

Fourier Analysis of Signals on Directed Acyclic Graphs (DAG) Using Graph Zero-Padding

Ljubiša Stanković^{a,*}, Miloš Daković^a, Ali Bagheri Bardi^{a,b}, Miloš Brajović^a,
Isidora Stanković^a

^a*University of Montenegro, Podgorica, 81000, Montenegro*

^b*Persian Gulf University, Bushehr, 7516913817, Iran*

Abstract

Directed acyclic graphs (DAGs) are used for modeling causal relationships, dependencies, and flows in various systems. However, spectral analysis becomes impractical in this setting because the eigendecomposition of the adjacency matrix yields all eigenvalues equal to zero. This inherent property of DAGs results in an inability to differentiate between frequency components of signals on such graphs. This problem can be addressed by alternating the Fourier basis or adding edges in a DAG. However, these approaches change the physics of the considered problem. To address this limitation, we propose a *graph zero-padding* approach. This approach involves augmenting the original DAG with additional vertices that are connected to the existing structure. The added vertices are characterized by signal values set to zero. The proposed technique enables the spectral evaluation of system outputs on DAGs (in almost all cases), that is the computation of vertex-domain convolution without the adverse effects of aliasing due to changes in a graph structure, with the ultimate goal of preserving the output of the system on a graph as if the changes in the graph structure were not performed.

Keywords: Directed Acycle Graph, Graph Signal Processing, Graph Fourier Transform, Zero Padding

*Corresponding author.

Email addresses: ljubisa@ucg.ac.me (Ljubiša Stanković), milos@ucg.ac.me (Miloš Daković), alibagheri@ucg.ac.me, bagheri@pgu.ac.ir (Ali Bagheri Bardi), milosb@ucg.ac.me (Miloš Brajović), isidoras@ucg.ac.me (Isidora Stanković)

1. Introduction

Graph signal processing is an emerging research area at the intersection between signal processing and graph theory, dealing with the analysis, processing, and interpretation of data defined on graphs [1, 2, 3, 4, 5, 6, 7, 8]. In many real-world applications, either the data domain or the data structure can be represented by graphs, with edges representing relationships or interactions among data points [6, 8].

One particular type of graph that is of great significance in various areas is the Directed Acyclic Graph (DAG) [9, 10, 11, 12, 13, 14]. The DAG is a special type of a directed graph, whose main characteristic is that it is free of cycles. DAGs are commonly used to model causal relationships, dependencies, and flows in various systems [9]. In this context, DAGs are used to model and analyze dynamic systems where information (or signal) flows from one vertex to another, causing a cascade of effects. This is particularly relevant in the fields such as epidemiology, finance, neuroscience, and machine learning, where understanding causal relationships between variables or events can lead to better predictions and interventions [9, 13, 15].

Spectral analysis and processing based on the standard Graph Fourier Transform (GFT) are challenging for signals on DAGs, since all eigenvalues of the adjacency matrix are equal to zero, rendering the frequency components of the analyzed signal indistinguishable [9]. The majority of endeavors aimed at introducing an appropriate candidate for the GFT concentrate on altering the Fourier basis for Jordan Normal Form (JNF) computation, which can significantly influence the graph signal processing methods [16, 17, 18, 19, 20, 21, 22, 23, 24]. To tackle the spectral analysis issue, while preserving the integrity of the Fourier basis, an algorithm has recently been proposed in [25]. The core concept entails strategically adding optimal edges to the graph until all non-trivial Jordan blocks are eliminated. This approach inherently changes the physics of the graph signal processing problem and the output of a system on this graph. It is also important to emphasize that, in practical scenarios, the identification of suitable edges is contingent upon having access to the Jordan normal form of the adjacency matrix.

The approach of altering the graph topology of a given digraph has recently garnered significant attention, particularly for the class of digraphs whose adjacency matrices are not diagonalizable. This presents notable challenges in the development of an efficient GFT [17, 18, 22, 23, 24, 25, 26].

A leading strategy proposed in [25] addresses this issue by adding new

edges to the digraph, thereby achieving diagonalizability. However, this method raises two potential practical drawbacks. First, the addition of new edges can significantly alter the topology of the original digraph, potentially compromising its structural or contextual integrity. Consequently, the efficiency and suitability of the resulting GFT become critical concerns. Second, the process of introducing these new edges may be computationally expensive, posing challenges for large-scale or dynamic graphs. These two issues have been addressed in [26].

Another prominent approach involves finding an approximate GFT for the given digraph, as discussed in detail in [18]. Mathematically, this approach seeks to identify an optimal basis whose vectors serve as approximate eigenvectors of the adjacency matrix of the original digraph. The efficiency of this method depends on the extent to which such a basis can serve as an eigenbasis for the adjacency matrix while minimizing changes to the graph topology. However, this crucial aspect—quantifying and limiting topological modifications has not been thoroughly addressed in the existing literature.

A different approach to designing a GFT is introduced in [16] and [17], which focuses on minimizing spectral dispersion to achieve a directed graph Fourier transform. Specifically, these works propose a GFT framework where the basis vectors are derived to spread frequency components effectively, capturing the inherent structure of the digraph. This method emphasizes balancing computational efficiency with topological fidelity, as the minimization process inherently seeks to preserve the original graph topology. However, challenges remain in scaling the approach to very large graphs and ensuring its adaptability to dynamically evolving digraphs.

An alternative methodology is presented in [22], which develops a graph signal processing framework for directed graphs using the Hermitian Laplacian. By constructing the Hermitian Laplacian, this approach ensures that the resulting operator is both diagonalizable and capable of capturing the structural properties of the digraph. This framework effectively retains the topological integrity of the original graph while enabling efficient computation of the GFT. However, its reliance on the Hermitian Laplacian introduces challenges related to the interpretability of the transformed domain and potential limitations in scalability for large-scale directed graphs.

In [24], a framework for directed graphs is established using normalized graph Laplacians. This approach extends the concept of Laplacians to directed graphs by constructing a normalized operator that ensures a well-defined spectral representation. The normalized Laplacian enables the definition

of a GFT while maintaining certain structural properties of the digraph. Although this method is mathematically robust and offers insights into spectral graph theory for directed graphs, practical challenges include computational overhead for large networks and the difficulty of interpreting the spectral components in the context of directed graph topology.

The main contribution of this study is introducing an algorithm to address the gap identified in [25]. This is achieved through implementing two operations. Firstly, the method of adding edges is altered. Instead of computing the JNF, the topology of the DAG is considered. Source vertices are identified in each iteration, and through appropriate connections between them, a graph with only one sink and source is formed, directing us to a Hamiltonian cycle. The introduction of new edges leads to a change in topology. Secondly, by adding some appropriate vertices (by replacing added edges with paths), and leveraging the Hamiltonian cycle, the same output of a system on graph signals on the original graph and the modified shapes can be preserved.

To this aim, we introduce a novel concept called *graph zero-padding*, inspired by the zero-padding in classical signal processing [27, p. 591], [28]. Graph zero-padding is implemented by adding some new, appropriate, vertices connected to the existing graph structure, with the signal values on those vertices set to zero. This concept opens the possibility to calculate the output of a graph system (filtering based on convolution), without aliasing effects (keeping the original DAG output as if the changes in the graph structure were not done). The proposed concept, along with the idea of forming a Hamiltonian path in a general DAG, is used to define a simple algorithm that can simultaneously eliminate the Jordan block associated with the eigenvalue zero. Indeed, through the implementation of this algorithm, all eigenvalues of the adjacency matrix of the zero-padded graph are non-zero. In the vast majority of situations, the eigenvalues will be distinct, indicating the achievement of diagonalizability, with only a few rare exceptions, that require further attention. Diagonalizability of the adjacency matrix allows the GFT analysis and application of various spectral tools on the considered graphs [29]. A simple interpretation of the speed of change (discrete frequency) in the GFT basis of a DAG, is also introduced using the eigenvector phases, as an alternative to the common total variation factor. The basic idea for the proposed concept was introduced in the special case of connected DAGs in [26].

We use the adjacency matrix as a suitable tool for the signal shift operator in directed graphs and apply it to construct graph Fourier transforms. In

classical (discrete-time domain) case, due to the distinctness of the eigenvalues of the adjacency matrix, a unique eigenvalue decomposition occurs, leading to the emergence of the discrete Fourier transform (on circular graphs). From the linear algebra perspective, any graph Fourier transform provides a basis, functioning as harmonic vectors (with specified frequencies) and allowing the representation of a graph signal on a frequency-characterized basis. Several tools, employing different approaches, have been suggested to broaden the scope of spectral analysis of graph signals in both directed and undirected graphs. Here, we will mention two more related works in a similar line but with different approaches and final goals. Instead of commencing with the decomposition of the adjacency matrix as the graph descriptor, a novel method based on a specific form of the Laplacian for directed graphs is proposed in [30] to construct a graph Fourier basis that minimizes the total variation. A similar approach to constructing a graph Fourier basis has been thoroughly investigated in [31], where the eigenvectors of the random walk operator [8] are regarded as a non-orthogonal Fourier-type basis for representing graph signals. The frequency is interpreted by connecting the variation in these eigenvectors (obtained from their Dirichlet energy) to the real part of their associated eigenvalues.

Finally, we will emphasize that our final goal is to use the adjacency matrix as a natural shift operator and to change the graph in such a way that the GFT and spectral analysis is possible, while the output of a graph system (graph filter) on the altered graph is the same as if the graph were not changed. This has not been considered or achieved in any other paper.

The paper is organized as follows. In Section 2, the basic theory regarding graphs signals on a graph, systems on a graph, and the GFT is presented. Next, in Section 3 we present some fundamental properties of DAGs. Classical signal processing in a graph framework is presented in Section 4. In Section 5 we present graph zero-padding on a special case of connected DAG. General DAG zero-padding is presented in Section 6. Section 7 concludes the paper. The proposed approach is illustrated in numerical examples.

2. Graph Signal Processing

Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consisting of a finite set \mathcal{V} of N vertices, and set $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ of edges, connecting the vertices, and reflecting their mutual relations. Each edge is represented by ordered pair of vertices $(m, n) \in \mathcal{V} \times \mathcal{V}$ indicating a starting vertex m and an ending vertex n . The graph can be

modeled by the adjacency matrix \mathbf{A} . The elements a_{mn} of this matrix have value 1 if an edge between vertex m and vertex n exists, and value 0 if such edge does not exist.

A graph signal is defined as an N -dimensional vector

$$\mathbf{x} = [x(1) \ x(2) \ \dots \ x(N)]^T, \quad (1)$$

where the signal value $x(n)$ is associated to the n -th vertex of the considered graph. Graph shift operator is a matrix \mathbf{T} that converts given graph signal \mathbf{x} into its shifted version \mathbf{y} ,

$$\mathbf{y} = \mathbf{T}\mathbf{x}.$$

The simplest and most commonly used shift operator on directed graphs is the adjacency matrix, i.e. $\mathbf{T} = \mathbf{A}$. That kind of shift is called a “backward shift”.

A linear system on a graph is defined as a linear combination of the signal and its shifted versions. The output of a linear system on graph signal can be calculated as

$$\mathbf{y} = h_0\mathbf{x} + h_1\mathbf{T}\mathbf{x} + h_2\mathbf{T}^2\mathbf{x} + \dots + h_S\mathbf{T}^S\mathbf{x}, \quad (2)$$

where S denotes the system order, and h_s , $s = 0, 1, \dots, S$, are the system coefficients, corresponding to classical impulse response [5, 6, 7].

Note that every system on a given signal graph can be determined with $S < N$ coefficients, due to the fact that an arbitrary matrix power \mathbf{T}^n can be expressed as a linear combination of matrix powers \mathbf{T}^m , $m = 0, 1, \dots, N - 1$.

The GFT is, for the case of directed graphs, defined with the assumption that the adjacency matrix is diagonalizable, meaning that there exists an invertible matrix \mathbf{V} and a diagonal matrix $\mathbf{\Lambda}$ such that

$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}.$$

Diagonal elements of matrix $\mathbf{\Lambda}$ are the eigenvalues, λ_k , $k = 1, 2, \dots, N$, of the adjacency matrix \mathbf{A} , whereas the columns of \mathbf{V} are the eigenvectors, \mathbf{v}_k , $k = 1, 2, \dots, N$. The GFT of a signal, $x(n)$, on vertices $n = 1, 2, \dots, N$, is defined as

$$X(k) = \sum_{n=1}^N x(n)u_k(n). \quad (3)$$

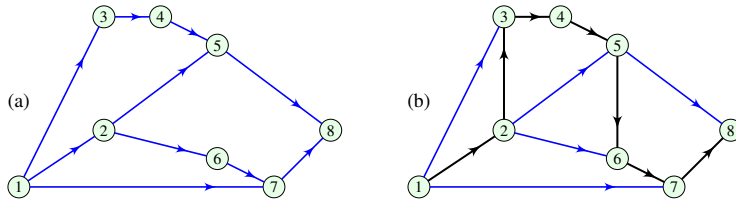


Figure 1: Examples of a directed acyclic graph (DAG): (a) Disconnected DAG (for example, a path connecting vertices 2 and 3 does not exist), (b) Connected DAG (there exists a path between any pair of vertices). A Hamiltonian path $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8$, which visits all vertices of the connected DAG exactly once, is denoted by black lines.

Here, $u_k(n)$ represents the value of the k th column of matrix \mathbf{V}^{-1} , corresponding to the vertex n . The matrix form of the GFT reads $\mathbf{X} = \mathbf{V}^{-1}\mathbf{x}$. The inverse graph Fourier transform (IGFT) is defined by

$$x(n) = \sum_{k=1}^N X(k)v_k(n), \quad (4)$$

or, in a matrix form, $\mathbf{x} = \mathbf{V}\mathbf{X}$. Here, $v_k(n)$ is the value of the k th eigenvector (column of matrix \mathbf{V}), corresponding to the vertex n .

The system on a graph signal can be written using the GFTs of the input and output signal as [6]

$$\begin{aligned} \mathbf{Y} &= (h_0\mathbf{I} + h_1\mathbf{\Lambda} + h_2\mathbf{\Lambda}^2 + \cdots + h_S\mathbf{\Lambda}^S)\mathbf{X}, \quad \text{or} \\ \mathbf{Y} &= \mathbf{H}(\mathbf{\Lambda})\mathbf{X}, \end{aligned} \quad (5)$$

where $\mathbf{H}(\mathbf{\Lambda})$ is the system transfer function.

3. Directed Acyclic Graphs (DAG)

In many cases of practical interest, especially for DAGs, the adjacency matrix is not diagonalizable and the Fourier analysis cannot be performed without graph modifications. Two examples of a DAG with $N = 8$ vertices are presented in Fig. 1. The graph Fig. 1(a) is not connected, and the graph Fig. 1(b) is connected.

Every DAG induces partial ordering on a vertex set. If vertex numbering follows this partial ordering, then the adjacency matrix will be an upper triangular matrix, with zeros on the main diagonal (since loops are not

allowed in acyclic graphs). A connected DAG is a specific case of these graphs, where for each pair of vertices (m, n) there exists a path from m to n or a path from n to m . For a connected DAG, the total ordering is achieved, as well as a unique numbering of graph vertices that corresponds to a Hamiltonian path (a path that visits all vertices of a connected DAG exactly once), as shown in Fig. 1(b). The existence of a Hamiltonian path in a DAG enforces a specific topology on the edge directions. All edges align with the Hamiltonian path direction, leading to the uniqueness of the Hamiltonian path. The uniqueness of the Hamiltonian path implies that two connected DAGs with different adjacency matrices (in upper triangular form) are non-isomorphic. This further indicates that the total number of connected DAGs with N vertices is

$$N_{\text{ConDAG}} = 2^{\frac{(N-1)(N-2)}{2}}. \quad (6)$$

We can derive this formula starting from the adjacency matrix of connected DAG, with nodes numbered according to their position in the Hamiltonian path, which have ones at the super-diagonal. Above the super-diagonal, we have $(N-2) + (N-3) + \dots + 1 = (N-1)(N-2)/2$ elements that could be either 1 or 0, so the total number of connected DAGs directly follows.

Remark: Adjacency matrix of a DAG is nilpotent with nilpotency index smaller than N . This means that there exists a nilpotency index m , such that $\mathbf{A}^m = \mathbf{0}$ and $\mathbf{A}^{m-1} \neq \mathbf{0}$. This further implies that all eigenvalues of a DAG adjacency matrix are equal to 0.

If the adjacency matrix, \mathbf{A} , is used as a shift operator on DAGs, then any system on a DAG can be written in the following form

$$\mathbf{y} = h_0\mathbf{x} + h_1\mathbf{A}\mathbf{x} + h_2\mathbf{A}^2\mathbf{x} + \dots + h_S\mathbf{A}^S\mathbf{x}, \quad (7)$$

where $S + 1$ is the nilpotency index of \mathbf{A} .

4. Classical Zero-Padding within the Graph Framework

If we have a classical signal in the discrete-time domain, $x(n)$, of length N , then its domain can be presented in the form of a directed path graph with N vertices, as shown in Fig. 2(a).

Remark: It is well known that for the calculation of the discrete Fourier transform (DFT) of this signal we must assume signal periodicity. In graph terminology, it means that we have to make the graph domain of the signal in a circular manner. The minimum number of samples for the DFT calculation of

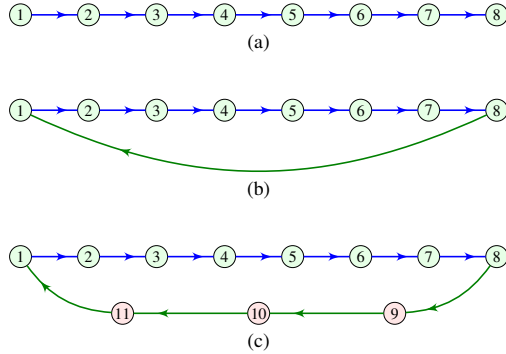


Figure 2: (a) The domain of classical discrete-time signal of length $N = 8$. (b) The domain of classical signal commonly used in DFT calculations with $N = 8$ (for example, as in [25]). (c) The domain of the zero-padded classical signal required for DFT-based calculations of the output of an FIR system order $M \leq 3$, with $N = 8$.

the assumed signal is N , meaning that we inherently add one edge connecting the last and the first vertex, see Fig. 2(b).

This necessary and obvious domain modification can be interpreted and justified within the graph terminology as well. The adjacency matrix, \mathbf{A} , of the path graph from Fig. 2(a) is a super-diagonal matrix

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

with all eigenvalues following from $\det(\mathbf{A} - \lambda \mathbf{I}) = 0$, being

$$\lambda_k = 0, \quad \text{for } k = 1, 2, \dots, N.$$

This adjacency matrix \mathbf{A} is not diagonalizable and the Fourier analysis on this graph (acting as the signal domain) is not possible). Note that, if we assume an undirected graph as the signal domain, then the Fourier analysis would be possible. Such an assumption, however, completely changes the physics of the problem, and is, therefore, not appropriate in this context.

Diagonalization by making the domain circular. If we make the domain circular, by adding one edge from the last (sink) vertex to the first (source) vertex, we get the adjacency matrix

$$\mathbf{A}_C = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \quad (8)$$

This matrix is obviously diagonalizable. Eigenvalues of matrix \mathbf{A}_C , of size $N \times N$, are obtained as the solutions of

$$\lambda^N = 1.$$

They are all distinct and equal to

$$\lambda_k = e^{j\frac{2\pi}{N}(k-1)}, \quad k = 1, 2, \dots, N,$$

where k is the frequency index.

Frequency interpretation. For $1 \leq k \leq \frac{N}{2}$ we talk about “positive” frequencies, while indices $\frac{N}{2} < k \leq N$ are associated to “negative” frequencies. Therefore, the graph spectral analysis on a circular graph coincides with classical DFT analysis. Note that the discrete frequency $\omega_k = \frac{2\pi}{N}(k-1)$ can be obtained as an angle of the corresponding eigenvalue λ_k , that is,

$$\omega_k = \arg\{\lambda_k\}.$$

Note that, in classical DFT, the frequency indicates the speed of change in the eigenvectors. Another way to measure the changes in the eigenvectors (with GFT basis) is by using the total variation [6, 14], i.e.

$$E_{TV}(k) = \left| 1 - \frac{\lambda_k}{\max_m |\lambda_m|} \right|.$$

This parameter is used in graph signal processing (especially for undirected graphs), to sort the eigenvectors from low to high frequency, but it cannot distinguish “positive” and “negative” frequencies.

Signal processing and zero-padding. Assume now that we want to process the considered classical signal $x(n)$ by a general FIR system whose impulse response h_n is of length $(S + 1)$, that is, by a system of order S . This classical system is defined within the graph framework by (2). Its discrete-time domain realization can be written as

$$y(n) = h_0x(n) + h_1x(n - 1) + \cdots + h_Sx(n - S).$$

If we want to use the DFT to calculate the output (which is almost a standard routine in DSP processors), then we must zero-pad the original signal by at least $M = S$ zeros before the DFT is applied (as illustrated in Fig. 2(c)) , [27, p. 591], [28]. After zero-padding, the DFT will produce the output that corresponds to the original non-periodic output signal that is shown in Fig. 3.

Within the graph terminology, the zero-padding of the original signal (before we make it periodic) means that we must add at least M vertices in the backward path from the sink to the source vertex, as illustrated in Fig. 4. The adjacency matrix of the zero-padded graph is of the same form as in (8), but of order $N + M$. It is diagonalizable with all distinct eigenvalues

$$\lambda_k = e^{j\frac{2\pi}{N+M}(k-1)}, \quad k = 1, 2, \dots, N + M,$$

and corresponds to the DFT analysis of a signal with $N + M$ samples, after zero-padding.

5. Connected DAG Zero-Padding

Connected DAG has one source vertex and one sink vertex. Moreover, there exists a path from sink to source passing through every vertex in a DAG (Hamiltonian path).

Based on the discussion in the previous section, we can conclude that a possible solution for a non-diagonalizable adjacency matrix is to add an edge from the sink to the source vertex in order to obtain a new graph with a diagonalizable adjacency matrix. If we want to keep the same output of a graph filter (of an order less than or equal to M) on the original and a modified DAG, then instead of adding an edge we should add a path of length M connecting the sink and the source in the original connected DAG.

With the described zero-padding, the shifts produced by $\mathbf{A}^m \mathbf{x}$, $m = 0, 1, 2, \dots, M$ on the original graph, will correspond to the shifts on the zero-padded graph, while the shifted signal values, will be trapped in the added

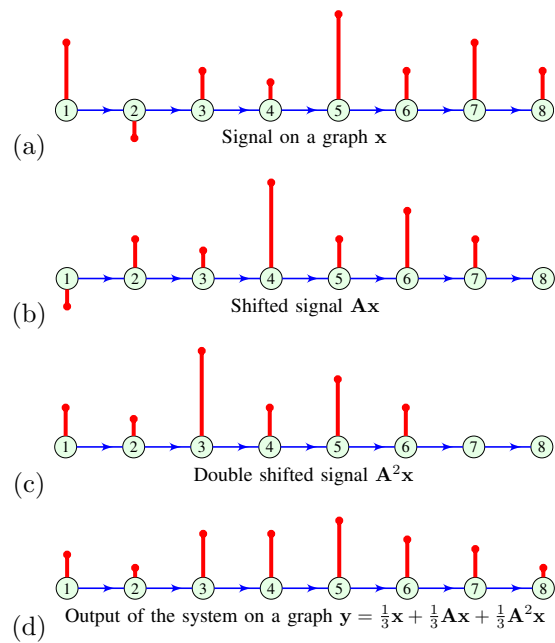


Figure 3: (a) Signal $x(n)$ of length $N = 8$ on a graph as its domain; (b,c) Shifted versions of the same signal; (d) Output of a system on a graph of order $S = 2$ with system coefficients equal to $1/3$.

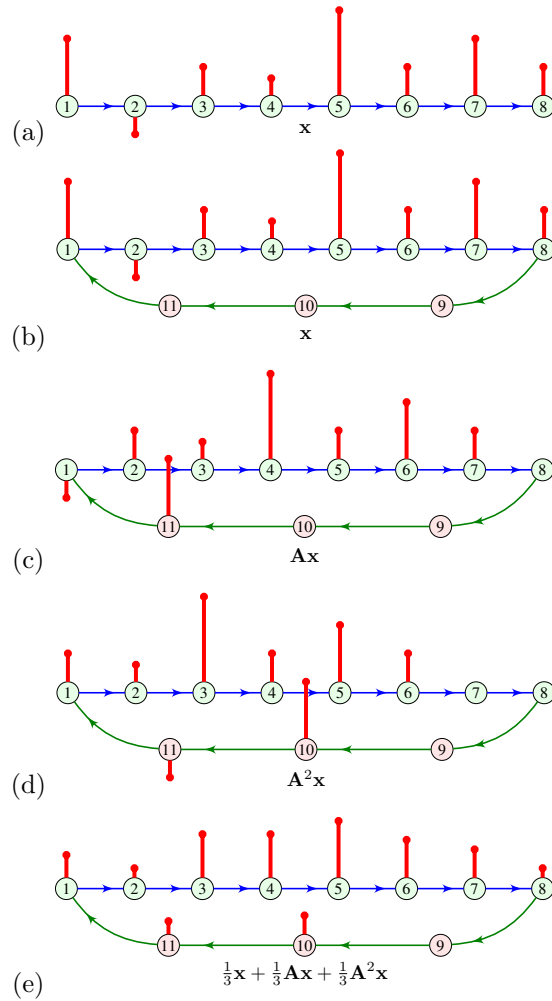


Figure 4: (a) Signal $x(n)$ of length $N = 8$ on a graph as its domain. (b) Signal on a graph as it must be used in the DFT output calculation on a FIR system whose order is $S \leq 3$. (c) Signal $x(n - 1)$ shifted on a graph from (b). (d) Signal $x(n - 2)$ on a graph from (b). (e) Output signal as a circular (DFT) convolution $\mathbf{y} = \mathbf{x} * [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$ equal (within the basic period) to the aperiodic convolution of the same signal and system.

path from the sink to the source that is long enough so they do not influence the signal at the vertices that belong to original graph, during the system processing time.

The adjacency matrix of the zero-padded graph can be presented in a block form as

$$\mathbf{A}_{ZP} = \begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{D} & \mathbf{J} \end{bmatrix}, \quad (9)$$

where \mathbf{A} is $N \times N$ adjacency matrix of the connected DAG, \mathbf{J} is a matrix of size $M \times M$, with ones at super-diagonal and zeros elsewhere. Matrix \mathbf{C} is of size $N \times M$, with all zeros except the lower left corner, where it has value 1. This block represents the connection of the DAG sink to the first vertex of the added path graph. Matrix \mathbf{D} is of size $M \times N$ with all zeros except the lower left corner where it has value 1. This block represents the edge from the last vertex of the added path graph to the source of the DAG.

Determinant in the zero-padded graph: It is easy to check that determinant of \mathbf{A}_{ZP} is non-zero. To be more specific, its values are either 1 or -1 .

To illustrate the proposed concept, consider a connected DAG with $N = 8$ vertices (Fig. 1(b)), with the adjacency matrix

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

For zero-padding with $M = 2$ additional vertices, the matrix \mathbf{A}_{ZP} has the

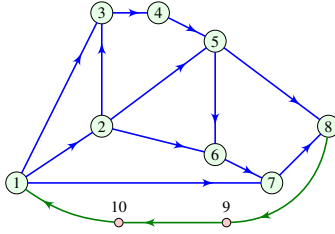


Figure 5: An illustration of a zero-padded DAG. The DAG shown in Fig. 1 (b) is zero-padded with additional vertices (marked in red) and edges (highlighted in green), necessary for the GFT-based analysis.

following form

$$\mathbf{A}_{ZP} = \left[\begin{array}{cccccccc|cc} 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right],$$

and we obtain a zero-padded graph presented in Fig. 5. This adjacency matrix is now diagonalizable and a system on this DAG can be analyzed and implemented in the spectral domain.

5.1. Numerical Examples

Five examples of zero-padded connected DAGs are presented in Fig. 6. The first case (a path graph) is the simplest. This graph transforms into a cycle graph after zero-padding (here we use zero-padding with $M = 2$), shown in Fig. 6(a). The classical Fourier analysis follows straightforwardly from the eigendecomposition of the adjacency matrix. Eigenvalues are equally spaced on the unit circle, as shown in Fig. 6 (second column).

In more complex DAG examples, as in Fig. 6(b,c,d), more edges are added to the path graph in order to keep the connectivity and avoid cycles. Here, we present graphs with 3, 5, and 10 additional edges. In all of these cases, we can see from Fig. 6 that the eigenvalues lie close to the unit circle,

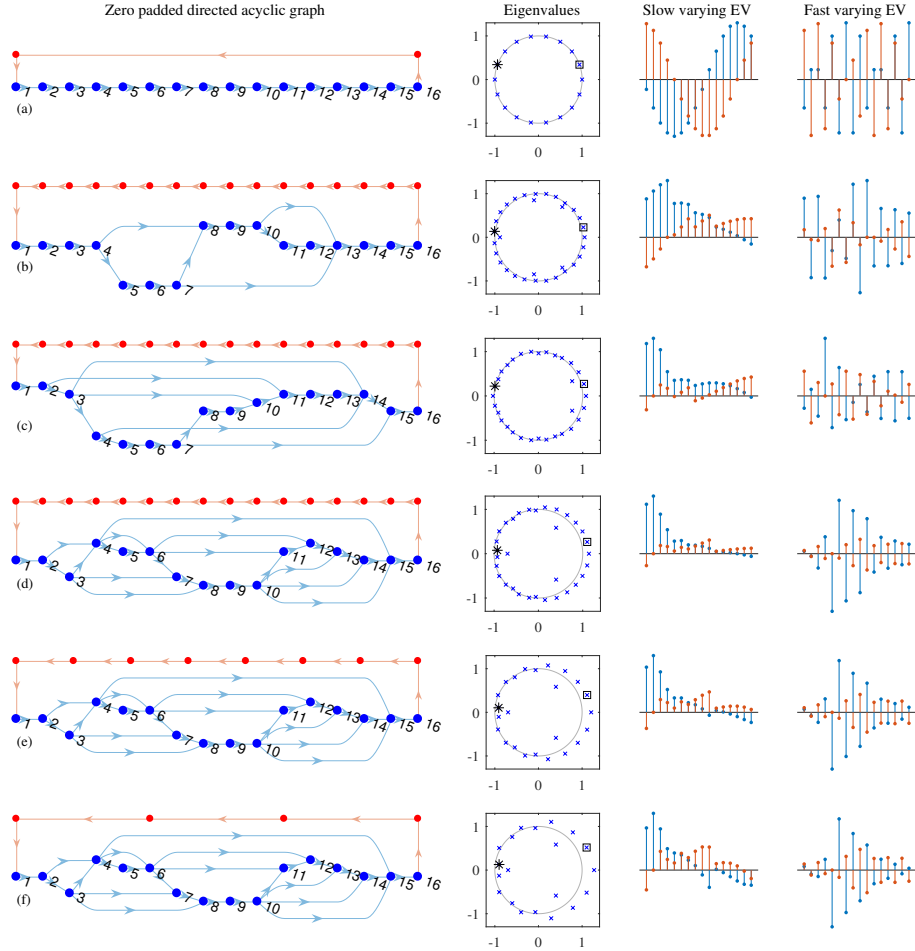


Figure 6: Examples of zero-padded connected DAGs. Simple path graph (a) zero-padded with $M = 2$ vertices; connected DAGs with 3, 5, and 10 additional edges compared to the path graph from (a), and zero-padded with $M = 16$ vertices (b-d); the same DAG as in (d) zero-padded with $M = 8$ and $M = 4$ (e,f). Original DAGs are presented with blue vertices and edges. Nodes and edges added by zero-padding are colored in red. For each zero-padded graph eigenvalues are presented. Two eigenvectors that correspond to a low and to a high frequency eigenvalue are also presented (real parts in blue and imaginary parts in red). The corresponding eigenvalues are marked by a square (low-frequency) and a star (high-frequency) on the eigenvalues plot.

with nonuniform, but almost equal spacing. Here, graph zero-padding with $M = N = 16$ is used. Finally in Fig. 6(e,f), the same DAG as in Fig. 6(d) is zero-padded with $M = 8$ and $M = 4$ vertices, respectively. Using zero-padding with $M < N$ guarantees equality of the output signal calculated in the vertex domain and output calculated using the spectral domain for systems of order up to M .

For each considered graph in Fig. 6, two eigenvectors are presented on the right. Since eigenvectors are complex, we present their real part in blue and the imaginary part in red. We choose one slow-varying eigenvector that corresponds to a low frequency (marked by a square in the eigenvalue plot in Fig. 6) and one fast-varying eigenvector with the corresponding eigenvalue marked by a star. From these plots, we observe that the concept of slow-varying and fast-varying Fourier basis functions (eigenvectors), obvious in the classical Fourier domain (circular graph), remains unchanged in the considered case of GFT defined on a zero-padded DAG.

6. General DAG zero-padding

For a general DAG, we should consider adding more than one edge (or path, if zero-padding is applied) to the original DAG. This step ensures that the adjacency matrix of the new graph becomes diagonalizable, with all non-zero eigenvalues, making the GFT analysis possible. Determining how to add the minimal number of edges to make the adjacency matrix diagonalizable is not an easy task. It can be solved using a combinatorial approach (exhaustive search), but only for a small graph size N . Another recently proposed approach uses matrix perturbation theory [25]. Here, we will propose a simple and fast, albeit sub-optimal strategy. It can be presented in two steps:

- First, we add a sufficient number of edges in order to make the DAG connected, while preserving the acyclic nature of the graph. We can use the procedure described in Algorithm 1. Note that if we intend to implement zero-padding, then each edge added in Algorithm 1 should be replaced by a path of length $M + 1$, incorporating M additional vertices along it.
- Secondly, we add an edge connecting sink and source of the connected DAG, obtained in the previous step.

Table 1: Total number of connected DAGs , number of DAGs, among them, whose eigenvalues of the adjacency matrix are all distinct after adding sink-to-source edge, and the number of such DAGs with an algebraic multiplicity of the adjacency matrix eigenvalue greater than one.

	Total DAGs	Diagonal.	Algeb. Mult. > 1	
$N = 7$	32,768	32,250	518	1.58%
$N = 7$ (1-ZP)	32,768	32,758	10	0.03%
$N = 8$	2,097,152	2,075,682	21,470	1.02%
$N = 8$ (1-ZP)	2,097,152	2,088,106	9,046	0.43%
$N = 8$ (2-ZP)	2,097,152	2,095,224	1,928	0.09%

Within Algorithm 1, we aim to make the DAG connected by establishing connections between existing vertices.

This procedure is performed directly, meaning that we search for all sources in a DAG, then proceed with determining the edges that will link these sources, subsequently removing these sources from the DAG. By repeating this process, we ensure that a sufficient number of edges is added to achieve connectivity. At the end, we find a Hamiltonian path in the new (connected) DAG and identify which edges from the previous procedure belong to the Hamiltonian path. These edges are essential for achieving the connectivity of the DAG. Any added edges outside the Hamiltonian path can safely be removed.

The while loop in Algorithm 1 will be executed at most N times, since at least one vertex is removed from \mathcal{G}_1 in each iteration. Within the while loop, identifying all sources in \mathcal{G}_1 requires $O(N^2)$ operations. The total computational complexity of Algorithm 1 is $O(N^3)$.

If we renumber vertices in the connected DAG according to their position in the Hamiltonian path, the adjacency matrix becomes upper triangular, with ones on the super-diagonal. Adding vertices in the return path with an appropriate numbering (sink + 1, sink + 2, ...) keeps ones at the adjacency matrix super-diagonal and adds 1 into its lower left corner.

Denote by K the number of edges needed to make the DAG connected. If we zero-pad each added edge, replacing it with a path consisting of M vertices, the size of the obtained adjacency matrix of the connected zero-padded DAG increases to $N + KM$. Adding a zero-padded return path additionally increases the adjacency matrix size to $N + (K + 1)M$.

Invertibility, Diagonalizability, and Algorithm’s Efficacy. After

Algorithm 1 Establishing Connectivity in a DAG

Input: Directed acyclic graph \mathcal{G} **Output:** Connected directed acyclic graph \mathcal{G}_2 $\mathcal{G}_1 \leftarrow \mathcal{G}$ $\mathcal{G}_2 \leftarrow \mathcal{G}$ **while** \mathcal{G}_1 is not empty **do** $s \leftarrow$ list of all sources in \mathcal{G}_1 **if** there is more than one source in s **then** connect sources in s by a path in \mathcal{G}_2 **else** remove source s from \mathcal{G}_1 **end if****end while** $h \leftarrow$ Hamiltonian path in \mathcal{G}_2 remove all edges from \mathcal{G}_2 that are not in \mathcal{G} and do not belong to h **return** \mathcal{G}_2

Algorithm 1 is executed, the adjacency matrix attains full rank, indicating that it becomes invertible. This property implies that all eigenvalues of the considered matrix are non-zero. However, the invertibility does not guarantee diagonalizability. The nondiagonalizability of the adjacency matrix of a graph obtained after Algorithm 1 is a very rare event. Even in these rare cases, after the added edges are zero-padded, it often happens that the resulting adjacency matrix becomes diagonalizable.

For relatively small graphs, with, for example, $N = 7$ and $N = 8$, it is feasible to construct all possible connected DAGs, with added sink-to-source connection (edge or path), and to examine the algebraic multiplicity of the eigenvalues. Having all distinct eigenvalues is sufficient for the diagonalizability. By adding only sink-to-source edge to all considered DAGs, the diagonalizability is achieved in 98,42% and 99% for $N = 7$ and $N = 8$, respectively. These results are presented in Table 1, rows one and three.

After adding one or two vertices to the sink-to-source path, we observe that the adjacency matrix's diagonalizability increases to above 99.6% of cases (rows two, four, and five in Table 1).

It is very interesting that all 518 cases (out of 32768) that were not diagonalizable with $N = 7$, become diagonalizable after adding one vertex in the sink-to-source path. However, 10 graphs that were diagonalizable after

the sink-to-source edge is added, become nondiagonalizable after this vertex is added. We checked that in all 32768 possible connected DAGs with $N = 7$, and in all 2097152 possible connected DAGs for $N = 8$, the diagonalizability is achieved either with a single sink-to-source edge or with one added vertex in the sink-to-source path. Since the number of added vertices, could be changed, as far as their number is greater or equal to the order of a system on the graph, it means that *in this way the diagonalizability can be achieved, in practically (almost) all cases.*

Weighted DAG case. The proposed approach can be directly extended to weighted DAGs (including the case of a random walk matrix used as shift in the system). The only modification required is to assign weights to the added edges.

In this manner, addressing the diagonalizability issue becomes even simpler, as we have flexibility in choosing the weights for the added edges. For instance, in connected DAGs of size $N = 7$ and $N = 8$, there are cases where adding a single unweighted edge is insufficient for achieving diagonalizability (refer to Table 1). However, by introducing a weighted edge with a weight of 0.5, all instances become diagonalizable.

6.1. Graph Zero-Padding from a Linear Algebraic Perspective

In the initial step, it becomes clear that the nilpotency of the adjacency matrix, with nilpotency index P , indicates that the geometrical multiplicity of the unique eigenvalue 0 is exactly $P + 1$. This, in turn, implies that in the Jordan normal form, there are precisely $P + 1$ nontrivial Jordan blocks associated with the eigenvalue 0. To achieve diagonalizability, it is imperative to eliminate all of these blocks. Algorithm 1 addresses the task of consolidating all these non-trivial Jordan blocks into a single block while maintaining the acyclic nature of the graph.

After successfully constructing the Hamiltonian path, an extra edge is added to connect the sink to the source. This requires the application of a new perturbation, which is in the form of a rank-one elementary matrix. Empirical evidence confirms that, in almost all scenarios, diagonalizability is successfully achieved in this phase, especially after zero-padding and adding additional vertices.

In exceptional cases, the second perturbation may introduce a few non-trivial Jordan blocks, which can also be resolved by the inclusion of a small number of randomly positioned edges. Through these insightful observations and a careful examination of Algorithm 1 on graphs, significant progress can

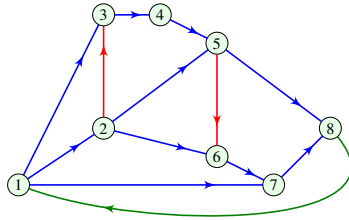


Figure 7: An example of a DAG with additional edges needed for the GFT analysis. Red edges are used to make the DAG connected with a Hamiltonian path 1-2-3-4-5-6-7-8, while the green edge is used for a sink-to-source edge, used to convert the Hamiltonian path to a Hamiltonian cycle.

be achieved by addressing the challenges in the way discussed in the recent related work [25].

The non-singularity of the GFT leads us to derive a boundary condition [25] when signals are presented in the polynomial (z -transform) representation. Suppose that the GFT is of size N . It can be considered as a polynomial transform that maps the N -dimensional space of polynomials, $\mathbb{C}_{N-1}[z]$, to \mathbb{C}^N . The Chinese Remainder Theorem implies the existence of a polynomial $p(z)$ of a degree N , such that it establishes an isomorphism between the quotient ring $\mathbb{C}[z]/p(z)$ and \mathbb{C}^N . The polynomial $p(z)$ describes the boundary condition for the considered case.

6.2. Examples

We will illustrate zero padding procedure for general DAG by applying it on a simple graph given in Fig 1(a), more complex graph with 20 vertices and USA temperature map graph.

For the DAG presented in Fig 1(a), we should add two edges in order to produce a connected DAG (for example, edges (2, 3) and (5, 6)). These edges are shown in red color in Fig. 7. By adding the return edge (8, 1), indicated by the green color in Fig. 7, we obtain the graph where we can define the GFT.

If we want to keep the same behavior of the original DAG and the modified DAG for signal processing and systems on graphs of maximal order M , we should insert new paths in the graph with additional M vertices, instead of each edge which is added in the previous procedure. For the DAG presented in Fig. 1(a), by using $M = 2$ and introducing new vertices, the obtained zero-padded graph is presented in Fig. 8 (a).

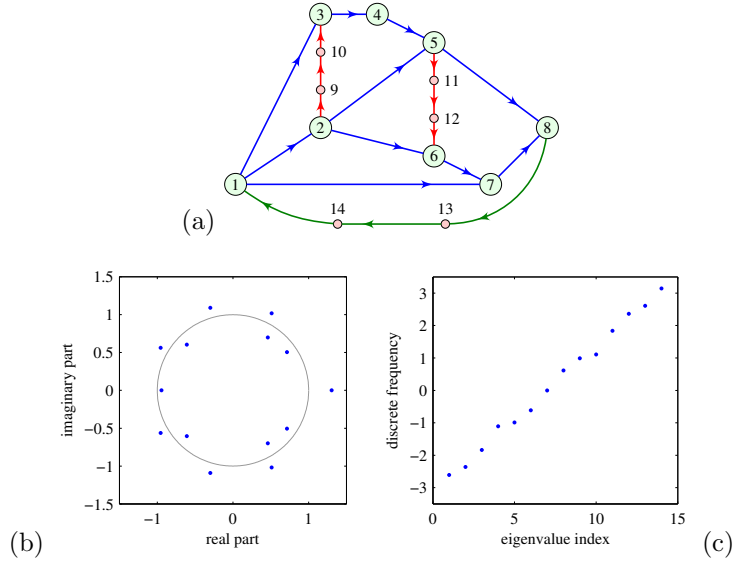


Figure 8: (a) An example of a zero-padded graph with additional vertices needed for achieving the same behavior of systems of order less than or equal to $S = 2$ on the original DAG and the obtained zero-padded graph. (b) Eigenvalues of the zero-padded graph; (c) Relationship between the eigenvalue index and the corresponding discrete frequency.

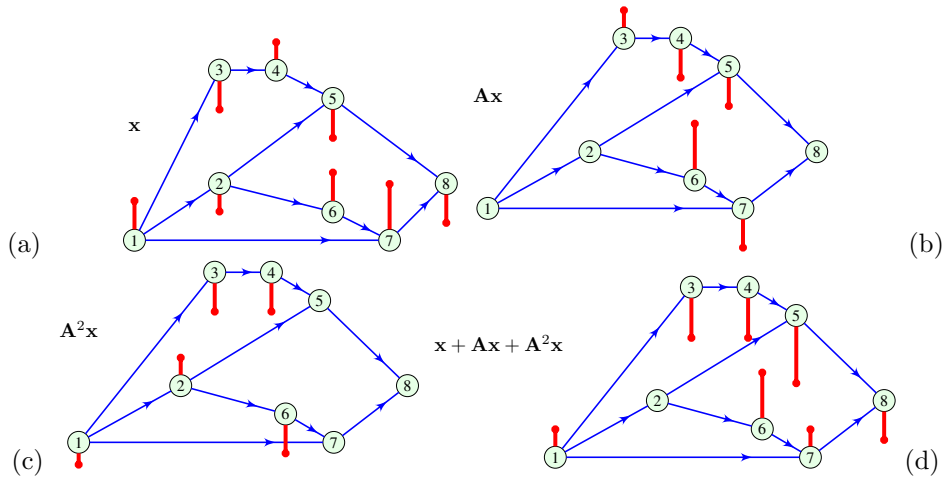


Figure 9: An example of signal processing using the second order system on a DAG.

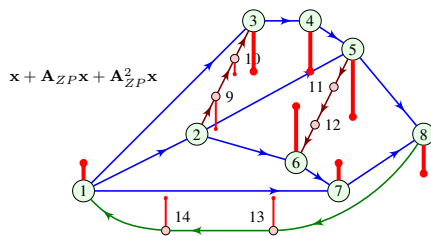


Figure 10: Signal processing on the DAG from Fig. 9, after zero-padding is done, so that the GFT can be used.

In the considered case, we can add more than M zero-padding vertices in the added paths (sometimes M is not known in advance). Having in mind that the adjacency matrix of the original DAG, shown in Fig. 1(a), has a nilpotency index of 5, meaning that any system on that graph is of order smaller than 5. By inserting $M = 4$ vertices in each path, an arbitrary system defined on the original DAG and applied to the zero-padded DAG will produce the same output (when we neglect signal values at the vertices added through the zero-padding procedure).

As an example, consider a signal on the DAG presented in Fig. 9(a). Shifted signals are presented in the same Figure (b,c). When we apply a system of the second order on this graph, defined by

$$\mathbf{y} = \mathbf{x} + \mathbf{A}\mathbf{x} + \mathbf{A}^2\mathbf{x},$$

the output signal is given in Fig. 9(d).

The same signal and system are then used on a zero-padded DAG shown in Fig. 8(a), assuming zero signal values at the added vertices (9, 10, ..., 14). The system output is presented in Fig. 10. The output signal at vertices (1, 2, ..., 8) is the same as in the case of the original DAG (see Fig. 9(d)). Signal values at the vertices added to the original DAG by zero-padding procedure could be discarded.

A more complex example is shown in Fig. 11(a). Here, we consider a DAG with $N = 20$ vertices and 54 edges. There are three sources (1, 3 and 5) and two sinks (19 and 20). This DAG is obviously not connected. Next, we add edges, according to Algorithm 1, in order to make the DAG connected. Added edges are shown in red color in Fig. 11(b). Then, we perform zero-padding by replacing each added edge by a path graph with $M = 3$ additional vertices, for each path, and adding a return path from the sink to the source, marked in green color in Fig. 11(c). Each vertex added by zero-padding is marked by

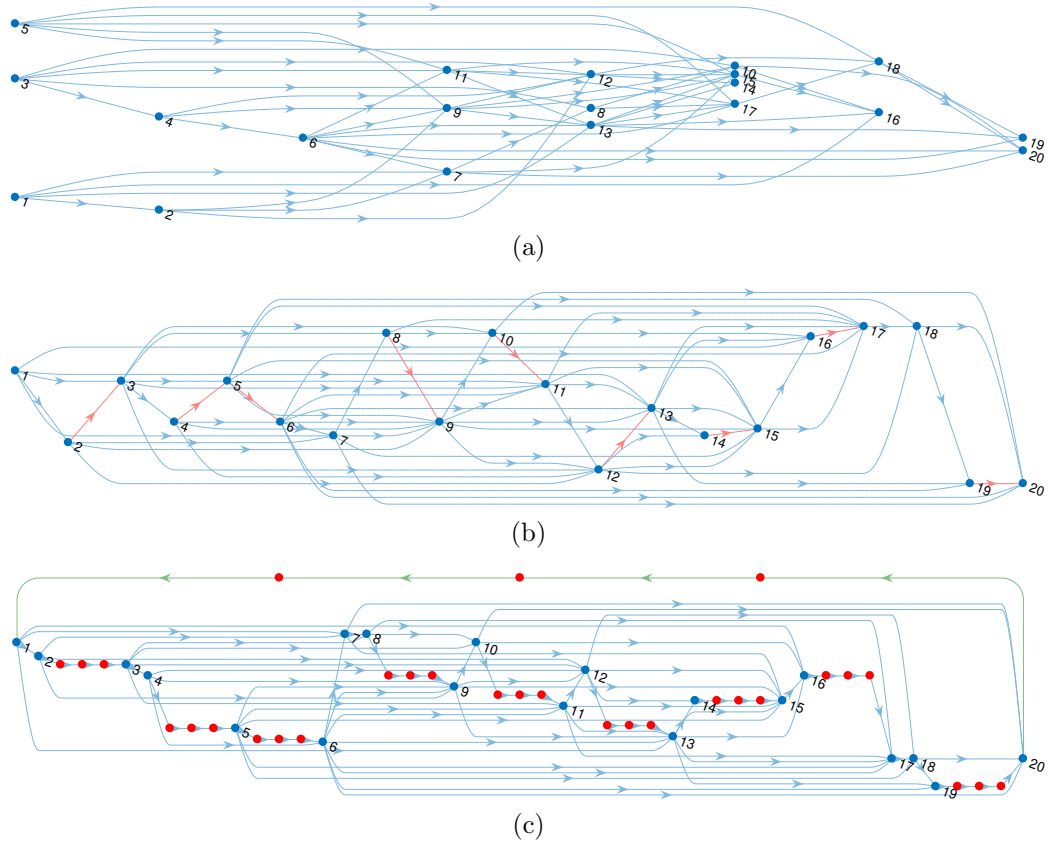


Figure 11: (a) Original DAG, (b) connected DAG with added edges marked by red color, (c) zero-padded DAG with added vertices ($M = 3$) marked by red color and return path marked by green color.

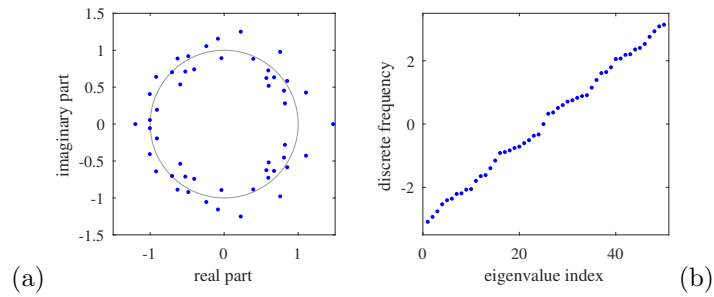


Figure 12: (a) Eigenvalues of the zero-padded connected DAG shown in Fig. 11(c); (b) the relation between the eigenvalue index and the discrete frequency value.

red color. The eigenvalues of the obtained zero-padded graph are given in Fig. 12(a). They are all distinct, ensuring the diagonalizability of the adjacency matrix. The relation between the eigenvalue index and the discrete frequency value, calculated as an angle of the corresponding eigenvalue is presented in Fig. 12(b). It can be seen that this relation is close to the linear one.

The signal processed on this zero-padded graph with a system of up to the third order would remain the same as in the original DAG, where the Fourier analysis was not possible. In order to check this numerically, we have generated a random uniformly distributed signal from 0 to 1 with 20 samples, $\mathbf{x} = \text{rand}(20, 1)$ on the graph given in Fig. 11(a). The graph and the signal are zero-padded according to the presented procedure (see Fig. 11(c)). Then the GFT of the zero-padded signal \mathbf{x}_{ZP} is calculated as $\mathbf{X}_{ZP} = \mathbf{V}^{-1}\mathbf{x}_{ZP}$. The obtained GFT is multiplied by the system transfer function

$$\mathbf{H}(\Lambda) = h_0\mathbf{I} + h_1\Lambda + h_2\Lambda^2 + h_3\Lambda^3,$$

in order to obtain the GFT of the output signal $\mathbf{Y}_{ZP} = \mathbf{H}(\Lambda)\mathbf{X}_{ZP}$. The output signal is calculated using the IGFT, as $\mathbf{y}_{ZP} = \mathbf{V}\mathbf{Y}_{ZP}$. Its values on the vertices from the original DAG are the same as the ones obtained by a direct calculation in the vertex domain.

Furthermore, we will consider the USA temperature map, commonly used as a benchmark case for directed graphs [17, 22, 25]. The results are shown in Fig. 13. As the number of zero-padded vertices, defined by M , increases, it becomes evident that the eigenvalues are located closer to the unit circle (Fig. 13(b,e)). Relationship between eigenvalue index and discrete frequency is almost linear (Fig. 13(c,f)) The GFT of the temperature signal calculated using $M = 3$ and $M = 15$ are also presented in Fig. 13(d,g).

7. Conclusion

In this paper, we have proposed a graph zero-padding concept in order to overcome the inherent limitations associated with spectral signal analysis and processing on DAGs. The main idea is motivated by the classical zero-padding technique and its interpretation within the graph signal processing framework.

In the case of a connected DAG, the extension is straightforward. It involves adding a single backward path (or edge), whose length is equal to the maximum order of the system used for graph signal processing. The determinant of the adjacency matrix of the zero-padded graph becomes either

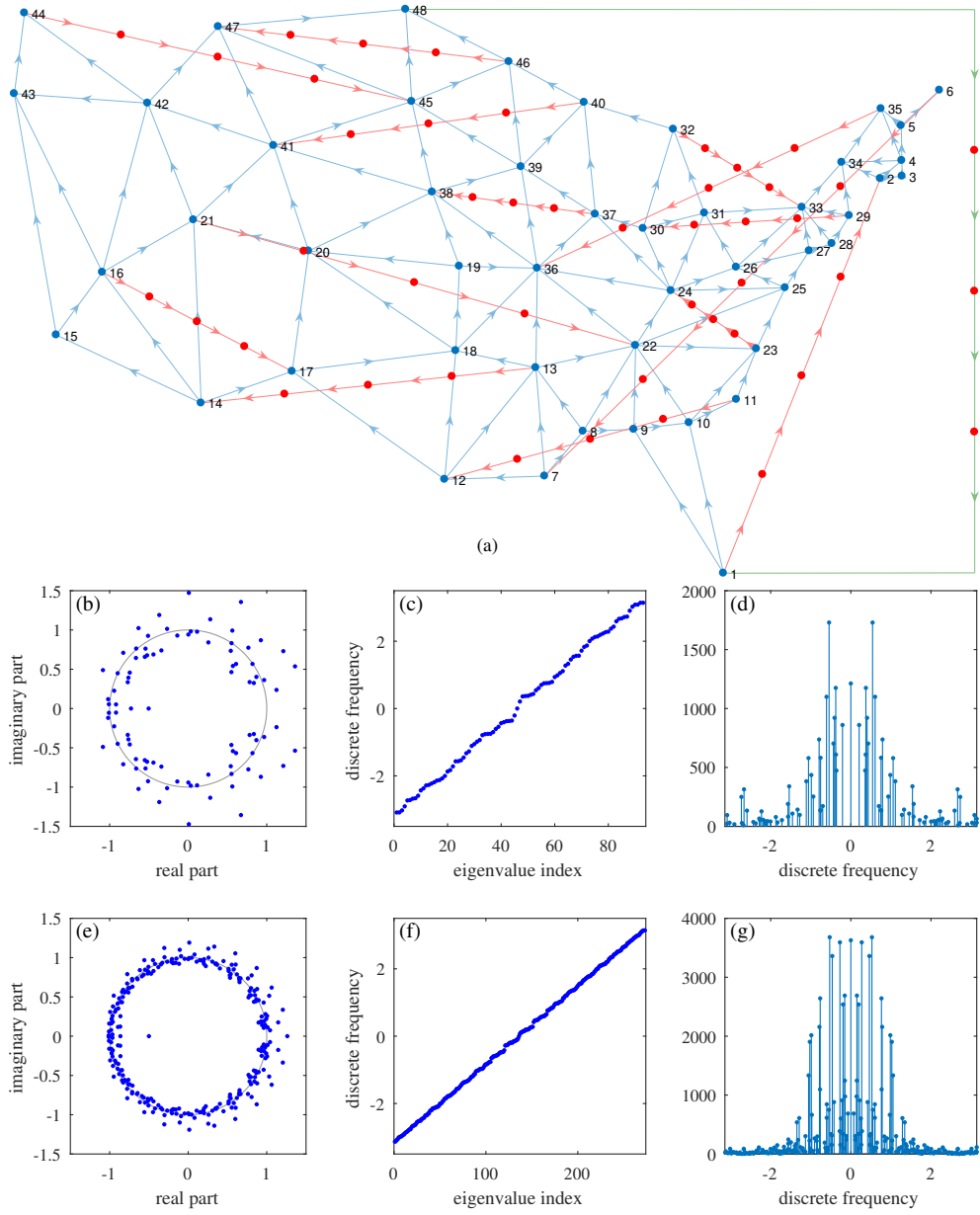


Figure 13: (a) The USA temperature map [17, 22, 25] zero-padded using $M = 3$. Edges and vertices added in the zero-padding procedure are marked in red, while the return path is marked in green. Additionally, scatter plots of imaginary and real part of the eigenvalues (b,e), relation between eigenvalue index and discrete frequency (c,f) and the corresponding GFTs of the temperature signal (d,g) with $M = 3$ (b,c,d) and $M = 15$ (e,f,g) are provided.

1 or -1 , indicating that all eigenvalues are nonzero. For a general form of a DAG, we have presented an algorithm to first ensure the connectivity of the DAG. Following this, we add a backward path to the connected graph.

All changes in the graph structure (including added edges and vertices) are made under the condition that the output signal obtained by a system on a zero-padded graph completely matches the output of the same system on the original DAG. Considering that the number of added vertices can vary, as long as their number is greater than or equal to the order of a system on the graph, we can conclude that diagonalizability can be achieved in practically (almost) all cases of DAGs using the presented approach.

References

- [1] G. Leus, A. G. Marques, J. M. Moura, A. Ortega, D. I. Shuman, Graph signal processing: History, development, impact, and outlook, *IEEE Signal Processing Magazine* 40 (4) (2023) 49–60.
- [2] F. Gama, E. Isufi, G. Leus, A. Ribeiro, Graphs, convolutions, and neural networks: From graph filters to graph neural networks, *IEEE Signal Processing Magazine* 37 (6) (2020) 128–138.
- [3] A. Sandryhaila, J. M. Moura, Discrete signal processing on graphs, *IEEE Transactions on Signal Processing* 61 (7) (2013) 1644–1656.
- [4] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, P. Vandergheynst, The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains, *IEEE Signal Processing Magazine* 30 (3) (2013) 83–98.
- [5] L. Stanković, D. Mandić, M. Daković, M. Brajović, B. Scalzo, S. Li, A. G. Constantinides, et al., Data analytics on graphs part I: Graphs and spectra on graphs, *Foundations and Trends® in Machine Learning* 13 (1) (2020) 1–157.
- [6] L. Stanković, D. Mandić, M. Daković, M. Brajović, B. Scalzo, S. Li, A. G. Constantinides, et al., Data analytics on graphs part II: Signals og graphs, *Foundations and Trends® in Machine Learning* 13 (1) (2020) 158–331.

- [7] L. Stanković, D. Mandić, M. Daković, M. Brajović, B. Scalzo, S. Li, A. G. Constantinides, et al., Data analytics on graphs part III: Machine learning on graphs, from graph topology to applications, *Foundations and Trends® in Machine Learning* 13 (4) (2020) 332–530.
- [8] L. Stanković, D. P. Mandić, M. Daković, I. Kisil, E. Sejdic, A. G. Constantinides, Understanding the basis of graph signal processing via an intuitive example-driven approach [lecture notes], *IEEE Signal Processing Magazine* 36 (6) (2019) 133–145.
- [9] B. Seifert, C. Wendler, M. Püschel, Causal Fourier analysis on directed acyclic graphs and posets, *IEEE Transactions on Signal Processing* 71 (2023) 3805–3820.
- [10] T. Pegolotti, B. Seifert, M. Püschel, Fast Möbius and zeta transforms, *arXiv preprint arXiv:2211.13706* (2022).
- [11] V. Mihal, M. Püschel, Möbius total variation for directed acyclic graphs, in: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2023, pp. 1–5.
- [12] B. Seifert, C. Wendler, M. Püschel, Learning Fourier-sparse functions on DAGs, in: *ICLR2022 Workshop on the Elements of Reasoning: Objects, Structure and Causality*, 2022.
- [13] S. Park, J. Kim, DAG-GCN: Directed acyclic causal graph discovery from real world data using graph convolutional networks, in: *2023 IEEE International Conference on Big Data and Smart Computing (BigComp)*, IEEE, 2023, pp. 318–319.
- [14] S. Chen, A. Sandryhaila, J. M. Moura, J. Kovacevic, Signal denoising on graphs via graph filtering, in: *2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, IEEE, 2014, pp. 872–876.
- [15] L. Stanković, D. P. Mandić, Understanding the basis of graph convolutional neural networks via an intuitive matched filtering approach [lecture notes], *IEEE Signal Processing Magazine* 40 (2) (2023) 155–165.
- [16] R. Shafipour, A. Khodabakhsh, G. Mateos, E. Nikolova, Digraph Fourier transform via spectral dispersion minimization, in: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, pp. 6284–6288.

- [17] R. Shafipour, A. Khodabakhsh, G. Mateos, E. Nikolova, A directed graph Fourier transform with spread frequency components, *IEEE Transactions on Signal Processing* 67 (4) (2018) 946–960.
- [18] J. Domingos, J. M. Moura, Graph Fourier transform: A stable approximation, *IEEE Transactions on Signal Processing* 68 (2020) 4422–4437.
- [19] J. A. Deri, J. M. Moura, Spectral projector-based graph Fourier transforms, *IEEE Journal of Selected Topics in Signal Processing* 11 (6) (2017) 785–795.
- [20] J. A. Deri, J. M. Moura, Agile inexact methods for spectral projector-based graph Fourier transforms, *arXiv preprint arXiv:1701.02851* (2017).
- [21] J. A. Deri, J. M. Moura, New York city taxi analysis with graph signal processing, in: *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, IEEE, 2016, pp. 1275–1279.
- [22] S. Furutani, T. Shibahara, M. Akiyama, K. Hato, M. Aida, Graph signal processing for directed graphs based on the Hermitian Laplacian, in: *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part I*, Springer, 2020, pp. 447–463.
- [23] R. Singh, A. Chakraborty, B. Manoj, Graph Fourier transform based on directed Laplacian, in: *2016 International Conference on Signal Processing and Communications (SPCOM)*, IEEE, 2016, pp. 1–5.
- [24] F. Bauer, Normalized graph Laplacians for directed graphs, *Linear Algebra and its Applications* 436 (11) (2012) 4193–4222.
- [25] B. Seifert, M. Püschel, Digraph signal processing with generalized boundary conditions, *IEEE Transactions on Signal Processing* 69 (2021) 1422–1437.
- [26] L. Stanković, M. Daković, M. Brajović, I. Stanković, A. B. Bardi, Zero-padding on connected directed acyclic graphs for spectral processing, in: *2023 31st Telecommunications Forum (TELFOR)*, IEEE, 2023, pp. 1–4.
- [27] B. P. Lathi, R. A. Green, *Essentials of digital signal processing*, Cambridge University Press, 2014.

- [28] L. Stanković, Digital signal processing with selected topics, CreateSpace, 2015.
- [29] L. Stanković, E. Sejdić, Vertex-frequency analysis of graph signals, Springer, 2019.
- [30] S. Sardellitti, S. Barbarossa, P. Di Lorenzo, On the graph Fourier transform for directed graphs, *IEEE Journal of Selected Topics in Signal Processing* 11 (6) (2017) 796–811.
- [31] H. Sevi, G. Rilling, P. Borgnat, Harmonic analysis on directed graphs and applications: From Fourier analysis to wavelets, *Applied and Computational Harmonic Analysis* 62 (2023) 390–440.