FAST PATH LOCALIZATION ON GRAPHS VIA MULTISCALE VITERBI DECODING

Yaoqing Yang[⋆†]*, Siheng Chen*[⋆]*, Mohammad Ali Maddah-Ali*[†]*, Pulkit Grover*[⋆]
Soummya Kar[⋆]*, Jelena Kovacevic*[⋆]

[⋆] Department of Electrical and Computer Engineering, Carnegie Mellon University
[†] Department of Wireless Communications, Nokia Bell Labs

ABSTRACT

We consider a problem of localizing the destination of an activated path signal supported on a graph. An "activated path signal" is a graph signal that evolves over time that can be viewed as the trajectory of a moving agent. We show that by combining dynamic programming and graph partitioning, the computational complexity of destination localization can be significantly reduced. Then, we show that the destination localization error can be upper-bounded using methods based on large-deviation. Using simulation results, we show a tradeoff between the destination localization error and the computation time. We compare the dynamic programming algorithm with and without graph partitioning and show that the computation time can be significantly reduced by using graph partitioning. The proposed technique can scale to the problem of destination localization on a large graph with one million nodes and one thousand time slots.

1. INTRODUCTION

Data with unstructured forms and rich types are being generated from various sources, such as social networks and the Internet of Things. These data are often generated with some inherent correlation that can be represented using graphs. This important feature inspired the emerging field on graph signal processing [1, 2], where signals are supported on a graph, instead of on the Euclidean domain (e.g., time signals or image signals). This key difference spurred works that aim to generalize classical problems and techniques to graph signal processing, including sampling [3, 4], recovery [5, 6], signal representations [7, 8], uncertainty principles [9, 10], and graph signal transforms [11–15].

In this paper, we study a special type of graph signals that evolve over time called "path signals". A path signal (see Fig. 1 for details) only has non-zero value on a connected trajectory, i.e., the signal is non-zero at only one location on each time slot of the graph signal, and the non-zero locations at consecutive time slots are connected on the graph. A path signal is a perfect abstraction of a moving agent on a graph, where the non-zero location of the path signal at a particular time slot represents the location of the moving agent at time t. Thus, the study of path signals is deeply related to the literature of tracking and surveillance [16–19]. Here, we study the path signal on large-scale graphs from the perspective of graph partitioning and graph signal dimension reduction. Due to the increasing size of graphs, many graph partitioning and graph signal dimension reduction techniques have been proposed, which include community detection and clustering on graphs [20–23], and signal coarsening on graphs [24–26]. These newly proposed techniques and related ideas have provided great improvements in computation speed and storage cost for algorithms on large-scale graphs, including PageRank [27], graph generation [28] and graph semantic summarization [29]. Here, we use dimension reduction techniques in signal tracking, with the specific focus on path signals.

We consider the problem of "destination localization" for a path signal on a large-scale graph. The aim is to estimate the final position of the moving agent from noisy observations of the path signal. We measure the accuracy of destination localization using the Euclidean distance in the real space where the graph is embedded in (i.e., a geometric graph). First, we propose to use dynamic programming to obtain an estimate of the destination of the path signal on the original graph. Then, we propose to use graph partitioning techniques to divide the graph into small communities and reduce the graph size by merging one community into one "super-node". Then, we carry out dynamic programming algorithms on a graph formed by these super-nodes and significantly reduce the number of states in dynamic programming, and hence improve the computation time. Using a large-deviation-based technique, we provide an upper bound on the error of the destination localization which can be computed in polynomial time for general graphs and general non-overlapping graph partitioning algorithms. Finally, we test this fast dynamic programming technique both on random geometric graphs and a real graph called AS-Oregon [30]. Our algorithm shows significant speedup compared to dynamic programming without graph partitioning, and it obtains comparable localization error. In MATLAB simulations, the proposed technique can scale to the path localization problem on a graph with one million nodes and with one thousand time slots.

The proposed path localization problem does not only apply to tracking problems, but also applies to road congestion monitoring and satellite searching. A signal path on a road network can be viewed as a slowly moving congested segment on the graph formed by roads and intersections. A signal path in satellite searching can be viewed as the trajectory of a moving plane debris or a small refugee lifeboat in the sea, while the observation noise may come from the satellite inaccuracy and the bad illumination condition at night.

978-1-5090-4117-6/17/$31.00 ©2017 IEEE

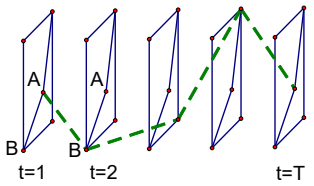

Fig. 1. This figure illustrates a path signal on a graph with five nodes. The nodes with non-zero signal value form a connected path (green dashed line). For example, the activated node at time $t = 1$ is $v_1 = A$ and the activated node at time $t = 2$ is $v_2 = B$. For the signal to be a path signal we require $(v_1, v_2) \in \mathcal{E}$. A path signal can be viewed as an abstraction of a moving agent on a graph.

2. SYSTEM MODEL: PATH SIGNAL AND PATH LOCALIZATION

Denote by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ an undirected graph with n nodes. Assume time is slotted. Suppose $\mathbf{x}_t, t = 1, 2, \ldots, T$ is a fixed series of (non-random) signals supported on the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ that evolve over time. The value $\mathbf{x}_t(v)$ denotes the signal value at time t and at node v. At each time slot, there is one node v_t such that $\mathbf{x}_t(v_t) = \mu$ and $\mathbf{x}_t(v) = 0$ for all other nodes $v \neq v_t$. The collection of the nodes $(v_1, v_2, \ldots, v_T)$ is a connected path, i.e., $(v_t, v_{t+1}) \in \mathcal{E}$ for all $t = 1, 2, \ldots, T-1$. We use $\mathbf{p}_0$ to denote this connected path and call it the "true path" on the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The true path $\mathbf{p}_0 = (v_1, v_2, \ldots, v_T)$ can represent the trajectory of the moving agent on the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ from time $t = 1$ to time $t = T$. Assume that we observe the fixed graph signal $\mathbf{x}_t, t = 1, 2, \ldots, T$ from the observation model

$$\mathbf{y}_t = \mathbf{x}_t + \mathbf{w}_t, t = 1, 2, \ldots, T, \tag{1}$$

where the noise $\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$. Our goal is to estimate the final position v_T of the path signal $(v_1, v_2, \ldots, v_T)$ on the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with elevated mean μ from the noisy observations $\mathbf{y}_t, t = 1, 2, \ldots, T$. We define the sum weight on an arbitrary path $\mathbf{p} = (v_1, v_2, \ldots, v_T) \in \mathcal{V}^T$ as

$$S(\mathbf{p}) = \sum_{t=1}^{T} \mathbf{y}_t(v_t). \tag{2}$$

Intuitively, a path with a higher sum weight should be more likely to be the true path $\mathbf{p}_0$. To see this, we have

$$\begin{aligned} \Pr\left((\mathbf{y}_1, \ldots, \mathbf{y}_T) | \mathcal{P}\right) &= \prod_{t=1}^{T} \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{y}_t - \mathbf{x}_t\|_2^2\right) \\ &= \exp\left(-\frac{1}{2\sigma^2} \sum_{t=1}^{T} \|\mathbf{y}_t - \mathbf{x}_t\|_2^2\right). \end{aligned}$$

Then,

$$\begin{aligned} &\arg\max_{\mathcal{P}} \Pr\left((\mathbf{y}_1, \ldots, \mathbf{y}_T) | \mathcal{P}\right) = \arg\min_{\mathcal{P}} \sum_{t=1}^{T} \|\mathbf{y}_t - \mathbf{x}_t\|_2^2 \\ &= \arg\max_{\mathcal{P}} \sum_{t=1}^{T} \mathbf{y}_t \cdot \mathbf{x}_t = \arg\max_{\mathcal{P}} \sum_{t=1}^{T} \mu \mathbf{y}_t(v_t). \end{aligned}$$

2.1. Dynamic Programming for Path Signal Localization

In Algorithm 1, we describe a dynamic programming algorithm to compute the path with the maximum sum weight. This algorithm is also known as the Viterbi decoding algorithm [31] in the specific context of convolutional decoding. The basic idea of Algorithm 1 is to record the path with the largest weight that ends all nodes v at each time slot t in the graph $\mathcal{G}$ for all time slots $t = 1, 2, \ldots, T$. Although the possible number of paths is exponential in t, algorithm 1 only has computational complexity $\mathcal{O}(nT)$, because only the optimal path, instead of all paths, that ends at a node v has to be recorded at each time t. However, the computational complexity $\mathcal{O}(nT)$ can still be high for a large graph and large overall time T, which motivates us to design algorithms that may not output the exact estimate, but have low computational complexity (see Section 3). Thus, we first define an error metric for the destination localization problem.

Algorithm 1 Dynamic Programming for Path Localization

INPUT: A graph $\mathcal{G}$ and observations $\mathbf{y}_t, t = 1, 2, \ldots, T$.
OUTPUT: A connected path $\hat{\mathbf{p}} = (\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_T)$.
INITIALIZE
Use $s_{v,t}$ to denote the sum weight until time t at node v. Use $\mathbf{p}_{v,t}$ to denote the path with the largest weight of length t that ends at node v. Initialize $\mathbf{p}_{v,1} = v$ for all $v \in \mathcal{V}$.
FOR t=2 : T

- For all nodes v in $\mathcal{V}$, find the path $\mathbf{p}_{u,t-1}$ with the largest sum weight $S(\mathbf{p}_{u,t-1})$ for all nodes u in the neighborhood $\mathcal{N}(v)$;
- Update $\mathbf{p}_{v,t} = (\mathbf{p}_{u,t-1}, v)$ for all $v \in \mathcal{V}$.

END
Denote by $\mathbf{p}_{v^*,T}$ the path with the largest sum weight in all paths of length T. Output $\hat{\mathbf{p}} = \mathbf{p}_{v^*,T}$.

Definition 1. The destination distance $D_F(\mathbf{p}_1, \mathbf{p}_0)$ between the estimated path $\mathbf{p}_1 = (\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_T)$ and the true path $\mathbf{p}_0 = (v_1, v_2, \ldots, v_T)$ is defined as the distance $d(\hat{v}_T, v_T)$ between the two nodes $\hat{v}_T$ and v_T, where d can be any distance measure defined for the graph $\mathcal{G}$, such as the Euclidean distance in the real space where the graph is embedded in (i.e., a geometric graph).

3. REDUCED-STATE DYNAMIC PROGRAMMING ON PARTITIONED GRAPHS

In this section, we describe a new way of doing path destination localization with graph partitioning and graph signal dimension reduction. The main idea is to first partition the graph into communities [32] and then localize the path on the graph formed by the communities[1].

[1]In this paper, we only apply existing graph partitioning algorithms such as [23]. The study of the optimal graph partitioning algorithm with low complexity is a very meaningful future work.

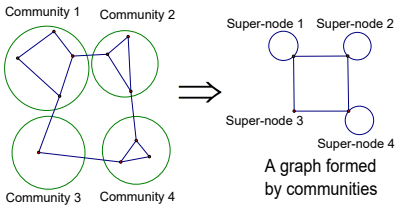

Fig. 2. The graph on the right is an illustration of the community graph. In the original graph $\mathcal{G}$, we partition the nodes into non-overlapping communities. Then, we shrink each community to one "super-node" and connect two super-nodes if there exist two connected nodes in the corresponding two communities. Note that the community graphs may have self-loops.

Suppose we partition the nodes of the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ into m non-overlapping communities

$$\mathcal{V} = \bigcup_{i=1}^{m} \mathcal{V}_i. \tag{3}$$

Then, we shrink each community into a "super-node" and construct a graph formed by these super-nodes. We use $\mathcal{G}_c = (\mathcal{V}_c, \mathcal{E}_c)$ to denote the new graph, where $\mathcal{V}_c$ with cardinality m is the set of "super-nodes", and two nodes $\mathcal{V}_i$ and $\mathcal{V}_j$ are connected if there exists two nodes $v_i \in \mathcal{V}_i$ and $v_j \in \mathcal{V}_j$ such that $(v_i, v_j) \in \mathcal{E}$ in the original graph $\mathcal{G}$. We call the graph $\mathcal{G}_c$ the community graph (see Fig. 2).

Consider the same observation model in (1) on a general graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ such that $\mathbf{x}_t(v_t) = \mu$ and $\mathbf{x}_t(v) = 0$ for $v \neq v_t$, and $(v_1, v_2, \ldots, v_T)$ is a connected path in $\mathcal{G}$. Suppose we use a graph-partitioning algorithm and obtain the community graph $\mathcal{G}_c = (\mathcal{V}_c, \mathcal{E}_c)$. We use a coarsened [24–26] version $\mathbf{u}_t$ (graph signal dimension reduction) of the original graph signal observation $\mathbf{y}_t$ as the graph signal on the community graph. The coarsened graph signal is defined as

$$\mathbf{u}_t(\mathcal{V}_i) = \max_{v \in \mathcal{V}_i} \mathbf{y}_t(v), i = 1, 2, \ldots, m. \tag{4}$$

Then, we use the same dynamic programming algorithm as Algorithm 1 to obtain an estimate of the trajectory of the path signal on the community graph.

Remark 1. Note that after graph partitioning, the sum weight maximization does not equal to the MLE. However, we still obtain a numerical method to iteratively compute an upper bound on the expectation of the destination distance between the true path and the path estimate on the community graph $\mathcal{G}_c$ (see Section 3.1). Note that our algorithm and bound apply to worst-case path signals and graph partitioning, which is different from the Bayesian settings in [18, 19].

3.1. A Numeric Method for Computing an Upper Bound on the Localization Error

Denote by $\mathcal{P}_0 = (\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_T)$ the true path in the community graph, and denote by $\hat{\mathcal{P}} = (\hat{\mathcal{V}}_1, \hat{\mathcal{V}}_2, \ldots, \hat{\mathcal{V}}_T)$ the path estimate. Note that for some t, the two paths may overlap, i.e., $\hat{\mathcal{V}}_t = \mathcal{V}_t$. Denote by S_T the sum weight of the true path $\mathcal{P}_0$ in the community graph, and denote by S_c the sum weight on the path estimate $\hat{\mathcal{P}}$ in the community graph. Then,

Algorithm 2 Coarsened Dynamic Programming for Path Localization

INPUT: A coarsened graph $\mathcal{G}_c = (\mathcal{V}_c, \mathcal{E}_c)$ and coarsened graph signal observations $\mathbf{u}_t, t = 1, 2, \ldots, T$.
OUTPUT: A connected path $\hat{\mathbf{p}} = (\hat{\mathcal{V}}_1, \hat{\mathcal{V}}_2, \ldots, \hat{\mathcal{V}}_T)$ on the community graph.
Call Algorithm 1 with inputs $\mathcal{G}_c = (\mathcal{V}_c, \mathcal{E}_c)$ and $\mathbf{u}_t, t = 1, 2, \ldots, T$.

$$\text{Sum weight on the true path } \mathcal{P}_0\text{: } S_T = \sum_{t=1}^{T} \mathbf{u}_t(\mathcal{V}_t), \tag{5}$$

$$\text{Sum weight on the path estimate } \hat{\mathcal{P}}\text{: } S_C = \sum_{t=1}^{T} \mathbf{u}_t(\hat{\mathcal{V}}_t). \tag{6}$$

In Algorithm 2, we select the path with the maximum sum weight. Therefore, we will choose the path estimate $\hat{\mathcal{P}}$ with sum weight S_C instead of the true path $\mathcal{P}_0$ with sum weight S_T, only if $S_C \geq S_T$. This event happens with exponentially low probability because the signal on the true path has a shifted mean value $\mu > 0$, while the signal on $\hat{\mathcal{P}}$ has mean value 0 when the two paths do not overlap.

Lemma 3.1. The probability that the sum weight S_C on any path $\hat{\mathcal{P}} = (\hat{\mathcal{V}}_1, \hat{\mathcal{V}}_2, \ldots, \hat{\mathcal{V}}_T)$ is greater or equal to the sum weight S_T on the true path can be upper bounded by

$$\Pr(S_C \geq S_T) \leq \prod_{t \in S} \exp\left(-\frac{\mu^2}{4\sigma^2}\right) |\hat{\mathcal{V}}_t|, \tag{7}$$

where $S \subset \{1, 2, \ldots, T\}$ is set of time slots when $\hat{\mathcal{V}}_t \neq \mathcal{V}_t$.

Using Lemma 3.1, we immediately obtain the following result on the destination distance defined in Definition 1.

Theorem 3.1. The expectation of the destination distance between the path estimate and the true path measured on the community graph is upper bounded by

$$\mathbb{E}\left[D_F(\mathcal{P}_0, \hat{\mathcal{P}})\right] \leq \sum_{\text{All possible paths } \hat{\mathcal{P}} = (\hat{\mathcal{V}}_1, \hat{\mathcal{V}}_2, \ldots, \hat{\mathcal{V}}_T)} \left[d(\mathcal{V}_T, \hat{\mathcal{V}}_T) \prod_{t \in S(\hat{\mathcal{P}})} \exp\left(-\frac{\mu^2}{4\sigma^2}\right) |\hat{\mathcal{V}}_t| \right], \tag{8}$$

where $S(\hat{\mathcal{P}}) \subset \{1, 2, \ldots, T\}$ is the set of time t when $\hat{\mathcal{V}}_t \neq \mathcal{V}_t$, and $d(\mathcal{V}_T, \hat{\mathcal{V}}_T)$ is the distance metric between the two super-nodes $\mathcal{V}_T$ and $\hat{\mathcal{V}}_T$ in the community graph.

Proof in Sketch The proof can be obtained by directly upper-bounding $\Pr(\hat{\mathcal{P}} \text{ is chosen as the estimate})$ with (7) in the following decomposition.

$$\mathbb{E}\left[D_F(\mathcal{P}_0, \hat{\mathcal{P}})\right] = \sum_{\text{All possible paths } \hat{\mathcal{P}}} \Pr(\hat{\mathcal{P}} \text{ is chosen}) D_F(\mathcal{P}_0, \hat{\mathcal{P}}). \tag{9}$$

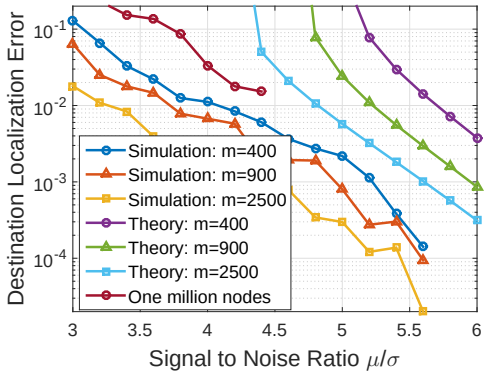

Fig. 3. Simulation results on the destination distance between the path estimate and the true path for different number of communities (super-nodes).

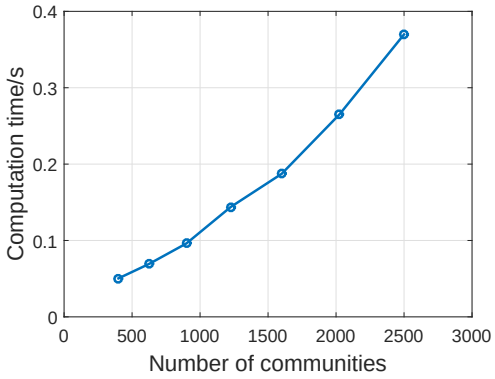

Fig. 4. Computation time of one step in Algorithm 2 versus the number of communities in the graph partitioning.

Since there are exponential number of possible paths in T time slots, one may think that the bound is not computable. However, the special structure of the bound (a sum-product structure) makes it computable in polynomial time.

Theorem 3.2. The upper bound on the expected destination distance in (8) can be computed in time $\mathcal{O}(mT)$, where m is the number of nodes in the community graph $\mathcal{G}_c = (\mathcal{V}_c, \mathcal{E}_c)$, and T is the number of time slots.

4. SIMULATION

First, we test the algorithm on a random geometric graph with 20000 randomly generated nodes that are distributed according to the Poisson point process on a square area with length 1. Two nodes are connected if they are within distance 0.02. Then, we partition the square areas into m sub-squares using direct square partitioning and merge the nodes in each square into one "super-node". The number of communities can be $m = 400$, 900 or 2500. After that, we generate a random walk on the graph to represent the positions of a moving agent and use Algorithm 2 to estimate the final position of the path signal from Gaussian observation noise. The destination distance error metric $d(\cdot, \cdot)$ in Definition 1 is defined as the Euclidean distance on the square area.

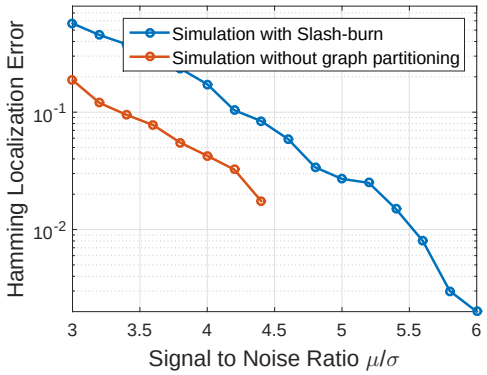

Fig. 5. Performance comparison between dynamic programming with and without graph partitioning on AS-Oregon graph.

The result in Fig. 3 shows the destination localization error versus the signal to noise ratio μ/σ with different number of communities in the graph partitioning stage. We also include one simulation for a geometric graph with one million nodes and 2500 communities. The result in Fig. 4 shows the computation time of one step in the FOR-loop in Algorithm 1 when the number of communities differ. We can see from Fig. 3 that when the number of communities increases, the localization error decreases, while the computation time increases. In practice, one should find the optimal number of communities to obtain a tradeoff between computation time and destination localization error.

Then, we test our algorithm on the AS-Oregon graph of Autonomous Systems (AS) peering information inferred from Oregon route-views [30]. We use Slashburn [23], which is a partitioning algorithm with interleaving stages of node ordering with node-centrality (slash) and removal of edges connected to high-centrality nodes (burn), to speed up the computation time. The result on the localization error versus the signal to noise ratio μ/σ is shown in Fig. 5. The computation time of one step of dynamic programming with and without graph partitioning (i.e., Algorithm 2 and Algorithm 1) are respectively 0.0085 seconds and 3.5344 seconds. Note that the destination distance (Euclidean distance) here does not have a specific meaning, so we use the Hamming distance $D_H(\mathcal{P}_0, \hat{\mathcal{P}}) = \sum_{t=1} \mathbb{1}(\hat{\mathcal{V}}_t \neq \mathcal{V}_t)$ instead.

5. CONCLUSIONS

In this paper, we study the problem of localizing the final position of a path signal on a large-scale graph. The proposed technique reduces the complexity of dynamic programming using graph partitioning. A meaningful future work is to study the effect of applying different low-complexity partitioning algorithms to the same path localization problem.

6. 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 Process. Mag.*, vol. 30, pp. 83–98, May 2013.

[2] A. Sandryhaila and J. M. F. Moura, "Big data processing with signal processing on graphs," *IEEE Signal Process. Mag.*, vol. 31, no. 5, pp. 80–90, 2014.

[3] Aamir Anis, Akshay Gadde, and Antonio Ortega, "Efficient sampling set selection for bandlimited graph signals using graph spectral proxies," *IEEE Trans. Signal Process.*, vol. 64, no. 14, pp. 3775–3789, 2015.

[4] S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević, "Discrete signal processing on graphs: Sampling theory," *IEEE Trans. Signal Process.*, vol. 63, pp. 6510 – 6523, Aug. 2015.

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

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

[7] X. Zhu and M. Rabbat, "Approximating signals supported on graphs," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Kyoto, Japan, Mar. 2012, pp. 3921 – 3924.

[8] D. Thanou, D. I. Shuman, and P. Frossard, "Learning parametric dictionaries for signals on graphs," *IEEE Trans. Signal Process.*, vol. 62, pp. 3849–3862, June 2014.

[9] A. Agaskar and Y. M. Lu, "A spectral graph uncertainty principle," *IEEE Trans. Inf. Theory*, vol. 59, no. 7, pp. 4338–4356, July 2013.

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

[11] S.l K. Narang, G. Shen, and A. Ortega, "Unidirectional graph-based wavelet transforms for efficient data gathering in sensor networks," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Dallas, TX, Mar. 2010, pp. 2902–2905.

[12] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Appl. Comput. Harmon. Anal.*, vol. 30, pp. 129–150, Mar. 2011.

[13] D. I. Shuman, B. Ricaud, and P. Vandergheynst, "Vertex-frequency analysis on graphs," *Appl. Comput. Harmon. Anal.*, February 2015, To appear.

[14] O. Teke and P. P. Vaidyanathan, "Extending classical multirate signal processing theory to graphsłpart i: Fundamentals," *IEEE Trans. Signal Process.*, vol. 65, no. 2, pp. 409–422, 2016.

[15] O. Teke and P. P. Vaidyanathan, "Extending classical multirate signal processing theory to graphsłpart ii: M-channel filter banks," *IEEE Trans. Signal Process.*, vol. 65, no. 2, pp. 423–437, 2016.

[16] S. Oh and S. Sastry, "Tracking on a graph," in *Proc. IEEE ISPN*. IEEE Press, 2005, p. 26.

[17] E. Arias-Castro, E. J. Candès, H. Helgason, and O. Zeitouni, "Searching for a trail of evidence in a maze," *The Annals of Statistics*, pp. 1726–1757, 2008.

[18] A. Agaskar and Y. M. Lu, "Detecting random walks hidden in noise: Phase transition on large graphs," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.* IEEE, 2013, pp. 6377–6381.

[19] M. Ting, A. O. Hero, D. Rugar, C.-Y. Yip, and J. A. Fessler, "Near-optimal signal detection for finite-state markov signals with application to magnetic resonance force microscopy," *IEEE Trans. Signal Process.*, vol. 54, no. 6, pp. 2049–2062, 2006.

[20] N. Tremblay and P. Borgnat, "Graph wavelets for multiscale community mining," *IEEE Trans. Signal Process.*, vol. 62, pp. 5227–5239, Oct. 2014.

[21] X. Dong, P. Frossard, P. Vandergheynst, and N. Nefedov, "Clustering on multi-layer graphs via subspace analysis on Grassmann manifolds," *IEEE Trans. Signal Process.*, vol. 62, no. 4, pp. 905–918, Feb. 2014.

[22] P.-Y. Chen and A. O. Hero, "Deep community detection," *IEEE Trans. Signal Process.*, vol. 63, no. 21, pp. 5706–5719, 2015.

[23] Y. Lim, U. Kang, and C. Faloutsos, "Slashburn: Graph compression and mining beyond caveman communities," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 12, pp. 3077–3089, 2014.

[24] S. Lafon and A. B. Lee, "Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 9, pp. 1393–1403, 2006.

[25] D. I. Shuman, M. J. Faraji, and P. Vandergheynst, "A multiscale pyramid transform for graph signals," *IEEE Trans. Signal Process.*, vol. 64, no. 8, pp. 2119–2134, 2016.

[26] P. Liu, X. Wang, and Y. Gu, "Graph signal coarsening: Dimensionality reduction in irregular domain," in *Proc. GlobalSIP 2014*. IEEE, 2014, pp. 798–802.

[27] J. Jung, K. Shin, L. Sael, and U. Kang, "Random walk with restart on large graphs using block elimination," *ACM Trans. Database Syst.*, vol. 41, no. 2, pp. 12, 2016.

[28] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, "Kronecker graphs: An approach to modeling networks," *J. Mach. Learn. Res.*, vol. 11, pp. 985–1042, 2010.

[29] D. Koutra, U. Kang, J. Vreeken, and C. Faloutsos, "Summarizing and understanding large graphs," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 8, no. 3, pp. 183–202, 2015.

[30] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: densification laws, shrinking diameters and possible explanations," in *Proc. ACM SIGKDD. Int. Conf. Knowl. Discovery Data Mining*. ACM, 2005, pp. 177–187.

[31] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, vol. 13, no. 2, pp. 260–269, 1967.

[32] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3, pp. 75–174, 2010.