Maximizing the Smallest Eigenvalue of Grounded Laplacian Matrix by Node Selection

Run Wang, Xiaotian Zhou, Zhongzhi Zhang, *Member, IEEE,* and Guanrong Chen, *Life Fellow, IEEE*

***Abstract*—The smallest eigenvalue of a grounded Laplacian matrix plays a pivotal role in complex networks, such as system control, convergence rate and the robustness of a system. In this paper, we focus on the node selection problem of maximizing the smallest eigenvalue of the grounded Laplacian matrix for a graph with n nodes and m edges, under an upper bound constraint $k \ll n$. We show this combinatorial optimization problem is NP-hard and the objective function is monotone but non-submodular. Since the optimal solution cannot be calculated directly, we adopt a greedy strategy to solve this problem by selecting one node at a time. Specifically, we employ derivatives and matrix perturbations to make our algorithm efficient with a time complexity of $\tilde{O}(km)$, where $\tilde{O}(\cdot)$ notation suppresses the $\mathrm{poly}(\log n)$ factors, and also applicable to large-scale networks. We conduct numerous experiments on different networks of various sizes to demonstrate the superiority of our algorithm in terms of efficiency and effectiveness compared to other methods.**

***Index Terms*—Grounded Laplacian, spectral properties, graph mining, linear algorithm, matrix perturbation, derivative, pinning control, convergence speed.**

I. Introduction

Spectral theory has already a long history and widespread usage in different scenarios. The eigensystem of adjacency matrix can evaluate network topology for immunization and infrastructure study. Besides, the eigensystem of Laplacian matrix can measure the network's connectivity. Our study focuses on a variant of Laplacian matrix, grounded Laplacian matrix, whose smallest eigenvalue matters broadly.

Grounded Laplacian matrix, a principal submatrix of the Laplacian matrix, functions as a significant model in network control study. Due to the impossibility of controlling all nodes in the network, steering a fraction of nodes is a significant alternative, which relates to grounded Laplacian matrix closely. Both pinning control and leader selection problem employ grounded Laplacian matrix as their model, and the eigenvalue of the grounded Laplacian matrix performs as efficient metrics in diverse control systems.

Specifically, the smallest eigenvalue of grounded Laplacian matrix can not only quantify node-selection scheme for pinning control strategy but also evaluate convergence rate and the system $\mathcal{H}_\infty$ norm in consensus dynamics. Then a spontaneous question arises as what nodes should be controlled with respect to the smallest eigenvalue of grounded Laplacian matrix. Nevertheless, prior researches remain elusive on scalable node-selection strategy for the smallest eigenvalue of grounded Laplacian matrix.

Therefore, our work focuses on the SMALLEST EIGENVALUE OPTIMIZATION problem: Given a graph $\mathcal{G} = (V, E)$ with $|V| = n$ nodes and $|E| = m$ edges, and an integer $0 < k < n$, how to select a subset $S \subset V$ and $|S| \leq k$ such that the smallest eigenvalue of the grounded Laplacian matrix induced from the graph and grounded node set S is maximized.

The SMALLEST EIGENVALUE OPTIMIZATION problem is inevitable and difficult. Preceding study on its inverse proportion with network size implies the necessity of the refined node-selection scheme on a large network. However, the fundamental properties of the problem resist advances. Firstly, its combinatorial nature leads to exponential computation complexity so that the brute-force algorithm fails even in medium-sized networks. Secondly, without submodularity, greedy algorithms, as the conventional resort for NP-hard problems, lose their approximation guarantee. To our best knowledge, the computation techniques for this problem are either empirical or exclusive for small networks.

Our work is a comprehensive study of the SMALLEST EIGENVALUE OPTIMIZATION problem. We prove the NP-hardness and non-submodularity of this combinatorial optimization problem. Then we propose a nearly linear time heuristic algorithm. It capitalizes on two analysis methods: network derivative mining, and matrix disturbance theory, to evaluate the eigen-gap for node selection. The consistency between these two methods justifies the reliability of our analysis. Sufficient experiments and excellent results warrant the performance of our algorithm.

II. Related Works

Grounded Laplacian matrix was first proposed in the study of linear system with nodes grounded [1] and is later employed in pinning control and consensus model study. Its smallest eigenvalue value relates to the efficacy of pinning control strategy, consensus convergence rate and system $\mathcal{H}_\infty$ norm,

This work was supported by the National Key R & D Program of China (Nos. 2018YFB1305104 and 2019YFB2101703), the National Natural Science Foundation of China (Nos. 61872093, 61803248, U19A2066 and 61672166), Shanghai Municipal Science and Technology Major Project (No. 2018SHZDZX01) and ZJLab. Run Wang was also supported by Fudan's Undergraduate Research Opportunities Program (FDUROP) under Grant No. 2195200241021. (Corresponding author: Zhongzhi Zhang)

Run Wang, Xiaotian Zhou and Zhongzhi Zhang are with the Shanghai Key Laboratory of Intelligent Information Processing, School of Computer Science, Fudan University, Shanghai 200433, China. Zhongzhi Zhang is also with the Shanghai Engineering Research Institute of Blockchains, Fudan University, Shanghai 200433, China. (e-mail: runwang18@fudan.edu.cn; 20210240043@fudan.edu.cn; zhangzz@fudan.edu.cn).

Guanrong Chen is with the Department of Electrical Engineering, City University of Hong Kong, Hong Kong SAR, China (e-mail:eegchen@cityu.edu.hk).

which are studied extensively. We introduce the relation between these applications with the smallest eigenvalue in turn. Then we present extant researches on this problem.

Pinning control is a strategy to steer a fraction of nodes to achieve synchronization or a targeted state on the whole network [2], [3], [4], [5]. To achieve the targeted state, it is impossible and unnecessary to control all nodes, and therefore pinning control was proposed [2], [3]. However, regarding which nodes to be controlled, different prior studies provide varied strategies, some of which are exclusive for specific networks or even contradict with each other. For example, max-degree scheme is recommended for small number of controlled nodes [3], [6], [7], while nodes with small degrees outperforms when the number grows [8], [9]. Besides, betweenness centrality [10] and ControlRank [11] are also pinning node-selection strategies but for specific situations. After the relation between the smallest eigenvalue of the grounded laplacian matrix and effectiveness of pinning node-selection is found [12], [13], detailed properties is investigated [9].

The grounded Laplacian matrix also functions in continuous-time diffusion dynamics, especially the opinion dynamics, where individuals are consistently exchanging opinions with their neighbors [14]. The opinions of each individual reach an equilibrium for some models [15], [16], [17], [18], and the convergence rate equals to the smallest eigenvalue of the grounded Laplacian matrix [19]. Its properties have been studied in [20], [21] and some researches investigate how to maximize the convergence rate by leader selection [22], [23].

In leader-follower consensus dynamics, the robustness is defined based on the system $\mathcal{H}_2$ or $\mathcal{H}_\infty$ norms, which map the grounded Laplacian matrix to the robustness metrics. And the system $\mathcal{H}_\infty$ is the reciprocal of the smallest eigenvalue of the grounded Laplacian matrix. The relationship between it and robustness to disturbances or time delay have been studied in some fields, including intelligent transportation [24], [25] and power systems [26]. Tight characterizations for the system norm have been provided in [27], and one leader-selection problem to minimize the robustness metric $\mathcal{H}_\infty$ is also considered.

Recent years witnessed different methods to optimize the smallest eigenvalue of grounded Laplacian matrix by node selection. A submodular approach is developed and allowed edge weight negative [28]. Besides, a feature-embedded evolutionary algorithm is proposed [29]. Close relation with the resistance distance is shown, which leads to a new approach [30]. And max-degree strategy is improved based on the eigenvalue's properties [9]. However, there is a lack of sufficient experiments on diverse and large-scale networks among these researches.

III. PRELIMINARY

In this section, we introduce the concepts relevant to the focus of this study, including the graph and associated matrices, and properties for later use.

A. Notations and graph

To point out firstly, in this paper scalars in $\mathbb{R}$ are denoted by normal lowercase letters like a, b, c, vectors by bold lowercase letters like $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$, sets by normal uppercase letters like A, B, C and matrixes by bold uppercase letters like $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$. We use the notation $\lambda(\boldsymbol{A})$ to denote the smallest eigenvalue of matrix $\boldsymbol{A}$. For the element in a vector or a matrix, $\boldsymbol{a}_i$ is used to denote the i-th element in the vector $\boldsymbol{a}$, and $\boldsymbol{A}_{ij}$ is used to denote the entry (i, j) of matrix $\boldsymbol{A}$. Specifically, we use $\boldsymbol{0}$ and $\boldsymbol{I}$ to denote zero matrix and identity matrix of proper dimension, notation $\boldsymbol{e}_i$ to denote a vector with i-th element being 1 and others 0, and notation $\boldsymbol{E}_{ij}$ to denote a matrix with the entry (i, j) being 1 and others being 0.

We denote an undirected connected binary graph as $\mathcal{G} = (V, E)$ where V is the set of nodes with size $|V| = n$ and $E \subseteq V \times V$ is the the set of edges with size $|E| = m$. Let $\boldsymbol{A} \in \{0, 1\}^{n\times n}$ be the adjacency matrix for graph $\mathcal{G}$ where $\boldsymbol{A}_{ij}$ equals 1 if node i and j are connected and 0 otherwise. The neighbors of node $i \in V$ in graph $\mathcal{G}$ are given by the set $\mathcal{N}_i = \{j \in V | (i, j) \in E\}$. The degree of node i is represented as $d_i = |\mathcal{N}_i|$ and the degree matrix is defined as $\boldsymbol{D} = \text{diag}(d_1, d_2, \ldots, d_n)$ accordingly. The Laplacian matrix for the graph is given by $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$.

B. Grounded Laplacian Matrix

Grounded Laplacian matrix is a variant of Laplacian matrix. Deleted rows and columns corresponding to selected nodes in nonempty set S of size k, grounded Laplacian matrix is the principal submatrix of Laplacian matrix $\boldsymbol{L}$ with size of $(n - k) \times (n - k)$. Let the grounded Laplacian matrix be $\boldsymbol{L}(S)$ for compactness of notation, and $\boldsymbol{L}(S)$ is a symmetric diagonally-dominant M-matrix(SDDM). The grounded Laplacian matrix has wide range of applications in various field such as pinning control [9], the convergence rate of a leader-follower opinion dynamical systems [22], the $\mathcal{H}_\infty$ norm of a system [27], and so on.

For ease of notation, the smallest eigenvalue of grounded Laplacian matrix, which is the focus of this work, is simplified as $\lambda(S) = \lambda(\boldsymbol{L}(S)) > 0$. And the eigenvector corresponding to the $\lambda(S)$ is denoted as $\boldsymbol{u}$, which can be nonnegative according to Perron-Frobenius theorem [31].

We define two operations on the grounded Laplacian matrix $\boldsymbol{L}(S)$: node removal and edge removal, which are closely related to the problem we are studying. We will see an increase in the smallest eigenvalue after these two operations.

Definition 3.1 (Node removal): For a graph $\mathcal{G} = (V, E)$, a set $S \subset V$ and any node $i \in V \backslash S$, the new grounded Laplacian matrix after removing node i is defined as $\boldsymbol{L}(S + i)$, which is a principle submatrix of $\boldsymbol{L}(S)$.

The smallest eigenvalue increases $\lambda_S(i) = \lambda(S + i) - \lambda(S) \geq 0$ after node removal according to Cauchy's interlacing theorem[32]. More specifically, if node i connects to any unselected node, then $\lambda_S(i) > 0$ holds.

Definition 3.2 (Edge removal): For a graph $\mathcal{G} = (V, E)$, a set $S \subset V$ and any edge $e = (i, j) \in (V \backslash S) \times (V \backslash S)$, the new grounded Laplacian matrix after removing edge e is defined as

$$\boldsymbol{L}(S + e) = \boldsymbol{L}(S) + \boldsymbol{E}_{ij} + \boldsymbol{E}_{ji}.$$

The smallest eigenvalue varies with $\lambda_S(e) = \lambda(S+e) - \lambda(S)$, which is a positive value.

C. Submodular Function

We give a brief definition of monotone non-decreasing and submodular set functions. For a set S, we use $S+u$ to denote $S \cup \{u\}$.

Definition 3.3 (Monotonicity): A set function $f : 2^V \to \mathbb{R}$ is monotone non-decreasing if $f(S) \leq f(T)$ holds for all $S \subseteq T$.

Definition 3.4 (Submodularity): A set function $f : 2^V \to \mathbb{R}$ is submodular if

$$f(S+u) - f(S) \geq f(T+u) - f(T)$$

holds for all $S \subseteq T \subseteq V$ and $u \in V$.

IV. PROBLEM FORMULATION

Based on above sections, the widespread applications of the smallest eigenvalue of have been stated. Especially the smallest eigenvalue $\lambda(S)$ can be employed to evaluate the control effectiveness, quantify the convergence rate of consensus system and measure the robustness of a system. In this section, we introduce the eigenvalue optimization problem under limited control nodes, which is hard to solve as follows.

A. Problem Statement

The node removal operation indicates that adding a new node to the control set causes the increase of the smallest eigenvalue of the corresponding grounded Laplacian matrix. So we propose the problem of maximizing the smallest eigenvalue subject to a cardinality constraint $|S| \leq k$, which is stated below.

Problem 1 (Smallest Eigenvalue Optimization): Given a unweighted, undirected and connected network $\mathcal{G} = (V, E)$, we aim to find a subset $S \subset V$ with k nodes so that the smallest eigenvalue of the grounded Laplacian matrix $\boldsymbol{L}(S)$ is maximized. The problem can be formulated as

$$S^* = \arg \max_{S \subset V, |S| \leq k} \lambda(S).$$

The optimization Problem 1 is not easy to handle, especially the objective function of the study is the smallest eigenvalue of the matrix. So we will give three examples to take a deeper look at some of its interesting properties.

B. Weak strictly monotonicity

For graphs that are specially structured, such as star graphs, after selecting the central node, selecting more nodes does not increase the smallest eigenvalue. As shown in Fig. 1, the result of the corresponding optimal solution is the same for k ranging from 1 to 7, because central node 1 is always selected in the optimal solution.

However, for general graphs, it is difficult to select a small number of nodes that makes picking more nodes useless, especially for $k \ll n$. The node removal operation have told us unless the new added node is not adjacent to the unselected

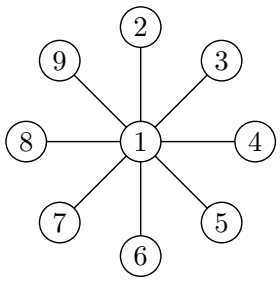

Figure 1: An example of a star graph

node, the smallest eigenvalue of the grounded Laplacian matrix will strictly increase after adding this new node. It is almost impossible for the selected nodes to be the optimal solution of the problem if they are closely adjacent to each other. So we call this strictly monotone increase as weak strictly monotone increase, where the strict monotonicity is satisfied under certain conditions but can be easily achieved in the real situation of this problem. Thus optimization Problem 1 is equivalent to the combinatorial optimization problem stated below.

Problem 2: Given a unweighted, undirected and connected network $\mathcal{G} = (V, E)$, we aim to find a subset $S \subset V$ with k nodes so that the smallest eigenvalue of the grounded Laplacian matrix $\boldsymbol{L}(S)$ is maximized. The problem can be formulated as

$$S^* = \arg \max_{S \subset V, |S| = k} \lambda(S).$$

C. Node centrality incompatibility

For each node i, we define a centrality $\lambda(i)$ based on the smallest eigenvalue of the grounded Laplacian matrix, where the most central node can lead to the largest change in the smallest eigenvalue. For example, in Fig. 2, node 6 is the most central node and if we select only one node, the selection of 7 is the best choice. Compared with other traditional node centrality, such as degree centrality, eigenvector centrality, closeness centrality and some other node centrality, node 6 is the most central node based on their standards. As the scale of this graph gets larger, that is, there are more and the same number of nodes in the star graph on the left and in the line graph on the right, many node centrality such as degree centrality will still consider the center of the star graph to be the most important node in this graph, but the node that gives the largest value of the smallest eigenvalue of the grounded Laplacian matrix will be gradually shifted to the right.

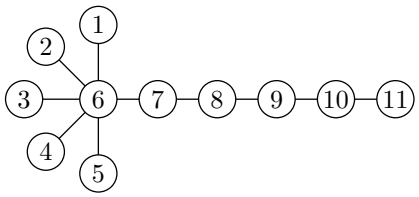

Figure 2: An example of a star-line graph

Although this new node centrality seems better than other node centralities, however, simply select top-k central nodes

based on this approach may ignore their global relevance, and it may significantly reduce the effectiveness of picking them together when k is large. We can see that in Fig. 2, the second central node based on $\lambda(i)$ is node 6, but the selection of node 6 and node 7 is not the optimal solution, since they are too close to each other, they are not fully functional in terms of contribution to the overall network.

So we define the centrality of a group of nodes by $\lambda(S)$ to overcome the limitations imposed by the centrality of a single node. When we add a new node i into the set S, $\lambda_S(i)$ can measure the importance of node i based on the previously selected set of nodes, which may far from $\lambda(i)$. Unlike the node centrality, this set centrality can better capture its properties and our problem turns to find a most important node set subject to the constraint k.

D. Non-submodularity

For a combinatorial optimization problem, finding the appropriate properties is crucial to help us solve it effectively. Submodularity is one of the key properties, with this property, one can propose a simple greedy algorithm by optimizing the objective function with one element in each iteration. And it yields a solution with $1-e^{-1}$ approximation ratio [33], which has been widely used in combinatorial optimization problems and has effectively solved many NP-hard problems in recent years.

However, there are still plenty of issues that do not satisfy this property, this greatly increases the difficulty of solving such problems. In our SMALLEST EIGENVALUE OPTIMIZATION problem, the objective function is not submodular. To show its non-submodularity, we give an example of a line graph with 7 nodes in Fig. 3.

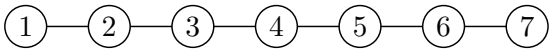

Figure 3: An example of a line graph

Set $A = \{1\}$, $B = \{1, 2\}$ and $v = 6$. We can simply calculate that

$$\lambda(A) = 0.0581, \qquad \lambda(A + v) = 0.3820,$$
$$\lambda(B) = 0.0810, \qquad \lambda(B + v) = 0.5858,$$

thus

$$\lambda(A + v) - \lambda(A) = 0.3239 < 0.5048 = \lambda(B + v) - \lambda(B).$$

Therefore, the objective function in SMALLEST EIGENVALUE OPTIMIZATION problem is non-submodular.

E. Optimal solution lacks correlation

Another weird nature of this problem is the optimal solution lacks correlation, that is for $k = x$ and $k = x + 1$, where x is a positive integer, the optimal solution for each case may be quite different. We can see from Fig. 3, the selection of node 4 is the solution for $k = 1$ while the selection of node 3 and node 5 is an optimal solution when $k = 2$, but for $k = 3$, the selection of node 2, node 4 and node 6 is the optimal solution. The optimal solutions corresponding to adjacent k lack some correlation, which may lead to the fact that when we take a node-by-node selection approach, even if we achieve optimality at each step, there is bound to be a certain gap with the optimal solution.

F. Hardness of the problem

Next, we show the hardness of our SMALLEST EIGENVALUE OPTIMIZATION problem.

Theorem 4.1: The SMALLEST EIGENVALUE OPTIMIZATION problem is NP-hard.

Proof. We consider the decision version of the problem as: given graph $\mathcal{G}$ and an integer k to find a set S of k grounded nodes to ensure the $\lambda(S) \geq t$. And it suffices to establish the theorem for $t = 3$ and $\mathcal{G}$ as a 3-regular graph. Since the problem is apparently in NP: given set S, we can check in poly-time if $\lambda(S)$ is less than 3 or not, we only need to prove the NP-completeness.

We reduce the problem from vertex cover problem, which is defined to find a set of nodes where every edge has at least one endpoint in.

First if S is a vertex cover of $\mathcal{G}$, then $V\backslash S$ is an independent set of $\mathcal{G}$, which means there is no edges connecting any pair of nodes in $V\backslash S$. It follows that $\boldsymbol{L}(S)$ is a diagonal matrix with diagonal entries being 3. Therefore $\lambda(S) = 3$.

Second we prove it by contradiction. Suppose $\lambda(S) \geq 3$ while set S is not vertex cover then $V\backslash S$ is not independent set. Thus $V\backslash S$ includes nodes adjacent to nodes in $V\backslash S$. Then $\boldsymbol{L}$ can be represented as:

$$\boldsymbol{L} = \begin{pmatrix} \boldsymbol{L}_1 & \mathbf{0} \\ \mathbf{0} & \boldsymbol{L}_2 \end{pmatrix},$$

where $\boldsymbol{L}_1$ includes nodes which are not adjacent with nodes in $V\backslash S$; therefore $\boldsymbol{L}_1$ is a diagonal matrix with diagonal entries being 3. Oppositely, $\boldsymbol{L}_2$ is a connected component. Then a lemma is proposed to support the proof.

Lemma 4.2: If set S is not the vertex cover of graph $\mathcal{G}$, the trace of $\boldsymbol{L}(S)^{-1}$ is larger than $\frac{n-k}{3}$.

Proof. If S is not $VC(G, k)$, the inverse of $\boldsymbol{L}(S)$ is

$$\begin{pmatrix} \boldsymbol{L}_1^{-1} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{L}_2^{-1} \end{pmatrix}, \tag{1}$$

For each node $v \in A$, its degree d_v is 3. Then we can write $\boldsymbol{L}_2$ into block form as

$$\boldsymbol{L}_2 = \begin{pmatrix} d_u & -\boldsymbol{b}^T \\ -\boldsymbol{b} & \boldsymbol{B} \end{pmatrix}, \tag{2}$$

where $\begin{pmatrix} d_u \\ -\boldsymbol{b} \end{pmatrix}$ is the colomn corresponding to the node v and $\boldsymbol{B}$ denote the corrosponding principal submatrix. By block-wise matrix inversion we have $(\boldsymbol{L}_2^{-1})_u = 1/\left(d_u - \boldsymbol{b}^T\boldsymbol{B}^{-1}\boldsymbol{b}\right)$. Because B is positive definite and b is not a zero vector (since A is a connected component), $\boldsymbol{b}^T\boldsymbol{B}^{-1}\boldsymbol{b} > 0$, which gives that $1/\left(d_u - \boldsymbol{b}^T\boldsymbol{B}^{-1}\boldsymbol{b}\right) > 1/d_u = 1/3$. Then $\mathrm{Tr}\left(\boldsymbol{L}_2^{-1}\right) > |A|/3$. And the trace of $\boldsymbol{L}_1^{-1}$ equals to $(n-|A|-k)/3$. In all, the trace of $\boldsymbol{L}(S)^{-1}$ is larger than $\frac{n-k}{3}$. □ According to lemma 4.2,

the smallest eigenvalue of $L(S)$ is smaller than 3 which leads to a contradiction, because

$$\lambda(L(S)) = \frac{1}{\lambda_{\max}(L(S)^{-1})} \leq \frac{n-k}{\mathrm{Tr}\left((L(S)^{-1})\right)} < 3,$$

which completes the proof. □

G. A simple heuristic algorithm

Since Problem 1 is a combinatorial problem, and we can naturally think of a simple solution. For each possible case where k nodes are selected in a set S, we calculate the smallest eigenvalue of the grounded Laplacian matrix generated by the set S separately and output the S^* which maximize the smallest eigenvalue. Obviously, this approach fails when n or k is slightly larger since the time complexity of it is $O(\binom{n}{k}m)$.

So due to the NP-hardness of the problem, we propose a greedy heuristic algorithm by optimizing the increment of the smallest eigenvalue for each step. Although there is no guarantee of error factor, we can still see in the later experiments that this method has a huge advantage over other schemes. All the operation we need here is Node Removal. First, the set S is set to be empty, then k nodes are added from set $V\backslash S$ step by step. In each iteration, for each $i \notin S$, we need to calculate the increment of the smallest eigenvalue $\lambda_S(i) = \lambda(S+i) - \lambda(S)$. The smallest eigenvalue of a grounded Laplacian matrix can be calculated by the numerical method in $O(m)$ time, thus a direct implementation of this approach takes $O(knm)$ time. The above analysis leads to our simple heuristic algorithm EXACT$(\mathcal{G}, k)$, as outlined in Algorithm 1.

Algorithm 1: EXACT$(\mathcal{G}, k)$

Input : A graph $\mathcal{G} = (V, E)$; an integer $k \leq |V|$
Output : S: a subset of V with $|S| = k$

Initialize solution $S = \emptyset$
for $i = 1$ *to* k **do**
  Compute $\lambda_S(j) = \lambda(S+j) - \lambda(S)$ for each $j \notin S$
  Select s s.t. $s \leftarrow \arg\max_{j \in V\backslash S} \lambda_S(j)$
  Update solution $S \leftarrow S + s$
return S

V. LINEAR TIME APPROXIMATION ALGORITHM

However, the simple algorithm takes too much time and is not feasible for large-scale network. The main difficulty is how to quickly calculate the increment of the smallest eigenvalue. So we select the nodes from two different perspectives.

Instead of removing a node directly, we consider the influence of removing an edge on the smallest eigenvalue from both continuous and discrete ways, which are derivative and matrix perturbation. And then we give a reasonable explanation to select a control node.

A. Derivative-based analysis

The derivative represents the change rate of a function. Similarly, if an element is changed in a matrix, the change rate of a function that maps a graph to a real number can also be defined by derivative. In some previous works [34], [35], [36], some authors use derivatives to measure the importance of an edge on a certain quantity. Although our task is to select several nodes, measure the importance of a node based on the sum of edges' importance incident to it is a normal practise. So for a matrix $\boldsymbol{L}$ and each edge $e = (i, j) \in E$, we fist define the derivative matrix as $\boldsymbol{B}(e) = \frac{\partial \lambda}{\partial \boldsymbol{L}_{ij}}$.

Lemma 5.1: Given a graph $\mathcal{G} = (V, E)$ with Laplacian matrix $\boldsymbol{L}$, let $S \subset V$ be a node set. The smallest eigenvalue-eigenvector pair of $\boldsymbol{L}(S)$ is defined as $(\lambda, \boldsymbol{u})$. Then for any edge $e = (i, j) \in E \cap (V\backslash S) \times (V\backslash S)$, we have

$$\boldsymbol{B}_{ij} = \boldsymbol{B}(e) = \frac{\partial \lambda}{\partial \boldsymbol{L}(S)_{ij}} = \boldsymbol{u}_i \boldsymbol{u}_j.$$

Proof. According to the properties of the eigenvector $\boldsymbol{u}$, we have $\boldsymbol{L}(S)\boldsymbol{u} = \lambda \boldsymbol{u}$. Take a derivative of this equation, we have

$$\frac{\partial \boldsymbol{L}(S)}{\partial \boldsymbol{L}(S)_{ij}}\boldsymbol{u} + \boldsymbol{L}(S)\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{L}(S)_{ij}} = \frac{\partial \lambda}{\partial \boldsymbol{L}(S)_{ij}}\boldsymbol{u} + \lambda \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{L}(S)_{ij}}.$$

Left multiplying $\boldsymbol{u}^\top$ on both side yields $\boldsymbol{B}_{ij} = \frac{\partial \lambda}{\partial \boldsymbol{L}(S)_{ij}} = \boldsymbol{u}_i \boldsymbol{u}_j$. □

$\boldsymbol{B}_{ij}$ can be regarded as the change rate of $\lambda(S)$ when the element of $\boldsymbol{L}(S)_{ij}$ is altered. We can think intuitively that the importance of each node is equal to the sum of the importance of the edge adjacent to it. So the change rate of $\lambda(S)$ after removing node i can be defined as

$$C_1(i) = \sum_{j \in \mathcal{N}_i - S} \boldsymbol{B}_{ij} = \boldsymbol{u}_i \sum_{j \in \mathcal{N}_i - S} \boldsymbol{u}_j. \tag{3}$$

B. Perturbation-based analysis

In another way, we can use the matrix perturbation theory [37] to get the change of the eigenvalue after perturbing a few elements in a matrix. Node removal operation has poor performance since too many elements may change including both diagonal and non-diagonal elements. So instead of directly removing row and column corresponding to node i from the original matrix, we start from the simplest case, where only one edge is removed by edge removal operation. We show the increment of the smallest eigenvalue more precisely based on the matrix perturbation theory. So our matrix perturbation is given in following steps.

Lemma 5.2: Let $\mathcal{G} = (V, E)$ be a connected graph and $\boldsymbol{L}$ be its Laplacian matrix, let $S \subset V$ be a node set. For the matrix $\boldsymbol{L}(S)$, we define $(\lambda, \boldsymbol{u})$ be its smallest eigenvalue-eigenvector pair. For any edge $e = (i, j) \in E \cap (V\backslash S) \times (V\backslash S)$, define $\Delta \boldsymbol{L}(S) = \boldsymbol{E}_{ij} + \boldsymbol{E}_{ji}$, we can perturb the matrix $\boldsymbol{L}(S)$ with $\Delta \boldsymbol{L}(S)$. λ varies with $\Delta\lambda$ and $\boldsymbol{u}$ varies with $\Delta \boldsymbol{u}$ after perturbing. We have

$$(\boldsymbol{L}(S) + \Delta \boldsymbol{L}(S))(\boldsymbol{u} + \Delta \boldsymbol{u}) = (\lambda + \Delta\lambda)(\boldsymbol{u} + \Delta \boldsymbol{u}), \tag{4}$$

and the eigen-change for the smallest eigenvalue can be approximated by

$$\Delta\lambda \approx 2\boldsymbol{u}_i\boldsymbol{u}_j > 0.$$

Proof. Left multiplying $\boldsymbol{u}^\top$ on both side in Eq. (4), we have

$$\boldsymbol{u}^\top\Delta\boldsymbol{L}(S)\boldsymbol{u} + \boldsymbol{u}^\top\Delta\boldsymbol{L}(S)\Delta\boldsymbol{u} = \boldsymbol{u}^\top\Delta\lambda\boldsymbol{u} + \boldsymbol{u}^\top\Delta\lambda\Delta\boldsymbol{u}.$$

The eigen-gap for the smallest eigenvalue is

$$\Delta\lambda = \frac{\boldsymbol{u}^\top\Delta\boldsymbol{L}(S)\hat{\boldsymbol{u}}}{\boldsymbol{u}^\top\hat{\boldsymbol{u}}} = \frac{\boldsymbol{u}_i\hat{\boldsymbol{u}}_j + \boldsymbol{u}_j\hat{\boldsymbol{u}}_i}{\boldsymbol{u}^\top\hat{\boldsymbol{u}}},$$

where $\hat{\boldsymbol{u}} = \boldsymbol{u} + \Delta\boldsymbol{u}$.

For a large scale network, the removal of one edge has very small influence on the network, and it does not influence the eigenvector $\boldsymbol{u}$ too much, which means $\Delta\boldsymbol{u} \approx \boldsymbol{0}$ [38], [39], [40]. Hence

$$\Delta\lambda \approx 2\boldsymbol{u}_i\boldsymbol{u}_j > 0,$$

which is consistent with the properties of the edge removal operation. □

If we repeat the operation of edge removal on all the edges incident to node i, then the Lemma 5.2 can be extended to the following form.

Lemma 5.3: Let $\mathcal{G} = (V, E)$ be a connected graph and $\boldsymbol{L}$ be its Laplacian matrix, let $S \subset V$ be a node set. For the matrix $\boldsymbol{L}(S)$, we define $(\lambda, \boldsymbol{u})$ be its smallest eigenvalue-eigenvector pair. For any node $i \in V\backslash S$, define $\Delta\bar{\boldsymbol{L}}(S) = \sum_{j\in\mathcal{N}_i - S}\boldsymbol{E}_{ij} + \boldsymbol{E}_{ji}$, we can perturb the matrix $\boldsymbol{L}(S)$ with $\Delta\bar{\boldsymbol{L}}(S)$. λ varies with $\Delta\bar{\lambda}$ and $\boldsymbol{u}$ varies with $\Delta\bar{\boldsymbol{u}}$ after perturbing. We have

$$(\boldsymbol{L}(S) + \Delta\bar{\boldsymbol{L}}(S))(\boldsymbol{u} + \Delta\bar{\boldsymbol{u}}) = (\lambda + \Delta\bar{\lambda})(\boldsymbol{u} + \Delta\bar{\boldsymbol{u}}),$$

and the eigen-gap for the smallest eigenvalue can be approximated by

$$\Delta\bar{\lambda} \approx 2\boldsymbol{u}_i \sum_{j\in\mathcal{N}_i - S}\boldsymbol{u}_j > 0.$$

According to Lemma 5.3, if we add a new node i into set S, then $\lambda + \Delta\bar{\lambda}$ equals to $\lambda(S+i)$ when $\lambda(S+i) < d_i$. Thus maximization $\lambda(S+i)$ equals to maximization $\Delta\bar{\lambda}$. According to Lemma 5.3, the increment of the smallest eigenvalue by removing the edge adjacent to node i can be defined as

$$C_2(i) \approx 2\boldsymbol{u}_i \sum_{j\in\mathcal{N}_i - S}\boldsymbol{u}_j. \tag{5}$$

Remark 5.4: Comparing Eqs. (3) and (5), we find that they are the same when the coefficients are ignored. Without directly calculating the influence of removing one node, we operate on each edge in continuous and discrete ways to get the importance of each node. The conclusion that the two expressions are consistent exhibits the coherence between two methods. So in the following parts, we will use

$$C(i) = \boldsymbol{u}_i \sum_{j\in\mathcal{N}_i - S}\boldsymbol{u}_j \tag{6}$$

to denote the importance of node i based on the matrix $\boldsymbol{L}(S)$, where $\boldsymbol{u}$ is the eigenvector corresponding to the smallest eigenvalue of $\boldsymbol{L}(S)$.

Remark 5.5: If S is an empty set, it is apparent that the smallest eigenvalue of $\boldsymbol{L}(S)$ is 0, and the corresponding eigenvector is $\boldsymbol{u} = \boldsymbol{1}$. Hence the importance of each node equals to the degree of node i if we neglect the coefficient, which is consistent with the conclusion in [9] that selecting nodes with large degrees is a good solution when only a few number of nodes can be controlled.

Algorithm 2: APPROXVECTOR$(\boldsymbol{M}, \epsilon)$

Input : A SDDM matrix $\boldsymbol{M}$; an error parameter $0 < \epsilon < 1$

Output : $\boldsymbol{u}$: an approximation of the smallest eigenvector of the matrix $\boldsymbol{M}$

1 Initialize vector $\boldsymbol{u}, \boldsymbol{v} = \boldsymbol{1}^\top$
2 Set error $e = 1$
3 **while** $e > \epsilon$ **do**
4 $\quad\boldsymbol{u} = \boldsymbol{v}$
5 $\quad\boldsymbol{v} = \text{SOLVE}(\boldsymbol{M}, \boldsymbol{u}, \epsilon)$
6 $\quad\boldsymbol{v} = \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_\infty}$
7 $\quad e = \|\boldsymbol{u} - \boldsymbol{v}\|_\infty$
8 **return** $\boldsymbol{u}$

C. Fast Algorithm

Based on prior analysis, the optimization of $\lambda(S)$ is reduced to the computation of the corresponding eigenvector $\boldsymbol{u}$. Before presenting our fast algorithm, we compare two methods for eigenvector computation.

1) Power Iteration Method: A classical method to calculate the leading eigenvector of a matrix is power iteration method. To employ this method, the $\boldsymbol{L}(S)$ matrix is transformed to $\boldsymbol{M} = x\boldsymbol{I} - \boldsymbol{L}(S)$. The choice of x will largely affect the speed of the calculation. One may think $\lambda_{\max}(\boldsymbol{L}(S))$ is a proper choice, where $\lambda_{\max}(\boldsymbol{L}(S))$ is the largest eigenvalue of the matrix $\boldsymbol{L}(S)$. However, relative to the smallest and second smallest eigenvalues λ_1, λ_2 of $\boldsymbol{L}(S)$, $\lambda_{\max}(\boldsymbol{L}(S))$ is still too large, resulting in a very slow iteration rate.

2) Approximate Eigenvector: Another method to obtain an estimate of $\boldsymbol{u}$ without the expense of computing the complete eigensystem is proposed in [41], [42], [43], where the smallest eigenvector of a SDDM matrix $\boldsymbol{L}(S)$ can be solved in nearly linear time based on Lemma 5.6 and inverse power method. For consistency of structure, we first show the Solver.

Lemma 5.6: There is a nearly linear time solver $\boldsymbol{x} = \text{SOLVE}(\boldsymbol{S}, \boldsymbol{y}, \epsilon)$ which takes a symmetric positive semi-definite matrix $\boldsymbol{S}_{n\times n}$ with m nonzero entries, a vector $\boldsymbol{b} \in \mathbb{R}^n$, and an error parameter $\delta > 0$, and returns a vector $\boldsymbol{x} \in \mathbb{R}^n$ satisfying $\left\|\boldsymbol{x} - \boldsymbol{S}^{-1}\boldsymbol{y}\right\|_{\boldsymbol{S}} \le \delta\left\|\boldsymbol{S}^{-1}\boldsymbol{y}\right\|_{\boldsymbol{S}}$ with high probability, where $\|\boldsymbol{x}\|_{\boldsymbol{S}} \overset{\text{def}}{=} \sqrt{\boldsymbol{x}^\top\boldsymbol{S}\boldsymbol{x}}$. The solver runs in expected time $\tilde{O}(m)$, where $\tilde{O}(\cdot)$ notation suppresses the $\text{poly}(\log n)$ factors.

Algorithm 2 combines the Solver into the inverse power method, which avoids inverting the matrix $\boldsymbol{M}$ directly and greatly improves the efficiency of calculations. Since the setting of parameters is not the focus of our study in the approximation calculation, in line 5, we simply set the error parameter in the Solver to ϵ, while specific details can be found in [43]. Also, we know that the smallest eigenvalue of

Algorithm 3: APPROX$(\mathcal{G}, k, \epsilon)$

Input : A graph $\mathcal{G} = (V, E)$; an integer $k < |V|$; an error parameter $\epsilon > 0$
Output : S: a subset of V with $|S| = k$

Initialize solution $S = \emptyset$
Let $\boldsymbol{L}$ be the Laplacian matrix of graph $\mathcal{G}$
for $i = 1$ *to* k **do**
  $\boldsymbol{u} =$ APPROXVECTOR$(\boldsymbol{L}(S), \epsilon)$
  Compute $C(j) = \boldsymbol{u}_j \sum_{t \in \mathcal{N}_j - S} \boldsymbol{u}_t$ for each $j \notin S$
  Select s s.t. $s \leftarrow \arg\max_{j \in V \setminus S} C(j)$
  Update solution $S \leftarrow S \cup s$
return S

the matrix $\boldsymbol{L}(S)$ is close to 0, leading to very few iterations, even one iteration can get a fairly good approximation.

Avoiding directly calculating $\Delta\lambda$ for each node, we now present a $\tilde{O}(km)$-time approximation algorithm APPROX for the problem as outlined in Algorithm 3, which is a simple adjustment made on the basis of algorithm EXACT.

VI. EXPERIMENTS

In this section, we show our algorithms' fairly good performance on diverse real world networks. Without loss of generosity, we choose connected networks with scale ranging up to million from KONECT [44] and SNAP[1]. Related information of these networks is shown in Table I, where networks are shown in increasing order of their numbers of nodes.

We implement our algorithm with Julia to use the SDDM solver SOLVE in Julia's Laplacian.jl package, and the error parameter ϵ is set to 0.01 in the subsequent experiments. We run our experiments on a Linux box with 4.2 GHz Intel i7-7700 CPU and 32G memory, using a single thread.

A. Accuracy of Eigen-gap Approximation

To begin with, we evaluate the approximation of eigen-gap $\Delta\lambda$ mentioned in Eq. (6) and compare it with accurate eigen-gap $\Delta\lambda$ for each node. We conduct the experiments on four network: Dolphins, Tribes, Karate and Email-univ. To measure the accuracy of our eigen-gap evaluation, we randomly select five nodes to be grounded firstly and then delete one more node to compute the accurate eigen-gap and $C(i)$ caused by its removal. According to Fig. 5, the accurate eigen-gap is linearly proportional to our estimate, which justifies our estimate.

B. Effectiveness

We display our algorithm's fairly good performance on diverse real life network here. The experiment results on small network, medium network and enormous network are displayed and analyzed.

The methods which we compare with are introduced as follows, inspired by the preceding node selection strategies from pinning control studies.

[1]https://snap.stanford.edu

Table I: Information about the networks with n vertices and m edges.

	Network	n	m
1	Tribes	16	58
2	Firm-Hi-Tech	33	147
3	Karate	34	78
4	Dolphins	62	159
5	685-bus	685	1282
6	Email-Univ	1133	5451
7	Bcspwr09	1723	2394
8	Routers-RF	2113	6632
9	US-Grid	4941	6594
10	Bcspwr10	5300	8271
11	Pages-Government	7057	89 455
12	WHOIS	7476	56 943
13	Pretty Good Privacy	10 680	24 340
14	Anybeat	12 645	67 053
15	Webbase-2001	16 062	25 593
16	As-CAIDA2007	26 475	53 381
17	Epinions	26 588	100 120
18	Email-EU	32 430	54 397
19	Internet-As	40 164	85 123
20	P2P-Gnutella	62 561	147 878
21	RL-Caida	190 914	607 610
22	DBLP-2010	226 413	716 460
23	Twitter-follows	404 719	713 319
24	Delicious	536 108	1 365 961
25	FourSquare	639 014	3 214 986
26	Youtube-Snap	1 134 890	2 987 624

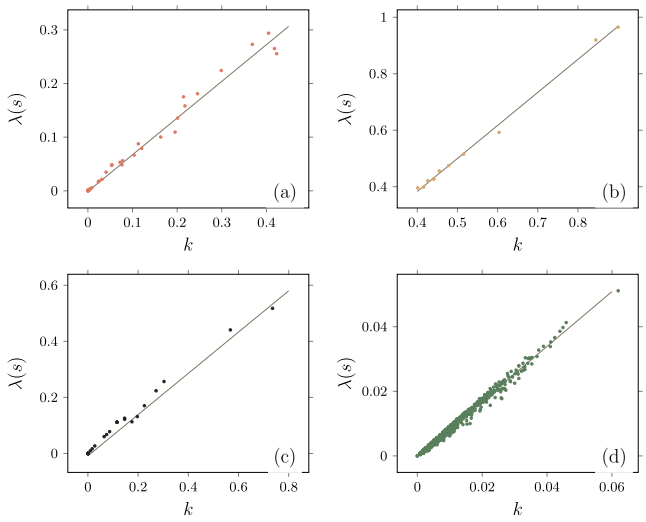

Figure 4: Eigen-gap approximation by $C(i)$ compared with accurate eigen-gap $\Delta\lambda$, to show the approximate linear relation on four small networks: (a)Dolphins, (b)Tribes, (c)Karate, (d)Email-Univ.

1) *Optimum*: choose k nodes in the set S that maximize $\lambda(S)$ by exhaustive search.
2) *Degree*: choose k nodes with largest degree.
3) *Eigenvector*: choose k nodes based on highest eigenvector centrality.
4) *Betweenness*: choose k nodes based on highest betweenness centrality.
5) *Closeness*: choose k nodes based on hightest closeness centrality.
6) *Exact*: choose k nodes returned by Algorithm 1.
7) *Approx*: choose k nodes returned by Algorithm 3.

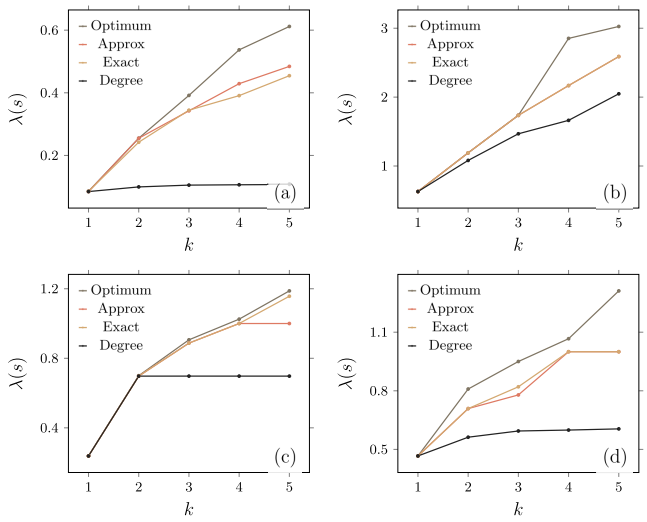

Figure 5: $\lambda(S)$ given by APPROX, EXACT, max-degree scheme, and the optimum result associated with k ranging from 1 to 5 on four small nework: (a) Dolphins, (b) Tribes, (c) Karate and (d) Firm-Hi-Tech.

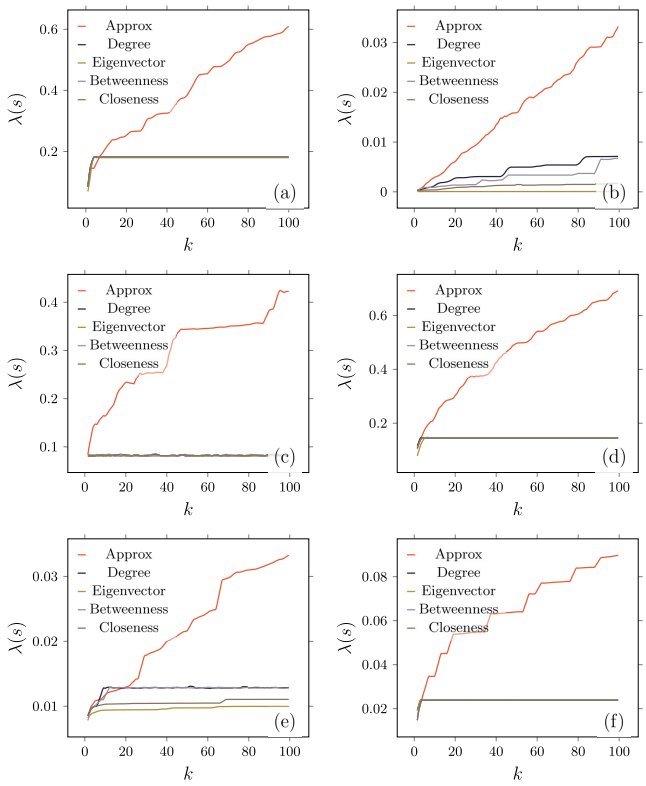

Figure 6: $\lambda(S)$ given by APPROX compares with other centrality such as degree, eigenvector, betweenness and closeness with k ranging from 1 to 100 on six medium nework: (a) Pages-Government, (b)US-Grid, (c) Anybeat, (d) WHOIS, (e) Pretty Good Privacy and (f) Epinions.

Experiments are first conducted on four small networks: Dolphins, Tribes, Karate and Firm-Hi-Tech. In Fig. 5, we display the effectiveness of our algorithms by comparing our two proposed algorithms with optimum results and max-degree scheme on networks with nodes number less than 100. Due to the exponential time required to compute the optimal solution, we only take k to a maximum of 5 in this experiment. As Fig. 5 shows, our algorithms EXACT and APPROX are close to the optimum results and better than max-degree scheme.

In order to better show the advantages of our algorithm, we compare APPROX with four other methods: Degree, Eigenvector, Betweenness and Closeness, in the case of choosing $k = 1, 2, \ldots, 100$ nodes. A comparison of results for these four algorithms is shown in Fig. 6. These four baselines are common node centrality, measure the importance of each node separately. In some special cases, such as Fig. 6(e), the schemes based on the degree centrality and the betweenness centrality have a better performance when k is less than 20. Since these two methods have been extensively studied in the past, it is reasonable that they have good results with fewer control nodes.

However, for a set of nodes, it is not wise to rely on them alone for selection because they can only obtain information about each node and cannot consider the correlation between nodes in the set in a comprehensive way. And we can assert that no matter how well a node centrality captures global and local information about the network, relying solely on node centrality, or adapting it accordingly for different networks, does not work very well when k is large.

We find that other schemes may have a better performance when very few nodes are selected, however, our method is able to keep the results growing rapidly. Because our method places more emphasis on the centrality of the set, we focus on maximizing the centrality of the set at each step, even if each node is added to the set one at a time. It signifies that the nodes selected by our algorithm are meaningful and outperforms picking nodes based on a certain metric in one step.

When the scale of the networks grows up, very few methods can be applied due to the amount of time and space required. Our algorithm APPROX can easily handle this situation by virtue of its nearly linear time complexity. And degree based approach has the time complexity of $O(n)$, and due to its effectiveness according to some existing research results, so we compare our method with Degree method on four networks with size ranging from 200,000 to 1,000,000. The comparison is shown in Fig. 7.

After the initial selection of a very small number of nodes with the largest degree, the effects in Figs. 7(b) and (c) show that the remaining nodes are difficult to play a more effective role. And Fig. 7(d) shows even though Degree approach is possible to achieve the same effect as our method in the first 10 nodes, the selection of the next 90 nodes hardly allows further improvement of the results. Although Degree approach is superior to our method in terms of speed, it produces a certain gap in effectiveness.

C. *Efficiency*

Although we have analyzed the time complexity of our two algorithms, we experimentally compare how much time they take for each step. We now show that algorithm APPROX runs

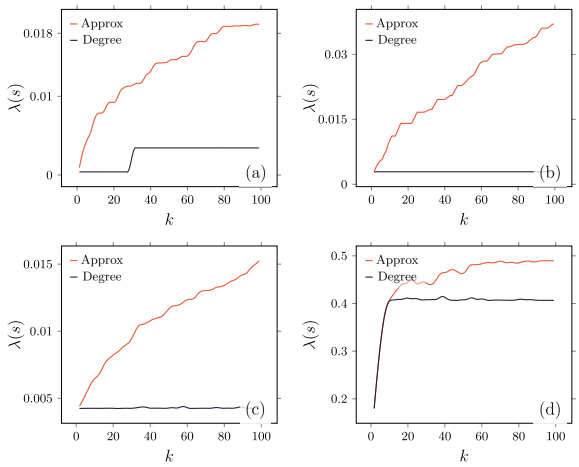

Figure 7: $\lambda(S)$ given by APPROX and max-degree scheme with k ranging from 1 to 100 on four enormous nework: (a) RL-Caida, (b) Delicious, (c) Youtube-Snap, (d) FourSquare.

Table II: Average running time for computing $\Delta\lambda$ and $C(i)$ of all nodes on a larger set of real-world networks of each iteration and their ratio.

Network	Time (seconds)		
	EXACT	APPROX	Ratio
685-bus	4.73	0.004	1182
Bcspwr09	22.60	0.011	2054
Routers-RF	52.15	0.046	1133
Bcspwr10	318.4	0.044	7236
Webbase-2001	2613	0.134	19500
As-Caida2007	5308	0.261	20337
Epinions	4931	0.229	21532
Email-EU	7148	0.126	56730
Internet-as*	-	0.227	-
P2P-gnutella*	-	0.482	-
RL-caida*	-	1.992	-
DBLP-2010*	-	2.141	-
Twitter-follows*	-	2.292	-
Delicious*	-	4.731	-
FourSquare*	-	10.95	-
Youtube-snap*	-	10.82	-

much faster than algorithm EXACT, especially on large-scale networks. We average the time taken for each iteration of each network. From Table II, we find that for the network with more than 40,000 nodes, which are marked with $*$, algorithm EXACT fails due to its tremendous time cost, while algorithm APPROX takes only a few seconds for each iteration even on the network with over one million nodes.

Also, we compare the two methods for computing the eigenvectors, as shown in Table III, although the theoretical time complexity is nearly the same, we can find that the method using Solver and the inverse power method is far superior to the method using the power method due to the influence of the number of iterations and the computational accuracy required. To get a more precise approximation, power method is likely to take a lot of time, and it fails on large-scaled network. For the large scaled networks, our method also provides a great convenience for computing the smallest eigenvectors of the SDDM matrix.

Table III: Average running time for computing the smallest eigenvector by power method and inverse power method on a larger set of real-world networks of each iteration and their ratio.

Network	Time (seconds)		
	Power Method	Inverse Power Method	Ratio
685-bus	0.030	0.012	2.5
Bcspwr09	0.057	0.004	14.25
Routers-RF	0.047	0.012	3.92
Bcspwr10	0.122	0.019	6.42
Epinions	1.082	0.160	6.76
Email-EU	0.687	0.068	10.10
P2P-gnutella*	1.091	0.359	3.04

VII. CONCLUSIONS

In this paper, we study the smallest eigenvalue optimization problem for grounded Laplacian matrix with n nodes and m edges by selecting k grounded nodes. Due to the wide range of applications in many fields, such as system control, convergence rate and the robustness of a system, while lack of effective and efficient approaches to solve this problem, we study this one problem in depth manner. Theoretically, we provide rigorous proofs of this problem's NP-hardness and non-submodularity. The complexity of the problem makes it difficult to generalize the solution to a large-scale network by directly computing the optimal solution. So we propose an efficient and generalizable algorithm to select limited nodes to maximize the smallest eigenvalue in $\tilde{O}(km)$ time. Finally, we conduct diverse experiments on a large number of real networks of different sizes to demonstrate the superiority of our algorithm.

REFERENCES

[1] U. Miekkala, "Graph properties for splitting with grounded laplacian matrices," *Bit*, vol. 33, no. 3, pp. 485–495, 1993.
[2] X. F. Wang and G. Chen, "Pinning control of scale-free dynamical networks," *Physica A*, vol. 310, no. 3-4, pp. 521–531, 2002.
[3] X. Li, X. Wang, and G. Chen, "Pinning a complex dynamical network to its equilibrium," *IEEE Trans. Circuits Syst. I-Regul. Pap.*, vol. 51, no. 10, pp. 2074–2087, 2004.
[4] X. Wang and G. Chen, "Synchronization in scale-free dynamical networks: robustness and fragility," *IEEE Trans. Circuits Syst. I-Regul. Pap.*, vol. 49, pp. 54–62, 2001.
[5] X. F. Wang and G. Chen, "Synchronization in small-world dynamical networks," *Int. J. Bifurcation Chaos*, vol. 12, pp. 187–192, 2002.
[6] Q. Miao, Z. Rong, T. Yang, and J. Fang, "Effects of degree correlation on the controllability of networks," *Physica A*, vol. 387, no. 24, pp. 6225–6230, 2008.
[7] L. F. R. Turci and E. E. N. Macau, "Performance of pinning-controlled synchronization," *Physical Review E*, vol. 84, no. 1, pp. 11 120–11 120, 2011.
[8] Y. Zou and G. Chen, "Choosing effective controlled nodes for scale-free network synchronization," *Physica A*, vol. 388, no. 14, pp. 2931–2940, 2009.
[9] H. Liu, X. Xu, J. Lu, G. Chen, and Z. Zeng, "Optimizing pinning control of complex dynamical networks based on spectral properties of grounded laplacian matrices," *IEEE Trans. Syst. Man Cybern. -Syst.*, vol. 51, no. 2, pp. 786–796, 2021.
[10] Z. H. Rong, L. Xiang, and L. L. Wen, "Pinning a complex network through the betweenness centrality strategy," in *IEEE*, 2009.
[11] Y. Y. Lu and X. F. Wang, "Pinning control of directed dynamical networks based on controlrank," *Int. J. Comput. Math.*, vol. 85, no. 8, pp. 1279–1286, 2008.
[12] L. Wang, H. P. Dai, H. Dong, Y. Y. Cao, and Y. X. Sun, "Adaptive synchronization of weighted complex dynamical networks through pinning," *Eur. Phys. J. B*, vol. 61, no. 3, pp. 335–342, 2008.

[13] W. Yu, G. Chen, Z. Wang, and W. Yang, "Ieee trans. syst. man cybern. part b-cybern." *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 6, pp. 1568–1577, 2009.

[14] D. P. Bertsekas and J. N. Tsitsiklis, "Parallel and distributed computation: numerical methods," 2003.

[15] H. J. Leblanc, H. Zhang, X. Koutsoukos, and S. Sundaram, "Resilient asymptotic consensus in robust networks," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 4, pp. 766–781, 2013.

[16] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[17] S. Sundaram and C. N. Hadjicostis, "Distributed function calculation via linear iterative strategies in the presence of malicious agents," *IEEE Trans. Autom. Control*, vol. 56, no. 7, pp. 1495–1508, 2011.

[18] A. Tahbaz-Salehi and A. Jadbabaie, "A necessary and sufficient condition for consensus over random networks," *IEEE Trans. Autom. Control*, vol. 53, no. 3, pp. 791–795, 2008.

[19] P. Barooah and J. P. Hespanha, "Graph effective resistance and distributed control: Spectral properties and applications," in *IEEE Conference on Decision & Control*, 2006.

[20] M. Pirani, E. M. Shahrivar, and S. Sundaram, "Coherence and convergence rate in networked dynamical systems," in *2015 54th IEEE Conference on Decision and Control (CDC)*. IEEE, 2015, pp. 968–973.

[21] W. Xia and M. Cao, "Analysis and applications of spectral properties of grounded laplacian matrices for directed networks," *Automatica*, vol. 80, pp. 10–16, 2017.

[22] A. Clark, B. Alomair, L. Bushnell, and R. Poovendran, "Leader selection in multi-agent systems for smooth convergence via fast mixing," in *Decision & Control*, 2012.

[23] A. Clark, L. Bushnell, and R. Poovendran, "Leader selection for minimizing convergence error in leader-follower systems: A supermodular optimization approach," in *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), 2012 10th International Symposium on*, 2012.

[24] B. Bamieh, M. Jovanović, P. Mitra, and S. Patterson, "Coherence in large-scale networks: Dimension-dependent limitations of local feedback," *IEEE Trans. Autom. Control*, vol. 57, pp. 2235–2249, 2012.

[25] I. Herman, D. Martinec, Z. Hurak, and M. Sebek, "Nonzero bound on fiedler eigenvalue causes exponential growth of h-infinity norm of vehicular platoon," *IEEE Trans. Autom. Control*, vol. 60, no. 8, pp. 2248–2253, 2015.

[26] E. Tegling, B. Bamieh, and D. F. Gayme, "The price of synchrony: Evaluating the resistive losses in synchronizing power networks," *IEEE Trans. Control Netw. Syst.*, vol. 2, no. 3, pp. 254–266, 2015.

[27] M. Pirani, E. M. Shahrivar, B. Fidan, and S. Sundaram, "Robustness of leader-follower networked dynamical systems," *IEEE Trans. Control Netw. Syst.*, vol. 5, no. 4, pp. 1752–1763, 2018.

[28] A. Clark, Q. Hou, L. Bushnell, and R. Poovendran, "Maximizing the smallest eigenvalue of a symmetric matrix: A submodular optimization approach," *Automatica*, vol. 95, pp. 446–454, 2018.

[29] J. Zhou and W. Tang, "Feature-embedded evolutionary algorithm for network optimization," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020.

[30] Bingjun, Wang, Hui, Liu, Jiangnqiao, Xu, Jiaqi, and Liu, "Pining control algorithm for complex networks," in *38th Chinese Control Conference*, 2019.

[31] C. R. Maccluer, "The many proofs and applications of perron's theorem," *SIAM Rev.*, 2000.

[32] R. B. Bapat, *Graphs and Matrices*. New York: Springer, 2010.

[33] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions-I," *Math. Program.*, vol. 14, no. 1, pp. 265–294, 1978.

[34] Y. Yi, L. Shan, H. Li, and Z. Zhang, "Biharmonic distance related centrality for edges in weighted networks," 07 2018, pp. 3620–3626.

[35] M. Siami, S. Bolouki, B. Bamieh, and N. Motee, "Centrality measures in linear consensus networks with structured network uncertainties," *IEEE Trans. Control Netw. Syst.*, vol. PP, pp. 1–1, 01 2017.

[36] J. Kang and H. Tong, "N2N: network derivative mining," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*. ACM, 2019, pp. 861–870.

[37] P. Mieghem, *Graph Spectra for Complex Networks*, 2011.

[38] Z. He, C. Yao, J. Yu, and M. Zhan, "Perturbation analysis and comparison of network synchronization methods," *Phys. Rev. E*, vol. 99, no. 5, 2019.

[39] A. Milanese, J. Sun, and T. Nishikawa, "Approximating spectral impact of structural perturbations in large networks," *Phys. Rev. E*, 2010.

[40] J. G. Restrepo, E. Ott, and B. R. Hunt, "Characterizing the dynamical importance of network nodes and links," *Physical Review Letters*, vol. 97, no. 9, p. 094102, 2006.

[41] J. Batson, D. A. Spielman, N. Srivastava, and S. H. Teng, "Spectral sparsification of graphs: Theory and algorithms," *Commun. ACM*, vol. 56, no. 8, pp. 87–94, 2013.

[42] D. A. Spielman and S.-H. Teng, "Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems," *SIAM J. Matrix Anal. Appl.*, vol. 35, no. 3, pp. 835–885, 2014.

[43] M. B. Cohen, R. Kyng, G. L. Miller, J. W. Pachocki, R. Peng, A. B. Rao, and S. C. Xu, "Solving SDD linear systems in nearly $m \log^{1/2} n$ time," in *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*. ACM, 2014, pp. 343–352.

[44] J. Kunegis, "Konect: the koblenz network collection," in *Proc. 22nd Int. Conf. World Wide Web*. ACM, 2013, pp. 1343–1350.

PLACE PHOTO HERE

Zhongzhi Zhang (M'19) received the B.Sc. degree in applied mathematics from Anhui University, Hefei, China, in 1997 and the Ph.D. degree in management science and engineering from Dalian University of Technology, Dalian, China, in 2006. From 2006 to 2008, he was a Post-Doctoral Research Fellow with Fudan University, Shanghai, China, where he is currently a Full Professor with the School of Computer Science. He has published over 140 papers in international journals or conferences. He has over 2800 ISI Web of Science citations with an H-index of 31 according to the Clarivate. He was one of the most cited Chinese researchers (Elsevier) in 2019. His current research interests include network science, graph data mining, social network analysis, spectral graph theory, and random walks.

Dr. Zhang was a recipient of the Excellent Doctoral Dissertation Award of Liaoning Province, China, in 2007, the Excellent Post-Doctor Award of Fudan University in 2008, the Shanghai Natural Science Award (third class) in 2013, and the Wilkes Award for the best paper published in The Computer Journal in 2019. He is a member of the IEEE.

PLACE PHOTO HERE

Guanrong Chen (M'89-SM'92-F'97) received the M.Sc. degree in computer science from Sun Yat-sen University, Guangzhou, China, in 1981, and the Ph.D. degree in applied mathematics from Texas A&M University, College Station, TX, USA, in 1987.

He was a Tenured Full Professor with the University of Houston, Houston, TX, USA. He has been a Chair Professor and the Founding Director of the Center for Chaos and Complex Networks, City University of Hong Kong, Hong Kong, since 2000.

Dr. Chen is a member of the Academia Europaea and a fellow of The World Academy of Sciences. He was the recipient of the 2011 Euler Gold Medal in Russia and the conferred Honorary Doctorate by the Saint Petersburg State University, Russia, in 2011, and by the University of Le Havre, Nomandie, France, in 2014. He is a Highly Cited Researcher in engineering since 2009 according to Thomson Reuters.