Signal Localization, Decomposition and Dictionary Learning on Graphs

Siheng Chen, Student Member, IEEE, Yaoqing Yang, Student Member, IEEE, José M. F. Moura, Fellow, IEEE, Jelena Kovačević, Fellow, IEEE

Abstract—Motivated by the need to extract meaning from large amounts of complex data that can be best modeled by graphs, we consider three critical problems in graph signal processing: localization, decomposition and dictionary learning. We are particularly interested in piecewise-constant graph signals that efficiently model local information in the vertex-domain; for such signals, we show that decomposition and dictionary learning are natural extensions of localization. For each of the three problems, we propose a specific graph signal model, an optimization problem and a computationally efficient solver. We then conduct an extensive empirical study to validate the proposed methods on both simulated and real data including the analysis of a large volume of spatio-temporal Manhattan urban data. Using our methods, we are able to detect both everyday and special events and distinguish weekdays from weekends from taxi-pickup activities. The findings validate the effectiveness of our approach and suggest that graph signal processing tools may aid in urban planning and traffic forecasting.

Index Terms—Signal processing on graphs, signal localization, signal decomposition, dictionary learning

I. INTRODUCTION

Data is being generated at an unprecedented level from a diversity of sources, including social networks, biological studies and physical infrastructure. To analyze such complex data has led to the birth of signal processing on graphs [1], [2], [3], which generalizes classical signal processing tools to data supported on graphs; the data is the graph signal indexed by the nodes of the underlying graph. Recent additions to the toolbox include sampling of graph signals [4], [5], [6], [7], recovery of graph signals [8], [9], [10], representations for graph signals [11], [12], [13], [14], uncertainty principles on graphs [15], [16], graph-based transforms [17], [18], [19], [20] and community detection and clustering on graphs [21], [22], [23].

We consider three graph signal problems: localization, decomposition and dictionary learning, each of which builds on the previous one. Localization identifies an activated piece in a single graph signal, decomposition uses such pieces to represent/approximate a graph signal and dictionary learning identifies shared activated pieces in multiple graph signals.

Localization. The aim is to identify one set of connected nodes where the graph signal switches values—we call an activated piece. For example, given a graph signal in Figure 1(a), we are looking for an underlying activated piece as in Figure 1(b). This is relevant in many real-world applications from localizing virus attacks in cyber-physical systems to activity in brain connectivity networks, to traffic events in city road networks. As the original signal is noisy, this task is related but not equivalent to denoising. Denoising aims to obtain a noiseless graph signal (see Figure 1(c)), which is not necessarily localized. In this paper, we focus on localizing activated pieces in noisy piecewise-constant signals in which case localization is in fact equivalent to denoising.

(a) Signal. (b) Activated piece. (c) Noiseless signal.

Fig. 1: Signal localization on graphs. Given a signal (a), the aim is to identify an activated piece (b) while denoising aims to obtain a noiseless signal (c). When smooth background is ignored, localization is equivalent to denoising.

Decomposition. The aim here is to represent or approximate a signal as a linear combination of building blocks—activated pieces. For example, given a graph signal in Figure 2(a), we decompose it into two activated pieces (b) and (c), respectively.

(a) Signal. (b) Piece 1. (c) Piece 2.

Fig. 2: Signal decomposition on graphs. Given a signal (a), we aim to decompose it into two activated pieces (b) and (c).

Dictionary learning. The aim here is to learn a graph dictionary of one-piece atoms from multiple graph signals. For example, given three graph signals in Figures 3(a)—(c), we look for the underlying shared activated pieces as in Figures 3(d) and (e). This is relevant in many real-world applications where we look to localize common patterns in a large data volume. For example, on weekday evenings, the area near Penn Station in Manhattan is notoriously crowded. We can model the number of passing vehicles at all the intersections...
by a graph signal on the Manhattan street network and use dictionary learning techniques to analyze traffic activities at various moments and automatically localize the patterns.

Fig. 3: Dictionary learning on graphs. Given signals (a)–(c), dictionary learning aims to find shared activated pieces (d), (e).

**Related Work.** We now briefly review related work on anomaly detection, decomposition and dictionary learning.

a) **Anomaly detection:** This is a classical task that identifies items, events, or observations that do not conform to an expected pattern or other items in a dataset [24]. Anomaly detection has been widely studied in different fields, including signal processing, statistics, machine learning, computer vision and data mining. For example, tasks include detecting fake reviews on Yelp, purchased followers on Twitter and inflated trust on eBay [25], [26]. Within the context of graphs, much of the literature considers detecting smooth or piecewise-constant graph signals from signals corrupted by Gaussian noise [27], [28], [29], [30], [31]. A recent work [32] considers detecting piecewise-constant graph signals from Bernoulli noise. In this paper, we localize anomalies (activated pieces) on graphs.

b) **Decomposition:** Signal decomposition/presentation is at the heart of most signal processing tasks [33] as signals are represented or approximated as a linear combination of basic building blocks. Fourier analysis can be seen as spectral decomposition of signals into sinusoidal components and is widely used in communication systems and digital devices. Wavelet and multiresolution techniques decompose signals into components at different scales and are widely used in image and video compression. Within the context of graphs, a number of counterparts to classical transforms have been found: the graph Fourier transform [1], [5], the windowed graph Fourier transform [19] and the wavelet transform [18]. In a number of works, representations are considered based on the graph Fourier domain [1], [34], [35], designing graph filters in the graph Fourier domain and achieving representations that are particularly localized in the graph Fourier domain. Yet others consider representations based on the vertex domain [36], [37], [38], [14], decomposing a graph into multiple sub-graphs of various sizes and achieving representations that are particularly localized in the graph vertex domain. In this paper, we look at representations based on the graph vertex domain in a data-adaptive fashion.

c) **Dictionary learning:** Dictionary learning finds a sparse representation of the input data as a linear combination of basic building blocks. This task is widely studied in image processing, computer vision and machine learning. Popular algorithms include the method of optimal directions [38], K-SVD [39] and online dictionary learning [40]. Within the context of graphs, [12] considers learning a series of polynomial graph filters as a graph dictionary, emphasizing the representations on the graph Fourier domain. In this paper, we are concerned with learning a graph dictionary that is sensitive to local changes and uses the representations in the graph vertex domain.

**Contributions.** We start with a basic localization problem whose goal is to use a set of connected nodes to approximate a noisy one-piece graph signal with unit magnitude, propose an efficient and effective solver, and extend it to arbitrary magnitude. Based on this, we further consider signal localization, signal decomposition and dictionary learning on graphs. We conduct extensive experiments to validate the proposed methods. The results show that cut-based localization is good at localizing the ball-shaped class and path-based localization is good at localizing the elongated path class. We further apply our algorithms to analyze urban data, where our methods automatically detect events and explore mobility patterns across the city from a large volume of spatio-temporal data.

In summary, the main contributions of the paper are:

- A novel and efficient solver for graph signal localization.
- A novel and efficient solver for graph signal decomposition.
- A novel and efficient solver for graph dictionary learning.
- Extensive empirical studies of the proposed solvers on both simulated data and real data (Minnesota road and Manhattan street networks).

**Notation.** Consider an undirected graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the set of nodes and $\mathcal{E} = \{e_1, \ldots, e_M\}$ is the set of edges. A graph signal $\mathbf{x}$ maps the graph nodes $v_n$ to the signal coefficients $x_n \in \mathbb{R}$; in vector form, $\mathbf{x} = [x_1, x_2, \ldots, x_N]^T \in \mathbb{R}^N$. Let $C \subseteq \mathcal{V}$ be a subset of nodes; we represent it by the indicator vector,

$$1_C = \begin{cases} 1, & v_i \in C; \\ 0, & \text{otherwise}. \end{cases}$$

that is, $1_C$ is a graph signal with 1s in $C$ and 0s in the complement node set $C = \mathcal{V} \setminus C$. We say that the node set $C$ is activated. When the node set $C$ forms a connected subgraph, we call $C$ a piece and $1_C$ a one-piece graph signal.

Define a piecewise-constant graph signal as

$$x = \sum_{i=1}^{K} \mu_i 1_{C_i},$$

where each $C_i$ is a piece, $\mu_i$ is a constant and $K$ is the number of pieces.

**Outline of the paper.** We present the three problems, graph signal localization, decomposition and dictionary learning in Sections [1], [III] and [V] respectively. Section [V] concludes the paper and provides pointers to future research directions.
II. SIGNAL LOCALIZATION ON GRAPHS

In this section, we propose a solver for the localization problem, validating it by simulations.

Consider localizing an activated piece \( C \in \mathcal{C} \) (where \( \mathcal{C} \) is the set containing all the pieces) in a noisy, piecewise-constant graph signal

\[
x = \mu \mathbf{1}_C + \epsilon,
\]

where \( \mathbf{1}_C \) is the indicator function defined in (1), \( \mu \) is the signal strength and \( \epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{1}_N) \) is Gaussian noise. We consider two cases: \( \mu = 1 \) and \( \mu \) arbitrary.

Previous works formulate this as a detection problem via a scan statistic searching for a most probable anomaly set, which is similar to localizing an activated piece. However, such a method is either computationally inefficient or hindered by strong assumptions. For example, in [27], the authors analyze the theoretical performance of detecting paths, blobs and spatial temporal sets by exhaustive search, which is clearly inefficient. In [29], [31], [32], the authors aim to detect a node set with an arbitrarily-shaped activated piece. Since we do not restrict the size and shape of a piece, searching over the set containing all the pieces) in a noisy, piecewise-constant graph signal: activated nodes and inactivated nodes. This is similar to graph cuts: cutting a series of edges with minimum cost and obtaining two isolated components. The difference is that one optimizes edge weights in graph cuts, while (4) optimizes signal coefficients. Inspired by [29], [32], we add the number of edges connecting activated nodes and inactivated nodes to the objective function to induce a connected component. Let \( \Delta \in \mathbb{R}^{M \times N} \) be the graph difference operator (the oriented incidence matrix of \( G \)). When \( e_i \) is an edge that connects the \( j \)th node to the \( k \)th node \((j < k)\), the elements of the \( \ell \)th row of \( \Delta \) are

\[
\Delta_{i, \ell} = \begin{cases} 
1, & \ell = j; \\
-1, & \ell = k; \\
0, & \text{otherwise}.
\end{cases}
\]

A signal \( \Delta C_1 \in \mathbb{R}^M \) records the first-order differences within \( 1_C \). The number of edges connecting \( C \) and \( \bar{C} \) is

\[
\| \Delta C_1 \|_0 = \| A 1_C \|_1 - \| 1_C^2 A \mathbf{1}_C, \]

where \( A \) is the adjacency matrix. \( \| A 1_C \|_1 \) is the sum of the degrees of the nodes in \( C \) and \( 1_C^2 A 1_C \) is the inner connection of nodes within \( C \). When there are no edges connecting the activated nodes, \( \| \Delta C_1 \|_0 = \| A 1_C \|_1 \). When all the activated nodes are well connected and form a piece, \( 1_C^2 A 1_C \) is often large and \( \| \Delta C_1 \|_0 \ll \sum_{i \in C} |d_i| \) with \( d_i \) the degree of the \( i \)th node. Thus, minimizing \( \| \Delta C_1 \|_0 \) induces \( C \) to be a piece. We thus solve

\[
C_\lambda = \mathbf{P}_C \left( \arg \min_{C} \| x - 1_C \|_2^2 + \lambda \| \Delta C_1 \|_0 \right), \quad (6a)
\]

\[
C^*_{\text{cut}} = \arg \min_{C} \| x - 1_C \|_2^2. \quad (6b)
\]

Given a \( \lambda \), (6a) solves the regularized optimization problem and extracts the largest connected component. In (6b), we optimize over \( \lambda \) and obtain a solution. The previous hard thresholding solution (5) is a subcase of (6b) when we force \( \lambda = 0 \).

Let us look at the regularized problem \((6a)\) in more detail,

\[
\min_{C} \| x - 1_C \|_2^2 + \lambda \| \Delta C_1 \|_0 = \min_{t \in \{0,1\}^N} \| x - t \|_2^2 + \lambda \| \Delta t \|_0
\]

\[
= \min_{t \in \{0,1\}^N} \sum_{i=1}^N (x_i - t_i)^2 + \lambda \sum_{j \in \text{Nei}(i)} |t_i - t_j| \]

where \( \text{Nei}(i) \) is the neighborhood of the \( i \)th node. That is, we have \( t_i = 1 \) when

\[
(x_i - 1)^2 + \lambda \sum_{j \in \text{Nei}(i)} |x_j - 1| < \frac{\alpha}{2} + \frac{\lambda}{2} \sum_{j \in \text{Nei}(i)} |x_j|.
\]

The solution is

\[
(t^*_\text{cut})_i = \begin{cases} 
1, & x_i > \frac{1}{2} + \frac{\lambda}{2} (\alpha_0 - \alpha_1); \\
0, & \text{otherwise},
\end{cases}
\]

where the projection operator \( \mathbf{P}_C(C) \) extracts the largest connected component (piece) in a node set \( C \). Thus, this solver simply performs hard thresholding and then finds a connected component among the nodes with nonzero elements on it.

Cut-based localization. The key idea in (5) is to partition all the nodes into two categories according to a given graph signal: activated nodes and inactivated nodes. This is similar to graph cuts: cutting a series of edges with minimum cost and obtaining two isolated components. The difference is that one optimizes edge weights in graph cuts, while (4) optimizes signal coefficients. Inspired by [29], [32], we add the number of edges connecting activated nodes and inactivated nodes to the objective function to induce a connected component. Let \( \Delta \in \mathbb{R}^{M \times N} \) be the graph difference operator (the oriented incidence matrix of \( G \)). When \( e_i \) is an edge that connects the \( j \)th node to the \( k \)th node \((j < k)\), the elements of the \( \ell \)th row of \( \Delta \) are

\[
\Delta_{i, \ell} = \begin{cases} 
1, & \ell = j; \\
-1, & \ell = k; \\
0, & \text{otherwise}.
\end{cases}
\]

A signal \( \Delta C_1 \in \mathbb{R}^M \) records the first-order differences within \( 1_C \). The number of edges connecting \( C \) and \( \bar{C} \) is

\[
\| \Delta C_1 \|_0 = \| A 1_C \|_1 - \| 1_C^2 A \mathbf{1}_C,
\]

where \( A \) is the adjacency matrix. \( \| A 1_C \|_1 \) is the sum of the degrees of the nodes in \( C \) and \( 1_C^2 A 1_C \) is the inner connection of nodes within \( C \). When there are no edges connecting the activated nodes, \( \| \Delta C_1 \|_0 = \| A 1_C \|_1 \). When all the activated nodes are well connected and form a piece, \( 1_C^2 A 1_C \) is often large and \( \| \Delta C_1 \|_0 \ll \sum_{i \in C} |d_i| \) with \( d_i \) the degree of the \( i \)th node. Thus, minimizing \( \| \Delta C_1 \|_0 \) induces \( C \) to be a piece. We thus solve

\[
C_\lambda = \mathbf{P}_C \left( \arg \min_{C} \| x - 1_C \|_2^2 + \lambda \| \Delta C_1 \|_0 \right), \quad (6a)
\]

\[
C^*_{\text{cut}} = \arg \min_{C} \| x - 1_C \|_2^2. \quad (6b)
\]

Given a \( \lambda \), (6a) solves the regularized optimization problem and extracts the largest connected component. In (6b), we optimize over \( \lambda \) and obtain a solution. The previous hard thresholding solution (5) is a subcase of (6b) when we force \( \lambda = 0 \).

Let us look at the regularized problem \((6a)\) in more detail,

\[
\min_{C} \| x - 1_C \|_2^2 + \lambda \| \Delta C_1 \|_0 = \min_{t \in \{0,1\}^N} \| x - t \|_2^2 + \lambda \| \Delta t \|_0
\]

\[
= \min_{t \in \{0,1\}^N} \sum_{i=1}^N (x_i - t_i)^2 + \lambda \sum_{j \in \text{Nei}(i)} |t_i - t_j| \]

where \( \text{Nei}(i) \) is the neighborhood of the \( i \)th node. That is, we have \( t_i = 1 \) when

\[
(x_i - 1)^2 + \lambda \sum_{j \in \text{Nei}(i)} |x_j - 1| < \frac{\alpha}{2} + \frac{\lambda}{2} \sum_{j \in \text{Nei}(i)} |x_j|.
\]

The solution is

\[
(t^*_\text{cut})_i = \begin{cases} 
1, & x_i > \frac{1}{2} + \frac{\lambda}{2} (\alpha_0 - \alpha_1); \\
0, & \text{otherwise},
\end{cases}
\]
where $\alpha_1$ counts the neighbors of $x_i$ equaling to 1 and $\alpha_0$ counts the neighbors of $x_i$ equal to 0. Theoretically, (7) is a local thresholding. In other words, the value of each element depends on the values of its neighbors. Computationally, we can solve the regularized optimization problem in (6a) by using the Boykov-Kolmogorov algorithm [42, 43]. We call this cut-based localization.

**Path-based localization.** Cut-based localization is an efficient solver to localize activated nodes that can be easily separated from inactivated ones. The difficulty of separating these two classes is described by the cut number, which is good at promoting ball-shaped pieces, but not good at promoting elongated paths. For example, to solve (3), we introduced $\|\Delta 1_C\|_0$ in (6a) to promote fewer connections between activated nodes and inactivated ones. This is because when $C$ is a piece, $1^T_C A 1_C$ is large. When nodes in $C$ are fully connected, $C$ forms a ball-shaped piece (see Figures 2(a)–(b)) and $1^T_C A 1_C = |C|(|C| - 1)/2$; however, when nodes in $C$ are weakly connected, for example, when $C$ forms an elongated path, $1^T_C A 1_C = |C| - 1$. In other words, (6a) seems better suited to capturing ball-shaped pieces than elongated paths. To address elongated paths, we restrict $\|A 1_C\|_\infty$, the maximum degree of the nodes in the activated set to be at most 2 and solve the following optimization problem,

$$
\min_{C} \|x - 1_C\|_2^2, \quad \text{subject to } \|A 1_C\|_\infty \leq 2.
$$

In other words, the constraint requires that each activated node connect to at most two activated nodes, which promotes elongated paths. To solve this combinatorial problem efficiently, we relax it to a convex problem,

$$
t^*_\text{path} = \min_{t \in \mathbb{R}^N} \|x - t\|_2^2,
$$

subject to $\|A t\|_\infty \leq 2$ and $0 \leq t \leq 1$.

We then set various thresholds for $t^*_\text{path}$, extract the largest connected component, optimize over the threshold to obtain a solution,

$$
C_\lambda = P_C (1_{t^*_\text{path} \geq \lambda}),
C^*_\text{path} = \arg \min_{C_\lambda} \|x - 1_{C_\lambda}\|_2^2.
$$

We call this path-based localization.

**Combined solvers.** We can combine these two solvers by choosing the one with a smaller objective value. The final solution is

$$
C^* = \text{Loc}_1(x) = \arg \min_{C \in \{C^*_\text{cut}, C^*_\text{path}\}} \|x - 1_C\|_2^2,
$$

where $\text{Loc}_1(\cdot)$ is the operator that finds an activated piece with unit magnitude. This solution is not the global optimum of (7); it considers two typical subsets in the feasible set and combines a graph-cut solver with a quadratic programming problem, both of which are efficient. In Section II-B, we validate it empirically.

2) **Localization with unknown magnitude:** We next consider a more general case where the magnitude of the activated piece is unknown, (2), with the associated optimization problem (3).

We iteratively solve $C$ and $\mu$ until convergence; that is, given $C$, we optimize over $\mu$ and then given $\mu$, we optimize over $C$. In the $k$th iteration,

$$
\mu^{(k)} = \min_{\mu} \|x - \mu 1_{C^{(k)}}\|_2^2 = \frac{x^T 1_{C^{(k)}}}{1^T_{C^{(k)}} 1_{C^{(k)}}},
$$

$$
C^{(k)} = \text{Loc}_1(1 - \mu x).
$$

We obtain a pair of local optima by alternatively minimizing these two variables. We denote this solver by

$$
\mu^*, C^* = \text{Loc}_{\text{unknown}}(x).
$$

B. **Experimental Validation**

We test our localization solver on two graphs: the Minnesota road network and the Manhattan street network. On each graph, we consider localizing two classes of simulated graph signals with different activated sizes under various noise levels.

1) **Minnesota road network:** We model the Minnesota road network as a graph with intersections as nodes and road segments between intersections as edges. The graph includes 2,642 nodes and 3,342 undirected edges. We simulate two classes of one-piece graph signals: a ball-shaped class and an elongated path class.

**Ball-shaped class.** To generate a ball-shaped piece, we randomly choose one node as a node center and assign all the other nodes that are within $k$ steps of the node center to an activated node set, where the radius $k$ is either 5 or 10. Figures 2(a)–(b) show examples of ball-shaped pieces with radii $k = 5$ and 10 (left and right columns, respectively). The nodes in green (lighter) indicate the activated nodes and the nodes in black (darker) indicate the inactivated nodes. We aim to localize the activated pieces from noisy one-piece graph signals. We vary the noise variance $\sigma^2$ from 0.1 to 1 with interval 0.1 and at each randomly generate 1,000 noisy one-piece graph signals.

We measure the quality of the localization with two measures: the Hamming distance and the $F_1$ score. The Hamming distance is equivalent to the Manhattan distance between two binary graph signals, that is, it counts the total number of mismatches between the two,

$$
d_H(\hat{C}, C) = \|1_{\hat{C}} - 1_C\|_1 = |C| + |\hat{C}| - |C \cap \hat{C}|,
$$

where $C$ is the ground truth for the activated piece and $\hat{C}$ is the localized piece. The lower the Hamming distance the better the quality: the Hamming distance reaches its minimum value at $d_H(\hat{C}, C) = 0$.

The $F_1$ score is the harmonic mean of the precision and recall, and measures the matching accuracy,

$$
F_1(\hat{C}, C) = 2 \frac{\text{precision}(\hat{C}, C) \cdot \text{recall}(\hat{C}, C)}{\text{precision}(\hat{C}, C) + \text{recall}(\hat{C}, C)},
$$

where precision is the fraction of retrieved instances that are relevant (true positives over the sum of true positives and false positives), while recall is the fraction of relevant instances that are retrieved (true positive over the sum of true positives and false negatives) [44]. Thus, with true positives $|C \cap \hat{C}|$, sum
Fig. 4: Localizing ball-shaped pieces in the Minnesota road network as a function of the noise level using hard thresholding (blue dashed line), cut-based localization (red solid line), path-based localization (yellow-circle line) and local-set-based piecewise-constant (purple-square line). Cut-based localization provides the best performance.

The Hamming distance emphasizes the difference between two sets while the $F_1$ score provides the best performance.

We observe that: (1) Cut-based localization works significantly better than the other three approaches as expected; hard thresholding is a special case of cut-based localization, path-based localization is designed for capturing elongated paths and LSPC is a data-independent dictionary and cannot adapt the shape of its atom to the given piece; (2) The noise level influences the localization performance; as it grows, the $F_1$ score decreases and the Hamming distance increases. (3) The size of the activated piece influences the localization: localizing a piece with radius 10 is easier than localizing a piece with radius 5; this was also observed in [32].

**Elongated path class.** To generate an elongated path, we randomly choose two nodes as starting and ending nodes and compute the shortest path between these two. We look at two categories of path lengths: shorter than 15 and longer than 80. Figures 5(a)–(b) show examples of paths with lengths 15 and

The path length is measured by the geodesic distance between the two end nodes.
Fig. 6: Localizing (a) Central Park in the Manhattan street network as a function of the noise level from (b) its noisy version. (c) Activated piece obtained by cut-based localization with $F_1 = 0.81$ and $d_H = 220$. (d) Activated piece obtained by path-based localization with $F_1 = 0.16$ and $d_H = 617$. Cut-based localization outperforms path-based localization.

Fig. 7: Localizing (a) 5th Avenue in the Manhattan street network as a function of the noise level from (b) its noisy version. (c) Activated piece obtained by cut-based localization with $F_1 = 0.32$ and $d_H = 120$. (d) Activated piece obtained by path-based localization with $F_1 = 0.44$ and $d_H = 231$. Path-based localization outperforms cut-based localization.

91 (left and right columns, respectively). We aim to localize the activated pieces from noisy path graph signals. We vary the noise variance $\sigma^2$ from 0.1 to 1 with interval 0.1 and at each randomly generate 1,000 noisy path graph signals at each noise level. We measure the quality of the localization by the Hamming distance and $F_1$ score.

Figures 5(c)–(d) show the $F_1$ scores and Figures 5(e)–(f) the Hamming distances of localizing elongated-path pieces with lengths 15 and 91 (left and right columns, respectively) as functions of the noise variance. We compare the same four methods as in Figure 4: hard thresholding (5) (hard, blue dashed line), cut-based localization (6b) (cut number, red solid line), path-based localization (8) (path, yellow-circle line) and local-set-based piecewise-constant dictionary (LSPC, purple-square line).

We observe that: (1) When the path is short, cut-based localization and path-based localization provide similar performance; when the path is long, path-based localization outperforms cut-based localization. (2) The noise level influences the localization performance; as it grows, the $F_1$ score decreases and the Hamming distance increases. Moreover, cut-based localization is more robust to noise than the path-based localization. (3) The path length does not influence the localization performance. This is different from what we observed in the ball-shaped case. Localization of elongated paths is more susceptible to noise and we can only localize multiple sections instead of a single, long path. Since we only extract the largest connected component, we usually obtain one section of a path, which is equivalent to localizing a short path.

(4) Localizing a path is harder than localizing a ball-shaped piece. The performance drop in Figure 5 with added noise is much steeper than in Figure 4.

2) Manhattan street network: We model the Manhattan street network as a graph with intersections as nodes and city streets as edges. The graph includes 13,679 nodes and 17,163 undirected edges. We generate two classes of one-piece graph signals: Central Park and 5th Avenue. Central Park is considered a ball-shaped piece and 5th Avenue is considered a path. We then compare the results with those from the Minnesota road network to validate our conclusions.

Figures 6(a)–(d) show a piece activating the nodes in Central Park, a noisy version with noise variance $\sigma^2 = 1$, the activated
Figures 6(e)–(f) show the noise, cut-based localization still provides accurate localization, even when the graph signal is corrupted by a high level of noise. The activated piece provided by cut-based localization and the activated piece provided by path-based localization, respectively. The Hamming distances of LSPC and PCD are 162 and 141, respectively.

Figures 7(a)–(d) show a piece activating the nodes along 5th Avenue as a function of the noise level, respectively. We see that, when the noise level is low, cut-based localization significantly outperforms the other methods, which is consistent with what we observed with the Minnesota road network. LSPC is less sensitive to noise and slightly outperforms the other methods when the noise level is high. Since LSPC is data-independent, it chooses the most relevant predesigned atom to fit a noisy signal; on the other hand, as a data-adaptive method, cut-based localization designs an atom from the noisy signal and it fits noisy data well. Overall, the localization performance of LSPC highly depends on the shape of its atoms: when LSPC has a predesigned atom matching the ground truth, it fails to localize well.

Figures 7(e)–(f) show the F1 score and Hamming distance when localizing 5th Avenue as a function of the noise level, respectively. The results are averaged over 1,000 runs. We see that the path-based localization is sensitive to noise; when the noise level is low, it outperforms the other method; when the noise level is high, cut-based localization performs similarly. This is similar to what we observed with the Minnesota road network.

We also observe that localizing 5th Avenue is harder than localizing Central Park because the results in especially when the noise level is high, which is again consistent with what we saw in the Minnesota road network.

III. SIGNAL DECOMPOSITION ON GRAPHS

We now extend the localization discussion by considering multiple activated pieces making it a decomposition problem. We extend the solver of the localization problem in Section II to the decomposition problem and validate it through simulations.

Consider localizing $K$ activated pieces $C_i$, $i = 1, 2, \ldots, K$, in a noisy, piecewise-constant graph signal

$$x = \sum_{i=1}^{K} \mu_i 1_{C_i} + \epsilon,$$

where $1_{C_i}$ is the indicator function defined in (1), $K \ll N$, $C_i$ are connected and $\epsilon \sim \mathcal{N}(0, \sigma^2 1_N)$ is Gaussian noise. Our goal is to develop an algorithm to efficiently decompose such a graph signal into several pieces. The corresponding optimization problem is

$$\min_{\mu_i, C_i} \left\| x - \sum_{i=1}^{K} \mu_i 1_{C_i} \right\|_2^2, \quad \text{subject to } C_i \in \mathcal{C}. \quad (11)$$

A. Methodology

The goal now is not only to denoise or approximate a graph signal, which would only give $\sum_{i=1}^{K} \mu_i 1_{C_i}$. We aim to analyze the composition of a graph signal through decomposition; that is, we localize each piece and estimate its corresponding magnitude, yielding $\mu_i$ and $C_i$. Since a graph signal is decomposed into several one-piece components, we call this piecewise-constant decomposition.

To solve the optimization problem (11), we update each piece at a time by coordinate descent [45]. When we freeze the other variables, optimizing over $\mu_i$ and $C_i$ is equivalent to localizing with unknown magnitude in Section II-A2 because

$$x - \sum_{j \neq i} \mu_j 1_{C_j} = \mu_i 1_{C_i} + \epsilon$$
We see that LSPC fails to localize ZIP code 10030 (d) show the results provided by LSPC and PCD, respectively.

This noisy graph signal into two activated pieces. Figures 9(c)–

piecewise-constant decomposition (11) (PCD) to decompose

in Manhattan, including restaurant distribution and taxi-pickup

selected ZIP codes. (2) We decompose two real graph signals

combinations of selected ZIP codes with the goal to localize the

simulate a series of piecewise-constant graph signals by a linear

on the Manhattan street network through two tasks: (1) We

generate a piecewise-constant graph signal,

Localizing ZIP codes: There are 43 ZIP codes in Man-

this graph Fourier (A)

there is a one-piece graph signal. Thus, for each piece, we solve the

localization problem (3) by using the localization solver (10),





σ

Figure 9(a) shows a piecewise-constant signal generated by

Σ

µ

emes, but PCD

B. Experimental Validation

We validate the piecewise-constant decomposition solver on the Manhattan street network through two tasks: (1) We simulate a series of piecewise-constant graph signals by a linear combinations of selected ZIP codes with the goal to localize the selected ZIP codes. (2) We decompose two real graph signals in Manhattan, including restaurant distribution and taxi-pickup activity and analyze their composition.

1) Localizing ZIP codes: There are 43 ZIP codes in Manhattan. In each study, we randomly select two of those and generate a piecewise-constant graph signal,

\[ x = \mu_1 1_{\text{ZIP}_1} + \mu_2 1_{\text{ZIP}_2} + \epsilon, \]

where the signal strength is uniformly distributed \( \mu_k \sim \mathcal{U}(0.5, 1.5) \); node set \( \text{ZIP}_i \) indicates the nodes belonging to the selected ZIP code; and the noise \( \epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{I}) \) with \( \sigma^2 \) varying from 0.1 to 1 with interval 0.1. At each noise level, we generate 1,000 piecewise-constant graph signals. Figure 9(a) shows a piecewise-constant signal generated by ZIP codes 10022 and 10030 and (b) shows a noisy graph signal with noise variance \( \sigma^2 = 1 \). We use LSPC [14] and piecewise-constant decomposition (PCD) to decompose this noisy graph signal into two activated pieces. Figures 9(c)–(d) show the results provided by LSPC and PCD, respectively. We see that LSPC fails to localize ZIP code 10030, but PCD successfully localizes both ZIP codes and provides acceptable localization performance.

To quantify the localization performance, we again use the \( F_1 \) score and the Hamming distance. Figures 9(a)–(b) show the \( F_1 \) score and Hamming distance when localizing ZIP codes in Manhattan as a function of noise level. The results at each noise level are averaged over 1,000 runs. Again, we see that PCD provides significantly better performance than LSPC, but PCD is sensitive to noise, while LSPC is robust to noise.

2) Restaurant and taxi-pickups: We now analyze two real graph signals: restaurant density and taxi-pickup activity. We aim to understand their components through decomposition.

Figure 10(a) shows the restaurant density including the positions of 10,121 restaurants in Manhattan. We project each restaurant to its nearest intersection and count the number of restaurants at each intersection. We use piecewise-constant decomposition to decompose this graph signal. Figures 10(c)–(d) show the decomposition by using three and nice pieces, respectively. Through the decomposition, we can get a rough idea about the distribution of restaurants. For example, areas around East Village and Times Square have more restaurants. By using more pieces, we can obtain a finer resolution and a better approximation.

Figure 10(b) shows the approximation error where the \( x \)-axis is the number of expansion coefficients and the \( y \)-axis is the normalized mean square error. We compare four methods: nonlinear approximation by using graph Fourier basis based on the adjacency matrix (graph Fourier (A), yellow-square line), nonlinear approximation by using graph Fourier basis based on the graph Laplacian matrix [3] (graph Fourier (A), purple-square line), sparse coding based on local-set-based piecewise-constant graph dictionary [14] (LSPC, blue dashed line)) and piecewise-constant decomposition (PCD, red solid line).
Fig. 12: Both restaurant distribution and taxi-pickup activity on Friday nights activate the same areas around East Village and Times Square. The $F_1$ score between the two Times Square pieces is 0.65 and the $F_1$ score between the two East Village pieces is 0.59, indicating a high overlap.

We see that, by using the same number of expansion coefficients (pieces), piecewise-constant decomposition significantly outperforms the other methods and graph Fourier basis based on adjacency matrix [1]. In other words, piecewise-constant decomposition provides an effective data-adaptive and structure-related representation.

Figure 11(a) shows the taxi-pickup activity in Manhattan, on Friday, June 5th, at 11 pm. We project each taxi pickup to its nearest intersection and count the number of pickups at each intersection. We use piecewise-constant decomposition to decompose this graph signal. Figures 11(c)–(d) show the decomposition by using three and nine pieces, respectively. Through this decomposition, we get a rough idea about the distribution of taxi-pickup activity. For example, areas around East Village and Times Square are busier.

Figure 11(b) shows the approximation error. Similarly to the results for the restaurant distribution, we see that using the same number of expansion coefficients (pieces), piecewise-constant decomposition significantly outperforms the other methods and graph Fourier basis on the adjacency matrix [1] outperforms the one based on the graph Laplacian matrix [3].

An interesting observation (but not surprising) is that both the restaurant distribution and the taxi-pickup activity on Friday nights activate the areas around East Village and Times Square (see Figure 12). Each piece in Figure 12 comes from one of the three pieces in Figures 10(c) and 11(c); we observe that the corresponding pieces highly overlap. The $F_1$ score between the two Times Square pieces is 0.65 and the $F_1$ score between the two East Village pieces is 0.59. This indicates that the taxi-pickup activity on Friday nights is highly correlated with the restaurant distribution, reporting a well-understood pattern of urban lifestyle.

IV. DICTIONARY LEARNING ON GRAPHS

We now extend signal decomposition, where we find activated pieces from a single graph signal, to dictionary learning where the aim is to find activated pieces that are shared by multiple graph signals. In other words, we learn activated pieces as building blocks that are used to represent multiple graph signals. We extend the solver of the decomposition problem in Section III to the dictionary learning problem and apply it to mine traffic patterns in the Manhattan data set.

Consider a matrix of $L$ graph signals as columns,

$$X = DZ + E \in \mathbb{R}^{N \times L},$$

with $D = [1_{C_1} \ 1_{C_2} \ \cdots \ 1_{C_K}]$ the graph dictionary with a predefined number of $K$ one-piece atoms as columns, the coefficient matrix $Z \in \mathbb{R}^{K \times L}$ is sparse, $C_i$ are connected and $E_{i,j} \sim \mathcal{N}(0, \sigma^2)$ is Gaussian noise. We store all the activated pieces in $D$ as building blocks. Thus, each graph signal (column in $X$) is approximated by a linear combination of several pieces from $D$. We learn the pieces $C_i$ from $X$ by solving the following optimization problem,

$$\min_{Z,C_i} \|X - [1_{C_1} \ 1_{C_2} \ \cdots \ 1_{C_K}] Z\|^2_F, \quad (12)$$

subject to $C_i \in \mathcal{C}$ and $\|Z\|_{0,\infty} \leq S,$

where the sparsity level $S$ helps avoid overfitting and $\|Z\|_{0,\infty} = \max_{i=1,\ldots,N} \|z_i\|_0$ with $z_i$ the $i$th column in $Z$, which is the maximum number of the nonzero elements in each column.

A. Methodology

The goal here is to learn those pieces and the corresponding coefficients from a matrix of graph signals. It is also possible to use the decomposition techniques in the previous section to localize pieces from each graph signal and to store all the activated pieces into the dictionary, but this approach cannot reveal the correlations among graph signals. We use a learning approach to find common pieces in the graph signals as it allows the same piece to be repeatedly used as a basic building block to represent graph signals and reveals the correlations among them.

To solve (12), we update the dictionary and the coefficient matrix successively; that is, given the dictionary, we optimize over the coefficient matrix and then, given the coefficient matrix successively; that is, given the dictionary, we optimize over the coefficient matrix.

When updating the coefficient matrix, we fix $C_i$ and consider

$$\min_Z \|X - [1_{C_1} \ 1_{C_2} \ \cdots \ 1_{C_K}] Z\|^2_F, \quad (13)$$

subject to $\|Z\|_{0,\infty} \leq S,$

where $\|\cdot\|_F$ denotes the Frobenius norm. This is a common sparse coding problem, which can be efficiently solved by orthogonal matching pursuit [40].

When updating the graph dictionary, we fix $Z$ and consider

$$\min_{C_i} \|X - [1_{C_1} \ 1_{C_2} \ \cdots \ 1_{C_K}] Z\|^2_F, \quad (14)$$

subject to $C_i \in \mathcal{C}.$
We use a public dataset of taxi pickups in Manhattan \(^2\), which is particularly interesting because taxis are valuable sensors of city life \([47], [48], [49]\). The information associated with taxi pickups provides insight into economic activity, human behavior and mobility patterns. We use the dataset for 2015 and focus on the rush hour periods (6-8 pm). We accumulate taxi-pickup activities within an hour and obtain 1,095y (365 \times 13) graph signals. Each graph signal thus shows taxi-pickup activity at each intersection in Manhattan at a specific hour. We next mine taxi-pickup patterns through two tasks: detecting events in Manhattan and checking whether weekdays and weekends exhibit differences in traffic patterns.

1) Event detection: We consider two types of events: common events and special events. When a place is frequently or periodically crowded, we consider it a common event, which shows a natural traffic behavior. When a place is rarely crowded, we consider it a special event, which shows an anomaly in traffic behavior. Traffic accidents or holiday celebrations usually lead to special events. We are going to use the learned pieces and their corresponding coefficients to analyze the taxi-pickup activities during rush hours. We set the size of the dictionary to \(K = 500\) and its sparsity to \(S = 30\). This means that \(K = 500\) activated pieces are learned from \(L = 1,095\) graph signals of dimension \(N = 13,679\) (13,679 intersections in Manhattan) and each graph signal is approximated by at most 30 pieces. When a piece is used to represent a graph signal, it means that the corresponding area is particularly crowded at a specific time and we need this piece to capture this traffic information. On the other hand, when a piece is not used, it just means that the corresponding area is not particularly crowded compared to other areas; it does not necessarily mean that there are no taxi pickups in the corresponding area because those taxi pickups may activate other related pieces.

**Common events.** Common events show natural traffic behaviors, which are detected by frequently-used pieces. When a row in the coefficient matrix has many nonzero entries, the corresponding piece is frequently used to represent graph signals. The top five most frequently-used pieces learned from taxi-pickup activities during rush hours are shown in Figure 13. For example, Figure 13(b) shows that a small area around Penn Station is frequently crowded and we use Piece 2 to capture this information. We then check which day uses this piece. Figure 13(b) shows a histogram of the usage, where we see that Piece 2 is more frequently used on Sundays (as well as Fridays and Saturdays) during the entire 2015. This indicates

**B. Experimental Validation**

We use the proposed dictionary learning techniques \((13), (14)\) to mine traffic patterns in Manhattan. We use a public dataset of taxi pickups in Manhattan, which is particularly interesting because taxis are valuable...
Fig. 14: When is Penn Station particularly crowded compared to other places? (a) Precise locations of Piece 2. (b) Histogram indicating that Penn Station is significantly more crowded than other places during weekends.

(a) Piece 2 Penn Station. (b) Histogram.

Fig. 15: Inflation of the balloons the day before Macy’s Thanksgiving Parade is detected. 6-8 pm on Nov 25th activates the 466th piece, indicating locations around W 82nd Street are particularly crowded.

(a) Piece W 82nd Street. (b) Zoom-in plot.

(c) Taxi-pickup activity at 9 pm, Nov. 25th. (d) Approximation by 30 pieces from the dictionary.

that, compared to other places in Manhattan, Penn Station is particularly crowded during weekends. This is intuitive due to a large number of commuters going through Penn Station and visiting the nearby Madison Square Garden on weekends.

Special events. Special events show anomalous traffic behaviors, which are detected by the rarely-used pieces. When a row in the coefficient matrix has a few nonzero entries, the corresponding piece is rarely used to represent graph signals. Figure 15(a) shows the 466th piece in the graph dictionary activating the area around West 82nd street and (b) shows a zoom-in plot. This piece is only used three times during the entire year of 2015, between 6-8 pm on November 25th, the night before Thanksgiving Day, indicating that the area around West 82nd street was much more crowded compared to other areas on that night only. Figures 15(c)–(d) show the taxi-pickup activities at 9 pm on November 25th and its approximation by using the graph dictionary, respectively. We thus successfully detect the inflation of the balloons for the Macy’s Thanksgiving parade that happens on 77th and 81st streets between Columbus and Central Park West, purely from the taxi-pickup activity.

Figure 16(a) shows the 377th piece in the graph dictionary activating the area around Apple Store on 5th Avenue and (b) shows a zoom-in plot. This piece is only used one time during the entire year of 2015, that is, 8 pm on July 3rd, the night before Independence Day. This indicates that the area around Apple Store on 5th Avenue was much more crowded compared to other areas on that night only. Figures 16(c)–(d) show the taxi-pickup activities at 8 pm on July 3rd and its approximation by using the graph dictionary, respectively.

Fig. 16: 8 pm on July 3rd activates the 377th piece, indicating that the 5th Ave. Apple store area is particularly crowded.

2) What day is today in Manhattan?: Are traffic patterns on weekends different from traffic patterns on weekdays? We use the learned graph dictionary to answer this question.

Graph dictionary learning not only provides traffic-correlated pieces, but also extracts traffic-correlated features through approximation. Similarly to principal component analysis, graph-dictionary-based sparse representation corresponds to unsupervised learning; it reduces the dimension of the representation, thereby extracting key information; however, as a general method, principal component analysis is unaware of the graph structure, while graph-dictionary-based sparse representation extracts traffic-correlated features based on the graph structure. Since there are 500 pieces in the graph dictionary, we reduce the dimension of each graph signal from 13,679 to 500 and the corresponding 500 expansion coefficients are traffic-correlated
features.

Fig. 17: Classifying to which day a graph signal belongs. Graph-dictionary-based sparse representation significantly outperforms principal component analysis.

Now we consider a 7-class classification task: given a graph signal, we aim to identify to which day of the week it belongs by using traffic-correlated features. Figure 17 shows the classification accuracy as a function of training data, where the $x$-axis is the number of months used in training and the $y$-axis is the classification accuracy. For example, when the month is $x = 1$, we use $93 (31 \times 3)$ graph signals in January as training data and use the remaining 1002 graph signals as testing data. Since this task involves 7 classes, the classification accuracy of a random guess is $14.29\%$. We compare three methods: principal component analysis (PCA, blue dotted line), average value of each graph signal (Average, yellow-circle line) and the graph-dictionary-based sparse representation (gDictionary, red solid line). We see that graph-dictionary-based sparse representation significantly outperforms principal component analysis and the average value and the classification accuracy of graph-dictionary-based sparse representation increases as the number of training data grows.

We now fix January graph signals as training data and the graph signals in the remaining 11 months as testing data; Figure 18 shows classification confusion matrix. We see that it is relatively easy to tell apart whether the day is a weekday or a weekend day, as the two groups are not confused with each other. There is also relatively little confusion between a Saturday and a Sunday. In contrast, weekdays are easily confused with each other indicating similar traffic patterns.

We further conduct a series of pairwise classifications to verify whether traffic patterns on weekdays are different from traffic patterns on weekends. We still fix graph signals in January as training data and graph signals in the remaining 11 months as testing data, but we only consider distinguishing one weekday from another weekday at a time. Figure 19 shows the resulting pairwise classification accuracies. We see that the accuracy of telling Thursdays apart from Sundays is around $90\%$, but the accuracy of telling Mondays apart from Tuesdays is merely above random. It is significantly easier to distinguish Sundays from weekdays.

$^4$Given $c$ classes and confusion matrix $M \in \mathbb{R}^{c \times c}$, element $M_{i,j}$ is a count of samples that belong to class $i$ but are classified as class $j$. Perfect classification yields an identity confusion matrix.

Fig. 18: Classification confusion matrix using January graph signals as training data showing that it relatively easy to tell weekdays from weekends apart.

Fig. 19: Pairwise classifications showing that it is significantly easier to distinguishing Sundays from weekdays.

3) Discussion: We considered event detection and day-of-the-week classification mined from taxi-pickup activities in Manhattan. In event detection, we were able to successfully detect the locations and times of both common and special events by using the graph dictionary learned from a spatio-temporal taxi-pickup data volume. In day-of-the-week classification, we were able to significantly outperform principal component analysis and provide insight into the traffic patterns on weekdays and weekends by using the same graph dictionary. These results suggest that the proposed graph dictionary learning techniques are a promising tool for exploring mobility patterns in a large volume of spatio-temporal urban data, which may aid in urban planning and traffic forecasting.

V. Conclusions

We studied three critical problems allowing piecewise-constant graph signals to serve as a tool for mining large amounts of complex data: graph signal localization, graph signal decomposition and graph dictionary learning. We used piecewise-constant graph signals to model local information in the vertex-domain and showed that decomposition and dictionary learning are natural extensions of localization. For each of these three problems, we proposed a specific graph signal model, an optimization problem and an efficient solver.

We conducted extensive validation studies on diverse datasets. The results show that cut-based localization is good at localizing ball-shaped classes and path-based localization is good at localizing elongated path class. We also used proposed methods to analyze taxi-pickup activity in Manhattan in 2015 and showed that based on these, we can successfully detect
