N_b} \sum_{j=1}^{N_a} \frac{M_{.i}}{n} \left(\frac{m_{j,i}}{M_{.i}} \log_2 \frac{m_{j,i}}{M_{.i}} \right), \qquad (21)$$

where m_{ij} is the number of nodes in new community C_i that originally belong to community C_j.

Suppose all the communities keep exactly the same after the attack, we have $E_{M_r} = E_{M_c} = 0$. With Eq. (18) we can conclude that E_{M_r} and E_{M_c} must be in the range of $[0, \log_2 N_b]$ and $[0, \log_2 N_a]$, respectively. Therefore, based on Eq. (14) and Eq. (17), the fitness of O_i for global attack is defined as

$$\phi_i = \Psi(d'|O_i) \times \left(\frac{E_{M_r}}{\log_2 N_b} + \frac{E_{M_c}}{\log_2 N_a} \right)|O_i. \qquad (22)$$

For target community attack, here we limit that any added or deleted link must be connected to at least one node in the target community, which greatly reduces the searching space of solutions. In this case, M is an $N_a \times 2$ matrix with element m_{ij} (resp., $\tilde{m}_{i,j}$) being the number of nodes in community C_i that originally belong (resp., don't belong) to target community C_j. For target community C_j, E_{M_r} and E_{M_c} are defined as

$$E_{M_r} = -\sum_{i=1}^{N_a} \frac{M_{i.}}{n} \left(\frac{m_{i,j}}{M_{i.}} \log_2 \frac{m_{i,j}}{M_{i.}} + \frac{\tilde{m}_{i,j}}{M_{i.}} \log_2 \frac{\tilde{m}_{i,j}}{M_{i.}} \right), \quad (23)$$

$$E_{M_c} = -\sum_{i=1}^{N_a} \frac{m_{i,j}}{M_{.i}} \log_2 \frac{m_{i,j}}{M_{.i}}. \qquad (24)$$

Based on the above definitions, we always have $\tilde{m}_{i,j}=M_{i.} - m_{i,j}$. The remaining variables are defined the same as those in Eq. (20) and Eq. (21).
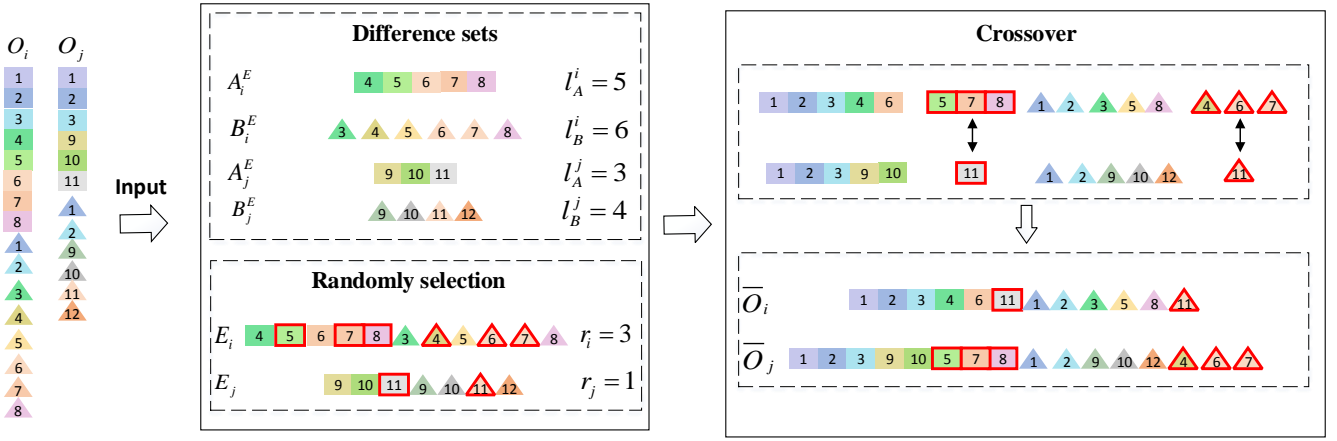
Fig. 3: An example of non-equal crossover operation. O_i and O_j (resp., $\bar{O}_i$ and $\bar{O}_j$) are the chromosomes before (resp., after) the crossover. l_A^i and l_B^i (resp. l_A^j and l_B^j) are the numbers of exchangeable addition and deletion genes, respectively, on chromosome O_i (resp. O_j). Furthermore, r_i and r_j are the numbers of crossed genes in O_i and O_j, respectively.

Similarly, with Eq. (18) we can conclude that E_{M_r} and E_{M_c} must be in the range of $[0, 1]$ and $[0, \log_2 N_a]$, respectively. Therefore, the fitness of O_i for target community attack is defined as

$$\phi_i = \Psi(d') \times \left(E_{M_r} + \frac{E_{M_c}}{\log_2 N_a} \right). \quad (25)$$

It should be noted that our fitness function based on entropy is somewhat similar to the deception score H mentioned in [30], i.e., both of them have a tendency to spread more nodes of the target community into more communities. The main difference is that by using normalization, in our method, the number of new communities will not be too large, making the attack more concealed.

Finally, for target node attack, it is quite easy to hide a node by rewiring links. In this study, however, we try to hide a node by only adding links to the target node. We believe that adding, rather than deleting, certain links are much more convenient for social network users. For target node t, the fitness of chromosome O_i thus is calculated by

$$\phi_i = \begin{cases} \frac{d_t}{d_t + \beta|O_i|} & \text{successful attack} \\ 0 & \text{otherwise} \end{cases}, \quad (26)$$

where d_t is the degree of target node t before the attack and $\beta|O_i|$ is the length of chromosome O_i. Thus, $d_t + \beta|O_i|$ is the degree of the target node after the attack represented by O_i.

For any kind of attack, after evaluating each individual in GA, we further use roulette selection method to select offspring. If the fitness of each individual in the population is ϕ_i $(i = 1, 2, \cdots, M)$ and the size of the population is M, the selection probability of individual O_i is calculated by

$$P(O_i) = \frac{\phi_i}{\sum_{j=1}^{M} \phi_j}. \quad (27)$$

2.4 Crossover

Traditionally, the length of chromosomes is set the same and is fixed throughout the evolution in GA. Here, we prefer to use *non-equal* crossover, where the length of chromosomes can change in the process of crossover, so that we can find the optimal solution with the smallest budget β, which is described by the following steps.

1) Choose two chromosomes O_i and O_j and form two subsets $E_i = \{A_i^E, B_i^E\} \subseteq O_i$ and $E_j = \{A_j^E, B_j^E\} \subseteq O_j$ by removing the identical elements that are considered nonexchangeable, where A_i^E (or A_j^E) and B_i^E (or B_j^E) are the *exchangeable* gene sets of addition and deletion, respectively, in chromosomes O_i (or O_j).
2) Calculate the lengths of A_i^E, B_i^E, A_j^E and B_j^E, and denote them by l_A^i, l_B^i, l_A^j and l_B^j, respectively.
3) Generate two random integers r_i and r_j in the intervals $[1, min(l_A^i, l_B^i)]$ and $[1, min(l_A^j, l_B^j)]$, respectively.
4) Suppose the numbers of rewired links by chromosomes O_i and O_j are β_i and β_j, respectively, and the threshold is θ. If $\beta_i - r_i + r_j$ or $\beta_j + r_i - r_j \notin [1, \theta]$, go to step 3); otherwise, go to step 5).
5) select r_i addition and deletion genes, respectively, from O_i, and select r_j addition and deletion genes, respectively, from O_j; exchange the genes selected from O_i with those selected from O_j, to generate two new chromosomes $\bar{O}_1$ and $\bar{O}_2$.

The whole process of crossover is shown in Fig. 3.

2.5 Mutation

Here, we further utilize the structural information around pairwise nodes as heuristic information to accelerate the searching process of GA.

For link addition, two nodes are of less similarity if the network distance (or the shortest path length) between them is relatively long. Therefore, in order to make the attack more effective, it is better to add the links between pairwise nodes of longer distance. Suppose the distance between a pair of disconnected nodes i and j is λ_{ij}, then the probability that a link addition gene to create a link between these two nodes is generated by mutation is defined as

$$P(a_k) = \frac{\lambda_{ij}}{\sum_{(i,j) \notin E} \lambda_{ij}}, \quad (28)$$

where a_k is the index of the pair of nodes (i, j).

For link deletion, the links inside communities always have lower betweenness than those across different communities. Therefore, in order to make the attack more effective, it is better to delete the links of lower betweenness. Given a link with its index b_k, its betweenness is calculated by

$$C_B(b_k) = \sum_{s,t \in V} \frac{\sigma(s,t|b_k)}{\sigma(s,t)}, \tag{29}$$

where V is the node set in the network, $\sigma(s,t)$ represents the total number of shortest paths between nodes s and t, and $\sigma(s,t|b_k)$ represents the number of shortest paths through the link. Then, the probability that a link deletion gene to remove the link is generated by mutation is defined as

$$P(b_k) = \frac{\frac{1}{C_B(b_k)}}{\sum_{k=1}^{m} \frac{1}{C_B(b_k)}}, \tag{30}$$

where m is the total number of links in the network.

3 EXPERIMENTS

In this part, we will perform the three kinds of attacks on several synthetic networks and real-world networks. For global attack, we first compare our EPA with three heuristic algorithms under different budges β, and we also design the experiment to show the ability of EPA to find the optimal budget. For target community attack, we first use EPA to get the optimal budget and then compare EPA with the other algorithms under this budget. For target node attack, we select some representative nodes as targets. For each experiment, we run 10 times and record the mean result. Our experimental environment consists of i7-8700 3.2GHz (CPU), GTX 1050Ti 4GB (GPU), 16GB memory and Windows 10.

3.1 Datasets

To evaluate the attack effect of EPA, we use three community detection algorithms, i.e., greedy (GRE) [13], Infomap Algorithm (INF) [21] and Louvain (LOU) [36], on the six synthetic networks and three real networks, described as follows, with their basic properties presented in TABLE 3 and TABLE 4, respectively. The descriptions of parameters for generating a synthetic network are listed in TABLE 5 for convenience.

- **The synthetic networks [37]:** These networks are generated by LFR benchmark, all of which are undirected and unweighted networks.
- **The USA college football (Football) [14]:** This network represents the matches between American football teams during the season of 2000.
- **email-Eu-core network (Email) [38]:** The network represents the emails between members of a large European research institution.
- **Political blogs (Pol.Blogs) [39]:** This network represents the political leaning collected from blog directories.

In experiments, we only consider undirected networks and remove the isolated nodes from data sets since they are meaningless in community detection. The three community

TABLE 3: The basic properties of the six synthetic networks.

Network	N	k	max k	μ	min c	max c
N-1000-μ-0.3	1000	10	50	0.3	50	100
N-1000-μ-0.5	1000	10	50	0.5	50	100
N-3000-μ-0.3	3000	10	50	0.3	100	200
N-3000-μ-0.5	3000	10	50	0.5	100	200
N-5000-μ-0.3	5000	15	100	0.3	100	200
N-5000-μ-0.5	5000	15	100	0.5	100	200

TABLE 4: The basic properties of the three networks.

Network	#Nodes	#Links	#Communities
Football	115	613	12
Email	1005	25571	42
Pol.Books	1490	19090	2

TABLE 5: The meaning of parameters in synthetic networks.

Parameter	Meaning
N	number of nodes
k	average degree
max k	maximum degree
μ	mixing parameter
min c	minimum community size
max c	maximum community size

detection algorithms we adopt are also briefly introduced as follows to make the paper self-contained.

- **GRE [13]:** Each node is considered as a separate community initially and the communities are fused in the direction of maximum increment of modularity Q.
- **INF [21]:** It's an information theory based algorithm which strive to compress the average description length for a random walk.
- **LOU [36]:** This is a modularity based algorithm which can generate a hierarchical community structure by compressing the communities continuously.

3.2 Baseline Attack Methods

Inspired by various community detection algorithms, we use the following four heuristic attack methods as the baselines for global attack.

- A_Q: A GA based method where the modularity Q is used to design the fitness function [35].
- A_S: Rather than using the entropy-based fitness function, here we use the average of deception score [30] as the fitness function and the rest is the same as EPA.
- A_B: Deleting the links with the highest betweenness centrality, while adding the links between the nodes with the longest distance.
- A_D: Deleting the links with the largest sum of degrees of their terminal nodes, while adding the links between the nodes with the longest distance.

For target community attack, we use the safeness based deception algorithm D_s [30] and random algorithm D_w [29] as the baseline attack methods, which can effectively hide the target community against different community detection algorithms, and they are briefly introduced as follows.

- D_s: Both link addition and deletion aim to maximize the safeness of target community defined in [30].

- D_w: This method randomly adds links inter different communities while deletes links intra communities.

For target node attack, we propose a heuristic algorithm D_r which randomly add links between target node and the nodes in other communities.

3.3 Performance Metrics

In order to verify the effectiveness of our EPA, we compare it with other baseline attack methods by using a series of metrics.

For global attack, we use Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI) to evaluate the community detection results. Then, we further evaluate the attack effects by comparing their values before and after attacks. In particular, NMI and ARI are defined as

$$\text{NMI} = \frac{-2\sum_{i=1}^{N_a}\sum_{j=1}^{N_b} m_{ij}\log(\frac{m_{ij}n}{M_iM_j})}{\sum_{i=1}^{N_a} M_i\log(\frac{M_i}{n}) + \sum_{j=1}^{N_b} M_j\log(\frac{M_j}{n})}, \quad (31)$$

$$\text{ARI} = \frac{\sum_{ij}\binom{m_{ij}}{2} - [\sum_i\binom{M_i}{2}\sum_i\binom{M_j}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_i\binom{M_i}{2} + \sum_j\binom{M_j}{2}] - [\sum_i\binom{M_i}{2}\sum_j\binom{M_j}{2}]/\binom{n}{2}}, \quad (32)$$

where M_i and M_j are the sums over row i and column j, respectively. N_a and N_b are the numbers of communities in control group (i.e., real community number) and test group (i.e., community number detected by certain community detection algorithm), respectively.

For target community attack, in addition to the fitness, we also use the deception score H [30] to evaluate the attack effect, which is defined as

$$H = \left[1 - \frac{|S(C)| - 1}{|C| - 1}\right] \times \left[\frac{1}{2}(1 - \max(R)) + \frac{1}{2}(1 - \bar{P})\right], \quad (33)$$

where $|S(C)|$ is the number of connected components in the subgraph induced by the members in C. $\bar{P}$ and R are the mean precision and recall rate, respectively.

For target node attack, we just use the percentage of target node degree increment in the attack to evaluate the results.

3.4 Experimental Results

3.4.1 Global Attack

In global attack, we fix the budget and rewire $k\%$ of links to compare our EPA with four baseline methods including A_Q, A_S, A_B and A_D. We choose LOU as the basic community detection algorithm, and meanwhile we also use the GRE and INF algorithms to verify the black-box attack effect. The population used in the experiment is 100, the maximum number of iterations is 200, the crossover and mutation rates are 0.6 and 0.1, respectively. TABLE 6 represents the community detection results before attack.

The community detection results, in terms of NMI and ARI, on different datasets obtained by various community detection algorithms, after the attacks by EPA and the four baseline attack methods for various percentages of rewired links, are presented in Fig. 4. Generally, we can find that, our EPA has the best attack effect on each performance metric, i.e., leading to smaller NMI and ARI, in most cases.

TABLE 6: The community detection results before attack.

Network	GRE		INF		LOU	
	NMI	ARI	NMI	ARI	NMI	ARI
Football	0.73	0.49	0.92	0.90	0.73	0.49
Email	0.45	0.16	0.68	0.30	0.58	0.33
Pol.Blog	0.69	0.79	0.52	0.65	0.64	0.77
N-1000-μ-0.3	0.73	0.54	0.97	0.95	0.99	0.98
N-1000-μ-0.5	0.37	0.19	0.78	0.70	0.85	0.74
N-3000-μ-0.3	0.63	0.34	0.96	0.96	1.0	1.0
N-3000-μ-0.5	0.28	0.14	0.71	0.56	0.57	0.41
N-5000-μ-0.3	0.75	0.44	1.00	1.00	1.0	1.0
N-5000-μ-0.5	0.42	0.15	0.95	0.93	0.98	0.95

More specifically, we find that EPA performs especially well on Pol.Blogs network, i.e., rewiring 4% links can decrease both NMI and ARI to near 0.3. This may be because LOU performs quite well in revealing the community structure of this network, and meanwhile this network is also easy to be disturbed since the connections are much sparse. In fact, none of the attack methods performs well on LFR generated networks with small mixing parameter, i.e., NMI and ARI still keep relatively high after any kind of attack, since the communities of these networks are relatively dense and thus are easier to be detected. Indeed, for larger mixing parameter, all attacks behave better, leading to smaller NMI and ARI. By comparison, EPA outperforms all the others on synthetic networks, no matter for large or small mixing parameters. Besides, we can also find that white-box attack (LOU) behaves better than black-box attacks (GRE and INF), which is quite intuitive since all the attack methods here are based on the LOU algorithm. More interestingly, by comparison, the communities detected by INF are relatively robust than those detected by GRE, under the attack of LOU based EPA. This may be because both GRE and LOU are modularity-based algorithms, while INF is based on information theory.

Now, let's focus on the influence of the attenuation factor c which controls the degree that the budget β penalizes the fitness and ultimately controls the optimal budget generate by EPA. We thus compare the attack results under different values of c, and fix the other parameters. For each value, we run the experiment 10 times and report the mean results in TABLE 7, where we can see that as c increases from 3 to 6, the final budget generated by EPA significantly decreases with a little bit sacrifice of attack effects, i.e., NMI and ARI increases slightly as c increases. We thus suggest to use relatively large value of c if we want to get a more concealed attack, while use relatively small value of c if the optimal attack effect is pursued.

3.4.2 Target Community Attack

Target community attack is also known as *community deception*. As an example, we visualize the attack effect of our EPA on the Dolphin network using t-SNE algorithm [40], as shown in Fig. 5. Here, we compare EPA with safeness based deception algorithm D_s [30] and random algorithm D_w [29]. In order to make fair comparison, we fix the budget β for each algorithm, i.e., we first use our EPA to find the optimal budget and then use it as the input of D_s and D_w. Again, we record the mean values of fitness and deception

Football

Metric	Algorithm	k=1 GRE	INF	LOU	k=2 GRE	INF	LOU	k=3 GRE	INF	LOU	k=4 GRE	INF	LOU	k=5 GRE	INF	LOU
NMI	EPA	.67	.88	.78	.65	.87	.77	.63	.86	.75	.62	.85	.73	.59	.81	.71
	A_O	.72	.92	.80	.74	.92	.78	.69	.92	.77	.72	.91	.75	.69	.90	.73
	A_S	.72	.90	.83	.71	.88	.80	.69	.88	.79	.70	.86	.79	.71	.86	.78
	A_B	.73	.92	.89	.79	.92	.89	.78	.92	.89	.72	.92	.89	.71	.92	.89
	A_D	.76	.92	.85	.77	.92	.85	.72	.92	.88	.78	.92	.86	.73	.92	.86
ARI	EPA	.42	.85	.56	.39	.85	.54	.35	.84	.53	.34	.84	.50	.30	.79	.47
	A_O	.44	.90	.61	.47	.90	.55	.49	.90	.54	.48	.89	.46	.48	.88	.47
	A_S	.50	.86	.65	.47	.85	.59	.41	.85	.58	.46	.84	.58	.48	.83	.57
	A_B	.49	.90	.80	.56	.90	.80	.55	.90	.80	.48	.90	.81	.42	.90	.82
	A_D	.53	.90	.72	.55	.90	.72	.48	.90	.72	.55	.90	.79	.50	.90	.71

Email

Metric	Algorithm	k=1 GRE	INF	LOU	k=2 GRE	INF	LOU	k=3 GRE	INF	LOU	k=4 GRE	INF	LOU	k=5 GRE	INF	LOU
NMI	EPA	.37	.61	.51	.34	.58	.50	.32	.57	.48	.32	.55	.46	.29	.54	.42
	A_O	.43	.64	.51	.41	.61	.51	.40	.60	.48	.40	.59	.46	.38	.58	.42
	A_S	.39	.61	.57	.36	.62	.55	.35	.60	.55	.34	.61	.53	.34	.60	.53
	A_B	.41	.63	.56	.39	.61	.54	.37	.60	.50	.36	.59	.50	.31	.59	.49
	A_D	.41	.63	.53	.40	.61	.53	.38	.60	.51	.37	.61	.53	.32	.61	.52
ARI	EPA	.14	.28	.25	.12	.27	.22	.12	.26	.24	.12	.25	.21	.10	.23	.22
	A_O	.16	.30	.23	.15	.28	.23	.15	.29	.25	.15	.27	.19	.14	.25	.18
	A_S	.15	.28	.34	.14	.29	.32	.14	.28	.32	.13	.28	.31	.13	.26	.31
	A_B	.16	.29	.32	.16	.28	.30	.15	.27	.26	.14	.26	.24	.11	.26	.25
	A_D	.16	.29	.25	.16	.28	.27	.15	.28	.25	.15	.28	.26	.12	.29	.27

Pol.Blogs

Metric	Algorithm	k=1 GRE	INF	LOU	k=2 GRE	INF	LOU	k=3 GRE	INF	LOU	k=4 GRE	INF	LOU	k=5 GRE	INF	LOU
NMI	EPA	.57	.44	.53	.43	.36	.45	.42	.33	.34	.34	.31	.31	.32	.28	.29
	A_O	.62	.47	.58	.52	.43	.47	.49	.37	.45	.48	.37	.44	.36	.34	.35
	A_S	.61	.48	.57	.48	.39	.50	.46	.38	.43	.40	.37	.38	.36	.33	.36
	A_B	.65	.51	.60	.63	.52	.60	.61	.52	.58	.60	.50	.52	.60	.51	.51
	A_D	.71	.51	.73	.69	.52	.59	.68	.52	.58	.65	.47	.55	.65	.50	.49
ARI	EPA	.73	.57	.68	.60	.49	.58	.58	.46	.38	.52	.43	.33	.47	.41	.31
	A_O	.76	.60	.69	.67	.54	.52	.65	.50	.62	.61	.50	.58	.53	.47	.38
	A_S	.76	.60	.72	.67	.52	.64	.65	.51	.58	.59	.51	.53	.52	.44	.48
	A_B	.74	.61	.69	.74	.61	.68	.70	.61	.65	.68	.59	.58	.68	.58	.56
	A_D	.82	.61	.71	.81	.61	.71	.79	.61	.70	.79	.56	.65	.78	.56	.54

N-1000-μ-0.3

Metric	Algorithm	k=1 GRE	INF	LOU	k=2 GRE	INF	LOU	k=3 GRE	INF	LOU	k=4 GRE	INF	LOU	k=5 GRE	INF	LOU
NMI	EPA	.71	.93	.95	.68	.92	.93	.65	.90	.92	.64	.89	.91	.62	.88	.90
	A_O	.73	.96	.95	.70	.96	.95	.67	.94	.92	.70	.91	.91	.69	.90	.89
	A_S	.78	.96	.98	.72	.94	.97	.70	.94	.96	.68	.92	.95	.68	.92	.95
	A_B	.80	.94	.98	.72	.93	.97	.69	.92	.95	.69	.91	.95	.68	.90	.95
	A_D	.76	.94	.97	.71	.93	.95	.68	.93	.94	.68	.92	.93	.67	.92	.93
ARI	EPA	.51	.92	.91	.47	.92	.88	.45	.88	.86	.46	.87	.86	.45	.86	.84
	A_O	.54	.95	.92	.53	.95	.89	.49	.92	.90	.53	.89	.87	.52	.88	.82
	A_S	.60	.95	.98	.57	.93	.97	.56	.93	.96	.51	.90	.95	.51	.90	.95
	A_B	.61	.94	.97	.58	.92	.95	.56	.90	.95	.56	.89	.95	.56	.88	.95
	A_D	.59	.94	.96	.57	.93	.95	.55	.91	.93	.53	.90	.92	.52	.89	.92

N-3000-μ-0.3

Metric	Algorithm	k=1 GRE	INF	LOU	k=2 GRE	INF	LOU	k=3 GRE	INF	LOU	k=4 GRE	INF	LOU	k=5 GRE	INF	LOU
NMI	EPA	.63	.94	.98	.60	.91	.97	.58	.90	.97	.58	.88	.96	.57	.86	.95
	A_O	.67	.95	.98	.65	.94	.98	.65	.93	.97	.64	.92	.95	.61	.90	.94
	A_S	.66	.96	.99	.64	.95	.99	.62	.95	.98	.61	.94	.98	.61	.92	.97
	A_B	.67	.95	.99	.64	.93	.99	.63	.90	.98	.60	.89	.98	.60	.89	.98
	A_D	.67	.95	.99	.63	.95	.98	.62	.93	.98	.62	.93	.97	.62	.93	.97
ARI	EPA	.37	.93	.94	.32	.90	.93	.31	.89	.93	.31	.87	.92	.30	.84	.91
	A_O	.44	.94	.94	.39	.93	.94	.39	.92	.93	.38	.90	.90	.36	.88	.89
	A_S	.39	.95	.99	.38	.94	.99	.38	.94	.98	.35	.92	.98	.34	.91	.96
	A_B	.40	.94	.99	.38	.93	.99	.37	.91	.99	.37	.89	.98	.34	.88	.98
	A_D	.42	.95	.99	.38	.94	.98	.36	.93	.97	.36	.93	.97	.36	.93	.96

N-5000-μ-0.3

Metric	Algorithm	k=1 GRE	INF	LOU	k=2 GRE	INF	LOU	k=3 GRE	INF	LOU	k=4 GRE	INF	LOU	k=5 GRE	INF	LOU
NMI	EPA	.65	.98	.99	.64	.96	.98	.63	.95	.98	.62	.94	.96	.61	.92	.95
	A_O	.76	1.0	1.0	.77	1.0	.99	.75	.99	1.0	.74	.99	.99	.70	.98	.99
	A_S	.70	1.0	1.0	.70	1.0	1.0	.66	1.0	.99	.65	.99	.99	.65	.94	.99
	A_B	.65	.98	1.0	.65	.97	.99	.65	.96	.99	.64	.96	.98	.64	.95	.97
	A_D	.67	.98	.98	.67	.97	.98	.63	.97	.98	.62	.96	.97	.62	.95	.96
ARI	EPA	.31	.98	.97	.31	.96	.96	.28	.94	.95	.28	.92	.94	.27	.90	.93
	A_O	.46	1.0	1.0	.45	1.0	.99	.44	.99	.99	.44	.99	.99	.38	.98	.99
	A_S	.36	1.0	1.0	.36	1.0	1.0	.32	.99	.99	.31	.99	.99	.31	.91	.99
	A_B	.32	.98	.99	.31	.96	.99	.31	.95	.99	.30	.93	.97	.30	.91	.95
	A_D	.34	.98	.98	.32	.96	.97	.29	.95	.95	.30	.94	.95	.28	.92	.94

N-1000-μ-0.5

Metric	Algorithm	k=1 GRE	INF	LOU	k=2 GRE	INF	LOU	k=3 GRE	INF	LOU	k=4 GRE	INF	LOU	k=5 GRE	INF	LOU
NMI	EPA	.33	.75	.60	.29	.73	.56	.29	.72	.51	.28	.70	.50	.27	.67	.48
	A_O	.38	.76	.60	.34	.76	.57	.32	.74	.53	.32	.72	.50	.28	.71	.45
	A_S	.37	.76	.75	.35	.75	.72	.35	.74	.71	.32	.73	.60	.31	.73	.58
	A_B	.37	.77	.70	.40	.75	.78	.35	.74	.63	.40	.72	.60	.36	.70	.60
	A_D	.34	.78	.79	.31	.78	.71	.31	.77	.69	.33	.77	.64	.31	.79	.67
ARI	EPA	.17	.66	.44	.16	.62	.40	.16	.61	.36	.16	.58	.35	.15	.55	.34
	AQ	.21	.67	.44	.20	.65	.40	.17	.64	.37	.18	.62	.32	.15	.60	.32
	A_S	.23	.67	.65	.22	.64	.61	.21	.63	.58	.18	.62	.47	.18	.63	.42
	A_B	.25	.69	.62	.28	.66	.72	.21	.64	.49	.28	.60	.47	.22	.59	.46
	A_D	.31	.69	.72	.26	.70	.58	.25	.67	.55	.22	.68	.49	.23	.70	.55

N-3000-μ-0.5

Metric	Algorithm	k=1 GRE	INF	LOU	k=2 GRE	INF	LOU	k=3 GRE	INF	LOU	k=4 GRE	INF	LOU	k=5 GRE	INF	LOU
NMI	EPA	.27	.68	.49	.26	.67	.46	.24	.65	.43	.21	.65	.41	.20	.61	.39
	A_O	.28	.69	.50	.26	.69	.47	.25	.67	.45	.22	.66	.44	.21	.64	.44
	A_S	.35	.70	.67	.33	.69	.61	.29	.67	.56	.24	.66	.54	.19	.66	.51
	A_B	.31	.68	.63	.30	.68	.63	.32	.67	.60	.28	.66	.53	.24	.64	.50
	A_D	.29	.71	.59	.28	.71	.56	.28	.70	.62	.27	.70	.47	.19	.71	.46
ARI	EPA	.15	.54	.33	.14	.52	.31	.12	.50	.27	.11	.50	.26	.09	.45	.24
	A_O	.15	.55	.34	.14	.55	.32	.13	.52	.33	.11	.50	.31	.11	.48	.31
	A_S	.22	.56	.50	.19	.53	.44	.17	.53	.40	.13	.50	.39	.09	.50	.38
	A_B	.20	.54	.52	.20	.54	.52	.21	.53	.46	.18	.51	.38	.14	.48	.37
	A_D	.16	.57	.43	.15	.58	.39	.15	.57	.44	.14	.58	.32	.08	.59	.32

N-5000-μ-0.5

Metric	Algorithm	k=1 GRE	INF	LOU	k=2 GRE	INF	LOU	k=3 GRE	INF	LOU	k=4 GRE	INF	LOU	k=5 GRE	INF	LOU
NMI	EPA	.37	.89	.92	.33	.87	.91	.30	.85	.87	.30	.82	.87	.25	.81	.85
	A_O	.51	.92	.93	.47	.92	.95	.44	.90	.92	.41	.88	.90	.41	.86	.90
	A_S	.46	.94	.97	.45	.92	.95	.44	.91	.94	.44	.90	.94	.40	.89	.94
	A_B	.41	.91	.94	.40	.89	.93	.38	.88	.91	.37	.86	.89	.35	.85	.89
	A_D	.42	.94	.95	.40	.93	.94	.35	.93	.93	.35	.92	.91	.37	.92	.92
ARI	EPA	.13	.86	.72	.10	.83	.70	.10	.79	.71	.10	.70	.69	.08	.70	.68
	A_O	.20	.90	.76	.17	.89	.89	.16	.86	.83	.15	.85	.81	.14	.81	.82
	A_S	.21	.91	.93	.20	.90	.88	.19	.88	.85	.19	.87	.85	.15	.86	.85
	A_B	.15	.89	.86	.15	.85	.82	.13	.83	.79	.13	.80	.76	.12	.76	.75
	A_D	.15	.92	.88	.15	.90	.88	.12	.89	.85	.12	.87	.81	.13	.86	.80

Fig. 4: The attack effects, reflected by the change of NMI and ARI, obtained by EPA and the four baseline attack methods by rewiring $k\%$ of links, with k varies from 1 to 5, for the three real-world networks and six synthetic networks. Darker colors represent better attack performances and the best performance of each k is shown in red box.

TABLE 7: The global attack results obtained by EPA under various attenuation factor c.

Network	c=3 β	NMI	ARI	c=4 β	NMI	ARI	c=5 β	NMI	ARI	c=6 β	NMI	ARI
Football	12.8	0.76	0.53	10.6	0.77	0.54	8.6	0.77	0.55	7.8	0.78	0.57
Email	566.6	0.48	0.22	506.4	0.49	0.23	134.5	0.51	0.25	113.7	0.53	0.26
Pol.Blogs	1947.1	0.33	0.35	1690.9	0.34	0.35	1172.8	0.37	0.36	1125.5	0.38	0.39

score H for all detected communities with at least 10 nodes on each network, as presented in TABLE 8-TABLE 9. Note that here, for synthetic networks, we only give the results of INF and LOU algorithms, since GRE is of high time complexity and thus is quite time-consuming on networks including more than thousands of nodes, especially we need to attack each community in a network.

We find that, again, our EPA behaves significantly better, in terms of much higher fitness and deception score H, than both D_s and D_w. And such results are quite robust. For instance, for the Football network, it seems that D_s and D_w lose their attack effects on INF and LOU, i.e., the values of Fitness and H are quite small by comparing with those on GRE, as presented in TABLE 8. However, by using our EPA, their values still keep relatively large for all the three community detection algorithms, indicating that EPA is effective in both white-box and black-box situations. Moreover, it seems that the synthetic networks with smaller mixing parameters μ always have lower fitness in the corresponding cases. The reason may be that the networks of smaller μ tend
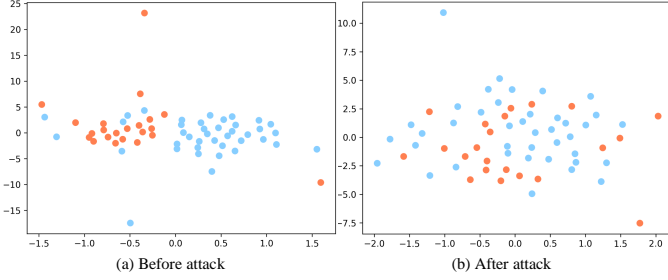
Fig. 5: The t-SNE visualization of community deception obtained by EPA on Dolphin network. The nodes of same color belongs to the same community detected by GRE, and the target community is marked by blue. (a) Before attack; (b) After attack.

TABLE 8: Deception results on real-world networks.

dataset	Alg	GRE		INF		Lou	
		Fitness	H	Fitness	H	Fitness	H
Football	EPA	**0.62**	**0.58**	**0.48**	**0.41**	**0.54**	**0.46**
	D_s	0.26	0.33	0.04	0.08	0.10	0.16
	D_w	0.16	0.22	0.00	0.00	0.06	0.09
Email	EPA	**0.68**	**0.57**	**0.54**	**0.45**	**0.78**	**0.66**
	D_s	0.40	0.52	0.06	0.13	0.46	0.54
	D_w	0.32	0.44	0.02	0.06	0.36	0.47
Pol.Blogs	EPA	**0.38**	**0.47**	**0.76**	**0.58**	**0.54**	**0.53**
	D_s	0.26	0.43	0.18	0.52	0.32	0.42
	D_w	0.22	0.40	0.10	0.33	0.30	0.38

TABLE 9: Deception results on synthetic networks.

dataset	Alg	INF		Lou	
		Fitness	H	Fitness	H
1000-0.3	EPA	**0.26**	**0.51**	**0.34**	**0.68**
	D_s	0.14	0.42	0.14	0.60
	D_w	0.04	0.18	0.08	0.55
1000-0.5	EPA	**0.32**	**0.52**	**1.06**	**0.80**
	D_s	0.12	0.36	0.54	0.61
	D_w	0.10	0.32	0.52	0.61
3000-0.3	EPA	**0.16**	**0.41**	**0.26**	**0.72**
	D_s	0.10	0.39	0.10	0.68
	D_w	0.04	0.23	0.06	0.63
3000-0.5	EPA	**0.18**	**0.48**	**1.08**	**0.84**
	D_s	0.12	0.40	0.50	0.60
	D_w	0.10	0.39	0.50	0.60
5000-0.3	EPA	**0.06**	**0.27**	**0.08**	**0.51**
	D_s	0.04	0.24	**0.08**	0.48
	D_w	0.02	0.18	0.02	0.46
5000-0.5	EPA	**0.14**	**0.43**	**0.34**	**0.82**
	D_s	0.10	0.37	0.12	0.70
	D_w	0.04	0.26	0.08	0.67

to greater differences between communities, making it more difficult to hide the target community, i.e., corresponding to a smaller fitness value, according to Eq. (23).

3.4.3 Target Node Attack

Similarly, for target node attack, we also visualize the attack effect of our EPA on the Dolphin network using t-SNE algorithm, as shown in Fig. 6. Here, we focus on attacking hub and bridge nodes in a network since they are always
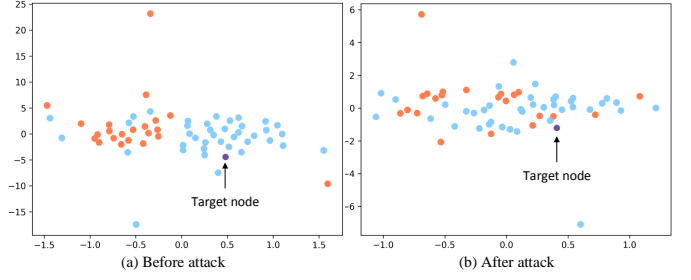

Fig. 6: The t-SNE visualization of target node attack obtained by EPA on Dolphin network. The target node originally belongs to the blue community. After the attack, it belongs to the red community. The communities are detected by GRE. (a) Before attack; (b) After attack.

TABLE 10: Target node attack on real-world networks.

Network	Node	GRE		INF		LOU	
		EPA	D_r	EPA	D_r	EPA	D_r
Football	T_1	**8.3%**	186%	**183%**	367%	**33%**	658%
	T_2	**9.1%**	155%	**27%**	182%	**18%**	145%
	T_3	**33%**	199%	**225%**	550%	**42%**	237%
Email	T_1	**8.4%**	15%	**25%**	35%	**0.3%**	0.87%
	T_2	**14%**	23%	**14%**	29%	**0.5%**	2.31%
	T_3	**18%**	32%	**32%**	61%	**5.2%**	12%
Pol.Blogs	T_1	**39%**	83%	**34%**	51%	**37%**	52%
	T_2	**38%**	75%	**16%**	33%	**17%**	27%
	T_3	**32%**	71%	**24%**	60%	**26%**	48%

more important in various applications. In particular, we first rank the nodes in each network based on their degree and betweenness, from large to small, respectively. Then, in addition to selecting the two nodes with the largest degree and betweenness, respectively, as the target nodes, we also sum the two orders for each node and choose the top one as our another target node. We name these three target nodes as T_1, T_2 and T_3, respectively. Finally, we compare EPA with the D_r algorithm that randomly adds links between target nodes and the nodes in different communities.

The percentages of degree increment in the attack for different strategies are presented in TABLE 10-TABLE 11. We can find that, in general, EPA performs significantly better than D_r, in terms of smaller percentage of degree increment, especially in the football network. Moreover, the bridge nodes of large betweenness are relatively easy to be attacked, since the neighbors of bridge nodes are always distributed in different communities, making them quite sensitive to link changes. On the contrary, it's relatively difficult to hide a node with both high degree and betweenness, since as hub nodes, a much large number of links always need to be added to change their communities. Note that, for real-world networks, based on EPA, the number of added links is much smaller than the degree of target nodes, indicating our method performs well in target node attack.

4 CONCLUSIONS

In this paper, we propose a novel Evolutionary Perturbation Attack (EPA) method, based on Genetic Algorithm (GA), to disturb community detection algorithms in three scales,

TABLE 11: Target node attack on synthetic networks.

dataset	Node	INF		Lou	
		EPA	D_r	EPA	D_r
1000-0.3	T_1	**67%**	336%	**53%**	406%
	T_2	**64%**	412%	**43%**	349%
	T_3	**70%**	409%	**55%**	445%
1000-0.5	T_1	**43%**	385%	**53%**	227%
	T_2	**46%**	320%	**22%**	58%
	T_3	**29%**	265%	**28%**	117%
3000-0.3	T_1	**142%**	340%	**40%**	625%
	T_2	**114%**	408%	**47%**	522%
	T_3	**194%**	493%	**54%**	596%
3000-0.5	T_1	**242%**	302%	**16%**	138%
	T_2	**60%**	279%	**8%**	19%
	T_3	**78%**	321%	**16%**	26%
5000-0.3	T_1	**126%**	625%	**264%**	1853%
	T_2	**97%**	535%	**224%**	1280%
	T_3	**338%**	691%	**416%**	2261%
5000-0.5	T_1	**60%**	560%	**55%**	706%
	T_2	**70%**	558%	**52%**	653%
	T_3	**128%**	672%	**83%**	815%

from local to global, by only changing a small fraction of links. In particular, we design a non-equal crossover strategy to treat the length of chromosome as a variable and naturally integrate it into GA; and we also integrate the network structural information, such as network distance and betweenness, into the algorithm to guide the search so as to find the better optimal attack strategy more quickly. Numerical experiments on six synthetic networks and three real-world validate the effectiveness and trasferability of our EPA method on attacking various community detection algorithms, i.e., by comparing with other attack methods of different scales, EPA behaves the best in most cases, achieving the state-of-the-art attack effects.

In the future, we will expand this work in the following three directions. First, we will propose new network coding methods and also utilize more network structural properties to improve the efficiency of EPA; second, we will try to integrate network embedding and deep learning graph models to improve the attack performance; third, we will do more experiments on more various networks, and further check the effectiveness of our EPA on the downstream algorithms based on community detection, to see the potential influence of EPA on many real applications.

ACKNOWLEDGMENTS

REFERENCES

[1] W. Chen and S.-H. Teng, "Interplay between social influence and network centrality: A comparative study on shapley centrality and single-node-influence centrality," in *Proceedings of the 26th International World Wide Web Conference on World Wide Web.* International World Wide Web Conferences Steering Committee, 2017, pp. 967–976.

[2] C. Ding, F. Xia, G. Gopalakrishnan, W. Qian, and A. Zhou, "Teamgen: An interactive team formation system based on professional social network," in *International Conference on World Wide Web Companion*, 2017, pp. 195–199.

[3] C. Fu, M. Zhao, L. Fan, X. Chen, J. Chen, Z. Wu, Y. Xia, and Q. Xuan, "Link weight prediction using supervised learning methods and its application to yelp layered network," *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[4] Q. Xuan, Z.-Y. Zhang, C. Fu, H.-X. Hu, and V. Filkov, "Social synchrony on complex networks," *IEEE transactions on cybernetics*, vol. 48, no. 5, pp. 1420–1431, 2018.

[5] K. An, Y.-C. Chiu, X. Hu, and X. Chen, "A network partitioning algorithmic approach for macroscopic fundamental diagrambased hierarchical traffic network management," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 4, pp. 1130–1139, 2018.

[6] J. Bao, T. He, S. Ruan, Y. Li, and Y. Zheng, "Planning bike lanes based on sharing-bikes' trajectories," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining.* ACM, 2017, pp. 1377–1386.

[7] Q. Xuan, F. Du, Y. Li, and T.-J. Wu, "A framework to model the topological structure of supply networks," *IEEE Transactions on Automation Science and Engineering*, vol. 8, no. 2, pp. 442–446, 2011.

[8] S. He, L. Huang, J. Shen, G. Gao, G. Wang, X. Chen, and L. Zhu, "Time synchronization network for east poloidal field power supply control system based on ieee 1588," *IEEE Transactions on Plasma Science*, 2018.

[9] F. D. Malliaros and M. Vazirgiannis, "Clustering and community detection in directed networks: A survey," *Physics Reports*, vol. 533, no. 4, pp. 95–142, 2013.

[10] M. R. Kamdar and M. A. Musen, "Phlegra: Graph analytics in pharmacology over the web of life sciences linked open data," in *Proceedings of the 26th International Conference on World Wide Web.* International World Wide Web Conferences Steering Committee, 2017, pp. 321–329.

[11] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E*, vol. 69, no. 2, p. 026113, 2004.

[12] M. Newman, "Community detection in networks: Modularity optimization and maximum likelihood are equivalent," *arXiv preprint arXiv:1606.02319*, 2016.

[13] M. E. Newman, "Fast algorithm for detecting community structure in networks," *Physical review E*, vol. 69, no. 6, p. 066133, 2004.

[14] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[15] A. Clauset, "Finding local community structure in networks," *Physical review E*, vol. 72, no. 2, p. 026132, 2005.

[16] P. Schuetz and A. Caflisch, "Multistep greedy algorithm identifies community structure in real-world and computer-generated networks," *Physical Review E*, vol. 78, no. 2, p. 026112, 2008.

[17] R. Guimera and L. A. N. Amaral, "Functional cartography of complex metabolic networks," *nature*, vol. 433, no. 7028, p. 895, 2005.

[18] D. Zhou and X. Wang, "A neighborhood-impact based community detection algorithm via discrete pso," *Mathematical Problems in Engineering*, vol. 4, pp. 1–15, 2016.

[19] F. Folino and C. Pizzuti, "An evolutionary multiobjective approach for community discovery in dynamic networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 8, pp. 1838–1852, 2014.

[20] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

[21] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proceedings of the National Academy of Sciences*, vol. 105, no. 4, pp. 1118–1123, 2008.

[22] P. Pons and M. Latapy, "Computing communities in large networks using random walks," in *International symposium on computer and information sciences.* Springer, 2005, pp. 284–293.

[23] M. Alamgir and U. Von Luxburg, "Multi-agent random walks for local clustering on graphs," in *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE, 2010, pp. 18–27.

[24] X. Fu, C. Wang, Z. Wang, and Z. Ming, "Threshold random walkers for community structure detection in complex networks," *Journal of Software*, vol. 8, no. 2, pp. 286–295, 2013.

[25] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical review E*, vol. 76, no. 3, p. 036106, 2007.

[26] M. J. Barber and J. W. Clark, "Detecting network communities by propagating labels under constraints," *Physical Review E*, vol. 80, no. 2, p. 026129, 2009.

[27] X. Liu and T. Murata, "Advanced modularity-specialized label propagation algorithm for detecting communities in networks," *Physica A: Statistical Mechanics and its Applications*, vol. 389, no. 7, pp. 1493–1500, 2010.

[28] S. Nagaraja, "The impact of unlinkability on adversarial community detection: effects and countermeasures," in *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, 2010, pp. 253–272.

[29] M. Waniek, T. P. Michalak, M. J. Wooldridge, and T. Rahwan, "Hiding individuals and communities in a social network," *Nature Human Behaviour*, vol. 2, no. 2, p. 139, 2018.

[30] V. Fionda and G. Pirro, "Community deception or: How to stop fearing community detection algorithms," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 4, pp. 660–673, 2018.

[31] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on neural networks for graph data," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 2847–2856.

[32] A. Bojchevski and S. Günnemann, "Adversarial attacks on node embeddings via graph poisoning," in *International Conference on Machine Learning*, 2019, pp. 695–704.

[33] J. Chen, Y. Wu, X. Xu, Y. Chen, H. Zheng, and Q. Xuan, "Fast gradient attack on network embedding," *arXiv preprint arXiv:1809.02797*, 2018.

[34] S. Yu, M. Zhao, C. Fu, J. Zheng, H. Huang, X. Shu, Q. Xuan, and G. Chen, "Target defense against link-prediction-based attacks via evolutionary perturbations," *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[35] J. Chen, L. Chen, Y. Chen, M. Zhao, S. Yu, Q. Xuan, and X. Yang, "Ga-based q-attack on community detection," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 3, pp. 491–503, 2019.

[36] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

[37] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical review E*, vol. 78, no. 4, p. 046110, 2008.

[38] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, p. 2, 2007.

[39] L. A. Adamic and N. Glance, "The political blogosphere and the 2004 us election: divided they blog," in *Proceedings of the 3rd international workshop on Link discovery*. ACM, 2005, pp. 36–43.

[40] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.