Annales Mathematicae et Informaticae
60 (2024) pp. 10–18
DOI: 10.33039/ami.2024.02.001
URL: https://ami.uni-eszterhazy.hu

New methods for maximizing the smallest eigenvalue of the grounded Laplacian matrix

Ahmad T. Anaqreh, Boglárka G.-Tóth, Tamás Vinkó

Department of Computational Optimization, Institute of Informatics,
University of Szeged, Hungary
{ahmad,boglarka,tvinko}@inf.u-szeged.hu

Abstract. Maximizing the smallest eigenvalue of the grounded Laplacian matrix is an NP-hard problem that involves identifying the Laplacian matrix's $(n-k) \times (n-k)$ principal submatrix obtained after removing k rows and corresponding columns. The challenge is to determine optimally the rows and columns to be deleted. Our proposed approach, motivated by the Gershgorin circle theorem, is used together with the degree centrality of the corresponding graph. Moreover, integer linear programming for the vertex cover problem has been employed as an additional method of solving the problem. The efficiency of the methods is demonstrated on real-world graphs.

Keywords: Laplacian matrix, grounded Laplacian matrix, eigenvalues

1. Introduction

The simplest way to represent graphs is their topological representation, where the graph is a set of nodes and edges. However, the spectral representation, such as Adjacency matrix or Laplacian matrix, can significantly help in describing the structural and functional behavior of the graph. Let $G = (V, E)$ be a simple, undirected graph with nodes set V and edges set E, where $|V| = n$ and $|E| = m$. For an edge (i, j) we consider that $(j, i) \in E$ for symmetry, but they count only one edge in total. The Adjacency matrix A is defined as

$$A_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Submitted: July 31, 2023
Accepted: February 6, 2024
Published online: February 9, 2024

Let d_i denote the degree of node $i \in V$, i.e., $d_i = \sum_j A_{ij}$ for all $i \in V$. The Laplacian matrix L of graph G is defined as follows:

$$L_{ij} = \begin{cases} d_i & \text{if } i = j, \\ -1 & \text{if } (i,j) \in E, \\ 0 & \text{otherwise,} \end{cases}$$

Note that the Laplacian matrix is a symmetric positive semidefinite matrix. It is well known that its n eigenvalues are non-negative real numbers, bounded by the double of the maximum vertex degree [1]. As a consequence, we have $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n \leq 2 \max_{i \in V} d_i$, where λ_i stands for the i-th eigenvalue of L.

Some applications of the Laplacian. According to Mohar *et al.* [12], the eigenvalues of the Laplacian matrix have their applications in diverse fields. One of the main applications is in graph theory, where the number of spanning trees of a graph G is determined by the multiplication of all non-zero eigenvalues of L [9]. Moreover, the sum of resistance distances over all node pairs can also be determined using the eigenvalues of the Laplacian matrix [6]. Another important implication of the Laplacian matrix is the Fiedler value [4], which corresponds to the second smallest eigenvalue (λ_2) and plays a crucial role in determining the connectivity of a graph. A graph is considered connected if its Fiedler value is greater than zero. Finally, the number of components in G is equal to the multiplicity of the 0 eigenvalue of L.

Grounded Laplacian. Let $G = (V, E)$, a simple undirected and connected graph, be given together with its Laplacian matrix L. The grounded Laplacian matrix $L(S)$, which was introduced in [11], is an $(n-k) \times (n-k)$ submatrix obtained by deleting k rows and their corresponding columns from the Laplacian matrix L, where $S \subset V$, $|S| = k$, $0 < k \ll n$. The smallest eigenvalue of $L(S)$ is denoted by $\lambda(S)$. Note that $L(S)$ is a symmetric positive definite matrix, thus all its eigenvalues are strictly positive real numbers. Hence, $\lambda(S) > 0$ holds.

Applications, complexity and algorithms. Without completeness, we mention some applications of $L(S)$. The value of the smallest eigenvalue $\lambda(S)$ of matrix $L(S)$ determines the convergence rate of a leader-follower networked dynamical system [13], as well as the effectiveness of pinning scheme of pinning control of complex dynamical networks [10], with large $\lambda(S)$ corresponding to fast convergence speed and good pinning control performance.

Finding $L(S)$ with the maximum possible $\lambda(S)$ has been shown to be an NP-hard problem [14]. Thus, solution methods based on heuristics are desired. The authors in [14] introduced two greedy-type algorithms. The first one, referred as the NAÏVE algorithm, involves k iterations. In each iteration, a candidate is chosen if adding it to set S maximizes $\lambda(S)$. The second algorithm, referred to as the FAST algorithm, evaluates a candidate node based on the sum of the eigenvalues of its adjacent nodes. The optimal candidate is chosen based on the maximum sum value. The eigenvalues for this computation are obtained from the eigenvector that corresponds to the smallest eigenvalue of a grounded Laplacian matrix, which is approximated using the SDDM solver [3] that determines the eigenvector without having to calculate the entire eigensystem.

2. Methodology

We propose two algorithms that differ from the approaches mentioned above. The first algorithm relies on the centrality of the nodes to select elements of set S, while the second algorithm is based on the vertex cover problem.

First approach. Our first method, denoted by Degree-G, is motivated by the well-known Gershgorin circle theorem [5]. The Gershgorin circle theorem provides bounds on the eigenvalues of a square matrix. Let B be a square matrix, with entries b_{ij}. For $i \in \{1, \dots, n\}$, let $R_i = \sum_{i \neq j} |b_{ij}|$. Let $D(b_{ii}, R_i) \subseteq \mathbb{C}$ be a closed circle centered at b_{ii} with radius R_i.

Theorem. *Every eigenvalue of B lies within at least one of the Gershgorin circles $D(b_{ii}, R_i)$.*

Figure 1 represents the Gershgorin circles of a Laplacian matrix. Note that we obtain the sharp, yet trivial, lower bound 0 on the eigenvalues of L.

$$L = \begin{pmatrix} 3 & -1 & -1 & -1 & 0 \\ -1 & 2 & 0 & -1 & 0 \\ -1 & 0 & 3 & -1 & -1 \\ -1 & -1 & -1 & 4 & -1 \\ 0 & 0 & -1 & -1 & 2 \end{pmatrix}$$

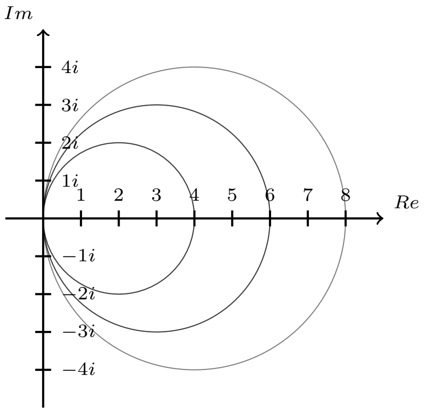


Figure 1. A Laplacian matrix (left) and its Gershgorin circles (right).

Before we give the details of our first approach, the concept of graph centralities needs to be briefly introduced. Given graph $G = (V, E)$, centrality is a function that assigns a non-negative real number to the nodes of G. Thus, upon calculating centrality values for G, it is possible to rank the nodes, which can be thought of as

Algorithm 1: Centrality-based Algorithm

```
1 $node\_cen = sort(centrality(V))$
2 for $i \in 1 \to k$ do
3     $remove(L, node\_cen[i])$
4 $compute(min\_eigen(L))$
```

assigning importance to the nodes. Traditionally, a larger centrality value indicates a higher importance of the node.

It is obvious that maximizing the lower bound on the smallest eigenvalue requires moving the circles further from the origin. Thus, the idea is to rank the nodes according to specific centrality, and then remove the corresponding row and column from the Laplacian matrix. The method is described in Algorithm 1. In this work, the degree centrality has been used.

As a simple demonstration, by applying the idea on the Laplacian matrix in Figure 1 with $k = 2$ we obtained $\lambda(S) = 1.27$, the output represented in Figure 2. In contrast, the greedy-type Naïve algorithm of [14] gives the set $S = \{2, 4\}$ for which we obtain $\lambda(S) = 1.2$. Note that none of these methods were able to obtain the optimal solution, that is $\lambda(S) = 1.47$ with $S = \{1, 5\}$, for this simple problem. Note that we ranked the nodes in ascending order for this method. We tried using the descending ranking as well, but overall it did not yield satisfactory results.

$$L = \begin{pmatrix} 3 & -1 & -1 \\ -1 & 3 & -1 \\ -1 & -1 & 4 \end{pmatrix}$$

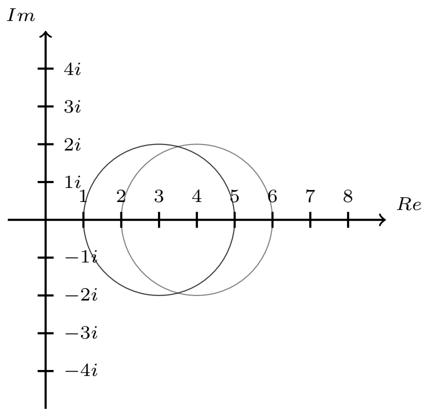


Figure 2. Illustration of the result of Algorithm 1 using degree centrality.

Second approach. The second method called Cover, also uses the Gershgorin circles, but it utilizes the so-called maximum k vertex cover problem as well. The maximum k vertex cover is based on the vertex cover problem [2], with the difference, that in the k vertex cover the search is for a set of k nodes that incident to the maximum number of edges of the graph rather than the minimum number of nodes that each edge in the graph is incident to. The integer linear programming model of the vertex cover is as follows:

$$\begin{gathered}\min \sum_{i\in V} x_i,\\ x_i + x_j \geq 1 \quad \forall (i,j) \in E,\\ x_i \in \{0,1\} \quad \forall i \in V.\end{gathered}$$

Note that the objective function represents the minimum number of nodes that incident to all the edges in the graph, where $x_i = 1$ are the covering nodes. The constraint requires that for each edge at least one of its endpoints should be a covering node.

The integer linear program of the maximum k vertex cover is defined as follows:

$$\begin{gathered}\max \sum_{j\in V} y_j,\\ \sum_{i\in V} x_i = k,\\ y_j \leq \sum_{\forall i\in V:(j,i)\in E} x_i \quad \forall j \in V,\\ k \in \mathbb{N}, \quad x_i, y_i \in \{0,1\},\ i = 1,\dots,n.\end{gathered}$$

Again, the variables x_i stand for the nodes that cover, by the edges, the maximum number of vertices in the graph, while y_i represents the vertices that are covered. The constraints ensure that only k vertices can be selected and that the value of y_i is 1 iff at least one of its adjacent nodes, x_i, is selected. Once we solve the above IP, we delete the rows and columns that correspond to the solution from the Laplacian matrix and then determine its smallest eigenvalue.

The maximum k vertex cover problem can have multiple solutions, so we thought that combining vertex cover and degree centrality could enhance our results. As a result, we modified the objective function in our linear program to the following:

$$\max \sum_{j\in V} y_j - \delta \sum_{j\in V} d_j x_j,$$

where δ is a small number so as to not change the main objective. For instance $\delta = 1/\sum_{j\in V} d_j$ can be chosen. This modification aims to select k nodes with the lowest degree to maximize the objective. The approach is denoted as Cover1.

Moreover, as the maximum k vertex cover problem can have multiple solutions, we utilized a method to explore the possibility of obtaining a better value for $\lambda(S)$ by the alternate solutions. The idea is to solve the linear program iteratively while including a constraint that prevents the solution from resembling previous ones. By checking various solutions, we can obtain different values for $\lambda(S)$ and select the one that gives the optimal result. This additional constraint is defined as follows for a given solution $S \subset V$ obtained before:

$$\sum_{i \in S} x_i \leq k - 1.$$

At each iteration, we need to ensure that the solution covers the maximum number of nodes, nc, that is, $\sum_{i \in V} y_i = nc$ must hold, otherwise, we need to stop the iteration. This approach is denoted as Cover2. The maximum number of iterations – and so the maximum number of alternative solutions – is fixed to be 100.

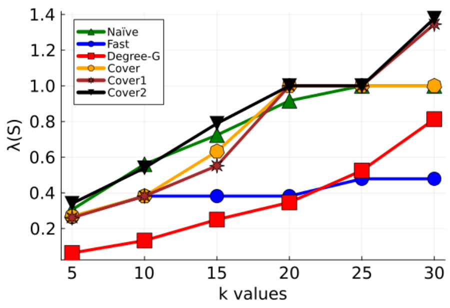
3. Numerical results

To demonstrate and compare the efficiency of the proposed methods we compare them with the two algorithms proposed by Wang *et al.* [14]. We implemented all the algorithms in Julia 1.7.0 using the package JuMP 0.22.1. As a solver, we used Gurobi 9.5.0., all on a computer with Intel Core i7-4600U CPU and 8GB RAM running Windows 10. Experiments were conducted on real-world networks, all of which are publicly accessible in KONECT [7], SNAP [8], and RWC at `http://tcs.uos.de/research/lip`.

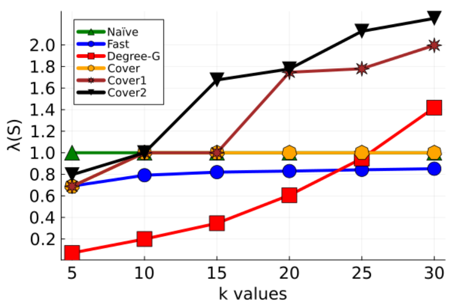
Results for different *k* values. Figure 3 shows the results achieved by applying the tested methods to various real-world graphs. The figure shows the smallest eigenvalue, $\lambda(S)$, for six different values of k using the discussed methods on selected graphs. The evaluated cases are connected with a solid line, representing an approximate value, since $\lambda(S)$ increases monotonically with k for each method. The higher the lines, the better the results are. The presented results show that method efficiencies vary from graph to graph, but there are some common features. It is obvious that the Degree-G and Fast methods are inferior to the Naïve and Cover methods. Interestingly, the Naïve method outperforms the other methods, although the Cover methods perform equally well or even better than Naïve at specific values of k.

Results for specific *k* values. Instead of using ad-hoc k values, we utilized the vertex cover integer program to obtain the lowest value of k that can provide a sufficient value for certainly increasing the lower bound of $\lambda(S)$. Table 1, displays the corresponding values of k and $\lambda(S)$ for various real-world graphs using the different methods we have discussed. The Naïve and Cover methods are clearly more effective than Degree-G and Fast methods. However, the Cover methods perform equally or even better than Naïve in all cases.

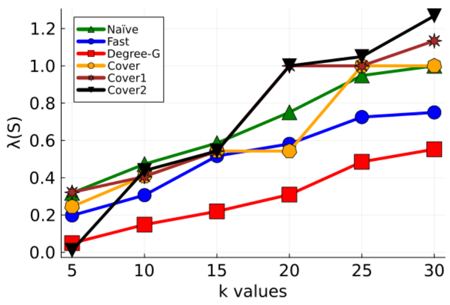
Additionally, we performed a time comparison between the different methods. Table 2 presents the runtime for $k = 5$ of each method with a time limit of one hour. The results demonstrate that the Degree-G and Fast methods are notably faster than the Naïve and Cover methods. It is clear that the Naïve and Cover algorithms exceeded the time limit for graphs containing thousands of nodes and edges, and it is evident that the Cover methods are generally faster than Naïve in completing the task.



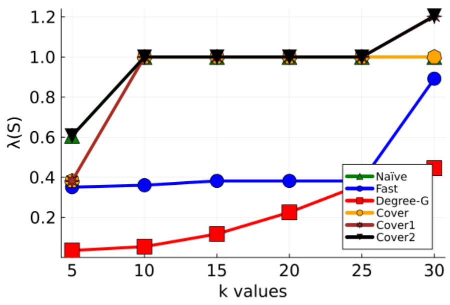
(a) Prison



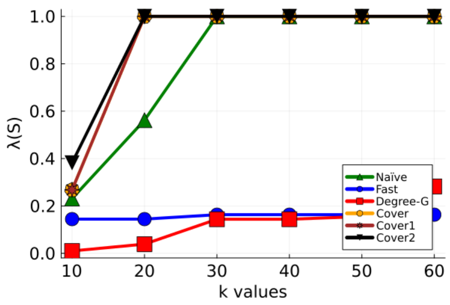
(b) Huck



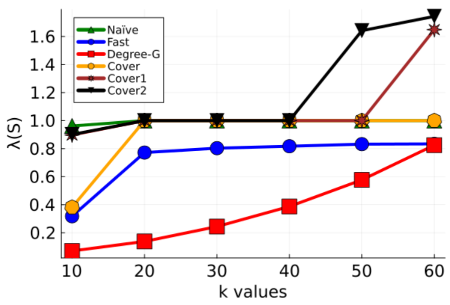
(c) Sanjuansur



(d) Jean



(e) Sfi



(f) Anna

Figure 3. Values of $\lambda(S)$ obtained by the algorithms for different k values.

Table 1. Values of $\lambda(S)$ for k value obtained from vertex cover.

Graph	N	M	k	Naïve	Fast	Degree-G	Cover	Cover1	Cover2
Prison	67	142	41	1	0.48	1.97	1.59	1.63	2.38
Huck	69	297	44	1	1	1.7	1	2.48	3.5
Sanjuansur	75	144	40	1.59	0.84	0.95	1.29	1.27	1.59
Jean	77	254	42	1	0.37	0.67	1	1.24	1.24
David	87	406	51	1	1	2.45	3.44	2.65	2.77
ieeebus	118	179	61	1	0.59	0.73	1	1.06	1.2
Sfi	118	200	53	1	0.27	0.24	1	1	1
Anna	138	493	58	1	0.83	0.87	1	1.65	1.65
Usair	332	2126	149	1	0.74	1	1	1.59	1.59
494bus	494	586	216	0.38	0.07	0.14	1	1	1
average				0.99	0.62	1.07	1.33	1.56	1.79

Table 2. The time (in seconds) for $k = 5$ to compute $\lambda(S)$.

Graph	N	M	Naïve	Fast	Degree-G	Cover	Cover1	Cover2
Prison	67	142	0.63	0.005	0.002	0.031	0.028	1.139
Huck	69	297	0.785	0.008	0.003	0.036	0.038	5.686
Sanjuansur	75	144	1	0.008	0.003	0.031	0.036	7.712
Jean	77	254	1.02	0.008	0.004	0.035	0.034	13.615
David	87	406	1.418	0.014	0.008	0.041	0.041	2.213
ieeebus	118	179	2.868	0.011	0.003	0.043	0.042	9.127
Sfi	118	200	2.708	0.011	0.008	0.041	0.041	7.724
Anna	138	493	3.88	0.016	0.014	0.067	0.066	14.543
Usair	332	2126	29.773	0.049	0.034	0.651	0.513	21.856
494bus	494	586	102.975	0.057	0.051	0.382	0.323	15.712
Email-Univ	1133	5451	1589.445	0.55	0.531	13.632	17.497	104.342
Routers-RF	2113	6632	>3600	1.448	1.515	55.288	57.114	272.99
US-Grid	4941	6594	>3600	14.255	15.551	312.706	315.643	1947.852
WHOIS	7476	56943	>3600	50.343	50.425	>3600	>3600	>3600
PGP	10680	24340	>3600	140.943	143.747	>3600	>3600	>3600
average			1075.789	13.852	14.126	505.532	506.094	641.634

4. Conclusion

Given the fact that maximizing the smallest eigenvalue of the grounded Laplacian matrix is NP-hard, it is desirable to establish efficient algorithms that can provide solutions of acceptable quality and reasonable running time. We have proposed two approaches and experimentally shown that, compared to the algorithms in the literature, these algorithms are competitive. The covering methods provide a better solution in a shorter time as the Naïve method, while Degree-G is almost as fast as the Fast method giving competitive results, especially for higher k values.

Acknowledgements. The research leading to these results has received funding from the national project TKP2021-NVA-09. Project no. TKP2021-NVA-09 has been implemented with the support provided by the Ministry of Innovation and Technology of Hungary from the National Research, Development, and Innovation Fund, financed under the TKP2021-NVA funding scheme. The work was also supported by the grant SNN-135643 of the National Research, Development, and Innovation Office, Hungary.

References

[1] W. Anderson, T. Morley: *Eigenvalues of the Laplacian of a Graph*, Linear and Multilinear Algebra 18.2 (1985), pp. 141–145, doi: 10.1080/03081088508817681.

[2] J. Chen, I. A. Kanj, W. Jia: *Vertex cover: further observations and further improvements*, Journal of Algorithms 41.2 (2001), pp. 280–301, doi: 10.1006/jagm.2001.1186.

[3] M. B. Cohen, R. Kyng, G. L. Miller, J. W. Pachocki, R. Peng, A. B. Rao, S. C. Xu: *Solving SDD linear systems in nearly m log1/2 n time*, in: Proceedings of the forty-sixth annual ACM symposium on Theory of computing, 2014, pp. 343–352, doi: 10.1145/2591796.2591833.

[4] M. Fiedler: *Algebraic connectivity of graphs*, Czechoslovak mathematical journal 23.2 (1973), pp. 298–305, doi: 10.21136/CMJ.1973.101168.

[5] G. H. Golub, C. F. Van Loan: *Matrix computations*, Johns Hopkins University Press, 2013.

[6] D. J. Klein, M. Randić: *Resistance distance*, Journal of mathematical chemistry 12 (1993), pp. 81–95, doi: 10.1007/BF01164627.

[7] J. Kunegis: *Konect: the koblenz network collection*, in: Proceedings of the 22nd international conference on world wide web, 2013, pp. 1343–1350, doi: 10.1145/2487788.2488173.

[8] J. Leskovec, R. Sosič: *Snap: A general-purpose network analysis and graph-mining library*, ACM Transactions on Intelligent Systems and Technology (TIST) 8.1 (2016), pp. 1–20, doi: 10.1145/2898361.

[9] H. Li, S. Patterson, Y. Yi, Z. Zhang: *Maximizing the number of spanning trees in a connected graph*, IEEE Transactions on Information Theory 66.2 (2019), pp. 1248–1260, doi: 10.1109/TIT.2019.2940263.

[10] H. Liu, X. Xu, J.-A. Lu, G. Chen, Z. Zeng: *Optimizing pinning control of complex dynamical networks based on spectral properties of grounded Laplacian matrices*, IEEE Transactions on Systems, Man, and Cybernetics: Systems 51.2 (2018), pp. 786–796, doi: 10.1109/TSMC.2018.2882620.

[11] U. Miekkala: *Graph properties for splitting with grounded Laplacian matrices*, BIT Numerical Mathematics 33.3 (1993), pp. 485–495, doi: 10.1007/BF01990530.

[12] B. Mohar: *Some applications of Laplace eigenvalues of graphs*, Springer, 1997, doi: 10.1007/978-94-015-8937-6_6.

[13] A. Rahmani, M. Ji, M. Mesbahi, M. Egerstedt: *Controllability of multi-agent systems from a graph-theoretic perspective*, SIAM Journal on Control and Optimization 48.1 (2009), pp. 162–186, doi: 10.1137/060674909.

[14] R. Wang, X. Zhou, W. Li, Z. Zhang: *Maximizing the Smallest Eigenvalue of Grounded Laplacian Matrix*, arXiv preprint arXiv:2110.12576 (2021).