Edge Sampling of Graphs: Graph Signal Processing Approach With Edge Smoothness

Kenta Yanagiya, *Member, IEEE*, Koki Yamada, *Member, IEEE*, Yasuo Katsuhara, Tomoya Takatani, and Yuichi Tanaka, *Senior Member, IEEE*

***Abstract*—Finding important edges in a graph is a crucial problem for various research fields, such as network epidemics, signal processing, machine learning, and sensor networks. In this paper, we tackle the problem based on sampling theory on graphs. We convert the original graph to a *line graph* where its nodes and edges, respectively, represent the original edges and the connections between the edges. We then perform node sampling of the line graph based on the *edge smoothness* assumption: this process selects the most critical edges in the original graph. We present a general framework of edge sampling based on graph sampling theory and reveal a theoretical relationship between the degree of the original graph and the line graph. We also propose an acceleration method for edge sampling in the proposed framework by using the relationship between the two types of Laplacian of the node and edge domains. Experimental results in synthetic and real-world graphs validate the effectiveness of our approach against some alternative edge selection methods.**

***Index Terms*—Graph signal processing, edge sampling, line graph, graph sparsification.**

I. Introduction

GRAPH signal processing (GSP) is a developing field of signal processing [2], [3], [4], [5]. GSP targets *graph signals* whose domain is represented as nodes of a graph. There exist various promising applications of GSP since many real-world signals have irregular underlying structures rather than the standard uniform-interval relationships.

Examples of graph signals include signals on social, brain, transportation, power networks, and point clouds [4], [6], [7], [8], [9], [10], [11], [12], [13]. GSP aims to extend theories and algorithms for standard signal processing like sampling, filtering, restoration, and compression.

Received 27 June 2024; revised 7 July 2025 and 18 November 2025; accepted 19 November 2025. Date of publication 9 December 2025; date of current version 19 December 2025. This work was supported in part by JSPS KAKENHI under Grant 24KJ1577. An earlier version of this paper was presented in part at the ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) [DOI: 10.1109/ICASSP43922.2022.9747724]. The associate editor coordinating the review of this article and approving it for publication was Prof. Chee Wei Tan. *(Corresponding author: Kenta Yanagiya.)*

Kenta Yanagiya and Yuichi Tanaka are with the Graduate School of Engineering, The University of Osaka, Suita 565–0871, Japan (e-mail: k_yanagiya@msp-lab.org; ytanaka@comm.eng.osaka-u.ac.jp).

Koki Yamada is with the Department of Electrical Engineering and Computer Science, Tokyo University of Agriculture and Technology, Koganei 184–8588, Japan (e-mail: k-yamada@go.tuat.ac.jp).

Yasuo Katsuhara and Tomoya Takatani are with Toyota Motor Corporation, Susono 410–1107, Japan (e-mail: yasuo_katsuhara@mail.toyota.co.jp; tomoya_takatani@mail.toyota.co.jp).

Digital Object Identifier 10.1109/TSIPN.2025.3639969

Graph signal sampling is a counterpart of that for standard, i.e., uniform-interval, signals [5]. Standard signals implicitly assume their structure because the sampling period is determined prior to sampling and is usually fixed throughout the sampling process. In contrast, graph signals could have irregular structures, and their corresponding graph frequencies are unevenly distributed. The optimal sampling of graph signals often depends on graphs, especially for noisy cases. Therefore, sampling methods on graphs have been studied extensively [5], [6], [7], [10], [14], [15], [16], [17], [18].

In graph signal sampling theory, the main theoretical target is to build an analog of sampling theory for standard time-domain signals for the graph setting. This includes the definition of bandlimitedness and the perfect recovery condition. A particular difference between standard and graph signal sampling is that regular uniform interval sampling sometimes does not reflect their node-wise relationships in the graph setting. Therefore, we need to consider *sampling set selection* for node-wise sampling [5], [14], [15], [16], [18], [19]. In a practical aspect, the acceleration of graph signal sampling is key because we sometimes encounter thousands and even millions of nodes. In terms of that, various fast sampling set selection methods have also been proposed [10], [14], [20], [21]. The details of graph signal sampling are described in Section II-A.

However, most existing sampling methods on GSP consider the sampling of *nodes* as an analogy of that in standard signal processing. There exists another entity for sampling in GSP: *edges*.

In this paper, we propose *edge sampling* based on graph signal processing techniques. As an analogy to node sampling, edge sampling refers to selecting the most essential edges from a given set of edges in the original graph. A new graph comprises the original nodes and sampled, i.e., selected, edges.

Edge sampling is required in various application fields. For example, in network epidemiology, we need to select the essential edges for preventing disease spreading by removing these edges [22], [23]. This technique is important for policymakers to determine an effective lockdown policy [24]. For network science, we often need to obtain a good abstraction of graphs, i.e., edge sparsification [25], [26], [27], to save computation and storage burden.

In this paper, we view edge sampling from a GSP perspective. In the proposed approach, we assume the *smoothness* of edge weights. The smoothness of edge weights refers to the similarity of the weights of adjacent edges, i.e., edges connected to the

same node. As we will show later, this edge smoothness can be found in various graphs. We utilize *graph frequency* to measure the smoothness. However, the smoothness in GSP usually refers to the *smoothness of the signal* on the nodes. To properly regard edge smoothness as signal smoothness, we convert the original graph into a *line graph* that represents the edge connection relationship of the original graph. We select nodes in the line graph based on a GSP technique. As a result, selecting important nodes in the line graph is regarded as selecting important edges in the original graph. We can use various effective and high-quality node sampling methods of GSP by converting to the line graph [10], [14], [18].

We also propose an acceleration method of edge sampling. Edge sampling often requires matrix multiplication(s) to reconstruct edge weights by filtering similar to the signal sampling counterpart. Since we treat edge weights as graph signals in the edge sampling, the size of the filter matrix is identical to the number of edges: it is generally larger than the number of nodes. As a result, the most calculation-intensive part of the edge sampling process is filtering. We propose an approximation method for the filtering process to avoid large matrix multiplication that does not require explicit line graph conversion.

In the experiments of synthetic and real-world graphs, our proposed edge sampling method outperforms alternative edge selection methods regarding several measures.

The remainder of this paper is organized as follows. Section II reviews the sampling of graph signals and edge sparsification/reduction. In Section III, the framework of the proposed method is described along with a smoothness prior in the edge domain. An acceleration method for edge sampling is proposed in Section IV. Section V presents numerical experiments on synthetic and real data. Finally, conclusions are given in Section VI.

Notation: A graph is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ are the sets of nodes and edges, respectively. The number of nodes is $N = |\mathcal{V}|$ unless otherwise specified. The adjacency matrix of $\mathcal{G}$ is represented as $\mathbf{W}$, where its (i, j)-entry $[\mathbf{W}]_{ij}$ is the edge weight between nodes i and j; $[\mathbf{W}]_{ij} = 0$ for unconnected nodes. In this paper, we consider undirected graphs without self-loops, i.e., $[\mathbf{W}]_{ji} = [\mathbf{W}]_{ij}$ and $[\mathbf{W}]_{ii} = 0$ for all i and j. The degree matrix $\mathbf{D}$ is a diagonal matrix whose ith diagonal element $[\mathbf{D}]_{ii} = d_i := \sum_{j \in \mathcal{N}_i} [\mathbf{W}]_{ij}$ where $\mathcal{N}_i$ is the neighbors of node i. In addition to the (weighted) degree, the number of edges connecting to the node i, i.e., unweighted degree, is defined as $k_i := |\mathcal{N}_i|$.

For an unweighted graph, the incidence matrix $\mathbf{B} \in \mathbb{R}^{N \times |\mathcal{E}|}$ is defined as follows:

$$[\mathbf{B}]_{i\alpha} = \begin{cases} 1 & \text{Edge } \alpha \text{ is incident to node } i, \\ 0 & \text{Otherwise.} \end{cases} \tag{1}$$

In other words, each column in $\mathbf{B}$ represents an edge, and two nonzero elements in each column correspond to the nodes connected by the edge. The incidence matrix for a weighted graph $\bar{\mathbf{B}}$ is similarly defined with replacing 1in (1) by $\sqrt{w_\alpha}$, where w_α is the weight of the edge α. In this paper, in addition to the incidence matrix described above, we also define a directed incidence matrix as follows:

$$[\bar{\mathbf{B}}]_{i\alpha} = \begin{cases} \sqrt{w_\alpha} & \text{Edge } \alpha \text{ is incident to node } i, \\ -\sqrt{w_\alpha} & \text{Edge } \alpha \text{ is incident to node } j, \\ 0 & \text{Otherwise.} \end{cases} \tag{2}$$

Note that $\tilde{\mathbf{B}} = |\bar{\mathbf{B}}|$ where $|\cdot|$ makes each element of the given matrix into the absolute value. Directed incidence matrices can also be defined for undirected graphs by considering pseudo-orientation[1].

Graph Laplacian is defined as $\mathbf{L} := \mathbf{D} - \mathbf{W}$. Note that $\mathbf{L} = \bar{\mathbf{B}}\bar{\mathbf{B}}^\top$. Since $\mathbf{L}$ is a real symmetric matrix, it always has an eigendecomposition $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, where $\mathbf{U} = [\mathbf{u}_0, \dots, \mathbf{u}_{N-1}]$ is an orthonormal matrix containing the eigenvectors $\mathbf{u}_i$, and $\mathbf{\Lambda} = \mathrm{diag}(\lambda_0, \dots, \lambda_{N-1})$ consists of the eigenvalues λ_i. These eigenvalues are assumed to be ordered as $0 = \lambda_0 < \lambda_1 \le \lambda_2 \cdots \le \lambda_{N-1} = \lambda_{\max}$ without loss of generality. We refer to λ_i as the *graph frequency*.

A graph signal $x : \mathcal{V} \to \mathbb{R}$ is a function that assigns a real value to each node. Graph signals can be written as vectors $\mathbf{x} \in \mathbb{R}^N$ whose ith element, $[\mathbf{x}]_i$, represents the signal value at the ith node. The graph Fourier transform (GFT) is defined as $[\hat{\mathbf{x}}]_i = \langle \mathbf{u}_i, \mathbf{x} \rangle = \sum_{j=0}^{N-1} [\mathbf{u}_i]_j [\mathbf{x}]_j$. The GFT projects a graph signal onto the space spanned by its eigenbases, and the graph Fourier coefficient $\hat{\mathbf{x}}_i$ measures the amount of contribution of $\mathbf{x}$ at graph frequency λ_i. Therefore, GFT enables frequency analysis of graph signals.

II. Related Work

In this section, we briefly review existing approaches to sampling theory on graphs and edge sparsification/reduction.

A. Graph Signal Sampling

Sampling of graph signals is a fundamental topic on GSP [5], [14], [15]. Node sampling selects samples on $\mathcal{S} \subset \mathcal{V}$ where $\mathcal{S}$ is called a *sampling set*. Since there is no "regular sampling" in general in the graph setting, the sampling quality depends on $\mathcal{S}$, especially in the noisy case.

Theoretical study and practical algorithms for choosing $\mathcal{S} \subset \mathcal{V}$ have been extensively studied. The review paper [5] describes their fundamental concepts related to sampling in the standard shift-invariant space. Below, we summarize representative graph signal sampling algorithms.

Deterministic selection: Deterministic selection algorithms aims to find the sampling set $\mathcal{S}$ to optimize a predetermined objective function [10], [14], [15], [21], [29], [30], [31]. In general, the sampling set selection is combinatorial and NP-hard [32]. Therefore, many deterministic approaches choose the sampling set in a greedy manner.

One of the advantages of the greedy deterministic algorithm is that we can add or remove the sampled nodes without rerunning the selection algorithm. Furthermore, the sampling set is fixed unless the network structure changes.

[1]The pseudo-orientation is induced by the lexicographic order of incident nodes i and j, i.e., edges are oriented from the node i to the node j $(i < j)$ [28].

Many works find the optimal sampling set by minimizing the reconstruction error in the presence of noise. Based on the optimal design of experiments [33], different optimization criteria can be selected to minimize reconstruction error. For example, the A-optimality criterion minimizes the average error, while the E-optimality criterion minimizes the worst-case error [10], [15]. While early methods are based on bandlimitedness of graph signals, sampling set selection with signal models beyond the bandlimited assumption has also been proposed [5], [17], [19], [31].

Selection algorithms often require eigendecomposition or singular value decomposition. Since those decompositions require a high computational cost, sampling set selection algorithms without those decompositions have been proposed [10], [14], [21], [30], [31]. Some methods avoid the decompositions with polynomial approximation of spectral graph filters or error covariance matrix [10], [14], [30]. In [21], Gerschgorin disk-based sampling set selection is proposed to avoid eigendecomposition.

Random selection: Random selection chooses nodes randomly based on a predetermined/precomputed probability distribution [20], [34], [35]. Once the distribution is determined, the selection itself can be quickly realized in a distributed manner. This is one of the advantages of random selection, in contrast to deterministic selection. In practice, random sampling may provide adequate performance on average, but it often requires more samples than deterministic approaches to achieve equivalent reconstruction performance [20].

Random selection that does not consider the underlying graph allows for quick selection of important nodes; however, the number of nodes sampled tends to be higher than graph-dependent approaches [15], [20], [34]. In graph-dependent sampling, the importance of nodes varies depending on the graph, i.e., it is assumed that important nodes are connected to many other nodes with large edge weights [34], [35].

Note that, for our proposed edge sampling, we can use any of the abovementioned methods according to applications, signal models, and/or computational complexity.

B. Edge Sparsification and Reduction

Removing edges in a graph has been studied in many fields, such as graph theory and network science. There are various methods with different motivations.

In graph theory, the reduction of edges is often called *edge sparsification* [25], [26], [27]. The motivation of edge sparsification is to preserve characteristics of the original graph, e.g., eigenvalue distribution and connectivity, in the modified graph. For example, sparsification is often performed to optimize the following objective function [25], [26].

$$\min_{\mathcal{F}\subset\mathcal{E}} |\mathcal{F}| \text{ s.t. } (1-\epsilon)\mathbf{x}^\top\mathbf{L}_0\mathbf{x} \leq \mathbf{x}^\top\mathbf{L}_1\mathbf{x} \leq (1+\epsilon)\mathbf{x}^\top\mathbf{L}_0\mathbf{x}, \quad (3)$$

where $\mathcal{F}$ is the edge set of sprsified graph, $\mathbf{L}_l$ ($l \in \{0,1\}$) is the graph Laplacian corresponds to the original graph $\mathcal{G}_0$ and sprsified graph $\mathcal{G}_1$, and $\epsilon \in [0,1]$ is a small constant.

Although there exist theoretical guarantees, edge sparsification methods have three major issues in real applications. First, they often have to modify edge weights, i.e., the edge weights in the sparsified graph are changed. Second, the number of removed edges cannot be determined before sparsification. Even under the same parameter(s), the number of removed edges can vary due to randomness in the algorithms. Third, when a random selection method is considered, the importance of selected edges is not ordered. In other words, if further edges are added/removed in a manipulated graph, the importance of previously selected edges is not available, and the whole process of random selection should be performed again. These are not desirable, especially for large graphs.

There have been several methods for reducing edges from a graph in the context of network analysis [25], [26], [27]. While some works utilized spectral properties of graphs [25], [26], random sampling is often used similarly to edge sparsification. Therefore, the same issues as edge sparsification can be found.

In network epidemiology, the reduction of edges is utilized to prevent the spreading of disease by restricting connections between regions, e.g., points-of-interest and cities [22], [23], [24]. Disease could spread along the underlying network, and it is known that the largest eigenvalue of its network topology greatly influences whether or not the infection spreads, as well as the infectiousness and infection rate of diseases [22], [36].

For example, the optimal edge reduction to prevent the spread of infection is considered by solving the following objective function to minimize the largest eigenvalue of the adjacency matrix [34].

$$\min_{\mathcal{F}\subset\mathcal{E}} \tilde{\lambda}_1 \text{ s.t. } |\mathcal{F}| = |\mathcal{E}| - N_r \quad (4)$$

where $\tilde{\lambda}_1$ is the largest eigenvalue of the adjacency matrix corresponding to the sparsified graph $\mathcal{G}_1$, and N_r is the number of removed edges. Although reduction algorithms can efficiently reduce the edges that contribute to the spread of infection, they are often application-specific.

In summary, 1) existing graph signal processing methods only focus on sampling nodes, not edges, and 2) edge sparsification/reduction also faces challenges mainly due to its randomness or ad-hoc designs. In contrast, our method is the first deterministic and generic edge sampling method based on GSP, which can utilize various specifications of sampling set selection on graphs.

III. Edge Sampling Using Graph Sampling Theory

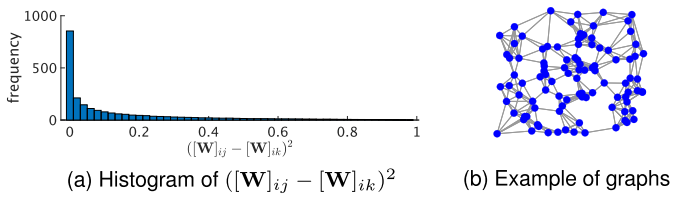
In this section, we describe our proposed edge sampling method. First, the formulation and overview of our method are presented, and then the details of the selection algorithm are discussed.

A. Formulation and Assumption

Suppose that the original graph $\mathcal{G}_0 = (\mathcal{V}, \mathcal{E})$ is given. In general, edge sampling can be represented as the following problem:

$$\text{find } \mathcal{F} \subset \mathcal{E} \text{ s.t. } \min f(\mathcal{G}_1), \quad (5)$$

where $\mathcal{G}_1 := (\mathcal{V}, \mathcal{F})$ and $f(\mathcal{G}_1)$ is some objective function. As mentioned in Section II-A, (5) is generally NP-hard [32]. We thus consider solving (5) approximately with a GSP technique.



(a) Histogram of $([\mathbf{W}]_{ij} - [\mathbf{W}]_{ik})^2$ (b) Example of graphs

Fig. 1. (a) Histogram of the distribution of the difference between adjacent edge weights, i.e., $([\mathbf{W}]_{ij} - [\mathbf{W}]_{ik})^2$, of random sensor networks ($N = 100$). Average of 100 trials. The horizontal and vertical axes represent the difference in the edge weights and their frequency, respectively. (b) An example of a used random sensor network.

In this paper, we consider the following objective function:

$$f(\mathcal{G}_1) := \|\mathbf{w} - \mathrm{Interp}(\mathbf{w}_{\mathcal{F}} + \mathbf{n})\|_2, \tag{6}$$

where $\mathbf{w} \in \mathbb{R}^{|\mathcal{E}|}$ is a vector whose $[\mathbf{w}]_\alpha$ is the edge weight of edge α, $\mathbf{w}_{\mathcal{F}}$ is the subvector which has the elements of $\mathbf{w}$ specified by $\mathcal{F}$, $\mathbf{n} \sim \mathcal{N}(0, \sigma_{\mathbf{n}}^2)$ is i.i.d. white Gaussian noise with the variance $\sigma_{\mathbf{n}}^2$, and $\mathrm{Interp}(\cdot)$ is a (linear) interpolation function corresponding to the chosen edge sampling method. In this paper, we assume that $\mathrm{Interp}(\cdot)$ can achieve the (approximately) perfect reconstruction of the original edge weight $\mathbf{w}$ from sampled ones $\mathbf{w}_{\mathcal{F}}$ in the noise-free case and also yield small errors in the noisy case. This is done by various graph signal interpolation methods [37], [38], [39], [40]. (6) leads to finding a good edge subset so that it well estimates the removed edge weights under the noisy condition. Furthermore, edge weights in a physical measurement are often perturbed or unstable during the sensing process, such as sensor networks and biomedical information processing [3], [28], [41]. Therefore, we consider the robustness against edge weight perturbation.

In this paper, we assume the smoothness of edge weights in $\mathcal{E}$. This can be formulated as follows:

$$\sum_{i \in \mathcal{V}} \sum_{j,k \in \mathcal{N}_i} ([\mathbf{W}]_{ij} - [\mathbf{W}]_{ik})^2 \leq \epsilon_e, \tag{7}$$

where ϵ_e is a small positive constant. The smoothness of edge weights (7) means the variation of the edge weights for adjacent edges is small. We call this *edge smoothness* in this paper.

1) Edge Smoothness for Unweighted Case: Simply, edge smoothness can be verified through unweighted graphs. An unweighted graph has edges whose weights are 1, and therefore, $[\mathbf{W}]_{ij} - [\mathbf{W}]_{ik} = 0$ between all adjacent edges j and k. In the case of unweighted graphs, (7) is always satisfied regardless of the value of ϵ_e.

2) Edge Smoothness for Weighted Case: We then consider edge smoothness in weighted graphs. Here, we assume that ϵ_s is set appropriately. For nodes distributed in space, we often estimate edge weights by (Euclidean) distances between nodes or some affinity (kernel) function. In such metric graphs, we typically observe small edge weight variation over the network since adjacent edges tend to have similar weights. The typical examples are wireless sensor networks and point clouds.

Here, we show a simple example of the edge weight variation between adjacent edges in the random sensor networks. Fig. 1(a) shows the histogram of $([\mathbf{W}]_{ij} - [\mathbf{W}]_{ik})^2$ for random sensor graphs constructed by kNN ($k = 6$) with 100 nodes (Fig. 1(b)), where $[\mathbf{W}]_{ij} = \exp(-\frac{\|\mathbf{p}_i - \mathbf{p}_j\|_2}{0.3})$ is the edge weight determined by the coordinates of nodes $\mathbf{p}_i \in \mathbb{R}^2$. Edge smoothness refers to that $([\mathbf{W}]_{ij} - [\mathbf{W}]_{ik})^2$ is concentrated at the origin. As observed from the figure, the assumption of edge smoothness is reasonable even for weighted graphs whose weights are determined by distances.

Many (but not all) physical networks also have edge smoothness. Representative examples are scale-free networks like 1) airline networks, 2) the Internet, 3) power grids, 4) water distribution systems, and 5) protein-protein interactions. It is well known that the number of edges in a scale-free network follows the power law [42]. In a scale-free network, many nodes only have a small number of connections. It is also well known that a scale-free network has a small number of hubs that are strongly connected (i.e., large edge weights).

The edge weights of physical scale-free networks often reflect the structural properties of edges or nodes connected to each edge, such as the edge centrality and node degree [43], [44], [45], [46]. For example, in water distribution networks, the diameters of water pipes connecting hubs are very large, but the other pipes have similar (but sometimes slightly different) diameters [47]. In this case, edge weight differences in (7) can be small, which leads to edge smoothness. Urban traffic networks and power grids also have similar tendencies because the road-segment capacities or impedances of transmission lines vary slowly across neighboring edges and demand continuity [48], [49]. This is a characteristic of real networks: its theoretical aspects are beyond our scope of this paper and left for future work.

3) Edge Smoothness in Frequency Domain: To introduce graph frequencies for the edge domain, we first define the GFT in the edge domain. By using the right singular vector matrix $\mathbf{V}$ obtained from the singular value decomposition $\bar{\mathbf{B}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$, the GFT of the edge domain is defined as follows:

$$\hat{\mathbf{w}} = \mathbf{V}^\top \mathbf{w}, \tag{8}$$

where $\hat{\mathbf{w}}$ is the graph Fourier coefficients. Note that, since $\bar{\mathbf{B}}\bar{\mathbf{B}}^\top$ is positive semidefinite, the corresponding singular values, denoted as $\sigma_i = \sqrt{\lambda_i}$, are real and non-negative. As in the definition of eigenvalue decomposition, the singular values are ordered in ascending order as $0 \leq \sigma_0 \leq \sigma_1 \leq \sigma_2 \cdots \leq \sigma_{N-1} = \sigma_{\max}$ and the corresponding singular vectors are sorted accordingly.

Similar to smoothness in the node domain, smoothness in the edge domain can be viewed as the energy of $\hat{\mathbf{w}}$ being dominant at the small singular value of $\bar{\mathbf{B}}$ [28], [50]. Therefore, we utilize the established definition of *signal smoothness* on graphs [5], [14], [15] for *edge smoothness* as follows:

$$\mathbf{w}^\top \mathbf{Q} \mathbf{w} = \sum_{\alpha=0}^{|\mathcal{E}|-1} [q(\mathbf{\Lambda}_L)]_\alpha [\hat{\mathbf{w}}]_\alpha^2 \leq \epsilon_s, \tag{9}$$

where $\mathbf{Q} := \mathbf{V} q(\mathbf{\Lambda}_L) \mathbf{V}^\top$ is an arbitrary graph filter with spectral kernel $q(\cdot)$ and ϵ_s is a small positive constant. In Appendix A, we show the mathematical relationship between edge smoothness in the edge domain (7) and in the frequency domain (9).

Based on the above assumptions and observations, if we can *flip* roles of the nodes and edges, we can use a node sampling

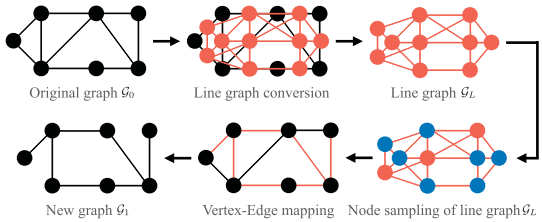


Fig. 2. Overview of the proposed method. The original graph is first converted into a line graph (red nodes correspond to the original edges). The important nodes (blue) are then selected from the line graph: this extracts the most important edges in the original graph.

method to select important edges. In the following, we describe our graph conversion and edge selection method.

B. Framework

The overview of the proposed edge sampling method is illustrated in Fig. 2. In contrast to existing edge sparsification and reduction approaches, we first convert the original graph $\mathcal{G}_0$ into a *line graph* $\mathcal{G}_L$ [51] and then perform sampling set selection on nodes of the line graph. A node in the line graph represents an edge in the original graph, and the edge weight of the line graph indicates the relationship between neighboring edges in the original graph. Therefore, *the node selection of the line graph can be regarded as the edge selection of the original graph*.

Since the nodes in the line graph refer to the original edges, the edge weight of the edge α can be regarded as the signal value on the node α of $\mathcal{G}_L$.

C. Graph Conversion

The original graph and its converted version have a concrete relationship. First of all, we formally define the line graph.

Definition 1 (Line graph): Suppose that the original graph $\mathcal{G}_0 = (\mathcal{V}, \mathcal{E})$ and its incidence matrix $\tilde{\mathbf{B}}$ are given. The adjacency matrix of the line graph $\mathbf{W}_L \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{E}|}$ is defined as follows [51]:

$$\begin{aligned}[\mathbf{W}_L]_{\alpha\beta} &= \sum_i [\tilde{\mathbf{B}}^\top]_{\alpha i}[\tilde{\mathbf{B}}]_{i\beta}(1-\delta_{\alpha\beta}) \\ &= \sum_i [\tilde{\mathbf{B}}^\top]_{\alpha i}[\tilde{\mathbf{B}}]_{i\beta} - 2[\mathbf{C}]_{\alpha\beta}, \end{aligned} \tag{10}$$

where α and β are edge indices of the original graph (and therefore, they are node indices in $\mathcal{G}_L$), $\delta_{\alpha\beta}$ is Kronecker delta, and $\mathbf{C} := diag(\mathbf{w})$ in which $\mathbf{w} \in \mathbb{R}^{|\mathcal{E}|}$ is the edge weight vector sorted by the edge indices. For unweighted graphs, the line graph is also obtained using (10) by replacing $\tilde{\mathbf{B}}$ and $\mathbf{C}$ with $\mathbf{B}$ and the identity matrix $\mathbf{I}_{|\mathcal{E}|}$, respectively.

Fig. 3 shows the illustrative example of the line graph conversion in Definition 1. The graph Laplacian $\mathbf{L}_L$ of the line graph can be introduced as $\mathbf{L}_L = \mathbf{D}_L - \mathbf{W}_L$ where $\mathbf{D}_L$ is the degree matrix of $\mathcal{G}_L$. In addition to the graph Laplacian $\mathbf{L}_L$, we also use *edge Laplacian* as a possible operator for signed line graphs, where directed edge weights can also be considered.

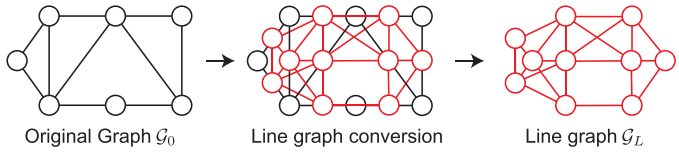


Fig. 3. Example of the line graph conversion. Red nodes in the line graph refer to the edges in the original graph.

Definition 2 (Edge Laplacian): Suppose a directed incidence matrix $\bar{\mathbf{B}}$ in (2) is given. The edge Laplacian is defined as follows [52]:

$$\mathbf{L}_e = \bar{\mathbf{B}}^\top \bar{\mathbf{B}}. \tag{11}$$

Note that the graph Laplacian $\mathbf{L}_L$ and edge Laplacian $\mathbf{L}_e$ are different graph operators but have the same network structure.

Here, we present the relationship between the degrees of the line graph and the original edge weights.

Proposition 1: Suppose that the incidence matrix $\mathbf{B}$, $\tilde{\mathbf{B}}$, or $\bar{\mathbf{B}}$ of the original graph $\mathcal{G}_0 = (\mathcal{V}, \mathcal{E})$ is given, and the adjacency matrix or the edge Laplacian of the line graph is given by (10) or (11). Then, the degree of the node α in the line graph, d_α, that corresponds to the edge α connecting the nodes i and j in the original graph is obtained as follows.

$$d_\alpha = \begin{cases} k_i + k_j - 2 & \text{for unweighted graphs,} \\ \sqrt{w_\alpha}(\tilde{d}_i + \tilde{d}_j) - 2w_\alpha & \text{for weighted graphs,} \\ \sqrt{w_\alpha}(\bar{d}_i - \bar{d}_j) & \text{for directed graphs,} \end{cases} \tag{12}$$

where w_α is the weight of the edge $\alpha = (i, j)$, k_i is the number of edges connecting to the node i, and $\tilde{d}_i$ and $\bar{d}_i$ are the weighted degree of the node i calculated by $\sum_j \sqrt{w_{ij}}$.

The proof of Proposition 1 is shown in Appendix B. Proposition 1 indicates that the high-degree nodes in the line graph correspond to the original edges connecting high-degree nodes. In sampling set selection based on signal smoothness, selected nodes often have high degrees. Hence, selecting important nodes in the line graph can be regarded as selecting important edges in the original graph.

D. Node Sampling of Line Graph

As previously mentioned, we can use an arbitrary sampling set selection for the line graph. The important property of GSP-based sampling set selection methods is that many of them are designed to be robust to noise [5]. This satisfies the requirement of (6).

While any sampling set selection algorithm can be utilized, two issues should be considered according to the applications. The first property is the distribution of the sampled edges. Especially for sensor networks, an edge distribution without increasing communication costs is desirable [53], [54], [55], [56]. Communication costs typically depend on the shortest path length between two nodes and thus, it is preferable to have a small graph diameter (i.e., the maximum shortest path length within the graph) [53]. Random graphs are known to have small

Algorithm 1: Edge Sampling via the line Graph using the FastGSSS.

Input: $\tilde{\mathbf{B}}$, $\mathbf{C}$, $g(\cdot)$, $\mathcal{E}$, $|\mathcal{F}|$;
$\mathbf{W}_L \leftarrow \tilde{\mathbf{B}}^\top\tilde{\mathbf{B}} - 2\mathbf{C}$;
$\mathbf{L}_L \leftarrow \mathbf{D}_L - \mathbf{W}_L$;
$\mathbf{T} \leftarrow \sqrt{|\mathcal{E}|}g(\mathbf{L}_L)$;
$\mathcal{S}_0 \leftarrow \varnothing$;
$\mathbf{c} \leftarrow \mathbf{0}_{|\mathcal{E}|}$;
for $m = 1, 2, \ldots, |\mathcal{F}|$ **do**
  $\eta \leftarrow \frac{1}{|\mathcal{E}|}\mathbf{1}_{|\mathcal{E}|}^\top\mathbf{c}$;
  for $\alpha \in \mathcal{S}_{m-1}^c$ **do**
    $\text{SCORE}[\alpha] \leftarrow \left\langle R(\eta\mathbf{1}_{|\mathcal{E}|} - \mathbf{c}), |\boldsymbol{T}_{g,\alpha}|\right\rangle$;
  end
  $\alpha^* \leftarrow \arg\max_\alpha \text{SCORE}[\alpha]$;
  $\mathcal{S}_m \leftarrow \mathcal{S}_{m-1} \overset{\alpha}{\cup} \alpha^*$;
  $\mathbf{c} \leftarrow \mathbf{c} + |\boldsymbol{T}_{g,\alpha}|$;
end
Output: $\mathcal{S} = \mathcal{S}_m$

Algorithm 2: Accelerated edge sampling via the line graph using the FastGSSS.

Input: $\bar{\mathbf{B}}$, $g(\cdot)$, ρ, $\mathcal{E}$, $|\mathcal{F}|$
$\mathbf{L} \leftarrow \bar{\mathbf{B}}\bar{\mathbf{B}}^\top$;
$g'(x) \leftarrow \frac{g(x)}{\rho + x}$;
$\mathbf{T} \leftarrow \sqrt{|\mathcal{E}|}\bar{\mathbf{B}}g'(\mathbf{L})\bar{\mathbf{B}}^\top$;
$\mathcal{S}_0 \leftarrow \varnothing$;
$\mathbf{c} \leftarrow \mathbf{0}_{|\mathcal{E}|}$;
for $m = 1, 2, \ldots, |\mathcal{F}|$ **do**
  $\eta \leftarrow \frac{1}{|\mathcal{E}|}\mathbf{1}_{|\mathcal{E}|}^\top\mathbf{c}$;
  for $\alpha \in \mathcal{S}_{m-1}^c$ **do**
    $\text{SCORE}[\alpha] \leftarrow \left\langle R(\eta\mathbf{1}_{|\mathcal{E}|} - \mathbf{c}), |\boldsymbol{T}_{g,\alpha}|\right\rangle$;
  end
  $\alpha^* \leftarrow \arg\max_\alpha \text{SCORE}[\alpha]$;
  $\mathcal{S}_m \leftarrow \mathcal{S}_{m-1} \overset{\alpha}{\cup} \alpha^*$;
  $\mathbf{c} \leftarrow \mathbf{c} + |\boldsymbol{T}_{g,\alpha}|$;
end
Output: $\mathcal{S} = \mathcal{S}_m$

diameters [57]. That is, an edge distribution similar to that of a random graph is beneficial for sensor networks.

On the other hand, in applications such as epidemiology, an edge distribution with isolated nodes is preferable [22]. This could be achieved by using a centralized selection algorithm.

The second property is computational complexity. The number of nodes in the line graph is $|\mathcal{E}|$, which is often greater than that of the original graph N. Hence, fast selection methods are preferred, especially for edge sampling. We summarize the proposed edge sampling framework in Algorithm 1.

IV. Accelerated Node Sampling of Line Graph

In this section, we consider graph sparsification as an application of edge sampling. Here, we consider FastGSSS [10] as a fast node sampling method for the proposed method since the sampled nodes are spatially uniform and appropriate for graph sparsification.

FastGSSS does not require the eigendecomposition of graph operators by using a Chebyshev polynomial approximation (CPA) of the filter kernel. However, we still need a high computation cost for edge sampling even if we use CPA because it requires the matrix multiplications of size $|\mathcal{E}| \times |\mathcal{E}|$. In the following, we describe the further acceleration method of FastGSSS for edge sampling.

A. Acceleration by Filtering on Original Graph

FastGSSS selects a node α of $\mathcal{G}_L$ by repeating $|\mathcal{F}|$ iterations of the following equation [10]:

$$\alpha^* = \underset{\alpha\in\mathcal{S}_m^c}{\arg\max}\left\langle R\left(\eta\mathbf{1}_{|\mathcal{E}|} - \sum_{\beta\in\mathcal{S}_m}|\boldsymbol{T}_{g,\beta}|\right), |\boldsymbol{T}_{g,\alpha}|\right\rangle, \quad (13)$$

where $\boldsymbol{T}_{g,\alpha}$ is the αth column of the localization operator $\mathbf{T}$ with the spectral kernel $g(\cdot)$, $\mathcal{S}_m$ and $\mathcal{S}_m^c$ are the selected nodes in the mth iteration and its rest of nodes $\mathcal{E}\backslash\mathcal{S}_m$, η is an aribtrary positive value, and $R(\cdot)$ is the ramp function satisfied with $[R(\mathbf{x})]_\alpha = [\mathbf{x}]_\alpha$ if $[\mathbf{x}]_\alpha \geq 0$ and 0 otherwise.

Let $g(x)$ be the spectral kernel; then, the localization operator $\mathbf{T}$ of the line graph is defined as follows [35]:

$$\mathbf{T} = \sqrt{|\mathcal{E}|}g(\mathbf{L}_e) = \sqrt{|\mathcal{E}|}g(\bar{\mathbf{B}}^\top\bar{\mathbf{B}}) = \sqrt{|\mathcal{E}|}\mathbf{V}g(\boldsymbol{\Lambda}_e)\mathbf{V}^\top, \quad (14)$$

where $\boldsymbol{\Lambda}_e$ and $\mathbf{V}$ are the eigenvalue matrix of $\mathbf{L}_e$ and its corresponding eigenvector matrix (also singular vector matrix of $\bar{\mathbf{B}}$), respectively.

Even when simply using the graph Laplacian $\mathbf{L}_L$ of $\mathcal{G}_L$, fast edge sampling can be achieved by using fast node sampling methods such as FastGSSS that do not require eigenvalue decomposition. Further acceleration of edge sampling is also possible when the edge Laplacian is used in the proposed framework.

The graph Laplacian of the original graph and the edge Laplacian of the line graph have the same nonzero eigenvalues [41]. In other words, the filter response of the $\mathbf{L}$ and the $\mathbf{L}_e$ are identical, and we assume that filtering in the edge domain can be approximated by filtering in the node domain of the original graph. The following method focuses on this relationship and designs $g'(\cdot)$ such that $\bar{\mathbf{B}}^\top g'(\mathbf{L})\bar{\mathbf{B}}$ approximates $g(\mathbf{L}_e)$.

Since the singular value decomposition of $\bar{\mathbf{B}}$ is $\bar{\mathbf{B}} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$, (14) in the graph frequency domain can also be represented as follows:

$$\begin{aligned}\mathbf{T} &= \sqrt{|\mathcal{E}|}\mathbf{V}g(\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma})\mathbf{V}^\top \\ &= \sqrt{|\mathcal{E}|}\mathbf{V}\text{diag}(g(\boldsymbol{\Sigma}_N^\top\boldsymbol{\Sigma}_N), 0)\mathbf{V}^\top \\ &= \sqrt{|\mathcal{E}|}\mathbf{V}_N g(\boldsymbol{\Lambda})\mathbf{V}_N^\top, \end{aligned} \quad (15)$$

where $\boldsymbol{\Sigma}_N$ and $\mathbf{V}_N$ are the singular value matrix with nonzero singular values of $\bar{\mathbf{B}}$ as diagonal elements and the singular vector matrix corresponding to them.

By using the spectral kernel[2] $g'(\boldsymbol{\Lambda}) = (\rho\mathbf{I}_N + \boldsymbol{\Lambda})^{-1}g(\boldsymbol{\Lambda})$ with a small positive constant ρ and the identity matrix $\mathbf{I}_N$, (15)

[2] $\rho\mathbf{I}_N$ is used to ensure invertibility.

TABLE I
COMPUTATIONAL COMPLEXITIES OF EDGE SAMPLING METHODS. PARAMETERS: $|\mathcal{F}|$: NUMBER OF SAMPLED EDGES; p: APPROXIMATION ORDER OF THE CHEBYSHEV POLYNOMIAL APPROXIMATION; J: NUMBER OF NON-ZERO ELEMENTS OF THE LOCALIZATION OPERATOR [35].

	Preparation	Selection
ESLG	$\mathcal{O}(\|\mathcal{E}\|^2 + p\|\mathcal{E}\|\|\mathcal{E}_L\| + J)$	$\mathcal{O}(J\|\mathcal{F}\|)$
A-ESLG	$\mathcal{O}(\|\mathcal{E}\|^2 + pN\|\mathcal{E}\| + J)$	$\mathcal{O}(J\|\mathcal{F}\|)$
ESLG-eig	$\mathcal{O}(\|\mathcal{E}\|^2 + \|\mathcal{E}\|^3 + J)$	$\mathcal{O}(J\|\mathcal{F}\|)$
A-ESLG-eig	$\mathcal{O}(\|\mathcal{E}\|^2 + N^3 + J)$	$\mathcal{O}(J\|\mathcal{F}\|)$
NetMelt [34]	$\mathcal{O}(N + \|\mathcal{E}\|)$	$\mathcal{O}(\|\mathcal{E}\|\|\mathcal{F}\|)$
MaxDegree	$\mathcal{O}(N\|\mathcal{E}\|)$	$\mathcal{O}(\|\mathcal{E}\|\log\|\mathcal{E}\|)$
GSparse [25]	$\mathcal{O}(\log N)$	$\mathcal{O}(\|\mathcal{E}\|)$

can be approximated as follows:

$$\begin{aligned}\mathbf{T} &= \sqrt{|\mathcal{E}|}\mathbf{V}\mathbf{\Sigma}^\top\mathbf{\Sigma}_N^{-1}g(\mathbf{\Lambda})\mathbf{\Sigma}_N^{-1}\mathbf{\Sigma}\mathbf{V}^\top \\ &= \sqrt{|\mathcal{E}|}\mathbf{V}\mathbf{\Sigma}^\top\mathbf{U}^\top\mathbf{U}\mathbf{\Lambda}^{-1}g(\mathbf{\Lambda})\mathbf{U}^\top\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top \\ &\approx \sqrt{|\mathcal{E}|}\mathbf{V}\mathbf{\Sigma}^\top\mathbf{U}^\top\mathbf{U}g'(\mathbf{\Lambda})\mathbf{U}^\top\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top \\ &= \sqrt{|\mathcal{E}|}\bar{\mathbf{B}}^\top g'(\mathbf{L})\bar{\mathbf{B}}. \end{aligned} \tag{16}$$

The approximation of the localization operator (16) reduces computational complexity by replacing the filtering on the line graph with the filtering on the original graph. We summarize the accelerated edge sampling method in Algorithm 2.

B. Computational Complexity

In this subsection, we discuss the computational complexity of the proposed method. Hereafter, the method using FastGSSS in the proposed framework (Section III) and its faster version (Section IV-A) are abbreviated as ESLG (Edge Sampling via the Line Graph with the CPA) and A-ESLG (Accelerated Edge Sampling via the Line Graph with the CPA), respectively. In addition, the proposed methods with the naive eigenvalue decomposition to calculate the localization operator $\mathbf{T}$ exactly are abbreviated as *ESLG-eig* and *A-ESLG-eig*, respectively. Table I shows the computational complexity required for edge sampling, where p is the approximate degree of the CPA, J is the number of nonzero elements of the graph localization operator, $|\mathcal{F}|$ is the number of edges to sample, and $|\mathcal{E}_L|$ is the number of edges in the line graph. Since all the methods require the preparation and selection phases, we separately compared them.

In the framework of the proposed method, the incidence matrix has only two elements in each column; thus, the computation of the graph Laplacian of $\mathcal{G}_L$ requires $\mathcal{O}(|\mathcal{E}|^2)$. Then it takes $\mathcal{O}(p|\mathcal{E}||\mathcal{E}_L|)$ to compute the localization operator. In the case of the faster method in Section IV-A, the computational complexity is $\mathcal{O}(|\mathcal{E}|^2 + pN|\mathcal{E}|)$ because the graph Laplacian of $\mathcal{G}_0$ is used to compute the localization operator of $\mathcal{G}_L$. In general, since $N \ll |\mathcal{E}_L|$, the computational complexity can be reduced. In the proposed framework without the CPA, we require the eigenvalue decomposition to calculate the localization operator in the sampling process. Therefore, the computational complexity of the localization operator in (13) and (16) increase from $\mathcal{O}(p|\mathcal{E}||\mathcal{E}_L|)$ to $\mathcal{O}(|\mathcal{E}|^3)$ and from $\mathcal{O}(|\mathcal{E}|^2 + pN|\mathcal{E}|)$ to $\mathcal{O}(N^3)$, respectively.[3]

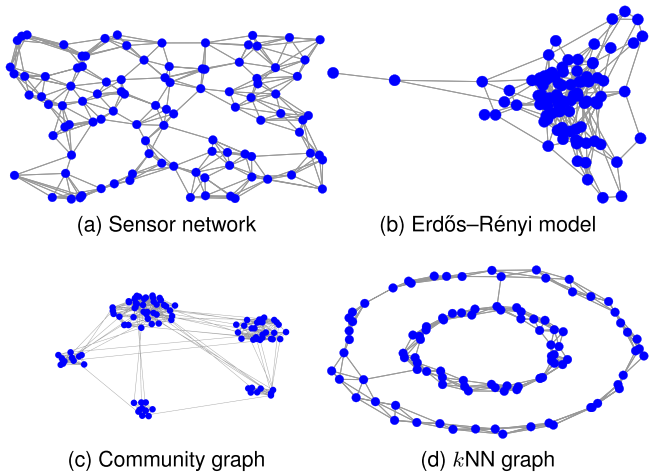


Fig. 4. Examples of graph with $N = 100$.

V. EXPERIMENTS

In this section, we perform the proposed edge sampling method for graph sparsification both for synthetic and real-world data to validate the effectiveness of the proposed approach.

A. Synthetic Data

1) Setup: In this experiment, we use the following weighted graphs with $N = 100$:

- Random sensor network: $|\mathcal{E}| = 360$,
- Random graph based on Erdős–Rényi model with a probability of connection of nodes 0.1: $|\mathcal{E}| = 238$,
- Community graph with 5 communities: $|\mathcal{E}| = 754$,
- kNN graph ($k = 6$) constructed from a point cloud with 2 clusters: $|\mathcal{E}| = 296$.

Figs. 4(a)–(d) show the examples of graphs used in the experiments. Edge weights in the frequency domain are generated with a bandlimited signal model:

$$\hat{\mathbf{w}} = [\hat{\mathbf{w}}_K^\top, \mathbf{0}_{|\mathcal{E}|-K}^\top]^\top + \mathbf{n}, \tag{17}$$

where $\hat{\mathbf{w}}_K$ is a random vector $\mathbb{R}^{K\times 1}$ whose elements are drawn from a normal distribution $\mathcal{N}(0, \sqrt{0.2})$, K is the bandwidth of $\mathbf{w}$, and $\mathbf{n}$ is an i.i.d. additive noise vector with $\mathcal{N}(0, 0.1)$. We set K to $\frac{|\mathcal{E}|}{10}$ in all of the experiments with synthetic data. We then transform the edge weights $\hat{\mathbf{w}}$ into the edge domain by $\mathbf{w} = \mathbf{V}\hat{\mathbf{w}}$, where $\mathbf{V}$ is the eigenvector matrix of the unweighted line graph.

The accuracy of the sparsification is compared in four measures: 1) edge weight reconstruction error, 2) mean squared error (MSE) of diffused random signals, 3) cluster inconsistency of spectral clustering, and 4) execution time.

1) *Edge weight reconstruction error:* We recover the removed edge weights with a graph signal reconstruction method based on the bandlimited assumption. The edge

[3]This is the complexity for dense matrices, i.e., the worst-case scenario [58].

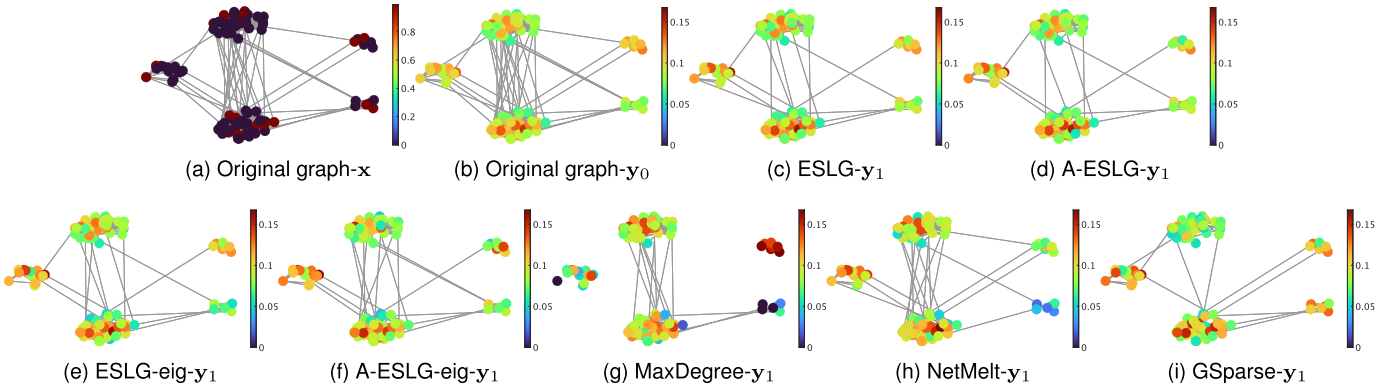


Fig. 5. Graph sparsification and diffusion example: Community graph. Diffused signals are also shown.

weight reconstruction error is given by the objective function (6) of the proposed methods. Let $\mathbf{\Phi}$ be the graph filter on $\mathcal{G}_L$ and be ordered as

$$\mathbf{\Phi} = \begin{bmatrix} \mathbf{\Phi}_{\mathcal{F},\mathcal{F}} & \mathbf{\Phi}_{\mathcal{F},\mathcal{F}^c} \\ \mathbf{\Phi}_{\mathcal{F}^c,\mathcal{F}} & \mathbf{\Phi}_{\mathcal{F}^c,\mathcal{F}^c} \end{bmatrix}, \tag{18}$$

where $\mathbf{\Phi}_{\mathcal{K}_0,\mathcal{K}_1} \in \mathbb{R}^{|\mathcal{K}_0|\times|\mathcal{K}_1|}$ ($\mathcal{K}_k \in \{\mathcal{F},\mathcal{F}^c\}$) is the submatrix of $\mathbf{\Phi}$, whose extracted rows and columns are specified by the sets $\mathcal{K}_0$ and $\mathcal{K}_1$. The following interpolation function is used as Interp$(\cdot)$ in (6) [40]

$$\text{Interp}(\mathbf{w}_{\mathcal{F}}) = \widetilde{\mathbf{\Phi}}\mathbf{\Phi}_{\mathcal{F},\mathcal{F}}^{-1}\mathbf{w}_{\mathcal{F}}, \tag{19}$$

where $\widetilde{\mathbf{\Phi}} = [\mathbf{\Phi}_{\mathcal{F},\mathcal{F}}^\top, \mathbf{\Phi}_{\mathcal{F}^c,\mathcal{F}}^\top]^\top$. In this experiment, we adopt the green function $l(x) = \frac{1}{x+0.005}$ for the spectral kernel of $\mathbf{\Phi}$ [59].

If $\mathcal{F}$ in $\mathcal{G}_1$ is a good abstraction of $\mathcal{E}$ in $\mathcal{G}_0$, the recovered edge weights should be close to the original ones.

2) *MSE of diffused random signals:* The MSE of diffused random signals is computed as follows. Let $\mathbf{L}_0$ be the graph Laplacian of $\mathcal{G}_0$ and $\mathbf{L}_1$ be that of $\mathcal{G}_1$. The diffused signal on $\mathcal{G}_l$ ($l \in \{0,1\}$) is represented as follows:

$$\mathbf{y}_l := h(\mathbf{L}_l)\mathbf{x} = \mathbf{U}_l h(\mathbf{\Lambda}_l)\mathbf{U}_l^\top \mathbf{x}, \tag{20}$$

where $\mathbf{x} \in \{0,1\}^N$ is the input signal, $h(\mathbf{L}_l)$ is the lowpass graph filter on $\mathcal{G}_l$, and $\mathbf{L}_l := \mathbf{U}_l\mathbf{\Lambda}_l\mathbf{U}_l^\top$. We set $h(\lambda) = e^{-t\lambda}$ where $t > 0$ is a parameter. Nonzero elements in $\mathbf{x}$ are randomly chosen, and their number is set to be 20. Finally, the MSE of the diffused signal is calculated by

$$\text{MSE}(\mathbf{y}_0, \mathbf{y}_1) = \frac{1}{N}\|\mathbf{y}_0 - \mathbf{y}_1\|_2^2. \tag{21}$$

If $\mathcal{G}_1$ preserves the original structure, the diffused signal values on $\mathcal{G}_1$ will become similar to those on $\mathcal{G}_0$. This results in a low MSE.

3) *Cluster inconsistency of spectral clustering:* Cluster inconsistency C is represented as follows:

$$C = 1 - \frac{1}{N}\sum_{i=0}^{N-1} s_i, \tag{22}$$

where s_i is the indicator element, in which $s_i = 1$ when the cluster assigned to node i after edge sampling is the same as that of the original graph, and 0 otherwise.

We utilize a spectral clustering method [60] for the experiment. If the graph spectrum after edge sampling maintains the original one, the clusters assigned before and after sampling will coincide. Therefore, C is expected to be small.

4) *Execution time:* We also measure the total execution time required for the preparation and selection phases. In this experiment, the number of the sampled edges $|\mathcal{F}|$ is fixed to $\frac{|\mathcal{E}|}{2}$.

For the proposed methods, we set the reconstruction error shown in (6) as the objective function $f(\mathcal{G}_1)$ in (5). In the proposed edge sampling method, we use FastGSSS [10] as a sampling set selection of the line graph. For the proposed methods, we set the approximation degree of the CPA to 6.

The performance is compared with the following edge sampling methods:

- *MaxDegree* (deterministic): Greedy selection. Edges having the largest $k_i + k_j$ are selected one by one.
- *NetMelt* (deterministic) [34]: Edge selection based on the score calculated from the eigenvectors of $\mathbf{L}_0$.
- *GSparse* (random) [25]: Graph sparsification based on effective resistances, where a random selection of edges is performed based on a probability proportional to the effective resistance of $\mathcal{G}_0$.

The proposed methods and the first two alternative methods are deterministic approaches: the number of edges is specified before edge sampling, and the edges to be removed are fixed as long as the graph is fixed. In contrast, [25] is a random approach that requires a sparsification parameter between $\frac{1}{\sqrt{N}}$ and 1. In other words, the random method cannot determine the number of removed edges. Note that, even under the same parameter, the removed edge positions of *GSparse* are changed in each realization, and their number can vary due to random selection and replacement.

2) Experimental Results: Fig. 5 shows the sparsified graphs by sampling the edges in half, as well as the diffused signals on them. The proposed methods and GSparse are almost connected, while MaxDegree and NetMelt often isolate nodes. It is also

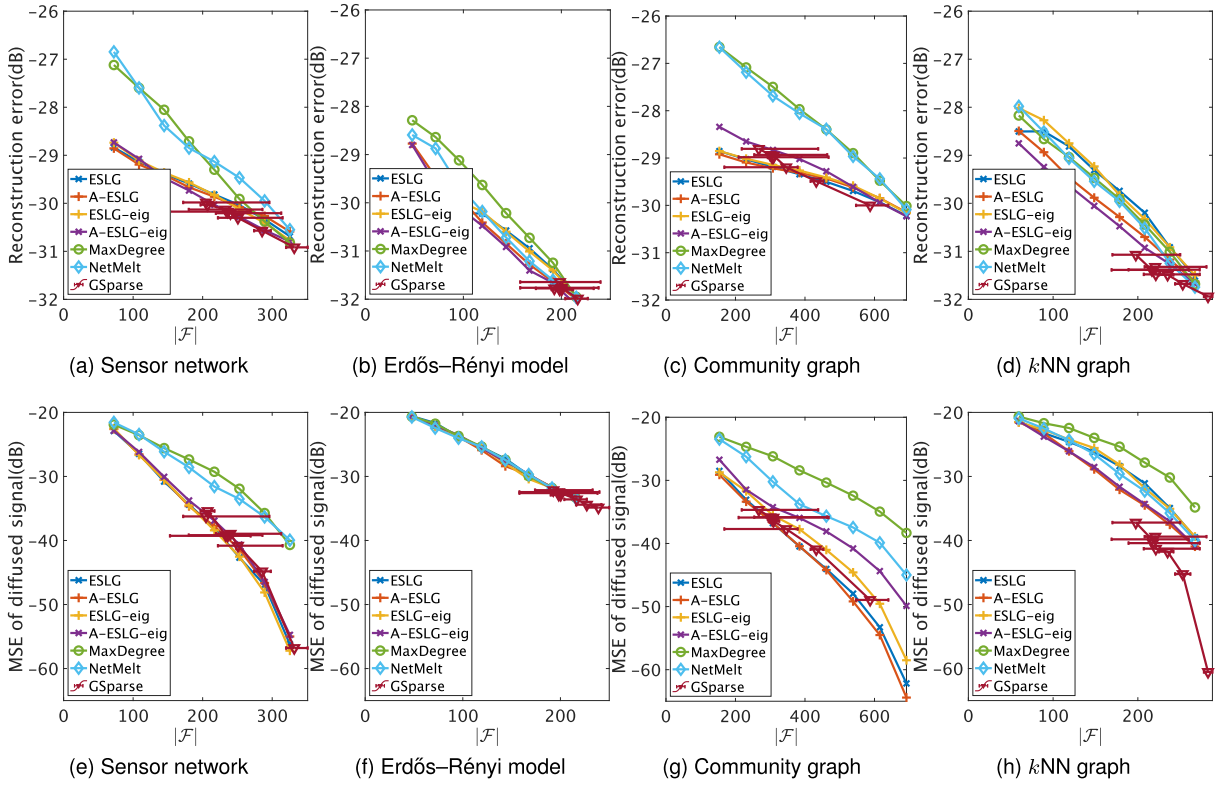


Fig. 6. Comparison of objective performances of sparsified graphs. (a)–(d) Normalized edge weight reconstruction errors. (e)–(h) MSEs of diffused signals in dB. Averaged results after 10 runs are shown. The horizontal lines of GSparse denote the variations of $|\mathcal{F}|$ (i.e., the minimum/maximum number of edges) in the experiment.

observed that the diffused signals of Fig. 5(b)–(d), and (g) are similar to each other.

Fig. 6(a)–(d) show $f(\mathcal{G}_1)$ in (6) as functions of $|\mathcal{F}|$. As previously mentioned, the number of removed edges by GSparse varies even under the same parameters. Therefore, we also illustrate such a variation in the figure. The proposed methods show consistently lower edge weight reconstruction errors than the other methods.

Fig. 6(e)–(h) show the average MSEs of the diffused signal according to $|\mathcal{F}|$ in the sparsified graph. As observed in the sparsified graphs, the proposed methods and GSparse present comparable MSEs and are better than MaxDegree and NetMelt. The proposed methods enable a wide range of edge sparsification factors because they can specify the number of edges, thanks to deterministic sampling. In contrast, GSparse has a small admissible range of $|\mathcal{E}|$. As previously mentioned, the number of removed edges by GSparse significantly varies under the same parameter, where its range sometimes exceeds 200, which is about one-third of $|\mathcal{E}|$.

Fig. 7 compares the cluster inconsistency C for each number of sampled edges $|\mathcal{F}|$. The proposed methods have a lower inconsistency rate due to sampling than the other alternative methods. In other words, the proposed methods select the important edges in the original graph and reduce the change of the graph spectrum due to sparsification. The results by ESLG and the A-ESLG oscillate due to the use of the same parameters regardless of the number of sampled edges $|\mathcal{F}|$.

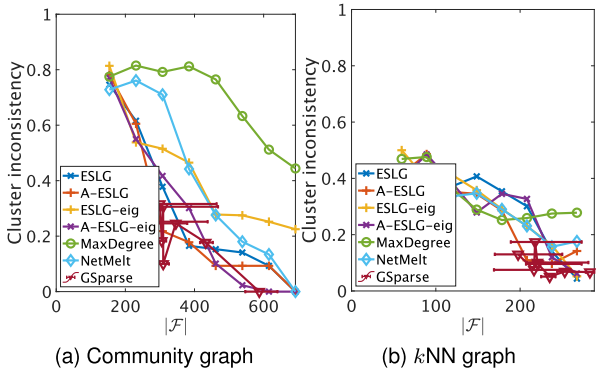


Fig. 7. Comparison of cluster inconsistency of sparsified graphs. Averaged results after 10 runs are shown. The horizontal lines of GSparse denote the variations of $|\mathcal{F}|$ (i.e., the minimum/maximum number of edges) in the experiment.

Table II shows the average execution time of each edge sampling method on 10 independent runs. While MaxDegree and NetMelt are fast because of their heuristic approaches, the accuracies are sacrificed as mentioned previously. The computation times for GSparse and ESLG without the CPA are comparable despite the differences between the deterministic and random methods. A-ESLG shows a consistent acceleration against ESLG due to the approximation of the localization operator by replacing the filtering on $\mathcal{G}_L$ with the filtering on $\mathcal{G}$. This is particularly significant for community graphs. Even if the

TABLE II
AVERAGE EXECUTION TIME ($\times 10^{-2}$ SEC) FOR SYNTHETIC DATA ON 10 INDEPENDENT REALIZATION. EACH DETERMINISITRC SAMPLING METHODS SELECT $\frac{|\mathcal{E}|}{2}$ SAMPLED EDGES. GSPARSE RANDOMLY SELECTS THE SAMPLED EDGES. WE ALSO SHOW THE STANDARD DEVIATION OF THE NUMBER OF SAMPLED EDGES $|\mathcal{F}|$ AND THE EXECUTION TIME OF A 10 REALIZATION.

Graph type	Deterministic							Random	
	$\lvert\mathcal{F}\rvert$	Time ($\times 10^{-2}$)						$\lvert\mathcal{F}\rvert$	Time ($\times 10^{-2}$)
		ESLG	A-ESLG	ESLG-eig	A-ESLG-eig	MaxDegree	NetMelt [34]		GSparse [25]
Sensor network	180 ± 3.84	3.36 ± 0.20	2.86 ± 0.42	9.41 ± 1.29	3.10 ± 0.27	0.01 ± 0.01	0.22 ± 0.05	239 ± 24.5	1.83 ± 0.19
Erdős–Rényi model	119 ± 4.19	2.06 ± 0.21	1.69 ± 0.25	2.08 ± 0.47	1.71 ± 0.52	0.01 ± 0.02	0.47 ± 0.44	200 ± 5.25	2.78 ± 1.31
Community graph	384 ± 28.1	21.1 ± 4.19	14.1 ± 3.82	48.1 ± 9.35	12.9 ± 1.84	0.01 ± 0.00	0.19 ± 0.03	309 ± 12.4	1.62 ± 0.04
kNN graph	148 ± 2.37	2.12 ± 0.11	1.54 ± 0.23	5.80 ± 0.40	1.73 ± 0.07	0.01 ± 0.00	0.18 ± 0.02	215 ± 8.07	2.47 ± 0.24

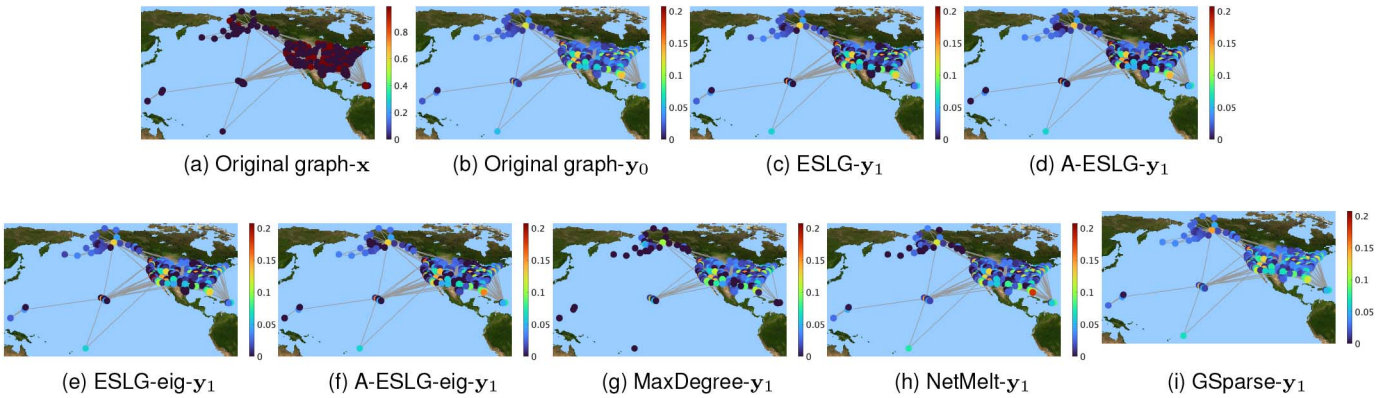


Fig. 8. Graph sparsification and diffusion example: USAir97 dataset. Diffused signals are also shown.

whole graph is sparse, $\mathbf{L}_L$ becomes dense when its subgraph is densely connected [61]. In the dense graph, a large computation burden is required for the matrix product of the graph Laplacian $\mathbf{L}_L$ in CPA. A-ESLG can avoid this computation by using the matrix product of $\mathbf{L}$; thus, the effective acceleration is achieved. Note that, in a few cases, A-ESLG-eig is the fastest among the proposed methods. This is because the naive eigendecomposition for small matrices works well with a sophisticated numerical computation library.

B. Real-World Data

We also perform sparsification experiments through edge sampling with real data.

1) Setup: We use the USAir97 dataset [62] as the actual network. USAir97 is a dataset whose nodes and edges represent airports and the flights between them. The number of nodes and edges is 332 and 2162, respectively. The graph is an undirected weighted graph, where the edge weights represent the number of flights between airports.

In this experiment, we set K in (17) and the number of nonzero elements in the input signal $\mathbf{x}$ in (20) to $\frac{|\mathcal{E}|}{300}$ and $\frac{N}{5}$, respectively. The other parameters were set to the same values as in the experiments with synthetic data. We compare the performance of the proposed method with the same method on synthetic data experiments. As evaluation criteria, we use the edge weight reconstruction error and the MSE of the diffused random signal (see Section V-A).

2) Experimental Results: Fig. 8 shows the visualization of graph sparsification results and diffused signals on these sparsified graphs. As in the experiments on synthetic data, both of the proposed methods can remove half of the number of edges (i.e., 1081 edges) without isolated nodes. Fig. 9(a) and (b) show the edge weight reconstruction error and the MSE of the diffused random signals, respectively. Overall, our proposed methods present satisfactory results in MSEs and flexibility on the number of edges.

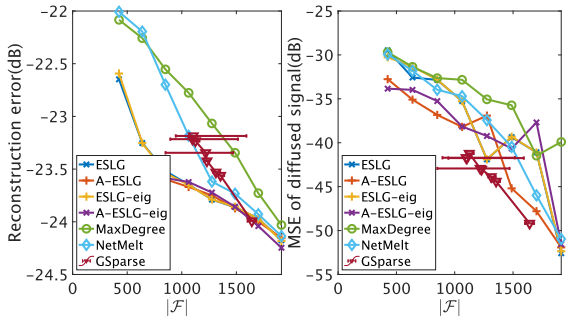


Fig. 9. Comparison of objective performances of sparsified graphs. (a)Normalized edge weight reconstruction errors. (b)MSE of diffused signals in dB. Averaged results after 10 runs are shown. The horizontal lines of GSparse denote the variations of $|\mathcal{F}|$ (i.e., the minimum/maximum number of edges) in the experiment.

VI. CONCLUSION

In this paper, we propose an edge sampling method based on graph sampling theory. We first convert the original graph into a

line graph and perform a sampling set selection of graphs. The edge smoothness characteristic is converted to signal smoothness via graph conversion. In this paper, we also propose a further acceleration method for our edge sampling approach by using a spectral relationship between the graph Laplacian of the line graph and the edge Laplacian. The experimental results on edge sparsification reveal the effectiveness of the proposed method against some alternative approaches.

Appendix A
Relationship Between Edge Domain Smoothness and Graph Frequency Domain Smoothness

Many GSP algorithms assume that a graph signal $\mathbf{x}$ is *smooth* [5], [7], [14]. In Section III-A, we introduced the smoothness of edge weight $\mathbf{w}$ by using the edge-domain assumption (7) and the frequency-domain assumption (9). Here, we provide a relationship between (7) and (9).

Suppose that the original graph $\mathcal{G}$ and its corresponding line graph $\mathcal{G}_L$ are given. Then, the edge smoothness (7) can be rewritten using the Laplacian quadratic form as follows [57]:

$$\mathbf{w}^\top \mathbf{L}_L \mathbf{w} = \hat{\mathbf{w}}^\top \mathbf{\Lambda}_L \hat{\mathbf{w}} = \sum_{\alpha=0}^{|\mathcal{E}|-1} \lambda_\alpha [\hat{\mathbf{w}}]_\alpha^2 \leq \frac{1}{4}\epsilon_e, \tag{23}$$

where $\mathbf{L}_L = \mathbf{U}_L \mathbf{\Lambda}_L \mathbf{U}_L^\top$ and $\hat{\mathbf{w}} = \mathbf{U}_L^\top \mathbf{w}$. The scaling term on the RHS comes from the double count of edge weight differences in (7).

The graph frequency domain smoothness (9) can reduce to the edge domain smoothness (7) if $q(\lambda_\alpha)$ in (9) is chosen appropriately. We show two representative examples below.

Linear high-pass filter: Here, we assume a graph high-pass filter with a linear response $q(\lambda_\alpha) = \lambda_\alpha$. Under this assumption, $\mathbf{Q} = \mathbf{L}_L$: (9) reduces to (23) and $\epsilon_s = \frac{1}{4}\epsilon_e$.

Ideal high-pass filter: The ideal high-pass filter is another example. It is defined as

$$q(\lambda_\alpha) = \begin{cases} \lambda_\alpha & \alpha \geq K, \\ 0 & \alpha < K \end{cases} \tag{24}$$

where K is the cutoff graph frequency. The edge smoothness in the frequency domain (9) in this case can be rewritten as

$$\mathbf{w}^\top \mathbf{Q} \mathbf{w} = \sum_{\alpha=K}^{|\mathcal{E}|-1} \lambda_\alpha [\hat{\mathbf{w}}]_\alpha^2 \leq \epsilon_s. \tag{25}$$

Since $\lambda_\alpha \geq 0$ for all α, the frequency domain smoothness (9) can be upper bounded by edge domain smoothness (23) as follows.

$$\begin{aligned} \sum_{\alpha=K}^{|\mathcal{E}|-1} \lambda_\alpha [\hat{\mathbf{w}}]_\alpha^2 &\leq \sum_{\alpha=0}^{K-1} \lambda_\alpha [\hat{\mathbf{w}}]_\alpha^2 + \sum_{\alpha=K}^{|\mathcal{E}|-1} \lambda_\alpha [\hat{\mathbf{w}}]_\alpha^2 \\ &= \mathbf{w}^\top \mathbf{L}_L \mathbf{w} \leq \frac{1}{4}\epsilon_e. \end{aligned} \tag{26}$$

The inequality in the second line coincides with (9) when $\epsilon_s := \frac{1}{4}\epsilon_e$.

Appendix B
Proof of Proposition 1

In this section, we provide the proof of Proposition 1.

Proof: The degree of the node i of the original graph $\tilde{d}_i$ can be expressed using the incidence matrix $\tilde{\mathbf{B}}$ as follows.

$$\tilde{d}_i = \sum_\alpha [\tilde{\mathbf{B}}]_{i\alpha}, \tag{27}$$

where α is the edge index. Furthermore, the degree d_α of the node α on the line graph is given by

$$d_\alpha = \sum_\beta [\mathbf{W}_L]_{\alpha\beta}. \tag{28}$$

The degree of the node α in (28) can be further rewritten as follows:

$$\begin{aligned} d_\alpha &= \sum_\beta [\mathbf{W}_L]_{\alpha\beta} \\ &= \sum_\beta \left(\sum_i [\tilde{\mathbf{B}}^\top]_{\alpha i} [\tilde{\mathbf{B}}]_{i\beta} - 2[\mathbf{C}]_{\alpha\beta} \right) \\ &= \sum_\beta \sum_i [\tilde{\mathbf{B}}^\top]_{\alpha i} [\tilde{\mathbf{B}}]_{i\beta} - 2\sum_\beta [\mathbf{C}]_{\alpha\beta} \\ &= \sum_i [\tilde{\mathbf{B}}^\top]_{\alpha i} \sum_\beta [\tilde{\mathbf{B}}]_{i\beta} - 2w_\alpha \\ &= \sum_i \tilde{d}_i [\tilde{\mathbf{B}}^\top]_{\alpha i} - 2w_\alpha. \end{aligned} \tag{29}$$

Since i and j are the nodes connected to the edge α, (29) is rewritten with w_α as follows:

$$\begin{aligned} d_\alpha &= \sqrt{w_\alpha}\tilde{d}_i + \sqrt{w_\alpha}\tilde{d}_j - 2w_\alpha \\ &= \sqrt{w_\alpha}(\tilde{d}_i + \tilde{d}_j) - 2w_\alpha. \end{aligned} \tag{30}$$

This is identical to (12) for weighted graphs.

If the original graph $\mathcal{G}_0$ is an unweighted graph without a pseudo-orientation, then $w_\alpha = 1$ and $\tilde{d}_i = k_j$. Hence, the degree of the edge α is $d_\alpha = k_i + k_j - 2$.

When $\mathcal{G}_0$ is a directed graph (or introduces a pseudo-orientation), the line graph is represented by the edge Laplacian. In this case, the line graph is derived from $\bar{\mathbf{B}}^\top \bar{\mathbf{B}}$ instead of $\tilde{\mathbf{B}}^\top \tilde{\mathbf{B}} - 2\mathbf{C}$: the second term in (29) becomes 0.

In addition, as mentioned in the Notation, $\bar{\mathbf{B}}$ has positive and negative elements in each column; thus, the degree of the edge α is $d_\alpha = \sqrt{w_\alpha}(\bar{d}_i - \bar{d}_j)$. This completes the proof. □

References

[1] K. Yanagiya, K. Yamada, Y. Katsuhara, T. Takatani, and Y. Tanaka, "Edge sampling of graphs based on edge smoothness," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2022, pp. 5932–5936.

[2] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.

[3] A. Ortega, P. Frossard, J. Kovačević, J. M. F. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proc. IEEE*, vol. 106, no. 5, pp. 808–828, May 2018.

[4] G. Cheung, E. Magli, Y. Tanaka, and M. Ng, "Graph spectral image processing," *Proc. IEEE*, vol. 106, no. 5, pp. 907–930, May 2018.

[5] Y. Tanaka, Y. C. Eldar, A. Ortega, and G. Cheung, "Sampling signals on graphs: From theory to applications," *IEEE Signal Process. Mag.*, vol. 37, no. 6, pp. 14–30, Nov. 2020.

[6] Y. Tanaka, "Spectral domain sampling of graph signals," *IEEE Trans. Signal Process.*, vol. 66, no. 14, pp. 3752–3767, Jul. 2018.

[7] Y. Tanaka and Y. C. Eldar, "Generalized sampling on graphs with subspace and smoothness priors," *IEEE Trans. Signal Process.*, vol. 68, pp. 2272–2286, 2020.

[8] H. E. Egilmez, E. Pavez, and A. Ortega, "Graph learning from data under Laplacian and structural constraints," *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 6, pp. 825–841, Sep. 2017.

[9] K. Yamada, Y. Tanaka, and A. Ortega, "Time-varying graph learning with constraints on graph temporal variation," 2020, *arXiv:2001.03346*.

[10] A. Sakiyama, Y. Tanaka, T. Tanaka, and A. Ortega, "Eigendecomposition-free sampling set selection for graph signals," *IEEE Trans. Signal Process.*, vol. 67, no. 10, pp. 2679–2692, May 2019.

[11] Y. Tanaka and A. Sakiyama, "M-channel oversampled graph filter banks," *IEEE Trans. Signal Process.*, vol. 62, no. 14, pp. 3578–3590, Jul. 2014.

[12] A. Sakiyama and Y. Tanaka, "Oversampled graph Laplacian matrix for graph filter banks," *IEEE Trans. Signal Process.*, vol. 62, no. 24, pp. 6425–6437, Dec. 2014.

[13] M. Onuki, S. Ono, M. Yamagishi, and Y. Tanaka, "Graph signal denoising via trilateral filter on graph spectral domain," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 2, no. 2, pp. 137–148, Jun. 2016.

[14] A. Anis, A. Gadde, and A. Ortega, "Efficient sampling set selection for bandlimited graph signals using graph spectral proxies," *IEEE Trans. Signal Process.*, vol. 64, no. 14, pp. 3775–3789, Jul. 2016.

[15] S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević, "Discrete signal processing on graphs: Sampling theory," *IEEE Trans. Signal Process.*, vol. 63, no. 24, pp. 6510–6523, Dec. 2015.

[16] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, "Sampling of graph signals with successive local aggregations," *IEEE Trans. Signal Process.*, vol. 64, no. 7, pp. 1832–1843, Apr. 2016.

[17] J. Hara, Y. Tanaka, and Y. C. Eldar, "Generalized graph spectral sampling with stochastic priors," in *Proc. IEEE Int. Conf. Acoust. Speech, Signal Process.*, 2020, pp. 5680–5684.

[18] M. Tsitsvero, S. Barbarossa, and P. Di Lorenzo, "Signals on graphs: Uncertainty principle and sampling," *IEEE Trans. Signal Process.*, vol. 64, no. 18, pp. 4845–4860, Sep. 2016.

[19] J. Hara and Y. Tanaka, "Sampling set selection for graph signals under arbitrary signal priors," in *Proc. 2022 IEEE Int. Conf. Acoust. Speech Signal Process.* 2022, pp. 5732–5736.

[20] G. Puy, N. Tremblay, R. Gribonval, and P. Vandergheynst, "Random sampling of bandlimited signals on graphs," *Appl. Comput. Harmon. Anal.*, vol. 44, no. 2, pp. 446–475, 2018.

[21] Y. Bai, F. Wang, G. Cheung, Y. Nakatsukasa, and W. Gao, "Fast graph sampling set selection using Gershgorin disc alignment," *IEEE Trans. Signal Process.*, vol. 68, pp. 2419–2434, 2020.

[22] C. Nowzari, V. M. Preciado, and G. J. Pappas, "Analysis and control of epidemics: A survey of spreading processes on complex networks," *IEEE Control Syst. Mag.*, vol. 36, no. 1, pp. 26–46, Feb. 2016.

[23] A. Nishi et al., "Network interventions for managing the COVID-19 pandemic and sustaining economy," *Proc. Nat. Acad. Sci.*, vol. 117, no. 48, pp. 30285–30294, Dec. 2020.

[24] S. Chang et al., "Mobility network models of COVID-19 explain inequities and inform reopening," *Nature*, vol. 589, no. 7840, pp. 82–87, Jan. 2021.

[25] D. A. Spielman and N. Srivastava, "Graph sparsification by effective resistances," *SIAM J. Comput.*, vol. 40, no. 6, pp. 1913–1926, Jan. 2011.

[26] D. A. Spielman and S.-H. Teng, "Spectral sparsification of graphs," *SIAM J. Comput.*, vol. 40, no. 4, pp. 981–1025, 2011.

[27] W.-S. Fung, R. Hariharan, N. J. A. Harvey, and D. Panigrahi, "A general framework for graph sparsification," *SIAM J. Comput.*, vol. 48, no. 4, pp. 1196–1223, Jan. 2019.

[28] M. T. Schaub and S. Segarra, "Flow smoothing and denoising: Graph signal processing in the edge-space," in *Proc. 2018 IEEE Glob. Conf. Signal Inf. Process.*, 2018, pp. 735–739.

[29] D. Thanou, D. I. Shuman, and P. Frossard, "Learning parametric dictionaries for signals on graphs," *IEEE Trans. Signal Process.*, vol. 62, no. 15, pp. 3849–3862, Aug. 2014.

[30] F. Wang, G. Cheung, and Y. Wang, "Low-complexity graph sampling with noise and signal reconstruction via neumann series," *IEEE Trans. Signal Process.*, vol. 67, no. 21, pp. 5511–5526, Nov. 2019.

[31] L. Dabush and T. Routtenberg, "Efficient sampling allocation strategies for general graph-filter-based signal recovery," 2025, *arXiv:2502.05583*.

[32] E. Herrendorf, C. Komusiewicz, N. Morawietz, and F. Sommer, "On the complexity of community-aware network sparsification," 2024, *arXiv:2402.15494*.

[33] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2023.

[34] C. Chen, H. Tong, B. A. Prakash, T. Eliassi-Rad, M. Faloutsos, and C. Faloutsos, "Eigen-optimization on large graphs by edge manipulation," *ACM Trans. Knowl. Discov. Data*, vol. 10, no. 4, pp. 1–30, 2016.

[35] N. Perraudin, B. Ricaud, D. I. Shuman, and P. Vandergheynst, "Global and local uncertainty principles for signals on graphs," *APSIPA Trans. Signal Inf. Process.*, vol. 7, 2018, Art. no. e3.

[36] C. W. Tan and P.-D. Yu, "Contagion source detection in epidemic and infodemic outbreaks: Mathematical analysis and network algorithms," *Found. Trends Netw.*, vol. 13, no. 2-3, pp. 107–251, 2023.

[37] S. Chen, A. Sandryhaila, J. M. F. Moura, and J. Kovacevic, "Signal recovery on graphs: Variation minimization," *IEEE Trans. Signal Process.*, vol. 63, no. 17, pp. 4609–4624, Sep. 2015.

[38] S. K. Narang, A. Gadde, E. Sanou, and A. Ortega, "Localized iterative methods for interpolation in graph structured data," in *Proc. 2013 IEEE Glob. Conf. Signal Inf. Process*, 2013, pp. 491–494.

[39] S. K. Narang, A. Gadde, and A. Ortega, "Signal processing techniques for interpolation in graph structured data," in *Proc. 2013 IEEE Int. Conf. Acoust. Speech Signal Process*., 2013, pp. 5445–5449.

[40] I. Pesenson, "Variational splines and paley–wiener spaces on combinatorial graphs," *Constr Approx*, vol. 29, no. 1, pp. 1–21, 2009.

[41] S. Barbarossa and S. Sardellitti, "Topological signal processing over simplicial complexes," *IEEE Trans. Signal Process.*, vol. 68, pp. 2992–3007, 2020.

[42] B. Fotouhi and M. G. Rabbat, "Degree correlation in scale-free graphs," *Eur. Phys. J. B*, vol. 86, no. 12, 2013, Art. no. 510.

[43] D.-H. Kim, J. D. Noh, and H. Jeong, "Scale-free trees: The skeletons of complex networks," *Phys. Rev. E*, vol. 70, no. 4, 2004, Art. no. 046126.

[44] H. Mattie and J.-P. Onnela, "Edge overlap in weighted and directed social networks," *Netw. Sci.*, vol. 9, no. 2, pp. 179–193, 2021.

[45] M. Haque, R. Sarmah, and D. K. Bhattacharyya, "A common neighbor based technique to detect protein complexes in PPI networks," *J. Genet. Eng. Biotechnol.*, vol. 16, no. 1, pp. 227–238, 2018.

[46] M. Á. Serrano, M. Boguñá, and A. Vespignani, "Extracting the multiscale backbone of complex weighted networks," *Proc. Natl. Acad. Sci.*, vol. 106, no. 16, pp. 6483–6488, 2009.

[47] K. Vairavamoorthy and M. Ali, "Optimal design of water distribution systems using genetic algorithms," *Comput. Aided Civil Eng*, vol. 15, no. 5, pp. 374–382, 2000.

[48] L. Elefteriadou, *An Introduction to Traffic Flow Theory*, vol. 84. Berlin, Germany: Springer, 2024.

[49] L. L. Grigsby, *Power Systems*. Boca Raton, FL, USA: CRC press, 2012.

[50] M. T. Schaub, Y. Zhu, J.-B. Seby, T. M. Roddenberry, and S. Segarra, "Signal processing on higher-order networks: Livin' on the edge... and beyond," *Signal Process.*, vol. 187, 2021, Art. no. 108149.

[51] T. S. Evans and R. Lambiotte, "Line graphs of weighted networks for overlapping communities," *Eur. Phys. J. B*, vol. 77, no. 2, pp. 265–272, Sep. 2010.

[52] T. M. Roddenberry, M. T. Schaub, and M. Hajij, "Signal processing on cell complexes," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process*., 2022, pp. 8852–8856.

[53] P. Santi, "Topology control in wireless ad hoc and sensor networks," *ACM Comput. Surv.*, vol. 37, no. 2, pp. 164–194, 2005.

[54] S. Kar and J. M. F. Moura, "Sensor networks with random links: Topology design for distributed consensus," *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 3315–3326, Jul. 2008.

[55] C. Nakas, D. Kandris, and G. Visvardis, "Energy efficient routing in wireless sensor networks: A comprehensive survey," *Algorithms*, vol. 13, no. 3, 2020, Art. no. 72.

[56] Y. Shang, "Fast distributed consensus seeking in large-scale sensor networks via shortcuts," *Int. J. Comput. Sci. Eng*, vol. 7, no. 2, 2012, Art. no. 121.

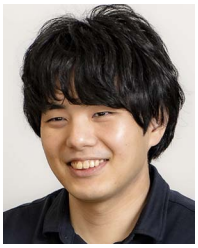
[57] F. R. K. Chung, *Spectral Graph Theory*, vol. 92, Providence, RI, USA: Amer. Math. Soc., 1997.

[58] G. Golub and C. Van Loan, *Matrix Computations*. Baltimore, MD, USA: Johns Hopkins Univ. Press, 2013.

[59] N. Perraudin et al., "GSPBOX: A toolbox for signal processing on graphs," 2016, *arXiv:1408.5781*.

[60] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 14, 2001, pp. 849–856.
[61] C. Godsil and G. Royle, *Algebraic Graph Theory*. Berlin, Germany: Springer, 2001, vol. 207.
[62] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *Proc. AAAI*, 2015, pp. 4292–4293.

Kenta Yanagiya (Member, IEEE) received the B.E. and M.E. degrees in computer and information sciences from the Tokyo University of Agriculture and Technology, Fuchu, Japan, in 2021 and 2023, respectively. He is currently working toward the Ph.D. degree with the Graduate School of The University of Osaka, Suita, Japan. His research interests include graph signal processing and network analysis.



Koki Yamada (Member, IEEE) received the B.S. degree in physics and informatics and the M.S. degree in science and engineering from Yamaguchi University, Yamaguchi, Japan, in 2015 and 2017, respectively, and the Ph.D. degree in engineering from the Tokyo University of Agriculture and Technology, Fuchu, Japan, in 2022. He is currently an Associate Professor with the Department of Electrical Engineering and Computer Science, the Tokyo University of Agriculture and Technology.

Yasuo Katsuhara was born in Kumamoto Prefecture, Japan, in 1989. He received the M.E. degree from Kumamoto University, Kumamoto, Japan, in 2014. In 2014, he joined Toyota Motor Corporation, Susono, Japan. He worked with Research Center, Silicon Valley, CA, USA, in 2017. His research focuses on emotion estimation using biometric data and future prediction using time-series data. His research focuses on analysis of financial data using machine learning.

Tomoya Takatani received the Ph.D. degree from the Nara Institute of Science and Technology, Ikoma, Japan, in 2006, where he conducted research on blind acoustic signal separation using unsupervised learning algorithms. In 2006, he joined Toyota Motor Corporation, where he has been focused on applied research in machine learning and artificial intelligence, with practical implementations in areas, such as speech communication robots—including KIROBO, Pocobee, and the Human Support Robot—brain–machine interfaces, and AI chatbot systems. From 2022 to 2025, he was a Research Coordinator with Toyota Konpon Research Institute, where he launched the "Search/Exploration Program" to explore next-generation interdisciplinary research themes. In 2025, he returned to Toyota Motor Corporation,Susono, Japan, where he continues to advance collaborative research between internal R&D teams and academic institutions. In 2006, he was awarded the Itakura Prize Innovative Young Researcher Award by the Acoustical Society of Japan.

Yuichi Tanaka (Senior Member, IEEE) received the B.E., M.E., and Ph.D. degrees in electrical engineering from Keio University, Yokohama, Japan, in 2003, 2005, and 2007, respectively. From 2006 to 2008, he was a Visiting Scholar with the University of California, San Diego, CA, USA. From 2007 to 2008, he was a Postdoctoral Scholar with Keio University, and was supported by the Japan Society for the Promotion of Science. From 2008 to 2012, he was an Assistant Professor with the Department of Information Science, Utsunomiya University, Tochigi, Japan. From 2012 to 2022, he was an Associate Professor with the Department of Electrical Engineering and Computer Science, Tokyo University of Agriculture and Technology, Fuchu, Japan. Since 2022, he has been a Professor with the Graduate School of Engineering, The University of Osaka, Osaka, Japan. His research interests include the field of high-dimensional signal processing and machine learning which include: graph signal processing, green IoT, geometric deep learning, sensor networks, image/video processing in extreme situations, biomedical signal processing, and remote sensing. He was the recipient of the Best Paper Awards in APSIPA ASC 2014 and 2015 and IEEE Signal Processing Society Japan Best Paper Award in 2016. He was also the recipient of Yasujiro Niwa Outstanding Paper Award in 2010, TELECOM System Technology Award in 2011, Ando Incentive Prize for the Study of Electronics in 2015, and DOCOMO Mobile Science Award in 2025. Since 2022, he was an Associate Editor for IEEE SIGNAL PROCESSING LETTERS and *APSIPA Transactions on Signal and Information Processing*, and also an Associate Editor for IEEE TRANSACTIONS ON SIGNAL PROCESSING from 2016 to 2020, and *IEICE Transactions on Fundamentals* from 2013 to 2017. He is currently the Board-of-Governors Member-at-Large of APSIPA and Vice Chair of APSIPA IVM (Image, Video, and Multimedia) Technical Committee.