RobustECD: Enhancement of Network Structure for Robust Community Detection

Jiajun Zhou, Zhi Chen, Min Du, Lihong Chen, Shanqing Yu, Guanrong Chen, *Fellow, IEEE*, and Qi Xuan, *Member, IEEE*

Abstract—Community detection, which focuses on clustering vertex interactions, plays a significant role in network analysis. However, it also faces numerous challenges like missing data and adversarial attack. How to further improve the performance and robustness of community detection for real-world networks has raised great concerns. In this paper, we explore *robust community detection* by enhancing network structure, with two generic algorithms presented: one is named *robust community detection via genetic algorithm* (*RobustECD-GA*), in which the modularity and the number of clusters are combined in a fitness function to find the optimal structure enhancement scheme; the other is called *robust community detection via similarity ensemble* (*RobustECD-SE*), integrating multiple information of community structures captured by various vertex similarities, which scales well on large-scale networks. Comprehensive experiments on real-world networks demonstrate, by comparing with two traditional enhancement strategies, that the new methods help six representative community detection algorithms achieve more significant performance improvement. Moreover, experiments on the corresponding adversarial networks indicate that the new methods could also optimize the network structure to a certain extent, achieving stronger robustness against adversarial attack.

Index Terms—Community detection, genetic algorithm, vertex similarity, structure enhancement, adversarial attack

1 Introduction

Community detection, or network clustering, which aims to identify groups of interacting vertices in a network in term of their structural properties, has recently attracted considerable attention from different fields like sociology, biology and computer science [1], [2]. Community structure is of ultra importance in network analysis. Typically in networks, vertices are organized into groups, called *communities*, *clusters* or *modules*, with dense connections within groups and sparse connections between them. For instance, in co-author networks, communities are formed by scientists with similar research interests in close fields; in social networks like Facebook, they can represent people focusing on similar topics. Many recent researches suggest that network properties at the community level are quite different from those at the global level, and thus ignoring community structure may miss many interesting features [3]. In fact, identifying communities in networks has played a significant role in exploiting essential network structures.

- *Jiajun Zhou, Lihong Chen, Shanqing Yu, and Qi Xuan are with the Institute of Cyberspace Security, College of Information Engineering, Zhejiang University of Technology, Hangzhou, Zhejiang 310023, China. E-mail: {jjzhou, 2111803032, yushanqing, xuanqi}@zjut.edu.cn.*
- *Zhi Chen is with the Department of Computer Sciences, University of Illinois Urbana-Champaign, Urbana, IL 61801 USA. E-mail: zhic4@illinois.edu.*
- *Min Du is with Palo Alto Networks, Santa Clara, CA 95054 USA. E-mail: min.du.email@gmail.com.*
- *Guanrong Chen is with the Department of Electrical Engineering, City University of Hong Kong, Hong Kong, China. E-mail: eegchen@cityu.edu.hk.*

Manuscript received 24 Nov. 2019; revised 22 Apr. 2021; accepted 3 June 2021. Date of publication 14 June 2021; date of current version 7 Dec. 2022. (Corresponding author: Qi Xuan.) Recommended for acceptance by P. Tsaparas. Digital Object Identifier no. 10.1109/TKDE.2021.3088844

To date, a large number of techniques have been developed to detect community structures in networks. However, despite the advance of various community detection methods, from spectral method, label propagation to deep learning, their capability to discover the true community structure faces numerous challenges, since these approaches strongly rely on the topological structure of the underlying network, which is vulnerable in real-world scenarios.

First, missing data and adversarial noise seriously affect the performance of community detection algorithms. Real-world networks are often flawed in integrity and suffer from missing data, since not all real-world relationships are reflected in a single network. For instance, users in social networks like Twitter seldom follow all their friends in activities. Moreover, missing data also occurs when crawling datasets from online networks with privacy restrictions. On the other hand, the accuracy of a network is very likely to be questioned when the information encoded in the network topology is perturbed by artificial noise, especially when the network suffers from adversarial attacks, which leads to the degradation of the performance of many network analysis methods. In particular, adversarial attacks against community detection aim to hide target communities or sensitive edges [4], [5], and finally generate specific adversarial networks, which can strongly impact the performance of detection algorithms. Existing community detection methods rarely consider missing data and adversarial noise in networks, increasing the risk to obtain wrong community structures.

Another challenge is the lack of a consensus on the formal definition of a network community structure [6]. Currently, there are no universal standards for the definition of community, and a large number of detection algorithms based on different technologies and ideas have been proposed, which led to a quality discrepancy among different

results. Moreover, modularity optimization in community detection has a resolution limit [7]. Clusters consisting of a number of vertices smaller than a threshold would not be detected because these clusters tend to merge into larger ones by modularity optimization. Large, but locally sparse communities probably tend to be subdivided into smaller ones during community partition.

It is believed that such challenges are mostly from unstable network structures. Networks with sparse community structures are vulnerable to adversarial attacks which can destroy network structures, leading to community detection deception. Generally, communities with weak structures could be absorbed from the outside or disintegrated from the inside of the network. Enhancing the network structure and improving the robustness of the network could be an effective way to address these challenges. In this paper, we explore *robust community detection* by enhancing network structures, and develop two algorithms. A heuristic idea comes from the fact that community structures show a high connection density of intra-communities and a sparse one of inter-communities. Agglomerating the intra-communities by adding edges between internal vertices and dividing the inter-communities by removing edges between communities, therefore, can strengthen the community structure in a network. It's a natural reversal of the studies about community detection deception in [4], [5] where they are proposed to weaken community structures via intra-community edge deletion and inter-community edge addition. Another idea for robust community detection is to enhance network structure with edge prediction, of which the task is to complement missing edges or predict future edges between pairwise vertices based on the current network structure. The vertex similarity indices can be used to guide network structure optimization, according to the following two assumptions: 1) vertices in the same community are aggregated based on their high similarity; 2) a larger similarity of pairwise vertices leads to a higher likelihood of edges between them [8]. The main contributions of our work are summarized as follows:

- First, we study the ***Robust Community Detection*** via network structure **E**nhancement (*RobustECD*), which can improve the performance of existing detection algorithms. To the best of our knowledge, our work is the first for enhancing community detection in both real-world networks and adversarial networks.
- Second, we develop two generic enhancement algorithms, namely *robust community detection via genetic algorithm* (*RobustECD-GA*) and *robust community detection via similarity ensemble* (*RobustECD-SE*). Experimental results in six real-world networks demonstrate the superiority of our methods in helping six community detection algorithms to achieve significant improvement of performances.
- Third, we test our enhancement algorithms on four adversarial networks, the results show that both *RobustECD-GA* and *RobustECD-SE* can optimize the network structure to a certain extent, and achieve robust community detection against adversarial attack.
- Finally, our methods could alleviate the resolution limit in modularity optimization, and help various community detection algorithms to achieve consensus, i.e., getting more consistent partitions.

The rest of the paper is organized as follows. First, in Section 2, we review the related works. Then, in Section 3, we describe our approaches in detail. Thereafter, we present extensive experiments in Section 4, with a series of discussions. Finally, we conclude the paper and outline future work in Section 5.

2 Related Work

2.1 Community Detection

Community detection strives to identify groups of interacting vertices by maximizing cluster quality measures such as modularity [9] and normalized mutual information [10]. The literature [11] has provided comprehensive reviews on community detection. For community detection in undirected networks, widely used methods concentrate on agglomerative [12], divisive [3], [13], hierarchical [13], [14], spectral [15], [16], random walk [17], [18], label propagation [19], [20], high-order [21] and deep learning [22], [23] methods.

2.2 Traditional Enhancement of Community Detection

Due to the deficiency of many detection methods, how to improve their performance in complicated real applications has become an important issue. In this paper, we focus on the problem of enhancing existing community detection methods. Existing traditional enhancement approaches suggest preprocessing networks via weighting or rewiring. Meo *et al.* [24] introduced a measure of κ-path edge centrality and proposed a weighting algorithm called WERW-Kpath to effectively compute the centrality as edge weight, which is better for community detection. Sun [25] weighted networks via a series of edge centrality indices and detected communities in the weighted network using a function that considers both links and link weights. Lai *et al.* [26] considered random walk for simulation on dynamic processes, and applied it to enhance modularity optimization, based on the intuition that pairwise vertices in the same community have similar dynamic patterns. Interestingly, Li *et al.* [27] considered motif-aware community detection which achieves community partition using motif information in networks, and proposed an edge enhancement approach called Edmot. Their method transfers the network into motif-based hypergraph and partitions it into modules, and then a new edge set is constructed to enhance the connectivity structure of the original network by fully connecting all modules. Lancichinetti *et al.* [28] proposed consensus clustering algorithm, which combines the information of different outputs to obtain a more representative partition, to analyze the time evolution of clusters in dynamics networks. Dahlin *et al.* [29] proposed the ensemble cluster that combines the ensemble method with clustering, and improve community detection by aggregating multiple runs of algorithms. On the other hand, model-based methods tend to integrate the enhancement into the whole community detection procedure. For example, He *et al.* [30] provided a framework to enhance the ability of non-negative matrix factorization (NMF) models to detect communities, which uses the NMF method to train a stochastic model constrained by vertex similarity.

2.3 Adversarial Attack on Community Detection

In this paper, since some experiments are conducted on networks with adversarial noise which are generated via

TABLE 1
Main Notations Used in This Paper

Symbol	Description
$\mathcal{G}$	The target network
$\mathcal{V}, \mathcal{E}, \mathcal{M}$	Sets of vertices, edges, communities in $\mathcal{G}$
n, m	Numbers of vertices, edges in $\mathcal{G}$
v, e	Vertex, edge in $\mathcal{G}$
$\mathcal{S}$	The studied community detection algorithm
$\mathcal{M}_S, \phi_S$	Set/Number of communities found by S in $\mathcal{G}$
$\mathcal{M}_{real}, \phi_{real}$	Set/Number of the ground-truth communities in $\mathcal{G}$
$\mathcal{E}_{add}, \mathcal{E}_{del}$	The schemes of edge addition/deletion
β_a, β_d	budget of edge addition/deletion
$\mathcal{Q}$	Modularity
ϕ_p	Size of population
$\mathcal{T}_{ga}$	Number of iterations
$\mathcal{G}_{co}, \mathcal{A}_{co}$	Co-occurrence network and its adjacency matrix
$\mathcal{T}$	Threshold of prune in $\mathcal{G}_{co}$
$\mathcal{H}$	Similarity metric (or similarity matrix)
$\mathcal{G}_{co}^{\mathcal{T}}$	Co-occurrence graph pruned with threshold $\mathcal{T}$
$\mathcal{M}^{\mathcal{T}}$	Community partition in pruned graph $\mathcal{G}_{co}^{\mathcal{T}}$
c, C	Consensus of a cluster/partition

adversarial attack [31], we briefly review the research on adversarial attack for community detection. Waniek *et al.* [32] proposed a simple heuristic method deployed by intra-community edge deletion and inter-community edge addition, and introduced a measure of concealment to express how well a community is hidden. Fionda *et al.* [5] introduced and formalized the community deception problem, and proposed a community deception algorithm based on safeness, which achieves a success in hiding a target community. Chen *et al.* [4] proposed an effective evolutionary computation strategy, namely genetic algorithm (GA)-based $\mathcal{Q}$-Attack, to achieve deception by negligibly rewiring networks. Li *et al.* [33] proposed an end-to-end graph neural framework that combines graph generator and graph partitioner, and achieved the generation of adversarial examples of high quality and generalization.

3 Methodology

In this section, we first formulate the problem of community detection, and then present two enhancement strategies. The main notations used in this paper are listed in Table 1.

3.1 Problem Formulation

Assume that an undirected and unweighted network is represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, which consists of a vertex set $\mathcal{V} = \{v_i \,|\, i = 1, \ldots, n\}$ and an edge set $\mathcal{E} = \{e_i \,|\, i = 1, \ldots, m\}$. The topological structure of graph $\mathcal{G}$ is represented by an $n \times n$ adjacency matrix $\mathcal{A}$ with $\mathcal{A}_{ij} = 1$ if $(v_i, v_j) \in \mathcal{E}$ and $\mathcal{A}_{ij} = 0$ otherwise. The nonexistent edge set $\bar{\mathcal{E}}$ is represented by $\{(v_i, v_j) \,|\, \mathcal{A}_{ij} = 0; i \neq j\}$. The task of community detection in a network is to find a vertex partition $\mathcal{M} = \{\mathcal{M}_i \,|\, i = 1, \ldots, k\}$, with $\bigcup \mathcal{M}_i = \mathcal{V}$ and $\mathcal{M}_i \bigcap \mathcal{M}_j = \emptyset$ for $i \neq j$, where set $\mathcal{M}_i$ is called a *community*. The ground-truth community partition of the network is denoted as $\mathcal{M}_{real}$. Note that the community overlapping problem will not be considered in this paper.

We further explore the network structure enhancement from heuristic and optimized approaches. In the enhancement scenario, a network will be rewired via edge modification, during which the edges removed from network are sampled from the candidate set $\mathcal{E}_{del}^c$, while the edges added to the network are sampled from the candidate pairwise vertices set $\mathcal{E}_{add}^c$. The construction of candidate sets varies for different methods, as further discussed below.

For a network $\mathcal{G}$, one can get the set of edges added/removed from $\mathcal{G}$ via sampling from the candidate sets:

$$\begin{aligned} \mathcal{E}_{del} &= \{\tilde{e}_i \,|\, i = 1, \ldots, \lceil m \cdot \beta_d \rceil\} \subset \mathcal{E}_{del}^c \,, \\ \mathcal{E}_{add} &= \{\tilde{e}_i \,|\, i = 1, \ldots, \lceil m \cdot \beta_a \rceil\} \subset \mathcal{E}_{add}^c \,, \end{aligned} \tag{1}$$

where β_a, β_d are the budget of edge addition/deletion and $\lceil x \rceil = \textbf{ceil}(x)$. Then, based on the modification scheme $\mathcal{E}_{mod} = (\mathcal{E}_{add}, \mathcal{E}_{del})$, the connectivity structure of the original network is optimized to generate a rewired network:

$$\mathcal{G}^* = (\mathcal{V}, \mathcal{E}^*) \quad with\ \mathcal{E}^* = \mathcal{E} \cup \mathcal{E}_{add} \backslash \mathcal{E}_{del}. \tag{2}$$

For the rewired networks obtained via enhancement, we expect that the community detection methods perform significantly better and the new partition $\mathcal{M}^*$ is closer to the ground-truth communities, i.e., there is a significant improvement in evaluation metrics after assigning $\mathcal{M}^*$ to $\mathcal{G}$.

3.2 Modularity-Based Structure Enhancement

Previous works [4], [5], [32] have shown that intra-community edge deletion and inter-community edge addition can facilitate the deployment of community deception attacks. By contrast, it is natural that agglomerating the intra-communities by adding edges between internal vertices and dividing the inter-communities by removing edges between communities, i.e., intra-community edge addition and inter-community edge deletion, can strengthen the community structures in a network. Meanwhile, the resolution limitation problem that cannot be neglected requires the proposed approach to be capable of combining these four basic community edge modifications organically. Therefore, based on modularity, we propose the first method, named *robust community detection via genetic algorithm* (*RobustECD-GA*), which aims to enhance community structure via adaptable community edge rewiring. The schematic depiction of *RobustECD-GA* is shown in Fig. 1.

3.2.1 Network Rewiring

Given a network $\mathcal{G}$, a community edge optimization strategy requires knowledge of the community structure to pick the optimal edge modification schemes and thus depends on the prior community detection algorithm $\mathcal{S}$ that generates the estimated partition $\mathcal{M}_S = \{\mathcal{M}_i \,|\, i = 1, \ldots, k\}$. For arbitrary pairwise vertices (v_i, v_j), the candidate sets of four basic community edge modifications are represented as follows:

$$\begin{aligned} \mathcal{E}_{intra-add}^c &= \{(v_i, v_j) \,|\, v_i, v_j \in \mathcal{M}_i, \mathcal{A}_{ij} = 0\} \,, \\ \mathcal{E}_{intra-del}^c &= \{(v_i, v_j) \,|\, v_i, v_j \in \mathcal{M}_i, \mathcal{A}_{ij} = 1\} \,, \\ \mathcal{E}_{inter-add}^c &= \{(v_i, v_j) \,|\, v_i \in \mathcal{M}_i, v_j \in \mathcal{M}_j, \mathcal{A}_{ij} = 0\} \,, \\ \mathcal{E}_{inter-del}^c &= \{(v_i, v_j) \,|\, v_i \in \mathcal{M}_i, v_j \in \mathcal{M}_j, \mathcal{A}_{ij} = 1\} \,, \end{aligned} \tag{3}$$

where $\mathcal{M}_i, \mathcal{M}_j \in \mathcal{M}_S$, $\mathcal{E}_{intra-del}^c \cup \mathcal{E}_{inter-del}^c = \mathcal{E}$ and $\mathcal{E}_{intra-add}^c \cup \mathcal{E}_{inter-add}^c = \bar{\mathcal{E}}$.

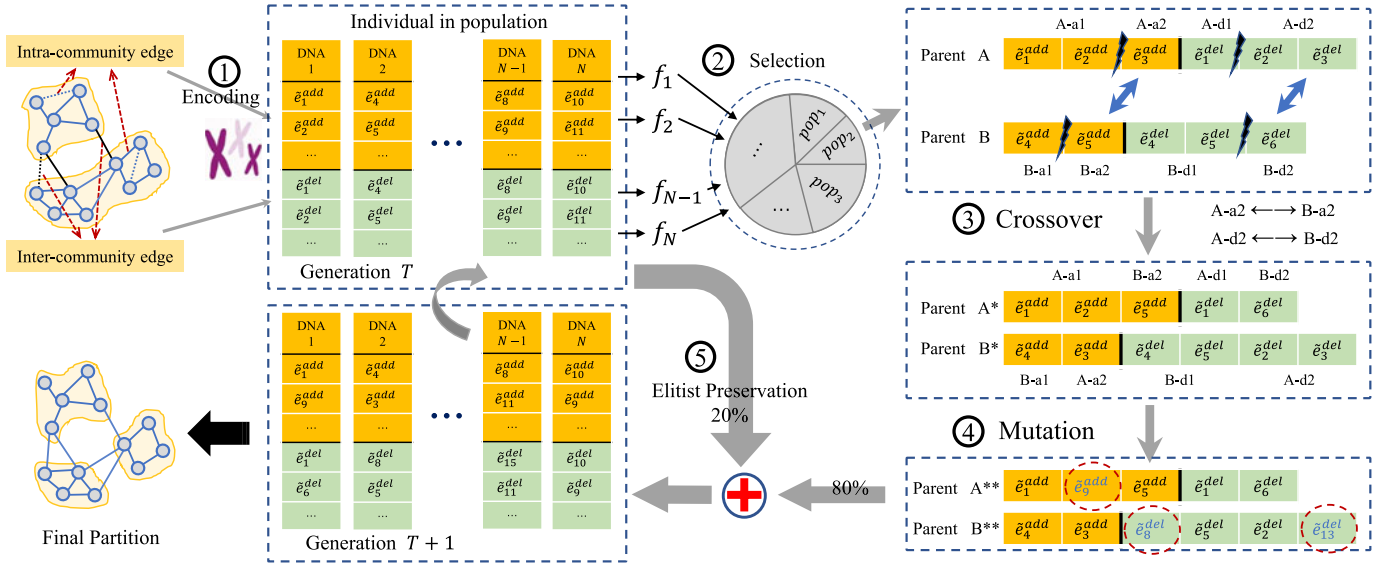


Fig. 1. Schematic depiction of *RobustECD-GA*. The workflow of evolution iteration proceeds as follows: 1) chromosome encoding for population initialization; 2) fitness calculation and individual selection; 3) chromosome crossover; 4) chromosome mutation; 5) elitist preservation.

The adaptable community edge rewiring consists of two parts, one is the required intra-community edge addition and inter-community edge deletion, and the other is an optional part that depends on the comparison between the number of communities in the estimated partition and the one in the ground truth:

$$\mathcal{E}_{mod} = \begin{cases} \left(\mathcal{E}_{add} \subset \bar{\mathcal{E}}, \mathcal{E}_{del} \subset \mathcal{E}^{c}_{inter-del}\right) & \phi_S > \phi_{real} \\ \left(\mathcal{E}_{add} \subset \mathcal{E}^{c}_{intra-add}, \mathcal{E}_{del} \subset \mathcal{E}\right) & \phi_S < \phi_{real} \\ \left(\mathcal{E}_{add} \subset \mathcal{E}^{c}_{intra-add}, \mathcal{E}_{del} \subset \mathcal{E}^{c}_{inter-del}\right) & \phi_S = \phi_{real} \end{cases} \tag{4}$$

where ϕ_{real} is the number of the ground-truth communities and ϕ_S is the number of communities in the estimated partition $\mathcal{M}_S$. Eq (4) lists three scenarios caused by resolution limit during initialization of edge modification:

- When $\phi_S > \phi_{real}$, there is a relatively high resolution, and large but locally sparse communities tend to be subdivided into smaller fragments. Ideally, extra inter-community edge addition is conducive to aggregate those fragments into integrate communities;
- When $\phi_S < \phi_{real}$, there is a relatively low resolution, and clusters consisting of a number of vertices smaller than a threshold tend to merge into larger ones. Ideally, extra intra-community edge deletion is conducive to subdivide large clusters;
- When $\phi_S = \phi_{real}$, there is a relatively suitable resolution. Both the inter-community edge addition and intra-community edge deletion are inoperative.

Notably, the aforementioned adaptable community edge rewiring requires knowledge of the ground-truth community (i.e., $\mathcal{M}_{real}$ and ϕ_{real}). When it comes to the dilemma that ground-truth communities are not available, the rewiring mechanism preserves only the required part, i.e., intra-community edge addition and inter-community edge deletion.

3.2.2 Evolutionary Optimization

We use the genetic algorithm (GA) due to its good performance in solving combinatorial optimization problems. Specifically, we design the encoding scheme of chromosome and the function of fitness as follows.

Algorithm 1. *RobustECD-GA*

Input: Target network $\mathcal{G}$, community detection algorithm $\mathcal{S}$, parameter for GA(ϕ_p, $\mathcal{P}_c$, $\mathcal{P}_m$, $\mathcal{P}_e$, $\mathcal{T}_{ga}$), budget β_a, β_d .
Output: New community partition $\mathcal{M}^*$

```
M_S ← detectCommunity(S, G);
P, F ← initializePop(G, M_S, φ_p, β_a, β_d);
Initialize current generation i = 0;
while i < T_ga do
    P_elitist ← retainElitist(F, P, P_e);
    P_select ← selection(F, P);
    P_crossover ← crossover(P_select, P_c);
    P_mutate ← mutation(P_crossover, P_m, M_S);
    F ← getFitness(G, S, P_mutate);
    P ← getNextGeneration(P_mutate, P_elitist)
Get the individual with highest fitness from the last population: E_mod ← getBestIndividual(F, P) ;
Rewire the original network to obtain G* via Eq. (2);
Feed G* into S to generate new community partition: M* ← detectCommunity(S, G*).;
end;
return M*;
```

- *Chromosome*. Chromosome represents an edge modification scheme $\mathcal{E}_{mod}$, consisting of two parts: $\mathcal{E}_{add}$ and $\mathcal{E}_{del}$, where a gene denotes an edge modification operation, including edge addition or deletion. The diagram of chromosome is shown in Fig. 2.
- *Fitness*. Modularity [9] has been widely applied in community detection task and is an effective evaluation metric to assess the quality of network partition (more details please refer to Section 4.2). Here, the fitness is defined as:

$$f = |\mathcal{Q}|/e^{|\phi_S - \phi_{real}|}\ , \tag{5}$$

where $\mathcal{Q}$ is the modularity of the partition for the target network and e is Euler's number. Denominator

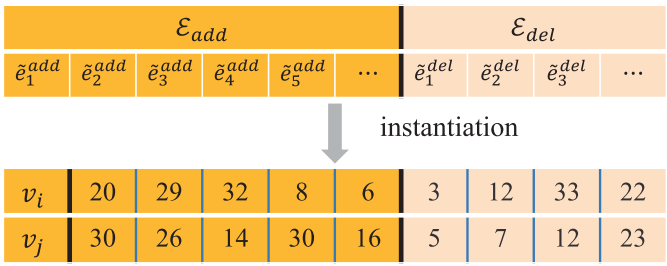


Fig. 2. The diagram of chromosome in *RobustECD-GA*. It consists of two parts including edge addition segment $\mathcal{E}_{add}$ and edge deletion segment $\mathcal{E}_{del}$. The instance of chromosome is initialized in the experiment for Karate dataset, with an edge addition segment of length 5 and an edge deletion segment of length 4.

$e^{|\phi_S - \phi_{real}|}$ is typically chosen to impose a penalty on the size of resolution. Modularity is divided by a large penalty term which is no less than e when $\phi_S \neq \phi_{real}$, and the fitness function degenerates to modularity when $\phi_S = \phi_{real}$ or no access to the ground-truth community. Individuals with larger modularity and more accurate partition generally have larger fitness.

The procedure of *RobustECD-GA* is shown in Algorithm 1. As mentioned above, *RobustECD-GA* requires the knowledge of the community structure, which guides the edge modification. We feed the target network $\mathcal{G}$ into community detection algorithm $\mathcal{S}$ to generate a general community partition $\mathcal{M}_S$ and then construct the candidate edge sets (line 1).

Algorithm 2. *RobustECD-SE*

Input: Target network $\mathcal{G}$, community detection algorithm $\mathcal{S}$, budget β_a.
Output: New community structure M^*
1 Compute similarity indices listed in Table 2: $\{\mathcal{H}_{\mathsf{CN}}, \ldots, \mathcal{H}_{\mathsf{RWR}}\} \leftarrow \mathsf{computeSimilarity}(\mathcal{G})$;
2 Obtain rewire schemes via sampling: $\{\mathcal{E}^1_{mod}, \ldots\} \leftarrow \mathsf{sample}(\mathcal{G}, \beta_a, \{\mathcal{H}_{\mathsf{CN}}, \ldots, \mathcal{H}_{\mathsf{RWR}}\})$;
3 Update graph via network rewiring : $\{\mathcal{G}^*_1, \ldots\} \leftarrow \mathsf{rewire}(\mathcal{G}, \{\mathcal{E}^1_{mod}, \ldots, \})$;
4 Obtain multiple partitions via community detection: $\{\mathcal{M}^*_1, \ldots\} \leftarrow \mathsf{detectCommunity}(\mathcal{S}, \{\mathcal{G}^*_1, \ldots\})$;
5 Get co-occurrence network from multiple partitions: $\mathcal{G}_{co}, \mathcal{A}_{co} \leftarrow \mathsf{getCoNetwork}(\{\mathcal{M}^*_1, \ldots\})$;
6 Threshold selection: $\mathcal{T}, \mathcal{M}^{\mathcal{T}}_{core} \leftarrow \mathsf{getOptimalThreshold}(\mathcal{G}_{co})$;
7 Get the final partition by assigning isolated vertices to core communities: $\mathcal{M}^* \leftarrow \mathsf{getFinalPartition}(\mathcal{M}^{\mathcal{T}}_{core}, \{\mathcal{H}_{\mathsf{CN}}, \ldots, \mathcal{H}_{\mathsf{RWR}}\})$;
8 **end**;
9 **return** $\mathcal{M}^*$;

During *initialization*, a parental generation $\mathcal{P} = \{\mathcal{E}^i_{mod} \,|\, i = 1, \ldots, \phi_p\}$ is randomly generated with a population size ϕ_p and each individual $\mathcal{E}^i_{mod}$ in the population has an unfixed size, i.e., the quantity of modified edges is not fixed for each initial modification scheme (line 2). During *selection*, the operation is conducted on *roulette*, which means that the probability for an individual to be selected is proportional to its fitness (line 6). *Crossover* is the process of combining the parental generation to generate new schemes and we apply *multi-point crossover* to swap gene segments between two parental chromosomes with a crossover rate $\mathcal{P}_c$ (line 7). *Mutation* prevents the algorithm from falling into local optimization. We traverse each gene in the chromosome and conduct the mutation operation with a mutation rate $\mathcal{P}_m$ (line 8). In so doing, we randomly replace the edge modification operation $\tilde{e}^{add}$ or $\tilde{e}^{del}$ with another one. Finally, *elitist preservation* is applied to retain excellent individuals, which refer to modification schemes with higher fitness. In particular, we retain excellent individuals by replacing the worst 20 percent of the offspring with the best 20 percent of the parents (line 5). Evolution is a process of iteration and we set the number of iterations $\mathcal{T}_{ga}$ as the evolutionary generation. The evolutionary optimization stops when it is convergent or this condition is satisfied.

3.3 Similarity-Based Structure Enhancement

Empirically, vertices in the same community is aggregated due to their high similarity. Vertex similarity can be defined as the number of common features that a pair of vertices share [42]. Previous works [8], [39] have shown that local, global and random-walk-based similarity indices perform excellently in capturing network structure features, and further unified them into three general forms of heuristics according to the subgraph involved in the similarity calculation, as summarized in Table 2 and Appendix D, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2021.3088844.[1] Therefore, we adopt the heuristics to aggregate those vertices of high similarity, i.e., considering the vertex similarity indices as the guidance of edge modification. Based on this, we propose the second method, named *robust community detection via similarity ensemble* (*RobustECD-SE*), which rewires a network via multiple similarity indices and aggregates corresponding community partitions to generate more accurate community structures. The schematic depiction of *RobustECD-SE* is shown in Fig. 3, and the procedure of *RobustECD-SE* is shown in Algorithm 2.

3.3.1 Network Rewiring

In *RobustECD-SE*, the similarity rewiring is a modification of network structure at the global level, so that the candidate set of edge modification is defined as $\mathcal{E}^c_{add} = \bar{\mathcal{E}}$. Note that only edge addition is considered in *RobustECD-SE* and edge deletion is neglected for several reasons: 1) slight improvement of performance; 2) extra time consumption, as further discussed in Appendix A, available in the online supplemental material.

Given a target network $\mathcal{G}$, the similarity matrix $\mathcal{H}$, which consists of similarity scores of arbitrary pairwise vertices, can be directly calculated (line 1). During *similarity rewiring*, we first assign all entries in $\mathcal{E}^c_{add}$ with relative weights that are associate with the vertex similarity scores. And we get the $\mathcal{E}_{add}$, the set of edges added to $\mathcal{G}$, via weighted random sampling from $\mathcal{E}^c_{add}$ with a budget of β_a, which means that the probability for an entry in $\mathcal{E}^c_{add}$ to be selected is proportional to its similarity score $\mathcal{H}_{ij}$ (line 2). After edge sampling, we update network via Eqs.(2) (line 3).

For each similarity index listed in Table 2, we conduct similarity rewiring for certain times, and finally obtain a series of rewired networks.

1. https://github.com/jjzhou012/RobustECD

TABLE 2
Summary of Similarity Indices Used in This Paper

Order	Category	Form	Used in paper
First	Local	$\mathcal{H}(i,j) = \lvert\Gamma(i) \cap \Gamma(j)\rvert \cdot g(i,j)$	Common Neighbors (CN) [8], Salton [34], Jaccard [35], Hub Promoted Index (HPI) [36]
Second	Local	$\mathcal{H}(i,j) = \sum_{z\in\Gamma(i)\cap(j)} g(z)$	Adamic–Adar Index (AA) [37], Resource Allocation Index (RA) [38]
High	Quasi-local, Global	$\mathcal{H}(i,j) = \eta \sum_{l=1}^{\infty} \gamma^l g(i,j,l)$ [39]	Local Path Index (LP) [40], Random Walk with Restart (RWR) [41]

$\mathcal{H}(i,j)$ is the similarity score of pairwise vertices (v_i, v_j), $\Gamma(i)$ is the 1-hop neighbors of v_i, $g(\cdot)$ is the nonnegative function under the given network, γ is a decaying factor between 0 and 1, η is a positive constant/function of γ.

3.3.2 Ensemble Optimization

After network rewiring, we feed all rewired networks into the *community detection* algorithm $\mathcal{S}$ to generate a series of community partitions (line 4). Due to the diversity of similarity indices and rewiring schemes, these partitions are likely to be non-unique and not necessarily better than the original partition $\mathcal{M}$. Ensemble learning, which achieves better classification or prediction performance by integrating multiple weak models, has been used for clustering tasks. Previous works on consensus and ensemble clustering [28], [29] have shown that these techniques can be combined with existing clustering methods and improve the stability and accuracy of community partitions.

During *partition ensemble*, we aggregate multiple partitions using a consensus matrix $\mathcal{A}_{co} = \{a_{ij}\}_{n\times n}$, in which element a_{ij} indicates the frequency of two vertices v_i and v_j assigned to the same community. A weighted co-occurrence network $\mathcal{G}_{co}$ can be generated by using $\mathcal{A}_{co}$ as the adjacency matrix (line 5). Once pairwise vertices appear in the same community in some partitions, $\mathcal{G}_{co}$ links them and assigns weights that correspond to the frequency of co-occurrence. A larger/smaller weight means a higher/lower likelihood that the pairwise vertices belong to the same community.

For the co-occurrence network $\mathcal{G}_{co}$, a natural idea is to consider those edges with larger/lower weights as intra/inter-community edges in the original network. Then, we can deploy community edge rewiring in $\mathcal{G}_{co}$ to optimize network structure. For inter-community edge deletion, we can prune $\mathcal{G}_{co}$ by setting a weight threshold $\mathcal{T}$. During *network pruning*, all edges with weights less than $\mathcal{T}$ are considered as inter-community edges and will be removed from $\mathcal{G}_{co}$ (line 6). We neglect intra-community edge addition since that the addition of new edges in $\mathcal{G}_{co}$ depends on the co-occurrence of pairwise vertices, but there is no access to more new partitions at this stage.

A visualization of network pruning in the co-occurrence network of Karate dataset is shown in Fig. 4. In this paper, we use eight similarity indices, and for each index, ten samplings are performed, to generate a total of eighty partitions, which determine the range of threshold $\mathcal{T} \in [0, 80]$. The original Karate network is shown in Fig. 4a, and there are two communities, with the vertices of the same color sharing the same ground-truth community label. Fig. 4b shows the co-occurrence network, which aggregates the information of eighty partitions and has dense connections. The last three subgraphs show the different pruned co-occurrence networks with various thresholds. When $\mathcal{T} = 20$, the pruned co-occurrence network still has only one connected component but two bridge vertices emerge, as shown in Fig. 4c. With the increase of the threshold, $\mathcal{G}_{co}$ is pruned to two connected components, matching exactly with the two clusters in the original network, as shown in Fig. 4d. When the threshold approaches the upper limit, generally, we'll get several small connected components that contain few vertices, or even isolated vertices, as shown in

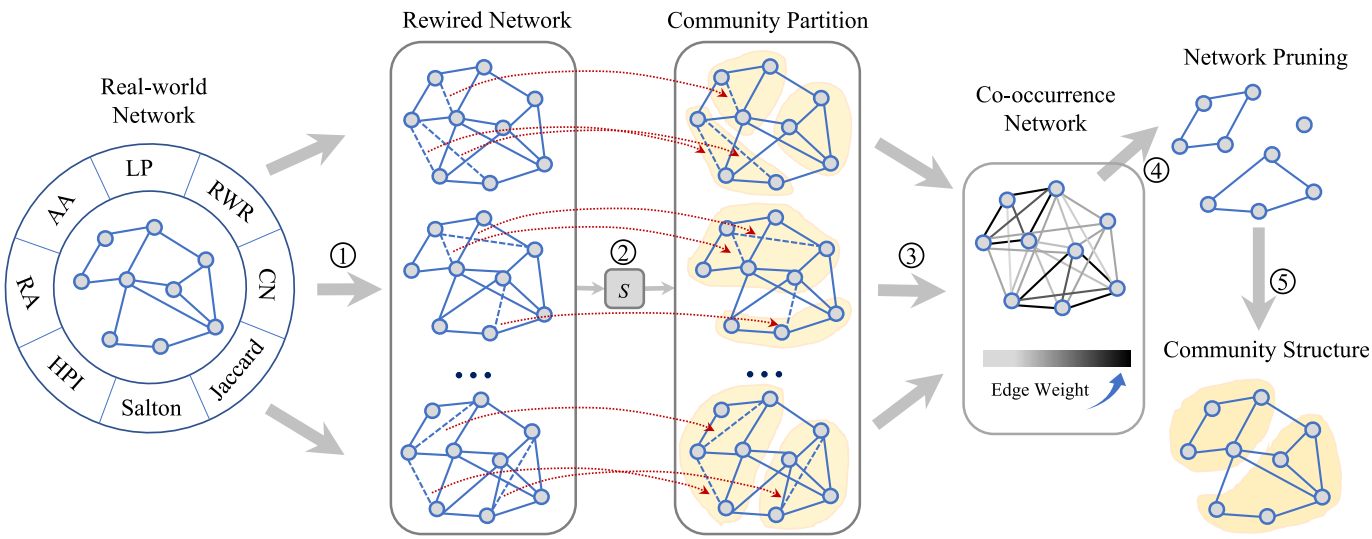


Fig. 3. Schematic depiction of *RobustECD-SE*. The complete workflow proceeds as follows: 1) similarity rewiring to generate rewired networks; 2) community detection to generate community partitions; 3) partition ensemble to generate co-occurrence network; 4) network pruning to identify core communities; 5) isolated vertices reassignment to get final community structure.

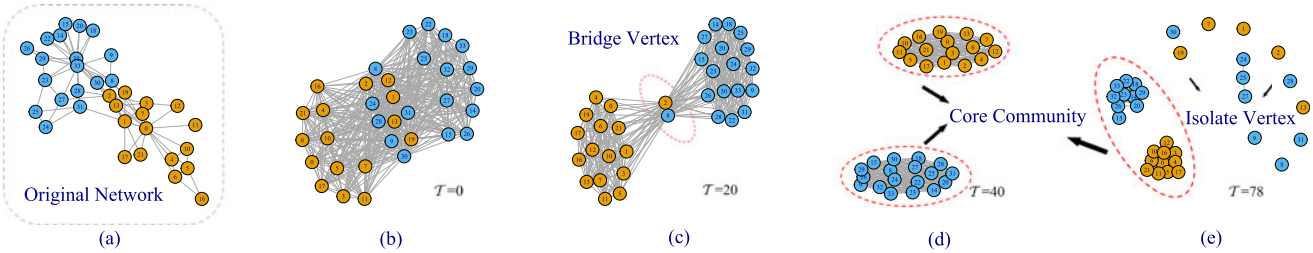


Fig. 4. Visualization of network pruning in the co-occurrence network of Karate dataset. The last four represent the co-occurrence network pruned with various values of threshold $\mathcal{T}$. Note that vertices with the same color share the same ground truth community label.

Fig. 4e. This phenomenon indicates that the selection of threshold actually influences the result of community partition, which is similar to the resolution limit problem in community detection.

After pruning, the co-occurrence network $\mathcal{G}_{co}$ is divided into several connected components, and those with large sizes will be treated as core communities $\mathcal{M}_{core}^{\mathcal{T}}$ (line 6). Generally, there exists several small connected components that contain few vertices, or even isolated vertices, when pruning $\mathcal{G}_{co}$ with a relatively large threshold. In order to get a final partition, these vertices in small connected components will be treated as isolated ones and assigned to the core community, to which it has the maximum average similarity (line 7). The ID of the core community that an isolated vertex v_i is assigned to, is defined as:

$$ID_{\mathcal{H}} = \arg\max_{k} \frac{1}{\phi_{core}^{k}} \sum_{j \in \mathcal{M}_{core}^{k}} \mathcal{H}(i,j) \ , \tag{6}$$

where $\mathcal{M}_{core}^{k}$ is the kth core community and ϕ_{core}^{k} is the number of vertices in $\mathcal{M}_{core}^{k}$. The final ID is determined by a *relative majority vote* of all similarity indices used in this paper:

$$ID = \text{realtiveMajorityVote}\{ID_{\mathcal{H}} \mid \mathcal{H} = \text{CN}, \ldots, \text{RWR}\} \ . \tag{7}$$

3.3.3 Threshold Selection

In order to address the resolution limit problem, we search the optimal threshold via a traversal procedure. Actually, the range of threshold $\mathcal{T} \in [0, 80]$ can be narrowed down to $\mathcal{T} \in \{a_{ij} \mid \forall a_{ij} \in \mathcal{A}_{co}\}$, which reduce the access times during traversal. We prune $\mathcal{G}_{co}$ with an accessed threshold $\mathcal{T}$ to yield a pruned network $\mathcal{G}_{co}^{\mathcal{T}}$, and evaluate the cluster partition of $\mathcal{G}_{co}^{\mathcal{T}}$ using *cluster consensus* metric, which can quantify the stability of clusters [47]. For a pruned co-occurrence network $\mathcal{G}_{co}^{\mathcal{T}}$ with cluster partition $\mathcal{M}^{\mathcal{T}} = \{\mathcal{M}_k \mid k = 1, \ldots, \phi^{\mathcal{T}}\}$, the consensus of cluster $\mathcal{M}_k$ is defined as

$$c(\mathcal{M}_k) = \frac{1}{\phi_k(\phi_k - 1)/2} \sum_{\substack{i,j \in \mathcal{M}_k \\ i<j}} a_{ij} \ , \tag{8}$$

where ϕ_k is the number of vertices in $\mathcal{M}_k$. The optimal threshold corresponds to that yield the $\mathcal{G}_{co}^{\mathcal{T}}$ with the maximum partition score, which is computed via a weighted sum of cluster consensus, as follows:

$$C(\mathcal{M}^{\mathcal{T}}) = \sum_{k=1}^{\phi^{\mathcal{T}}} \frac{\phi_k}{n} c(\mathcal{M}_k) \ , \tag{9}$$

$$\mathcal{T} = \arg\max_{\mathcal{T}} \ C(\mathcal{M}^{\mathcal{T}}) \ . \tag{10}$$

Note that Eq (10) is the heuristic definition of the optimal threshold, which can alleviate the resolution limit problem to a certain extent. Considering the complexity of the calculation during threshold selection, we can further simplify this process by the following approximation:

$$\mathcal{T}' = \arg\min_{\mathcal{T}} \ |\phi_{core}^{\mathcal{T}} - \phi_{real}| \ , \tag{11}$$

where $\phi_{core}^{\mathcal{T}}$ is the number of core communities yielded during network pruning. $\mathcal{T}'$ is the approximate optimal threshold, and we can obtain the core communities, of which the number is closest to that of the ground-truth. Note that this approximation depends on the knowledge of the ground-truth community.

4 Experiments

4.1 Datasets

We evaluate the proposed approaches against six real-world networks and four adversarial networks. For all networks, the ground-truth community labels are available. Table 3 provides an overview of the networks considered, including the number of communities found by each community detection method. Specifically, the six real-world networks consist of four small benchmark networks and two large-scale networks with missing data. The networks with missing data are subnetworks extracted from the Amazon product co-purchasing network and DBLP collaboration network [46], respectively. The four adversarial networks are generated via adversarial attack on four benchmark networks. Refer to Appendix B, available in the online supplemental material, for more details about datasets.

4.2 Evaluation Metrics

Benefiting from the availability of the ground-truth community labels, we evaluate the community partitions using supervised metrics like normalized mutual information [10] and adjusted rand index. Note that we design the fitness in *RobustECD-GA* using modularity $\mathcal{Q}$, thus it is not suitable as the evaluation metric.

- *Modularity ($\mathcal{Q}$)* [9]. Modularity is commonly used to measure the quality of community partition for a network with unknown community structure. The basic idea is to compare the network with the corresponding *null models*, which refer to random graph models that have some of the same properties as the

TABLE 3
Real-World Networks

Network	m	n	ϕ_{real}	Description	Number of communities (ϕ_S)					
					INF	FG	WT	LOU	LP	N2VKM
Karate	34	78	2	Zachary Karate club [43]	3	3	4	4	2	2
Polbooks	105	441	3	Books about US politics [44]	6	4	5	4	4	3
Football	115	613	12	American College football [13]	12	6	10	10	9	12
Polblogs	1490	19090	2	Political blogs [45]	306	277	416	276	272	2
Amazon-sub	10077	24205	3251	Amazon product co-purchasing [46]	651	97	509	69	638	\
DBLP-sub	26183	137529	5051	DBLP collaboration [46]	1143	120	4361	49	787	

ϕ_S is the number of communities found by the specific community detection method $\mathcal{S}$.

network but are completely random in other aspects. For a given network and a specific community partition $\mathcal{M}$, modularity is defined as:

$$\mathcal{Q} = \frac{1}{2m}\sum_{ij}\left(\mathcal{A}_{ij} - \frac{k_i k_j}{2m}\right)\delta(l_i, l_j)\ , \tag{12}$$

where m is the number of edges, $\mathcal{A}_{ij}$ is the element of adjacency matrix $\mathcal{A}$, k_i, k_j are the degree of vertices v_i, v_j, respectively. l_i, l_j are the community labels of vertices v_i, v_j in $\mathcal{M}$, respectively. $\delta(l_i, l_j) = 1$ if $l_i = l_j$ and $\delta(l_i, l_j) = 0$ otherwise.

- *Normalized Mutual Information (NMI)* [10]. NMI is a commonly used criterion to evaluate the similarity of two clustering results. It quantifies how much information the estimated partition contains in the real partition. For two clustering results X and Y, the NMI is defined as:

$$I_{\,\mathrm{norm}}\,(X, Y) = \frac{2I(X, Y)}{H(X) + H(Y)}, \tag{13}$$

where $I(X, Y) = H(Y) - H(X|Y)$ is the mutual information of X and Y, $H(Y)$ is the Shannon entropy of Y, and $H(X|Y)$ is the conditional entropy of X given Y.

- *Adjusted Rand Index (ARI)* [48]. ARI is the corrected-for-chance version of the Rand index (RI), which measures the degree of agreement between an estimated partition and a real partition. It is defined as

$$ARI = \frac{RI - E[RI]}{\max(RI) - E[RI]}. \tag{14}$$

Both NMI and ARI require the ground-truth community labels for evaluation purpose and the values are generally in the range between 0 to 1. For both metrics, a larger value indicates a better partition.

4.3 Community Detection Methods

We consider the following six community detection algorithms in our experiments.

- *Infomap (INF)* [17]. Infomap decomposes a network into modules by compressing the description of the information flow, i.e., it detects communities by minimizing the encoding length for a random walk.
- *Fast Greedy (FG)* [14]. This is a bottom-up hierarchical agglomeration algorithm. It merges individual vertices into communities based on a greedy modularity maximization strategy.
- *WalkTrap (WT)* [49]. It detects communities based on the idea that short random walks tend to stay in the same community.
- *Louvain (LOU)* [12]. This is a multi-level modularity optimization algorithm. It initializes each vertex with a separate community, and moves vertices between communities iteratively in a way that maximizes the vertices' local contributions to the overall modularity.
- *Label Propagation (LP)* [19]. This method detects communities by initializing each vertex with a unique label and re-assigning each vertex the dominant label in its neighbourhood in each iteration.
- *Node2vec + Kmeans (N2VKM)* [50]. This is a network embedding method, which learns lower-dimensional representations for vertices by biased random walk and skip-Gram. The K-means algorithm is then used to detect communities by clustering the embedded vectors of vertices in an Euclidean space.

4.4 Baseline Enhancement Methods

We compare our methods with the following two typical traditional enhancement methods, one is based on network rewiring and the other utilizes network weighting.

- *EdMot* [27]. It is an edge enhancement approach for motif-aware community detection via network rewiring and is proposed to address the hypergraph fragmentation issue. This strategy is recently proposed and achieves the state-of-the-art enhancement effect in several community detection methods.
- *WERW-KPath* [24]. It is an enhancement approach for community detection via network weighting. It exploits random walks to compute the κ-path edge centrality, which is then used to weight the edges. The strategy shows better interpretability and effectiveness among a series of weighted methods.

4.5 Experiment Setup

For all datasets, edges in networks are treated as undirected and self-loops will be removed. The main parameter settings for the proposed algorithms are shown in Table 4.

In *RobustECD-GA*, the general parameters of GA are set as empirical values. Note that the budget β_a and β_d controls

TABLE 4
Main Parameters Setting for the Proposed Algorithms

Method	Parameter	Value
RobustECD-GA	GA(ϕ_p, $\mathcal{P}_c$, $\mathcal{P}_m$, $\mathcal{P}_e$, $\mathcal{T}_{ga}$)	{120, 0.8, 0.02, 0.2, 1000}
	β_a	{0.01, 0.02, ..., 2.9}
	β_d	{0.01, 0.02, ..., 0.29}
RobustECD-SE	β_a	{0.1, 0.2, ..., 2.9}

the upper limit of the chromosome size during Initialization, i.e., each chromosome will be initialized with an unfixed size not larger than $\lceil m \cdot (\beta_a + \beta_d) \rceil$. We vary β_a and β_d in {0.01, 0.02, ..., 2.9} and {0.01, 0.02, ..., 0.29}, respectively. In *RobustECD-SE*, we vary β_a in {0.1, 0.2, ..., 2.9}.

In addition, for the community detection algorithm N2VKM, we take the number of clusters K in K-means the same as that of the ground-truth communities, and use the default setting for parameters in *Node2vec*. Specifically, the walk length is 80, the number of walks per node is 10, the embedding dimension is 128, and both return hyper parameter p and in-out parameter q are equal to 1. We repeat all experiments for 50 times and report the average metrics and their standard deviations of community detection.

4.6 Evaluation

We evaluate the benefit of the proposed enhancement strategies, answering the following research questions:

- *RQ1*: Can *RobustECD* improve the performance of community detection in real-world networks combined with existing community detection algorithms?
- *RQ2*: Does *RobustECD* still work when it comes to adversarial networks?
- *RQ3*: How does the selection of various similarity indices in *RobustECD-SE* affect the performance?
- *RQ4*: How does *RobustECD* achieve interpretable enhancement of community detection?

4.6.1 Enhancement in Real Network

Table 5 reports the results of the enhancement for six community detection algorithms, from which one can observe that there is a significant boost in detection performance across all six real-world networks. First, compared with those traditional enhancement algorithms, these detection algorithms combined with the proposed *RobustECD* framework obtain much better results in most cases. The *RobustECD-GA* and *RobustECD-SE* achieves 87.50 and 88.24 percent success rate on enhancing community detection. The success rate refers to the percentage of enhanced results which are better than the original results in term of both NMI and ARI. These phenomenon provide a positive answer to **RQ1**, indicating that *RobustECD* can improve the performance of existing community detection algorithms and alleviate the problems of resolution limit and missing data. Meanwhile, the results in Amazon and DBLP sub-networks also indicates the effectiveness of the *RobustECD-SE* in large-scale networks.

Second, we define the relative improvement rate (RIMP) for each metric as follows:

$$RIMP = \begin{cases} (Met_{en} - Met_{ori})/Met_{ori} & Met_{ori} > 0 \\ Met_{en} - Met_{ori} & Met_{ori} = 0 \end{cases}, \tag{15}$$

where Met_{ori} and Met_{en} refer to the metric of the original and the enhanced results, respectively. Note that we also consider the extreme case that the original metrics may go down to 0 in the adversarial networks. In Table 5, the far-right column of each metrics gives the average relative improvement rate (Avg RIMP) in metric, from which one can see that *RobustECD-GA* and *RobustECD-SE* achieve competitive performance, and significantly outperform baselines.

Third, considered detection algorithms have different performance on the real datasets, but generally obtain more similar community partitions during robust enhancement. For instance, we use standard deviation to measure the consistency of results of different detection algorithms. For *RobustECD-SE*, the standard deviations of the original ARI for the six detection algorithms on the six networks are (0.118, 0.059, 0.153, 0.046, 0.030, 0.026), while those of the corresponding enhanced ARI are (0.067, 0.018, 0.039, 0.029, 0.023, 0.025). The decrease in standard deviation indicates that *RobustECD* can stable network structure and achieve the consistency of detection performance. Moreover, *RobustECD* achieves perfect enhancement on some community detection algorithms applied to small datasets. For instance, when enhancing FG and LOU via *RobustECD-SE* on Karate dataset, both NMI and ARI are equal to 1, suggesting that FG and LOU algorithms can detect community structures completely correctly after enhancement.

4.6.2 Enhancement in Adversarial Network

Adversarial attack aims to degrade the performance of algorithms by perturbing the network structure or attacking the computational process. In social networks, the adversarial attack on community detection or link prediction probably facilitates to hide the real community structure or sensitive links. Table 6 reports the results of enhancing community detection in adversarial networks, which are generated by slightly modifying the networks structures via certain adversarial attacks. Note that here we report the community detection results in original networks and adversarial networks as references. As we can see, compared with the original results, the performance metrics display a significant decrease in adversarial networks, indicating that adversarial attack has indeed broken the network structure and achieved a community detection deception. Then during structure enhancement, our algorithms significantly outperform the baselines in all adversarial networks. In fact, both *RobustECD-GA* and *RobustECD-SE* help the six community detection algorithms achieve huge improvements on detection performances, which is even better than the results in original networks. However, the baselines Edmot and WERW-KPath have mediocre performance and may fail with the increase of the attack strength. Such results indicate that our enhancement algorithms could not only help partially or even fully recover the network structures destroyed by adversarial attacks, but also improving the robustness of existing community detection algorithms against adversarial noise, positively answering **RQ2**.

TABLE 5
Community Detection Results in the Real Networks

Dataset	Method	Community Detection													
		NMI							ARI						
		INF	FG	WT	LOU	LP	N2VKM	Avg RIMP	INF	FG	WT	LOU	LP	N2VKM	Avg RIMP
Karate	original	0.699±0.000	0.598±0.000	0.600±0.000	0.587±0.000	0.689±0.283	0.705±0.175	—	0.702±0.000	0.491±0.000	0.513±0.000	0.462±0.000	0.687±0.320	0.716±0.212	—
	WERW-Kpath	0.618±0.071	0.607±0.040	0.528±0.059	0.518±0.043	0.622±0.185	0.644±0.186	-8.70%	0.557±0.124	0.593±0.046	0.398±0.107	0.407±0.042	0.604±0.229	0.652±0.227	-9.20%
	Edmot	0.699±0.000	0.598±0.000	0.600±0.000	0.587±0.000	0.685±0.203	0.713±0.177	0.09%	0.702±0.000	0.491±0.000	0.513±0.000	0.462±0.000	0.686±0.237	0.728±0.200	0.26%
	RobustECD-GA	**0.912±0.176**	0.878±0.071	**0.984±0.049**	0.867±0.090	0.705±0.180	**0.838±0.019**	35.03%	**0.923±0.165**	0.912±0.051	**0.988±0.035**	0.898±0.083	0.714±0.217	**0.872±0.024**	54.98%
	RobustECD-SE	0.825±0.220	**1.000±0.000**	0.821±0.088	**1.000±0.000**	**0.847±0.136**	0.834±0.114	**38.95%**	0.797±0.240	**1.000±0.000**	0.852±0.086	**1.000±0.000**	**0.871±0.124**	0.869±0.100	**57.98%**
Polbooks	original	0.493±0.000	0.531±0.000	0.559±0.000	0.512±0.000	0.554±0.025	0.556±0.017	—	0.536±0.000	0.638±0.000	0.681±0.000	0.558±0.000	0.647±0.041	0.662±0.009	—
	WERW-Kpath	0.462±0.002	0.546±0.020	0.531±0.031	0.509±0.020	0.552±0.034	0.563±0.016	-1.36%	0.435±0.000	0.658±0.026	0.591±0.069	0.571±0.034	0.654±0.054	0.660±0.015	-4.30%
	Edmot	0.493±0.000	0.531±0.000	0.559±0.000	0.512±0.000	0.561±0.026	0.564±0.016	0.45%	0.536±0.000	0.638±0.000	0.681±0.000	0.558±0.000	0.661±0.035	0.667±0.018	0.49%
	RobustECD-GA	0.526±0.121	0.554±0.000	0.554±0.000	0.554±0.000	0.554±0.014	0.589±0.017	4.04%	0.621±0.143	**0.652±0.000**	0.652±0.000	0.652±0.000	**0.670±0.007**	**0.684±0.013**	6.25%
	RobustECD-SE	**0.574±0.014**	**0.569±0.001**	**0.586±0.017**	**0.560±0.011**	**0.598±0.009**	**0.589±0.009**	**8.61%**	**0.677±0.014**	0.636±0.000	**0.687±0.015**	**0.669±0.006**	0.665±0.010	0.677±0.011	**8.64%**
Football	original	0.924±0.000	0.698±0.000	0.887±0.000	0.890±0.000	0.888±0.037	0.912±0.012	—	0.897±0.000	0.474±0.000	0.815±0.000	0.807±0.000	0.784±0.103	0.872±0.025	—
	WERW-Kpath	0.924±0.000	0.698±0.000	0.887±0.000	0.890±0.000	0.885±0.030	0.915±0.011	0.00%	0.897±0.000	0.474±0.000	0.815±0.000	0.807±0.000	0.779±0.083	0.876±0.020	-0.03%
	Edmot	0.924±0.000	0.698±0.000	0.887±0.000	0.890±0.000	0.885±0.030	0.915±0.011	0.00%	0.897±0.000	0.474±0.000	0.815±0.000	0.807±0.000	0.779±0.083	0.876±0.020	-0.03%
	RobustECD-GA	**0.927±0.000**	0.862±0.018	**0.927±0.000**	**0.909±0.000**	0.896±0.023	**0.927±0.000**	**5.50%**	0.889±0.000	0.746±0.042	**0.889±0.000**	0.847±0.000	0.793±0.082	**0.889±0.000**	12.27%
	RobustECD-SE	0.924±0.000	**0.877±0.021**	0.923±0.009	0.906±0.014	**0.915±0.018**	0.898±0.021	**5.50%**	**0.897±0.000**	**0.785±0.061**	0.881±0.018	**0.849±0.029**	**0.869±0.032**	0.849±0.029	**14.52%**
Polblogs	original	0.330±0.001	0.378±0.000	0.318±0.000	0.376±0.000	0.375±0.053	0.458±0.067	—	0.439±0.001	0.528±0.000	0.419±0.000	0.521±0.000	0.515±0.105	0.489±0.053	—
	WERW-Kpath	0.329±0.001	0.376±0.002	0.316±0.001	0.370±0.001	0.375±0.037	0.453±0.025	-0.69%	0.437±0.003	0.525±0.003	0.414±0.003	0.515±0.002	0.518±0.074	0.480±0.040	-0.77%
	Edmot	0.329±0.000	0.378±0.000	0.318±0.000	0.376±0.000	0.376±0.053	0.457±0.031	-0.04%	0.437±0.000	0.528±0.000	0.419±0.000	0.521±0.000	0.517±0.105	0.486±0.049	-0.11%
	RobustECD-GA	0.453±0.000	0.525±0.000	0.504±0.000	0.529±0.000	0.519±0.005	0.381±0.002	32.82%	0.472±0.000	0.618±0.000	0.599±0.000	0.622±0.000	0.616±0.002	0.556±0.001	20.04%
	RobustECD-SE	**0.517±0.007**	**0.551±0.006**	**0.556±0.009**	**0.551±0.005**	**0.529±0.007**	**0.499±0.006**	**45.64%**	**0.619±0.005**	**0.642±0.004**	**0.643±0.006**	**0.644±0.004**	**0.628±0.005**	**0.569±0.005**	**29.66%**
Amazon -sub	original	0.775±0.000	0.592±0.000	0.703±0.000	0.607±0.000	0.760±0.001	—	—	0.110±0.000	0.034±0.000	0.048±0.000	0.045±0.000	0.069±0.004	—	—
	WERW-Kpath	0.777±0.000	0.632±0.000	**0.748±0.000**	0.633±0.000	**0.770±0.000**	—	3.80%	**0.112±0.000**	0.048±0.000	0.052±0.000	0.052±0.000	**0.076±0.000**	—	15.91%
	Edmot	0.775±0.000	0.593±0.000	0.703±0.000	0.607±0.000	0.761±0.000	—	0.06%	0.110±0.000	0.034±0.000	0.048±0.000	0.045±0.000	0.073±0.000	—	1.28%
	RobustECD-SE	**0.779±0.000**	**0.663±0.009**	0.732±0.005	**0.657±0.004**	0.768±0.001	—	**5.18%**	0.107±0.000	**0.053±0.004**	**0.058±0.008**	**0.053±0.000**	0.067±0.001	—	**18.72%**
DBLP -sub	original	0.698±0.000	0.348±0.000	0.683±0.000	0.431±0.000	0.436±0.000	—	—	0.061±0.000	0.004±0.000	0.004±0.000	0.010±0.000	0.000±0.000	—	—
	WERW-Kpath	0.701±0.000	0.350±0.000	**0.688±0.000**	0.438±0.000	**0.649±0.000**	—	10.44%	0.062±0.000	0.004±0.000	0.005±0.000	0.011±0.000	**0.008±0.000**	—	6.43%
	Edmot	0.699±0.000	0.354±0.000	0.683±0.000	0.431±0.000	0.446±0.000	—	0.83%	0.060±0.000	0.003±0.000	0.004±0.000	0.010±0.000	0.000±0.000	—	-2.60%
	RobustECD-SE	**0.704±0.000**	**0.582±0.000**	0.684±0.000	**0.598±0.000**	0.638±0.000	—	**30.66%**	**0.064±0.000**	**0.007±0.000**	**0.011±0.000**	**0.019±0.000**	0.003±0.000	—	**72.08%**

4.6.3 *Impact of Similarity in RobustECD-SE*

Thanks to the outstanding performance of *RobustECD*, we further investigate the impact of the similarity indices in the *RobustECD-SE*. Fig. 5 shows the results of all similarity indices (*RobustECD-SE(all)*) and single index (*RobustECD-SE (single)*), respectively (see the Appendix E, available in the

TABLE 6
Community Detection Results in the Adversarial Networks

Dataset	Method	Community Detection													
		NMI							ARI						
		INF	FG	WT	LOU	LP	N2VKM	Avg RIMP	INF	FG	WT	LOU	LP	N2VKM	Avg RIMP
Karate (noise)	original	0.699±0.000	0.598±0.000	0.600±0.000	0.587±0.000	0.689±0.283	0.705±0.175	63.90%	0.702±0.000	0.491±0.000	0.513±0.000	0.462±0.000	0.687±0.320	0.716±0.212	70.66%
	Attack	0.000±0.000	0.447±0.000	0.487±0.000	0.250±0.000	0.475±0.337	0.399±0.231	—	0.000±0.000	0.361±0.000	0.330±0.000	0.180±0.000	0.498±0.354	0.427±0.261	—
	WERW-Kpath	0.173±0.093	0.296±0.050	0.444±0.082	0.341±0.060	0.373±0.211	0.384±0.198	-2.36%	0.116±0.114	0.258±0.044	0.359±0.133	0.281±0.043	0.357±0.255	0.401±0.238	2.26%
	Edmot	0.001±0.000	0.447±0.000	0.479±0.000	0.250±0.000	0.418±0.345	0.427±0.225	-1.09%	0.000±0.000	0.361±0.000	0.525±0.000	0.180±0.000	0.446±0.367	0.448±0.257	8.93%
	RobustECD-GA	**0.720±0.132**	0.576±0.000	**0.670±0.119**	0.352±0.128	0.371±0.327	**0.836±0.183**	44.48%	**0.768±0.122**	0.668±0.000	**0.743±0.096**	0.367±0.131	0.393±0.350	**0.882±0.152**	**79.39%**
	RobustECD-SE	0.425±0.336	**0.709±0.218**	0.576±0.075	**0.484±0.173**	**0.828±0.197**	0.529±0.340	**53.31%**	0.427±0.373	**0.720±0.252**	0.593±0.102	**0.448±0.208**	**0.857±0.198**	0.552±0.352	78.68%
Polbooks (noise)	original	0.493±0.000	0.531±0.000	0.559±0.000	0.512±0.000	0.554±0.025	0.556±0.017	26.69%	0.536±0.000	0.638±0.000	0.681±0.000	0.558±0.000	0.647±0.041	0.662±0.009	47.95%
	Attack	0.418±0.004	0.482±0.000	0.393±0.000	0.343±0.000	0.461±0.030	0.462±0.012	—	0.459±0.010	0.530±0.000	0.351±0.000	0.252±0.000	0.534±0.053	0.581±0.011	—
	WERW-Kpath	0.485±0.006	0.560±0.019	0.503±0.013	0.481±0.010	0.552±0.032	0.565±0.020	23.74%	0.505±0.013	0.641±0.028	0.541±0.041	0.552±0.015	0.649±0.054	0.660±0.015	39.88%
	Edmot	0.490±0.000	0.482±0.000	0.494±0.000	0.367±0.000	0.566±0.031	0.571±0.018	16.05%	0.533±0.000	0.530±0.000	0.539±0.000	0.344±0.000	0.653±0.051	0.666±0.013	23.85%
	RobustECD-GA	0.559±0.000	0.559±0.000	0.559±0.000	0.559±0.000	0.585±0.017	0.592±0.022	34.99%	0.646±0.000	**0.646±0.000**	0.646±0.000	0.646±0.000	**0.692±0.011**	0.658±0.015	57.64%
	RobustECD-SE	**0.590±0.014**	**0.565±0.012**	**0.599±0.008**	**0.564±0.010**	**0.599±0.008**	**0.643±0.026**	**40.72%**	**0.656±0.015**	0.628±0.013	**0.691±0.012**	**0.672±0.007**	0.665±0.009	**0.729±0.018**	**62.49%**
Football (noise)	original	0.924±0.000	0.698±0.000	0.887±0.000	0.890±0.000	0.888±0.037	0.912±0.012	11.27%	0.897±0.000	0.474±0.000	0.815±0.000	0.807±0.000	0.784±0.103	0.872±0.025	52.52%
	Attack	0.809±0.000	0.658±0.000	0.809±0.000	0.755±0.000	0.800±0.051	0.838±0.027	—	0.498±0.000	0.375±0.000	0.498±0.000	0.481±0.000	0.503±0.142	0.719±0.055	—
	WERW-Kpath	0.795±0.003	0.643±0.017	0.878±0.022	0.812±0.025	0.765±0.059	0.841±0.026	1.34%	0.486±0.001	0.378±0.028	0.770±0.066	0.623±0.055	0.449±0.143	0.725±0.057	12.10%
	Edmot	0.809±0.000	0.658±0.000	0.809±0.000	0.755±0.000	0.781±0.051	0.835±0.027	-0.46%	0.498±0.000	0.375±0.000	0.498±0.000	0.481±0.000	0.444±0.116	0.714±0.061	-2.07%
	RobustECD-GA	**0.927±0.000**	**0.791±0.000**	**0.927±0.000**	**0.909±0.000**	0.797±0.071	**0.870±0.015**	**12.20%**	**0.889±0.000**	**0.597±0.000**	**0.889±0.000**	**0.847±0.000**	0.430±0.211	0.753±0.049	**47.09%**
	RobustECD-SE	0.809±0.000	0.762±0.024	0.909±0.020	0.886±0.014	**0.863±0.051**	0.862±0.021	9.38%	0.498±0.000	0.488±0.056	0.843±0.063	0.793±0.033	**0.680±0.189**	**0.769±0.043**	34.40%
Polblogs (noise)	original	0.330±0.001	0.378±0.000	0.318±0.000	0.376±0.000	0.375±0.053	0.458±0.067	8.60%	0.439±0.001	0.528±0.000	0.419±0.000	0.521±0.000	0.515±0.105	0.489±0.053	9.42%
	Attack	0.303±0.001	0.348±0.000	0.299±0.000	0.336±0.000	0.340±0.061	0.434±0.034	—	0.404±0.002	0.490±0.000	0.392±0.000	0.465±0.000	0.453±0.122	0.455±0.017	—
	WERW-Kpath	0.302±0.007	0.348±0.012	0.303±0.004	0.318±0.003	0.375±0.055	0.435±0.036	1.03%	0.401±0.005	0.490±0.017	0.398±0.008	0.359±0.007	0.520±0.118	0.459±0.020	-1.02%
	Edmot	0.304±0.000	0.348±0.000	0.299±0.000	0.336±0.000	0.360±0.050	0.431±0.037	0.92%	0.404±0.000	0.490±0.000	0.392±0.000	0.465±0.000	0.494±0.101	0.454±0.018	1.47%
	RobustECD-GA	**0.473±0.000**	**0.527±0.000**	0.428±0.000	**0.532±0.000**	**0.530±0.023**	0.377±0.030	41.96%	0.535±0.000	**0.620±0.000**	0.599±0.000	**0.625±0.000**	**0.625±0.031**	0.345±0.023	26.66%
	RobustECD-SE	0.444±0.007	0.505±0.007	**0.558±0.005**	0.493±0.004	0.500±0.005	**0.469±0.023**	**46.69%**	**0.562±0.005**	0.604±0.004	**0.637±0.004**	0.601±0.003	0.602±0.004	**0.548±0.011**	**34.58%**

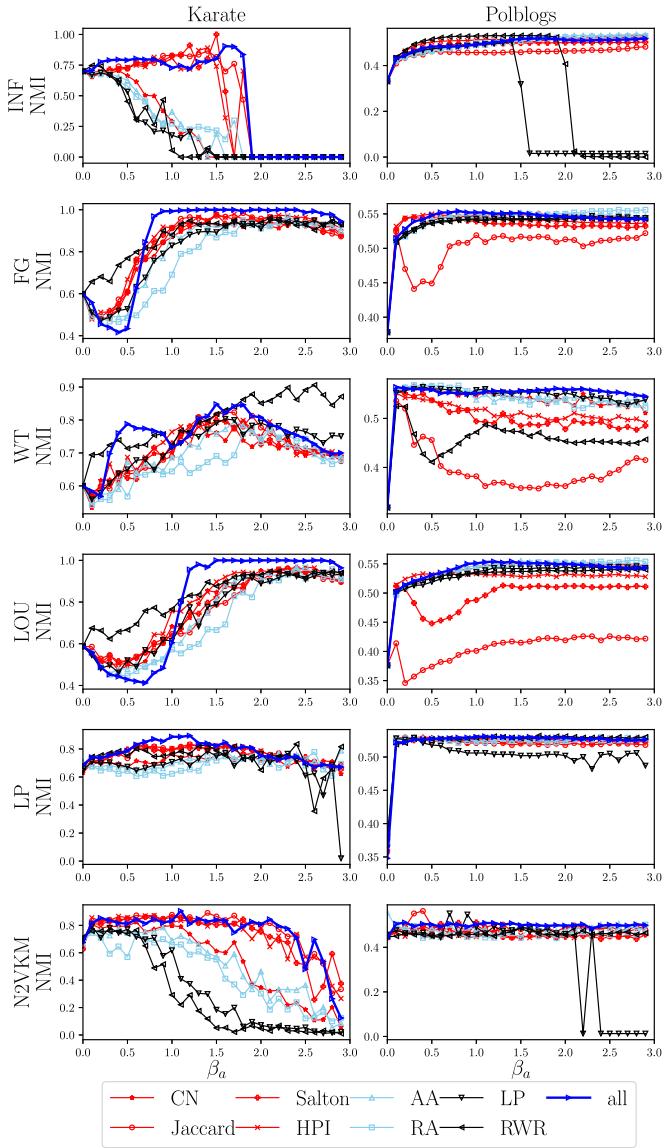


Fig. 5. The impact of similarity metrics on the performance of *RobustECD-SE* in term of NMI.

online supplemental material and Appendix F, available in the online supplemental material, for more details about the impact of similarity indices). We first summarized three impact effects:

- *Complementary*: *RobustECD-SE(all)* outperforms all *RobustECD-SE(single)*s, indicating that these single similarity indices are complementary.
- *Redundant*: *RobustECD-SE(single)*s with some indices achieve competitive performance against *RobustECD-SE(all)*, indicating that the other indices in *RobustECD-SE(single)*s that have relative poor performance are redundant.
- *Negative*: *RobustECD-SE(single)*s with some indices outperform *RobustECD-SE(all)*, indicating that the other indices in *RobustECD-SE(single)*s that have relative poor performance are negative.

From the comparison results, we observe that *RobustECD-SE(all)* generally outperforms *RobustECD-SE(single)*, and receives more stable results in most cases. And the impact of single

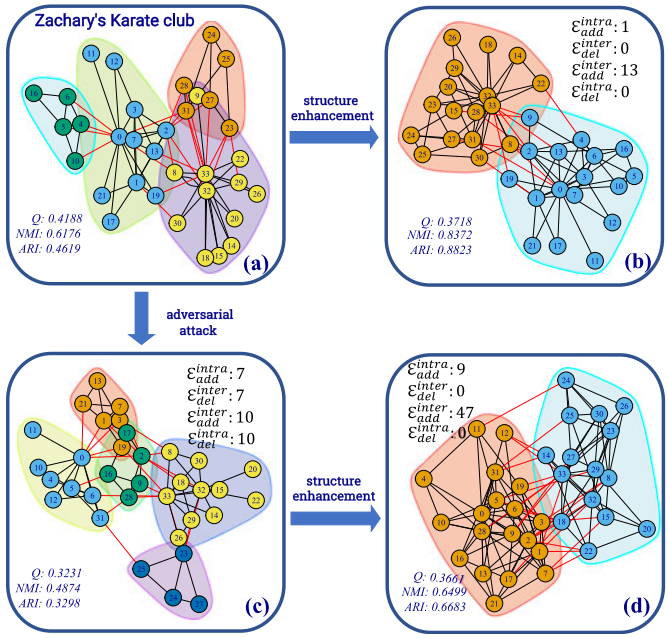


Fig. 6. Enhancement for LOU in Karate network.

similarity index behaves differently on various networks and various budgets, answering **RQ3**. In Karate ($\beta_a \in (1.0, 2.0)$), *RobustECD-SE(single)*s with first-order similarity have relatively good performance while those with second-order and high-order similarity have relatively poor performance. Since that the scale of Karate is particularly small and first-order similarity are sufficient to capture structure features. In this case, first-order similarity indices are complementary, second-order and high-order similarity could be redundant or even negative. In Polblogs ($\beta_a \in (0.5, 2.5)$), *RobustECD-SE(single)*s achieve competitive performance against *RobustECD-SE(all)* except for those with Jaccard, Salton and high-order similarity, which turn out to be redundant.

4.6.4 Explanatory Visualization of RobustECD-GA

Next, we further investigate how the *RobustECD* optimizes the performances of different detection algorithms. Since the mechanism of *RobustECD-SE* has been presented in Fig 4, we only visualize the *RobustECD-GA* for algorithm LOU in Karate network, as shown in Fig. 6.

The community structure found by LOU in the original network is shown in Fig. 6a, where there are four communities. Since the number of communities found by LOU is more than the ground truth ($\phi_{real} = 2$), as mentioned in Table 3, extra inter-community edge addition is available when $\phi_S > \phi_{real}$ ($S = \texttt{LOU}$). The result of *RobustECD-GA* in the original network is shown in Fig. 6b, where the enhancement scheme consists of 1 intra-community edge addition and 13 inter-community edge additions, and achieves a significant improvement on community detection, leading to the increase of 35.56 and 91.02 percent in NMI and ARI, respectively. Essentially, a large number of inter-community edge additions successfully merge small clusters into larger ones, resulting in more accurate partitions.

Notably, the decrease of modularity here (from 0.4188 to 0.3718) can explain why we don't design modularity as fitness function directly. By comparing the information in Figs. 6a

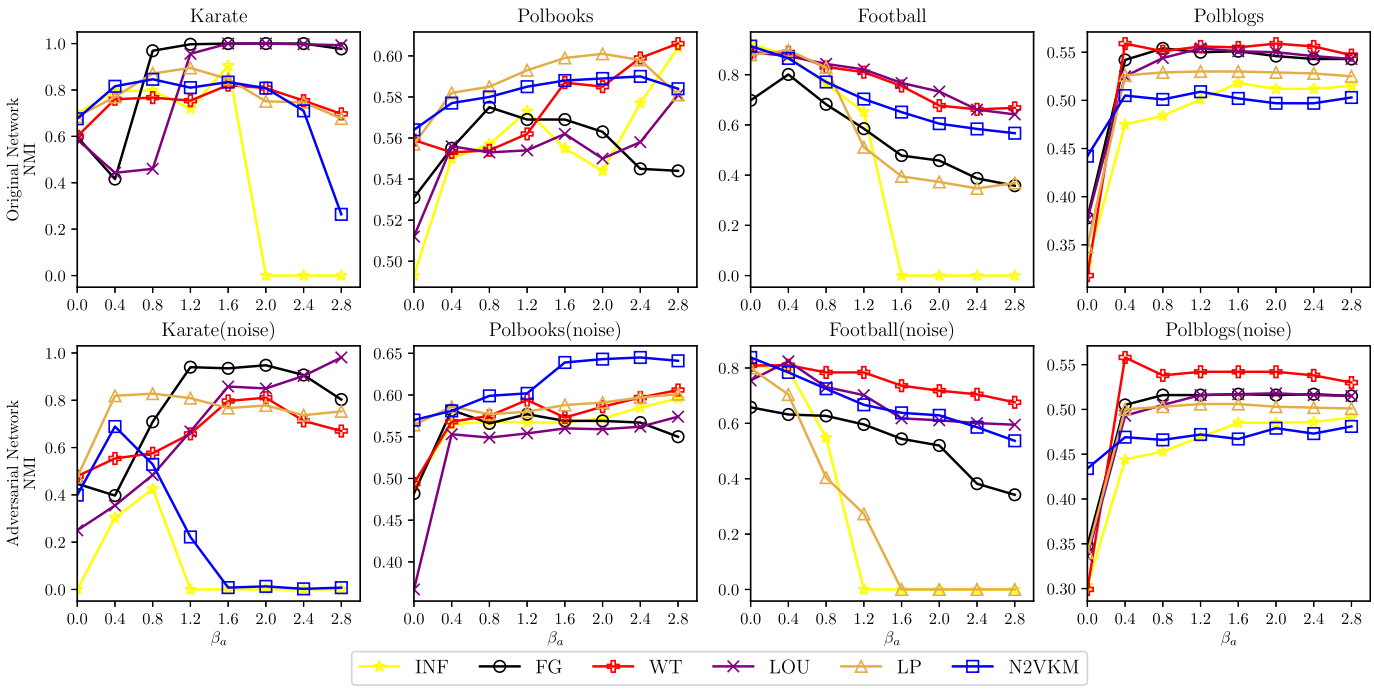


Fig. 7. The impact of modification budget β_a on the performance of *RobustECD-SE*.

and 6b, a community partition with larger modularity does not mean closer to the ground-truth community structure. Therefore, if we have knowledge of community information, we can combine modularity with the true number of clusters, to obtain more accurate optimization guidance. This has been shown to have excellent performance in enhancing community detection. However, when facing unlabeled networks, the fitness function degrades to modularity, i.e., $f = \mathcal{Q}$, so the *RobustECD-GA* may be weakened to a certain extent.

Now, consider the adversarial network obtained by $\mathcal{Q}$-Attack, as shown in Fig. 6c. $\mathcal{Q}$-Attack keeps the number of edges unchanged during community deception and achieves a 22.85 percent reduction in modularity with an attack budget of 17. As we can see, community structure suffers from structural damage and a new cluster that contains the fringe vertices in the original network is discovered. We then deploy structure enhancement to this adversarial network and obtain the enhanced network shown in Fig. 6d. Compared with the community partition in Fig. 6c, LOU achieves a better partition, which even surpasses that in the original network shown in Fig. 6a. Such result suggests that our enhancement algorithms can indeed help the existing community detection algorithms defend against adversarial attacks. More interestingly, it seems that such structure enhancement not only repairs the broken network structure caused by adversarial attack, but also further optimizes it to obtain a clearer community structure, answering **RQ4**.

4.6.5 Parameter Sensitivity

In this subsection, we discuss the impact of key parameters on the performance of *RobustECD-SE* (see the Appendix C, available in the online supplemental material, for the sensitivity analysis in *RobustECD-GA*). The metrics with a budget of 0 corresponds to the original results.

First, we present the evaluate results of *RobustECD-SE* in Fig. 7, from which one can see that such impact behaves differently on various networks. Specifically, Polblogs has a low average NMI equal to 0.373, and the performance of *RobustECD-SE* is relatively stable with the increase of budget; Polbooks has an average NMI equal to 0.534, and its performance curve is messy but basically goes up; for Karate with the average NMI equal to 0.646, when the budget is relatively large, FG and LOU performs well while INF and N2VKM suffers from negative enhancement; Football has an average NMI up to 0.869, the performance of *RobustECD-SE* drops steadily with the increase of budget. Essentially, the impact of modification budget on the performance of *RobustECD-SE* is influenced by the network structure. That is, for networks with weak community structures and low average performance metrics, like Polblogs, most community detection algorithms have huge spaces to be enhanced. But, for those networks with strong community structures like Football, it is difficult for *RobustECD-SE* to further enhance the community detection, i.e., adding or deleting more links may even weaken the stable community structure, leading to the degradation of performance.

Finally, Fig. 8 shows the evaluate results of *RobustECD-SE* in large-scale networks. As we can see, *RobustECD-SE* still works in large-scale networks in most cases, and the performance drops slowly as the budget increase.

4.6.6 Computational Complexity Analysis

In order to compare the efficiency of our *RobustECD*, we roughly estimated their time complexity as follows.

- The most computationally expensive part of *RobustECD-GA* is the calculation of fitness, which consists of modularity $\mathcal{Q}$ and the number of communities ϕ_S, so an extra community detection is necessary before the

TABLE 7
The Average Running Time (s) of the Four Enhancement Algorithms

Time (s) / Dataset / Method	Karate	Polbooks	Football	Polblogs	Amazon	DBLP
Edmot	0.01	0.03	0.05	1.37	2.50	24.10
WERW-Kpath	2.10	11.40	25.77	1200.00	5000.00	68900.00
RobustECD-GA	47.40	136.00	170.00	13900.00	——	——
RobustECD-SE	0.12	0.24	0.38	17.10	357.00	5440.00

The test is performed on LOU with the same experimental setup.

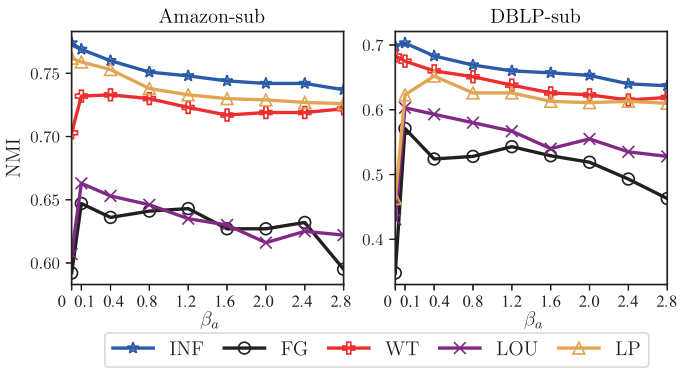


Fig. 8. The impact of modification budgets β_a on the performance of *RobustECD-SE* for large-scale networks.

calculation of fitness. Besides, selection, crossover and mutation are consisted of sampling and edge operations, and have a cost of $\mathcal{O}(|\mathcal{E}|)$, where $|\mathcal{E}|$ is the number of edges in the original network. So *RobustECD-GA* runs in time $\mathcal{O}(\phi_p \cdot \mathcal{T}_{ga} \cdot \max(|S|, |\mathcal{E}|))$, where $|S|$ is the time complexity of the target community detection algorithm.

- The most computationally expensive part of *RobustECD-SE* is the threshold selection, which has a cost of $\mathcal{O}(|\mathcal{E}_{co}|)$, where $|\mathcal{E}_{co}|$ is the number of edges in the co-occurrence network. $\mathcal{G}_{co}$ can be much denser than the original network and have up to $n(n-1)/2$ edges. So the time complexity of *RobustECD-SE* is no more than $\mathcal{O}(n^2)$.

Moreover, we evaluate the efficiency of *RobustECD* by directly comparing the running time with baselines. The average running time (in seconds) of the four algorithms are presented in Table 7. As we can see, although the *RobustECD-GA* performs well on small-scale networks, it is limited by the optimization mode and does not scale well on large networks. Instead, *RobustECD-SE* has a relatively small time complexity and scales well on large networks.

5 Conclusion

In this paper, we proposed to enhance network structure to improve the performance of existing community detection algorithms. In particular, we put forward two structure enhancement algorithms, namely *RobustECD-GA* and *RobustECD-SE*, taking both robustness and generalization into account. Extensive experimental results demonstrate the superiority of our methods in helping six common community detection algorithms achieve significant performance improvements for both real-world networks and adversarial networks, and further solve the resolution limit in modularity optimization and achieve consensus partitions. We believe this could be a fruitful avenue of future research that address more complex situations like overlapping community in dynamic networks.

Acknowledgments

The authors would like to thank all the members in the IVSN Research Group, Zhejiang University of Technology. This work was supported in part by the National Natural Science Foundation of China under Grant 61973273, in part by the Zhejiang Provincial Natural Science Foundation of China under Grants LR19F030001 and LGF20F020016, in part by the National Key R&D Program of China under Grant 2020YFB1006104, and in part by the Hong Kong Research Grants Council through the GRF under Grant CityU11206320. Jiajun Zhou and Zhi Chen contributed equally to this work.

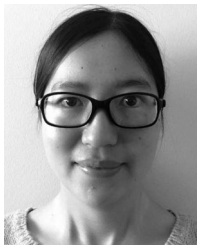
References

[1] S. H. Strogatz, "Exploring complex networks," *Nature*, vol. 410, no. 6825, pp. 268–276, 2001.
[2] M. E. J. Newman, "The structure and function of complex networks," *SIAM Rev.*, vol. 45, no. 2, pp. 167–256, 2003.
[3] M. E. J. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Phys. Rev. E*, vol. 74, no. 3, 2006, Art. no. 036104.
[4] J. Chen *et al.*, "GA-based Q-attack on community detection," *IEEE Trans. Comput. Social Syst.*, vol. 6, no. 3, pp. 491–503, Jun. 2019.
[5] V. Fionda and G. Pirro, "Community deception or: How to stop fearing community detection algorithms," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 4, pp. 660–673, Apr. 2018.
[6] M. Ciglan, M. Laclavík, and K. Nørvåg, "On community detection in real-world networks and the importance of degree assortativity," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2013, pp. 1007–1015.
[7] S. Fortunato and M. Barthelemy, "Resolution limit in community detection," *Proc. Nat. Acad. Sci. USA*, vol. 104, no. 1, pp. 36–41, 2007.
[8] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Phy. A: Statist. Mech. Appl.*, vol. 390, no. 6, pp. 1150–1170, 2011.
[9] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Phys. Rev. E*, vol. 69, no. 2, 2004, Art. no. 026113.
[10] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *J. Statist. Mech.: Theory Experiment*, vol. 2005, no. 9, 2005, Art. no. P09008.
[11] M. El-Moussaoui, T. Agouti, A. Tikniouine, and M. El Adnani, "A comprehensive literature review on community detection: Approaches and applications," *Procedia Comput. Sci.*, vol. 151, pp. 295–302, 2019.
[12] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Statist. Mech.: Theory Experiment*, vol. 2008, no. 10, 2008, Art. no. P10008.
[13] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proc. Nat. Acad. Sci. USA*, vol. 99, no. 12, pp. 7821–7826, 2002.
[14] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Phys. Rev. E*, vol. 70, no. 6, 2004, Art. no. 066111.
[15] U. von Luxburg, "A tutorial on spectral clustering," *Statist. Comput.*, vol. 17, no. 4, pp. 395–416, 2007.
[16] S. Fortunato, "Community detection in graphs," *Phys. Rep.*, vol. 486, no. 3–5, pp. 75–174, 2010.
[17] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proc. Nat. Acad. Sci. USA*, vol. 105, no. 4, pp. 1118–1123, 2008.
[18] V. Zlatić, A. Gabrielli, and G. Caldarelli, "Topologically biased random walk and community finding in networks," *Phys. Rev. E*, vol. 82, no. 6, 2010, Art. no. 066109.
[19] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Phys. Rev. E*, vol. 76, no. 3, 2007, Art. no. 036106.

[20] P.-Z. Li, L. Huang, C.-D. Wang, J.-H. Lai, and D. Huang, "Community detection by motif-aware label propagation," *ACM Trans. Knowl. Discov. Data*, vol. 14, no. 2, pp. 1–19, 2020.
[21] L. Huang, H.-Y. Chao, and Q. Xie, "MuMod: A micro-unit connection approach for hybrid-order community detection," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 1, 2020, pp. 107–114.
[22] D. Bo, X. Wang, C. Shi, M. Zhu, E. Lu, and P. Cui, "Structural deep clustering network," in *Proc. Web Conf.*, 2020, pp. 1400–1410.
[23] S. Fan, X. Wang, C. Shi, E. Lu, K. Lin, and B. Wang, "One2multi graph autoencoder for multi-view graph clustering," in *Proc. Web Conf.*, 2020, pp. 3070–3076.
[24] P. De Meo, E. Ferrara, G. Fiumara, and A. Provetti, "Enhancing community detection using a network weighting strategy," *Inf. Sci.*, vol. 222, pp. 648–668, 2013.
[25] P. G. Sun, "Weighting links based on edge centrality for community detection," *Phy. A: Statist. Mech. Appl.*, vol. 394, pp. 346–357, 2014.
[26] D. Lai, H. Lu, and C. Nardini, "Enhanced modularity-based community detection by random walk network preprocessing," *Phys. Rev. E*, vol. 81, no. 6, 2010, Art. no. 066118.
[27] P.-Z. Li, L. Huang, C.-D. Wang, and J.-H. Lai, "EdMot: An edge enhancement approach for motif-aware community detection," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 479–487.
[28] A. Lancichinetti and S. Fortunato, "Consensus clustering in complex networks," *Sci. Rep.*, vol. 2, 2012, Art. no. 336.
[29] J. Dahlin and P. Svenson, "Ensemble approaches for improving community detection methods," *J. Radioanal. Nucl. Chem.*, vol. 79, no. 3, pp. 191–194, 1997.
[30] D. He, H. Wang, D. Jin, and B. Liu, "A model framework for the enhancement of community detection in complex networks," *Phy. A: Statist. Mech. Appl.*, vol. 461, pp. 602–612, 2016.
[31] C. Szegedy *et al.* "Intriguing properties of neural networks," in *Proc. 2nd Int. Conf. Learn. Representations*, 2014, pp. 1–10.
[32] M. Waniek, T. P. Michalak, M. J. Wooldridge, and T. Rahwan, "Hiding individuals and communities in a social network," *Nat. Hum. Behav.*, vol. 2, no. 2, pp. 139–147, 2018.
[33] J. Li, H. Zhang, Z. Han, Y. Rong, H. Cheng, and J. Huang, "Adversarial attack on community detection by hiding individuals," in *Proc. Web Conf.*, 2020, pp. 917–927.
[34] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval.* Auckland, New Zealand: Mcgraw-Hill, 1983.
[35] P. Jaccard, "Étude comparative de la distribution florale dans une portion des alpes et des jura," *Bull. de la Societe Vaudoise des Sciences Naturelles*, vol. 37, pp. 547–579, 1901.
[36] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A.-L. Barabási, "Hierarchical organization of modularity in metabolic networks," *Science*, vol. 297, no. 5586, pp. 1551–1555, 2002.
[37] L. A. Adamic and E. Adar, "Friends and neighbors on the web," *Social Netw.*, vol. 25, no. 3, pp. 211–230, 2003.
[38] T. Zhou, L. Lü, and Y.-C. Zhang, "Predicting missing links via local information," *Eur. Phys. J. B*, vol. 71, no. 4, pp. 623–630, 2009.
[39] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 5165–5175.
[40] L. Lü, C.-H. Jin, and T. Zhou, "Similarity index based on local paths for link prediction of complex networks," *Phys. Rev. E*, vol. 80, no. 4, 2009, Art. no. 046122.
[41] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Comput. Netw. ISDN Syst.*, vol. 30, no. 1–7, pp. 107–117, 1998.
[42] D. Lin *et al.*, "An information-theoretic definition of similarity," in *Proc. Int. Conf. Mach. Learn.*, 1998, pp. 296–304.
[43] W. W. Zachary, "An information flow model for conflict and fission in small groups," *J. Anthropol. Res.*, vol. 33, no. 4, pp. 452–473, 1977.
[44] M. E. Newman, "Modularity and community structure in networks," *Proc. Nat. Acad. Sci. USA*, vol. 103, no. 23, pp. 8577–8582, 2006.
[45] L. A. Adamic and N. Glance, "The political blogosphere and the 2004 U.S. election: Divided they blog," in *Proc. 3rd Int. Workshop Link Discov.*, 2005, pp. 36–43.
[46] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowl. Inf. Syst.*, vol. 42, no. 1, pp. 181–213, 2015.
[47] S. Monti, P. Tamayo, J. Mesirov, and T. Golub, "Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data," *Mach. Learn.*, vol. 52, no. 1–2, pp. 91–118, 2003.
[48] L. Hubert and P. Arabie, "Comparing partitions," *J. Classification*, vol. 2, no. 1, pp. 193–218, 1985.
[49] P. Pons and M. Latapy, "Computing communities in large networks using random walks," in *Proc. Int. Symp. Comput. Inf. Sci.*, 2005, pp. 284–293.
[50] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 855–864.

Jiajun Zhou received the BS degree in automation in 2018 from the Zhejiang University of Technology, Hangzhou, China, where he is currently working toward the MS degree in control theory and engineering with the College of Information Engineering. His research interests include graph mining and deep learning, with a focus on network security.

Zhi Chen received the BS and MS degrees in EECS from UC Berkeley, in 2019 and 2020, respectively. He is currently working toward the PhD degree in computer science with the University of Illinois, Urbana-Champaign. At UC Berkeley, he was a research assistant with the Center for Long-Term Cybersecurity from 2019 to 2020, and with BAIR Lab in 2018. His research interests include security and machine learning.



Min Du received the bachelor's and master's degrees from Beihang University, and the PhD degree from the School of Computing, University of Utah, in 2018. From 2018 to 2019, she was a postdoctoral scholar with EECS Department, UC Berkeley. She is currently an AI security researcher with Palo Alto Networks. Her research interests include big data analytics and machine learning security.

Lihong Chen received the BS degree from the Zhejiang University of Technology, Hangzhou, China, in 2018. She is currently working toward the MS degree with the College of Information Engineering, Zhejiang University of Technology, Hangzhou, China. Her research interests include social network analysis, evolutionary computing, and deep learning.

Shanqing Yu received the MS degree from the Graduate School of Information, Production, and Systems, Waseda University, Japan, in 2008, and the PhD degree from the School of Computer Engineering in 2011. She is currently a lecturer with the College of Information Engineering, Zhejiang University of Technology. Her research interests include intelligent computation, data mining, and intelligent transport systems.

Guanrong Chen (Fellow, IEEE) received the MSc degree in computer science from Sun Yat-sen University, Guangzhou, China, in 1981, and the PhD degree in applied mathematics from Texas A&M University, College Station, Texas, in 1987. Since 2000, he has been a chair professor and the founding director of the Centre for Chaos and Complex Networks, City University of Hong Kong. Prior to that, he was a tenured full professor with the University of Houston, Texas. He was the recipient of the 2011 Euler Gold Medal, Russia, and conferred the Honorary Doctorates by the Saint Petersburg State University, Russia in 2011, and by the University of Le Havre, Normandy, France, in 2014. He is a member of the Academy of Europe and a fellow of The World Academy of Sciences, and is a highly cited researcher in engineering and in mathematics according to Thomson Reuters.

Qi Xuan (Member, IEEE) received the BS and PhD degrees in control theory and engineering from Zhejiang University, Hangzhou, China, in 2003 and 2008, respectively. From 2008 to 2010, he was a postdoctoral researcher with the Department of Information Science and Electronic Engineering, Zhejiang University, and in 2010 and 2017, a research assistant with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong. From 2012 to 2014, he was a postdoctoral fellow with the Department of Computer Science, University of California, Davis, CA, USA. He is currently a professor with the Institute of Cyberspace Security, College of Information Engineering, Zhejiang University of Technology, Hangzhou, China. His research interests include network science, graph data mining, cyberspace security, machine learning, and computer vision.