Node Clustering of Time-Varying Graphs Based on Temporal Label Smoothness

Katsuki Fukumoto*, Koki Yamada*, and Yuichi Tanaka*†
* Tokyo University of Agriculture and Technology, Tokyo, Japan
† PRESTO, Japan Science and Technology Agency, Saitama, Japan

***Abstract*—We propose a node clustering method for time-varying graphs based on the assumption that the cluster labels are changed smoothly over time. Clustering is one of the fundamental tasks in machine learning, data mining, and signal processing. Although most existing studies focus on the clustering of nodes in static graphs, we often encounter time-varying graphs for time-series data, e.g., social networks, brain functional connectivity, and point clouds. In this paper, we formulate a clustering of nodes in time-varying graphs as an optimization problem based on spectral clustering, with a smoothness constraint of the node labels. Experiments on synthetic and real-world time-varying graphs are conducted to validate the effectiveness of the proposed approach.**

I. Introduction

Clustering is an important and fundamental technique in signal processing, machine learning, and data mining, to name a few [1]–[3]. Roughly speaking, clustering divides a chunk of data into multiple communities and reveals their structure. Features of the data points belonging to the same community are expected to have similar characteristics and vice versa. That is, the intra-community relationship is stronger than that for the inter-community. Such a relationship is often given by a graph, where nodes of the graph correspond to the data points, and the relationships among the data points are given by edges. Therefore, clustering nodes of the graph is an important and widely studied problem.

Many methods have been proposed for clustering nodes of a graph [4]–[9]. In this paper, we focus on unsupervised learning where it is assumed that we have a graph, but we do not have training data associated with it. The unsupervised node clustering of graphs can be classified into two approaches. One is the node-domain approach, where the nodes are clustered based on several node-wise features. Its examples include clustering based on a goodness-of-partition metric called modularity [6]. The other is the spectral-domain approach, where the node clustering is performed based on the spectral characteristics of graphs, e.g., the polarities of eigenvector elements [10]. Its representative method is the well-known spectral clustering (SC) [11]. Note that the node- and spectral-domain approaches are related to each other. For example, SC with normalized graph Laplacian is an approximation of the minimum cut problem [4], [5].

These previous node clustering methods are mostly designed for static graphs. However, we often encounter time-varying (TV) graphs, i.e., a set of graphs where each graph represents a relationship between nodes at a certain time slot [12], [13]. Examples of TV graphs include social networks [14] and brain functional connectivity [15]. The clustering of TV graphs is a crucial problem since TV graphs are found in many applications. However, there are few clustering methods for clustering TV graphs, and most algorithm are designed in an ad-hoc manner [16]–[20]. For example, existing methods have a dependency on the initial clustering result or resulting cluster labels often fluctuate.

In this paper, we propose a clustering method for TV graphs for capturing smooth temporal evolution and noise robustness. We realize this by using the assumption that cluster labels in each node are TV but smoothly changed over time. Our formulation is based on SC, and we add a regularization term on the temporal variation of cluster labels. By solving the optimization problem, the node features on each node are estimated and utilized for clustering. Through experiments with synthetic and real-world data, our proposed method outperforms clustering for static graphs.

The rest of the paper is organized as follows. Section II presents a brief review of spectral graph theory. Section III describes related works for node clustering. The proposed method is described in Section IV. Section V provides experimental results with synthetic and real-world data. Finally, we conclude the paper in Section VI.

II. Review of Spectral Graph Theory

In this paper, we consider a weighted undirected graph $\mathcal{G} = (V, E, \mathbf{W})$, where $V := \{v_0, v_1, \dots\}$ is a set of nodes, E is a set of edges, and $\mathbf{W} \in \mathbb{R}^{N \times N}$ is a weighted adjacency matrix. The number of nodes is given by $N = |V|$. Each element of the weighted adjacency matrix is defined as

$$[\mathbf{W}]_{mn} = \begin{cases} w_{mn} & \text{if } v_m \text{ and } v_n \text{ are connected,} \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

That is, $w_{mn} \geq 0$ is the weight of the edge between v_m and v_n. In addition, the degree of the ith node d_i is defined as

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

$\mathbf{D} := \mathrm{diag}(d_0, d_1, \dots)$ is called a degree matrix. Using $\mathbf{D}$ and $\mathbf{W}$, a graph Laplacian is given by

$$\mathbf{L} = \mathbf{D} - \mathbf{W}. \tag{3}$$

If we consider $\mathbf{f} \in \mathbb{R}^N$ as a signal on a graph, i.e., features on the nodes, the quadratic form of the graph Laplacian is given by

$$\mathbf{f}^\top \mathbf{L}\mathbf{f} = \frac{1}{2}\sum_{m,n=1}^{N} w_{mn}\left(f_m - f_n\right)^2, \tag{4}$$

where f_m and f_n are the signal values at v_m and v_n, respectively. In fact, (4) represents the smoothness of the signal $\mathbf{f}$ on $\mathcal{G}$.

III. Related Work

In this section, we review existing works for node clustering of graphs.

A. Spectral Clustering

Here, we describe the formulation of SC for static graphs [10]. Since the algorithm of SC can help understand our formulation, we present its details.

The goal of clustering nodes in $\mathcal{G}$ is to divide them so that they are strongly connected within the same cluster and are weakly connected between different clusters.

First, let us define the strength of connection between clusters. For a set of nodes $A \subset V$ and $\bar{A} \subset V \setminus A$, the connection between A and $\bar{A}$ is defined as

$$c(A, \bar{A}) := \sum_{i\in A, j\in \bar{A}} w_{ij}. \tag{5}$$

For a given K, clustering should be performed by choosing a partition of $A_1, ..., A_K$ which minimizes

$$\mathrm{Cut}\left(A_1, \ldots, A_K\right) := \frac{1}{2}\sum_{i=1}^{K} c\left(A_i, \bar{A}_i\right). \tag{6}$$

We often need to balance the number of nodes in $|A_i|$ to avoid very small clusters. For this purpose, RatioCut [10], [21] is proposed. Its cost function is defined as follows:

$$\mathrm{RatioCut}\left(A_1, \ldots, A_K\right) := \sum_{i=1}^{k} \frac{\mathrm{Cut}\left(A_i, \bar{A}_i\right)}{|A_i|}. \tag{7}$$

Hereafter, we assume $K = 2$, i.e., the number of clusters is two for simplicity. For RatioCut, the current problem to be solved is formulated as follows:

$$\min_{A\subset V} \mathrm{RatioCut}(A, \bar{A}). \tag{8}$$

This problem is combinatorial and NP-hard [22]. Therefore, many relaxation methods to solve (8) have been considered [23]–[25].

First, we define a vector $\mathbf{c} \in \mathbb{R}^N$ as follows:

$$c_i = \begin{cases} \sqrt{|\bar{A}|/|A|} & \text{if } v_i \in A, \\ -\sqrt{|A|/|\bar{A}|} & \text{if } v_i \in \bar{A}. \end{cases} \tag{9}$$

If we can find $\mathbf{c}$, clustering can be immediately done by using the polarity of $\mathbf{c}$. In fact, RatioCut can be represented using the graph Laplacian of $\mathcal{G}$ based on (9). This is because the Laplacian quadratic form (4) is represented as follows by $\mathbf{c}$:

$$\begin{aligned}
\mathbf{c}^\top \mathbf{L}\mathbf{c} =& \frac{1}{2}\sum_{i,j=1}^{N} w_{ij}\left(c_i - c_j\right)^2 \\
=& \frac{1}{2}\sum_{i\in A, j\in \bar{A}} w_{ij}\left(\sqrt{\frac{|\bar{A}|}{|A|}} + \sqrt{\frac{|A|}{|\bar{A}|}}\right)^2 \\
&+ \frac{1}{2}\sum_{i\in \bar{A}, j\in A} w_{ij}\left(-\sqrt{\frac{|\bar{A}|}{|A|}} - \sqrt{\frac{|A|}{|\bar{A}|}}\right)^2 \\
=& \mathrm{cut}(A, \bar{A})\left(\frac{|\bar{A}|}{|A|} + \frac{|A|}{|\bar{A}|} + 2\right) \\
=& \mathrm{cut}(A, \bar{A})\left(\frac{|A| + |\bar{A}|}{|A|} + \frac{|A| + |\bar{A}|}{|\bar{A}|}\right) \\
=& |V| \cdot \mathrm{RatioCut}(A, \bar{A}).
\end{aligned}$$

Note that $\mathbf{c}$ satisfies

$$\begin{aligned}
\sum_{i=1}^{N} \mathbf{c}_i &= \sum_{i\in A}\sqrt{\frac{|\bar{A}|}{|A|}} - \sum_{i\in \bar{A}}\sqrt{\frac{|A|}{|\bar{A}|}} \\
&= |A|\sqrt{\frac{|\bar{A}|}{|A|}} - |\bar{A}|\sqrt{\frac{|A|}{|\bar{A}|}} = 0
\end{aligned} \tag{10}$$

and

$$\|\mathbf{c}\|_2^2 = \sum_{i=1}^{N} c_i^2 = |A|\frac{|\bar{A}|}{|A|} + |\bar{A}|\frac{|A|}{|\bar{A}|} = |\bar{A}| + |A| = N. \tag{11}$$

These result in that $\mathbf{c}$ is orthogonal to $\mathbf{1}$ and $\|\mathbf{c}\|_2^2 = N$.

As a result, (8) can be rewritten as follows:

$$\begin{aligned}
&\min_{\mathbf{c}\in\mathbb{R}^N} \mathbf{c}^\top \mathbf{L}\mathbf{c} \\
&\text{subject to } \mathbf{c} \perp \mathbf{1},\ \mathbf{c}_i \text{ in (9)}, \|\mathbf{c}\|_2^2 = N.
\end{aligned} \tag{12}$$

However, it is still combinatorial and NP-hard because the values in $\mathbf{c}$ are fixed as in (9). To approximately solve (12), the condition on $\mathbf{c}$ is relaxed to have arbitrary values. This leads to the following optimization problem.

$$\min_{\mathbf{c}\in\mathbb{R}^N} \mathbf{c}^\top \mathbf{L}\mathbf{c} \quad \text{subject to } \mathbf{c} \perp \mathbf{1},\ \|\mathbf{c}\|_2^2 = N. \tag{13}$$

It is well known that the solution of (13) is given by *Fiedler vector*, the eigenvector corresponding to the second smallest eigenvalue of $\mathbf{L}$. After solving (13), the cluster labels can be obtained by performing K-means clustering to $\mathbf{c}$.

B. Clustering of TV Graphs

TV graph clustering can be classified into three main categories [26]. The first method is to simply perform static clustering at each time and the cluster labels obtained at time t are arranged to correspond to the cluster labels obtained at time $t-1$ [16]. This does not take into account the temporal evolution.

The second method is to iteratively compute the clusters of the graph [17], [19], [20]. It computes the clusters at the

current time by considering the clusters obtained at one or more earlier time slots. This approach aims to reflect the temporal dependency in clustering, however, the initial cluster at $t = 0$ highly affects the resulting TV clusters.

The third method is to transform the TV graph into a single network and then perform clustering for static graphs [18]. However, this does not properly reflect temporal evolution like the first approach.

In contrast to existing approaches, our method obtains clusters of the entire TV graphs by solving an optimization problem.

IV. Proposed Method

In this section, we formulate node clustering of TV graphs based on the label smoothness. First, we formulate a clustering problem of TV graphs. Second, its optimization algorithm is introduced.

A. Formulation

As mentioned previously, SC methods are mostly designed for static graphs, therefore, it is difficult to perform node clustering taking into account temporal relationships. To tackle the problem, we reformulate node clustering for TV graphs.

First of all, we assume that the cluster labels change smoothly over time. This is reasonable as long as the time-series data are obtained with a sufficiently high sampling frequency. Here, we assume that the number of clusters is $K = 2$. It is relaxed later in Section IV-C.

Suppose that a set of graphs $\{\mathcal{G}_t\}_{t=1}^T$ and its associated graph Laplacians $\{\mathbf{L}_t\}$ are given, where t is the time index. Based on (13), clustering of nodes in $\{\mathcal{G}_t\}$ can be formulated as the following problem:

$$\min_{\mathbf{c}_t \in \mathbb{R}^N} \mathbf{c}_t^\top \mathbf{L}_t \mathbf{c}_t \text{ subject to } \mathbf{c}_t \perp \mathbf{1}, \|\mathbf{c}_t\|_2^2 = N. \tag{14}$$

This is a straightforward extension of SC on static graphs. However, (14) only considers variations of the node labels in each time slot. This may lead to that node labels in $\mathcal{G}_t$ and $\mathcal{G}_{t-1}$ are significantly different. However, they should be similar to each other.

The smoothness of node labels can be cast into their small variations, i.e., $\|\mathbf{c}_t - \mathbf{c}_{t-1}\|$ is small. Therefore, we add the ℓ_1 constraints of $\mathbf{c}_t$ onto (14) as follows:

$$\begin{aligned} &\min_{\mathbf{c}_t \in \mathbb{R}^N} \sum_{t=1}^T \mathbf{c}_t^\top \mathbf{L}_t \mathbf{c}_t + \alpha \sum_{t=2}^T \|\mathbf{c}_t - \mathbf{c}_{t-1}\|_1 \\ &\text{subject to } \mathbf{c}_t \perp \mathbf{1}, \|\mathbf{c}_t\|_2^2 = N, \end{aligned} \tag{15}$$

where $\alpha \in \mathbb{R}_{\geq 0}$ is a parameter.

We further rewrite (15) by introducing the concatenated label vector $\mathbf{c} = [\mathbf{c}_1^\top, \mathbf{c}_2^\top, \ldots, \mathbf{c}_T^\top]^\top$:

$$\min_{\mathbf{c} \in \mathbb{R}^{NT}} \mathbf{c}^\top \mathbf{L} \mathbf{c} + \alpha \|\mathbf{\Phi c}\|_1 \quad \text{subject to } \mathbf{c}_t \perp \mathbf{1}, \|\mathbf{c}_t\|_2^2 = N, \tag{16}$$

where $\mathbf{L} := \operatorname{diag}(\mathbf{L}_1, \ldots, \mathbf{L}_T)$ and $\mathbf{\Phi}$ is a linear operator satisfying $\mathbf{\Phi c} = \mathbf{c} - \hat{\mathbf{c}}$, in which $\hat{\mathbf{c}} = [\mathbf{c}_1^\top, \mathbf{c}_1^\top, \mathbf{c}_2^\top, \ldots, \mathbf{c}_{T-1}^\top]^\top$.

In the following subsection, we present an algorithm to solve (16).

B. Optimization

We consider to solve (16) based on the concept of the primal-dual splitting (PDS) algorithm [27]. The PDS algorithm solves a problem in the following form:

$$\min_{\mathbf{c}} f_1(\mathbf{c}) + f_2(\mathbf{c}) + f_3(\mathbf{Mc}), \tag{17}$$

where f_1 is a differentiable convex function with the β-Lipschitzian gradient ∇f_1 for some $\beta > 0$; f_2 and f_3 are proper lower semicontinuous convex functions which are proximable; and $\mathbf{M}$ is a linear operator.

To solve (16) using PDS, we transform (16) into its applicable form. First, we rewrite (16) as follows by introducing indicator functions:

$$\min_{\mathbf{c} \in \mathbb{R}^{NT}} \frac{1}{2} \mathbf{c}^\top \mathbf{L} \mathbf{c} + \alpha \|\mathbf{\Phi c}\|_1 + \sum_{j=1}^T \iota_{A_{\mathbf{1}\epsilon}}([\mathbf{c}]_j) + \iota_{B_N}([\mathbf{c}]_j), \tag{18}$$

where $[\mathbf{c}]_j$ is the jth chunk in $\mathbf{c}$. The indicator functions $\iota_{A_{\mathbf{1}\epsilon}}$ and ι_{B_N} are defined as follows:

$$\iota_{A_{\mathbf{1}\epsilon}}(\mathbf{x}) = \begin{cases} 0 & \text{if } \mathbf{x} \in A_{\mathbf{1}\epsilon}, \\ \infty & \text{otherwise}, \end{cases} \tag{19}$$

$$\iota_{B_N}(\mathbf{x}) = \begin{cases} 0 & \text{if } \mathbf{x} \in B_N, \\ \infty & \text{otherwise}, \end{cases} \tag{20}$$

where

$$A_{\mathbf{1}\epsilon} = \{\mathbf{x} \in \mathbb{R}^N | |\mathbf{x}^\top \mathbf{1}| \leq \epsilon\}, \tag{21}$$

$$B_N = \{\mathbf{x} \in \mathbb{R}^N | \|\mathbf{x}\|_2^2 - N = 0\}. \tag{22}$$

Then, (18) is split into the following PDS-applicable forms:

$$\begin{aligned} f_1(\mathbf{c}) &:= \mathbf{c}^\top \mathbf{L} \mathbf{c} \quad \text{with } \beta = \lambda_{\max}(\mathbf{L}), \\ f_2(\mathbf{c}) &:= \sum_{j=1}^T \iota_{B_N}([\mathbf{c}]_j), \\ f_3(\mathbf{d}) &:= \sum_{j=1}^T \iota_{A_{\mathbf{1}\epsilon}}([\mathbf{d}_1]_j) + \alpha \|\mathbf{d}_2\|_1, \\ \mathbf{M} &= \begin{bmatrix} \mathbf{I} \\ \mathbf{\Phi} \end{bmatrix}, \end{aligned}$$

where $\lambda_{\max}(\mathbf{L})$ is the maximum eigenvalue of $\mathbf{L}$, and $\mathbf{d} := \mathbf{Mc} = [\mathbf{d}_1^\top, \mathbf{d}_2^\top]^\top$ is the dual variable.

The proximal operator of f_2 is given by

$$\operatorname{prox}_{\gamma \iota_{B_N}}([\mathbf{z}]_j) = \sqrt{\frac{N}{[\mathbf{z}]_j^\top [\mathbf{z}]_j}} [\mathbf{z}]_j. \tag{23}$$

Moreover, the proximal operator of f_3 is calculated by dividing it into two terms. The proximal operator of its first term,

$\sum_{j=1}^{T} \iota_{A_{\mathbf{1}\epsilon}}([\mathbf{d}_1]_j)$, is given by

$$\mathrm{prox}_{\gamma\iota_{A_{\mathbf{1}\epsilon}}}([\mathbf{z}]_j) = \begin{cases} [\mathbf{z}]_j - \frac{[\mathbf{z}]_j^\top \mathbf{1} - \epsilon}{[\mathbf{z}]_j^\top [\mathbf{z}]_j} & \text{if } [\mathbf{z}]_j^\top \mathbf{1} > \epsilon, \\ [\mathbf{z}]_j - \frac{[\mathbf{z}]_j^\top \mathbf{1} + \epsilon}{[\mathbf{z}]_j^\top [\mathbf{z}]_j} & \text{if } [\mathbf{z}]_j^\top \mathbf{1} < -\epsilon, \\ [\mathbf{z}]_j & \text{otherwise.} \end{cases} \quad (24)$$

Furthermore, the proximal operator of $\|\cdot\|_1$ is known to be the element-wise soft-thresholding operation [28]:

$$\left[\mathrm{prox}_{\gamma\|\cdot\|_1}(\mathbf{z})\right]_i = \mathrm{sgn}(z_i)\max\{0, |z_i| - \gamma\}. \quad (25)$$

The details of the algorithm are shown in Algorithm 1 where e is a small real value.

Algorithm 1 TV Clustering with Label Smoothness

Input: $\mathbf{c}^{(0)}, \mathbf{d}_1^{(0)}, \mathbf{d}_2^{(0)}$
Output: $\mathbf{c}^{(i)}$
  while $\|\mathbf{c}^{(i+1)} - \mathbf{c}^{(i)}\|/\|\mathbf{c}^{(i)}\| > e$ **do**
    $\mathbf{c}^{(i+1)} = \mathrm{prox}_{\gamma_1 \iota_{B_N(\cdot)}}(\mathbf{c}^{(i)} - \gamma_1(\mathbf{L}\mathbf{c}^{(i)} + \mathbf{d}_1^{(i)} + \mathbf{\Phi}^\top \mathbf{d}_2^{(i)}))$
    $\mathbf{d}_1^{(i+1)} = \mathrm{prox}_{\gamma_2 \iota_{A_{\mathbf{1}\epsilon}}(\cdot)^*}(\mathbf{d}_1^{(i)} + \gamma_2(2\mathbf{c}^{(i+1)} - \mathbf{c}^{(i)}))$
    $\mathbf{d}_2^{(i+1)} = \mathrm{prox}_{\gamma_2 \|\cdot\|_1^*}(\mathbf{d}_2^{(i)} + \gamma_2 \Phi(2\mathbf{c}^{(i+1)} - \mathbf{c}^{(i)}))$
  end while

Note that B_N in (22) is a nonconvex set, and therefore, the entire problem becomes nonconvex. Although we observed the algorithm works well in practice, extending it to a convex optimization is left for future work.

C. Extension to Arbitrary Number of Clusters

In the previous subsection, we assume $K = 2$. However, the number of clusters is often greater than two. In this subsection, we describe additional formulations for clustering graphs into arbitrary K.

To split $\{\mathcal{G}_t\}$ into multiple clusters, we need to have $(K-1)$ $\mathbf{c}_t$'s. Suppose that $\{\mathbf{c}_t^{(1)}, \ldots, \mathbf{c}_t^{(\ell-1)}\}$, $\ell - 1 < K$, is obtained before calculating the ℓth cluster vector $\mathbf{c}_t^{(\ell)}$. With the spirit of SC, we need to solve the following problem:

$$\begin{aligned} \min_{\mathbf{c}_t \in \mathbb{R}^N} & \sum_{t=1}^{T} \mathbf{c}_t^\top \mathbf{L}_t \mathbf{c}_t + \alpha \sum_{t=2}^{T} \|\mathbf{c}_t - \mathbf{c}_{t-1}\|_1 \\ \text{subject to} \quad & \mathbf{c}_t \perp \mathbf{1}, \mathbf{c}_t \perp \{\mathbf{c}_t^{(1)}, \ldots, \mathbf{c}_t^{(\ell-1)}\}, \|\mathbf{c}_t\|_2^2 = N. \end{aligned} \quad (26)$$

The indicator function $\iota_{A_{\mathbf{1}\epsilon}}(\cdot)$ in (19) is redefined as follows so that it can be computed for any vectors:

$$\iota_{A_{\mathbf{v}\epsilon}}(\mathbf{x}) = \begin{cases} 0 & \text{if } \mathbf{x} \in A_{\mathbf{v}\epsilon}, \\ \infty & \text{otherwise,} \end{cases} \quad (27)$$

where

$$A_{\mathbf{v}\epsilon} = \{\mathbf{x} \in \mathbb{R}^N | |\mathbf{x}^\top \mathbf{v}| \le \epsilon\}.$$

The proximal operator of $\iota_{A_{\mathbf{1}\epsilon}}(\cdot)$ is given by

$$\mathrm{prox}_{\gamma\iota_{A_{\mathbf{v}\epsilon}}}([\mathbf{z}]_j) = \begin{cases} [\mathbf{z}]_j - \frac{[\mathbf{z}]_j^\top \mathbf{v} - \epsilon}{[\mathbf{z}]_j^\top [\mathbf{z}]_j} & \text{if } [\mathbf{z}]_j^\top \mathbf{v} > \epsilon, \\ [\mathbf{z}]_j - \frac{[\mathbf{z}]_j^\top \mathbf{v} + \epsilon}{[\mathbf{z}]_j^\top [\mathbf{z}]_j} & \text{if } [\mathbf{z}]_j^\top \mathbf{v} < -\epsilon, \\ [\mathbf{z}]_j & \text{otherwise.} \end{cases} \quad (28)$$

The algorithm to solve (26) is similar to Algorithm 1. We first determine $\{\mathbf{c}_t^{(1)}\}$ by Algorithm 1, and the other $\{\mathbf{c}_t^{(\ell)}\}$'s are sequentially computed.

V. Experiments

In this section, we evaluate the performance of the proposed method through experiments using synthetic and real-world data.

A. Synthetic Data

We synthesize TV graphs based on the stochastic block model (SBM) [29]. SBM is a random graph model where the intra- and inter-cluster edges are generated randomly according to the predefined edge probabilities. The intra-cluster edge probability is usually larger than that for inter-clusters.

In this experiment, the number of clusters and that of frames are set to $K = 3$ and $T = 100$, respectively. Each cluster has 50 nodes at $t = 1$. In short, the number of all nodes $N = 150$. For all t, the inter- and intra-cluster edge connection probabilities are set to 0.5 and 0.3, respectively. For every 10 frame, clusters of all nodes are reassigned with a probability of 5%.

1) Accuracy Measure: For calculating the clustering accuracy, we employ the method in [30]. First, we create a matrix $\mathbf{P} \in \mathbb{R}^{N \times N}$ that indicates clusters as follows:

$$[\mathbf{P}]_{mn} = \begin{cases} 1 & \text{if } v_m \text{ and } v_n \text{ belong to the same cluster,} \\ 0 & \text{otherwise.} \end{cases} \quad (29)$$

Second, by comparing $\mathbf{P}$ of the ground-truth graph and its estimation, we calculate the ratio of the correct classification as follows:

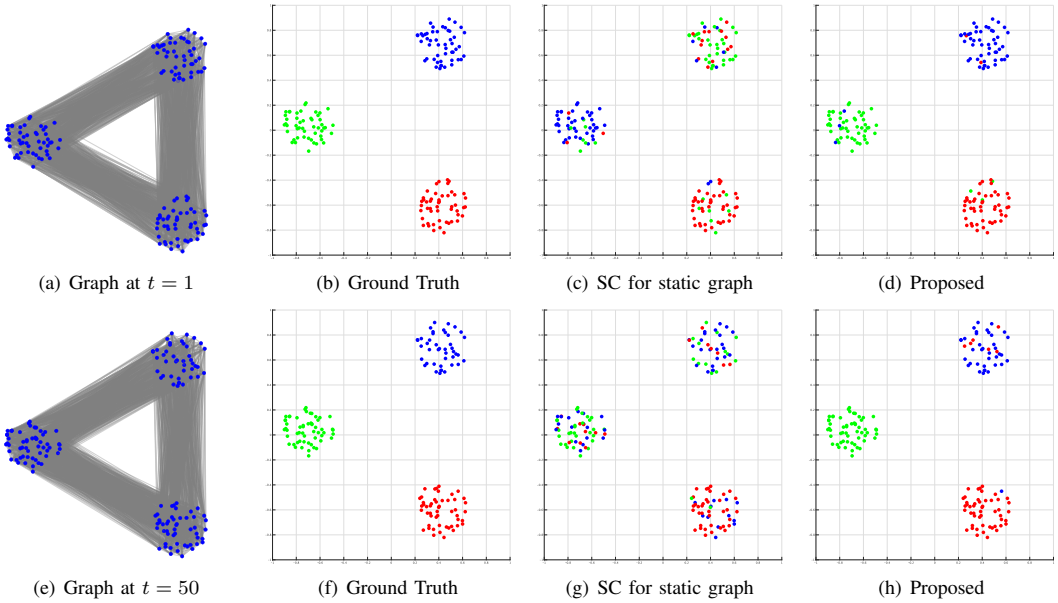
$$\text{accuracy} = \frac{\sum_{i,j=1}^{N} \text{count}_{ij} - N}{N^2 - N}, \quad (30)$$

$$\text{count}_{ij} = \begin{cases} 1 & \text{if } [\mathbf{P}]_{ij} = [\tilde{\mathbf{P}}]_{ij}, \\ 0 & \text{otherwise,} \end{cases} \quad (31)$$

where $\tilde{\mathbf{P}}$ is the same as (29) but is created from the estimated clusters.

2) Results: We compare the experimental results of the proposed method with SC for static graphs. Fig. 1 shows the clustering results where node colors indicate the cluster labels. Fig. 2 also compares the accuracy of clustering in (30) as functions of t.

As shown in Figs. 1(c) and 1(g), SC for static graphs fails to extract accurate clusters because edge connection probabilities of intra- and inter-clusters are close to each other. In contrast, the proposed method estimates the correct clusters because of the constraint of the temporal label evolution.



(a) Graph at $t = 1$ (b) Ground Truth (c) SC for static graph (d) Proposed

(e) Graph at $t = 50$ (f) Ground Truth (g) SC for static graph (h) Proposed

Fig. 1. Experimental results of the synthetic data. Top row: $t = 1$. Bottom row: $t = 50$.

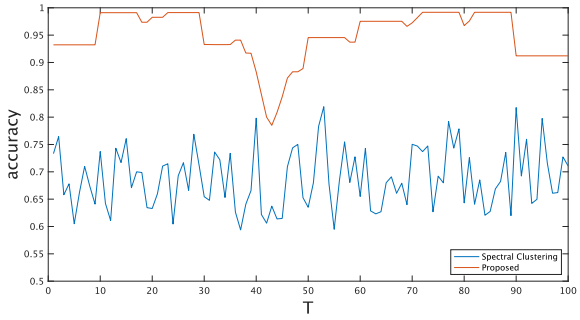


Fig. 2. Clustering accuracy according to t.

In Fig. 2, the accuracy of the proposed method is consistently higher than that of the SC for static graphs. These results indicate that the regularization of the temporal evolution of cluster labels is effective for TV graph clustering.

B. Real-World Data

For an experiment using real-world data, we perform clustering of dynamic point clouds. We use a dynamic point cloud in the dataset of [31]. Its examples are shown in Figs. 3(a) and (e). First, it is randomly downsampled to $N = 251$ points and we use $T = 200$ consecutive frames in it. Then, the graph is constructed by k-nearest neighbor with $k = 9$. For the experiment, random edges are added between nodes for studying robustness against incorrect edges. At $t = 1$, no random edges are added and the probability of random edges increases by 10^{-4} per frame until $t = 100$, and then it decreases by 10^{-4} from $t = 101$ to $t = 200$. The number of clusters is set to $K = 5$.

Fig. 3 shows the experimental results for the dynamic point clouds. Similar to the synthetic data, the node color indicates a cluster label.

As shown in Figs. 3(d) and (h), nodes within the same body parts are estimated as one cluster by the proposed method even in noisy situation. In contrast, SC for static graphs "mixes" several body parts in one cluster. This results in the effectiveness of the proposed method against the static clustering approach.

VI. Conclusions

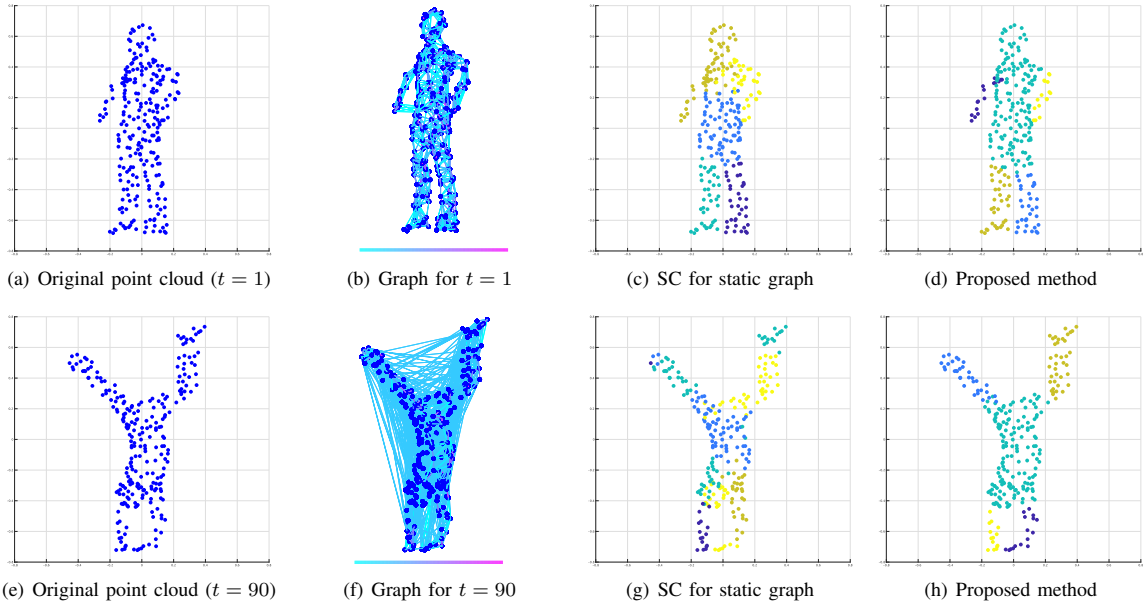
In this paper, we propose a node clustering method for TV graphs. Our method assumes that the node labels are smoothly changed over time. We formulate an optimization problem based on SC with a regularization term for the label temporal evolution. Through the experiments, it is observed that the proposed method extracts more accurate clusters than clustering for static graphs, even in noisy situations.

Acknowledgments

This work is supported in part by JST PRESTO under grant JPMJPR1935 and JSPS KAKENHI under grant 20H02145.

References

[1] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on neural networks*, vol. 16, no. 3, pp. 645–678, 2005.
[2] A. Saxena, M. Prasad, A. Gupta, N. Bharill, O. P. Patel, A. Tiwari, M. J. Er, W. Ding, and C.-T. Lin, "A review of clustering techniques and developments," *Neurocomputing*, vol. 267, pp. 664–681, 2017.
[3] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long, "A survey of clustering with deep learning: From the perspective of network architecture," *IEEE Access*, vol. 6, pp. 39 501–39 514, 2018.
[4] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 888–905, 2000.
[5] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems*, 2002, pp. 849–856.



(a) Original point cloud ($t = 1$) (b) Graph for $t = 1$ (c) SC for static graph (d) Proposed method

(e) Original point cloud ($t = 90$) (f) Graph for $t = 90$ (g) SC for static graph (h) Proposed method

Fig. 3. Experimental results for dynamic point clouds.

[6] M. E. Newman, "Fast algorithm for detecting community structure in networks," *Physical review E*, vol. 69, no. 6, p. 066133, 2004.

[7] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E*, vol. 69, no. 2, p. 026113, 2004.

[8] M. E. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Physical review E*, vol. 74, no. 3, p. 036104, 2006.

[9] H. Shen, X. Cheng, K. Cai, and M.-B. Hu, "Detect overlapping and hierarchical community structure in networks," *Physica A: Statistical Mechanics and its Applications*, vol. 388, no. 8, pp. 1706–1712, 2009.

[10] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[11] S. White and P. Smyth, "A spectral clustering approach to finding communities in graphs," in *Proceedings of the 2005 SIAM International Conference on Data Mining*. SIAM, 2005, pp. 274–285.

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

[13] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro, "Time-varying graphs and dynamic networks," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 27, no. 5, pp. 387–408, 2012.

[14] J. Zhang and J. M. Moura, "Diffusion in social networks as SIS epidemics: Beyond full mixing and complete graphs," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 4, pp. 537–551, 2014.

[15] M. G. Preti, T. A. Bolton, and D. Van De Ville, "The dynamic functional connectome: State-of-the-art and perspectives," *Neuroimage*, vol. 160, pp. 41–54, 2017.

[16] J. Hopcroft, O. Khan, B. Kulis, and B. Selman, "Tracking evolving communities in large linked networks," *Proceedings of the National Academy of Sciences*, vol. 101, no. suppl 1, pp. 5249–5253, 2004.

[17] T. Aynaud and J.-L. Guillaume, "Static community detection algorithms for evolving networks," in *8th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*. IEEE, 2010, pp. 513–519.

[18] J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu, "Graphscope: Parameter-free mining of large time-evolving graphs," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007, pp. 687–696.

[19] F. Liu, D. Choi, L. Xie, and K. Roeder, "Global spectral clustering in dynamic networks," *Proceedings of the National Academy of Sciences*, vol. 115, no. 5, pp. 927–932, 2018.

[20] A. Karaaslanli and S. Aviyente, "Constrained spectral clustering for dynamic community detection," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 8474–8478.

[21] L. Hagen and A. B. Kahng, "New spectral methods for ratio cut partitioning and clustering," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 11, no. 9, pp. 1074–1085, 1992.

[22] D. Wagner and F. Wagner, "Between min cut and graph bisection," in *International Symposium on Mathematical Foundations of Computer Science*. Springer, 1993, pp. 744–750.

[23] L. Hagen and A. Kahng, "Fast spectral methods for ratio cut partitioning and clustering," in *1991 IEEE International Conference on Computer-Aided Design Digest of Technical Papers*. IEEE Computer Society, 1991, pp. 10–11.

[24] P. K. Chan, M. D. Schlag, and J. Y. Zien, "Spectral k-way ratio-cut partitioning and clustering," *IEEE Transactions on computer-aided design of integrated circuits and systems*, vol. 13, no. 9, pp. 1088–1096, 1994.

[25] T. Roxborough and A. Sen, "Graph clustering using multiway ratio cut (Software demonstration)," in *International Symposium on Graph Drawing*. Springer, 1997, pp. 291–296.

[26] G. Rossetti and R. Cazabet, "Community discovery in dynamic networks: A survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 2, pp. 1–37, 2018.

[27] L. Condat, "A primal–dual splitting method for convex optimization involving Lipschitzian, proximable and linear composite terms," *Journal of optimization theory and applications*, vol. 158, no. 2, pp. 460–479, 2013.

[28] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[29] P. W. Holland, K. B. Laskey, and S. Leinhardt, "Stochastic blockmodels: First steps," *Social networks*, vol. 5, no. 2, pp. 109–137, 1983.

[30] A. Banerjee, C. Krumpelman, J. Ghosh, S. Basu, and R. J. Mooney, "Model-based overlapping clustering," in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, 2005, pp. 532–537.

[31] J. Gall, C. Stoll, E. De Aguiar, C. Theobalt, B. Rosenhahn, and H.-P. Seidel, "Motion capture using joint skeleton tracking and surface estimation," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 1746–1753.