Finding Representative Sampling Subsets on Graphs via Submodularity

Li, Tianyi; Leus, Geert

DOI
10.1109/ICASSP48485.2024.10448026

Publication date
2024

Document Version
Final published version

Published in
2024 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2024 - Proceedings

ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 979-8-3503-4485-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICASSP48485.2024.10448026

Finding Representative Sampling Subsets on Graphs via Submodularity

Tianyi Li
Signal Processing Systems
Delft University of Technology
The Netherlands

Geert Leus
Signal Processing Systems
Delft University of Technology
The Netherlands

***Abstract*—In this work, we deal with the problem of reconstructing a complete bandlimited graph signal from partially sampled noisy measurements. For a known graph structure, an efficient greedy algorithm is presented to partition the graph nodes into disjoint subsets such that sampling the graph signal from any subset leads to a sufficiently accurate reconstruction. Furthermore, we consider a scenario where the graph is massive and data processing centrally is no longer practical. To overcome this issue, a distributed framework is proposed that allows us to implement partitioning algorithms in a parallelized fashion. Finally, we provide numerical simulation results on synthetic and real-world data to show that our proposals outperform the state-of-the-art.**

***Index Terms*—Graph signal processing, sampling on graphs, submodular optimization.**

I. Introduction

Graph signal processing (GSP) is an emerging field that studies graph-supported signals, extending the notion of time-varying signals to signals on irregular domains like graphs [1].

Due to the massive size of many real-world graphs, e.g., an internet of things (IoT) sensor network, sampling all the data points is often highly energy-consuming. Motivated by this challenge, we focus on solving the following problem in this work: given a graph structure, how to partition it into disjoint subsets where noisy measurements are performed such that the entire graph signal can be reconstructed by all subsets with high accuracy on average. With such a partition, we can reduce transmission energy by sampling representative subsets one by one at each sampling round instead of transmitting readings from every sensor.

Many works exist that focus on designing the "best" sampling subset to reconstruct a graph signal [2], [3], [4], [5]. In particular, [6] aims at finding one fixed-size sampling subset based on the minimum mean square error (MSE) criterion. A later work [7] concentrates on finding a partition with the maximum number of disjoint subsets for a given MSE threshold. Then [8] considers the problem that minimizes the MSE of the worst subset for a fixed number of subsets. Several greedy heuristics are proposed for these partitioning problems, but none yields a performance guarantee.

In this paper, we formulate the problem as a submodular welfare (SW) problem based on D-optimal design and present a greedy algorithm that yields a $\frac{1}{2}$-approximation ratio. Then, a distributed framework is proposed, which is helpful for large-scale datasets. Numerical results on synthetic and real-world graphs show the advancement of our methods. In particular, we show that our presented algorithm outperforms the simultaneous iterative partitioning (SIP) algorithm proposed in [8], which is considered the state-of-the-art.

II. Background

A. Graph Signal Processing

A graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is defined as a finite set of nodes $\mathcal{N} = \{v_1, v_2, \ldots, v_N\}$ and a limited set of edges $\mathcal{E}$ [9], which is a generic data representation form that is useful for depicting the topological structure of data domains in various applications, including social networks, transportation networks, sensor networks, brain networks, etc. [1]. The weight w_{ij} associated with node i and j usually represents the similarity between the two vertices; a high weight suggests the corresponding nodes are similar, while a zero weight means that the pair of nodes are not connected.

Algebraic matrices, for example, the adjacency matrix and the Laplacian matrix, can represent graphs. The (weighted) adjacency matrix $\mathbf{A}$ of a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is simply given by

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

In this work, we assume that all the graphs are undirected. Hence, their corresponding adjacency matrices are symmetric. We also introduce the degree matrix $\mathbf{D} = \operatorname{diag}(d_1, \ldots, d_N)$, where

$$d_i = \sum_{j=1}^{N} w_{ij}. \tag{2}$$

Once we obtain the adjacency matrix and the degree matrix of a graph, its Laplacian matrix is given by

$$\mathbf{L} = \mathbf{D} - \mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top, \tag{3}$$

where the second equality shows its (spectral) eigen decomposition: $\mathbf{\Lambda} = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_N)$ is a diagonal matrix collecting the eigenvalues (graph frequencies) λ_n and the orthogonal matrix $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N]$ collects the eigenvectors (graph frequency modes) $\mathbf{u}_n$. Without loss of generality, we assume the eigenvalues are placed in increasing order for convenience.

By assigning a scalar value to each node, we can define a signal on top of the graph, called a graph signal, defined as $\mathbf{x} = [x_1, x_2, \ldots, x_N]^\top$, where the ith value corresponds to node i. Based on the eigendecomposition of $\mathbf{L}$, the Fourier transform of $\mathbf{x}$ on the graph is defined as $\hat{\mathbf{x}} = \mathbf{U}^\top \mathbf{x}$. We say a graph signal $\mathbf{x}$ is K-bandlimited if $\hat{x}_k = 0, \forall k > K$. Apparently, for a K-bandlimited graph signal, $\mathbf{x} = \mathbf{U}_K \hat{\mathbf{x}}_K$, where $\mathbf{U}_K$ stacks the first K columns of $\mathbf{U}$ and $\hat{\mathbf{x}}_K$ collects the first K entries of $\hat{\mathbf{x}}$.

B. Submodular Function

Here we review some definitions related to submodular functions.

Definition 1 (Submodularity)**.** *A set function f: $2^{\mathcal{N}} \to \mathbb{R}$ is a submodular function iff for all $\mathcal{A}, \mathcal{B} \subseteq \mathcal{N}$,*

$$f(\mathcal{A}) + f(\mathcal{B}) \geqslant f(\mathcal{A} \cup \mathcal{B}) + f(\mathcal{A} \cap \mathcal{B}).$$

An equivalent definition known as "diminishing returns" is given by

$$f(\mathcal{A} \cup \{k\}) - f(\mathcal{A}) \geqslant f(\mathcal{B} \cup \{k\}) - f(\mathcal{B}),$$

for all $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{N}, k \notin \mathcal{B}$ and $k \in \mathcal{N}$.

Definition 2 (Monotonicity)**.** *A submodular function f : $2^{\mathcal{N}} \to \mathbb{R}$ is monotone if*

$$f(\mathcal{A}) \leqslant f(\mathcal{B}),$$

$\forall \mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{N}$.

Definition 3 (Normalization)**.** *A submodular function f is said to be normalized if*

$$f(\varnothing) = 0.$$

III. System Model

We consider the model $\mathbf{y} = \mathbf{x} + \mathbf{w}$, where $\mathbf{y}, \mathbf{x}, \mathbf{w} \in \mathbb{R}^N$, $\mathbf{x}$ is a Gaussian K-bandlimited graph signal with zero mean and covariance $\mathbf{\Lambda_x} = \mathbb{E}[\mathbf{x}\mathbf{x}^\top] = \mathbf{U}_K \mathbf{\Lambda}_{\hat{\mathbf{x}}} \mathbf{U}_K^\top$ with $\mathbf{\Lambda}_{\hat{\mathbf{x}}} = \mathbb{E}[\hat{\mathbf{x}}_K \hat{\mathbf{x}}_K^\top]$ and $\mathbf{w}$ is a Gaussian noise vector with zero mean and covariance $\mathbf{\Lambda_w} = \mathbb{E}[\mathbf{w}\mathbf{w}^\top] = \mathbf{I}$.

Now consider the problem: given a partial noisy measurement of the graph signal $\mathbf{y}_\mathcal{S} = \mathbf{\Phi}\mathbf{y}$, where $\mathbf{\Phi} \in \{0,1\}^{|\mathcal{S}| \times N}$ denotes the sampling operator that samples the nodes $\mathcal{S}$ from $\mathcal{N}$, we are asked to reconstruct the entire graph signal. The estimated signal $\mathbf{x}'$ is denoted as $\mathbf{x}' = \mathbf{\Psi}\mathbf{y}_\mathcal{S} = \mathbf{\Psi}\mathbf{\Phi}\mathbf{y}$, where $\mathbf{\Psi}$ is the interpolation operator. The corresponding error covariance matrix $\mathbf{K}$ is given by

$$\mathbf{K} = \mathbb{E}(\mathbf{x} - \mathbf{x}')(\mathbf{x} - \mathbf{x}')^\top. \tag{4}$$

As shown in detail in [6], the error covariance matrix of the optimal interpolation corresponding to subset $\mathcal{S}$ is given by

$$\mathbf{K} = \mathbf{U}_K \left(\mathbf{\Lambda}_{\hat{\mathbf{x}}}^{-1} + \mathbf{U}_K^\top \mathbf{\Phi}^\top \mathbf{\Phi} \mathbf{\Lambda}_{\boldsymbol{w}}^{-1} \mathbf{\Phi}^\top \mathbf{\Phi} \mathbf{U}_K \right)^{-1} \mathbf{U}_K^\top \tag{5}$$

$$= \mathbf{U}_K \left(\sum_{i \in \mathcal{S}} \tilde{\mathbf{u}}_i \tilde{\mathbf{u}}_i^\top + \mathbf{\Lambda}_{\hat{\mathbf{x}}}^{-1} \right)^{-1} \mathbf{U}_K^\top, \tag{6}$$

where $\mathbf{U}_K = [\tilde{\mathbf{u}}_1 \cdots \tilde{\mathbf{u}}_N]^\top$. According to (6), the error covariance matrix $\mathbf{K}$ is totally impacted by $\mathcal{S}$ through the middle factor, which we denote as $\mathbf{K}(\mathcal{S})$:

$$\mathbf{K}(\mathcal{S}) = \left(\sum_{i \in \mathcal{S}} \tilde{\mathbf{u}}_i \tilde{\mathbf{u}}_i^\top + \mathbf{\Lambda}_{\hat{\mathbf{x}}}^{-1} \right)^{-1}. \tag{7}$$

An often used scalarization to evaluate the quality of estimation is based on the log volume of the η confidence ellipsoid [10]: $\log\det\{(\mathbf{\Lambda}_{\hat{\mathbf{x}}}^{-1} + \sum_{i \in \mathcal{S}} \tilde{\mathbf{u}}_i \tilde{\mathbf{u}}_i^\top)^{-1}\}$, which is called the D-optimality criterion. Based on this, we define another scalar function $f(\mathcal{S}) = \log\det\{\mathbf{\Lambda}_{\hat{\mathbf{x}}}^{-1} + \sum_{i \in \mathcal{S}} \tilde{\mathbf{u}}_i \tilde{\mathbf{u}}_i^\top\} - \log\det\{\mathbf{\Lambda}_{\hat{\mathbf{x}}}^{-1}\}$, which is composed of minus the log volume of the η confidence ellipsoid where the inverse operator is removed compared to the D-optimal design, and a normalization constant $\log\det\{\mathbf{\Lambda}_{\hat{\mathbf{x}}}^{-1}\}$ that guarantees $f(\mathcal{S}) = \varnothing$. According to [11], $f(\mathcal{S})$ qualifies how informative the node signals are when $\mathcal{S}$ is sampled. A larger function value indicates that set $\mathcal{S}$ carries more information.

An important fact we will exploit later is that $f(\mathcal{S})$ is a normalized monotone submodular function. A proof can be found in [12].

IV. Problem Formulation

We can now formulate our partitioning problem as maximizing the sum of information contributions over all sampling subsets:

$$\begin{aligned} \text{maximize} \quad & f(\mathcal{S}_1) + \cdots + f(\mathcal{S}_P) \\ \text{subject to} \quad & \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_P = \mathcal{N} \\ & \mathcal{S}_i \cap \mathcal{S}_j = \varnothing, \forall i, j, \end{aligned} \tag{8}$$

where $\mathcal{N}$ is the ground set of nodes. The problem of maximizing the sum of monotone submodular functions like (8) is called an SW problem [13]. Our partitioning problem is a special case of such problems where the functions for all subsets are the same.

V. Algorithm

In [14], a simple greedy algorithm giving a $\frac{1}{2}$-approximation is presented, where the performance guarantee holds for any P. The idea is as follows: Initialize all subsets with the empty set. Then, for each node, always add it to the subset which benefits the most. This algorithm is shown in Algorithm 1.

Algorithm 1 Greedy Algorithm for Graph Partitioning

Input: ground set $\mathcal{N}$
Output: partition $\{\mathcal{S}_1, \cdots, \mathcal{S}_P\}$

```
S_p ← ∅, p = 1, ··· , P
for i ∈ N do
    j ← arg max_{p∈{1,...,P}} f(S_p ∪ {i}) − f(S_p)
    S_j ← S_j ∪ {i}
end for
```

VI. Distributed Partitioning

To handle massive data, several distributed algorithms for submodular maximization have been proposed recently [15], [16], [17], where a single representative subset is found. In this work, we propose a distributed framework for the partitioning problem where disjoint representative subsets can be obtained by parallelizing any serial partitioning algorithm.

The MapReduce style computing model [18] has proven successful in many large-scale machine learning and data mining algorithms [19], [20], [15]. The general idea of MapReduce is to distribute the data to independent machines and let all machines process their allocated data in parallel. Based on this idea, we propose a distributed graph partitioning framework. We distribute all nodes to M machines such that each node is assigned to one of the machines. Then each machine partitions its allocated nodes using Algorithm 1 or any other algorithm so that we obtain M local partitions. In the last step, we merge all local partitions obtained by the M machines to get a global partition. This process is shown in Algorithm 2. Here $\mathcal{S}_p^{\mathrm{d}}$ denotes the p-th representative subset obtained by the distributed algorithm, $\mathcal{V}_m$ is the set of nodes allocated to the m-th machine, and $\mathcal{S}_p^{(m)}$ yields the p-th local representative subset corresponding to the m-th machine.

Algorithm 2 Distributed Partitioning

Input: ground set $\mathcal{N}$, partition size P, number of machines M
Output: partition $\{\mathcal{S}_1^{\mathrm{d}}, \cdots \mathcal{S}_P^{\mathrm{d}}\}$

1: Phase 1: Distribute nodes into M disjoint sets $\mathcal{V}_1, \cdots, \mathcal{V}_M$
2: **for** $m \in \{1, \cdots, M\}$ **do in parallel**
3: Phase 2: run some partitioning algorithm, resulting in the output $\{\mathcal{S}_1^{(m)}, \cdots \mathcal{S}_P^{(m)}\}$
4: **end for**
5: Phase 3: Merge the partitions: $\mathcal{S}_p^{\mathrm{d}} \leftarrow \mathcal{S}_p^{(1)} \cup \mathcal{S}_p^{(2)} \cup \cdots \cup \mathcal{S}_p^{(M)}, \forall p = 1, \cdots, P$

In the remainder of this section, we derive a lower bound for this distributed framework. To assist this derivation, the following lemma is introduced first.

Lemma 1. *For any disjoint subsets $\mathcal{S}_1, \mathcal{S}_2, \cdots, \mathcal{S}_l$,*

$$f(\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_l) \leqslant f(\mathcal{S}_1) + \cdots + f(\mathcal{S}_l). \tag{9}$$

Proof. Denote $f(\mathcal{A} \mid \mathcal{B}) = f(\mathcal{A} \cup \mathcal{B}) - f(\mathcal{B})$. Since f is normalized, when $\mathcal{A}$ and $\mathcal{B}$ are disjoint, $f(\mathcal{A} \cap \mathcal{B}) = 0$. Thus,

$$f(\mathcal{A} \mid \mathcal{B}) \leqslant f(\mathcal{A}). \tag{10}$$

Using the telescoping sum, we have

$$f(\mathcal{N}) = f(\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_l) \tag{11}$$
$$= f(\mathcal{S}_1) + f(\mathcal{S}_2 \mid \mathcal{S}_1) + \cdots + f(\mathcal{S}_l \mid \mathcal{S}_{l-1} \cup \cdots \cup \mathcal{S}_1). \tag{12}$$

Combining (10) and (12) leads to (9). □

We are now ready to state the main theorem. Here we define $opt(P)$ as the optimal value of (8) where P indicates the number of subsets; accordingly, $opt(M)$ is the optimal value of the problem (8) if the number of subsets is equal to the number of machines M.

Theorem 1. *If the nodes are distributed to the M machines as $\mathcal{V}_1, \cdots, \mathcal{V}_M$ uniformly at random in Phase 1, then for any partitioning algorithm in Phase 2, we have*

$$\mathbb{E}\left[\sum_{p=1}^{P} f(\mathcal{S}_p^{\mathrm{d}})\right] \geqslant \left[1 - \left(1 - \frac{1}{M}\right)^{M-1}\right] \cdot \frac{opt(M)}{M}. \tag{13}$$

Proof. By lemma 1, for any partition $\{\mathcal{S}_1^{(m)}, \cdots, \mathcal{S}_P^{(m)}\}$ of $\mathcal{V}_m$, we have

$$f(\mathcal{S}_1^{(m)}) + \cdots + f(\mathcal{S}_P^{(m)}) \geqslant f(\mathcal{S}_1^{(m)} \cup \cdots \cup \mathcal{S}_P^{(m)}) \tag{14}$$
$$= f(\mathcal{V}_m), \forall m = 1, \cdots, M. \tag{15}$$

Since f is monotone,

$$f(\mathcal{S}_p^{(1)} \cup \mathcal{S}_p^{(2)} \cup \cdots \cup \mathcal{S}_p^{(M)}) \geqslant f(\mathcal{S}_p^{(m)}), \forall m, p. \tag{16}$$

Therefore,

$$\sum_{p=1}^{P} f(\mathcal{S}_p^{\mathrm{d}}) = \sum_{p=1}^{P} f(\mathcal{S}_p^{(1)} \cup \mathcal{S}_p^{(2)} \cup \cdots \cup \mathcal{S}_p^{(M)}) \tag{17}$$
$$\geqslant f(\mathcal{S}_1^{(m)}) + f(\mathcal{S}_2^{(m)}) + \cdots + f(\mathcal{S}_P^{(m)}) \tag{18}$$
$$\geqslant f(\mathcal{S}_1^{(m)} \cup \mathcal{S}_2^{(m)} \cup \cdots \cup \mathcal{S}_P^{(m)}) \tag{19}$$
$$= f(\mathcal{V}_m), \forall m. \tag{20}$$

Taking the expectation on both sides, we obtain

$$\mathbb{E}[\sum_{p=1}^{P} f(\mathcal{S}_p^{\mathrm{d}})] \geqslant \mathbb{E}[f(\mathcal{V}_m)]. \tag{21}$$

Because every node in $\mathcal{V}_m$ is independently assigned with probability $\frac{1}{M}$, by the proof of the positive part of Theorem 1.6 in [21], we have

$$\mathbb{E}[f(\mathcal{V}_m)] \geqslant \left[1 - \left(1 - \frac{1}{M}\right)^{M-1}\right] \cdot \frac{opt(M)}{M}, \tag{22}$$

and hence the result holds. □

The following corollary is of interest when we have more machines than sampling subsets, i.e., $M \geq P$.

Corollary 1. *If $M \geqslant P$ and the nodes are distributed to the M machines as $\mathcal{V}_1, \cdots, \mathcal{V}_M$ uniformly at random in Phase 1, then for any partitioning algorithm in Phase 2, we have*

$$\mathbb{E}\left[\sum_{p=1}^{P} f(\mathcal{S}_p^{\mathrm{d}})\right] \geqslant \left[1 - \left(1 - \frac{1}{M}\right)^{M-1}\right] \cdot \frac{opt(P)}{M}. \tag{23}$$

Proof. We have to prove that when $M \geqslant P$, $opt(M) \geqslant opt(P)$. Without loss of generality, we assume $M = P + 1$. Denote the optimal solution w.r.t. $opt(M)$ by $\{\mathcal{V}_1^*, \cdots, \mathcal{V}_M^*\}$, and the optimal solution w.r.t. $opt(P)$ by $\{\mathcal{S}_1^*, \cdots, \mathcal{S}_P^*\}$. For an arbitrary $j = 1, \cdots, P$, we partition $\mathcal{S}_j^*$ into $\mathcal{S}_{j1}$ and $\mathcal{S}_{j2}$. By lemma 1 we have $f(\mathcal{S}_{j1}) + f(\mathcal{S}_{j2}) \geqslant f(\mathcal{S}_j^*)$ for any partition $\{\mathcal{S}_{j1}, \mathcal{S}_{j2}\}$. Therefore,

$$f(\mathcal{S}_1^*)+\cdots+f(\mathcal{S}_{j1})+f(\mathcal{S}_{j2})+\cdots+f(\mathcal{S}_P^*) \geqslant opt(P). \quad (24)$$

Since $opt(M)$ is the optimal value for M disjoint subsets, we have

$$opt(M) \geqslant f(\mathcal{S}_1^*)+\cdots+f(\mathcal{S}_{j1})+f(\mathcal{S}_{j2})+\cdots+f(\mathcal{S}_P^*), \quad (25)$$

so the result holds. □

VII. Numerical Results

In this section, we test the presented approaches on synthetic and real-world graphs to show that signals can be accurately reconstructed. Graphs in this section are generated by the GSPBOX toolbox [22].

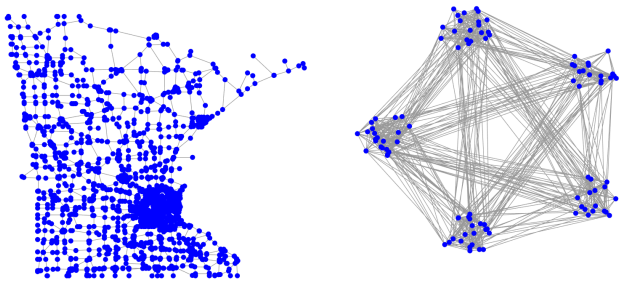

Fig. 1. Examples of graphs: the Minnesota road network with 2642 nodes(left) and a graph generated by the stochastic block model with 100 nodes(right).

The parameters are set as follows: $\mathbf{\Lambda}_{\boldsymbol{w}} = \mathbf{I}$, and $\mathbf{\Lambda}_{\hat{\mathbf{x}}} = \sigma_x \mathbf{I}$, where $\sigma_x = 2$. The objective values obtained are all normalized by the factor $P(\log\det\{\mathbf{\Lambda}_{\hat{\mathbf{x}}}^{-1} + \frac{1}{P}\mathbf{I}\} - \log\det\{\mathbf{\Lambda}_{\hat{\mathbf{x}}}^{-1}\})$ since it is an upper bound of the problem (8). The proof for this is trivial: $P(\log\det\{\mathbf{\Lambda}_{\hat{\mathbf{x}}}^{-1} + \frac{1}{P}\mathbf{I}\} - \log\det\{\mathbf{\Lambda}_{\hat{\mathbf{x}}}^{-1}\})$ is the optimal value of the convex relaxed problem of (8), and since the solution of the original problem lies in the feasible set of the convex relaxed problem, we obtain the above upper bound.

A. Minnesota road graph

In this experimental setup, we use a real-world graph, the Minnesota road map graph with $N = 2642$ nodes. The parameters are set as follows: the number of disjoint subsets $P = 5$, the bandwidth of the graph signal $K = 20$, the number of machines for distributed implementation $M = 4$. The partition obtained by Algorithm 1 is shown in Fig. 2(left). The results of the presented approaches, along with the SIP algorithm, are shown in the second column of Table I. We remark here that the distributed version of each algorithm shown in the table is implemented using Algorithm 2. In Phase 1, nodes are distributed to the machines uniformly at random; in Phase 2, we then use the Greedy or SIP algorithm accordingly.

TABLE I
Performance of presented partitioning techniques

	Normalized objective value	
	Minnesota	***Stochastic block model***
Greedy	0.9999	0.8988 ± 0.0175
SIP	0.9847	0.6668 ± 0.0422
Distributed Greedy	0.9995	0.9091 ± 0.0182
Distributed SIP	0.9981	0.8880 ± 0.0173

B. Graphs produced by stochastic block model

We next test our results on 10000 realizations of a stochastic block model with $N = 100$, where the intra-cluster edge probability is set to 0.7, and the inter-cluster edge probability to 0.3. P is set to 5, the bandwidth to $K = 10$, and the number of machines to $M = 4$. The partition obtained by Algorithm 1 on one realization is shown in Fig. 2(right). The results of the presented approaches and the SIP algorithm are shown in the third column of Table I. Note that we denote by Mean ± STD the mean and standard deviation of the results of these realizations.

Table I shows that the presented algorithm outperforms SIP and is near-optimal. In addition, our distributed framework does not decrease the performance of these partitioning techniques. In contrast, it even boosts SIP to a fairly significant extent.

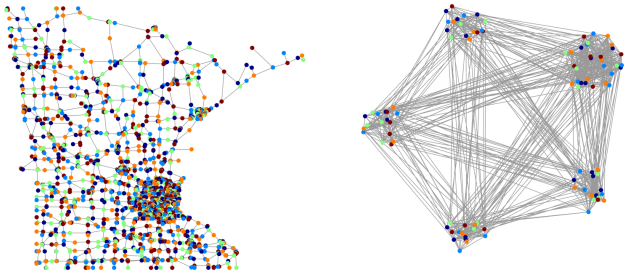

Fig. 2. The result of applying Algorithm 1 on the Minnesota road map graph(left) and a random graph generated by the stochastic block model(right).

VIII. Conclusion

In this work, we have formulated the graph node partitioning problem. We have shown that our problem formulation is a particular case of an SW problem. A distributed framework is proposed afterward, which allows to parallelize these algorithms by distributing the nodes over some local machines. We have tested our proposed methods on synthetic and real-world graphs and showed that these methods are near-optimal and outperform the state-of-the-art partitioning scheme. A potential further work is to derive a tighter lower bound for the distributed framework.

REFERENCES

[1] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3):83–98, 2013.

[2] Alessandro Chiumento, Nicola Marchetti, and Irene Macaluso. Energy efficient wsn: a cross-layer graph signal processing solution to information redundancy. In *2019 16th International Symposium on Wireless Communication Systems (ISWCS)*, pages 645–650. IEEE, 2019.

[3] Siheng Chen, Aliaksei Sandryhaila, and Jelena Kovačević. Sampling theory for graph signals. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3392–3396. IEEE, 2015.

[4] Sundeep Prabhakar Chepuri, Yonina C Eldar, and Geert Leus. Graph sampling with and without input priors. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4564–4568. IEEE, 2018.

[5] Guillermo Ortiz-Jiménez, Mario Coutino, Sundeep Prabhakar Chepuri, and Geert Leus. Sparse sampling for inverse problems with tensors. *IEEE Transactions on Signal Processing*, 67(12):3272–3286, 2019.

[6] Luiz FO Chamon and Alejandro Ribeiro. Greedy sampling of graph signals. *IEEE Transactions on Signal Processing*, 66(1):34–47, 2017.

[7] Josefine Holm, Federico Chiariotti, Morten Nielsen, and Petar Popovski. Lifetime maximization of an internet of things (iot) network based on graph signal processing. *IEEE Communications Letters*, 25(8):2763–2767, 2021.

[8] Roshni Chakraborty, Josefine Holm, Torben Bach Pedersen, and Petar Popovski. Finding representative sampling subsets in sensor graphs using time series similarities. *arXiv preprint arXiv:2202.08504*, 2022.

[9] Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001.

[10] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[11] Lingya Liu, Cunqing Hua, Jing Xu, Geert Leus, and Yiyin Wang. Greedy sensor selection: Leveraging submodularity based on volume ratio of information ellipsoid. *IEEE Transactions on Signal Processing*, pages 1–14, 2023.

[12] Manohar Shamaiah, Siddhartha Banerjee, and Haris Vikalo. Greedy sensor selection: Leveraging submodularity. In *49th IEEE conference on decision and control (CDC)*, pages 2572–2577. IEEE, 2010.

[13] Jan Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 67–74, 2008.

[14] Benny Lehmann, Daniel Lehmann, and Noam Nisan. Combinatorial auctions with decreasing marginal utilities. In *Proceedings of the 3rd ACM conference on Electronic Commerce*, pages 18–28, 2001.

[15] Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed submodular maximization: Identifying representative elements in massive data. *Advances in Neural Information Processing Systems*, 26, 2013.

[16] Rafael da Ponte Barbosa, Alina Ene, Huy L Nguyen, and Justin Ward. A new framework for distributed submodular maximization. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 645–654. Ieee, 2016.

[17] Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed submodular maximization. *The Journal of Machine Learning Research*, 17(1):8330–8373, 2016.

[18] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[19] Cheng-Tao Chu, Sang Kim, Yi-An Lin, YuanYuan Yu, Gary Bradski, Kunle Olukotun, and Andrew Ng. Map-reduce for machine learning on multicore. *Advances in neural information processing systems*, 19, 2006.

[20] Jaliya Ekanayake, Shrideep Pallickara, and Geoffrey Fox. Mapreduce for data intensive scientific analyses. In *2008 IEEE Fourth International Conference on eScience*, pages 277–284. IEEE, 2008.

[21] Moran Feldman. Maximizing symmetric submodular functions. *ACM Transactions on Algorithms (TALG)*, 13(3):1–36, 2017.

[22] Nathanaël Perraudin, Johan Paratte, David Shuman, Lionel Martin, Vassilis Kalofolias, Pierre Vandergheynst, and David K Hammond. Gspbox: A toolbox for signal processing on graphs. *arXiv preprint arXiv:1408.5781*, 2014.