

Spectral Algorithms for Temporal Graph Cuts

Arlei Silva
University of California
Santa Barbara, CA
arlei@cs.ucsb.edu

Ambuj Singh
University of California
Santa Barbara, CA
ambuj@cs.ucsb.edu

Ananthram Swami
Army Research Laboratory
Adelphi, MD
ananthram.swami.civ@mail.mil

ABSTRACT

The sparsest cut problem consists of identifying a small set of edges that breaks the graph into balanced sets of vertices. The normalized cut problem balances the total degree, instead of the size, of the resulting sets. Applications of graph cuts include community detection and computer vision. However, cut problems were originally proposed for static graphs, an assumption that does not hold in many modern applications where graphs are highly dynamic. In this paper, we introduce sparsest and normalized cuts in temporal graphs, which generalize their standard definitions by enforcing the smoothness of cuts over time. We propose novel formulations and algorithms for computing temporal cuts using spectral graph theory, divide-and-conquer and low-rank matrix approximation. Furthermore, we extend temporal cuts to dynamic graph signals, where vertices have attributes. Experiments show that our solutions are accurate and scalable, enabling the discovery of dynamic communities and the analysis of dynamic graph processes.

CCS CONCEPTS

• Information systems → Data mining;

KEYWORDS

Graph mining; Spectral Theory

ACM Reference Format:

Arlei Silva, Ambuj Singh, and Ananthram Swami. 2018. Spectral Algorithms for Temporal Graph Cuts. In *Proceedings of The Web Conference 2018 (WWW 2018)*. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3178876.3186118>

1 INTRODUCTION

Temporal graphs represent how a graph changes over time, being ubiquitous in data mining and Web applications. Users in social networks present dynamic behavior, leading to the evolution of communities [3]. In hyperlinked environments, such as blogs, new topics drive modifications in content and link structure [26]. Communication, epidemics and mobility are other scenarios where temporal graphs can enable the understanding of complex processes. However, several key concepts and algorithms for static graphs have not been generalized to temporal graphs [12, 36].

This paper focuses on cut problems in temporal graphs, which consist of finding a small sets of edges (or cuts) that break the

graph into balanced sets of vertices. Two traditional graph cut problems are the *sparsest cut* [18, 31] and the *normalized cut* [9, 40]. In sparsest cuts, the resulting partitions are balanced in terms of size, while in normalized cuts, the balance is in terms of total degree (or volume) of the resulting sets. Graph cuts have applications in community detection, image segmentation, clustering, and VLSI design. Moreover, the computation of graph cuts based on eigenvectors of graph-related matrices is one of the earliest results in spectral graph theory [9], a subject with great impact in information retrieval [29], graph sparsification [46], and machine learning [27].

One of our motivations to study graph cuts in this new setting is the emerging field of *Signal Processing on Graphs (SPG)* [41]. SPG is a framework for the analysis of data residing on vertices of a graph, as a generalization of traditional signal processing. Temporal cuts can be applied as bases for signal processing on dynamic graphs.

Our Contributions. We propose formulations of sparsest and normalized cuts in a sequence of graph snapshots. The idea is to extend classical definitions of these problems while enforcing the smoothness (or stability) of cuts over time. Our formulations can be understood using a multiplex view of the temporal graph, where additional edges connect the same vertex in different snapshots.

Figure 1 shows a sparse temporal graph cut for a school network [47], where children are connected based on proximity. Vertices are naturally organized into communities resulting from classes. However, there is a significant amount of interaction across classes (e.g. during lunch). Major changes in the contact network can be noticed during the experiment, causing several vertices to move across partitions—identified with vertex shapes/colors. The temporal cut is able to capture such trends while keeping the remaining vertex assignments mostly unchanged.

Traditional spectral solutions, which compute approximated cuts as rounded eigenvectors of the Laplacian matrix, do not generalize to our setting. Thus, we propose new algorithms, still within the framework of spectral graph theory, for the computation of temporal cuts. We further exploit important properties of our formulation to design efficient approximation algorithms for temporal cuts combining *divide-and-conquer* and low-rank matrix approximation.

In order to also model dynamic data embedded on the vertices of a graph, we apply temporal cuts as data-driven wavelet bases. Our approach exploits smoothness in both space and time, illustrating how the techniques presented in this paper provide a powerful and general framework for the analysis of dynamic graphs.

Summary of contributions.

- We generalize sparsest and normalized cuts to temporal graphs; we further extend temporal cuts to graph signals.
- We propose efficient approximate algorithms for temporal cuts via spectral graph theory and divide-and-conquer.
- We evaluate our methods extensively, applying them for community detection and signal processing on graphs.

This paper is published under the Creative Commons Attribution 4.0 International (CC BY 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.

WWW 2018, April 23–27, 2018, Lyons, France

© 2018 IW3C2 (International World Wide Web Conference Committee), published under Creative Commons CC BY 4.0 License.

ACM ISBN 978-1-4503-5639-8/18/04.

<https://doi.org/10.1145/3178876.3186118>

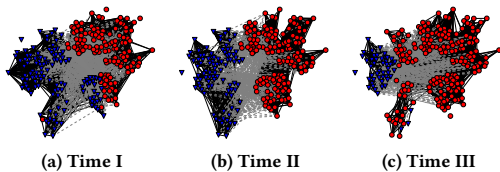


Figure 1: Temporal graph cut capturing major changes in the network interactions. Figure is better seen in color.

Related Work. Computing graph cuts is a traditional problem [2, 31] with a diverse set of applications, ranging from image segmentation [40] to community detection [32]. This paper is focused on the sparsest and normalized cut problems, which are of particular interest due to their connections with the spectrum of the Laplacian matrix, mixing time of random walks, geometric embeddings, effective resistance, and graph expanders [9, 18, 46].

Community detection in temporal graphs has attracted great interest in recent years. An evolutionary spectral clustering technique was proposed in [8]. The idea is to minimize a cost function $\alpha.CS + \beta.CT$, where CS is a snapshot cost and CT is a temporal cost. FacetNet [34] and estrangement [21] apply a similar approach under different clustering models. An important limitation of these solutions is that they perform community assignments in a step-wise manner, being highly subject to local optima. In incremental clustering [7, 38], the main goal is to avoid recomputation, and not to capture long-term structural changes. Multi-view clustering [5, 50] combines different subsets of features (or views) from a given dataset but does not model how objects navigate across clusters over time. Different from spatio-temporal data clustering [33, 39], we do not assume that our data is embedded in an Euclidean space.

A formulation for temporal modularity that simultaneously partitions snapshots using a multiplex graph [22] was proposed in [37]. A similar idea was applied in [48] to generalize eigenvector centrality. In this paper, we propose generalizations for temporal cut problems by studying spectral properties of multiplex graphs [17, 44]. As one of our contributions, we exploit the link between multiplex graphs and block tridiagonal matrices to efficiently approximate temporal cuts [11, 14]. While extending spectral graph theory to tensors seems to be a more natural approach to our problems, eigenvectors are well-studied only for symmetric tensors [19], which is not our case due to the time dimension.

Our definition of temporal cuts is a special case of non-uniform cuts [49]—the second graph is a sequence of disconnected cliques to enforce cuts over time. Different from the general case, which requires more sophisticated (and computationally intensive) schemes [10], a relaxation for temporal cuts can be computed as an eigenvector of a linear combination of two matrices (see Theorem 1).

Signal processing on graphs [41, 42] is an interesting application of temporal cuts. Traditional signal processing operations are also relevant when the signal is embedded into sparse irregular spaces. For instance, in machine learning, object similarity can be represented as a graph and labels as signals to solve semi-supervised learning tasks [6, 15]. In this paper, we show how temporal cuts can be applied as bases for representing dynamic graph signals, even in the case where the graph structure also changes over time.

2 TEMPORAL GRAPH CUTS

This section introduces temporal cuts (Section 2.1) and spectral algorithms for these problems (Section 2.2). Faster algorithms, using divide-and-conquer and low-rank matrix approximation, are presented in Section 2.3. We also discuss theoretical guarantees (Section 2.4) and generalizations for temporal cuts (Section 2.5).

2.1 Definitions

A temporal graph \mathcal{G} is a sequence of snapshots $\langle G_1, G_2, \dots, G_m \rangle$ where G_t is the snapshot at timestamp t . G_t is a tuple (V, E_t, W_t) where V is a fixed set with n vertices, E_t is a dynamic set of undirected edges and $W_t : E_t \rightarrow \mathbb{R}$ is an edge weighting function.

We model temporal graphs as *multiplex graphs*, which connect vertices from different graph layers. We denote as $\chi(\mathcal{G}) = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ the multiplex view of \mathcal{G} , where $\mathcal{V} = \{v_t | v \in V \wedge t \in [1, m]\}$ ($|\mathcal{V}| = nm$) and $\mathcal{E} = E_1 \cup \dots \cup E_t \cup \{(v_t, v_{t+1}) | v \in V \wedge t \in [1, m-1]\}$. Thus, \mathcal{E} also includes ‘vertical’ edges between nodes v_t and v_{t+1} . The edge weighting function $\mathcal{W} : \mathcal{E} \rightarrow \mathbb{R}$ is defined as:

$$\mathcal{W}(u_r, v_s) = \begin{cases} W_t(u, v), & \text{if } (u, v) \in E_t \wedge r = t \wedge s = t \\ \beta, & \text{if } u = v \wedge |r - s| = 1 \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

As a result, each vertex $v \in V$ has m representatives $\{v_1, \dots, v_m\}$ in $\chi(\mathcal{G})$. Besides the intra-layer edges corresponding to the connectivity of each snapshot E_t , temporal edges (v_t, v_{t+1}) connect consecutive versions of a vertex v at different layers, which is known as *diagonal coupling* [4]. Intra-layer edge weights are the same as in \mathcal{G} while inter-layer weights are set to β .

2.1.1 Sparsest Cut. A graph cut (X, \bar{X}) divides V into two disjoint sets: $X \subseteq V$ and $\bar{X} = V - X$. We denote the weight of a cut $|(X, \bar{X})| = \sum_{u \in X, v \in \bar{X}} W(u, v)$. The *cut sparsity* σ is the ratio of the cut weight and the product of the sizes of the sets [31]:

$$\sigma(X) = \frac{|(X, \bar{X})|}{|X||\bar{X}|} \quad (2)$$

Here, we extend the notion of cut sparsity to temporal graphs. A temporal cut $\langle (X_1, \bar{X}_1), \dots, (X_m, \bar{X}_m) \rangle$ is a sequence of graph cuts where (X_t, \bar{X}_t) is a cut of the graph snapshot G_t . The idea is that in temporal graphs, besides the cut weights and partition sizes, we also care about the smoothness (i.e. stability) of the cuts over time. We formalize the temporal cut sparsity σ as follows:

$$\sigma(X_1, \dots, X_m; \beta) = \frac{\sum_{t=1}^m |(X_t, \bar{X}_t)| + \sum_{t=1}^{m-1} \Delta(X_t, \bar{X}_{t+1})}{\sum_{t=1}^m |X_t||\bar{X}_t|} \quad (3)$$

where $\Delta(X_t, \bar{X}_{t+1}) = \beta|(X_t, \bar{X}_{t+1})|$ is the number of vertices that move from X_t to \bar{X}_{t+1} (or $|X_t \cap \bar{X}_{t+1}|$) times the constant β , which allows different weights to be given to the cut smoothness.

Figure 2 shows two alternative cuts for a temporal graph ($\beta = 1$). Cut I (Figure 2a) is smooth, since no vertex changes partitions, and it has weight 5. Cut II (Figure 2b) is a sparser temporal cut, with weight 3 and only one vertex changing partitions. Notice that cut I becomes sparser than cut II if β is set to 2 instead of 1. We formalize the sparsest cut problem in temporal graphs as follows.

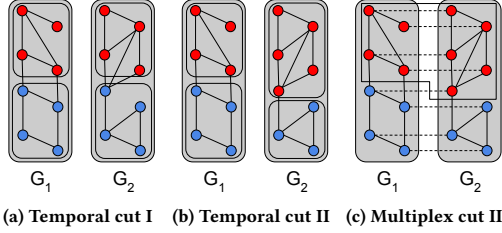


Figure 2: Two temporal cuts for the same graph and multiplex view of cut II. For $\beta = 1$, cut I has a sparsity of $(2 + 3 + 0)/(4 \times 4 + 4 \times 4) = 0.16$ while cut II has sparsity of $(2 + 1 + 1)/(4 \times 4 + 5 \times 3) = 0.13$. Thus, II is a sparser cut.

DEFINITION 1. **Sparsest temporal cut.** The sparsest cut of a temporal graph \mathcal{G} , for a constant β , is defined as:

$$\arg \min_{X_1, \dots, X_m} \sigma(X_1, \dots, X_m; \beta)$$

The sparsest temporal cut is a generalization of the sparsest cut problem and thus also NP-hard [18].

An interesting property of the multiplex model is that temporal cuts in \mathcal{G} become standard—single graph—cuts in the multiplex view $\chi(\mathcal{G})$. We can evaluate the sparsity of a cut in \mathcal{G} by applying the original formulation (Expression 2) to $\chi(\mathcal{G})$, since both edges cut and partition changes in \mathcal{G} become edges cut in $\chi(\mathcal{G})$. As an example, we show the multiplex view of cut II (Figure 2b) in Figure 2c. However, notice that not every standard cut in $\chi(\mathcal{G})$ is a valid temporal cut. For instance, cutting all the temporal edges (i.e. separating the two snapshots in our example) would be allowed in the standard formulation, but would lead to an undefined value of sparsity as the denominator in Expression 3 will be 0. Therefore, we cannot directly apply existing sparsest cut algorithms to $\chi(\mathcal{G})$ and expect to achieve a sparse temporal cut for \mathcal{G} .

The connection between temporal cuts and multiplex networks is one of the main motivations for our formulation. Moreover, Equation 3 is general enough to capture different dynamic behaviors depending on the constant β . More specifically, if $\beta \rightarrow \infty$, the sparsity σ will be minimized for a constant cut over the snapshots. On the other hand, if $\beta \rightarrow 0$, σ approximates the sparsity of single snapshot cuts. Interestingly, these two extreme regimes have been studied in the context of random-walks on dynamic graphs [13].

2.1.2 *Normalized Cut.* A limitation of Equation 2, is that it favors sparsity over partition size balance. In community detection, this often leads to “whisker communities” [28, 32]. Normalized cuts [40] take into account the *volume* (i.e. sum of the degrees of vertices) of the partitions, being less prone to this effect. The normalized version of the cut sparsity is defined as:

$$\phi(X) = \frac{|(X, \bar{X})|}{\text{vol}(X) \cdot \text{vol}(\bar{X})} \quad (4)$$

where $\text{vol}(Y) = \sum_{v \in Y} \text{deg}(v)$ and $\text{deg}(v)$ is the degree of v .

We also generalize the normalized sparsity ϕ to temporal graphs:

$$\phi(X_1, \dots, X_m; \beta) = \frac{\sum_{t=1}^m |(X_t, \bar{X}_t)| + \sum_{t=1}^{m-1} \Delta(X_t, \bar{X}_{t+1})}{\sum_{t=1}^m \text{vol}(\bar{X}_t) \text{vol}(X_t)} \quad (5)$$

Next, we define the normalized cut problem for temporal graphs.

DEFINITION 2. **Normalized temporal cut.** The normalized temporal cut of \mathcal{G} , for a constant β , is defined as:

$$\arg \min_{X_1, \dots, X_m} \phi(X_1, \dots, X_m; \beta)$$

Computing optimal normalized temporal cuts is also NP-hard. The next section introduces spectral approaches for temporal cuts.

2.2 Spectral Approaches

Similar to the single graph case, we also exploit spectral graph theory in order to compute good temporal graph cuts. Let A be an $n \times n$ weighted adjacency matrix of a graph $G(V, E)$, where $A_{u,v} = W(u, v)$ if $(u, v) \in E$ or 0, otherwise. The degree matrix D is an $n \times n$ diagonal matrix, with $D_{v,v} = \text{deg}(v)$ and $D_{u,v} = 0$ for $u \neq v$. The Laplacian matrix of G is defined as $L = D - A$. Let L_t be the Laplacian matrix of the graph snapshot G_t and I_z be an $z \times z$ identity matrix. We define the Laplacian of the temporal graph \mathcal{G} as the Laplacian of its multiplex view $\chi(\mathcal{G})$:

$$\mathcal{L} = \begin{pmatrix} L_1 + \beta I_n & -\beta I_n & 0 & \dots & 0 \\ -\beta I_n & L_2 + 2\beta I_n & -\beta I_n & \dots & 0 \\ \vdots & & \ddots & \dots & -\beta I_n \\ 0 & 0 & \dots & -\beta I_n & L_m + \beta I_n \end{pmatrix}$$

The matrix \mathcal{L} can be also written in a more compact form using the Kronecker product as $\mathbf{diag}(L_1, \dots, L_m) + \beta(L^\ell \otimes I_n)$, where L^ℓ is the Laplacian of a line graph. Similarly, we define the degree matrix \mathcal{D} of \mathcal{G} as $\mathbf{diag}(D_1, D_2, \dots, D_m)$, where D_t is the degree matrix of G_t . Let $C = nI_n - \mathbf{1}_{n \times n}$ be the Laplacian of a clique with n vertices. We define another $nm \times nm$ Laplacian matrix $C = I_m \otimes C$, which is the Laplacian of a graph composed of m isolated cliques. This matrix will be applied to enforce valid temporal cuts over the snapshots of \mathcal{G} . Further, we define a size- nm indicator vector \mathbf{x} where each vertex $v \in V$ is represented m times, one for each snapshot. The value $\mathbf{x}[v_t] = 1$ if $v_t \in X_t$ and $\mathbf{x}[v_t] = -1$ if $v_t \in \bar{X}_t$.

2.2.1 *Sparsest Cut.* The next lemma shows how the matrices \mathcal{L} and C can be applied to compute the sparsity of temporal cuts.

LEMMA 2.1. The sparsity σ of a temporal cut is equal to $\frac{\mathbf{x}^\top \mathcal{L} \mathbf{x}}{\mathbf{x}^\top C \mathbf{x}}$.

PROOF. Since \mathcal{L} is the Laplacian of $\chi(\mathcal{G})$:

$$\begin{aligned} \mathbf{x}^\top \mathcal{L} \mathbf{x} &= \sum_{(u,v) \in \mathcal{E}} (\mathbf{x}[u] - \mathbf{x}[v])^2 W(u, v) \\ &= 4 \sum_{t=1}^m |(X_t, \bar{X}_t)| + 4 \sum_{t=1}^{m-1} \Delta(X_t, \bar{X}_{t+1}) \end{aligned}$$

Regarding the denominator:

$$\begin{aligned}\mathbf{x}^\top \mathbf{C} \mathbf{x} &= \sum_{t=1}^m \sum_{u \neq v} (\mathbf{x}[v_t] - \mathbf{x}[u_t])^2 \\ &= 4 \sum_{t=1}^m |X_t| |\bar{X}_t|\end{aligned}$$

□

Based on Lemma 2.1, we can obtain a relaxed solution, $\mathbf{x} \in [-1, 1]^{nm}$, for the sparsest temporal cut problem (Definition 1) in $O(n^3 m^3)$ time via generalized eigenvalue computation [16], or an approximate solution in $O(n^2 m \log^2(n^2 m))$ time using a fast Laplacian linear system solver [24, 45]. Solutions are later rounded to adhere to the original (discrete) constraint $\mathbf{x} \in \{-1, 1\}^{nm}$. The next lemma supports a more efficient solution for our relaxation.

LEMMA 2.2. *The matrices C and \mathcal{L} commute.*

PROOF. First, we show that C commutes with any Laplacian L_t :

$$CL_t = (nI_n - \mathbf{1}_{n \times n})L_t = nL_t = L_t C$$

Using the Kronecker product notation (see Section 2.2):

$$\begin{aligned}C\mathcal{L} &= (I_m \otimes C)[\mathbf{diag}(L_1, \dots, L_m) - \beta(L^\ell \otimes I_n)] \\ &= \mathbf{diag}(CL_1, \dots, CL_m) - \beta(I_m L^\ell) \otimes (CI_n) \\ &= \mathbf{diag}(L_1 C, \dots, L_m C) - \beta(L^\ell I_m) \otimes (I_n C) \\ &= \mathcal{L}C\end{aligned}$$

□

A relaxation of the sparsest temporal cut can be computed based on a linear combination of matrices C and \mathcal{L} .

THEOREM 1. *A relaxed solution for the sparsest temporal cut problem can, alternatively, be computed as:*

$$\mathbf{y}^* = \arg \max_{\mathbf{y} \in [-1, 1]^{nm}} \frac{\mathbf{y}^\top [(3(n+2\beta)C - \mathcal{L})\mathbf{y}]}{\mathbf{y}^\top \mathbf{y}} \quad (6)$$

which is the largest eigenvector of $3(n+2\beta)C - \mathcal{L}$.

PROOF. The spectrum of C is in the form $(\mathbf{e}_i, \lambda_1) = (\mathbf{1}_n, 0)$ and $\lambda_2 = \dots = \lambda_n = n$ for any vector $\mathbf{e}_i \perp \mathbf{1}_n$. As a consequence, the spectrum of C is in the form $\lambda_1 = \dots = \lambda_m = 0$ and $\lambda_{m+1} = \dots = \lambda_{nm} = n$ for any $\mathbf{e}_i \perp \mathbf{span}\{e_1, \dots, e_m\}$. Lemma 2.2 implies that any linear combination of C and \mathcal{L} has the same eigenvectors as \mathcal{L} [20]. Upper bounding the eigenvalues of a Laplacian matrix [1]:

$$0 \leq \frac{\mathbf{y}^\top \mathcal{L} \mathbf{y}}{\mathbf{y}^\top \mathbf{y}} \leq 2(n+2\beta) \quad (7)$$

This implies that:

$$\frac{(\mathbf{y}^*)^\top C (\mathbf{y}^*)}{(\mathbf{y}^*)^\top (\mathbf{y}^*)} = n \quad (8)$$

as it guarantees a strictly positive ratio (i.e. eigenvalue) in Equation 6. Based on Equation 8, we can re-write \mathbf{y}^* as:

$$\mathbf{y}^* = \arg \min_{\mathbf{y} \in [-1, 1]^{nm}, \mathbf{y}^\top C \mathbf{y} > 0} \frac{\mathbf{y}^\top \mathcal{L} \mathbf{y}}{\mathbf{y}^\top \mathbf{y}} \quad (9)$$

Moreover, from Lemma 2.1:

$$\mathbf{x}^* = \arg \min_{\mathbf{x} \in [-1, 1]^{nm}, \mathbf{x}^\top C \mathbf{x} > 0} \frac{\mathbf{x}^\top \mathcal{L} \mathbf{x}}{\mathbf{x}^\top C \mathbf{x}} \quad (10)$$

Equations 9 and 10 are related to an *eigenvalue* and a *generalized eigenvalue problem*, respectively, and can be written as follows:

$$\mathcal{L} \mathbf{y} = \lambda \mathbf{y}, \quad \mathcal{L} \mathbf{x} = \lambda' C \mathbf{x} \quad (11)$$

where λ and λ' are minimized and $\mathbf{y}^\top C \mathbf{y}, \mathbf{x}^\top C \mathbf{x} > 0$. From Equation 8, we know that $C \mathbf{y} = n \mathbf{y}$. Thus, $\mathcal{L} \mathbf{y} = (\lambda/n) C \mathbf{y}$ is a corresponding solution (same eigenvector) to the generalized problem. □

The matrix $3(n+2\beta)C - \mathcal{L}$ is a Laplacian of a multiplex graph in which temporal edges have weight $w'(v_t, v_{t+1}) = -\beta$ and intra-layer edges have weight $w'(u, v) = 3(n+2\beta) - w(u, v)$. This leads to a reordering of the spectrum of \mathcal{L} where cuts containing only temporal edges have negative associated eigenvalues and sparse cuts for each Laplacian L_t become dense cuts for a new Laplacian $[3(n+2\beta)C - \mathcal{L}]$. In terms of complexity, computing the difference $3(n+2\beta)C - \mathcal{L}$ takes $O(n^2 m)$ time and the largest eigenvector of $[3(n+2\beta)C - \mathcal{L}]$ can be calculated in $O(n^2 m)$. The resulting complexity is a significant improvement over the $O(n^3 m^3)$ standard alternative, specially if the number of snapshots is large.

2.2.2 *Normalized Cut.* We follow the steps of the previous section to compute normalized temporal cuts. *See the extended version of this paper [43] for proofs of Lemma 2.3 and Theorem 2.*

LEMMA 2.3. *The normalized sparsity ϕ of a temporal cut is equal to $\frac{\mathbf{x}^\top \mathcal{L} \mathbf{x}}{\mathbf{x}^\top \mathcal{D}^{\frac{1}{2}} C \mathcal{D}^{\frac{1}{2}} \mathbf{x}}$.*

We also define an equivalent of Theorem 1 for normalized cuts.

THEOREM 2. *A relaxed solution for the sparsest temporal cut problem can, alternatively, be computed as:*

$$\mathbf{y}^* = \arg \max_{\mathbf{y} \in [-1, 1]^{nm}} \frac{\mathbf{y}^\top [(3(n+2\beta)C - (\mathcal{D}^+)^{\frac{1}{2}} \mathcal{L} (\mathcal{D}^+)^{\frac{1}{2}})] \mathbf{y}}{\mathbf{y}^\top \mathbf{y}} \quad (12)$$

the largest eigenvector of $3(n+2\beta)C - (\mathcal{D}^+)^{\frac{1}{2}} \mathcal{L} (\mathcal{D}^+)^{\frac{1}{2}}$.

The interpretation of matrix $3(n+2\beta)C - (\mathcal{D}^+)^{\frac{1}{2}} \mathcal{L} (\mathcal{D}^+)^{\frac{1}{2}}$ is similar to the one for sparsest cuts, with temporal edges having negative weights. Moreover, the complexity of computing the largest eigenvector of such matrix is also $O(n^2 m)$. This quadratic cost on the size of the graph, for both sparsest and normalized cut problems, becomes prohibitive even for reasonably small graphs. The next section is focused on faster algorithms for temporal graph cuts.

2.3 Fast Approximations

By definition, sparse temporal cuts are sparse in each snapshot and smooth across snapshots. Similarly, normalized temporal cuts are composed of a sequence of good normalized snapshot cuts that are stable over time. This motivates *divide-and-conquer* approaches for computing temporal cuts that first find a good cut on each snapshot (*divide*) and then combine them (*conquer*). These solutions have the potential to be much more efficient than the ones based on Theorems 1 and 2 if the conquer step is fast. However, they could lead to sub-optimal results, as optimal temporal cuts might not be composed of each snapshot's best cuts. Instead, better *divide-and-conquer* schemes can explore multiple snapshot cuts in the

conquer step to avoid local optima. Since we are working in the spectral domain, it is natural to take eigenvectors of blocks of $M_S = 3(n + 2\beta)C - \mathcal{L}$ and $M_N = 3(n + 2\beta)C - (\mathcal{D}^+)^{\frac{1}{2}} \mathcal{L} (\mathcal{D}^+)^{\frac{1}{2}}$, as continuous notions of snapshot cuts. This strategy is supported by the well-known connections between higher-order eigenvectors of the Laplacian matrix and the sparsity of multiway cuts [30, 35].

This section will describe our general *divide-and-conquer* approach. We will focus our discussion on the sparsest cut problem and then briefly show how it can be generalized to normalized cuts. The following theorem is the basis of our algorithm.

THEOREM 3. *The eigenvalues of the matrix $M_S = 3(n + 2\beta)C - \mathcal{L}$ are the same as the ones for the matrix Q :*

$$Q = \Lambda - \beta \begin{pmatrix} I_n & -U_1^\top U_2 & 0 & \dots & 0 \\ -U_2^\top U_1 & 2I_n & -U_2^\top U_3 & \dots & 0 \\ \vdots & & \ddots & \dots & U_{m-1}^\top U_m \\ 0 & 0 & \dots & U_m^\top U_{m-1} & I_n \end{pmatrix}$$

where $U_t \Lambda_t U_t^\top$ is the eigendecomposition of $M_t = (3(n + 2\beta)C - L_t)$ and $\Lambda = \text{diag}(\Lambda_1 \dots \Lambda_m)$. An eigenvector e_j of M_S is computed as $\mathcal{U} \cdot e_j^Q$, where $\mathcal{U} = \text{diag}(U_1 \dots U_m)$ and e_j^Q is an eigenvector of Q .

PROOF. We show that M_S and Q are similar matrices under the change of basis \mathcal{U}^\top and thus $M = (\mathcal{U}^\top)^{-1} Q \mathcal{U}^\top$. Let's define matrices $\mathcal{B} = \beta(L^\ell \otimes I_n)$ and $M_t = 3(n - 2\beta)C - L_t$. Because L_t is symmetric, $\mathcal{U}^{-1} = \mathcal{U}^\top$. For an eigenvector matrix \mathcal{U} , $\mathcal{U} \mathcal{U}^\top$ is an $nm \times nm$ identity matrix \mathcal{I} . We rewrite M_S as:

$$\begin{aligned} M_S &= \text{diag}(M_1, M_2 \dots M_m) - \mathcal{B} \\ &= \mathcal{U} \Lambda \mathcal{U}^\top - \mathcal{I} \mathcal{B} \mathcal{I} \\ &= \mathcal{U} \Lambda \mathcal{U}^\top - \mathcal{U} \mathcal{U}^\top \mathcal{B} \mathcal{U} \mathcal{U}^\top \\ &= \mathcal{U} (\Lambda - \mathcal{U}^\top \mathcal{B} \mathcal{U}) \mathcal{U}^\top \\ &= (\mathcal{U}^\top)^{-1} Q \mathcal{U}^\top \end{aligned}$$

□

Q has $O(n^2 m)$ non-zeros, being asymptotically as sparse as M_S . However, Q can be block-wise sparsified using low-rank approximations of the matrices M_t . Given a constant $r \leq n$, we approximate each M_t as $U_t \Lambda'_t U_t^\top$, where Λ'_t contains only the top- r eigenvalues of M_t . The benefits of such a strategy are the following: (1) The cost of computing the eigendecomposition of M_t changes from $O(n^3)$ to $O(rn^2)$; (2) the cost of multiplying eigenvector matrices decreases from $O(n^3)$ to $O(r^3)$; and (3) the number of non-zeros in Q is reduced from $O(n^2 m)$ to $O(r^2 m)$. Similar to the case of general block tridiagonal matrices [14], we can show that the error associated with such approximation is bounded by $2\lambda_{r+1}^{max}$, where λ_{r+1}^{max} is the largest $(r+1)$ -nth eigenvalue of the approximated matrices M_t .

We improve our approach by speeding-up the eigendecomposition of the matrices M_t . The idea is to operate over the original Laplacians L_t , which are expected to be sparse. The eigendecomposition of a matrix with $|E|$ non-zeros can be performed in time $O(n|E|)$ and for real-world graphs $|E| \ll n^2$. The following Lemma shows how the spectrum of M_t can be computed based on L_t .

LEMMA 2.4. *Let $\lambda_1^L, \lambda_2^L \dots \lambda_n^L$ be the eigenvalues of a Laplacian matrix L in increasing order with associated eigenvectors $e_1^L, e_2^L \dots e_n^L$. The eigenvectors e_i^L are also eigenvectors of $3(n + 2\beta)C - L$ with*

Algorithm 1 Spectral Algorithm

Require: Temporal graph \mathcal{G} , rank r , constant β
Ensure: Temporal cut $\langle (X_1, \bar{X}_1), \dots (X_m, \bar{X}_m) \rangle$
1: **for** $G_t \in \mathcal{G}$ **do**
2: Compute bottom- r eigendecomposition $U_t' \Lambda_t' U_t'^\top \approx L_t$
3: Fix $\Lambda_t \leftarrow \text{diag}(0, 3(n + 2\beta)n - \lambda_n, \dots, 3(n + 2\beta)n - \lambda_2)$
4: **end for**
5: $Q \leftarrow \text{diag}(\Lambda_1, \dots, \Lambda_m) - \mathcal{B}$
6: **for** $t \in [1, m - 1]$ **do**
7: $Q_{t,t+1} \leftarrow U_t'^\top U_t'$, $Q_{t+1,t} \leftarrow U_{t+1}'^\top U_{t+1}'$
8: **end for**
9: Compute largest eigenvector $\mathbf{x}^* \leftarrow \arg \max_{\mathbf{x}} \mathbf{x}^\top Q \mathbf{x} / \mathbf{x}^\top \mathbf{x}$
10: **return** rounded cut **sweep**($\mathcal{U} \cdot \mathbf{x}^*$, \mathcal{G} , β)

associated eigenvalues $\lambda_1 = 0$ and $\lambda_i = 3(n + 2\beta)n - \lambda_{n-i+1}$ for $i > 0$.

PROOF. The spectrum of C is $(\mathbf{e}_1, \lambda_1) = (\mathbf{1}_n, 0)$ and $\lambda_2 = \dots = \lambda_n = n$ for any vector $\mathbf{e}_i \perp \mathbf{1}_n$. As M_i is also a Laplacian, it follows that $\lambda_1 = 0$ and $e_1 = e_1^L$. Also, by definition $L \cdot e_i^L = \lambda_i^L \cdot e_i^L$, and thus $(3(n + 2\beta)C - L)e_i^L = (3(n + 2\beta)n - \lambda_i)e_i^L$ for $i > 0$. □

Algorithm 1 describes our *divide-and-conquer* approach for approximating the sparsest temporal cut. Its inputs are the temporal graph \mathcal{G} , the rank r that controls the accuracy of the algorithm, and a constant β . It returns a cut $\langle (X_1, \bar{X}_1) \dots (X_m, \bar{X}_m) \rangle$ that (approximately) minimizes the sparsity ratio defined in Equation 3. In the *divide* phase, the top- r eigenvalues/eigenvectors of each matrix M_t —related to the bottom- r eigenvalues/eigenvectors of L_t —are computed using Lemma 2.4 (steps 1-4). The *conquer* phase (steps 5-9) consists of building the matrix Q —based on the blocks $Q_{t,t+1}$ adjacent to the diagonal—and then computing its largest eigenvector as a relaxed version of a temporal cut. The resulting eigenvector is discretized using a standard sweep algorithm (**sweep**) over the vertices sorted by their corresponding value of \mathbf{x}^* . The selection criteria for the sweep algorithm is the sparsity ratio (Equation 3).

The time complexity of our algorithm is $O(mr \sum_{t=1}^m |E_t| + mr^3)$. The *divide* step has cost $O(mr \sum_{t=1}^m |E_t|)$, which corresponds to the computation of r eigenvectors/eigenvalues of matrices L_t with $O(|E_t|)$ non-zeros each. As snapshots are processed independently, this part of the algorithm can be easily parallelized. In the *conquer* step, the most time consuming operation is computing $m - 1$ $r \times r$ matrix products in the construction of Q , which takes $O(r^3 m)$ time. Our algorithm has space complexity of $O(r^2 m)$ due to the number of non-zeros in the sparse representation of Q .

We follow the same general approach discussed in this section to efficiently compute normalized temporal cuts. As in Theorem 3, we can compute the eigenvectors of M_N using divide-and-conquer. However, each block M_t will be in the form $3(n - 2\beta)C - (D_t^+)^{\frac{1}{2}} L_t (D_t^+)^{\frac{1}{2}}$. Moreover, similar to Lemma 2.4, we can also compute the eigendecomposition of M_t based on $(D_t^+)^{\frac{1}{2}} L_t (D_t^+)^{\frac{1}{2}}$.

2.4 Approximation Guarantees

The algorithms presented in Sections 2.2 and 2.3 are based on eigenvector computations that are relaxations of temporal cut problems. A natural question to ask is: *Do they provide any approximation guarantees with respect to the optimal solution for the problems?* Notice that, for the fast solutions discussed in the previous section, the number of top eigenvalues (r) considered by Algorithm 1 gives

some control over the quality of the approximations. Therefore, we focus on bounding the error induced by the relaxations described in Lemmas 2.1 and 2.3 and by Theorems 1 and 2.

Our analysis is heavily based on a recent generalization of the Cheeger inequality [10, 23], which relates the ratio between the weights of the same cut on two graphs, G and H , with the same set of vertices, and the generalized eigenvalue involving their Laplacian matrices L_G and L_H . This generalization is the main ingredient to proof the following Lemma regarding our spectral algorithm for the sparsest temporal cut problem (Definition 1):

LEMMA 2.5. *The temporal sparsity ratio $\lambda(\mathcal{G})$ achieved by our relaxation is such that:*

$$\lambda(\mathcal{G}) \geq \frac{\varphi(\chi(\mathcal{G})) \min(\sigma_{X_1, \dots, X_m}(X_1, \dots, X_m; \beta))}{8} \quad (13)$$

where $\lambda(\mathcal{G}) = \min_{\mathbf{x}^\top \mathbf{C} \mathbf{x} > 0} \frac{\mathbf{x}^\top \mathcal{L} \mathbf{x}}{\mathbf{x}^\top \mathbf{C} \mathbf{x}}$ and $\varphi(\chi(\mathcal{G}))$ is the (standard) conductance of the multiplex view of \mathcal{G} .

The lemma follows directly from [23, Theorem 1], by setting G to our temporal graph \mathcal{G} and H to the sequence of cliques with Laplacian \mathbf{C} , and thus the proof is omitted. Similar to the classical Cheeger inequality [9], the proof of its generalized version is also constructive, and the rounding algorithm applied by our solutions achieves this bound. We can interpret Lemma 2.5 as follows. If the temporal graph \mathcal{G} has a low-sparsity cut compared to the conductance of its multiplex view $\chi(\mathcal{G})$, then our relaxations will find a good approximate (possibly different) temporal cut. A similar bound also holds for normalized temporal cuts (Definition 2).

2.5 Generalizations

Here, we briefly address several generalizations of temporal cuts that aim to increase the applicability of this work.

Arbitrary swap costs: While we have assumed uniform swap costs β , generalizing our formulation to arbitrary (non-negative) swap costs for pairs (v_t, v_{t+1}) is straightforward.

Multiple cuts: Multi-cuts can be computed based on the top eigenvectors of our temporal cut matrices, as proposed in [40]. We use k -means to obtain a k -way partition.

3 SIGNAL PROCESSING ON GRAPHS

We apply graph cuts as data-driven wavelet bases for dynamic signals. Given a sequence of signals $(\mathbf{f}_{(1)}, \dots, \mathbf{f}_{(m)})$, $\mathbf{f}_{(i)} \in \mathbb{R}^n$, on a temporal graph \mathcal{G} , our goal is to discover a temporal cut that is sparse, smooth, and separates vertices with dissimilar signal values. A previous work [42] has shown that a relaxation of the L_2 energy (or importance) $\|a\|_2$ of a wavelet coefficient a for a single graph snapshot with signal \mathbf{f} can be computed as:

$$\frac{(|X| \sum_{v \in \bar{X}} \mathbf{f}[v] - |\bar{X}| \sum_{u \in X} \mathbf{f}[u])^2}{|X| |\bar{X}|} \propto - \frac{\mathbf{x}^\top \mathbf{C} \mathbf{S} \mathbf{C} \mathbf{x}}{\mathbf{x}^\top \mathbf{C} \mathbf{x}} \quad (14)$$

where \mathbf{x} is an indicator vector and $S_{u,v} = (\mathbf{f}[v] - \mathbf{f}[u])^2$. Sparsity is enforced by adding a Laplacian regularization factor $\alpha \mathbf{x}^\top \mathbf{L} \mathbf{x}$, where α is a user-defined constant, to the denominator of Equation 14. This formulation supports an algorithm for computing graph wavelets, which we extend to dynamic signals. Following the same approach as in Section 2.1, we apply the multiplex graph representation to compute the energy of the dynamic wavelet coefficient:

$$\frac{\sum_{t=1}^m \Theta_t^2 + \sum_{t=1}^{m-1} \Theta_t \Theta_{t+1}}{\sum_{t=1}^m |X_t| |\bar{X}_t|} \quad (15)$$

where $\Theta_t = |X_t| \sum_{v \in \bar{X}_t} \mathbf{f}[v] - |\bar{X}_t| \sum_{u \in X_t} \mathbf{f}[u]$. The first term in the numerator of Equation 15 is the sum of the numerator of Equation 14 over all snapshots. The second term acts on sequential snapshots and enforces the partitions to be consistent over time —i.e. X_t 's to be jointly associated with either large or small values. Intuitively, the energy is maximized for partitions that separate different values and are also balanced in size. The next theorem provides a spectral formulation for the energy of dynamic wavelets.

THEOREM 4. *The energy of a dynamic wavelet is proportional to $-\frac{\mathbf{x}^\top \mathbf{C} \mathbf{S} \mathbf{C} \mathbf{x}}{\mathbf{x}^\top \mathbf{C} \mathbf{x}}$, where $S_{u,v} = (\mathbf{f}[u] - \mathbf{f}[v])^2$ for values within one snapshot from each other.*

We apply Theorem 4 (see proof in [43]) to compute a relaxation of the optimal dynamic wavelet as a regularized eigenvalue problem:

$$\mathbf{x}^* = \arg \min_{\mathbf{x} \in [-1, 1]^{nm}} \frac{\mathbf{x}^\top \mathbf{C} \mathbf{S} \mathbf{C} \mathbf{x}}{\mathbf{x}^\top \mathbf{C} \mathbf{x} + \alpha \mathbf{x}^\top \mathcal{L} \mathbf{x}} \quad (16)$$

where \mathbf{C} and \mathcal{L} are matrices defined in Section 2.1. Optimizations discussed in [42] can be applied to efficiently approximate Equation 16. The resulting algorithm has complexity $O(pn \sum_t |E_t| + qn^2 m^2)$, where p and q are small constants. Similar to Algorithm 1, we apply a sweep procedure to obtain a cut from \mathbf{x}^* .

4 EXPERIMENTS

4.1 Eigenvector Computation

We applied the Lanczos method [16, 25] to implement the algorithms discussed in this paper¹. However, our algorithms can also be implemented using other eigensolvers from the literature.

4.2 Datasets

School is a contact network where vertices represent children from a primary school and edges are created based on proximity detected by sensors [47], with 242 vertices, 17K edges and 3 snapshots. *Stock* is a correlation network of US stocks' end of the day prices² with 500 vertices, 27K edges, and 26 snapshots (one for each year in the interval 1989-2015). *DBLP* is a sample from the DBLP collaboration network. Vertices corresponding to two authors are connected in a given snapshot if they co-authored a paper in the corresponding year. We selected authors who published at least 5 papers one of the following conferences: KDD, CVPR, and FOCS. The resulting temporal network has 3.4K vertices, 16.4K edges, and 4 snapshots.

We also use a synthetic data generator. Its parameters are a graph size n , partition size $k < n$, number of hops h , and noise level $0 \leq \epsilon \leq 1$. Edges are created based on a $\lceil \sqrt{n} \rceil \times \lceil \sqrt{n} \rceil$ grid, where each vertex is connected to its h -hop neighbors. A partition is a sub-grid initialized with $\lceil \sqrt{k} \rceil \times \lceil \sqrt{k} \rceil$ dimensions ($k = n/2$). A value $\pi(v) = 1 + N(0, \epsilon)$ is assigned to vertices inside the partition and the remaining vertices receive iid realizations of a Gaussian $N(0, \epsilon)$. Given the node values, the weight w of an edge (u, v) is

¹Data and code: <https://github.com/arleilps/time-cuts>

²Quandl data: <https://www.quandl.com/data/>

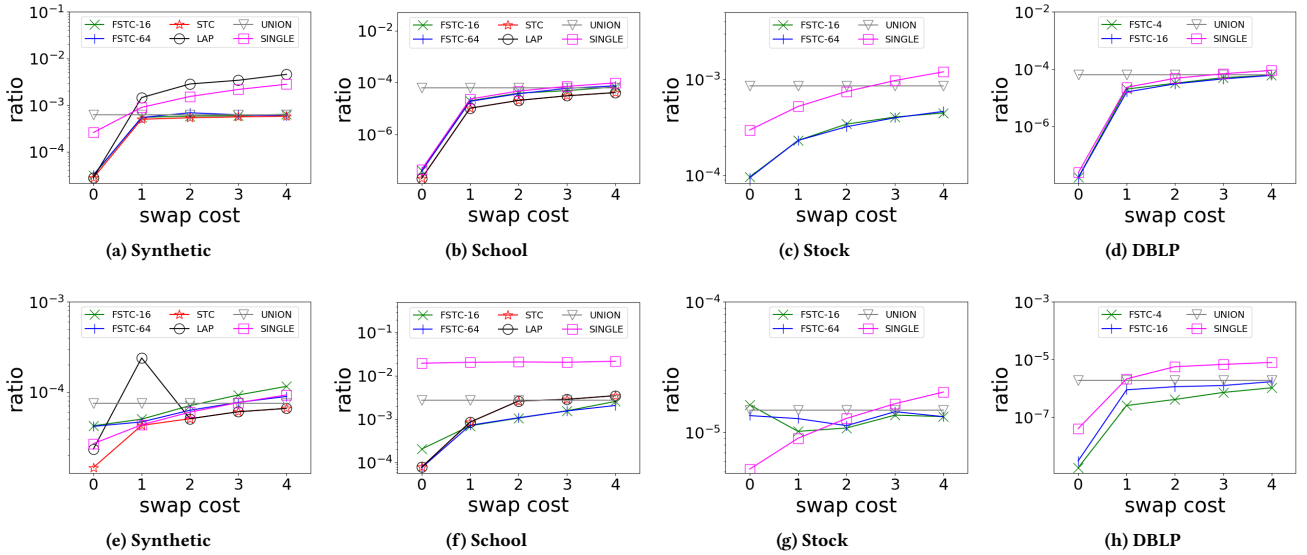


Figure 3: Sparsity ratios for sparsest (a-d) and normalized (e-h) cuts. *STC* achieves the smallest ratio in most of the settings. *FSTC* also achieves good results, specially for $r = 64$, being able to adapt to different swap costs.

set as $\exp(|\pi(v) - \pi(u)|)$. To produce the dynamics, we move the partition along the main diagonal of the grid.

To evaluate our wavelets for dynamic signals, we apply our approach to *Traffic* [42], a road network from California with 100 vertices, 200 edges, and 12 snapshots. Average vehicle speeds measured at the vertices were taken as a dynamic signal for the timespan of a Friday in April, 2011. Moreover, we apply the *heat equation* to generate synthetic signals over the *School* network. Different from *Traffic*, which has a static structure, the resulting dataset (*School-heat*) is dynamic in structure and signal (see details in [43]).

4.3 Approximation and Performance

Two general approaches for computing temporal cuts, for both sparsest and normalized cuts, are evaluated in this section. The first approach, *STC*, combines Theorems 1 and 2, for sparsest and normalized cuts, respectively, and the same rounding scheme applied by Algorithm 1. The second approach, *FSTC-r*, for a rank r , applies the fast approximation described in Section 2.3.

We consider three baselines in this evaluation. *SINGLE* discovers the best cut on each snapshot and then binds them into one temporal cut. *UNION* computes the best average cut over all the snapshots. *LAP* is similar to our approach, but operates directly on the Laplacian matrix \mathcal{L} . Notice that each of these baselines can be applied to either sparsest and normalized cuts as long as the appropriate (standard or normalized) Laplacian matrix is used. Each experiment was repeated 10 times and we report the average results.

Figure 3 shows quality results (sparsity ratios) of the methods. We vary the swap cost (β) within a range that enforces local and global (or stable) patterns. The values of β shown are normalized to integers for ease of comparison. *STC* and *LAP* took too long to finish for the *Stock* and *DBLP* datasets, and thus their results are omitted. Our approach, *STC*, achieves the best results (smallest ratios) in

most of the settings. For the *School* dataset, *LAP* also achieves good results, which is due to the small number of snapshots in the graph. As expected, *UNION* performs well for large swap costs, while *SINGLE* achieves good results when swap costs are close to 0. Our fast approximation (*FSTC*) is able to identify low-sparsity cuts in most of the settings, outperforming *SINGLE* and *UNION*. Notice that even though a larger value of r generates a better approximation for the temporal matrix, as discussed in Section 2.3, the quality of the temporal cut is not guaranteed to increase monotonically with the value of r (see Figure 3g). However, a larger r often leads to a better approximation (i.e. lower sparsity ratio).

Figure 4 shows the performance results (running time) using synthetic data for sparsest (Figure 4a-4d) and normalized (Figures 4e-4h) cuts. We vary the number of vertices, density, number of snapshots, and also the rank of *FSTC*. Similar conclusions can be drawn for both problems. *UNION* is the fastest method, as it operates over an $n \times n$ matrix. *STC* and *LAP*, which process $nm \times nm$ matrices, are the most time consuming methods. *STC* is even slower than *LAP*, due to its denser matrix. *SINGLE* and *FSTC* achieve similar performance, with running times close to *UNION*'s. Figures 4d and 4h illustrate how the rank r of the matrix approximation performed by *FSTC* enables significant performance gains compared to *STC*.

4.4 Community Detection

Dynamic community detection is an interesting application for temporal cuts. Two approaches from the literature, *FacetNet* [34] and *GenLovain* [4], are used as the baselines. We focus our evaluation on *School* and *DBLP*, which have most meaningful communities. The following metrics are considered for comparison:

Cut: Total weight of the edges across partitions computed as $\sum_{t=1}^m \sum_{v,u \in G_t} w(u,v)(1 - \delta(c_v, c_u))$, where c_v and c_u are the partitions to which v and u are assigned, respectively.

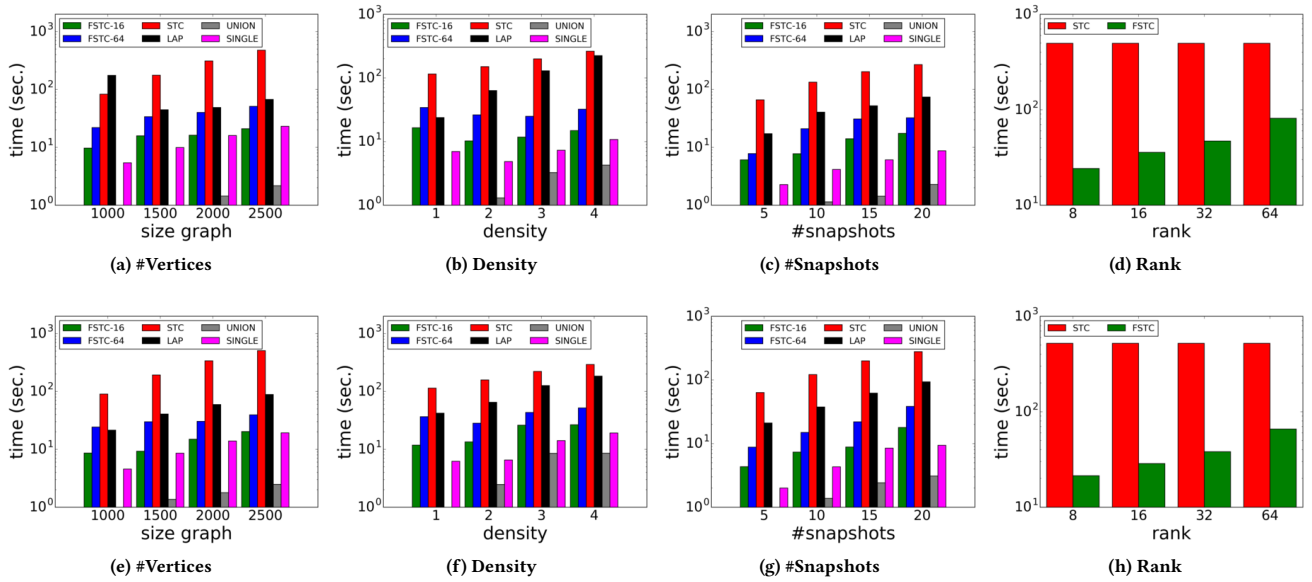


Figure 4: Running time results for sparsest (a-d) and normalized (e-h) cuts on synthetic data.

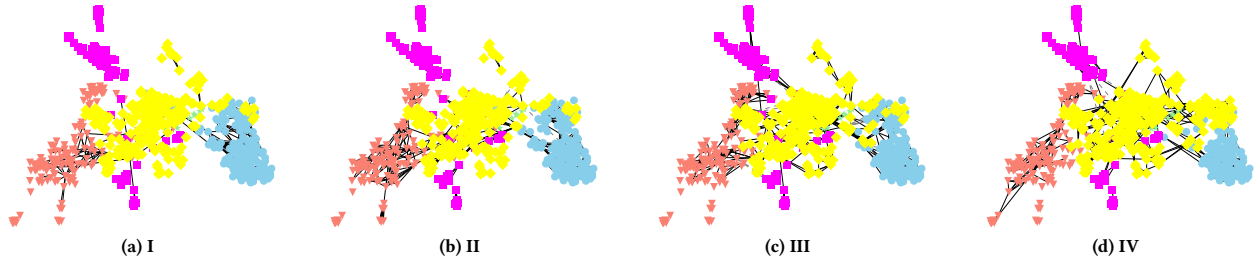


Figure 5: Dynamic communities discovered using *sparsest cuts* for the *DBLP* dataset (4 snapshots). *Better seen in color.*

Sparsity: Sparsity ratio (Equation 3) for k -cuts [40]:

$$\sum_{i=1}^k \frac{\sum_{t=1}^m |(X_{i,t}, \bar{X}_{i,t}| + \beta \sum_{t=1}^{m-1} |(X_{i,t}, \bar{X}_{i,t+1}|)}{\sum_{t=1}^m |X_{i,t}| |\bar{X}_{i,t}|}$$

N-sparsity: Normalized k -cut ratio (similar to sparsity).

Modularity: Temporal modularity, as defined in [4].

Baseline parameters were varied within a range of values and the best results were chosen. For *GenLovain*, we fixed the number of partitions by agglomerating pairs that maximize modularity [4] and for *FacetNet*, we assign each vertex to its highest weight partition.

Community detection results, for 2 and 5 communities, are shown in Table 1. For *School*, both *GenLovain* and our methods found the same communities ($\beta = 0.25$) when $k = 2$, outperforming *FacetNet* in all the metrics. However, for $k = 5$, different communities were discovered by the methods, with *Sparsest* and *Normalized Cuts* achieving the best results in terms of sparsity and n-sparsity, respectively. Our methods also achieve competitive results in terms of modularity. Similar results were found using *DBLP* ($\beta = 0.5$), although *Sparsest* and *Normalized Cuts* switch as the best method for each other's metric in some settings. This is possible because

our algorithms are approximations (i.e. not optimal). We illustrate the communities found in the *School* ($k = 2$) and *DBLP* ($k = 5$) datasets in Figures 1 and 5, respectively.

4.5 Signal Processing on Graphs

We finish our evaluation with the analysis of dynamic signals on graphs. In Figure 6, we illustrate three dynamic wavelets for *Traffic* discovered using our approach under different settings. First, in Figures 6a-6d, we consider cuts that take only the graph signal into account by setting both the regularization parameter α and the smoothness parameter β to 0, which leads to a cut that follows the traffic speeds but has many edges and is not smooth. Next (Figures 6e-6h), we increase α to 200, producing a much sparser cut that is still not smooth. Finally, in Figures 6i-6l, we increase the smoothness β to 10, which forces most of the vertices to remain in the same partition despite of speed variations.

We also evaluate our approach in signal compression, which consists of computing a compact representation for a dynamic signal. As a baseline, we consider the *Graph Fourier* scheme [41] applied to the temporal graph (i.e. the multiplex view of the graph). The

S
(a) School

k	Method	Cut	Sparsity	N-sparsity	Modularity
2	GenLovain	2.6	1.0e-4	5.0e-3	102.0
	Facetnet	6.0	3.8e-4	.012	95.7
	Sparsest	2.6	1.0e-4	5.0e-3	102.0
	Norm.	2.6	1.0e-4	5.0e-3	102.0
5	GenLovain	8.0	6.8e-4	2.7e-2	110.0
	Facetnet	10.0	8.4e-4	3.0e-2	106.0
	Sparsest	8.3	6.4e-4	2.6e-2	109.0
	Norm.	6.1	9.9e-4	1.8e-2	110.0

(b) DBLP

k	Method	Cut	Sparsity	N-sparsity	Modularity
2	GenLovain	80.	3.9e-4	1.3e-5	38,612
	Facetnet	267.0	2.6e-3	8.9e-5	33,091
	Sparsest	9.0	7.6e-5	3.6e-6	38,450
	Norm.	19.0	1.2e-4	3.8e-6	38,516
5	GenLovain	174.	1.3e-3	4.1e-5	39,342
	Facetnet	501.0	7.2e-3	2.8e-4	30,116
	Sparsest	40.0	5.2e-4	6.2e-5	38,498
	Norm.	31.0	4.0e-4	1.0e-5	39,015

Table 1: Community detection results for *Sparsest* and *Normalized Cuts* (and two baselines) using *School* and *DBLP* datasets. Our methods achieve the best results for most of the metrics and are competitive in terms of modularity.

size of the representation (k) is the number of partitions and the number of top eigenvectors for our method and *Graph Fourier*, respectively. Figures 7a and 7b show the compression results in terms of L_2 error using a fixed representation size k for the *Traffic* and *School-heat* datasets, respectively. We vary the value of the regularization parameter α , which controls the impact of the network structure over the wavelets computed, for our method. As expected, a larger value of α leads to a higher L_2 error. However, even for a high regularization, our approach is still able to compute wavelets that accurately compress the signal, outperforming the baseline.

5 CONCLUSION

This paper studied cut problems in temporal graphs. Extensions of two existing graph cut problems, *sparsest* and *normalized cuts*, by enforcing the smoothness of cuts over time, were introduced. To solve these problems, we have proposed spectral approaches based on multiplex graphs by computing relaxed temporal cuts as eigenvectors. Scalable versions of our solutions using divide-and-conquer and low-rank matrix approximation were also presented. In order to compute cuts that take into account also graph signals, we have extended graph wavelets to the dynamic setting. Experiments have shown that our temporal cut algorithms outperform the baseline methods in terms of quality and are competitive in running time. Moreover, temporal cuts enable the discovery of dynamic communities and the analysis of dynamic graph processes.

This work opens several lines for investigation: (i) temporal cuts can be applied to many scenarios other than the ones considered in this paper (e.g., computer vision); (ii) Perturbation Theory can

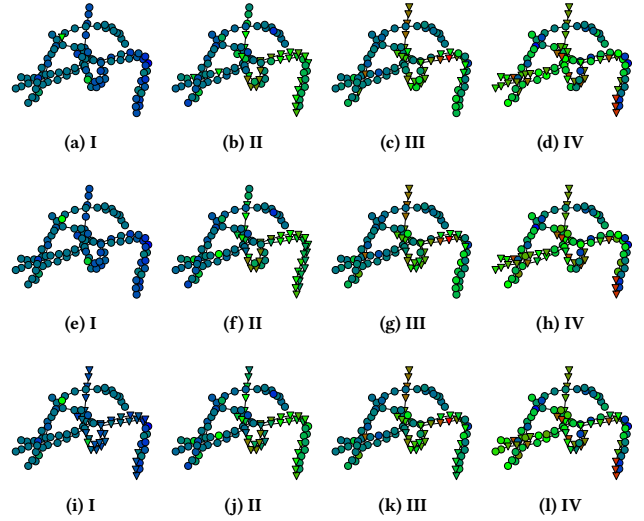


Figure 6: Wavelet cut of a 4-snapshot dynamic traffic network with vehicle speeds as a signal. Vertex colors correspond to speed values (red for high and blue for low) and shapes indicate the partitions for 3 different settings: $\alpha = 0$, and $\beta = 1$ (a-d, no network effect), $\alpha = 200$, and $\beta = 1$. (e-h, large network effect with low smoothness), and $\alpha = 200$, and $\beta = 10$. (i-l, large network effect and high smoothness) .

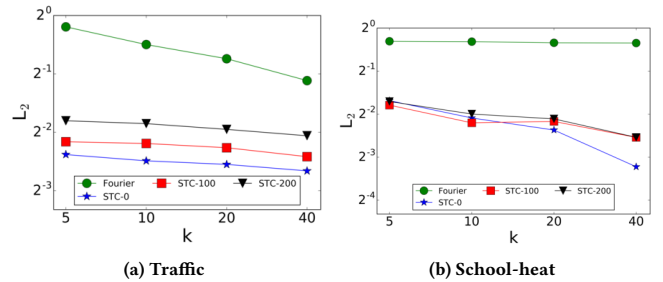


Figure 7: L_2 error with different representation sizes k for *Graph Fourier* and our approach while setting the regularization parameter α to 200, 100, and 0.

support fast updates for temporal cuts in graph streams [48]; finally, (iii) we want to investigate the relationship between cuts and random-walks on temporal graphs [13].

Acknowledgment. Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053 (the ARL Network Science CTA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

REFERENCES

- [1] William Anderson Jr and Thomas Morley. 1985. Eigenvalues of the Laplacian of a graph. *Linear and multilinear algebra* 18, 2 (1985), 141–145.
- [2] Sanjeev Arora, Satish Rao, and Umesh Vazirani. 2009. Expander flows, geometric embeddings and graph partitioning. *J. ACM* 56, 2 (2009), 5.
- [3] Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. 2006. Group formation in large social networks: membership, growth, and evolution. In *KDD*. ACM, New York, NY, USA, 44–54.
- [4] Marya Bazzi, Mason A Porter, Stacy Williams, Mark McDonald, Daniel J Fenn, and Sam D Howison. 2016. Community detection in temporal multilayer networks, with an application to correlation networks. *Multiscale Modeling & Simulation* 14, 1 (2016), 1–41.
- [5] Steffen Bickel and Tobias Scheffer. 2004. Multi-view clustering. In *ICDM*. IEEE, Washington, DC, USA, 19–26.
- [6] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine* 34, 4 (2017), 18–42.
- [7] Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. 2004. Incremental clustering and dynamic information retrieval. *SIAM J. Comput.* 33, 6 (2004), 1417–1440.
- [8] Yun Chi, Xiaodan Song, Dengyong Zhou, Koji Hino, and Belle L Tseng. 2007. Evolutionary spectral clustering by incorporating temporal smoothness. In *KDD*. ACM, New York, NY, USA, 153–162.
- [9] Fan RK Chung. 1997. *Spectral graph theory*. American Mathematical Society, Providence, RI, USA.
- [10] Mihai Cucuringu, Ioannis Koutis, Sanjay Chawla, Gary Miller, and Richard Peng. 2016. Simple and Scalable Constrained Clustering: a Generalized Spectral Method. In *AISTATS*. PMLR, Cadiz, Spain, 445–454.
- [11] JJM Cuppen. 1980. A divide and conquer method for the symmetric tridiagonal eigenproblem. *Numer. Math.* 36, 2 (1980), 177–195.
- [12] Thomas Erlebach, Michael Hoffmann, and Frank Kammer. 2015. On temporal graph exploration. In *ICALP*. Springer, Berlin, Heidelberg, 444–455.
- [13] Daniel Figueiredo, Philippe Nain, Bruno Ribeiro, Edmundo de Souza e Silva, and Don Towsley. 2012. Characterizing continuous time random walks on time varying graphs. In *SIGMETRICS*. ACM, New York, NY, USA, 307–318.
- [14] Wilfried N Gansterer, Robert C Ward, Richard P Muller, and William A Goddard III. 2003. Computing approximate eigenpairs of symmetric block tridiagonal matrices. *SIAM Journal on Scientific Computing* 25, 1 (2003), 65–85.
- [15] Matan Gavish, Boaz Nadler, and Ronald Coifman. 2010. Multiscale Wavelets on Trees, Graphs and High Dimensional Data: Theory and Applications to Semi Supervised Learning. In *ICML*. Omnipress, USA, 367–374.
- [16] Gene H Golub and Charles F Van Loan. 2012. *Matrix computations*. Vol. 3. JHU Press, Baltimore, MD, USA.
- [17] Sergio Gomez, Albert Diaz-Guilera, Jesus Gomez-Gardenes, Conrad J Perez-Vicente, Yamir Moreno, and Alex Arenas. 2013. Diffusion dynamics on multiplex networks. *Physical review letters* 110, 2 (2013), 028701.
- [18] Lars Hagen and Andrew B Kahng. 1992. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems* 11, 9 (1992), 1074–1085.
- [19] Christopher J Hillar and Lek-Heng Lim. 2013. Most tensor problems are NP-hard. *Journal of the ACM (JACM)* 60, 6 (2013), 45.
- [20] Roger A Horn and Charles R Johnson. 1990. *Matrix analysis*. Cambridge University Press, New York, NY, USA.
- [21] Vikas Kawadia and Sameet Sreenivasan. 2012. Sequential detection of temporal communities by estrangement confinement. *Scientific reports* 2 (2012), 794–794.
- [22] Mikko Kivela, Alex Arenas, Marc Barthélemy, James P Gleeson, Yamir Moreno, and Mason A Porter. 2014. Multilayer networks. *Journal of complex networks* 2, 3 (2014), 203–271.
- [23] Ioannis Koutis, Gary Miller, and Richard Peng. 2014. A Generalized Cheeger Inequality. <https://arxiv.org/abs/1412.6075>. (2014).
- [24] Ioannis Koutis, Gary L. Miller, and Richard Peng. 2011. A Nearly-m Log N Time Solver for SDD Linear Systems. In *FOCS*. IEEE, Washington, DC, USA, 590–598.
- [25] J Kuczynski and H Woźniakowski. 1992. Estimating the largest eigenvalue by the power and Lanczos algorithms with a random start. *SIAM journal on matrix analysis and applications* 13, 4 (1992), 1094–1122.
- [26] Ravi Kumar, Jasmine Novak, Prabhakar Raghavan, and Andrew Tomkins. 2005. On the bursty evolution of blogspace. *World Wide Web* 8, 2 (2005), 159–178.
- [27] John Lafferty and Guy Lebanon. 2005. Diffusion Kernels on Statistical Manifolds. *JMLR* 6 (2005), 129–163.
- [28] Kevin Lang. 2006. Fixing two weaknesses of the spectral method. In *NIPS*. MIT Press, Cambridge, MA, USA, 715–722.
- [29] Amy N Langville and Carl D Meyer. 2011. *Google’s PageRank and beyond: The science of search engine rankings*. Princeton University Press, Princeton, New Jersey.
- [30] James R Lee, Shayan Oveis Gharan, and Luca Trevisan. 2014. Multiway spectral partitioning and higher-order cheeger inequalities. *J. ACM* 61, 6 (2014), 1–30.
- [31] Tom Leighton and Satish Rao. 1988. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *FOCS*. IEEE, White Plains, NY, 422–431.
- [32] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. 2009. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics* 6, 1 (2009), 29–123.
- [33] Yifan Li, Jiawei Han, and Jiong Yang. 2004. Clustering Moving Objects. In *KDD*. ACM, New York, NY, USA, 617–622.
- [34] Yu-Ru Lin, Yun Chi, Shenghuo Zhu, Hari Sundaram, and Belle L. Tseng. 2008. Facetnet: A Framework for Analyzing Communities and Their Evolutions in Dynamic Networks. In *WWW*. ACM, New York, NY, USA, 685–694.
- [35] Anand Louis, Prasad Raghavendra, Prasad Tetali, and Santosh Vempala. 2012. Many Sparse Cuts via Higher Eigenvalues. In *STOC*. ACM, New York, NY, USA, 1131–1140.
- [36] Othon Michail. 2016. An introduction to temporal graphs: An algorithmic perspective. *Internet Mathematics* 12, 4 (2016), 239–280.
- [37] Peter J Mucha, Thomas Richardson, Kevin Macon, Mason A Porter, and Jukka-Pekka Onnela. 2010. Community structure in time-dependent, multiscale, and multiplex networks. *Science* 328, 5980 (2010), 876–878.
- [38] Huazhong Ning, Wei Xu, Yun Chi, Yihong Gong, and Thomas S Huang. 2010. Incremental spectral clustering by efficiently updating the eigen-system. *Pattern Recognition* 43, 1 (2010), 113–127.
- [39] James Rosswog and Kanad Ghose. 2008. Detecting and Tracking Spatio-temporal Clusters with Adaptive History Filtering. In *ICDMW*. IEEE, Washington, DC, USA, 448–457.
- [40] Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (2000), 888–905.
- [41] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. 2013. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine* 30, 3 (2013), 83–98.
- [42] Arlei Silva, Xuan Hong Dang, Prithwish Basu, Ambuj Singh, and Ananthram Swami. 2016. Graph Wavelets via Sparse Cuts. In *KDD*. ACM, New York, NY, USA, 1175–1184.
- [43] Arlei Silva, Ambuj Singh, and Ananthram Swami. 2017. Spectral Algorithms for Temporal Graph Cuts. <http://arxiv.org/abs/1702.04746.pdf>. (2017).
- [44] Albert Sole-Ribalta, Manlio De Domenico, Nikos E Kouvaris, Albert Diaz-Guilera, Sergio Gomez, and Alex Arenas. 2013. Spectral properties of the Laplacian of multiplex networks. *Physical Review E* 88, 3 (2013), 032807.
- [45] Daniel A. Spielman and Shang-Hua Teng. 2004. Nearly-linear Time Algorithms for Graph Partitioning, Graph Sparsification, and Solving Linear Systems. In *STOC*. ACM, New York, NY, USA, 81–90.
- [46] Daniel A Spielman and Shang-Hua Teng. 2011. Spectral sparsification of graphs. *SIAM J. Comput.* 40, 4 (2011), 981–1025.
- [47] Juliette Stehlé, Nicolas Voirin, Alain Barrat, Ciro Cattuto, Lorenzo Isella, Jean-François Pinton, Marco Quaghiotto, Wouter Van den Broeck, Corinne Régis, Bruno Lina, and others. 2011. High-resolution measurements of face-to-face contact patterns in a primary school. *PLoS one* 6, 8 (2011), e23176.
- [48] Dane Taylor, Sean A Myers, Aaron Clauset, Mason A Porter, and Peter J Mucha. 2017. Eigenvector-based centrality measures for temporal networks. *Multiscale Modeling & Simulation* 15, 1 (2017), 537–574.
- [49] Luca Trevisan. 2013. Is Cheeger-type Approximation Possible for Nonuniform Sparsest Cut? <https://arxiv.org/pdf/1303.2730.pdf>. (2013).
- [50] Chang Xu, Dacheng Tao, and Chao Xu. 2013. A survey on multi-view learning. <https://arxiv.org/pdf/1304.5634.pdf>. (2013).