GraphLite: Compact Representation of Smooth Graphs Learned under Log-Degree Regularization

Fatemeh Kasraei, Arash Amini, and Stefano Rini

***Abstract*— Graph representations of data offer a rich framework for advanced signal processing applications. However, in many practical scenarios, constructing the graph is computationally demanding, and storing it can be prohibitively expensive– often requiring significantly more memory than the signal itself. This paper introduces *GraphLite*, a novel algorithm tailored for one of the good performing graph learning methods, to compactly represent the learned graph through an auxiliary vector of the same dimension as the signal. This auxiliary representation, which consists of the inverse node degree profile, arises naturally from the structure of the optimal solution and can be pre-computed and stored alongside the signal. The result is a lightweight, lossless graph representation that retains compatibility with core graph signal processing (GSP) operations. GraphLite offers a flexible and memory-efficient tool for downstream tasks in applications where both online graph construction and storage are bottlenecks.**

***Index Terms*—Compact graph representation; Graph summarization; Succinct learned graph.**

Graph signal processing (GSP) has emerged as a powerful framework for analyzing complex data structures that cannot be effectively represented by traditional regular formats [1]. Tools such as the graph Fourier transform, spectral clustering, and graph sampling have made GSP attractive across domains ranging from neuroscience to transportation and genomics. A fundamental challenge in GSP is the construction of the underlying graph, typically by linking signal elements as vertices with weighted edges reflecting pairwise similarity. Desirable properties for these graphs include smoothness, sparsity, and connectivity. However, storing the connectivity among N vertices may require $\mathcal{O}(N^2)$ memory in the worst case, a significant burden in high-dimensional applications. This has motivated a range of sparse graph construction techniques, though sparsity often comes at the expense of other structural goals. In this work, we introduce *GraphLite* – a method to compactly encode graphs learned via log-degree regularization. Rather than storing the full adjacency matrix $\mathbf{W}$, *GraphLite* compresses the graph into a single vector $\mathbf{p}$, representing inverse node degrees, which has the same dimension as the signal $\mathbf{x}$. This enables a lossless reconstruction of the original graph from $\mathbf{x}$ and $\mathbf{p}$ using a lightweight decoder. A conceptual overview of *GraphLite* is shown in Fig. 1. Once a smooth graph is learned from a signal using log-degree regularization, *GraphLite* allows its structure to be stored efficiently alongside the signal. This makes *GraphLite* particularly useful in scenarios where dense graphs are desirable but full storage or real-time recomputation is infeasible.

Related Work: We first review the results in GSP which are most relevant to the development of the paper. The literature on graph processing and compression has expanded to encompass a variety of thematic areas, each demonstrating the field's adaptability and depth. Graph-based methods have been employed to compress data, resulting in structures that encapsulate minimal yet sufficient information, including the graph structure itself. For example, [2], [3] focus on creating graphs with particular structures. Similarly, studies like [4], [5] delve into graph learning methods that are custom-made for image compression, maintaining a constant graph structure to facilitate the transmission of minimal information. Significant effort has been made to compress graphs derived from specific domains and data sources, such as web graphs [6] and social graphs [7]. The study [8] investigates succinct representations of separable, undirected, and unlabeled graphs, and [9] examines space-efficient representations of boolean graphs without specific combinatorial properties, focusing solely on their density. Many notable researchers have utilized supernodes and superedges to achieve a compact representation of graphs. A common approach employed by most of them is greedy merging. For example, [10] proposed an algorithm assuming a uniform reconstruction scheme. Meanwhile, [11], [12] focused on preserving degrees even when the degree distribution is skewed. Additionally, methods such as [13], [14] employ correction edge sets to enable lossless graph reconstruction. For a comprehensive understanding of lossless graph compression and efficient graph representation, readers are referred to [15]–[17], which provide an extensive overview of the field. Additionally, for those interested in neural-network-based graph summarization techniques, [18] is recommended, although it is beyond the scope of this article.

Notation: Uppercase bold letters represent matrices, while lowercase bold letters denote column vectors. The element at the i-th row and j-th column of matrix $\mathbf{W}$ is denoted by $\mathbf{W}_{ij}$. Similarly, $\mathbf{x}_i$ represents the i-th element of the vector $\mathbf{x}$. $\mathbf{0}$ is the all-zero vector. $\mathbf{I}_N \in \mathbb{R}^{N\times N}$ is the identity matrix. $\mathbf{J}_N \in \mathbb{R}^{N\times N}$ and $\mathbf{J}_{M\times N} \in \mathbb{R}^{M\times N}$ are matrices filled with all ones. The symbol $\|\mathbf{w}\|_2$ denotes the Euclidean norm of vector $\mathbf{w}$, and $|\mathcal{V}|$ denotes the cardinality of the set $\mathcal{V}$. Finally, $[N]$ represents the set $\{1, 2, \ldots, N\}$.

I. Preliminaries

Consider a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with nodes $\mathcal{V}$ and edges $\mathcal{E}$. Let the adjacency matrix be indicated as $\mathbf{W} \in \mathbb{R}^{N\times N}$ where N is the number of nodes, i.e. $|\mathcal{V}| = N$. We consider undirected

F. Kasraei and A. Amini are with the EE department at Sharif University of Technology, Tehran, Iran – email kasraei.fatemeh@ee.sharif.edu and aamini@sharif.edu , respectively. S. Rini is with the ECE department at the National Yang-Ming Chao-Tung University (NYCU), Hsinchu, Taiwan – email: stefano.rini@nycu.edu.tw.

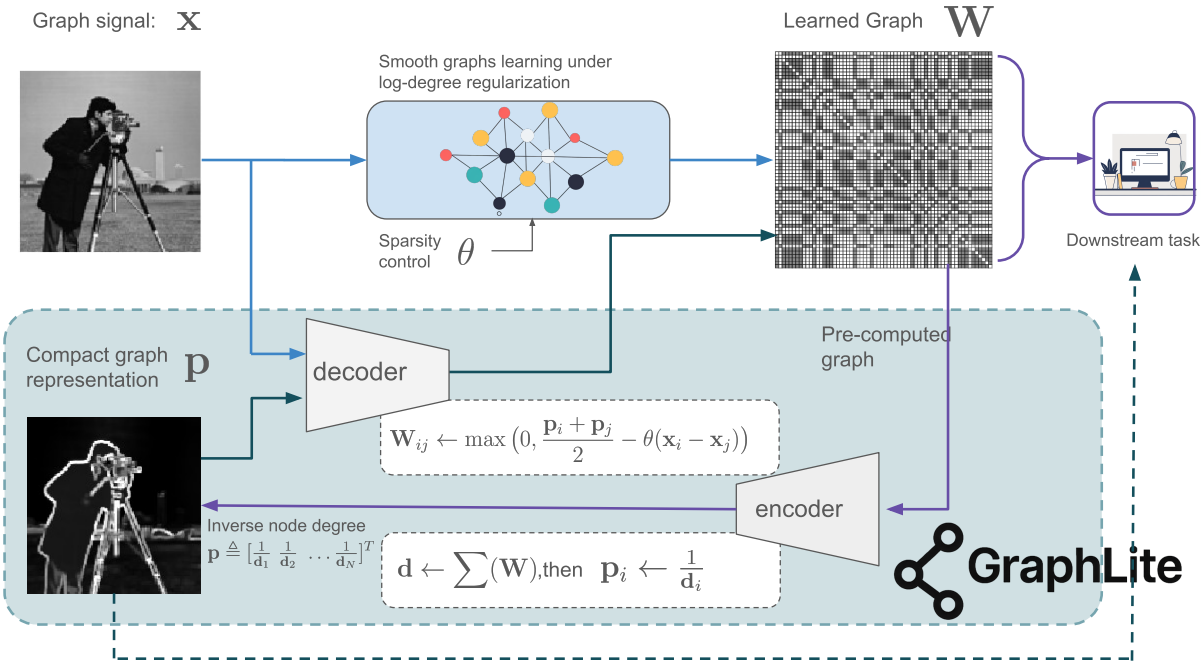

Fig. 1: Overview of the *GraphLite* framework. Given a graph signal $\mathbf{x}$, a smooth graph $\mathbf{W}$ is learned using log-degree regularization. Instead of storing the full adjacency matrix $\mathbf{W}$, *GraphLite* encodes the graph into a compact representation $\mathbf{p}$ consisting of inverse node degrees. This vector, which has the same dimension as $\mathbf{x}$, can be stored alongside the signal and later used to losslessly reconstruct $\mathbf{W}$ via a decoder.

graphs without self-loops: accordingly, the weight (adjacency) matrix is symmetric $\mathbf{W} = \mathbf{W}^T$ and $\mathrm{Diag}(\mathbf{W}) = \mathbf{0}$. The space of all valid weight matrices is defined as

$$\mathcal{W}_v = \left\{\mathbf{W} \in \mathbb{R}_+^{N\times N} : \mathbf{W} = \mathbf{W}^T,\ \mathrm{Diag}(\mathbf{W}) = 0\right\}.$$

For convenience of notation, we define the vectorization operation $\mathbf{w} = \mathrm{vect}(\mathbf{W})$ as in the left-hand side of Fig. 2: $\mathbf{w}$ is obtained by consolidating all upper triangular elements of $\mathbf{W}$. This operation forms a simplified space $\mathcal{V}_v$. This space, defined as $\mathcal{V}_v = \left\{\mathbf{w} \in \mathbb{R}_+^{\frac{N(N-1)}{2}}\right\}$, provides a more concise representation than $\mathcal{W}_v$. The degree of the node $v_i \in \mathcal{V}$ is defined as $\mathbf{d}_i = \sum_j \mathbf{W}_{ij}$. The degree matrix $\mathbf{D}$, is the diagonal matrix with $\mathbf{D}_{ii} = \mathbf{d}_i$. Let $\mathbf{d}$ be the degree vector: By construction of the matrix $\mathbf{A}$ in Fig. 2, it follows that $\mathbf{W} = \mathbf{A}\mathbf{w}$. A graph signal is represented by $\mathbf{x} \in \mathbb{R}^N$ [1]. The matrix $\mathbf{Z} \in \mathbb{R}^{N\times N}$ captures the pairwise distances between graph signal of the nodes. Define $\mathbf{z} = \mathrm{vect}(\mathbf{Z})$.

II. Problem Formulation

The graph learning problem involves determining an adjacency matrix that captures the distances between signal elements while maintaining desirable properties such as smoothness, sparsity and connectivity. In particular, smoothness – which measures the regularity of a graph signal over the graph– plays a fundamental role in various GSP theorems and algorithms. Smoothness is often approximated by the decreasing rate of graph Fourier transform coefficients [4] and is quantified using the Dirichlet energy measure:

$$\frac{1}{2}\sum_{i,j}\mathbf{W}_{ij}(\mathbf{x}_i - \mathbf{x}_j)^2 = \frac{1}{2}\sum_{i,j}\mathbf{W}_{ij}\mathbf{Z}_{ij} = \mathbf{x}^T\mathbf{L}\mathbf{x}. \quad (1)$$

[1]The extension to a vector graph signal is not considered here for brevity.

An algorithm to learn smooth graphs is proposed in [19] by formulating the optimization problem

$$\mathbf{W}_{\mathrm{opt}} = \underset{\mathbf{W}\in\mathcal{W}_v}{\mathrm{argmin}} \underbrace{\sum_{i,j}\mathbf{W}_{ij}\mathbf{Z}_{ij} + f(\mathbf{W})}_{J(\mathbf{w})}, \quad (2)$$

where $f(\mathbf{W})$ is a regularization term that varies according to the application. By choosing an appropriate $f(\mathbf{W})$, constraints such as sparsity can be applied to the graph structure [2]. An effective choice for $f(\mathbf{W})$ is proposed in [19], allowing control over the sparsity rate while avoiding isolated nodes as

$$J(\mathbf{w}) = 2\theta\mathbf{z}^T\mathbf{w} - \sum_i \log(\mathbf{A}\mathbf{w})_i + \|\mathbf{w}\|_2^2. \quad (3)$$

Minimizing $J(\mathbf{w})$ results in a graph with several desirable properties: (i) $\mathbf{z}^T\mathbf{w}$ governs smoothness, (ii) the log barrier term controls node connectivity, (iii) the L_2 norm regulates weight magnitude to induce density/sparsity, (iv) adjusting the parameter θ fine-tunes the amount of sparsity, and (v) the overall cost function is convex.

Our proposed algorithm, GraphLite, represents the optimal solutions of (3) in a way that is both computationally and memory efficient.

III. The GraphLite Algorith

Our main theoretical result consists of expressing the minimizer of (3) in a different form resembling a closed-form

[2]Sparsity here refers to the number of edges in the graph relative to the number of edges in a complete graph with the same number of vertices, defined as $\rho = \frac{E}{N(N-1)}$.

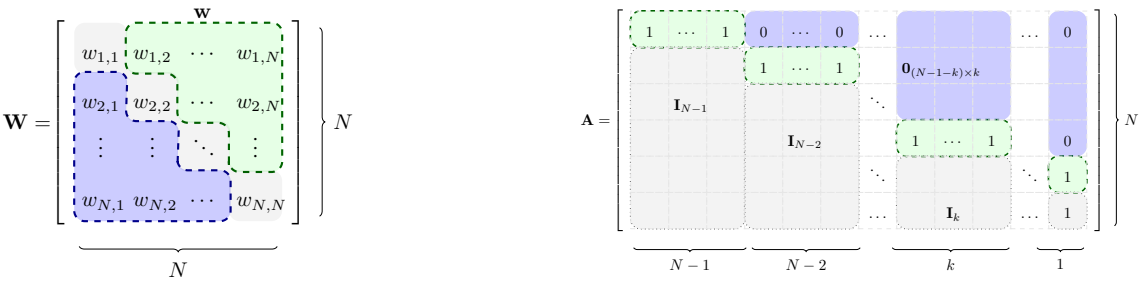

Fig. 2: A graphical representation of the matrices $\mathbf{W}$ and $\mathbf{A}$ in Sec. I. If $\mathbf{w}$ represents the upper-triangular part of $\mathbf{W}$ in form of a vector (scanning row by row), $\mathbf{A}\mathbf{w}$ represents the degree vector $\mathbf{d}$.

solution, and showing that the latter form provides a compact representation for the graph.

Theorem 1: For the minimization problem in (3), the optimal solution satisfies

$$\mathbf{w}_{\text{opt}} = \tfrac{1}{2}\max\big(\mathbf{0}\,,\, \mathbf{A}^T\mathbf{p}_{\text{opt}} - 2\theta\mathbf{z}\big), \tag{4}$$

where the max of two vectors is interpreted element-wise, $\mathbf{p} \triangleq [\frac{1}{\mathbf{d}_1}\ \frac{1}{\mathbf{d}_2}\ \cdots \frac{1}{\mathbf{d}_N}]^T$, and $\mathbf{A}$ is defined in Fig. 2.

Proof: As the cost function $J(\mathbf{w})$ in (3) is convex and the constraint $\mathbf{w} \in \mathbb{R}_+^{\frac{N(N-1)}{2}}$ limits the minizer to a convex half space, we conclude that the optimal solution exists. So, by levering the Karush–Kuhn–Tucker (KKT) optimality condition, we can conclude that for all $i \in [N(N-1)/2]$ it holds that

$$\big(\nabla_{\mathbf{w}} J(\mathbf{w})\big|_{\mathbf{w}_{\text{opt}}}\big)_i = \begin{cases} 0, & \mathbf{w}_{\text{opt},i} > 0, \\ \geq 0 & \mathbf{w}_{\text{opt},i} = 0. \end{cases} \tag{5}$$

Eq. (5), in turn, implies that

$$\min\Big(\nabla_{\mathbf{w}} J(\mathbf{w})\big|_{\mathbf{w}_{\text{opt}}}\ ,\ 2\mathbf{w}_{\text{opt}}\Big) = \mathbf{0}. \tag{6}$$

Let $\{\mathbf{b}_1, \ldots, \mathbf{b}_{\frac{N(N-3)}{2}}\} \in \mathbb{R}^{\frac{N(N-1)}{2}}$ be any orthonormal basis for $\text{null}(\mathbf{A})$, and define $\mathbf{B} \triangleq [\mathbf{b}_1, \ldots, \mathbf{b}_{\frac{N(N-3)}{2}}]^T$. As the sum of rows of $\mathbf{A}$ is $[2, \ldots, 2]$, we conclude that the sum of entries in each $\mathbf{b}_i$ is zero. To simplify the following notations, we introduce $\mathbf{u} \triangleq \mathbf{C}\mathbf{w}$, where $\mathbf{C} \triangleq \big[\mathbf{A}^T, \sqrt{\eta}\,\mathbf{B}^T\big]^T$ is invertible:

$$\mathbf{C}^{-1} = \tfrac{1}{\eta}\big(\mathbf{I}_M - \tfrac{\mu^2}{4}\mathbf{J}_M\big)\mathbf{C}^T. \tag{7}$$

Further, let $\eta = N-2$, $\mu = \frac{2}{N-1}$ and $M = \frac{N(N-1)}{2}$. It is noteworthy that the initial N elements of the vector $\mathbf{u}$ coincide with the values represented by the vector $\mathbf{d}$. Now, we can rewrite (3) as follows:

$$\begin{aligned} J(\mathbf{w}) &= 2\theta\mathbf{z}^T\mathbf{w} - \sum_{i\in[N]} \log(\mathbf{A}\mathbf{w})_i + \|\mathbf{w}\|_2^2 \\ &= 2\theta\mathbf{z}^T\mathbf{w} - \sum_{i\in[N]} \log(\mathbf{A}\mathbf{w})_i + \tfrac{1}{\eta}\big(\|\mathbf{C}\mathbf{w}\|_2^2 - \mu\mathbf{w}^T\mathbf{J}_M\mathbf{w}\big) \\ &= 2\theta\mathbf{z}^T\mathbf{C}^{-1}\mathbf{u} - \sum_{i\in[N]} \log\mathbf{u}_i + \tfrac{1}{\eta}\big(\|\mathbf{u}\|_2^2 - \tfrac{\mu}{4M}\|\mathbf{J}_{M\times N}\mathbf{u}_{[N]}\|_2^2\big) \\ &= \tilde{J}(\mathbf{u}). \end{aligned} \tag{8}$$

Since $\mathbf{w}$ and $\mathbf{u}$ are interchangable, we can express $\nabla_{\mathbf{w}} J(\mathbf{w})$ in terms of $\nabla_{\mathbf{u}} \tilde{J}(\mathbf{u})$:

$$\begin{aligned} \nabla_{\mathbf{w}} J(\mathbf{w}) &= \mathbf{C}^T\nabla_{\mathbf{u}}\tilde{J}(\mathbf{u}) \\ &= 2\theta\mathbf{z} + \mathbf{C}^T\Big(\begin{bmatrix}-\mathbf{p} \\ \mathbf{0}_{\frac{N(N-2)}{2}}\end{bmatrix} + \tfrac{2}{\eta}\mathbf{u} - \tfrac{\mu}{2\eta}\begin{bmatrix}\mathbf{J}_N\mathbf{u}_{[N]} \\ \mathbf{0}_{\frac{N(N-2)}{2}}\end{bmatrix}\Big) \\ &= 2\theta\mathbf{z} - \mathbf{A}^T\mathbf{p} + \tfrac{2}{\eta}\mathbf{C}^T\mathbf{u} - \tfrac{\mu}{\eta}\mathbf{J}_{M\times N}\mathbf{u}_{[N]}. \end{aligned} \tag{9}$$

Besides, $\mathbf{w} = \mathbf{C}^{-1}\mathbf{u} = \frac{1}{\eta}\mathbf{C}^T\mathbf{u} - \frac{\mu}{2\eta}\mathbf{J}_{M\times N}\mathbf{u}_{[N]}$. This reveals that

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = 2\theta\mathbf{z} - \mathbf{A}^T\mathbf{p} + 2\mathbf{w}. \tag{10}$$

Now, if we recall (6) and plug-in the right-hand-side of (10) instead of $\nabla_{\mathbf{w}} J(\mathbf{w})\big|_{w_{\text{opt}}}$, the claim is achieved. ∎

Theorem 1 provides a simple and compact method for encoding the optimal learned graph $\mathbf{w}$ (or $\mathbf{W}$), after it is obtained. As we explain below, the vector $\mathbf{p}_{\text{opt}}$ plays the role of this compact representation. Note that the vector $\mathbf{p}_{\text{opt}}$ is of size N, i.e., the same as the length of graph signal, while the number of entries in $\mathbf{w}$ or $\mathbf{W}$ scales quadratically with N. According to (4), for each i, j, the optimal weight of the edge between the ith and jth node is equal to

$$\mathbf{W}_{\mathbf{opt},ij} = \mathbf{W}_{\mathbf{opt},ji} = \max\big(0\,,\, \tfrac{\mathbf{p}_{\text{opt},i}+\mathbf{p}_{\text{opt},j}}{2} - \theta\mathbf{Z}_{ij}\big). \tag{11}$$

This shows that if the average of the reciprocal degrees of the nodes i and j is less than $\theta\mathbf{Z}_{ij}$ (a multiple of their distance), then, there shall be no edge between the two in the graph; otherwise, the difference of these two values determines the weight of the connecting edge. The *GraphLite* encoding and decoding procedures are illustrated in Fig. 1. By incorporating both the signal data and $\mathbf{p}_{\text{opt}}$, the graph can be stored in a compact and efficient form without loss of information. Based on (11), the *GraphLite* is also well-suited for local processing tasks, such as neighborhood-based filtering or localized graph signal analysis, where only part of the adjacency structure is needed. By allowing direct access to compact substructures, it improves both memory use and computational efficiency in these applications.

IV. Numerical Evaluations

The practical significance of the *GraphLite* is best illustrated through its application in image processing, as demonstrated in Figs. 1 and 3. We start by constructing the distance matrix across pixels based on the ℓ_2 distance between each pixel and its eight neighbors within a 3×3 window. This matrix,

TABLE I: Relative output size and computation time. Each entry is formatted as *Size / Time*(s).

Dataset	**Sparsity**	**Methods**		
		DPGS	**SDSumm**	***GraphLite***
Indian Pines [20]	1.5%	3.55×10^{-2}/3.78	1.25×10^{-2}/1078.18	9.5×10^{-3}/3.0×10^{-2}
	50%	2.5×10^{-3}/67.60	4.8×10^{-4}/1084.94	9.5×10^{-3}/ 4.9×10^{-2}
IDEAL Household [21]	1.5%	6.08×10^{-1}/8.7×10^{-3}	6.7×10^{-1}/3.0×10^{-3}	3.9×10^{-3}/4.7×10^{-4}
	50%	8.76×10^{-2}/13.8×10^{-3}	7.1×10^{-2}/3.1×10^{-3}	3.9×10^{-3}/5.9×10^{-4}

(a) Original image

(b) Compact graph representation

Fig. 3: An example of the the *GraphLite* algorithm.

along with the parameter θ, serves as input for learning the graph $\mathcal{G}$. We optimized the cost function in (3) using GSPBox [22], specifically the `gsp_learn_graph_log_degrees` function. Notably, we fine-tuned the iteration stop criteria to prioritize solution proximity over the number of iterations, resulting in increased accuracy. We should highlight again that our contribution is to compact this graph and not learning the graph.

Illustration: The learned graph $\mathcal{G}$ is characterized by its weight matrix. As illustrated in Figs. 1, it can also be represented compactly using the vector $\mathbf{p}_{\text{opt}}$ along with a distance matrix that can be readily computed from the image itself. Figs. 1 and 3 show two grayscale images: a 126×126 pixel Cameraman and a 150×200 pixel autumn scene, respectively, along with their corresponding compact graph representations. As shown, the reshaped $\mathbf{p}_{\text{opt}}$ retains a strong connection with the original image. In the compact graph representation, distinct boundaries such as the separation between the tree line and the sky, as well as between the reeds and the grass, are clearly visible. This underscores the potential of jointly processing the image alongside the compact graph representation, thereby enhancing its overall efficiency.

Comparison with graph summarization techniques: As most graph summarization techniques are lossy, the reconstruction error of the adjacency matrix is potentially the most important metric to check [12]. However, the *GraphLite* is lossless; therefore, we ignore this metric here.

Compared to the original graph size of $\mathcal{O}(N^2)$, the *GraphLite* demonstrates remarkable space efficiency, achieving $\mathcal{O}(N)$. This indicates that regardless of the sparsity level of the adjacency matrix, the relative output size, defined as the ratio of the summarized representation to the original size, is $\frac{2N}{\binom{N}{2}} = \frac{4}{N-1}$, or more generally, $\frac{2(1+K)}{N-1}$ when the graph signal is $\mathbf{X} \in \mathbb{R}^{N \times K}$, i.e., each node is associated with a signal vector of dimension K. significantly reducing the storage requirements compared to the original graph size.

In Fig. 4 and Table I we compare the proposed *GraphLite* with two benchmark methods: degree-preserved graph summarization (DPGS) [11], and spectral characteristic-preserved summarization (SDSumm) [12]. For DPGS and SDSumm, reconstruction requires the summarized graph, original node degrees, and node-to-supernode mappings, totaling $\frac{N_s(N_s-1)}{2} + 2N$, where N_s is the number of supernodes in the summarized graph. Fig. 4 presents results on synthetic image data, including: i) Checkerboard images with region size 1, used to ensure that the graph construction method does not affect the results. Notably, it is straightforward to demonstrate that for this data, when adjusted to achieve half sparsity graph under the minimization problem of (3), all non-zero weights are $\sqrt{2/(N-2)}$. ii) Checkerboard images with region size 4. iii) The Cameraman image, divided into blocks of different sizes. Results are averaged over blocks of the same size, and tolerances are also reported.

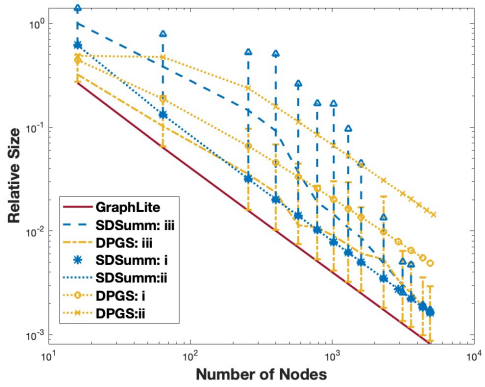

Fig. 4: Relative Output Size Comparison. (i) Checkerboard with region size 1, (ii) Checkerboard with region size 4, and (iii) Cameraman image.

Table I summarizes the results on two real-world datasets: the Indian Pines hyperspectral image (as presented in [23]) and the IDEAL Household dataset [21], where we specifically use the mean electricity usage during weekends as the signal. It is noteworthy that the *GraphLite* consistently maintains a constant relative output size across different number of nodes, regardless of the input data and sparsity level. This stability sets it apart from other methods. Furthermore, our approach yields a smaller relative output size in most cases, signifying efficient summarization.

As shown in Table I, the *GraphLite* can be constructed with negligible time and computational overhead. Our method builds directly on standard graph learning procedures widely used in GSP. Unlike general-purpose summarization methods that often introduce additional processing, our compact representation incurs minimal cost beyond the graph learning step itself—an operation already integral to many graph learning methods.

REFERENCES

[1] 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 Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[2] G. Shen, W.-S. Kim, S. K. Narang, A. Ortega, J. Lee, and H. Wey, "Edge-adaptive transforms for efficient depth map coding," in *28th Picture Coding Symposium*. IEEE, 2010, pp. 566–569.

[3] W.-S. Kim, S. K. Narang, and A. Ortega, "Graph based transforms for depth video coding," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 813–816.

[4] G. Fracastoro, D. Thanou, and P. Frossard, "Graph transform learning for image compression," in *2016 Picture Coding Symposium (PCS)*, 2016, pp. 1–5.

[5] ——, "Graph transform optimization with application to image compression," *IEEE Transactions on Image Processing*, vol. 29, pp. 419–432, 2020.

[6] P. Boldi and S. Vigna, "The webgraph framework i: compression techniques," in *The Web Conference*, 2004. [Online]. Available: https://api.semanticscholar.org/CorpusID:14428620

[7] P. Boldi, M. Rosa, M. Santini, and S. Vigna, "Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks," 2011.

[8] A. Farzan and J. Munro, "Succinct representations of arbitrary graphs," 09 2008, pp. 393–404.

[9] A. Farzan and J. I. Munro, "Succinct encoding of arbitrary graphs," *Theor. Comput. Sci.*, vol. 513, p. 38–52, nov 2013. [Online]. Available: https://doi.org/10.1016/j.tcs.2013.09.031

[10] K. Lee, H. Jo, J. Ko, S. Lim, and K. Shin, "Ssumm: Sparse summarization of massive graphs," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 144–154.

[11] H. Zhou, S. Liu, K. Lee, K. Shin, H. Shen, and X. Cheng, *DPGS: Degree-Preserving Graph Summarization*, pp. 280–288. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611976700.32

[12] H. Zhou, S. Liu, H. Shen, and X. Cheng, *Graph Summarization for Preserving Spectral Characteristics*, pp. 271–279. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611978032.31

[13] M. Lai, Y. Huang, Z. Liu, and K. Wu, "An optimized lossless graph summarization for large-scale graphs," in *2023 IEEE 29th International Conference on Parallel and Distributed Systems (ICPADS)*, 2023, pp. 355–362.

[14] J. Ko, Y. Kook, and K. Shin, "Incremental lossless graph summarization," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '20. ACM, aug 2020. [Online]. Available: http://dx.doi.org/10.1145/3394486.3403074

[15] M. Besta and T. Hoefler, "Survey and taxonomy of lossless graph compression and space-efficient graph representations," *ArXiv*, vol. abs/1806.01799, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:46940392

[16] Y. Liu, T. Safavi, A. Dighe, and D. Koutra, "Graph summarization methods and applications: A survey," *ACM computing surveys (CSUR)*, vol. 51, no. 3, pp. 1–34, 2018.

[17] P. Mishra, S. Kumar, and M. K. Chaube, "Graph interpretation, summarization and visualization techniques: A review and open research issues," *Multimedia Tools and Applications*, vol. 82, no. 6, pp. 8729–8771, 2023.

[18] N. Shabani, J. Wu, A. Beheshti, Q. Z. Sheng, J. Foo, V. Haghighi, A. Hanif, and M. Shahabikargar, "A comprehensive survey on graph summarization with graph neural networks," *IEEE Transactions on Artificial Intelligence*, 2024.

[19] V. Kalofolias, "How to learn a graph from smooth signals," in *The 19th International Conference on Artificial Intelligence and Statistics (AISTATS 2016). Journal of Machine Learning Research (JMLR)*, 2016.

[20] M. F. Baumgardner, L. L. Biehl, and D. A. Landgrebe, "220 band aviris hyperspectral image data set: June 12, 1992 indian pine test site 3," https://engineering.purdue.edu/~biehl/MultiSpec/hyperspectral.html, 2015.

[21] M. Pullinger, J. Kilgour, N. Goddard, N. Berliner, L. Webb, M. Dzikovska, H. Lovell, J. Mann, C. Sutton, J. Webb, and M. Zhong, "The ideal household energy dataset, electricity, gas, contextual sensor data and survey data for 255 uk homes," *Scientific Data*, vol. 8, no. 1, p. 146, 2021. [Online]. Available: https://doi.org/10.1038/s41597-021-00921-y

[22] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, and D. K. Hammond, "GSPBOX: A toolbox for signal processing on graphs," *ArXiv e-prints*, Aug. 2014.

[23] D. Hong, L. Gao, J. Yao, B. Zhang, A. Plaza, and J. Chanussot, "Graph convolutional networks for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 7, pp. 5966–5978, 2021.