

Brute-force search and Warshall algorithms for matrix-weighted graphs

Minh Hoang Trinh*

Tayan Polytechnic, Tan Hoi, O Dien, Hanoi, 10000, Vietnam

Hyo-Sung Ahn

Gwangju Institute of Science and Technology, 123 Cheomdangwagi-ro, Buk-gu, Gwangju, 500-712, Republic of Korea

Abstract

Although research on the control of networked systems has grown considerably, graph-theoretic and algorithmic studies on matrix-weighted graphs remain limited. To bridge this gap in the literature, this work introduces two algorithms-the brute-force search and the Warshall algorithm-for determining connectedness and clustering in undirected matrix-weighted graphs. The proposed algorithms, which are derived from a sufficient condition for connectedness, emphasize a key distinction between matrix-weighted and scalar-weighted graphs. While the existence of a path between two vertices guarantees connectedness in scalar-weighted graphs, connectedness in matrix-weighted graphs is a collective contribution of all paths joining the two vertices. Proofs of correctness and numerical examples are provided to illustrate and demonstrate the effectiveness of the algorithms.

Keywords: graph and matrices, graph theory, algorithms, multiagent systems.

2008 MSC: 05C50, 05C85, 68R10, 94C14

1. Introduction

A matrix-weighted graph is a multidimensional generalization of a classical graph, with multidimensional state vectors associate with vertices and positive semidefinite matrices associated with edges. Notably, matrix-weighted graphs is capable of representing intra-and cross-layer interactions in multidimensional networks (Kivelä et al., 2014; Trinh et al., 2024; Vu et al., 2025), as well as clustering phenomena arising under the consensus algorithm (Trinh et al., 2018).

Several properties of matrix-weighted graphs with positive definite matrix weights have been introduced in (Barooah and Hespanha, 2005, 2008). The authors in (Trinh et al., 2018) proposed matrix-weighted graph and consensus with positive semidefinite matrix weights. Physical interpretations of matrix-weighted graphs in circuit theory (Barooah and Hespanha, 2005; Atik and Kannan, 2019; Mahato and Kannan, 2023), and applications in consensus and clustering dynamics (Trinh et al., 2017; Tran et al., 2021; Miao and Su, 2021), multidimensional opinion

*Corresponding author. E-mail: minhtrinh@ieee.org

dynamics (Ahn et al., 2020; Pan et al., 2018), privacy enhancement consensus protocols (Pan et al., 2025), formation control and network localization (Barooah and Hespanha, 2008; Zhao and Zelazo, 2016) were also considered in the literature. The author in Hansen (2021) examines various abstract properties of the matrix-weighted Laplacian via cellular sheaf theory.

Although interest in matrix-weighted networks has grown, algorithmic methods for their analysis remain limited. A main difficulty in developing algorithms for matrix-weighted graph lies in the semidefiniteness of matrix weights: the existence of an edge does not ensure that two vertices are connected in every dimension, and topological connectedness does not ensure the graph to be connected. To further exploit this point, in traditional graphs, determining connectedness can be simply conducted via a depth-first search algorithm since the existence of a path guarantees connectedness between any two vertices. The Warshall algorithm (Warshall, 1962) offers another approach to determining connectedness in traditional graph using the adjacent matrix and successive boolean logic multiplications. However, the existing above algorithms cannot be directly applied to determine clusters (or connected components) in matrix-weighted graphs. Until now, the algorithm presented in (Trinh et al., 2018) appears to be the only one available. Mainly, the algorithm in (Trinh et al., 2018) partitions a matrix-weighted graph into subgraphs according to its positive trees and subsequently combines these subgraphs based on algebraic conditions along the paths connecting them.

Given that connectivity is a fundamental property in graph theory, this paper presents two heuristic algorithms to determine whether a matrix-weighted graph is connected and, if not, to produce a reasonable partition of its vertices into clusters. We first prove a sufficient condition for two vertices to be *connected*, i.e., the kernel of the matrix-weighted Laplacian contains only vectors whose subvectors associated with these vertices are always equal. Our proof relies on the notions of the parallel and series matrix additions introduced in (Anderson and Duffin, 1969). The connectivity assessment between two vertices is divided into smaller kernel determination problems for each simple paths joining them, and these path kernels are then combined to derive a conclusion for the assessment. The divide-and-conquer strategy expands an algorithm in (Trinh et al., 2018), which was proposed to determine all clusters from an initial partition of a matrix-weighted graph based on positive trees. Building on the connectivity condition, we introduce brute-force search and Warshall algorithms for connectedness and clustering in undirected matrix-weighted graphs. The series and parallel matrix sums act as the OR and AND (“ $\vee$ ” and “ $\wedge$ ”) logic operators in these algorithms, respectively. In designing the Warshall algorithm, the “ $\vee$ ” and “ $\wedge$ ” operators are generalized to block matrices, which enables a compact description of kernel computations in parallel. It is worth noting that to reduce the computational complexity, a novel decision operator is also defined and employed in each immediate step of the Warshall algorithm. Although the Warshall algorithm was introduced in (Trinh and Ahn, 2025), there has been no formal analysis. In this paper, we provide proof of correctness, worst-case computational complexities of both algorithms, and demonstrate the use of the algorithms

The remaining sections of this paper are outlined as follows. Section 2 provides essentials of the parallel and series additions of symmetric positive semidefinite matrices and matrix-weighted graphs. Brute-force search and Warshall algorithms are considered in Section 3. Section 4 contains several examples using the algorithms, and Section 5 concludes the paper.

2. Preliminaries

2.1. Parallel and series additions of symmetric positive semidefinite matrices

In this subsection, the matrices $\mathbf{A}_k \in \mathbb{R}^{d \times d}$, $k = 1, 2, \dots, N$, $d \geq 2$, are assumed to be symmetric ($\mathbf{A}_k^\top = \mathbf{A}_k$) and positive semidefinite (i.e., $\forall \mathbf{x} \in \mathbb{R}^d$, $\mathbf{x}^\top \mathbf{A}_k \mathbf{x} \geq 0$). We use $\mathbf{0}_d$ and $\mathbf{I}_d$ to denote the zero and the identity matrices of dimension $d \times d$.

The *series addition*, or the “ $\vee$ ” operator, of two symmetric positive semidefinite matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ is defined as $\mathbf{A}_1 \vee \mathbf{A}_2 = \mathbf{A}_1 + \mathbf{A}_2$. Similarly, $\bigvee_{i=1}^N \mathbf{A}_k = ((\mathbf{A}_1 \vee \mathbf{A}_2) \vee \dots) \vee \mathbf{A}_N$ implies that the “ $\vee$ ” operators are sequentially applied.

The *parallel addition*, or the “ $\wedge$ ” operator, of two matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ is defined as

$$\mathbf{A}_1 \wedge \mathbf{A}_2 = \mathbf{A}_1 (\mathbf{A}_1 + \mathbf{A}_2)^\dagger \mathbf{A}_2, \quad (1)$$

where $(\mathbf{A}_1 + \mathbf{A}_2)^\dagger$ denotes the Moore-Penrose pseudo inverse of $\mathbf{A}_1 + \mathbf{A}_2$. In a similar manner, we write $\bigwedge_{k=1}^N \mathbf{A}_k = ((\mathbf{A}_1 \wedge \mathbf{A}_2) \wedge \dots) \wedge \mathbf{A}_N$ to imply that the “ $\wedge$ ” operators are sequentially applied.

We have the following lemmas on the “ $\vee$ ” and “ $\wedge$ ” operators ([Anderson and Duffin, 1969](#)).

Lemma 1 (The “ $\vee$ ” operator). *The “ $\vee$ ” operator of $d \times d$ symmetric positive semidefinite matrices satisfies the following properties:*

- i. *Symmetric positive semidefinite:* $\mathbf{A}_1 \vee \mathbf{A}_2 = (\mathbf{A}_1 \vee \mathbf{A}_2)^\top \geq 0$,
- ii. *Commutativity:* $\mathbf{A}_1 \vee \mathbf{A}_2 = \mathbf{A}_2 \vee \mathbf{A}_1$,
- iii. *Kernel space:* $\ker(\mathbf{A}_1 \vee \mathbf{A}_2) = \ker(\mathbf{A}_1) \cap \ker(\mathbf{A}_2)$,
- iv. *Associativity:* $(\mathbf{A}_1 \vee \mathbf{A}_2) \vee \mathbf{A}_3 = \mathbf{A}_1 \vee (\mathbf{A}_2 \vee \mathbf{A}_3) = \mathbf{A}_1 \vee \mathbf{A}_2 \vee \mathbf{A}_3$.

Lemma 2 (The “ $\wedge$ ” operator). *The “ $\wedge$ ” operator of $d \times d$ symmetric positive semidefinite matrices satisfies the following properties:*

- i. *Symmetric positive semidefinite:* $\mathbf{A}_1 \wedge \mathbf{A}_2 = (\mathbf{A}_1 \wedge \mathbf{A}_2)^\top \geq 0$,
- ii. *Commutativity:* $\mathbf{A}_1 \wedge \mathbf{A}_2 = \mathbf{A}_2 \wedge \mathbf{A}_1$,
- iii. *Kernel space:* $\ker(\mathbf{A}_1 \wedge \mathbf{A}_2) = \ker(\mathbf{A}_1) \cup \ker(\mathbf{A}_2)$,
- iv. *Associativity:* $(\mathbf{A}_1 \wedge \mathbf{A}_2) \wedge \mathbf{A}_3 = \mathbf{A}_1 \wedge (\mathbf{A}_2 \wedge \mathbf{A}_3) = \mathbf{A}_1 \wedge \mathbf{A}_2 \wedge \mathbf{A}_3$.

Specially, we can write $\ker(\mathbf{A} \wedge \mathbf{A}) = \ker(\mathbf{A})$, $\ker(\mathbf{A} \wedge \mathbf{0}_d) = \mathbb{R}^d$, $\ker(\mathbf{A} \vee \mathbf{A}) = \ker(\mathbf{A}) = \ker(\mathbf{A} \vee \mathbf{0}_d)$.

The next lemma, whose proof is given in [Appendix A.1](#), is about expressions involving both operators.

Lemma 3 (Distributivity of “ $\wedge$ ” and “ $\vee$ ” operators). *The following statements hold*

- i. $\ker((\mathbf{A}_1 \wedge \mathbf{A}_2) \vee \mathbf{A}_3) = \ker(\mathbf{A}_1 \vee \mathbf{A}_3) \cap \ker(\mathbf{A}_1 \vee \mathbf{A}_2)$,
- ii. $\ker((\mathbf{A}_1 \vee \mathbf{A}_2) \wedge \mathbf{A}_3) = \ker(\mathbf{A}_1 \wedge \mathbf{A}_3) \cup \ker(\mathbf{A}_1 \wedge \mathbf{A}_2)$.

2.2. Matrix-weighted graphs

An undirected matrix-weighted graph (Trinh et al., 2018) is defined by $G = (V, E, W)$, where $V = \{v_1, \dots, v_n\}$ is the set of $|V| = n$ vertices, $E \subset V \times V$ is the set of $|E| = m$ edges, and $W = \{\mathbf{A}_{ij} \in \mathbb{R}^{d \times d} | (v_j, v_i) \in E\}$ is the set of matrix weights. We assume G does not have self-loops (edges connecting the same vertices) and multi-edge (multiple edges connecting a pair of vertices). The matrix weights are symmetric positive semidefinite, and as the graph is undirected, the matrix weights satisfy $\mathbf{A}_{ij} = \mathbf{A}_{ji} = \mathbf{A}_{ij}^\top$ for all $(v_j, v_i) \in E$. If an edge (v_j, v_i) has $\mathbf{A}_{ij}$ positive definite, then it is called a positive definite edge. Otherwise, the edge $(v_j, v_i) \in E$ is referred to as a positive semidefinite edge.

If $(v_j, v_i) \in E$, then two vertices v_i and v_j are adjacent to each other. The neighbor set of a vertex v_i can be defined as $\mathcal{N}_i = \{v_j \in V | (v_i, v_j) \in E\}$. A *complete graph* has $\mathcal{N}_i = V \setminus \{v_i\}, \forall i \in V$. A *path* in G is defined as a sequences of edges joining adjacent vertices in G . For example, $\mathcal{P} = v_{i1}v_{i2} \dots v_{il}$ contains edges $(v_{ik}, v_{i,k+1}) \in E$ connects the start vertex v_{i1} to the end vertex v_{il} . The length of $\mathcal{P}$ equals the number of edges in $\mathcal{P}$, and in this example, $|\mathcal{P}| = l - 1$. The path $\mathcal{P}$ is a circuit if and only if it contains no repeated edges, and a simple path if and only if it contains no repeated vertices. A *cycle* is a path with the same starting and ending vertices.

Corresponding to the matrix-weighted graph G , we can define the *topological graph* (V, E) . The matrix-weighted graph G is *topologically connected* if and only if for any pairs of vertices in V , there exists a path in (V, E) joining them. The concept of *connectedness* in a matrix-weighted graph, however, requires both topological connectedness and algebraic conditions related to the matrix-valued weights. We label the edges in E as $e_1, \dots, e_m$ and for each edge $e_k = (v_i, v_j) \in E$, a vertex is chosen as the start vertex and the other is the end vertex of the edge. Correspondingly, the incident matrix $\mathbf{H} = [h_{ki}] \in \mathbb{R}^{m \times n}$ of the graph can be defined with

$$h_{ki} = \begin{cases} 1, & \text{if vertex } v_i \text{ is the starting vertex of } e_k, \\ -1, & \text{if vertex } v_i \text{ is the ending vertex of } e_k, \\ 0, & \text{otherwise.} \end{cases}$$

Let $\mathbf{D}_i = \sum_{j \in \mathcal{N}_i} \mathbf{A}_{ij}$ be the degree of a vertex $v_i \in V$. The matrix-weighted adjacency matrix and the degree matrix of G are correspondingly given as $\mathbf{A} = [\mathbf{A}_{ij}] \in \mathbb{R}^{dn \times dn}$ and $\mathbf{D} = \text{blkdiag}(\mathbf{D}_1, \dots, \mathbf{D}_n)$. Then, we can define the matrix-weighted Laplacian of G as $\mathbf{L} = \mathbf{D} - \mathbf{A}$. The matrix-weighted Laplacian can also be expressed as $\mathbf{L} = (\mathbf{H}^\top \otimes \mathbf{I}_d)\mathbf{W}(\mathbf{H} \otimes \mathbf{I}_d)$, where “ $\otimes$ ” denotes the Kronecker product and $\mathbf{W} = \text{blkdiag}(\dots, \mathbf{A}_{ij}, \dots) \in \mathbb{R}^{dm \times dm}$ is the block diagonal matrix with matrices $\mathbf{A}_{ij}$ in the main diagonal and in the same order as we label the edges.

The connectivity of a matrix-weighted graph is defined based on its corresponding matrix-weighted Laplacian.

Definition 1 (Connectedness/Clustering). *Let G be an undirected matrix-weighted graph with the matrix-weighted Laplacian $\mathbf{L}$. Then, G is connected if and only if $\text{rank}(\mathbf{L}) = dn - d$. Otherwise, the graph G is clustering.*

Clearly, topologically connectedness is necessary for a matrix-weighted graph to be connected. Thus, in this work, all considered matrix-weighted graphs are assumed to be topologically connected.

3. Algorithms

In this section, we propose brute-force search and Warshall algorithms for determining connectedness and clustering in a given matrix-weighted graph. The parallel and series matrix addi-

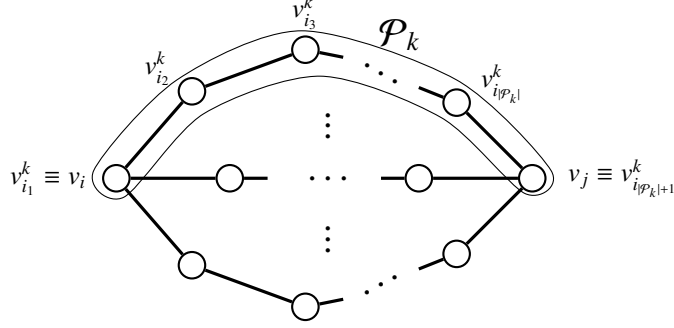


Figure 1: The path $\mathcal{P}_k = v_{i_1}^k v_{i_2}^k \dots v_{i_{|P_k|+1}}^k$ joins two vertices v_i and v_j .

tions offer quantitative tools to evaluate the connectivity.

3.1. Brute-force search algorithm

For each pair of vertices $v_i \neq v_j$, $v_i, v_j \in V$, let

$$\mathcal{S}_{ij}^\infty = \{\mathcal{P}_k = v_1^k \dots v_{|P_k|+1}^k \mid v_1^k = v_i, v_{|P_k|+1}^k = v_j\}, \quad (2)$$

$$\mathcal{S}_{ij} = \{\mathcal{P}_k \in \mathcal{S}_{ij}^\infty \mid \mathcal{P}_k \text{ is simple}\}, \quad (3)$$

be respectively the set of all paths from v_i to v_j and the set of all simple paths from v_i to v_j .

For each path $\mathcal{P}_k$ (see Fig. 1 for an illustration), we define

$$\ker(\mathcal{P}_k) \triangleq \bigcup_{j=1}^{|\mathcal{P}_k|} \ker(\mathbf{A}_{v_j^k v_{j+1}^k}) = \ker\left(\bigwedge_{j=1}^{|\mathcal{P}_k|} \mathbf{A}_{v_j^k v_{j+1}^k}\right). \quad (4)$$

By sorting the elements in the sets $\mathcal{S}_{ij}$ and $\mathcal{S}_{ij}^\infty$ by their lengths, we can decompose them into subsets $\mathcal{S}_{ij}^t$, $t = 1, 2, \dots$, containing all paths of length t from v_i to v_j . Since $\mathcal{S}_{ij}$ only admits simple paths, the lengths of every element in $\mathcal{S}_{ij}$ is at most $n - 1$.

Let

$$\ker(\mathcal{S}_{ij}^t) = \bigcap_{\mathcal{P}_k \in \mathcal{S}_{ij}^t} \ker(\mathcal{P}_k) = \ker\left(\bigvee_{k=1}^{|\mathcal{S}_{ij}^t|} \bigwedge_{j=1}^t \mathbf{A}_{v_j^k v_{j+1}^k}\right), \quad (5)$$

$$\ker(\mathcal{S}_{ij}) = \bigcap_{t=1}^{n-1} \ker(\mathcal{S}_{ij}^t), \quad \ker(\mathcal{S}_{ij}^\infty) = \bigcap_{t=1}^{\infty} \ker(\mathcal{S}_{ij}^t). \quad (6)$$

Consider an arbitrary path $\tilde{\mathcal{P}} \in \mathcal{S}_{ij}^\infty$ joining v_i and v_j of length longer than n . Then, $\tilde{\mathcal{P}}$ must visit some vertices in V more than one time before ending at v_j . Thus, $\tilde{\mathcal{P}}$ must contain a simple path of length t ($1 \leq t \leq n - 1$) in $\tilde{\mathcal{P}}$ after removing all edges belonging to cycles joining the repeated vertices in $\tilde{\mathcal{P}}$. Due to the commutative and associative properties of the wedge operator, we can write

$$\ker(\tilde{\mathcal{P}}) = \ker(\mathcal{P}) \cup \ker(E(\tilde{\mathcal{P}}) \setminus E(\mathcal{P})). \quad (7)$$

It follows that $\ker(\mathcal{P}) \subseteq \ker(\tilde{\mathcal{P}})$, and $\ker(\mathcal{S}_{ij}^t) = \ker(\mathcal{S}_{ij}^t) \cap \ker(\tilde{\mathcal{P}})$. Therefore,

$$\ker(\mathcal{S}_{ij}^\infty) = \bigcap_{t=1}^{\infty} \ker(\mathcal{S}_{ij}^t) = \bigcap_{t=1}^{n-1} \ker(\mathcal{S}_{ij}^t) = \ker(\mathcal{S}_{ij}). \quad (8)$$

Based on (8), we have the following definition.

Definition 2. Two vertices v_i and v_j belong to a same cluster if and only if for all vectors $\mathbf{x} = [\mathbf{x}_1^\top, \dots, \mathbf{x}_n^\top]^\top \in \mathbb{R}^{dn}$ in $\ker(\mathbf{L})$, there holds $\mathbf{x}_i = \mathbf{x}_j$.

The following theorem provides a sufficient condition for two vertices to belong to a cluster and a matrix weighted graph to be connected.

Theorem 1. Consider an undirected matrix-weighted graph $G = (V, E, W)$ with the matrix-weighted Laplacian $\mathbf{L}$. The following claims hold.

i. Two vertices $v_i, v_j \in V$ belong to a same cluster if

$$\dim(\ker(\mathcal{S}_{ij})) = 0. \quad (9)$$

ii. The graph is connected if Eq. (9) holds for all pairs of distinct vertices $v_i, v_j \in V$.

Proof. i. Suppose that there are $|\mathcal{S}_{ij}| = q$ different paths labeled as $\mathcal{P}_1, \dots, \mathcal{P}_q$ in the set $\mathcal{S}_{ij}$.

Consider a path $\mathcal{P} = v_{i_1} v_{i_2} \dots v_{i_{|\mathcal{P}|}} v_{i_{|\mathcal{P}|+1}} \in \mathcal{S}_{ij}$, with $i_1 \equiv i$ and $i_{|\mathcal{P}|+1} \equiv j$. Consider an arbitrary vector $\mathbf{x} = [\mathbf{x}_1^\top, \dots, \mathbf{x}_n^\top]^\top \in \mathbb{R}^{dn}$, where each $\mathbf{x}_i \in \mathbb{R}^d$ is associated with $v_i \in V$ such that $\mathbf{L}\mathbf{x} = \mathbf{0}$.

Since $\ker(\mathbf{L}) = \ker((\mathbf{H}^\top \otimes \mathbf{I}_d)\mathbf{W}(\mathbf{H} \otimes \mathbf{I}_d)) = \ker(\mathbf{W}(\mathbf{H} \otimes \mathbf{I}_d))$, the restriction of the equation $\mathbf{L}\mathbf{x} = \mathbf{0}_{dn}$ to the subgraph $\mathcal{P}$ is (Zelazo and Mesbahi, 2010)

$$\begin{bmatrix} \mathbf{A}_{i_1 i_2} & & & \\ & \mathbf{A}_{i_2 i_3} & & \\ & & \ddots & \\ & & & \mathbf{A}_{i_{|\mathcal{P}|} i_{|\mathcal{P}|+1}} \end{bmatrix} (\mathbf{H}_{\mathcal{P}} \otimes \mathbf{I}_d) \mathbf{x} = \mathbf{0}_{d|\mathcal{P}|}, \quad (10)$$

where $\mathbf{H}_{\mathcal{P}}$ is the incident matrix corresponding to the path graph $\mathcal{P}$. Equivalently,

$$\mathbf{A}_{i_k i_{k+1}} (\mathbf{x}_{i_k} - \mathbf{x}_{i_{k+1}}) = \mathbf{0}_d, \quad (11)$$

for $k = 1, \dots, |\mathcal{P}|$. It follows that

$$\mathbf{x}_{i_2} = \mathbf{x}_{i_1} + (\mathbf{I}_d - \mathbf{A}_{i_1 i_2}^\dagger \mathbf{A}_{i_1 i_2}) \mathbf{y}_1, \quad (12a)$$

$\vdots$

$$\mathbf{x}_{i_{|\mathcal{P}|+1}} = \mathbf{x}_{i_{|\mathcal{P}|}} + (\mathbf{I}_d - \mathbf{A}_{i_{|\mathcal{P}|} i_{|\mathcal{P}|+1}}^\dagger \mathbf{A}_{i_{|\mathcal{P}|} i_{|\mathcal{P}|+1}}) \mathbf{y}_{|\mathcal{P}|}, \quad (12b)$$

for some arbitrary vectors $\mathbf{y}_k \in \mathbb{R}^d$, $k = 1, \dots, |\mathcal{P}|$. Summing these equations (12) side by side and eliminating common terms from both sides, we obtain

$$\mathbf{x}_j - \mathbf{x}_i = \mathbf{x}_{i_{|\mathcal{P}|+1}} - \mathbf{x}_{i_1} = \sum_{r=1}^{|\mathcal{P}|} (\mathbf{I}_d - \mathbf{A}_{i_r i_{r+1}}^\dagger \mathbf{A}_{i_r i_{r+1}}) \mathbf{y}_r. \quad (13)$$

Since $(\mathbf{I}_d - \mathbf{A}_{i_r i_{r+1}} \mathbf{A}_{i_r i_{r+1}}^\dagger) \mathbf{y}_r \in \ker(\mathbf{A}_{i_r i_{r+1}})$, $\forall r = 1, \dots, |\mathcal{P}|$, it follows that¹

$$(\mathbf{x}_j - \mathbf{x}_i) \in \bigcup_{k=1}^{|\mathcal{P}|} \ker(\mathbf{A}_{i_k i_{k+1}}) = \ker(\mathcal{P}). \quad (14)$$

As there are q paths $\mathcal{P}_k$ in $\mathcal{S}_{ij}$, by aggregating q equations as in (14), and using our assumption on $\ker(\mathcal{S}_{ij})$, we have

$$(\mathbf{x}_j - \mathbf{x}_i) \in \bigcap_{k=1}^q \ker(\mathcal{P}_k) = \bigcap_{k=1}^q \ker(\mathcal{P}_k) = \{\mathbf{0}_d\}, \quad (15)$$

and this implies that $\mathbf{x}_i = \mathbf{x}_j$.

Therefore, if $\dim(\bigcap_{k=1}^q \ker(\mathcal{P}_k)) = 0$, the kernel of $\mathbf{L}$ contains only vectors $\mathbf{x} = [\mathbf{x}_1^\top, \dots, \mathbf{x}_n^\top]^\top$ satisfying $\mathbf{x}_i = \mathbf{x}_j$, i.e., two vertices v_i, v_j belong to a same cluster.

ii. This claim follows immediately from (i). $\square$

Remark 1. It will be useful to have an intuition of the theoretical result in Theorem 1, particularly on how connectivity between two vertices in a matrix-weighted graph works. Let four

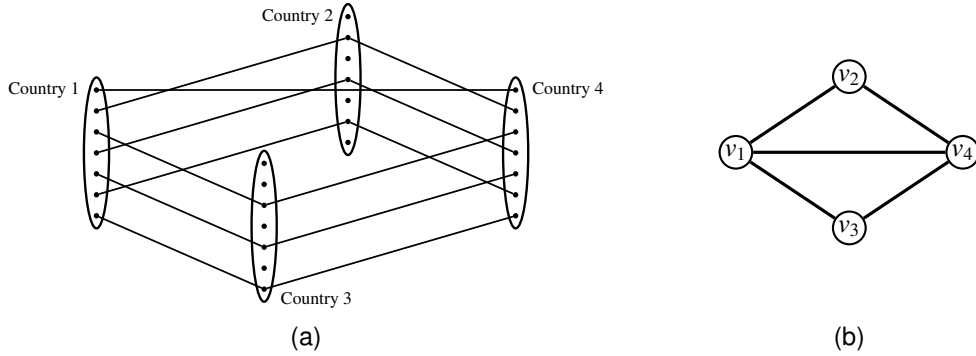


Figure 2: The matrix-weighted graph describing a four-country network: (a) each day of week corresponds to a small black node in each country and two nodes from two countries are connected by a line if there is a flight between two countries in the corresponding day; (b) Abstraction of the four-country network by a matrix-weighted graph.

countries A, B, C, D be denoted by vertices v_1, v_2, v_3, v_4 . There are flights from A to B and B to D every Monday, Wednesday, and Friday, flights from A to C and from C to D every Tuesday, Thursday, and Saturday, and a direct flight from A to D every Sunday. We say A and D connected as in all days of week, they can be reached within a day, either by direct or by transit flights. The situation is captured in the matrix-weighted graph 2.

Correspondingly, we may describe connectedness between two countries i, j by a 7×7 matrix $\mathbf{A}_{ij} = \text{diag}(a_{kk}^{ij})$, with $a_{kk}^{ij} \in \{0, 1\}$ and $a_{kk}^{ij} = 1$ means there is a direct flight from i to j in the k -th day of week. In the graph 2, $\mathbf{A}_{14} = \text{diag}(1, \mathbf{0}_6)$ (with the convention that Sunday is the first day of week), $\mathbf{A}_{12} = \mathbf{A}_{24} = \text{diag}(0, 1, 0, 1, 0, 1)$, and $\mathbf{A}_{13} = \mathbf{A}_{34} = \text{diag}(0, 0, 1, 0, 1, 0)$. Theorem 1 (i) implies that v_1 and v_4 are connected.

¹ An equivalent expression of this fact is $(\bigwedge_{r=1}^{|\mathcal{P}|} \mathbf{A}_{i_r i_{r+1}})(\mathbf{x}_j - \mathbf{x}_i) = \mathbf{0}_d$.

Remark 2. The condition (9) is only sufficient because q equations related to (10) comprise a subset of equations in the matrix equation $\mathbf{L}\mathbf{x} = \mathbf{0}_{dn}$. Since a subset of equations in $\mathbf{L}\mathbf{x} = \mathbf{0}_{dn}$ implies $\mathbf{x}_i = \mathbf{x}_j$, we do not need to consider the remaining equations to conclude $\mathbf{x}_i = \mathbf{x}_j$. Theorem 1 can be considered as an extension of (Trinh et al., 2018)[Theorem 3], where the graph G is firstly condensed into clusters associated with positive trees before iteratively assessing condition (ii) in (Trinh et al., 2018)[Theorem 3], which is actually (9). It also shows that the condition (ii) in (Trinh et al., 2018)[Theorem 3] is only a sufficient condition. A counter example showing that the possibility that $\mathbf{x}_i \neq \mathbf{x}_j$ may happen is given in Example 4 in Section 4.

It is remarked that the proof of Theorem 1 (i) involves considering all paths between v_i and v_j , which hints a connection with the concept of k -connectedness in classical graph theory Diestel (2025). Menger's theorem states that a topological graph are k -connected if there exist k independent paths between any two vertices. In matrix-weighted graphs, connectivity is jointly determined by the graph's topology (paths connecting vertices) and the values of the matrix weights. While there are many paths joining two vertices, if the matrix weights are not well chosen, they are still not belong to a same cluster. It is not hard to show that if there exists a subset of paths between v_i and v_j which do not share any common vertex rather than the starting and ending vertices, the condition (9) is also necessary and sufficient condition for v_i and v_j to belong to a same cluster according to Definition 2.

Algorithm 1 determines connectedness of an undirected matrix-weighted graph G . The lines 1–14 of Algorithm 1 iterately perform brute-force search for all paths in $\mathcal{S}_{ij}$ and determine whether v_i, v_j are connected based on equation (10), for all $1 \leq i < j \leq n$. The matrix $\mathbf{M} = [m_{ij}]$ contains 0, 1 elements, with $m_{ij} = 1$ if (10) is satisfied for a pair i, j . The lines 15–34 in the algorithm sort vertices into clusters based on the value of $\mathbf{M}$.

Remark 3 (Computational complexity analysis). The computation complexity of Algorithm 1 depend on the numbers of vertices n , the numbers of paths, and the dimension d of the matrix weights. A worst-case complexity analysis can be given as follows:

- i. Computing all eigenvalues and corresponding eigenvectors of a symmetric positive semidefinite matrix in $\mathbb{R}^{d \times d}$: $O(d^3)$. Thus, the complexity of computing the eigenvectors and eigenvalues of symmetric positive semidefinite matrices in the graph is of $O((n-2)^2 d^3)$ (Rosen, 2019).
- ii. For each path $\mathcal{P}$ between v_i and v_j , determine the linear dependency and finding a basis from a set of vectors in $\mathbb{R}^d$ (by Gauss elimination algorithm) to determine $\ker(\mathcal{P})$ is $O(d^3)$. In the worst case (G is the complete graph of n vertices), there are $|\mathcal{S}_{ij}| = 1 + (n-2) + (n-2)(n-1) + \dots + (n-2)! = (n-2)! \sum_{j=1}^{n-1} \frac{1}{j!} \leq (n-2)!e$ paths.² An additional linear dependency test is conducted to determine $\ker(\mathcal{S}_{ij}) = \bigcap_{l=1}^{n-1} \ker(\mathcal{S}_{ij}^l)$. Thus, the computational complexity test for linear dependency tests is of $O((n-2)!d^3)$.

Since the condition (9) is tested $\frac{1}{2}(n-1)n$ times (for each distinct pairs of vertices v_i, v_j), the total computational complexity is upper bounded by

$$\left(O(n^2 d^3) + O((n-2)!d^3) \right) \frac{1}{2}(n-1)n = O(n!d^3).$$

² $e \approx 2.71828$ denotes the Euler's number

Algorithm 1 Brute-force search algorithm

```
1:  $\mathbf{M} = [m_{ij}] \leftarrow \mathbf{I}_n$ ;
2:  $i \leftarrow 1$ ;
3: repeat
4:    $j \leftarrow i + 1$ ;
5:   repeat
6:     Find the set  $\mathcal{S}_{ij}$  of all paths  $\mathcal{P}_{ij}^t$  between  $v_i, v_j$  with  $t = 1, \dots, n - 1$ ;
7:     Compute  $\ker(\mathcal{S}_{ij})$ ;
8:     if  $\dim(\ker(\mathcal{S}_{ij})) == 0$  then
9:        $m_{ij} \leftarrow 1$ ;
10:    end if
11:     $j \leftarrow j + 1$ ;
12:  until  $j == n$ 
13:   $i \leftarrow i + 1$ ;
14: until  $i == n$ 
15: if  $\mathbf{M} == \mathbf{1}_n \mathbf{1}_n^\top$  then
16:    $G$  is connected;
17: else
18:    $G$  is clustering;
19:    $C_G(1) \leftarrow \{C_j = \{v_j\}, j = 1, \dots, n\}$ ;
20:    $i \leftarrow 1$ ;
21:   repeat
22:      $j \leftarrow i + 1$ ;
23:     repeat
24:       if  $m_{ij} == 1$  then
25:         if  $C_i \in C_G(i)$  and  $C_j \in C_G(i)$  then
26:            $C_i \leftarrow C_i \cup C_j$ ;
27:            $C_G(i + 1) \leftarrow C_G(i) \setminus C_j$ ;
28:         end if
29:       end if
30:        $j \leftarrow j + 1$ ;
31:     until  $j == n$ 
32:      $i \leftarrow i + 1$ ;
33:   until  $i == n$ 
34: end if
```

Remark 4. The test of $\dim(\mathcal{S}_{ij})$ in lines 7–10 of Algorithm 1 can be incorporated after each set of paths $\mathcal{P}_{ij}^t$ joining v_i and v_j was considered to accelerate the algorithm. That is, initially, $\ker(\mathcal{S}_{ij}) \leftarrow \mathbb{R}^d$. For $t = 1$ to $n - 1$, we compute $\ker(\mathcal{P}_{ij}^t)$, update $\ker(\mathcal{S}_{ij}) \leftarrow \ker(\mathcal{S}_{ij}) \cap \ker(\mathcal{P}_{ij}^t)$, and examine whether $\dim(\ker(\mathcal{P}_{ij}^t))$ is zero or not. If the result is positive, m_{ij} is set to 1 and the computation of $\mathcal{S}_{ij}$ can be continue with another pair of v_i, v_j .

3.2. Warshall algorithm

In this subsection, we propose Warshall algorithm for matrix-weighted graphs. In matrix-weighted graph, for any two vertices v_i and v_j , three possible scenarios may happen: (i) v_i and

v_j belong to a same cluster; (ii) v_i and v_j does not belong to a same cluster but are connected by some paths in G ; and (iii) there is no path joining v_i and v_j . We exploit the sufficient condition for connectedness in Theorem 1 to determine the connectedness of a given matrix-weighted graph.

Since connectedness between two clusters of a matrix-weighted graph is an aggregated efforts between different paths in the graph, in addition to the connected state (case (i)) and the disconnected state (case (iii)), we define an *undecided state* which is only determined after a certain algebraic condition was satisfied. The undecided state sets two vertices v_i, v_j do not belong to a same clusters only if all possible paths in $\mathcal{S}_{ij}$ have been examined. If for some $t < n - 1$, $\dim(\ker(\mathcal{S}_{ij})) = 0$, the undecided state assigns v_i and v_j to a same cluster, allowing the connectivity test between these vertices to terminate earlier.

Let $G = (V, E, W)$ be an undirected matrix-weighted graph of n vertices with the corresponding matrix-weighted adjacency matrix $\mathbf{A} = [\mathbf{A}_{ij}]$. For computational efficiency, we define the decision operator $\mathcal{D}$ for a positive semidefinite matrix $\mathbf{A}_{ij} \in \mathbb{R}^{d \times d}$

$$\mathcal{D}(\mathbf{A}_{ij}) = \begin{cases} \mathbf{I}_d, & \text{if } \dim(\mathbf{A}_{ij}) = d, \\ \mathbf{\Theta}_d, & \text{if } \mathbf{A}_{ij} = \mathbf{\Theta}_d, \\ \mathbf{A}_{ij}, & \text{if } \dim(\mathbf{A}_{ij}) < d, \end{cases} \quad (16)$$

and for a block matrix $\mathbf{A} = [\mathbf{A}_{ij}]$,

$$\mathcal{D}(\mathbf{A}) = [\mathcal{D}(\mathbf{A}_{ij})]. \quad (17)$$

Several properties of the decision operator are given in the following lemma, whose proof can be found in [Appendix A.2](#).

Lemma 4. *Let $\mathbf{A}_1, \mathbf{A}_2 \in \mathbb{R}^{d \times d}$ be symmetric positive semidefinite matrices, there holds*

- i. $\ker(\mathcal{D}(\mathbf{A}_1)) = \ker(\mathbf{A}_1)$,
- ii. $\ker(\mathcal{D}(\mathbf{A}_1 \vee \mathbf{A}_2)) = \ker(\mathcal{D}(\mathbf{A}_1) \vee \mathcal{D}(\mathbf{A}_2))$,
- iii. $\ker(\mathcal{D}(\mathbf{A}_1 \wedge \mathbf{A}_2)) = \ker(\mathcal{D}(\mathbf{A}_1) \wedge \mathcal{D}(\mathbf{A}_2))$,
- iv. *If $\mathbf{A}_2$ is positive definite, $\ker(\mathcal{D}(\mathbf{A}_1 \wedge \mathbf{A}_2)) = \ker(\mathbf{A}_1)$ and $\mathcal{D}(\mathbf{A}_1 \vee \mathbf{A}_2) = \mathbf{I}_d$.*

Let $\mathbf{A} = [\mathbf{A}_{ij}]$ and $\mathbf{B} = [\mathbf{B}_{ij}] \in \mathbb{R}^{dn \times dn}$ be block matrices, each submatrix is of size $d \times d$. The “ $\wedge$ ” operator for two block matrices $\mathbf{A}$ and $\mathbf{B}$ is defined as $\mathbf{C} = [\mathbf{C}_{ij}] = \mathbf{A} \wedge \mathbf{B}$, where

$$\mathbf{C}_{ij} = \bigvee_{k=1}^n (\mathbf{A}_{ik} \wedge \mathbf{B}_{kj}), \quad \forall i, j = 1, \dots, n, \quad (18)$$

and the “ $\vee$ ” operator for two block matrices $\mathbf{A}$ and $\mathbf{B}$, denoted as $\mathbf{D} = [\mathbf{D}_{ij}] = \mathbf{A} \vee \mathbf{B}$, satisfies

$$\mathbf{D}_{ij} = \mathbf{A}_{ij} \vee \mathbf{B}_{ij}, \quad \forall i, j = 1, \dots, n. \quad (19)$$

The “ $\wedge$ ” and “ $\vee$ ” operators for block matrices are defined as generalizations of the usual matrix multiplication and matrix addition.

We define the power of a block matrix $\mathbf{A} = [\mathbf{A}_{ij}]$ combined with the decision operator recursively as follows

$$\mathbf{A}^0 = \mathbf{I}_{dn}, \quad (20a)$$

$$\mathbf{A}^1 = \mathcal{D}(\mathbf{A}), \quad (20b)$$

$$\mathbf{A}^k = \mathcal{D}(\mathbf{A}^{k-1} \wedge \mathcal{D}(\mathbf{A})), \quad k \geq 2. \quad (20c)$$

Note that we may also write $\mathbf{A}^1 = \mathcal{D}(\mathbf{A} \wedge \mathbf{I}_{dn})$ without changing the subsequent definition.

For an n -vertex matrix-weighted graph G , let

$$\mathbf{M}(G, 0) = \mathbf{A}^0 = \mathbf{I}_{dn}, \quad (21a)$$

$$\mathbf{M}(G, k) = \mathcal{D}(\mathbf{M}(G, k-1) \vee \mathbf{A}^k), \quad k \geq 1. \quad (21b)$$

The combinations of the operators “ $\vee$ ”, “ $\wedge$ ” and “ $\mathcal{D}$ ” in equation (21) is characterized in the following lemma, whose proof can be found in [Appendix A.3](#).

Lemma 5 (Monotonicity in connectivity). *Consider an undirected matrix-weighted graph G which has the matrix-weighted adjacency matrix $\mathbf{A} \in \mathbb{R}^{dn \times dn}$ with block matrices $\mathbf{A}_{ij} \in \mathbb{R}^{d \times d}$. Defining the matrices $\mathbf{M}(G, k), k = 1, \dots, n-1$, as in (21), then,*

- i. $\ker([\mathbf{M}(G, k)]_{ij}) = \ker\left(\left[\mathcal{D}\left(\bigvee_{i=0}^k \mathbf{A}^i\right)\right]_{ij}\right)$, and
- ii. $\ker([\mathbf{M}(G, k)]_{ij}) \subseteq \ker([\mathbf{M}(G, k-1)]_{ij})$, where we use $[\mathbf{M}(G, k)]_{ij}$ to denote the ij -th block matrix in $\mathbf{M}$.

The Warshall algorithm (Algorithm 2)³ is proposed to determine connectedness and clustering of an undirected matrix-weighted graph G . The main result of this subsection is stated in the following theorem.

Theorem 2. *Consider an undirected matrix-weighted graph $G = (V, E, W)$ of n vertices with matrix weights $\mathbf{A}_{ij} = \mathbf{A}_{ij}^\top \in \mathbb{R}^{d \times d}$. The following claims on Algorithm 2 hold.*

- i. *Two vertices $v_i, v_j \in V$ belong to the same cluster if there exists $k \leq n-1$ such that the ij -th block matrix of $\mathbf{M}(G, k)$ satisfies $[\mathbf{M}(G, k)]_{ij} = \mathbf{I}_d$.*
- ii. *The graph G is connected if $\mathbf{M}(G, n-1) = \mathbf{1}_n \mathbf{1}_n^\top \otimes \mathbf{I}_d$. If there is no path joining v_i to v_j , then $[\mathbf{M}(G, n-1)]_{ij} = [\mathbf{M}(G, n-1)]_{ji} = \mathbf{0}_d$.*

Proof. i. As has been proved in subsection 3.1, we only consider sets of paths of length up to $n-1$ since these sets contain all simple paths joining v_i and v_j .

Since the graph is undirected, $\mathcal{S}_{ij}^t = \mathcal{S}_{ji}^t, \forall v_i \neq v_j \in V$ and $\forall 1 \leq t \leq n-1$. We will show that $\ker([\mathbf{A}^t]_{ij}) = \ker(\mathcal{S}_{ji}^t), \forall i \neq j$ and $\forall t = 1, \dots, n-1$, by mathematical induction.

- The claim holds for $t = 0$, as $[\mathbf{A}^0]_{ii} = \mathbf{I}_d$ (vertex v_i belongs to the same cluster with itself) and $[\mathbf{A}^0]_{ij} = \mathbf{0}_d$ for all $i \neq j$.
- The claim holds for $t = 1$, as $[\mathbf{A}^1]_{ij} = \mathbf{A}_{ij}$ is the matrix weight corresponding to the edge (v_j, v_i) , $\ker(\mathcal{S}_{ji}^1) = \ker(\mathbf{A}_{ij}), \forall i \neq j \in V$.
- Suppose that the claim holds until $t \geq 1$, i.e., $\ker(\mathcal{S}_{ji}^s) = \ker([\mathbf{A}^s]_{ij}), \forall i \neq j \in V$ and $s = 0, 1, \dots, t$. We prove that the claim is also true for $t+1$. By expanding the formula

$$[\mathbf{A}^{t+1}]_{ij} = \bigvee_{k=1}^n ([\mathbf{A}^t]_{ik} \wedge \mathbf{A}_{kj})$$

³We use the name Warshall algorithm as it uses an analogous boolean matrix multiplication as in the Warshall algorithm for topological graph [Warshall \(1962\)](#).

Algorithm 2 Warshall algorithm

```

1:  $k \leftarrow 0$ ;
2:  $\mathbf{M} \leftarrow \mathbf{I}_{dn}$ ;
3: repeat
4:    $\mathbf{M} \leftarrow \mathcal{D}(\mathbf{M} \vee (\mathbf{M} \wedge \mathbf{A}))$ ;
5:    $k \leftarrow k + 1$ ;
6: until  $k == n$  or  $\mathbf{M} == \mathbf{1}_n \mathbf{1}_n^\top \otimes \mathbf{I}_d$ 
7: if  $\mathbf{M} == \mathbf{1}_n \mathbf{1}_n^\top \otimes \mathbf{I}_d$  then
8:    $G$  is connected;
9: else
10:   $G$  is clustering;
11:   $C_G(1) \leftarrow \{C_j = \{v_j\}, j = 1, \dots, n\}$ ;
12:   $i \leftarrow 1$ ;
13:  repeat
14:     $j \leftarrow i + 1$ ;
15:    repeat
16:      if  $\mathbf{M}_{ij} == \mathbf{I}_d$  then
17:        if  $C_i \in C_G(i)$  and  $C_j \in C_G(i)$  then
18:           $C_i \leftarrow C_i \cup C_j$ ;
19:           $C_G(i + 1) \leftarrow C_G(i) \setminus C_j$ ;
20:        end if
21:      end if
22:       $j \leftarrow j + 1$ ;
23:    until  $j == n$ 
24:     $i \leftarrow i + 1$ ;
25:  until  $i == n$ 
26: end if

```

and noting that a path of length $t + 1$ is formed by inserting a path $\mathcal{P}_{ik}^t$ of length t from some $v_k \in \mathcal{N}_j$ to v_i and the edge $(v_j, v_k) \in E$. Since $\ker([\mathbf{A}^t]_{ik}) = \bigcap_{\mathcal{P}_{ik}^t \in \mathcal{S}_{ik}^t} \ker(\mathcal{P}_{ik}^t)$ captures the intersection of the kernel of all paths of length t from v_k to v_i , all paths $\mathcal{P}_{ik,j}^{t+1} = \mathcal{P}_{ik}^t + (v_j, v_k)$ has $\ker(\mathcal{P}_{j,ki}^{t+1}) = \ker([\mathbf{A}^t]_{ik} \wedge \mathbf{A}_{kj})$. The proof follows by taking the intersection of all kernels $\ker(\mathcal{P}_{j,ki}^{t+1})$ over $k = 1, \dots, n$.

- By mathematical induction, the claim holds for all $t = 0, 1, \dots, n - 1$.

Finally, it follows from Lemma 5 that the kernel of the matrix block $[\mathbf{M}(G, n - 1)]_{ij}$ is the intersection of all $\ker(\mathcal{P}_{ji}^t)$, $t = 0, 1, \dots, n - 1$. Applying the condition (9), it follows that v_i, v_j belong to a same cluster if $[\mathbf{M}(G, n - 1)]_{ij} = \mathbf{I}_d$.

ii. Based on (i), it is clear that G is connected if all v_i, v_j belong to a same cluster, or equivalently, $\mathbf{M}(G, n - 1) = \mathbf{1}_n \mathbf{1}_n^\top \otimes \mathbf{I}_d$. Trivially, if v_i and v_j is topologically disconnected, there exists no path connecting them. Thus, $[\mathbf{M}(G, n - 1)]_{ij} = \mathbf{0}_d$. $\square$

Remark 5 (Computational complexity analysis). *For the algorithm 2, a worst-case computational complexity can be derived as follows*

- i. Determine the graph's topology using a Depth-First Search algorithm is $O(|V| + |E|)$ (Rosen, 2019). In the worst-case, G is a complete graph and the computational complexity is upper bounded by $O(n^2)$.
- ii. Due to symmetry of matrices $\mathbf{A}^r$, $r = 0, \dots, t-1$ and the fact that the block matrices in the diagonal are always $[\mathbf{A}^r]_{ii} = \mathbf{I}_d$, the block matrix “ $\wedge$ ” operator $\mathbf{A}^t = \mathbf{A}^{t-1} \wedge \mathbf{A}$ requires computing $\frac{1}{2}n(n-1)$ submatrices $[\mathbf{A}^t]_{ij}$, with $1 < i < j \leq n$ according to (18). To compute each submatrix in (18), we need n parallel matrix addition $[\mathbf{A}^{t-1}]_{ik} \wedge \mathbf{A}_{kj}$ (1) and one series matrix addition.⁴ Thus, computing $[\mathbf{A}^t]_{ij}$ is of $O(nd^3)$, and $\mathbf{A}^t$ is of $O(n^3d^3)$.
- iii. To determine $\mathbf{M}(G, t)$, the symmetry also reduces the computation burden. We need to compute the “ $\vee$ ” operator $n-1$ times to determine $[\mathbf{M}(G, t-1) \vee \mathbf{A}^t]_{ij}$, $1 < i < j \leq n$. The computations from the previous steps can be reused here, with the cost of storing eigenvectors and eigenvalues decomposition of $[\mathbf{M}(G, t-1)]_{ij}$ and $[\mathbf{A}^t]_{ij}$. This gives a computational complexity of $O(d^3)$ accounting for determining a set of linearly independent eigenvectors from those in $\mathbf{M}(G, t-1)$ and $\mathbf{A}^t$. If the calculations from previous steps are not saved in the memory, the computation cost is tripled, as we need to find the set of eigenvectors and eigenvalues of each matrix $[\mathbf{M}(G, t-1)]_{ij}$ and $[\mathbf{A}^t]_{ij}$ again before determining the intersection of them for $[\mathbf{M}(G, t-1) \vee \mathbf{A}^t]_{ij}$. After this step, the decision operator $\mathcal{D}$ is immediately applied $\mathcal{D}([\mathbf{M}(G, t-1) \vee \mathbf{A}^t]_{ij})$, and it is reasonable to assume that the set of linearly independent eigenvectors of $[\mathbf{M}(G, t-1) \vee \mathbf{A}^t]_{ij}$ is still available. Thus, $\mathcal{D}([\mathbf{M}(G, t-1) \vee \mathbf{A}^t]_{ij})$ adds $O(d)$ to the computational complexity. To sum up, determining $\mathbf{M}(G, t)$ is of $(O(d^3) + O(d))\frac{1}{2}n(n-1) = O(d^3n^2)$.
- iv. In the worst-case, we need to compute until $t = n-1$ to obtain $\mathbf{M}(G, n-1)$. The total computational cost is thus upper bounded by $O(n^2) + O(d^3n^2)(n-2) = O(d^3n^3)$.

In comparison with the brute-force search algorithm, the worst-case computational complexity is substantially reduced. In brute-force search, the kernels all possible paths in $\mathcal{S}_{ij}$ have to be sequentially computed before aggregated to determine $\ker(\mathcal{S}_{ij})$. In contrast, the Warshall algorithm uses “ $\vee$ ” operator in determining $[\mathbf{A}^t]_{ij}$ and $[\mathbf{M}(G, t)]_{ij}$ at each step to aggregate the kernels of $\mathcal{S}_{ij}^t$ (saved in $[\mathbf{A}^t]_{ij}$) into $\cap_{r=1}^{t-1} \mathcal{S}_{ij}^{r-1}$ (saved in $[\mathbf{M}(G, t-1)]_{ij}$), and simultaneously for all $1 < i < j \leq n$. Since the recurrence relations (20)–(21) reuse computing results from the previous step, the worst-case computational complexity is significantly reduced.

4. Numerical examples

In this section, we consider several examples to illustrate the Algorithms 1 and 2. In each example, we focus on the output of Algorithm 2 and use the condition (9) in Algorithm 1 to verify the result.

⁴Since series matrix addition requires finding a set of independent eigenvectors, $n-1$ series matrix addition can be perform only one time, after n parallel matrix addition have been completed.

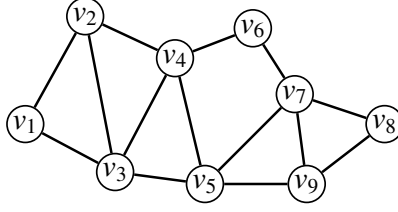


Figure 3: The matrix-weighted graph considered in Example 1.

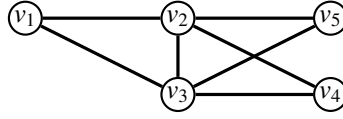


Figure 4: The matrix-weighted graph considered in Example 2

4.1. Example 1: Connected matrix-weighted graph

Consider a nine-vertex matrix-weighted graph $G = (V, E, W)$ as depicted in Fig. 3. Then, $n = 9, m = 14$, and $d = 3$. The matrix weights are given as follows

$$\begin{aligned} \mathbf{A}_{12} = \mathbf{A}_{67} &= \begin{bmatrix} 1 & \frac{\sqrt{3}}{2} & 0 \\ \frac{\sqrt{3}}{2} & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \mathbf{A}_{23} = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{A}_{13} = \mathbf{A}_{24} = \mathbf{A}_{34} = \mathbf{A}_{59} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \\ \mathbf{A}_{35} = \mathbf{A}_{57} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \mathbf{A}_{45} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}, \mathbf{A}_{78} = \mathbf{A}_{79} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{A}_{46} = \mathbf{A}_{89} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \end{aligned}$$

It can be checked that the matrix-weighted Laplacian of G satisfies $\text{rank}(\mathbf{L}) = 24 = dn - d$, suggesting that the matrix-weighted graph is connected.

Using Algorithm 1 to test connectedness, even in this small graph has become inefficient.

Using Algorithm 2, we found that the algorithm terminates at $k = 6 < n - 1 = 8$ and $\mathbf{M}(G, k) = \mathbf{1}_9 \mathbf{1}_9^T \otimes \mathbf{I}_2$. The numerical result verifies that the matrix-weighted graph G is connected.

4.2. Example 2: A graph which is topologically connected but not connected

In this example, a matrix-weighted graph of five vertices depicted in Fig. 4 is considered ($n = 5, d = 2$). The matrix weights corresponding to the edges are given as follows:

$$\mathbf{A}_{12} = \mathbf{A}_{24} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \mathbf{A}_{13} = \mathbf{A}_{34} = \mathbf{A}_{35} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{A}_{23} = \mathbf{A}_{25} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.$$

Note that G has no positive definite edge. The rank of the matrix-weighted Laplacian is smaller than $8 = dn - d$, which implies that the matrix-weighted graph is not connected.

After using Algorithm 2, we obtain the matrix $\mathbf{M}(G, 4)$ as follows

$$\mathbf{M}(G, 4) = \begin{bmatrix} 1 & 0 & 0.95 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1.125 & 0 & 1 & 0 & 0.7179 \\ 0.95 & 0 & 1 & 0 & 0.7 & -0.7 & 0.95 & 0 & 0.7 & -0.7 \\ 0 & 0 & 0 & 1 & -0.7 & 0.7 & 0 & 0 & -0.7 & 0.7 \\ 0 & 0 & 0.7 & -0.7 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1.125 & -0.7 & 0.7 & 0 & 1 & 0 & 1.125 & 0 & 1 \\ 1 & 0 & 0.95 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1.125 & 0 & 1 & 0 & 0.7179 \\ 0 & 0 & 0.7 & -0.7 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0.7179 & -0.7 & 0.7 & 0 & 1 & 0 & 0.7179 & 0 & 1 \end{bmatrix}.$$

It follows from the pattern of identity submatrices matrix $[\mathbf{M}(G, 4)]_{ij} = \mathbf{I}_2$ that the graph G has three clusters $C_1 = \{v_1, v_4\}$, $C_2 = \{v_2\}$ and $C_3 = \{v_3, v_5\}$. It is interesting that two vertices v_1 and v_4 does not have any positive path joining them, and are not adjacent to each other, are actually lying on a same cluster.

We can use condition (9) to verify that v_1 and v_4 belong to a same cluster. The paths in $\mathcal{S}_{14}$ and their kernels are listed below.

- $\mathcal{P}_1 = v_1 v_2 v_4$, $\ker(\mathcal{P}_1) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$,
- $\mathcal{P}_2 = v_1 v_3 v_4$, $\ker(\mathcal{P}_2) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$,
- $\mathcal{P}_3 = v_1 v_2 v_3 v_4$, $\ker(\mathcal{P}_3) = \mathbb{R}^2$,
- $\mathcal{P}_4 = v_1 v_3 v_2 v_4$, $\ker(\mathcal{P}_4) = \mathbb{R}^2$,
- $\mathcal{P}_5 = v_1 v_2 v_5 v_3 v_4$, $\ker(\mathcal{P}_5) = \mathbb{R}^2$,
- $\mathcal{P}_6 = v_1 v_3 v_5 v_2 v_4$, $\ker(\mathcal{P}_6) = \mathbb{R}^2$,

Therefore, $\ker(\mathcal{S}_{14}) = \bigcap_{k=1}^6 \mathcal{P}_k = \{\mathbf{0}_2\}$ and $\dim(\ker(\mathcal{S}_{14})) = 0$.

It is also observe that all submatrices $[\mathbf{M}(G, 4)]_{ij}$ are non-zero in this example, which suggests that the topological graph of G is connected. However, this is actually not generally true that a connected matrix-weighted graph of n vertices has all nonzero matrix block $[\mathbf{M}(G, n-1)]_{ij}$. The next example illustrates this fact.

4.3. Example 3: Zero submatrix in $\mathbf{M}(G, n-1)$ does not imply topological disconnectedness

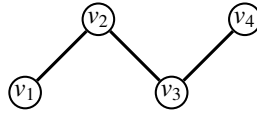


Figure 5: The matrix-weighted graph considered in Example 3

We consider a line graph of four vertices $G(V, E, W)$ in Fig. 5 with

$$\mathbf{A}_{12} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{A}_{23} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \mathbf{A}_{34} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}.$$

Clearly, there is only one path from v_1 to v_4 , which is $\mathcal{P} = v_1 v_2 v_3 v_4$. It is not hard to verify that $\ker(\mathbf{A}_{12}) = \text{im}\left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}\right)$, $\ker(\mathbf{A}_{23}) = \text{im}\left(\begin{bmatrix} 0 \\ 1 \end{bmatrix}\right)$, and $\ker(\mathbf{A}_{34}) = \{\mathbf{0}_2\}$ and thus $\ker(\mathcal{P}) = \ker(\mathbf{A}_{12}) \cup \ker(\mathbf{A}_{23}) \cup \ker(\mathbf{A}_{34}) = \mathbb{R}^2$. Thus, the matrix block $[\mathbf{M}(G, 3)]_{14} = \Theta_d$ while the graph is topologically connected. Indeed, the computation gives

$$\mathbf{M}(G, 3) = \left[\begin{array}{cc|cc|cc|cc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0.75 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0.9208 & 0 & 0.3 & 0 \\ 0 & 0.75 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0.9208 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ \hline 0 & 0 & 0.3 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{array} \right],$$

which verifies the computation based on path kernels. This example demonstrates that $\mathbf{M}(G, n-1)$ also cannot be used to check topological connectedness.

4.4. Example 4: The algorithms are heuristic

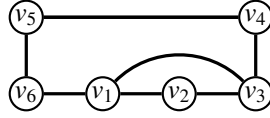


Figure 6: The graph in Example 4

Consider the matrix-weighted graph as depicted in Fig. 6, with

$$\begin{aligned} \mathbf{A}_{12} &= \mathbf{A}_{23} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad \mathbf{A}_{13} = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}, \quad \mathbf{A}_{45} = \mathbf{A}_{56} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \\ \mathbf{A}_{16} &= \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{A}_{34} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}. \end{aligned}$$

Three paths between v_1 and v_6 and their corresponding kernels are

- $\mathcal{P}_1 = v_1 v_6$, $\ker(\mathcal{P}_1) = \text{im}\left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}\right)$,
- $\mathcal{P}_2 = v_1 v_2 v_3 v_4 v_5 v_6$, $\ker(\mathcal{P}_1) = \mathbb{R}^2$,
- $\mathcal{P}_3 = v_1 v_3 v_4 v_5 v_6$, $\ker(\mathcal{P}_1) = \mathbb{R}^2$.

Then, $\ker(\mathcal{S}_{16}) = \bigcap_{k=1}^3 \ker(\mathcal{P}_k) = \text{im}\left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}\right)$. The connectivity test (9) is not satisfied.

The connectivity test (9) satisfies for v_1, v_3 as

$$\ker(v_1 v_2 v_4) \cap \ker(v_1 v_3) = \text{im}\left(\begin{bmatrix} 1 \\ -1 \end{bmatrix}\right) \cap \text{im}\left(\begin{bmatrix} -2 \\ 1 \end{bmatrix}\right) = \{\mathbf{0}_2\}.$$

Identify v_1 and v_3 as a single vertex $v_{1,3}$, there are two paths from $v_{1,3}$ to v_6 : $Q_1 = v_{1,3}v_6$ and $Q_2 = v_{1,3}v_4v_5v_6$, with $\ker(Q_1) = \text{im}\left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}\right)$, $\ker(Q_2) = \text{im}\left(\begin{bmatrix} 0 \\ 1 \end{bmatrix}\right)$, $\ker(Q_1) \cap \ker(Q_2) = \{0_2\}$. Thus, v_1, v_3, v_6 are actually belong to the same cluster. The connectivity test (9) fails to provide a precise clustering of the graph in this example.

5. Conclusion

Two heuristic algorithms for assessing connectedness and clustering in undirected matrix-weighted graphs have been considered. The brute-force search algorithm requires listing all edges in an undirected graphs, making the algorithm become intractable for large graphs with dense edges. Since the algebraic rank test can be performed with the Laplacian matrix $\mathbf{L}$ directly, the Warshall algorithm - which requires the matrix-weighted adjacency matrix - is inefficient for solely assessing connectedness. The main advantage of the Warshall algorithm is its ability to classify vertices into distinct clusters while maintaining lower computational complexity than the brute-force search. Ongoing works are focusing on exact algorithms for connectedness and clustering in matrix-weighted graphs, as well as designing graphs with preset clusters. Furthermore, the path decomposition approach introduced in Section 3 offers a promising approach for development of k -connectivity in matrix-weighted graphs.

References

- Ahn, H.S., Tran, Q.V., Trinh, M.H., Ye, M., Liu, J., Moore, K.L., 2020. Opinion dynamics with cross-coupling topics: Modeling and analysis. *IEEE Transactions on Computational Social Systems* 7, 632–647. doi:[10.1109/TCSS.2020.2974899](https://doi.org/10.1109/TCSS.2020.2974899).
- Anderson, W.N., Duffin, R.J., 1969. Series and parallel addition of matrices. *Journal of Mathematical Analysis and Applications* 26, 576–594. doi:[10.1016/0022-247X\(69\)90200-5](https://doi.org/10.1016/0022-247X(69)90200-5).
- Atik, F., Kannan, M.R., 2019. Resistance matrices of graphs with matrix weights. *Linear Algebra and Its Applications* 571, 41–57. doi:[10.1016/j.laa.2019.02.011](https://doi.org/10.1016/j.laa.2019.02.011).
- Barooah, P., Hespanha, J.P., 2005. Distributed estimation from relative measurements in sensor networks, in: *Proc. of the 3rd International Conference on Intelligent Sensing and Information Processing*, Bangalore, India, IEEE, USA. pp. 226–231. doi:[10.1109/ICISIP.2005.1619440](https://doi.org/10.1109/ICISIP.2005.1619440).
- Barooah, P., Hespanha, J.P., 2008. Estimation from relative measurements: Electrical analogy and large graphs. *IEEE Transactions on Signal Processing* 56, 2181–2193. doi:[10.1109/TSP.2007.912270](https://doi.org/10.1109/TSP.2007.912270).
- Diestel, R., 2025. *Graph theory*. volume 173. Springer Nature. URL: <https://diestel-graph-theory.com/>.
- Hansen, J., 2021. Expansion in matrix-weighted graphs. *Linear Algebra and its Applications* 630, 252–273. URL: [10.1016/j.laa.2021.08.009](https://doi.org/10.1016/j.laa.2021.08.009).
- Kivelä, M., Arenas, A., Barthélemy, M., Gleeson, J.P., Moreno, Y., Porter, M.A., 2014. Multi-layer networks. *Journal of Complex Networks* 2, 203–271. URL: <https://doi.org/10.1093/comnet/cnu016>, doi:[10.1093/comnet/cnu016](https://doi.org/10.1093/comnet/cnu016).

- Mahato, I., Kannan, M.R., 2023. Squared distance matrices of trees with matrix weights. *AKCE International Journal of Graphs and Combinatorics* 20, 168–176. URL: [10.1080/09728600.2023.2236172](https://doi.org/10.1080/09728600.2023.2236172).
- Miao, S., Su, H., 2021. Second-order consensus of multiagent systems with matrix-weighted network. *Neurocomputing* 433, 1–9. doi:[10.1016/j.neucom.2020.12.056](https://doi.org/10.1016/j.neucom.2020.12.056).
- Pan, L., Shao, H., Lu, Y., Mesbahi, M., Li, D., Xi, Y., 2025. Privacy-preserving average consensus via matrix-weighted inter-agent coupling. *Automatica* 174, 112094. doi:[10.1016/j.automatica.2024.112094](https://doi.org/10.1016/j.automatica.2024.112094).
- Pan, L., Shao, H., Mesbahi, M., Xi, Y., Li, D., 2018. Bipartite consensus on matrix-valued weighted networks. *IEEE Transactions on Circuits and Systems II: Express Briefs* 66, 1441–1445. doi:[10.1109/TCSII.2018.2884483](https://doi.org/10.1109/TCSII.2018.2884483).
- Rosen, K.H., 2019. Discrete Mathematics and Its Applications. McGraw-Hill. URL: <https://www.mheducation.com/highered/product/Discrete-Mathematics-and-Its-Applications-Rosen.html>.
- Tran, Q.V., Trinh, M.H., Ahn, H.S., 2021. Discrete-time matrix-weighted consensus. *IEEE Transactions on Control of Network Systems* 8, 1568–1578. doi:[10.1109/TCNS.2021.3068367](https://doi.org/10.1109/TCNS.2021.3068367).
- Trinh, M.H., Ahn, H.S., 2025. Matrix-weighted graphs: Theory and Applications. Lecture Notes in Control and Information Sciences, Springer - Cham, Switzerland. URL: <https://link.springer.com/book/9783032030788>.
- Trinh, M.H., Le-Phan, N.M., Ahn, H.S., 2024. The networked input-output economic problem. arXiv e-prints doi:[10.48550/arXiv.2412.13564](https://doi.org/10.48550/arXiv.2412.13564).
- Trinh, M.H., Nguyen, C.V., Lim, Y.H., Ahn, H.S., 2018. Matrix-weighted consensus and its applications. *Automatica* 89, 415–419. doi:[10.1016/j.automatica.2017.12.024](https://doi.org/10.1016/j.automatica.2017.12.024).
- Trinh, M.H., Ye, M., Ahn, H.S., Anderson, B.D.O., 2017. Matrix-weighted consensus with leader-following topologies, in: Proc. of the Asian Control Conference, Gold Coast, AU, IEEE. IEEE, USA. pp. 1795–1800. doi:[10.1109/ASCC.2017.8287446](https://doi.org/10.1109/ASCC.2017.8287446).
- Vu, H.H., Nguyen, Q.N., Pham, T.V., Nguyen, C.V., Trinh, M.H., 2025. Consensus seeking in diffusive multidimensional networks with a repeated interaction pattern and time-delays, in: Proc. of the IEEE Conference on Control Technology and Applications (CCTA), IEEE. pp. 109–114. doi:[10.1109/CCTA53793.2025.11151462](https://doi.org/10.1109/CCTA53793.2025.11151462).
- Warshall, S., 1962. A theorem on boolean matrices. *Journal of the ACM* 9, 11–12. doi:[10.1145/321105.321107](https://doi.org/10.1145/321105.321107).
- Zelazo, D., Mesbahi, M., 2010. Edge agreement: Graph-theoretic performance bounds and passivity analysis. *IEEE Transactions on Automatic Control* 56, 544–555.
- Zhao, S., Zelazo, D., 2016. Localizability and distributed protocols for bearing-based network localization in arbitrary dimensions. *Automatica* 69, 334–341. doi:[10.1016/j.automatica.2016.03.010](https://doi.org/10.1016/j.automatica.2016.03.010).

Appendix A. Proofs

Appendix A.1. Proof of Lemma 3

i. Based on the distributivity of the intersection over the union, we have: $\ker((\mathbf{A}_1 \wedge \mathbf{A}_2) \vee \mathbf{A}_3) = \ker((\mathbf{A}_1 \wedge \mathbf{A}_2)) \cap \ker(\mathbf{A}_3) = (\ker(\mathbf{A}_1) \cup \ker(\mathbf{A}_2)) \cap \ker(\mathbf{A}_3) = (\ker(\mathbf{A}_1) \cap \ker(\mathbf{A}_3)) \cup (\ker(\mathbf{A}_2) \cap \ker(\mathbf{A}_3)) = \ker(\mathbf{A}_1 \vee \mathbf{A}_3) \cap \ker(\mathbf{A}_2 \vee \mathbf{A}_3)$.

ii. Similarly, the identity follows from $\ker((\mathbf{A}_1 \vee \mathbf{A}_2) \wedge \mathbf{A}_3) = \ker((\mathbf{A}_1 \vee \mathbf{A}_2)) \cup \ker(\mathbf{A}_3) = (\ker(\mathbf{A}_1) \cap \ker(\mathbf{A}_2)) \cup \ker(\mathbf{A}_3) = (\ker(\mathbf{A}_1) \cup \ker(\mathbf{A}_3)) \cap (\ker(\mathbf{A}_2) \cup \ker(\mathbf{A}_3)) = \ker(\mathbf{A}_1 \wedge \mathbf{A}_3) \cup \ker(\mathbf{A}_2 \wedge \mathbf{A}_3)$.

Appendix A.2. Proof of Lemma 4

i. This property follows directly from the definition of the decision operator. If $\mathbf{A}_1$ is positive definite, $\ker(\mathbf{A}_1) = \ker(\mathbf{I}_d) = \mathcal{D}(\mathbf{A}_1)$. Otherwise, the decision operator does not change $\mathbf{A}_1$, thus, leaves its kernel unaltered.

ii. Consider two possibilities as follows:

- If $\exists \mathbf{A}_i, i \in \{1, 2\}$ positive definite. Then, $\mathbf{A}_1 + \mathbf{A}_2$ is positive definite and thus, the equality holds.
- $\mathbf{A}_i, i \in \{1, 2\}$ are not positive definite. Then, $\mathcal{D}(\mathbf{A}_1) \vee \mathcal{D}(\mathbf{A}_2) = \mathbf{A}_1 \vee \mathbf{A}_2$ and thus, the equality holds.

iii. Consider two possibilities as follows:

- If $\exists \mathbf{A}_i, i \in \{1, 2\}$ positive definite, without loss of generality, says $\mathbf{A}_1$. Then, $\ker(\mathcal{D}(\mathbf{A}_1) \wedge \mathcal{D}(\mathbf{A}_2)) = \ker(\mathbf{A}_2) = \ker(\mathcal{D}(\mathbf{A}_1 \wedge \mathbf{A}_2))$. The equality holds.
- $\mathbf{A}_i, i \in \{1, 2\}$ are not positive definite. Then, $\mathcal{D}(\mathbf{A}_1) \wedge \mathcal{D}(\mathbf{A}_2) = \mathbf{A}_1 \wedge \mathbf{A}_2$ and thus, the equality follows from (i).

iv. These identities are special cases of (ii) and (iii).

Appendix A.3. Proof of Lemma 5

i. Clearly, the output of the “ $\wedge$ ” (or the “ $\vee$ ”) operator of two symmetric positive semidefinite matrices is again a positive semidefinite matrix. The decision operator also outputs a positive semidefinite matrix given that the operand is a positive semidefinite matrix. Thus, these three operators can be applied in sequences and the power of a block matrix $\mathbf{A} \in \mathbb{R}^{dn \times dn}$ is well-

defined. We have

$$\begin{aligned}
\ker([\mathbf{M}(G, k)]_{ij}) &= \ker\left(\left[\mathcal{D}\left(\mathbf{M}(G, k-1) \vee \mathbf{A}^k\right)\right]_{ij}\right) \\
&= \ker\left(\left[\mathcal{D}\left(\mathcal{D}(\mathbf{M}(G, k-2) \vee \mathbf{A}^{k-1}) \vee \mathbf{A}^k\right)\right]_{ij}\right) \\
&= \ker\left(\left[\mathcal{D}\left((\mathcal{D}(\mathbf{M}(G, k-2)) \vee \mathcal{D}(\mathbf{A}^{k-1})) \vee \mathbf{A}^k\right)\right]_{ij}\right) \\
&= \ker\left(\left[\mathcal{D}\left(\mathcal{D}(\mathbf{M}(G, k-2)) \vee \mathbf{A}^{k-1} \vee \mathbf{A}^k\right)\right]_{ij}\right) \\
&\quad \vdots \\
&= \ker\left(\left[\mathcal{D}\left(\mathcal{D}(\mathbf{M}(G, 1)) \vee \mathbf{A}^2 \vee \dots \vee \mathbf{A}^{k-1} \vee \mathbf{A}^k\right)\right]_{ij}\right) \\
&= \ker\left(\left[\mathcal{D}\left(\bigvee_{i=0}^k \mathbf{A}^i\right)\right]_{ij}\right). \tag{A.1}
\end{aligned}$$

ii. We have $[\mathbf{M}(G, k)]_{ij} = \mathcal{D}([\mathbf{M}(G, k-1)]_{ij} \vee [\mathbf{A}^k]_{ij})$. It follows from Lemma 1(iii) that $\ker([\mathbf{M}(G, k)]_{ij}) = \ker(\mathcal{D}([\mathbf{M}(G, k-1)]_{ij} \cap \ker(\mathcal{D}([\mathbf{A}^k]_{ij}))) \subseteq \ker(\mathcal{D}([\mathbf{M}(G, k-1)]_{ij}))$, which completes the proof.