Hindawi
Security and Communication Networks
Volume 2021, Article ID 8609278, 11 pages
https://doi.org/10.1155/2021/8609278

Research Article

GNS: Forge High Anonymity Graph by Nonlinear Scaling Spectrum

Yong Zeng, Yixin Li, Zhongyuan Jiang, and Jianfeng Ma

School of Cyber Engineering, Xidian University, Xian 710126, China

Correspondence should be addressed to Yong Zeng; yzeng@mail.xidian.edu.cn

Received 7 May 2021; Accepted 15 September 2021; Published 30 September 2021

Academic Editor: Qing Yang

It is crucial to generate random graphs with specific structural properties from real graphs, which could anonymize graphs or generate targeted graph data sets. The state-of-the-art method called spectral graph forge (SGF) was proposed at INFOCOM 2018. This method uses a low-rank approximation of the matrix by throwing away some spectrums, which provides privacy protection after distributing graphs while ensuring data availability to a certain extent. As shown in SGF, it needs to discard at least 20% spectrum to defend against deanonymous attacks. However, the data availability will be significantly decreased after more spectrum discarding. Thus, is there a way to generate a graph that guarantees maximum spectrum and anonymity at the same time? To solve this problem, this paper proposes graph nonlinear scaling (GNS). We firmly prove that GNS can preserve all eigenvectors meanwhile providing high anonymity for the forged graph. Precisely, the GNS scales the eigenvalues of the original spectrum and constructs the forged graph with scaled eigenvalues and original eigenvectors. This approach maximizes the preservation of spectrum information to guarantee data availability. Meanwhile, it provides high robustness towards deanonymous attacks. The experimental results show that when SGF discards only 10% of the spectrum, the forged graph has high data availability. At this time, if the distance vector deanonymity algorithm is used to attack the forged graph, almost 100% of the nodes can be identified, while when achieving the same availability, only about 20% of the nodes in the forged graph obtained from GNS can be identified. Moreover, our method is better than SGF in capturing the real graph's structure in terms of modularity, the number of partitions, and average clustering.

1. Introduction

In recent years, the security and privacy of data have received extensive attention. How to protect data privacy while maintaining data availability is a hot research topic. Xiong et al. [1] propose a privacy and availability data-clustering scheme based on the k-means algorithm and differential privacy. Aldossary and Allen [2] discuss the privacy and availability of data in cloud storage and analyzes the current risks and feasible solutions. However, these methods have only been studied for data in Euclidean space. For non-Euclidean data, such as graph data, Day et al. [3] proposed a graph differential privacy method that some nodes or edges can be removed to provide the security of the query. According to [4], the deletion of a node may cause the cascading failure of multiple nodes, thus affecting the structural characteristics of the network. Pu et al. [5] also mentioned that the deletion of edges would weaken the effectiveness of the graph data for downstream tasks. Therefore, some scholars have proposed that forge random graphs with specific properties to protect the privacy of graph data. This provides structural features of graph data while protecting its privacy. This method is widely used in the following ways: privacy-preserving social network publishing, generating specific data sets for model training, and relationship anonymity. The realization of these purposes requires that the generated graph have certain anonymity while meeting some structural metrics, maintaining the structural features.

There are many previous works done in this area, Milo et al. [6] generated a graph that only satisfies a given degree sequence. Calvert et al. [7] proposed two methods to generate networks with targeted topology. They mainly focus on the aspect of hierarchy and locality present in the network. In [8], an algorithm that generates the synthetic graph with a target joint degree matrix is proposed. In INFOCOM 2018, Baldesi et al. [9] propose an algorithm called spectral graph forge (SGF) for generating random graphs that preserves the community structure from a real network of interest. It uses a low-rank approximation of the modularity matrix to generate forged graphs that match the target modularity within the user-selectable degree of accuracy. The modularity is also known as an essential metric of community structure, used for community detection and network analysis. As compared with two excellent previous works [10, 11], the SGF shows better performance in many metrics: modularity, partition number, degree sequence, and average clustering.

However, the low-rank approximation module in the SGF algorithm focuses more on the principal components. Its parameter α selected by the user determine how much the spectrum will be used. The closer α is to 1; the less spectrum will be used. This kind of method improves the algorithm's efficiency while keeping the general features of the original graph. Nevertheless, according to reference [12], even when a suitable local scaling affinity matrix is given, clustering cannot be obtained by the first kth eigenvectors of the matrix. Moreover, for a sparse graph, eigenvectors are more important when clustering [13]. It also can be seen from the experimental results in SGF that the clustering of the real graph can be well preserved only when α is above 0.9, that is, using more than 90% spectrum. Therefore, we can naturally find that more spectrum is needed to meet high data availability.

Meanwhile, the anonymity of the SGF algorithm decreases with the increase of α. As we analyzed, every component of the spectrum is useful. Reducing the α will inevitably lead to the loss of information. In other words, the SGF algorithm needs to make a trade-off between performance and anonymity. Therefore, is there an algorithm to maintain the original spectrum as much as possible while also providing high anonymity? Solving this problem has become the motivation of this paper.

Therefore, we propose graph nonlinear scaling (GNS). GNS scales eigenvalues of the original spectrum and leverages the scaled eigenvalue and original eigenvectors to generate a forged graph. Considering both data availability and anonymity, we list four heuristic rules to guarantee that the new spectrum has similar structural properties to the real one while has high anonymity. In summary, this nonlinear scaling method preserves all information carried by the eigenvectors without leaking the privacy of the original graph.

In the experimental part, the effectiveness of the GNS algorithm is evaluated from three aspects. The first one is the correctness of the scaling rules. And the effectiveness of different functions is compared through three metrics on a real-world data set. Results show that our scaling rules can preserve the properties of the original graph completely. The second one is the applicability of the GNS. We calculated three typical metrics on different kinds of data set, which are modularity, number of partitions, and average clustering. Compared with the SGF, our method has better performance in clustering coefficient and modularity due to all eigenvectors are used. The last one is the anonymity ability. Publishing the graph generated by a real one usually tends to disclose the privacy information in the real graph. Attackers often use a deanonymity attack to explore the relationship between two graphs and identify the nodes. Therefore, we use the ability to resist the deanonymity attack to evaluate the algorithm's anonymity. Experimental results show that the GNS maintains the original spectrum structure and has a strong resistance to a deanonymous attack.

2. Preliminaries

In this section, we present some background knowledge on random graph generation and introduce the role of each component in the spectrum.

2.1. Notation

Graph. A graph is a pair $G = (V, E)$, where V is a set whose elements are called vertex (singular: vertex) and E is a set of paired vertices, whose elements are called edges.

Adjacency matrix. An adjacency matrix is a square matrix used to represent a finite graph. The elements of the matrix indicate whether pairs of vertices are adjacent or not in the graph. In the special case of a finite simple graph, the adjacency matrix is a (0,1) matrix with zeros on its diagonal. If the graph is undirected (i.e., all of its edges are bidirectional), the adjacency matrix is symmetric.

Spectrum. The spectrum of a matrix is the set of its eigenvalues. More generally, if $T\colon V \longrightarrow V$ is a linear operator over any finite-dimensional vector space, its spectrum is the set of scalars λ such that $T - \lambda I$ is not invertible. The whole spectrum provides valuable information about a matrix.

Principal component. It is the dominant eigenvalue, which is the largest in absolute value. It is used in many applications, such as PageRank and PCA.

Deanonymous attack. Deanonymization attacks attempt to identify the nodes in a graph, exploiting similarities between the two graphs and potentially auxiliary information.

Algebraic growth. Algebraic growth is a change that has a constant rate.

Exponential growth. Exponential growth is a specific way that a quantity may increase over time. It occurs when the instantaneous rate of change (i.e., the derivative) of a quantity with respect to time is proportional to the quantity itself.

2.2. Role of Spectrum. When analyzing the utility of a graph for networks, the metrics usually used are degree [14], eigenvectors [15], eigenvalues [16], clustering coefficient

[17, 18], spectral radius [19], etc. The spectrum is closely related to these graph utility metrics. In this paper, we focus on the spectrum of networks. It mainly includes the eigenvalues of the adjacency matrix and Laplace matrix of the network.

There are two important eigenvalues for the analysis of spectrum, which are the largest eigenvalue λ_1 of adjacency matrix A and the second-largest eigenvalue v_2 of Laplace matrix L. The eigenvalues of A encode information about the cycles of a network as well as its diameter [16]; the largest eigenvalue λ_1 is related to some metrics such as the maximum degree and the number of communities. In [20], the authors study how the virus propagates in real networks. In this paper, the general epidemic threshold for an arbitrary graph is proposed. It is proved that under a reasonable approximation, the epidemic threshold for a network is closely related to λ_1. The eigenvalues of the Laplace matrix encode tree structural information about the original network. The second-largest eigenvalue v_2 is closely related to the community partition of the original network. In [21], $1 - v_2$ is the lower bound of the normal cut of graphs. For the corresponding eigenvectors, it also can help acquire clusters or communities in the network. When the eigenvectors are combined with other available network statistics (e.g., node degree), they can be used to describe various network properties.

Many studies have shown that smaller eigenvalues in the spectrum are also critical. In [22], the eigenvectors associated with small eigenvalues carry smoothly varying signals, encouraging neighbour nodes to share similar values, so that small eigenvectors are referred to as low-frequency components in graphs. As pointed out by [23], the low-frequency components in graphs help preserve intracluster information in the network and reflect the local characteristics of the graph. Similarly, reference [21] proposed that for a network structure, the importance of a node group is entirely determined by the smallest eigenvalue of its Laplacian primary submatrix, which can be used to identify nodes (groups) that have a systemic effect on the whole situation and control such nodes (groups) to achieve the best overall effect at the lowest cost.

Spectrum plays a pivotal role in some popular researches at present. The PCA method is a typical method of operating on eigenvalues, which is widely used in image processing [24] and pattern recognition [25]. Karhunen–Loève transform is a method for eigenvalues in image processing, which is equivalent to the PCA method when the matrix of the K–L transform is a covariance matrix. In the field of quantum mechanics, especially in atomic physics and molecular physics, the atomic orbitals and molecular orbits in the Hartree–Fock equation can be defined as the eigenvectors of the Fock operator [26, 27]. The corresponding eigenvalues can be explained by the Koopmans theorem as the ionization potential. In the field of social networks, the computation of eigenvalues and eigenvectors is very extensive. Researchers not only can study the network properties based on the spectrum but also reconstruct the network based on the spectrum of the published networks [9].

3. Our Method

In this section, we will elaborate on our nonlinear scaling method and prove that the scaling spectrum can provide both data availability and anonymity.

3.1. How to Preserve Eigenvectors

Problem. $*$: For a real symmetric matrix A, we have $A \cdot X = X \cdot \Lambda$, where Λ is the diagonal matrix composed of eigenvalues, $X = (\alpha_1, \ldots, \alpha_n)$, and α_i is the corresponding eigenvector of eigenvalue $\lambda_i, i = 1, \ldots, n$. The goal is to generate a matrix B, which has the same eigenvectors as those of A.

Matrix A is diagonalizable because it is a real symmetry matrix. Then, we have

$$X^{-1}AX = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \lambda_n \end{pmatrix} = \Lambda. \tag{1}$$

Lemma 1 (see [28]). *Let λ be the eigenvalue of matrix A, and X is the corresponding eigenvector of λ. λ^m are the eigenvalues of A^m, and X is the corresponding eigenvector of λ^m. $A^m x = \lambda^m x$, where $m \in N$.*

Proof (Mathematical induction). If λ is the eigenvalue of a matrix, X is the corresponding eigenvectors of λ. The following formula can be obtained. When$m = 1$, $A^m x = \lambda^m x$ established.

$$Ax = \lambda x. \tag{2}$$

Let $A^n x = \lambda^n x$, when $m = n$, then there is the following formula:

$$\begin{aligned} A^{n+1}x &= A(A^n)x \\ &= A(\lambda^n)x \\ &= \lambda^n(Ax) \\ &= \lambda^n(\lambda x) \\ &= \lambda^{n+1}x. \end{aligned} \tag{3}$$

From the above, it can be concluded that, for $m \in N$, the proposition is established; λ^m is the eigenvalue of matrix A^m, and X is the eigenvector corresponding to λ^m. There is

$$A^m X = \Lambda^m X. \tag{4}$$

□

Theorem 1 (eigenvectors' preserving). *For a real symmetric matrix A of order n, if B is a polynomial matrix of it, matrices A and B have the same eigenvectors.*

Proof. $B = f(A)$; when B is the polynomial matrix of A, $f(x)$ is the Lagrange polynomial as follows:

$$f(x) = ax^m + bx^{m-1} + \cdots + c. \tag{5}$$

The lemma shows that $A^m X = \Lambda^m X$; there is the following formula:

$$\begin{aligned}&\left(aA^m + bA^{m-1} + \cdots + c\right)X \\ &\quad = aA^m X + bA^{m-1}X + \cdots + cX \\ &\quad = a\Lambda^m X + b\Lambda^{m-1}X + \cdots + cX \\ &\quad = f(\Lambda)X.\end{aligned} \tag{6}$$

From formulas (7) and (8) is established. When B is a polynomial matrix of matrix A, it has the same eigenvectors as matrix A.

$$aA^m + bA^{m-1} + \cdots + c = f(A) = B, \tag{7}$$

$$B \cdot X = f(\Lambda) \cdot X. \tag{8}$$

To sum up, the problem $*$ can be solved as follows: firstly, the spectral decomposition of matrix A is performed, $A = X \cdot \Lambda \cdot X^{-1}$. Then the Lagrange polynomial $f(x)$ is used to scale Λ to $f(\Lambda)$, noted as Λ'. Finally, a new matrix B is constructed by the following formula. In this case, the eigenvectors of matrices B and A are the same.

$$B = X^{-1}\Lambda' X. \tag{9}$$

□

3.2. How to Scale. In the last section, the correctness of the nonlinear scaling method is proved; a polynomial scaling will preserve all eigenvectors of matrix A. However, the eigenvector preserving theorem does not tell us what kind of polynomial should be used. The scaling rules will be given with specific analysis in this section to guarantee data availability and anonymity.

For a matrix A, its spectrum is a space in which the eigenvectors are the basis vectors, and the eigenvalues refer to the stretch lengths in each direction. Therefore, the Lagrange polynomials used for scale should not change the original spectrum's structural features, such as the corresponding relationship between eigenvectors and eigenvalues. Only in this case can the algorithm guarantee the data availability of the forged graph. Correspondingly, considering the privacy protection of graph data, the deanonymity attack should not be neglected. When the forgery graph is attacked, the number of identified nodes can reflect the anonymity ability of the algorithm. All of these puts forward higher requirements for the selection of polynomials in the scaling process.

Let us take a toy for example to explain in detail: Suppose the original spectrum with three base vectors $\alpha_1 = [1, 2, 3]$, $\alpha_2 = [4, -2, -0.5]$, and $\alpha_3 = [3, 5, 4]$, each vector corresponding to the stretch length of $\lambda_1 = 18$, $\lambda_2 = 7$, and $\lambda_3 = 0.6$. Our scaling goal is to ensure data availability while also maintaining high privacy.

3.2.1. Principal Keeping. For data availability, ensuring the original spectrum's feature is the first priority. More precisely, the importance of components before and after mapping should be consistent. It can be seen from Figure 1, the graph on the left is the original spectrum with three base vectors α_1, α_2, and α_3. The corresponding stretch length is 18, 7, and 0.6, respectively; α_1 is the principal component; and α_3 is the smallest component. In the right graph, the base vectors remain unchanged; with stretch lengths scaled by $f(x) = \cos(x)$, they become 0.66, 0.75, and 0.82; α_3 becomes the main component while α_1 is the smallest one. It is clear that the features of the spectrums have been significantly changed. Therefore, the first rule to choose a scaling function is that the importance of each component must be guaranteed to remain unchanged.

3.2.2. Algebraic Growth. Figure 2 shows the results after mapping by three functions with different properties, all of which conform to the above order-keeping rules. Compared with $f(x) = e^x$ and $f(x) = \log(x)$, the spectrum scaled by $f(x) = 0.1x^2 + x$ is the closest to the original spectrum. The reason is that both $f(x) = e^x$ and $f(x) = \log(x)$ scale the eigenvalues in an exponential way, while Lagrange polynomial guarantees the algebraic growth of the eigenvalues so that the spectrums' features are not greatly changed. In particular, we can see that the exponential growth of $f(x) = e^x$ makes the principal component α_1 grow rapidly, leading to the neglect of other components in the whole space.

3.2.3. Nonlinear Scaling. For a deanonymity attack, the far the distance between matrices A and B, the less node will be identified. In this perspective, although the first-order polynomial is the best form to keep the algebraic growth, it only enlarges the values in equal proportion and does not change any structure features. At this point, if the adversary uses a strong deanonymity attack, most of the nodes will be identified. Therefore, the scaling function should be nonlinear. That is, the Lagrange polynomials cannot just be linear first-order functions.

3.2.4. Distribution Dependency. Adaptive parameters are necessary. The range of eigenvalues of different spectrums is very different. For the sake of getting better performance, adaptive parameters are needed to adjust the changing trend of function.

To sum up, we give four heuristic rules to select the optimal scaling function. For the toy example, we give a concrete function and analyze it. First of all, to not change the order of the original eigenvalues and make sure it scales at the algebraic level, we use $f(x) = ax^2 + bx$, which is a monotone increasing within the range of λ_i. Secondly, we can adjust a and b to make the trend of function more suitable for the current eigenvalue distribution to maintain the proportion of each component that does not change in essence. Therefore, we give the following scaling function: $f(x) = 0.05x^2 + x$ for this toy example. Figure 3 shows that the characteristics of the original space are well preserved after being scaled.

Figure 4 describes in detail how to apply the above rules to design the scaling function.

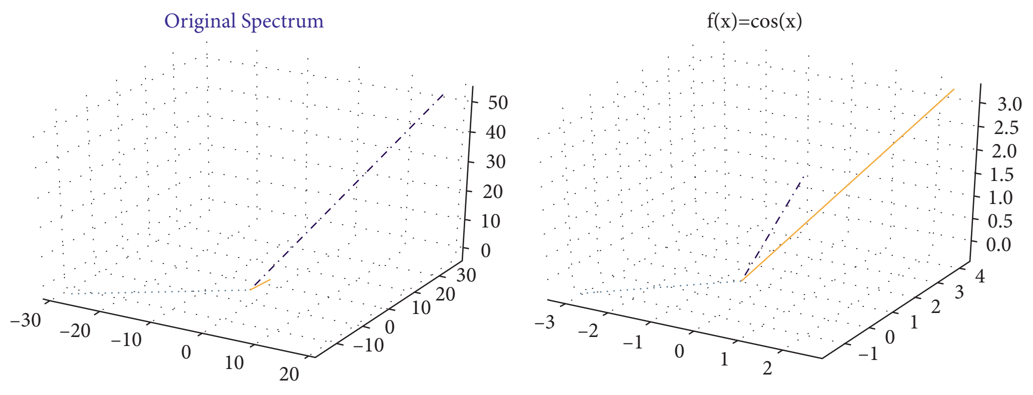


Figure 1: Scaling by the nonmonotonic function $f(x) = \cos(x)$.

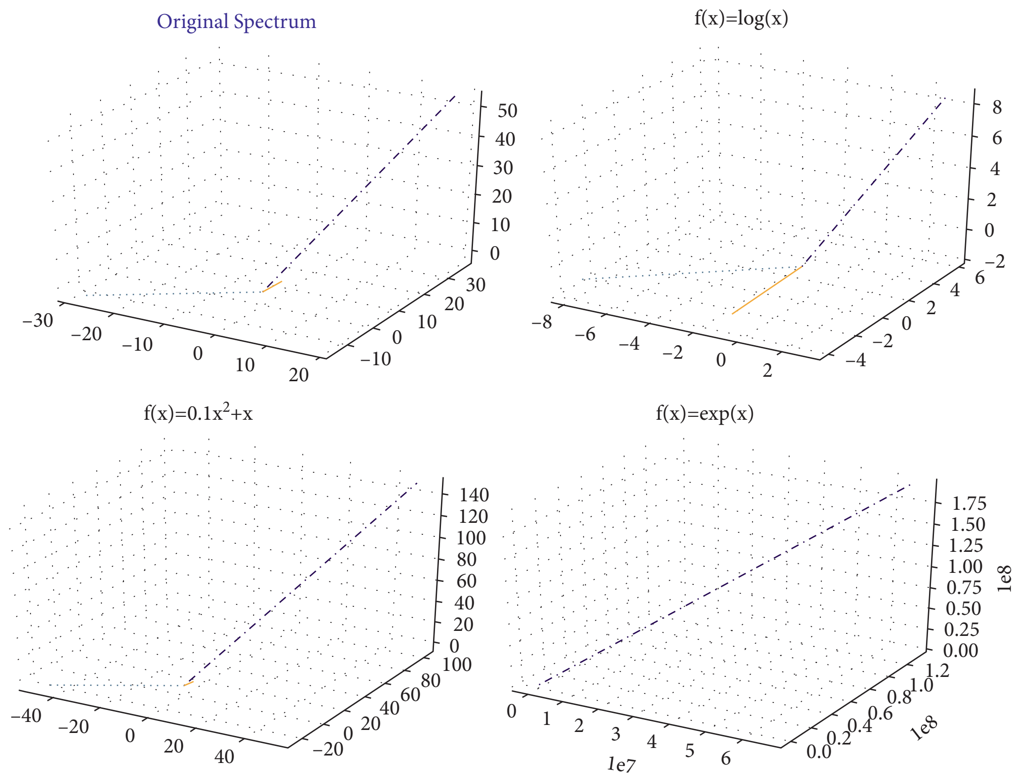


Figure 2: Scaling by functions of different forms.

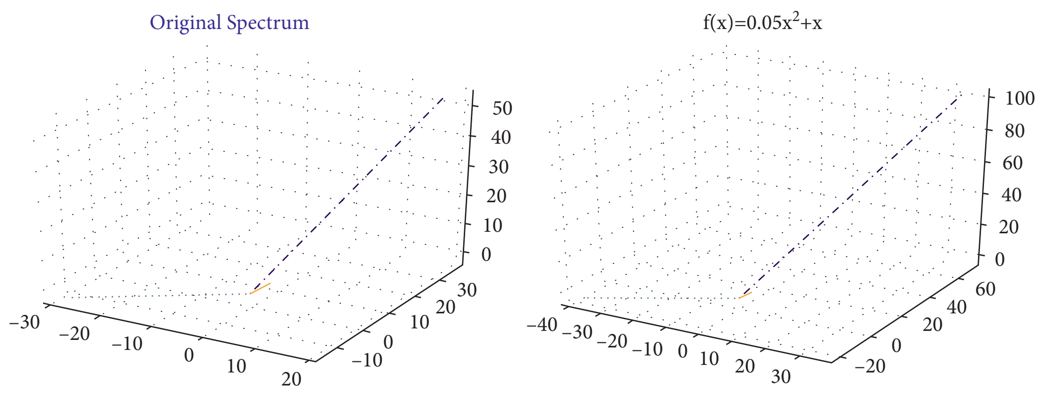
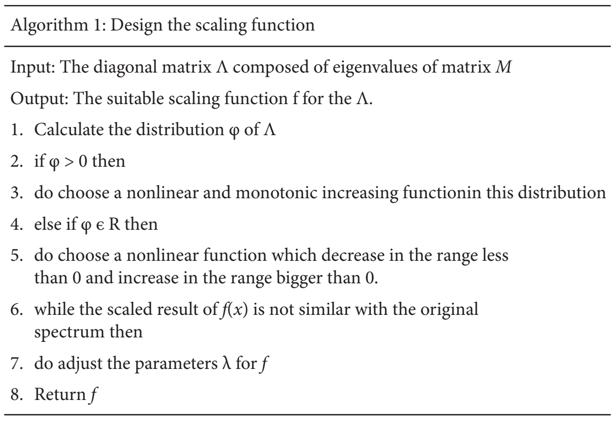


Figure 3: Scaling by example functions.


Algorithm 1: Design the scaling function

Input: The diagonal matrix Λ composed of eigenvalues of matrix *M*

Output: The suitable scaling function f for the Λ.

```
1. Calculate the distribution φ of Λ
2. if φ > 0 then
3. do choose a nonlinear and monotonic increasing functionin this distribution
4. else if φ є R then
5. do choose a nonlinear function which decrease in the range less
   than 0 and increase in the range bigger than 0.
6. while the scaled result of f(x) is not similar with the original
   spectrum then
7. do adjust the parameters λ for f
8. Return f
```

FIGURE 4: Algorithm 1 (design the scaling function).

```
    Input: The diagonal matrix $\Lambda$ composed of eigenvalues of matrix $M$
    Output: The suitable scaling function $f$ for the $\Lambda$
(1) Calculate the distribution $\varphi$ of $\Lambda$
(2) if $\varphi > 0$ then
(3) do choose a nonlinear and monotonic increasing function in this distribution
(4) else if $\varphi \in \mathbb{R}$ then
(5) do choose a nonlinear function that decreases in the range less than 0 and increases in the range bigger than 0
(6) while the scaled result of $f(x)$ is not similar to the original spectrum then
(7) do adjust the parameters $\lambda$ for $f$
(8) Return $f$
```

ALGORITHM 1: Design the scaling function

3.3. Efficiency of the Scaling Rules. In the last section, we intuitively show the impact of different scaling functions through a toy example's spectrum image. This section will explain the effect of different functions through three metrics on a real-world data set. The first metric is modularity, which can well reflect the conservation level of the community structure. The second is the clustering coefficient, which shows the change degree of local features. At last, we measured the degree series correlation. It is a vital perspective to reflect the network topology. The scaling function for this data set designed by Algorithm 1 is $f(x) = 0.01(x + 30)^2 + 0.2x$. It perfectly keeps the importance order of each component in this spectrum while the scaling presents an algebraic growth.

For these three important indexes, we take the ratio before and after scaling as the evaluation standard. As shown in Figure 5, the best performance occurs when the scaling function is $f(x)$. Modularity ratio and average clustering ratio are away from 1 with the function's order increase. In Algorithm 1, the distribution parameters are added to adjust the change trend of the function, which makes $f(x)$ the most suitable for the spectrum. A series of rules in Algorithm 1 ensure that the spectrum before and after scaling will not change essentially.

3.4. Graph Nonlinear Scaling Algorithm (GNS). Based on the analysis above, we summarize the method in this paper as follows. As shown in Figure 6, the algorithm takes matrix M as the input. First, spectrum decomposition on matrix M is performed, where V is the eigenvectors and Λ represents the eigenvalues. The second step is selecting the scaling function $f(x)$, which is the key to this algorithm. As shown in the figure, the scaled Λ is expressed as Λ'. And the new matrix M' is constructed by V and Λ'. This method is conducive to preserving the structural features of the real spectrum.

Next, we will further demonstrate the GNS algorithm in the form of pseudocode in Figure 7. Algorithm 2 is the overall process for obtaining the target matrix, where Algorithm 1 is used to design the scaling function.

4. Experiments

In this section, we evaluate the performance of GNS on three representative data sets. We start by describing the data sets used in this experiment, followed by the results on three metrics. And we compared the results with state-of-the-art method SGF. It shows that our method is better than SGF in those metrics with a high correlation with eigenvectors. For those metrics that have a loose connection with eigenvectors, our results are almost the same as those of the SGF method. Finally, we apply the state-of-the-art deanonymization attack: the Distance Vector attack [29], to assess the anonymousness of GNS.

4.1. Evaluation Setup. Three data sets will be used in this experiment. The first one is the karate club data set, which is the classic network in the field of social networks. Second

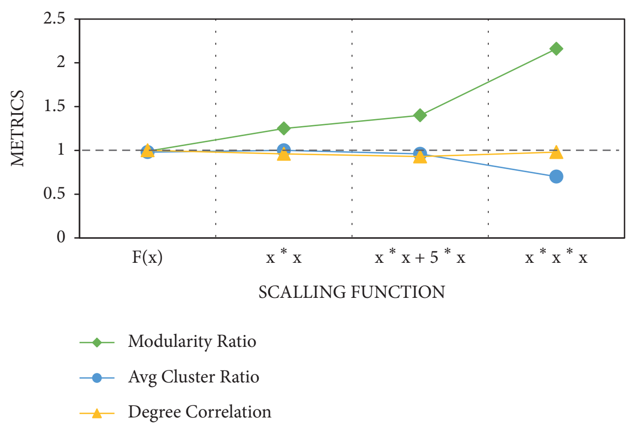


Figure 5: Performance on different functions with three metrics.

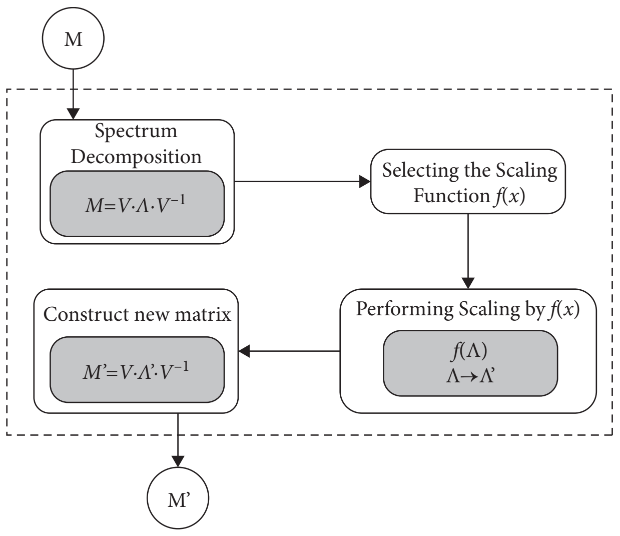


Figure 6: The pipeline of GNS.

```
Algorithm 2: Graph non-linearly scaling
Input: The adjacency matrix of matrix $M$
Output: Theadjacency matrix of matrix with the same eigenvectors of M.
1. V, $\Lambda$= similar diagonalization ($M$)
2. $f$(x)= Design the scaling function ($\Lambda$)
3. $\Lambda'$=f($\Lambda$)
4. $M'=V\cdot\Lambda\cdot V^{-1}$
5. Return $M'$
```

Figure 7: Algorithm 2 (graph nonlinearly scaling).

and third data sets are the Facebook and Twitter ego networks, all from McAuley and Leskovec [30]. The karate data set can reflect the effect of our algorithm on the classic social network. The Facebook data set is used to be a powerful way to show that our algorithm does surpass SGF. The third data set proves the general validity of our method in a large-scale social network.

In this experiment, for a more convincing demonstration that our approach is indeed superior to the SGF algorithm, we only modify the low-rank approximation module of SGF according to the idea of the control variable method, and the other modules are consistent. To be more specific, we also take the modularity matrix as the input of GNS and process Bernoulli samples on the output matrix to obtain the final graph.

4.1.1. Karate Club Network. This network is a social network constructed by Zachary, a sociologist, by observing an American University karate club. The network consists of 34 nodes and 78 sides. The individual node represents a club member, while the side represents the friendship between the members. The karate club network has become a classic data set in the detection of the complex network community structure.

4.1.2. Facebook Ego Network. This data set is the user data of Facebook collected from the App side, including its attributes, social circles, and ego network. There are 4,039 users and 88,234 connections. It consists of a group of ego networks derived from the Facebook social network, including

```
Input: The adjacency matrix of matrix $M$
Output: The adjacency matrix of the matrix with the same eigenvectors of $M$
(1) V, $\Lambda$ = similar diagonalization ($M$)
(2) $f(x)$ = Design the scaling function ($\Lambda$)
(3) $\Lambda' = f(\Lambda)$
(4) $M' = V \cdot \Lambda \cdot V^{-1}$
(5) Return $M'$
```

Algorithm 2: Graph nonlinearly scaling

10 different networks, ranging from 52 to 1,034 nodes. In this experiment, we took the edges from all ego nets combined as a whole graph.

4.1.3. Twitter Ego Network. The Twitter data set was crawled from public sources, which includes node features (profiles), circles, and ego networks. There are 81,306 users and 1,768,149 connections. Twitter data set includes 973 ego networks. Edges from the top 100 ego networks were combined as a directed graph in this experiment.

4.2. Results on Global and Local Metrics. To verify the effectiveness of GNS, we calculated three metrics on three kinds of data sets. At the same time, the performance of the SGF algorithm is compared. The second data set was the same one used in the SGF paper. Experimental results show that our algorithm has better performance in clustering and modularity due to all eigenvectors are preserved.

4.2.1. Modularity. As the definition in [31], modularity is the fraction of the edges that fall within the given group minus the expected fraction if the edges were distributed at random. We evaluated it on three data sets. The calculation of modularity uses the famous algorithm proposed by Lefebvre et al. [32]. Consider m_0 for the modularity of the input graph and m_1 for the forged graph. m_0/m_1 is used to judge the difference of modularity before and after processing; the more it closes to 1, the graph's global features are captured better.

As shown in Figure 8, the GNS algorithm's effect is almost the same as that of SGF using 90% spectrum; the modularity ratios on these three data sets are about 1. When SGF uses only a 40% spectrum, the forged graph does not precisely keep the real graph's structural features. Its modularity ratio is relatively far away from 1 compared with the GNS.

Meanwhile, it is worth noting that SGF ($\alpha = 0.9$) keeps 90% original spectrum leading to an inevitable problem: the distinguishability between forged and original graphs is too small. In contrast, the GNS algorithm uses Lagrange polynomials to scale the eigenvalues in the original spectrum. That ensures the GNS does not directly use the original spectrum to create a new graph, guaranteeing its anonymity. This is also reflected in the subsequent deanonymous attack experiments.

4.2.2. Number of Partitions. As we targeted global features in this experiment, the number of partitions is also a necessary global metric to measure. It shows how many communities in the real graph are preserved. In Figure 9, the partition ratio is calculated by n_0/n_1, where n_0 represents the partition number detected in the original graph and n_1 is that in the forged graph. Ideally, the value is 1. The result is similar to that of the modularity ratio. The performance of GNS on the three data sets is close to that of SGF ($\alpha = 0.9$), and both of them surpass SGF ($\alpha = 0.4$) due to more spectrum information is used.

4.2.3. Average Clustering. As a local property of networks, average clustering represents the extent of triadic closure. The average clustering ratio between the input graph and output was examined. The target value is 1. As shown in Figure 10, the experimental results are in line with our reasoning. The GNS algorithm using more spectrum information performs way better than SGF ($\alpha = 0.4$), And it is slightly better than SGF ($\alpha = 0.9$).

4.3. Deanonymity Experiment. According to the analysis, the forged graph's eigenvectors are consistent with those of the real graph, and the eigenvalues are scaled by a Lagrange polynomial. Let the real eigenvalues as the plaintext and the scaled eigenvalues as the ciphertext. The application scenario of publishing the forged graph is considered here. In this scenario, the adversary can only obtain the forged graph. Therefore, the real eigenvalue cannot be deduced.

However, the nodes may be transparent for the attacker if they use a strong deanonymity attack. As introduced, a deanonymization attack will exploit the similarities between two graphs to identify nodes with only a few seeds. In our evaluation, the state-of-the-art deanonymization attack: The distance vector (DV) attack is used. About 5% of nodes are set as the ground truth seed to feed in the DV attack. At each time, we record the percentage of the identified node to verify the privacy protection ability. The experiment results on Facebook data set are shown in Figure 11.

As shown in Figure 11, we compared the ability to resist the deanonymous attack of GNS and SGF algorithms under the same modularity ratio.$|MR - 1|$ is the absolute value between the modularity ratio and 1 where the modularity ratio is noted as MR. Based on this, we divide the values into four boxes, as shown in Figure 11, each box represents a level of data availability. For SGF, it can be seen that as the value is

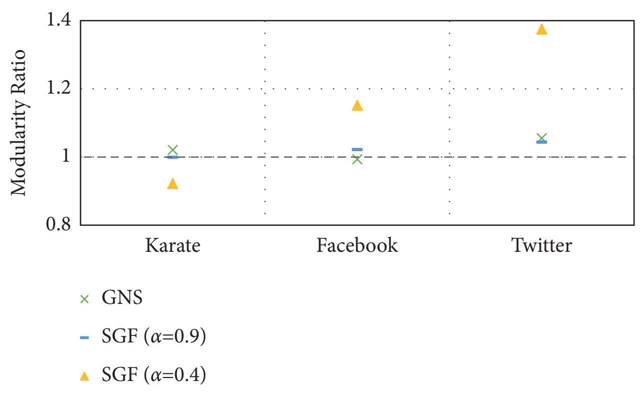


Figure 8: The modularity ratio on the karate, Facebook, and Twitter data sets.

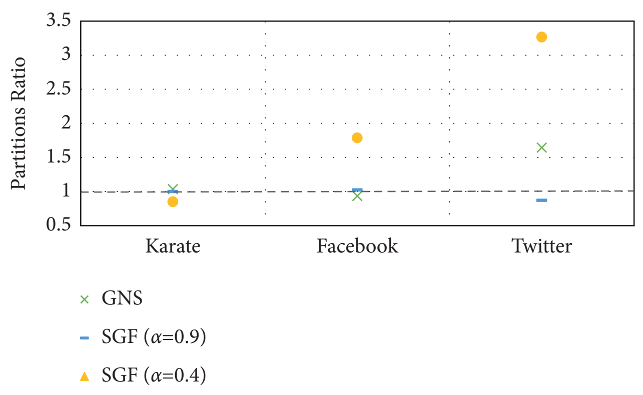


Figure 9: The partitions ratio on karate, Facebook, and Twitter data sets.

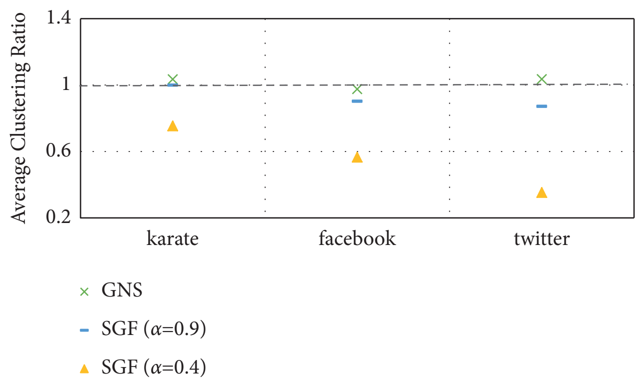


Figure 10: The average clustering ratio on karate, Facebook, and Twitter data sets.

closer to 0, the more global features being captured, the number of identified nodes is also increasing rapidly. Especially when the value is between 0.00 and 0.04, the percentage points are close to 100%. Only when the SGF algorithm makes a compromise by discarding part of the spectrum, the percentage of identified nodes will be below 40%, whereas its modularity is far lower than expected. The trend line of SGF presents a nearly exponential growth. On the contrary, GNS has only about 40% nodes, which are recognized when the distance is between 0.00 and 0.04. The growth trend is nearly an algebraic growth. From this, we can conclude that compared to SGF, the GNS algorithm has a stronger ability to resist deanonymity attacks. Furthermore, high data availability can still be guaranteed. In other words, when we release the GNS-processed graph, the spectrum of the forged graph is similar to the original one. More importantly, it is difficult for the adversary to reveal too much information from the forged image, which ensures the privacy of the original graph.

Table 1 shows the performance of different methods on Facebook data set, comparing the data availability and anonymity. As shown in the table, when the modularity ratio

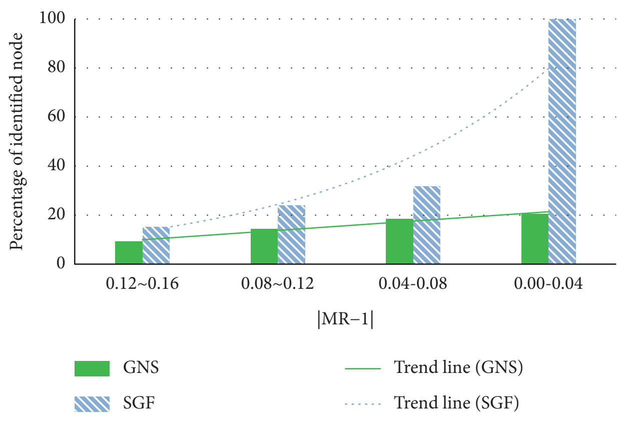


Figure 11: The percentage of identified nodes of SMG and SGF compared under the same modularity, MR is the modularity ratio. The distance vector attack is used.

Table 1: Comparison of availability and anonymity of data obtained by different methods on Facebook.

Method	Modularity ratio	Percentage of the identified node by distance vector attack (%)
GNS	0.993	20.6
SGF ($\alpha = 0.9$)	1.022	100
SGF ($\alpha = 0.4$)	1.151	15.2

approaches 1, for the GNS algorithm, only 20.6% of the nodes in the forged graph are identified and provide data availability and anonymity at the same time. When α equals 0.4, the SGF algorithm can achieve a similar degree of anonymity as GNS, while its modularity ratio is 1.151 at this time, which means the algorithm does not capture the global structure characteristics very well, the data availability is relatively low when it has high anonymity.

5. Conclusion

This paper presents a GNS algorithm for graph generation and graph anonymization. The forged graph has a similar spectrum to the original one, like a perfect stand-in of the real graph. More precisely, the new spectrum is constructed by the scaled eigenvalues and original eigenvectors. Therefore, the formed graph maintains the structure of the original graph very well. Moreover, the scaling function designed by rules guarantees that the new graph does not disclose the private information of the real one. GNS is suitable for any situation where eigenvectors are needed. This paper mainly studies the applications in the graph area. Compared with the state-of-the-art algorithm spectral graph forge (SGF), our results not only retain the data availability but also outperform the SGF algorithm in anonymity.

Data Availability

The social network data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] J. Xiong, J. Ren, L. Chen et al., "Enhancing privacy and availability for data clustering in intelligent electrical service of IoT," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1530–1540, 2018.

[2] S. Aldossary and W. Allen, "Data security, privacy, availability and integrity in cloud computing: issues and current solutions," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 4, pp. 485–498, 2016.

[3] W. Y. Day, N. Li, and M. Lyu, "Publishing graph degree distribution with node differential privacy," in *Proceedings of the 2016 International Conference on Management of Data*, pp. 123–138, San Francisco, CA, USA, July 2016.

[4] C. Hong, J. Zhang, W.-B. Du, J. M. Sallan, and O. Lordan, "Cascading failures with local load redistribution in interdependent Watts-Strogatz networks," *International Journal of Modern Physics C*, vol. 27, no. 11, Article ID 1650131, 2016.

[5] C. Pu, K. Wang, and Y. Xia, "Robustness of link prediction under network attacks," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 8, pp. 1472–1476, 2019.

[6] R. Milo, N. Kashtan, S. Itzkovitz, M. Newman, and U. Alon, "On the uniform generation of random graphs with prescribed degree sequences," 2003, https://arxiv.org/abs/cond-mat/0312028.

[7] K. L. Calvert, M. B. Doar, and E. W. Zegura, "Modeling internet topology," *IEEE Communications Magazine*, vol. 35, no. 6, pp. 160–163, 1997.

[8] M. Gjoka, B. Tillman, and A. Markopoulou, "Construction of simple graphs with a target joint degree matrix and beyond," in *Proceeding of the 2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 1553–1561, Hong Kong, China, May 2015.

[9] L. Baldesi, C. T. Butts, and A. Markopoulou, "Spectral graph forge: graph generation targeting modularity," in *Proceeding of the IEEE INFOCOM 2018-IEEE Conference on Computer*

Communications, pp. 1727–1735, Honolulu, HI, USA, April 2018.

[10] T. A. B. Snijders and K. Nowicki, "Estimation and prediction for stochastic block models for graphs with latent block structure," *Journal of Classification*, vol. 14, no. 1, pp. 75–100, 1997.

[11] S. Trajanovski, F. A. Kuipers, J. Martín-Hernández, and P. Van Mieghem, "Generating graphs that approach a prescribed modularity," *Computer Communications*, vol. 36, no. 4, 2013.

[12] W. Ye, S. Goebl, C. Plant, and C. Bohm, "FUSE: full spectral clustering," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1985–1994, San Francisco, CA, USA, August 2016.

[13] A. Jamakovic and S. Uhlig, "On the relationship between the algebraic connectivity and graph's robustness to node and link failures," in *Proceeding of the 3rd EuroNGI Conference on Next Generation Internet Networks*, Trondheim, Norway, May 2007.

[14] X. Ying and X. Wu, "Randomizing social networks: a spectrum preserving approach," in *Proceedings of the 2008 SIAM International Conference on Data Mining*, pp. 739–750, Society for Industrial and Applied Mathematics, Atlanta, GA, USA.

[15] K. Liu and E. Terzi, "Towards identity anonymization on graphs," in *Proceedings of the 2008 ACMSIGMOD International Conference on Management of data*, pp. 93–106, ACM, Vancouver, Canada, June 12 2008.

[16] L. Wu, X. Ying, and X. Wu, "Reconstruction from randomized graph via low rank approximation," in *Proceedings of the SIAM International Conference on Data Mining*, pp. 60–71, Society for Industrial and Applied Mathematics, Columbus, OH, USA, May 2010.

[17] L. Zou, L. Chen, and M. T. Özsu, "k-automorphism," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 946–957, 2009.

[18] M. Brautbar and M. Kearns, "A clustering coefficient network formation game," in *Proceeding of the International Symposium on Algorithmic Game Theory*, pp. 224–235, Athens, Greece, October 2011.

[19] M. S. Birman and M. Z. Solomjak, *Spectral Theory of Self-Adjoint Operators in Hilbert Space*, Springer Science & Business Media, Berlin, Germany, 2012.

[20] Y. Wang, D. Chakrabarti, C. Wang, and C. Faloutsos, "Epidemic spreading in real networks: an eigenvalue viewpoint, Reliable Distributed Systems," in *Proceedings of the 22nd International Symposium on*, pp. 25–34, Florence, Italy, October 2003.

[21] L. Hui, X. Xu, G. Chen, and X. Xu, "Optimizing pinning control of complex dynamical networks based on spectral properties of grounded Laplacian matrices," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 99, pp. 1–11, 2018.

[22] C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec, "Learning structural node embeddings via diffusion wavelets," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1320–1329, London United Kingdom, August 23 2018.

[23] T. Maehara, "Revisiting graph neural networks: all we have is low-pass filters," 2019, https://arxiv.org/abs/1905.09550.

[24] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision, Cengage Learning*, Springer, Berlin, Germany, 2014.

[25] N. M. Nasrabadi, "Pattern recognition and machine learning," *Journal of Electronic Imaging*, vol. 16, no. 4, Article ID 049901, 2007.

[26] A. Szabo and N. S. Ostlund, *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure theory*, Courier Corporation, Chelmsford, MA, USA, 2012.

[27] B. N. Khoromskij, V. Khoromskaia, and H.-J. Flad, "Numerical solution of the Hartree-Fock equation in multilevel tensor-structured format," *SIAM Journal on Scientific Computing*, vol. 33, no. 1, pp. 45–65, 2011.

[28] J. Steven, *Leon Linear Algebra with Applications*, Bargain Smart Plug, London, UK, 7th edition, 2006.

[29] S. Ji, W. Li, P. Mittal, X. Hu, and R. A. Beyah, "SecGraph: a uniform and open-source evaluation system for graph data anonymization and de-anonymization," in *Proceedings of the 24th USENIX Conference on Security Symposium*, pp. 303–318, Washington, DC, USA, August 2015.

[30] J. J. McAuley and J. Leskovec, "Learning to discover social circles in ego networks," *News in Physiological Sciences*, vol. 2012, pp. 548–556, 2012.

[31] M. E. J. Newman, "Modularity and community structure in networks, 2006 APS March Meeting," *American Physical Society*, vol. 103, no. 23, pp. 8577–8582, 2006.

[32] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and experiment*, vol. 2008, no. 10, Article ID P10008, 2008.