This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2933833, IEEE Transactions on Knowledge and Data Engineering

Target Defense Against Link-Prediction-Based Attacks via Evolutionary Perturbations

Shanqing Yu, Minghao Zhao, Chenbo Fu, Jun Zheng, Huimin Huang, Xincheng Shu, Qi Xuan, *Member, IEEE* and Guanrong Chen, *Fellow, IEEE*

Abstract—In social networks, by removing some target-sensitive links, privacy protection might be achieved. However, some hidden links can still be re-observed by link prediction methods on observable networks. In this paper, the conventional link prediction method named Resource Allocation Index (RA) is adopted for privacy attacks. Several defense methods are proposed, including heuristic and evolutionary approaches, to protect targeted links from RA attack. In particular, incremental computation is proposed for accelerating the calculation of fitness in evolutionary approaches. This is the first time to study privacy protection for targeted links against similarity based link prediction attacks. Some links are randomly selected from original network as targeted links for experimentation. The experimental results on nine real-world networks demonstrate the superiority of the evolutionary perturbations, especially EDA, for defending against RA attack. Moreover, experimental results show that the proposed perturbation generated by EDA is transferable and can even defend against other link prediction attacks which are based on high order similarity between pairwise nodes, although it is designed to prevent RA attack.

Index Terms—social network, link prediction, resource allocation index, target defense, evolutionary perturbations, transferability.

1 Introduction

MANY complex systems in the real world can be represented by networks, where the nodes denote the entities in the real systems and links capture various relationships among them [1]–[5]. For human society, it can be naturally described as social networks where nodes and links represent the individual people and the relationship between them, respectively. While social networks contain abundant information, there is an increasing concern about privacy issues since more and more personal information could be obtained by others online. In the field of social privacy protection, many algorithms have been developed for protecting the privacy of users, such as identity, attributes and relationship, from different situations in which different public information was exposed to adversaries [6]–[9]. In social network, there may be some sensitive links indicating certain kind of relationship between two individuals, e.g. romantic relationship [10], transaction or communication relationship. In some cases, these sensitive relationships relate to personal privacy so that people are unwilling to make them public. Considering this situation, we focus on preserving link privacy in social networks in this paper.

In retrospect, Zheleva *et al.* [9] proposed the concept of link re-identification attack, which refers to inferring sensitive relationships from anonymized network data. If the sensitive links can be identified by the released data, then this means privacy breach. In early days, link perturbation was considered as a common technique to preserve sensitive links [11], e.g., a data publisher can randomly remove real links and insert fake links into an original network so as to preserve the sensitive links from being identified. Ying *et al.* [12] investigated the relationship between the level of link randomization and the possibility to infer the presence of a link in a network. Further, Ying *et al.* [13] investigated the effect of link randomization on protecting privacy of sensitive links, and they found that similarity indices can be utilized by adversaries to significantly improve the accuracy in predicting sensitive links. Unlike the above situations, Fard *et al.* [14] assumed that all links in a network are sensitive, and they proposed to apply subgraph-wise perturbations into a directed network, which randomize the destination of a link within some subnetworks thereby limiting the link disclosure. It should be noted that their method perturbs every link in the network based on a certain probability.

Different from the attacks mentioned above, we assume that the adversaries may apply link prediction to detect sensitive links from observable networks. link prediction, which attempts to uncover missing links or to predict future interactions between pairwise nodes based on observable links and other external information, can be applied to predict the potential relationship between two individuals [15]. From another perspective, it may also increase the risk of sensitive link disclosure. One naive way to hide sensitive links is to remove them directly from the published network, However, it may still be disclosed by link prediction and consequently lead to privacy breach. Michael *et al.* [16] presented a link reconstruction attack, which is a method that attacker can use to infer a user's connections to others with high accuracy, but they did not mention how to defend against the so-called link-reconstruction attack. Naturally, one can consider finding a way to prevent link prediction

- *S. Yu, M. Zhao, C. Fu, J. Zheng, H. Huang, X. Shu, and Q. Xuan are with the College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China (e-mail: yushanqing@zjut.edu.cn, yzbyzmh1314@163.com, cbfu@zjut.edu.cn, 2111703109@zjut.edu.cn, saffronHuang@163.com, sxc.shuxincheng@foxmail.com, xuanqi@zjut.edu.cn).*
- *G. Chen is with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong SAR, China (e-mail: eegchen@cityu.edu.hk).*
- *Corresponding author: Qi Xuan.*

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2933833, IEEE Transactions on Knowledge and Data Engineering

attack. Since link-reconstruction attack or link prediction attack aims to find out some real but unobservable links, the defense against link prediction attacks is also target-directed, which means that one has to preserve the targeted links from being successfully predicted by link prediction.

Currently, most existent approaches on link prediction are based on the similarity between pairwise nodes with the assumption that the more similar a pair of nodes are, the more likely a link exists between them. In sociology, triadic closure is a popular concept to explain the evolution of a social network, i.e., if the triad of three individuals is not closed, then the person connected to the other two individuals would like to close this triad in order to achieve a closure in the relationship network [17]. Here, triadic closure breeds several neighborhood-based indices to measure the proximity of pairwise nodes. Zhou *et al.* [18] compared several neighborhood-based similarity indices on disparate networks, and they found that Resource Allocation (RA) index behaves best. Later, Lü *et al.* [19] published a survey on link prediction. Recently, some new link prediction methods based on network embedding have emerged in academic community, such as DeepWalk [20] and Node2vec [21]. Many of them considers high-order similarity between nodes. In general, since many link prediction algorithms are designed based on network structures, data owners can add well-designed perturbations into the original network to reduce the risk of targeted-link disclosure due to link prediction attacks.

Most recently, there exists some works studying the adversarial attacks on neural networks for graph data, e.g. Zügner et al. [22] proposed adversarial attack on node classification model based on graph convolutional networks [23]. One motivation for this study is to test the robustness of deep neural networks for graphs to adversarial attacks. It will be further shown that our proposed defensive methods can also serve as adversarial attack on link prediction which is based on traditional similarity indices. Note that, since unrestricted perturbations are possible to change the network structure tremendously, one thus needs to set some restrictions to make the perturbations imperceptible [22], e.g. limit the size of link perturbation. One the other hand, the size of perturbation can also be considered as the defensive cost, i.e., it could be easier for individuals to add/delete fewer friends than more. This reveals that the key of defending against link prediction attacks depends on how to find Optimal Link Perturbation (OLP), which takes the defensive effect and the size of perturbation into account, to invalid link prediction algorithms, so that the removed sensitive links are hard to be correctly predicted. However, to the best of our knowledge, there are very few studies on this perspective of conducting target defense against link prediction attacks. In this paper, the proposed approach is to randomly select some links as sensitive ones, assuming that the adversary employs the RA index to predict the missing links. Several approaches are proposed, including heuristic and evolutionary perturbations, to defend against this attack. The main contributions of this paper are summarized as follows:

- A novel target defense method against link prediction attacks is proposed to hide the sensitive links from being predicted.
- Techniques using heuristic as well as evolutionary perturbations to defend link prediction attacks are developed. For heuristic perturbation, it is designed for reducing the similarity of top ranked node pairs. For evolutionary perturbations, a new fitness function is designed, taking into consideration of both precision and AUC, which can reduce the calculation of fitness values by computing the increments after perturbations according to our proof.
- Massive experiments have done and the results demonstrate the superiority of evolutionary perturbations, especially EDA (estimation of distribution algorithm), and it is found that the perturbation generated by EDA is transferable and can even defend against other link prediction attacks which are based on high order similarity between pairwise nodes.

The rest of the paper is organized as follows. In Sec. 2, the network model and some standard metrics about link prediction are introduced. The proposed target defense against link prediction attacks is discussed. In Sec. 3, several methods are developed, including heuristic and evolutionary perturbations. In Sec. 4, experiments are conducted to examine the proposed methods, and the experimental results are analyzed in detail. In Sec. 5, the paper is concluded with some discussions on future directions of research.

2 Network Model and Performance Metrics

2.1 Network Model

An undirected network is presented by $G(V, E)$, where V and E denote the sets of nodes and links, respectively. Define the set of all node pairs in the network by $\Omega = \{(i, j)|\{i, j\} \in V, i \neq j\}$. In an undirected network, link (i, j) is considered the same as link (j, i) and Ω contains $\binom{k}{2} = \frac{|V|(|V|-1)}{2}$ possible links. All observable links E are divided into two groups: the training set E^T and the validation set E^V, where $E^T \cup E^V = E$ and $E^T \cap E^V = \emptyset$. Furthermore, we define the set of unknown node pairs and the set of non-existent node pairs by $U = \Omega - E^T$ and $N = \Omega - E$, respectively. Obviously, $U = N + E^V$.

2.2 Link Prediction

In link prediction, the objective is to predict E^V within U utilizing the structure information provided by E^T. There are several metrics to measure the similarity of node pairs, among which local indices are computationally efficient and well-performed. According to [18] and [24], the RA index stands out from several local similarity-based indices. It is defined as follows:

$$RA_{ij} = \sum_{k \in \Gamma(i) \cap \Gamma(j)} \frac{1}{d_k}, \tag{1}$$

where $\Gamma(i)$ denotes the one-hop neighbors of node i and d_k denotes the degree of node k. To reduce the RA value of a certain node pair, one can decrease the number of common neighbors between them or increase the degree of their common neighbors.

In this paper, two standard metrics are used for measuring the performance of link prediction, i.e., precision and

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2933833, IEEE Transactions on Knowledge and Data Engineering

AUC (the area under the receiver operating characteristic curve), which are defined as follows:

- **Precision**
  Precision is the ratio of real missing links to predicted links. To calculate it, first sort node pairs in U in descending order according to their similarity values, as $\hat{U}$, and then the top-k node pairs $\hat{U}^k$ in $\hat{U}$ are chosen as the predicted ones. The definition of precision is

$$precision = \frac{|\hat{U}^k \cap E^V|}{|E^V|}, \tag{2}$$

  where $k = |E^V|$ will be used in this paper.
- **AUC**
  AUC measures the probability that a random missing link in E^V is given a higher similarity value than a random non-existent node pair in N. To calculate it, choose each missing link in E^V and each non-existent link in N to compare their values: if there are $n_>$ times where a missing link has a higher value and $n_=$ times their values are the same, then the AUC is computed as follows:

$$AUC = \frac{n_> + 0.5n_=}{n}, \tag{3}$$

  where $n = |E^V| * |N|$ will be used in this paper.

2.3 Target Defense in Link Prediction Attack

A new target defense method is proposed here against link prediction attacks. For instance, as shown in Fig. 1, simply deleting the sensitive link (i, j) cannot make it hidden, because the value of RA_{ij} is equal to $\frac{5}{6}$, which is much higher than other node pairs in the network. The deleted link could be easily re-identified, if RA is employed to predict missing links. However, if one deletes link (i, q) and meanwhile inserts link (p, v), then $RA_{ij} = \frac{1}{3}$, which is smaller than (u, p).

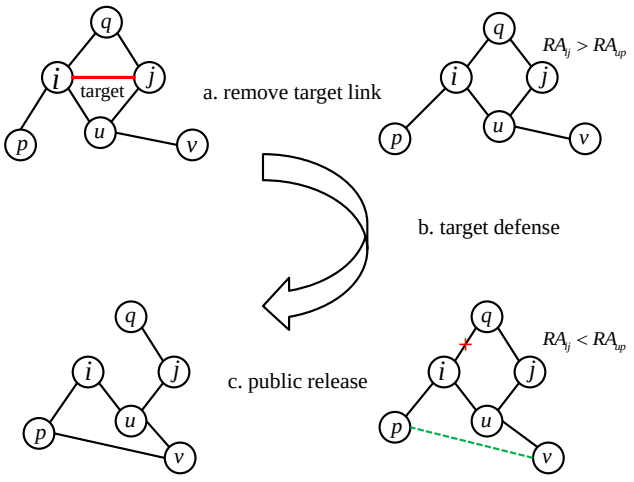


Fig. 1. The whole process of defending link prediction attacks via link perturbations. Red link (i, j) is sensitive, which needs to be hidden, red cross means deleting link (i, q) and green dashed link (p, v) is the inserted one. The released network can preserve the sensitive link (i, j) from being identified by the RA index, to some extent.

TABLE 1
The meaning of each notation before and after perturbations.

Meaning	before	after
training set	E^T	$\tilde{E}^T$
validation set	E^V	-
the set of all node pairs in network	Ω	-
the set of unknown node pairs	U	$\tilde{U}$
the set of non-existent node pairs	N	$\tilde{N}$
link perturbation	-	$\tilde{E}$
the set of deleted links	E_{del}	-
the set of inserted links	E_{add}	-
deleted links	ℓ_{del}	-
inserted links	ℓ_{del}	-

In a network $G(V, E)$, the set E is divided into two disjoint groups: E^T and E^V. Take E^V as the set of sensitive links. The set of link perturbations $\tilde{E}$ is added into E^T, which means deleting E_{del} from E^T and then inserting E_{add} into E^T so as to reduce sensitive links E^V being re-identified by a link prediction algorithm $\mathcal{L}$, which consequently leads to the risk of privacy leakage.

Although the defensive effect may be improved as the number of modified links increases, the size of modifications should be minimized, taking into account both defensive effect and the size of perturbation. Therefore, the following two restricted conditions are considered:

- the numbers of deleted and inserted links should be identical to make the total number of links unchanged.
- the numbers of deleted and inserted links should be sparse to ensure data utility.

Ut supra, target defense against link prediction attacks can be described mathematically as follows:

$$\begin{aligned} \min_{\tilde{E}} \quad & \mathcal{L}(E^T + \tilde{E}, E^V) \\ \text{s.t.} \quad & \begin{cases} |\tilde{E}| = |E_{add}| + |E_{del}| = 2 \cdot |E_{del}| \ll |E^T| \\ E_{del} \subset E^T, \quad E_{add} \subset N\,. \end{cases} \end{aligned} \tag{4}$$

After being perturbed, E^T, U and N of the original network are changed to $\tilde{E}^T$, $\tilde{U}$ and $\tilde{N}$, respectively. It then follows that

$$\tilde{E}^T = E^T - E_{del} + E_{add}, \tag{5}$$
$$\tilde{U} = \Omega - (E^T + \tilde{E}) = U + E_{del} - E_{add}, \tag{6}$$
$$\tilde{N} = \Omega - (E^T + \tilde{E}) - E^V = N + E_{del} - E_{add}\,. \tag{7}$$

The relationship of each link set after perturbation is shown in Fig. 2, where the set of non-existent node pairs before and after perturbation are shown in grey. Finally, the notations mentioned above are listed in TABLE 1 for convenience.

3 Methods

In this paper, we assume that the adversary uses the RA index to predict the missing links. The objective here is to preserve E^V from being predicted via adding link perturbations. Then, random, heuristic and evolutionary perturbations are applied, respectively, as discussed below.

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2933833, IEEE Transactions on Knowledge and Data Engineering

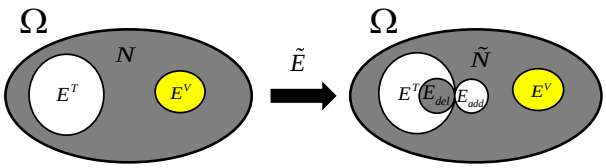


Fig. 2. The venn diagram of each node-pair set after adding perturbation $\tilde{E}$ into the original network.

3.1 Random Perturbations

3.1.1 Randomly Link Rewiring (RLR)

For a given network in which sensitive links are removed, one can randomly delete some links ℓ_{del} from E^T and then insert the same number of new links ℓ_{add}, which exist in N.

3.1.2 Randomly Link Swapping (RLS)

Another common practice for randomizing a network is link swapping, which not only keeps the total number of links unchanged but also preserves the degrees of nodes [25]. As shown in Fig. 3, link swapping removes two randomly chosen links from E^T and creates two new links existing in N before swapping.

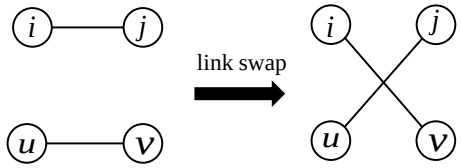


Fig. 3. An example for link swapping. If either (i, v) or (j, u) already exists in E^T, no swapping is performed, and another attempt is made to find a suitable link pair.

3.2 Heuristic Perturbation

To hide the links in E^V, one can degrade the performance of link prediction through decreasing the similarity values of node pairs in E^V and then increasing the values of non-existent node pairs. Here, a greedy strategy is adopted to rewire links. First, the RA index is used to calculate the similarity values of node pairs $(i, j) \in \Omega$ and then sort them in descending order according to their values.

For each node pair $(i, j) \in \Omega$, there are three cases, i.e., $(i, j) \in E^T$, $(i, j) \in N$, and $(i, j) \in E^V$. One can traverse the ordered node pairs and conduct different operations by deleting or inserting links for each case.

- $(i, j) \in E^T$
  If the current node pair (i, j) exists in E^T, directly delete the link. Because the deleted (i, j) will exist in $\tilde{U}$ and (i, j) has a high RA value.
- $(i, j) \in N$
  If the current node pair (i, j) exists in N, select the node k whose degree is the smallest in the one-hop neighborhoods of i and j, except their common neighbors. Then, insert (i, k) or (j, k) to increase the RA value of the current node pair (i, j).
- $(i, j) \in E^V$
  If the current node pair (i, j) exists in E^V, select the common neighbor k whose degree is the smallest and then delete (i, k) or (j, k); or select two common neighbors k and l whose degrees are among the smallest. Then, insert (k, l) to reduce the RA value of the node pair (i, j).

The above operation is shown in Fig. 4. It should ensure that all deleted links $\ell_{del} \in E^V$ and inserted links $\ell_{add} \in N$. The intact pseudo-codes of the algorithms are described in Algorithms 1, 2 and 3.

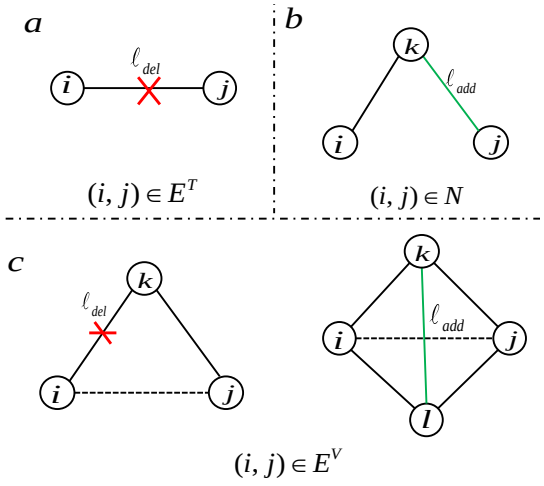


Fig. 4. Four different operations through deleting or inserting links for (i, j) in corresponding situations.

Algorithm 1: Heuristic Perturbation (HP)

Input: E^T, E^V, N, Ω and m.
Output: E_{del} and E_{add}.

Initialize E_{del} and E_{add};
sort Ω in descending order to get $\hat{\Omega}$;
for *each* $(i, j) \in \hat{\Omega}$ **do**
  if $(i, j) \in E^T \cup E^V$ **then**
    if $|E_{del}| < m$ **then**
      ℓ_{del} = ***Delete Link***$((i, j))$;
      $E_{del} = E_{del} \cup \ell_{del}$;
      continue;
    end
  end
  if $(i, j) \in E^V \cup N$ **then**
    if $|E_{add}| < m$ **then**
      ℓ_{add} = ***Add Link***$((i, j))$;
      $E_{add} = E_{add} \cup \ell_{add}$;
    end
  end
  if $|E_{del}| = m$ *and* $|E_{add}| = m$ **then**
    Break;
  end
end
return E_{del} and E_{add}.

3.3 Evolutionary Perturbations

Link perturbation can be treated as a combinatorial optimization problem and the number of candidate com-

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2933833, IEEE Transactions on Knowledge and Data Engineering

Algorithm 2: ***Delete Link***

Input: node pair (i, j)
Output: ℓ_{del}
if $(i, j) \in E^T$ **then**
  $E^T = E^T - (i, j)$;
  return (i, j);
end
if $(i, j) \in E^V$ **then**
  Find $k = \arg\min_k \{d_k | k \in \Gamma(i) \cap \Gamma(j)\}$;
  Find $l = \arg\max_l \{d_l | l \in \{i, j\}\}$;
  if (k, l) *exists* **then**
    $E^T = E^T - (k, l)$;
    return (k, l);
  end
end

Algorithm 3: ***Add Link***

Input: node pair (i, j)
Output: ℓ_{add}
if $(i, j) \in E^V$ **then**
  Find $k = \arg\min_k \{d_k | k \in \Gamma(i) \cap \Gamma(j)\}$;
  Find $l = \arg\min_l \{d_l | l \in \{\Gamma(i) \cap \Gamma(j) - k\}\}$;
  if (k, l) *exists and* $(k, l) \notin E^V$ **then**
    $E^T = E^T \cup (k, l)$;
    return (k, l);
  end
end
if $(i, j) \in N$ **then**
  Find $k = \arg\min_k \{d_k | k \in \{\Gamma(i) \cup \Gamma(j) - \Gamma(i) \cap \Gamma(j)\}\}$;
  if k *exists* **then**
    $E^T = E^T \cup \{(k, l) | (k, l) \notin E^V, l \in \{i, j\}\}$;
    return $\{(k, l) | (k, l) \notin E^V, l \in \{i, j\}\}$;
  end
end

binations of deleted/inserted links is $C^m_{|E^T|} \cdot C^m_{|N|}$, where m is the number of deleted/inserted links.

To hide sensitive links in E^V, one needs to reduce the possibility that these links can be predicted. Both AUC and precision are indicators for evaluating the performance of a link prediction algorithm. AUC evaluates the global performance while precision focuses on pairs with highest similarity values. Therefore, an evolutionary algorithm is designed here to find the OLP by iteratively executing evolutionary operations, such as selection, crossover, and mutation. When the fitness function considering both AUC and precision, converges, it will be considered that the approximate OLP has been found.

Two evolutionary algorithms, i.e. Genetic Algorithm (GA) [26] and Estimation of Distribution Algorithm (EDA) [27] are developed in this paper. The chromosome design, fitness function, selection and mutation operation of the two algroithms are described as follow.

- **Chromosome**

Del	ℓ^1_{del}	ℓ^2_{del}	ℓ^3_{del}	ℓ^4_{del}	ℓ^5_{del}
Add	ℓ^1_{add}	ℓ^2_{add}	ℓ^3_{add}	ℓ^4_{add}	ℓ^5_{add}

Fig. 5. The diagram of chromosome in GA and EDA, respectively. It consists of two parts, i.e. deleted links $\ell_{del} \in E^T$ and inserted links $\ell_{add} \in N$.

The chromosome consists of two parts: $\ell^i_{del} \in E^T$ and $\ell^i_{add} \in N$, $i = 1, 2, ..., m$. The number of deleted and inserted links are identical according to the assumption. The diagram of chromosome is shown in Fig. 5, where the length of chromosome depends on the experimental setup.

- **Fitness**
Two major evaluation indices for link prediction performance, i.e. precision and AUC, are considered in the fitness design. Note that directly taking both precision and AUC as fitness is time-consuming. Thus, a new fitness function is designed to simplify the calculation, as follows:

$$\max_{\ell \in E^V, n \in \tilde{N}} Fitness = \alpha \cdot \sum_{n \in \tilde{N}} \delta(RA_n > \max_{\ell \in E^V} \{RA_\ell\}) + \left(\frac{1}{|\tilde{N}|} \sum_{n \in \tilde{N}} RA_n - \frac{1}{|E^V|} \sum_{\ell \in E^V} RA_\ell\right), \tag{8}$$

where $\delta(x)$ is an indicator function: $\delta(x) = 1$ when x is true; otherwise, $\delta(x) = 0$, and α is a tunable parameter. The fitness function consists of two parts: (1) denoting the number of non-existent links with higher similarity values than the most predictable sensitive pairs, which has greater influence on the precision; (2) denoting the difference in the average similarity values between non-existent pairs and sensitive pairs, which affects the AUC more.

- **Selection Operation**
The selection operation is conducted on roulette. To make the fitness values positive, we apply exponential transform (e^x) to the fitness. At the same time, retain n_{elite} elites according to the fitness values.
- **Mutation Operation**
Select $n_{mutation}$ chromosomes based on the fitness values for mutation operation. Then, traverse each link in a chromosome and conduct mutation operation to it according to a mutation rate pm. Specifically, one randomly replaces the deleted link ℓ_{del} with another one from E^T, and randomly replace the inserted link ℓ_{add} with another one in N. If it happens to encounter collision, that is, there exist duplicate links in the chromosome, then repeat the above operation until collision disappears.

3.3.1 Genetic Algorithm (GA)

- **Crossover Operation**
Single point crossover is used here and $n_{crossover}$ chromosomes are selected according to the values of fitness and crossover rate pc. If it happens to encounter collision, as shown in Fig. 6, retreat the crossover operation of the duplicate red links.

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2933833, IEEE Transactions on Knowledge and Data Engineering

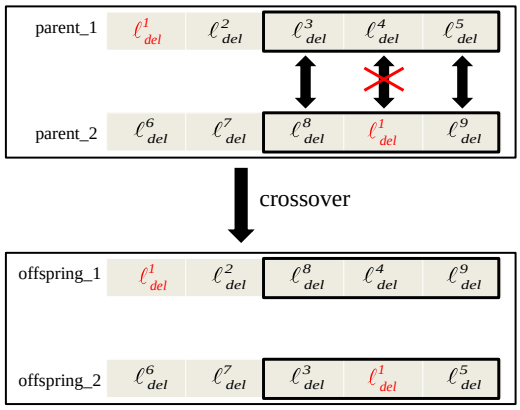


Fig. 6. Crossover of chromosomes in the perspective of deleted links. Red genotype will encounter collision after a direct exchange.

Algorithm 4: GA for Link Perturbation

Input: m, $n_{iteration}$, n_{elite}, $n_{crossover}$, $n_{mutation}$, pc, and pm.
Output: population

Initialize population;
while *not convergence or not reach to* $n_{iteration}$ **do**
  calculate the fitness of population;
  retain n_{elite} $elites$;
  Cro = ***Selection Operation***($n_{crossover}$);
  $\hat{Cro}$ = ***Crossover Operation***(pc, Cro);
  Mut = ***Selection Operation***($n_{mutation}$, $\hat{Cro}$);
  $\hat{Mut}$ = ***Mutation Operation***(pm, Mut);
  population = $elites \cup \hat{Cro} \cup \hat{Mut}$;
end
return population;

Algorithm 5: EDA for Link Perturbation

Input: m, $n_{iteration}$, n_{elite}, $n_{estimation}$, n_{eda}, $n_{mutation}$, and pm.
Output: pop

Initialize population;
while *not convergence or not reach to* $n_{iteration}$ **do**
  calculate the fitness of population;
  retain n_{elite} $elites$;
  $\hat{Eda}$ = ***Generate Population***($n_{estimation}$, n_{eda});
  Mut = ***Selection Operation***($n_{mutation}$, $\hat{Eda}$);
  $\hat{Mut}$ = ***Mutation Operation***(pm, Mut);
  population = $elites \cup \hat{Eda} \cup \hat{Mut}$;
end
return pop;

3.3.2 Estimation of Distribution Algorithm (EDA)

EDA is a new kind of random optimization algorithm based on statistics theory. EDA has obvious differences with GA. As mentioned above, GA applies the crossover operation to generate new individuals; however, EDA searches better individuals through preference sampling and statistical learning. Specifically, one samples $n_{estimation}$ individual chromosomes according to their fitness values and then estimates the probability distributions of the deleted links

Algorithm 6: Generate Population

Input: $n_{estimation}$ and n_{eda}.
Output: population

Initialize population;
Est = ***Selection Operation***($n_{estimation}$);
Estimate the distributions of $P(\ell_{del})$ and $P(\ell_{del})$ by statistical sampling, where $P(\ell_{del}) \approx P(\ell_{del}|Est)$ and $P(\ell_{del}) \approx P(\ell_{add}|Est)$;
while *not reach to* n_{eda} **do**
  Sample E_{del} and E_{add} according to $P(\ell_{del})$ and $P(\ell_{add})$ respectively to generate new individual;
end
return population;

$P(\ell_{del})$ and inserted links $P(\ell_{add})$ based on simple statistics. Finally, one generates n_{eda} chromosomes according to their respective distributions. The pseudo-codes of GA and EDA are described in Algorithms 4, 5 and 6.

3.3.3 Accelerating the Fitness Calculation

Since one should calculate the fitness for each individual at every iteration, the speed of fitness calculation directly affects the computational efficiency of the evolutionary algorithm. It is proposed here to accelerate the calculation of fitness by only recalculating the increments for each individual. Note that, for RA, deleting or inserting (i,j) only affects the values of one-hop neighborhood.

THEOREM 1. *Denote by* $\tilde{N}_{ij}^{recalc}$ *the set of node pairs whose RA values need to be recalculated in* $\tilde{N}$. *For the RA index, by either deleting or inserting* (i,j), *one has* $|\tilde{N}_{ij}^{recalc}| \leq \frac{1}{2}(k_i^2+k_j^2+k_i+k_j)+1$.

Proof. Assume that $(u,v) \in \tilde{N}_{ij}^{recalc}$, and denote by G the original network and by $\Gamma(i)$ the one-hop neighbors of node i in G. First, consider $(i,j) \in G$, and delete (i,j) from G.

- $u=i, v=j$
  Before deleting (i,j), one has $(i,j) \notin N$; after deleting (i,j), one has $(i,j) \in \tilde{N}$. Thus, RA_{uv} needs to be calculated.
- $u \neq i, v \neq j$
  If i or $j \in \Gamma(u) \cap \Gamma(v)$ in G, then RA_{uv} needs to be recalculated after deleting (i,j) and $(u,v) \in \{(\Gamma(i),\Gamma(i)),(\Gamma(j),\Gamma(j))\}$; if $\{i,j\} \notin \Gamma(u) \cap \Gamma(v)$ in G, then RA_{uv} needs not to be recalculated.
- $u \neq i, v=j$
  If $i \in \Gamma(u) \cap \Gamma(j)$ in G, then RA_{uv} needs recalculation after deleting (i,j) and $(u,v) \in \{(\Gamma(i),j)\}$; if $i \notin \Gamma(u) \cap \Gamma(j)$ in G, RA_{uv} needs not to be recalculated.
- $u=i, v \neq j$
  $(u,v) \in \{(i,\Gamma(j))\}$ by symmetry.

To sum up, $(u,v) \in \{(i,j),(\Gamma(i),\Gamma(i)),(\Gamma(j),\Gamma(j)),(\Gamma(i),j),(i,\Gamma(j))\}$ and the same for inserting (i,j). Thus,

$$\begin{aligned} \left|\tilde{N}_{ij}^{recalc}\right| &\leq \frac{1}{2}k_i(k_i-1)+\frac{1}{2}k_j(k_j-1)+(k_i+k_j)+1 \\ &= \frac{1}{2}(k_i^2+k_j^2+k_i+k_j)+1\,. \end{aligned} \tag{9}$$

□

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2933833, IEEE Transactions on Knowledge and Data Engineering

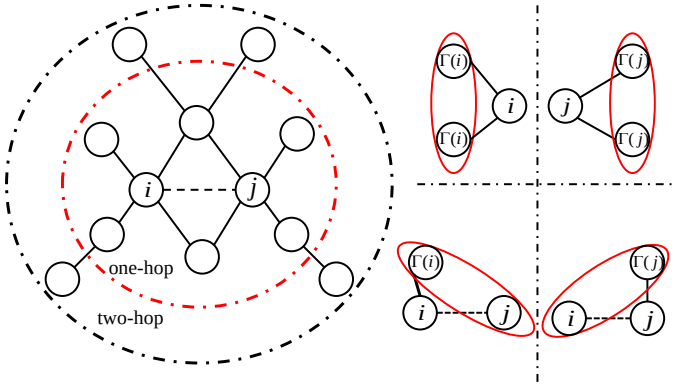


Fig. 7. **Left**: neighborhood of (i, j), where one-hop neighbors are within the red dashed circle. **Right**: four kinds of node pairs whose RA values need to be recalculated if (i, j) is deleted or inserted.

An example to illustrate the influence of deleting (i, j) is shown in Fig. 7. The set $\tilde{N}$ of non-existent node pairs in the perturbed network can be written as

$$\begin{aligned}\tilde{N} &= N + E_{del} - E_{add} \\ &= (N - \tilde{N}_{ij}^{recalc}) + \tilde{N}_{ij}^{recalc} + (E_{del} - E_{add}) \,. \quad (10)\end{aligned}$$

Thus, one only needs to recalculate the RA values in $\tilde{N}_{ij}^{recalc}$, E_{del} and E_{add} to update the fitness leaving RA values in $(N - \tilde{N}_{ij}^{recalc})$ unchanged. Consequently, the complexity of calculating the similarity values of node pairs in the perturbed network is significantly reduced, approximately from $O(|V|^2)$ to $O(m\langle k\rangle^2)$, where m is the number of deleted/inserted links and $\langle k\rangle$ is the average degree of G.

4 Datasets and Experimental Results

4.1 Data Description

Here, nine networks are selected as follows, with topological features shown in TABLE 2.

- Mexican political elite (Mexican) is an undirected network, which contains the core of the political elite including the presidents and their closest collaborators. Links represent significant political, kinship, friendship, or business ties among them [28].
- Dolphin social network (Dolphin) is an undirected social network of dolphins living in a community and links present the frequent associations between pair-wise dolphins [29].
- Train Bombing (Bomb) is an undirected network, which contains contacts between suspected terrorists involved in the train bombing of Madrid on March 11, 2004, as reconstructed from newspapers. Nodes represent terrorists and link between two terrorists means that there was a contact between them [30].
- Lesmis is an undirected network of characters in Victor Hugo's famous novel *Les Miserables*. Nodes denote characters and two nodes are connected if the corresponding characters co-appear in the same chapter of the book [31].
- Game of Thrones (Throne) is an undirected network of character interactions from the novel *A Storm of Swords*, where nodes denote the characters in the novel and a link denotes that two characters are mentioned together in the text [32].
- Jazz is a collaboration network of jazz musicians. Each node is a jazz musician and a link denotes that two musicians have played together in a band [33].
- CS-AARHUS (CS) is a multi-layer social network which consists of five kinds of ralationships (i.e., Facebook, leisure, work, co-authorship and lunch) between the employees of computer science department [34]. In this paper, we simply compress this multi-layer network into single-layer network.
- Email is an undirected communication network at a certain University in the south of Catalonia in Spain. Nodes represent users and each link represents that at least one email was sent between two users [35].
- Geom is a collaboration network of computational geometry. Each node and link represents author and co-author relationship, respectively [36].

TABLE 2
Basic topological features of nine networks. $|V|$ and $|E|$ are the numbers of nodes and edges, respectively; $\langle k\rangle$ is the average degree; C is the clustering coefficient and $\langle d\rangle$ is the average distance.

	$\|V\|$	$\|E\|$	$\langle k\rangle$	C	$\langle d\rangle$
Mexican	35	117	6.686	0.448	2.106
Dolphin	62	159	5.129	0.259	3.357
Bomb	64	243	7.594	0.622	2.691
Lesmis	77	254	6.597	0.573	2.641
Throne	107	352	6.579	0.551	2.904
Jazz	198	2742	27.697	0.617	2.235
CS	61	353	11.574	0.592	2.063
Email	1133	5451	9.622	0.220	3.606
Geom	6158	11898	3.864	0.486	5.313

4.2 Experimental Results

4.2.1 Defensive Effects of Different Link Perturbations

All links are randomly divided into 10 uniform and disjoint sets. One of the sets is selected as a validation set E^V to be the set of sensitive links that need to be protected, and the rest is used as a training set E^T. Use the above-mentioned five methods including random, heuristic and evolutionary perturbations, respectively, to make a crosswise comparison.

In order that every link has the chance to be used for both training and validation set, a 10-fold cross-validation is used to calculate the average precision and AUC. To ensure the sparsity of the perturbations, the proportion of deleted and inserted links in the training set are limited to observe the downtrend of defensive effect with an increasing proportion of perturbations. The proportion is defined as the ratio of deleted/inserted links to all links in the training set. For example, if the proportion equals 0.1, it means that the deleted and inserted links account for 10% of the training set, respectively. To perform a reasonable comparison, add the perturbations using different methods in the same training set and then calculate the precision and AUC in the same validation set.

In RLR, randomly delete and insert a certain proportion of links in each training set and then repeat one hundred times to calculate the precision and AUC. In RLS,

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2933833, IEEE Transactions on Knowledge and Data Engineering

TABLE 3
Average precision and AUC in six networks with different values of α when the proportion of deleted and inserted equals 0.06.

Datasets(GA)	Precision				AUC			
	$\alpha=0$	$\alpha=0.01$	$\alpha=0.1$	$\alpha=1$	$\alpha=0$	$\alpha=0.01$	$\alpha=0.1$	$\alpha=1$
Mexican	0.0818	**0.0364**	**0.0364**	**0.0364**	**0.708**	0.740	0.758	0.754
Dolphin	0.0667	0.00667	0.00667	**0**	**0.698**	0.741	0.749	0.752
Bomb	0.379	0.329	**0.288**	**0.288**	**0.878**	0.899	0.908	0.910
Lesmis	0.272	0.252	**0.244**	0.248	**0.872**	0.879	0.899	0.898
Throne	**0.0943**	0.0971	0.111	0.114	**0.835**	0.863	0.869	0.876
Jazz	**0.397**	0.422	0.439	0.417	**0.952**	0.956	0.958	0.958
Datasets(EDA)	Precision				AUC			
	$\alpha=0$	$\alpha=0.01$	$\alpha=0.1$	$\alpha=1$	$\alpha=0$	$\alpha=0.01$	$\alpha=0.1$	$\alpha=1$
Mexican	0.0727	**0.0273**	**0.0273**	0.0364	**0.701**	0.745	0.744	0.744
Dolphin	0.0533	0.00667	0.00667	**0**	**0.689**	0.736	0.748	0.752
Bomb	0.392	0.317	0.225	**0.129**	**0.867**	0.897	0.909	0.905
Lesmis	0.276	0.224	0.156	**0.0680**	**0.859**	0.894	0.889	0.889
Throne	0.0886	0.0771	**0.0314**	0.0343	**0.816**	0.884	0.879	0.869
Jazz	**0.401**	0.434	0.429	0.425	**0.951**	0.959	0.957	0.955

TABLE 4
Parameters settings for evolutionary algorithms, including GA and EDA, in each network.

Item	Meaning	Value
α	weight in fitness	-
m	number of deleted/inserted links	-
$n_{iteration}$	number of iterations	1000
e_{elite}	number of retained elites	10
$n_{crossover}$	number of chromosomes for crossover	50
$n_{mutation}$	number of chromosomes for mutation	50
pc	crossover rate	0.7
pm	mutation rate	0.1
$n_{estimation}$	number of chromosomes for estimation	250
n_{eda}	number of generated population in EDA	50

conducting link swapping once means deleting two links and adding two other links at the same time. So, one may conduct half times of link swapping comparing to RLR. In HP, similarly conduct one hundred times and then obtain the average precision and AUC. In GA and EDA, the main parameter to be tuned is the α of the fitness, and the results are shown in TABLE 3 and other parameters are set to be the empirical values, as shown in TABLE 4. It can be seen that α value will influence the precision and AUC of the perturbed network. When α increases, the precision values of most perturbed networks tend to decrease and when α equals to 0, the AUC values of all perturbed networks are the lowest. Since there does not exist any fixed α that is optimal for all networks, one may select the value of α separately for each individual network. In the experiments, for Mexican, Dolphin, Lesmis and Throne, set $\alpha = 0.01$; for Bomb, set $\alpha = 1$; for Jazz, set $\alpha = 0$. Then, repeat five times in each training set and calculate the average precision and AUC of the optimal individuals in the final population. For further study, our source codes are available online. [1]

The final results are shown in Fig. 8 and Fig. 9, from which it is found that evolutionary perturbations, especially those obtained by EDA, are superior to RLR, RLS and HP on most networks. The experimental results also show that the effect of HP is getting better as the proportion of perturbations increases. Especially, in a larger scale network with larger perturbation, HP even outperforms evolutionary methods when measured in precision and it is well expected because the design of HP is favorable to reduce the precision. Besides, we have calculated the the precision of the top-k ranked links. In the experiments, for Mexican, Dolphin, Bomb, Lesmis and Throne, we set k equal to 10; for Jazz, we set k equal to 100. The final results are consistent with the conclusion when measured in precision. [2]

Moreover, instead of randomly choosing links as sensitive ones, we conduct a special case in the multi-layer network, i.e. CS, which contains different kinds of relationships. We label the co-authorship as sensitive and delete them from the original network. Then we use each method to perturb the network. From TABLE 5 we can see that EDA still outperforms the baseline methods in terms of AUC while the precision is relatively low at the same time.

To test the performance in larger networks, we complement the experiments in Email and Geom. We randomly sample 100 links as sensitive ones and perturb the remaining networks with the fixed-size perturbation (deleting 100 links and inserting 100 links) by each baseline method. We set $\alpha = 0$ and other parameters remain the same as above. The results are shown in TABLE 5, where we can find that EDA can always achieve the lowest AUC with relatively low precision while HP can always achieve the lowest precision.

Larger-scale perturbations in larger-scale networks mean that the search space are larger. Correspondingly, increasing the scale of the evolutionary method, e.g. increasing the numbers of individuals and iterations, may yield better results. However, time consumption can also exceed the allowable tolerance. Clearly, improving evolutionary methods or introducing parallel computing may reduce time consumption, which will be left for future studies.

Last but not the least, the running time before and after accelerating the fitness calculation has also been compared. In Fig. 10, experiments on GA and EDA with the same parameters in the two different networks, i.e. Lesmis and Jazz, indicate that the computational efficiency is increased significantly after acceleration.

1. https://github.com/Zhaominghao1314

2. Due to size limitation, the result tables are posted online together with the source codes.

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2933833, IEEE Transactions on Knowledge and Data Engineering

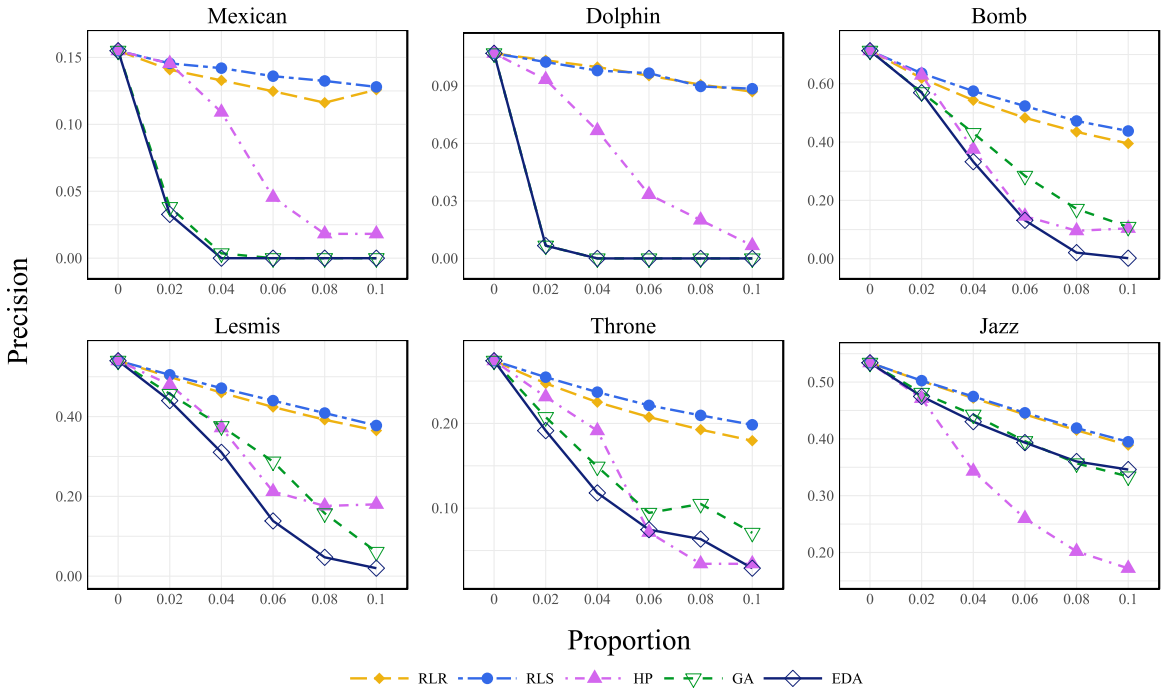


Fig. 8. Average precision of each network perturbed by different methods with increasing proportions of deleted and inserted links.

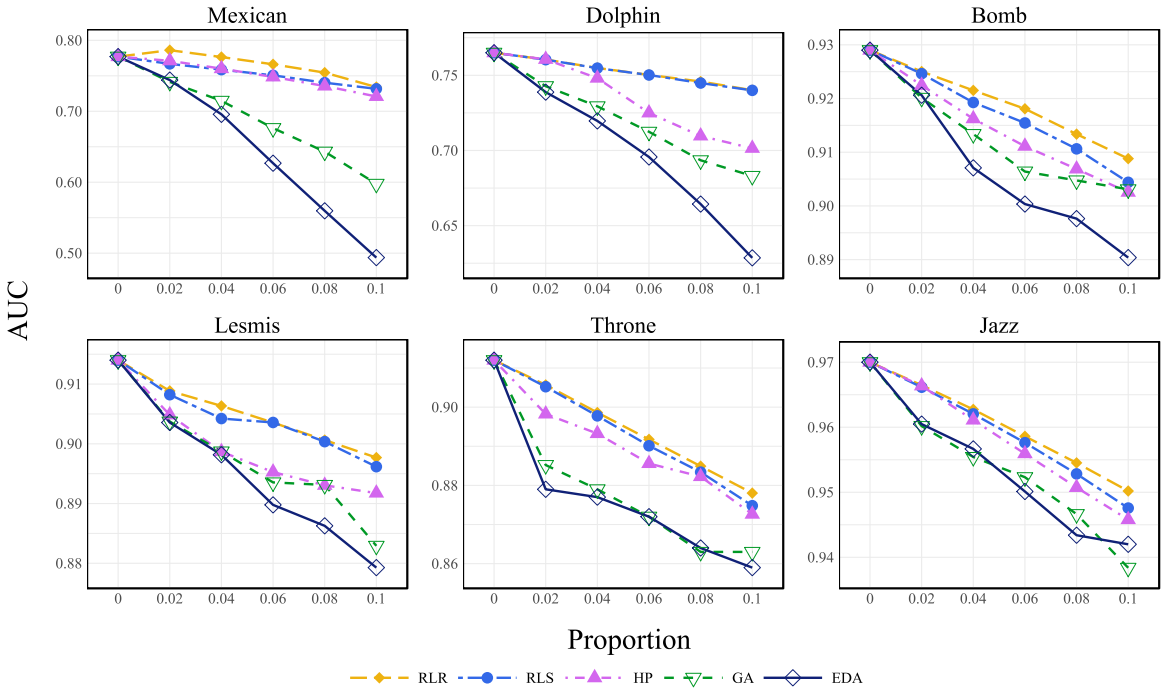


Fig. 9. Average AUC of each network perturbed by different methods with increasing proportions of deleted and inserted links.

TABLE 5
The precision and AUC of the networks perturbed by each method in three networks, i.e. CS, Email and Geom. HP (RA) means HP is based on RA and EDA (RA) means that the fitness of EDA is based on RA as well.

RA	Precision					AUC				
	original	RLR	RLS	HP (RA)	EDA (RA)	original	RLR	RLS	HP (RA)	EDA (RA)
CS	0.381	0.239	0.253	**0**	0.0952	0.929	0.902	0.894	0.923	**0.763**
Email	0.0590	0.0524	0.0568	**0.0120**	0.0320	0.872	0.869	0.869	0.864	**0.855**
Geom	0.297	0.272	0.274	**0.00600**	0.233	0.973	0.972	0.971	0.969	**0.940**

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2933833, IEEE Transactions on Knowledge and Data Engineering

TABLE 6
Transferability of perturbations generated by each method measured by different link prediction methods in six networks.

Mexican	Precision					AUC				
	original	RLR	RLS	HP(RA)	EDA(RA)	original	RLR	RLS	HP(RA)	EDA(RA)
RA	0.155	0.125	0.128	0.0182	**0**	0.777	0.734	0.732	0.721	**0.495**
CN	0.118	0.0993	0.106	0.0273	**0**	0.760	0.720	0.715	0.702	**0.516**
Jaccard	0.109	0.0876	0.0796	0.0273	**0**	0.752	0.709	0.701	0.693	**0.496**
PA	0.0546	0.0677	0.0596	0.0364	**0.0300**	0.625	0.616	0.625	0.588	**0.557**
AA	0.127	0.120	0.119	0.0182	**0**	0.776	0.733	0.731	0.721	**0.504**
LP($\alpha = 0.5$)	0.127	0.0896	0.105	0.0545	**0.0165**	0.729	0.692	0.682	0.684	**0.564**
DeepWalk	0.0455	0.0534	0.0469	0.0364	**0.0209**	0.742	0.687	0.650	0.678	**0.556**
HOPE	0.0818	0.0864	0.0631	0.0182	**0.00193**	0.738	0.717	0.698	0.681	**0.494**
Dolphin	Precision					AUC				
	original	RLR	RLS	HP(RA)	EDA(RA)	original	RLR	RLS	HP(RA)	EDA(RA)
RA	0.107	0.0865	0.0885	0.00667	**0**	0.765	0.740	0.740	0.701	**0.629**
CN	0.113	0.0953	0.0954	0.0333	**0**	0.760	0.736	0.736	0.700	**0.648**
Jaccard	0.113	0.0920	0.0973	0.0200	**0.00455**	0.760	0.737	0.736	0.699	**0.651**
PA	0.0133	0.0121	0.0132	**0**	**0**	0.623	0.616	0.623	0.589	**0.583**
AA	0.133	0.0978	0.0999	0.0133	**0**	0.765	0.741	0.741	0.702	**0.632**
LP($\alpha = 0.5$)	0.120	0.107	0.103	0.0533	**0.0226**	0.792	0.768	0.762	0.772	**0.711**
DeepWalk	0.0467	0.0454	0.0403	0.0200	**0.0131**	0.759	0.750	0.728	0.741	**0.676**
HOPE	0.0533	0.0527	0.0435	0.00667	**0.00326**	0.742	0.729	0.723	0.689	**0.641**
Bomb	Precision					AUC				
	original	RLR	RLS	HP(RA)	EDA(RA)	original	RLR	RLS	HP(RA)	EDA(RA)
RA	0.713	0.395	0.438	0.104	**0.00182**	0.929	0.909	0.904	0.903	**0.891**
CN	0.571	0.452	0.483	**0.271**	0.317	0.915	0.897	0.888	0.891	**0.883**
Jaccard	0.475	0.359	0.327	0.292	**0.274**	0.912	0.894	0.886	0.876	**0.874**
PA	0.229	0.186	0.219	**0.138**	0.139	0.774	0.767	0.774	0.749	**0.744**
AA	0.658	0.459	0.498	0.200	**0.197**	0.926	0.907	0.901	0.902	**0.891**
LP($\alpha = 0.5$)	0.488	0.379	0.389	0.404	**0.325**	0.880	0.864	**0.848**	0.868	0.851
DeepWalk	0.192	0.144	**0.118**	0.175	0.133	0.865	0.855	**0.826**	0.842	0.827
HOPE	0.354	0.274	0.231	0.250	**0.216**	0.911	0.890	0.877	**0.855**	0.861
Lesmis	Precision					AUC				
	original	RLR	RLS	HP(RA)	EDA(RA)	original	RLR	RLS	HP(RA)	EDA(RA)
RA	0.540	0.365	0.378	0.180	**0.0199**	0.914	0.898	0.896	0.891	**0.879**
CN	0.484	0.359	0.357	0.204	**0.182**	0.906	0.895	0.888	**0.881**	**0.881**
Jaccard	0.0360	0.202	**0.0606**	0.0920	0.114	0.914	0.870	0.859	0.854	**0.850**
PA	0.104	0.0936	0.102	0.0800	**0.0768**	0.782	0.780	0.782	**0.768**	0.777
AA	0.524	0.380	0.378	0.192	**0.111**	0.912	0.898	0.893	0.888	**0.886**
LP($\alpha = 0.5$)	0.376	0.311	0.300	0.316	**0.236**	0.875	0.871	**0.856**	0.867	0.869
DeepWalk	0.156	0.130	**0.0761**	0.136	0.136	0.831	0.823	**0.806**	0.814	0.815
HOPE	0.0280	0.0766	**0.0213**	0.0720	0.0658	0.875	0.861	0.852	0.843	**0.839**
Throne	Precision					AUC				
	original	RLR	RLS	HP(RA)	EDA(RA)	original	RLR	RLS	HP(RA)	EDA(RA)
RA	0.274	0.180	0.198	0.0343	**0.0291**	0.912	0.878	0.875	0.873	**0.859**
CN	0.200	0.174	0.181	0.109	**0.103**	0.892	0.865	0.859	0.849	**0.847**
Jaccard	0.0600	0.0553	0.0414	0.0543	**0.0357**	0.861	0.837	0.825	**0.813**	**0.813**
PA	0.123	0.110	0.122	0.100	**0.0929**	0.769	0.767	0.769	0.764	**0.754**
AA	0.240	0.192	0.203	**0.0486**	0.0814	0.908	0.878	0.873	0.869	**0.859**
LP($\alpha = 0.5$)	0.163	0.145	0.157	0.117	**0.116**	0.867	0.852	**0.838**	0.860	0.845
DeepWalk	0.0457	0.0314	**0.0262**	0.0400	**0.0262**	0.839	0.821	0.795	0.814	**0.794**
HOPE	0.0314	0.0299	0.0211	0.0286	**0.0209**	0.850	0.832	0.820	**0.801**	**0.801**
Jazz	Precision					AUC				
	original	RLR	RLS	HP(RA)	EDA(RA)	original	RLR	RLS	HP(RA)	EDA(RA)
RA	0.534	0.390	0.395	**0.172**	0.346	0.970	0.950	0.948	0.946	**0.942**
CN	0.497	0.382	0.378	**0.237**	0.346	0.954	0.938	**0.930**	0.933	0.931
Jaccard	0.512	0.390	0.398	**0.275**	0.359	0.960	0.945	0.942	**0.926**	0.939
PA	0.132	0.121	0.130	**0.106**	0.113	0.770	0.761	0.769	**0.746**	0.754
AA	0.518	0.391	0.391	**0.220**	0.353	0.961	0.944	0.938	0.939	**0.937**
LP($\alpha = 0.5$)	0.359	0.295	0.281	**0.265**	0.272	0.908	0.889	**0.874**	0.902	0.883
DeepWalk	0.307	0.264	0.226	**0.210**	0.275	0.923	0.914	0.907	**0.903**	0.908
HOPE	0.273	0.255	0.235	**0.157**	0.229	0.936	0.919	0.919	**0.887**	0.913

4.2.2 *Transferability of Different Link Perturbations*

Although the above algorithms defend against specific link prediction attacks well, i.e., RA attack, an adversary may use other methods to do link prediction. Therefore, it is desirable to know the transferability of the evolutionary perturbation especially acquired from EDA by checking the performance of other link prediction algorithms. For this

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2933833, IEEE Transactions on Knowledge and Data Engineering

TABLE 7
Transferability of perturbations generated by each method measured by different link prediction methods in three networks, i.e. CS, Email and Geom.

CS	Precision					AUC				
	original	RLR	RLS	HP(RA)	EDA(RA)	original	RLR	RLS	HP(RA)	EDA(RA)
CN	0.190	0.154	0.167	0.0952	**0.0635**	0.896	0.875	0.861	0.896	**0.764**
Jaccard	0.190	0.208	0.218	0.143	**0.0952**	0.939	0.916	0.907	0.930	**0.814**
PA	0	0.00300	**0**	**0**	**0**	0.605	0.596	0.605	0.617	**0.536**
AA	0.238	0.181	0.182	**0.0476**	0.0794	0.913	0.891	0.880	0.910	**0.768**
LP($\alpha = 0.5$)	0.0476	0.0776	0.0581	0.143	**0.0317**	0.790	0.776	0.755	0.787	**0.693**
DeepWalk	0.190	0.173	0.145	0.190	**0.143**	0.923	0.911	0.902	0.913	**0.825**
HOPE	0.143	0.184	0.173	0.190	**0.0952**	0.960	0.939	0.935	0.950	**0.840**
Email	Precision					AUC				
	original	RLR	RLS	HP(RA)	EDA(RA)	original	RLR	RLS	HP(RA)	EDA(RA)
CN	0.0550	0.0520	0.0521	**0.0320**	0.0440	0.871	0.866	0.867	0.862	**0.854**
Jaccard	0.00900	0.00620	0.00620	0.00700	**0.00533**	0.868	0.862	0.864	0.859	**0.851**
PA	0.00300	**0.00200**	0.00300	0.00700	0.00300	0.812	0.810	0.812	0.811	**0.807**
AA	0.0580	0.0549	0.0563	**0.0200**	0.0427	0.873	0.869	0.869	0.864	**0.856**
LP($\alpha = 0.5$)	0.0310	0.0298	0.0296	**0.0240**	0.0287	0.922	0.919	0.920	0.921	**0.916**
DeepWalk	0.0140	0.0109	0.0118	0.0140	**0.0103**	0.861	0.863	**0.858**	0.863	**0.858**
HOPE	0.00200	0.00190	0.00130	0.00200	**0.00100**	0.878	0.875	0.875	0.871	**0.869**
Geom	Precision					AUC				
	original	RLR	RLS	HP(RA)	EDA(RA)	original	RLR	RLS	HP(RA)	EDA(RA)
CN	0.0820	0.0811	0.0813	**0.0730**	0.0807	0.972	0.971	0.970	0.968	**0.939**
Jaccard	0.0190	0.0167	0.0179	0.0220	**0.0131**	0.972	0.971	0.970	0.968	**0.939**
PA	0.0150	**0.0130**	0.0150	0.0150	0.0145	0.783	0.783	0.783	0.785	**0.766**
AA	0.149	0.134	0.137	**0.0580**	0.122	0.973	0.971	0.970	0.970	**0.940**
LP($\alpha = 0.5$)	0.0490	0.0475	0.0488	**0.0450**	0.0483	0.975	0.974	0.973	0.975	**0.944**
DeepWalk	0.0850	0.0724	0.0708	0.0820	**0.0441**	0.968	0.968	0.967	0.967	**0.928**
HOPE	0.0130	0.0103	0.0109	0.0160	**0.00931**	0.897	0.905	0.899	0.893	**0.883**

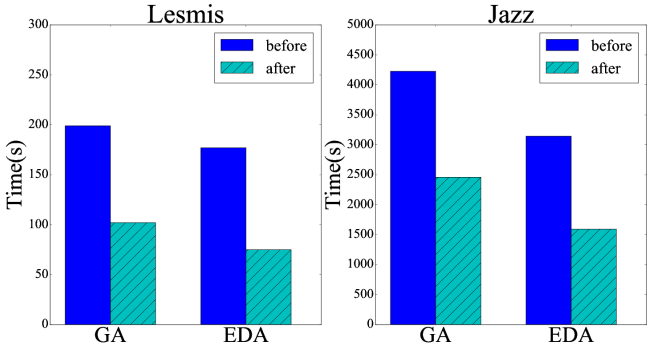


Fig. 10. Average running time of GA and EDA with the same parameters in two networks before and after accelerating the fitness calculation.

purpose, the final perturbed networks obtained with the maximum perturbation (proportion equals to 0.1) generated by EDA together with HP are retained for each network, for which five different similarity indices and two new method of network embedding are selected as test indices, i.e., Common Neighbors (CN), Jaccard, Preferential Attachment (PA), Adamic-Adar (AA) and Local Path (LP) [19], DeepWalk [20] and HOPE [37], to calculate the average precision and AUC of the perturbed networks. To be specific, we use the cosine similarity of node embedding vectors trained from DeepWalk and HOPE as the predictor. For Mexican, Dolphin, Bomb, Lesmis, Throne, Jazz and CS, we set the dimension of node embedding vectors as 32; for Email and Geom, we set it as 128. The remaining parameters are set as defaults in the open-source codes [3].

The results are summarized in TABLE 6 and TABLE 7. One can see that the transferability of EDA is better, especially when measured by AUC, comparing to the random perturbations (RLR/RLS) and heuristic perturbation (HP), in most networks. However, it can also be found that, in some cases, the transferability of EDA is inferior even than RLS when checked by LP and DeepWalk, which may be due to the fact that both LP and DeepWalk consider the higher-order similarity of node pairs while the fitness of EDA only considers first-order similarity within one-hop neighbors, which limits its transferability of defensive effect.

4.2.3 *Topological Features of The Perturbed Networks*

Finally, we calculate some topological features of the networks perturbed by each method to see how those methods influence the network structure. The results are shown in TABLE 8. The values of topological features after perturbation nearest to those of original network is shown in bold and underline. To be specific, $\lambda_{\max}$ denotes the largest spectrum of the adjacency matrix. $\langle C_D \rangle$, $\langle C_B \rangle$ and $\langle C_C \rangle$ are the average degree centrality, betweenness centrality and closeness centrality, respectively. We can find that none of the compared methods changes the mentioned topological features too much while RLS behaves the best (it leads to the smallest changes of network properties).

5 Conclusions and Research Outlook

In this paper, a target defense method against link prediction attacks, e.g., RA attack, is proposed for social networks. Both heuristic and evolutionary perturbations techniques are used, taking into account both defensive effect and the size of perturbation. A special fitness function is designed, which considers two evaluation indices of link

3. https://github.com/thunlp/openne

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2933833, IEEE Transactions on Knowledge and Data Engineering

TABLE 8
Six topological features of the nine original networks and perturbed networks.

Mexican	original	RLR	RLS	HP(RA)	EDA(RA)
$\langle k \rangle$	6.057	**6.057**	**6.057**	6.063	**6.057**
C	0.406	**0.341**	0.340	0.241	0.318
$\lambda_{\max}$	7.518	7.368	**7.505**	6.875	7.352
$\langle C_D \rangle$	0.178	**0.178**	**0.178**	**0.178**	**0.178**
$\langle C_B \rangle$	0.037	**0.036**	0.035	**0.036**	0.034
$\langle C_C \rangle$	0.462	0.466	0.473	**0.464**	0.477
Dolphin	original	RLR	RLS	HP(RA)	EDA(RA)
$\langle k \rangle$	4.723	4.698	**4.723**	**4.723**	4.721
C	0.237	**0.208**	0.201	0.143	0.181
$\lambda_{\max}$	6.626	6.412	**6.546**	6.069	6.436
$\langle C_D \rangle$	0.0788	0.0779	**0.0788**	**0.0788**	0.0787
$\langle C_B \rangle$	0.0418	0.0362	0.0354	**0.0400**	0.0342
$\langle C_C \rangle$	0.298	0.325	0.331	**0.306**	0.340
Bomb	original	RLR	RLS	HP(RA)	EDA(RA)
$\langle k \rangle$	6.932	6.879	**6.932**	**6.932**	6.943
C	0.551	0.418	**0.425**	0.349	0.389
$\lambda_{\max}$	12.300	11.396	**12.015**	10.532	10.989
$\langle C_D \rangle$	0.111	0.110	**0.111**	**0.111**	0.112
$\langle C_B \rangle$	0.029	0.026	0.026	**0.028**	0.027
$\langle C_C \rangle$	0.374	0.392	0.402	**0.383**	0.390
Lesmis	original	RLR	RLS	HP(RA)	EDA(RA)
$\langle k \rangle$	6.092	6.040	**6.092**	**6.092**	6.078
C	0.512	0.400	**0.406**	0.358	0.380
$\lambda_{\max}$	10.925	10.307	**10.747**	9.878	9.997
$\langle C_D \rangle$	0.0821	0.0807	**0.0821**	**0.0821**	0.0818
$\langle C_B \rangle$	0.0235	**0.0234**	0.0225	0.0245	0.0232
$\langle C_C \rangle$	0.375	**0.374**	0.385	0.365	0.378
Throne	original	RLR	RLS	HP(RA)	EDA(RA)
$\langle k \rangle$	6.016	5.963	**6.016**	**6.016**	6.037
C	0.487	0.352	**0.380**	0.253	0.354
$\lambda_{\max}$	12.028	11.139	**11.856**	10.212	11.104
$\langle C_D \rangle$	0.0576	0.0566	**0.0576**	**0.0576**	0.0580
$\langle C_B \rangle$	0.0193	0.0186	0.0180	**0.0191**	0.0183
$\langle C_C \rangle$	0.344	0.348	0.360	**0.346**	0.355
Jazz	original	RLR	RLS	HP(RA)	EDA(RA)
$\langle k \rangle$	24.993	24.930	**24.993**	**24.993**	24.942
C	0.555	0.398	**0.428**	0.372	0.395
$\lambda_{\max}$	36.156	33.781	**35.618**	31.364	33.865
$\langle C_D \rangle$	0.127	**0.127**	**0.127**	**0.127**	**0.127**
$\langle C_B \rangle_{(\times 10^{-3})}$	6.563	5.671	5.825	**5.991**	5.585
$\langle C_C \rangle$	0.447	0.478	0.475	**0.465**	0.483
Email	original	RLR	RLS	HP(RA)	EDA(RA)
$\langle k \rangle$	9.446	9.445	**9.446**	**9.446**	9.462
C	0.215	0.198	**0.204**	0.166	0.200
$\lambda_{\max}$	20.395	20.076	**20.306**	19.558	20.136
$\langle C_D \rangle_{(\times 10^{-3})}$	8.344	8.343	**8.344**	**8.344**	8.372
$\langle C_B \rangle_{(\times 10^{-3})}$	2.321	**2.307**	2.299	2.304	2.304
$\langle C_C \rangle$	0.280	**0.281**	0.282	0.282	0.282
Geom	original	RLR	RLS	HP(RA)	EDA(RA)
$\langle k \rangle$	3.832	**3.832**	**3.832**	**3.832**	3.842
C	0.477	0.460	**0.465**	0.451	0.463
$\lambda_{\max}$	28.747	28.495	**28.599**	27.730	28.550
$\langle C_D \rangle_{(\times 10^{-4})}$	6.223	**6.223**	**6.223**	**6.223**	6.257
$\langle C_B \rangle_{(\times 10^{-4})}$	2.427	2.819	2.543	**2.432**	2.810
$\langle C_C \rangle$	0.0671	0.0713	0.0702	**0.0670**	0.0718
CS	original	RLR	RLS	HP(RA)	EDA(RA)
$\langle k \rangle$	10.885	**10.885**	**10.885**	**10.885**	**10.885**
C	0.558	0.460	**0.481**	0.426	0.466
$\lambda_{\max}$	14.267	13.842	**14.211**	13.172	13.881
$\langle C_D \rangle$	0.181	**0.181**	**0.181**	**0.181**	**0.181**
$\langle C_B \rangle$	0.0187	**0.0181**	**0.0181**	0.0180	0.0179
$\langle C_C \rangle$	0.485	**0.491**	0.492	**0.491**	0.494

prediction, i.e. precision and AUC, and can reduce computation by incrementally updating the fitness due to locality of changes after perturbations. The experimental results on nine real-world networks show that evolutionary perturbations, especially those obtained by EDA, outperform other baseline methods in most networks and HP can even outperform EDA in large networks with large perturbations when measured in precision. Finally, the transferability of evolutionary perturbations generated by EDA is verified, showing that the defensive effect of perturbation generated by EDA which is designed to defend against RA attack is transferable in most networks and can even be used to defend against other link prediction attacks which are based on high order similarity between pairwise nodes. Last but not the least, we find that the changes of some network properties after perturbations are very small while RLR is relatively better.

For large-scale networks, especially when the number of links to be hidden is very large, the size of perturbation needs to rise accordingly. However, the proposed algorithm does not take advantage of the network structure, consequently the evolutionary efficiency is limited. Therefore, improving evolutionary methods or introducing parallel computing so as to reduce computation time are good topics for future studies. Further, as many state-of-the-art link prediction methods are proposed constantly, e.g. neural network based, it is also interesting to explore how to defend against such methods.

ACKNOWLEDGMENT

This work was partially supported by the National Natural Science Foundation of China (61572439, 11505153) and by the Hong Kong Research Grants Council under the GRF Grant CityU11200317.

REFERENCES

[1] D. J. Watts and S. H. Strogatz, "Collective dynamics of small-world networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.
[2] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.
[3] S. H. Strogatz, "Exploring complex networks," *Nature*, vol. 410, no. 6825, pp. 268–276, 2001.
[4] M. E. J. Newman, "The structure and function of complex networks," *SIAM Review*, vol. 45, no. 2, pp. 167–256, 2003.
[5] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D. U. Hwang, "Complex networks: Structure and dynamics," *Physics Reports*, vol. 424, no. 4-5, pp. 175–308, 2006.
[6] K. Liu and E. Terzi, "Towards identity anonymization on graphs," in *ACM SIGMOD International Conference on Management of Data*, 2008, pp. 93–106.
[7] J. Cheng, W. C. Fu, and J. Liu, "K-isomorphism:privacy preserving network publication against structural attacks," in *ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June*, 2010, pp. 459–470.
[8] C. H. Tai, P. S. Yu, D. N. Yang, and M. S. Chen, "Privacy-preserving social network publication against friendship attacks," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011, pp. 1262–1270.
[9] E. Zheleva and L. Getoor, "Preserving the privacy of sensitive relationships in graph data," in *Privacy, Security, and Trust in KDD*. Springer, 2008, pp. 153–171.
[10] L. Backstrom and J. Kleinberg, "Romantic partnerships and the dispersion of social ties: a network analysis of relationship status on facebook," in *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*. ACM, 2014, pp. 831–841.

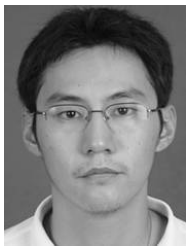
1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2933833, IEEE Transactions on Knowledge and Data Engineering

[11] M. Hay, G. Miklau, D. Jensen, P. Weis, and S. Srivastava, "Anonymizing social networks," *Computer Science Department Faculty Publication Series*, p. 180, 2007.
[12] X. Ying and X. Wu, "Randomizing social networks: a spectrum preserving approach," in *SIAM International Conference on Data Mining, SDM 2008, April 24-26, 2008, Atlanta, Georgia, Usa*, 2008, pp. 739–750.
[13] ——, "On link privacy in randomizing social networks," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2009, pp. 28–39.
[14] A. M. Fard, K. Wang, and P. S. Yu, "Limiting link disclosure in social network analysis through subgraph-wise perturbation," in *Proceedings of the 15th International Conference on Extending Database Technology*. ACM, 2012, pp. 109–119.
[15] L. D and K. J, "The linkprediction problem for social networks," *Journal of the American society for information science and technology*, vol. 58, no. 7, pp. 1019–1031, 2007.
[16] M. Fire, G. Katz, L. Rokach, and Y. Elovici, "Links reconstruction attack," in *Security and Privacy in Social Networks*. Springer, 2013, pp. 181–196.
[17] A. Rapoport, "Spread of information through a population with socio-structural bias: I. assumption of transitivity," *The Bulletin of Mathematical Biophysics*, vol. 15, no. 4, pp. 523–533, 1953.
[18] T. Zhou, L. Lü, and Y.-C. Zhang, "Predicting missing links via local information," *The European Physical Journal B-Condensed Matter and Complex Systems*, vol. 71, no. 4, pp. 623–630, 2009.
[19] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: Statistical Mechanics and its Applications*, vol. 390, no. 6, pp. 1150–1170, 2011.
[20] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2014, pp. 701–710.
[21] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 855–864.
[22] 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.
[23] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.
[24] 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 & Data Engineering*, vol. PP, no. 99, pp. 1–1, 2018.
[25] Q. Xuan, Y. Li, and T. J. Wu, "Optimal symmetric networks in terms of minimizing average shortest path length and their sub-optimal growth model," *Physica A Statistical Mechanics & Its Applications*, vol. 388, no. 7, pp. 1257–1267, 2009.
[26] M. M, *An introduction to genetic algorithms*. MIT press, 1998.
[27] M. N. Zlochin M, Birattari M and D. M, "Model-based search for combinatorial optimization: A critical survey," *Annals of Operations Research*, vol. 131, no. 3-4, pp. 373–395, 2004.
[28] J. Gil-Mendieta and S. Schmidt, "The political network in mexico ," *Social Networks*, vol. 18, no. 18, pp. 355–381, 1996.
[29] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, "The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations," *Behavioral Ecology & Sociobiology*, vol. 54, no. 4, pp. 396–405, 2003.
[30] B. Hayes, "Computing science: Connecting the dots," *American Scientist*, vol. 94, no. 5, pp. 400–404, 2006.
[31] D. E. Knuth, *The Stanford GraphBase: a platform for combinatorial computing*. Addison-Wesley Reading, 1993, vol. 37.
[32] A. Beveridge and J. Shan, "Network of thrones," *Math Horizons*, vol. 23, no. 4, pp. 18–22, 2016.
[33] P. M. Gleiser and L. Danon, "Community structure in jazz," *Advances in complex systems*, vol. 6, no. 04, pp. 565–573, 2003.
[34] M. Magnani, B. Micenkova, and L. Rossi, "Combinatorial analysis of multiple networks," *arXiv preprint arXiv:1303.4986*, 2013.
[35] R. Guimera, L. Danon, A. Diaz-Guilera, F. Giralt, and A. Arenas, "Self-similar community structure in a network of human interactions," *Physical review E*, vol. 68, no. 6, p. 065103, 2003.
[36] V. Batagelj and A. Mrvar, "Pajek datasets [ol]," *http://vlado.fmf.uni-lj.si/pub/networks/data/*, 2006.
[37] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 1105–1114.

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

Minghao Zhao received the BS degree from Wuhan University of Technology, China, in 2014. He is working toward the MS degree in the School of Information Engineering, Zhejiang University of Technology. His research interests include social networks analysis and machine learning.



Chenbo Fu received the BS degree in Physics from Zhejiang University of Technology in 2003, MS and PhD Degrees in Physics from Zhejiang University, in 2009 and 2013, respectively. He was a postdoctoral researcher in the School of Information Engineering, Zhejiang University of Technology and was a research assistant in the Department of Computer Science, University of California at Davis in 2014. Currently, he is a lecturer in the School of Information Engineering, Zhejiang University of Technology. His research interests include network algorithms design, social networks data mining, chaos synchronization, network dynamics and machine learning.

Jun Zheng received BS degree from YangZhou University , China, in 2017. He is working toward the MS degree in the School of Information Engineering, Zhejiang University of Technology. His research interests include data mining, social network analysis, network optimization and cyberspace security research.

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2933833, IEEE Transactions on Knowledge and Data Engineering

Huimin Huang is currently working toward the BS degree in automation in the School of Information Engineering at Zhejiang University of Technology, China. Her current research interests include social networks data mining, computer vision, machine learning and deep learning.

Xincheng Shu received the BS degree from Zhejiang University of Technology in 2017. He is working toward the MS degree in the School of Information Engineering, Zhejiang University of Technology. His research interests include information diffusion and machine learning.

Qi Xuan received BS and PhD degrees in control theory and engineering from Zhejiang University, Hangzhou, China, in 2003 and 2008, respectively. He was a Postdooctoral Researcher with the Department of Information Science and Electronic Engineering, Zhejiang University, from 2008 to 2010, and a Research Assistant with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong, China, in 2010 and also in 2017. From 2012 to 2014, he was a Postdoctoral Researcher with the Department of Computer Science, University of California at Davis, Davis, CA, USA. He is a member of the IEEE and is currently a Professor with the School of Information Engineering, Zhejiang University of Technology. His current research interests include network-based algorithms design, social networks data mining, social synchronization and consensus, reaction-diffusion network dynamics, machine learning, and computer vision.

Guanrong Chen (M'89-SM'92-F'97) 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. He has been a Chair Professor and the Founding Director of the Centre for Chaos and Complex Networks at the City University of Hong Kong since year 2000, prior to that he was a tenured Full Professor at the University of Houston, Texas, USA. He was awarded the 2011 Euler Gold Medal, Russia, and conferred Honorary Doctorate by the Saint Petersburg State University, Russia in 2011 and by the University of Le Havre, Romandie, France in 2014. He is a Member of the Academia of Europe and a Fellow of The World Academy of Sciences, and is a Highly Cited Researcher in Engineering as well as in Mathematics according to Thomson Reuters.

1041-4347 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.