Fairness-Aware Clique-Preserving Spectral Clustering of Temporal Graphs

Dongqi Fu
University of Illinois at Urbana-Champaign
Illinois, USA
dongqif2@illinois.edu

Dawei Zhou
Virginia Tech
Virginia, USA
zhoud@vt.edu

Ross Maciejewski
Arizona State University
Arizona, USA
rmacieje@asu.edu

Arie Croitoru
George Mason University
Virginia, USA
acroitor@gmu.edu

Marcus Boyd
University of Maryland, College Park
Maryland, USA
boydma@umd.edu

Jingrui He
University of Illinois at Urbana-Champaign
Illinois, USA
jingrui@illinois.edu

ABSTRACT

With the widespread development of algorithmic fairness, there has been a surge of research interest that aims to generalize the fairness notions from the attributed data to the relational data (graphs). The vast majority of existing work considers the fairness measure in terms of the low-order connectivity patterns (e.g., edges), while overlooking the higher-order patterns (e.g., k-cliques) and the dynamic nature of real-world graphs. For example, preserving triangles from graph cuts during clustering is the key to detecting compact communities; however, if the clustering algorithm only pays attention to triangle-based compactness, then the returned communities lose the fairness guarantee for each group in the graph. Furthermore, in practice, when the graph (e.g., social networks) topology constantly changes over time, one natural question is how can we ensure the compactness and demographic parity at each timestamp efficiently. To address these problems, we start from the static setting and propose a spectral method that preserves clique connections and incorporates demographic fairness constraints in returned clusters at the same time. To make this static method fit for the dynamic setting, we propose two core techniques, *Laplacian Update via Edge Filtering and Searching* and *Eigen-Pairs Update with Singularity Avoided*. Finally, all proposed components are combined into an end-to-end clustering framework named F-SEGA, and we conduct extensive experiments to demonstrate the effectiveness, efficiency, and robustness of F-SEGA.

CCS CONCEPTS

• **Mathematics of computing → Graph algorithms**; • **Information systems → Clustering**.

WWW '23, April 30–May 04, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583423

KEYWORDS

Spectral Clustering, Fairness, Clique Patterns, Temporal Graphs

ACM Reference Format:
Dongqi Fu, Dawei Zhou, Ross Maciejewski, Arie Croitoru, Marcus Boyd, and Jingrui He. 2023. Fairness-Aware Clique-Preserving Spectral Clustering of Temporal Graphs. In *Proceedings of the ACM Web Conference 2023 (WWW '23), April 30–May 04, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3543507.3583423

1 INTRODUCTION

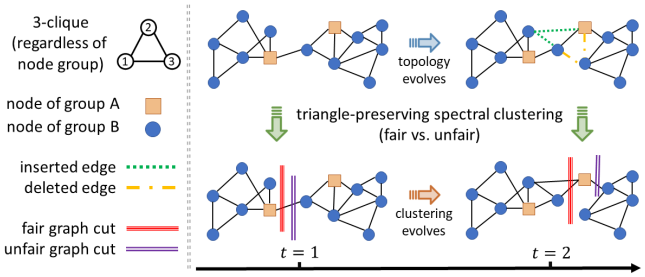


Figure 1: Evolving Structure breaks the Fairness during the Continuous Clustering. At $t = 0$, the unfair cut (purple) aligns with the fair cut (red); then, at $t = 1$, unfair communities are discovered for only considering the cluster compactness but ignoring the demographic fairness.

Nowadays, for the applications that require fair clustering, research interests are devoted to designing various fair objectives [6, 11, 24, 26, 27, 29, 32], to ensure the fairness for the social good. For example, based on relational data, the fairness constraints are established to guide the spectral clustering such that the demographics of each detected community are proportional to the whole distribution [29]. But the majority of existing graph clustering algorithms consider the fairness constraints in terms of the low-order connectivity patterns (e.g., edges) while overlooking the higher-order patterns (e.g., k-cliques). Actually, clique density is a key factor in many graph clustering applications. For example, k-clique preserved clusters (especially $k = 3$) play a fundamental role in understanding the social network structures [15, 23, 30, 39], as well as in identifying protein complexes and discovering new modules [1, 48].

Therefore, designing fairness constraints (e.g., proportional demographics) for high-order (e.g., clique-preserving) clustering has a huge application potential but largely remains nascent. For example, in the citation network or co-author network, the high-order clusters can be used for the expert team formation [31], but the dense connections usually occur at the intra-subject level. Without proportional demographics, the formed team could not handle interdisciplinary tasks requiring diverse backgrounds. Another example is triangle-preserved communities, which are suited for the community-driven recommendations [43]. However, when these communities are tasked with voting, without proportional demographics, the voice of different groups, especially minority groups, can barely be heard.

When we focus on clique-preserving dense clusters, it is not clear how to guarantee fairness at the same time. Furthermore, when the graph structure evolves [2, 18, 20, 28] , it can be even more challenging to ensure the clustering compactness and demographic parity simultaneously (e.g., evolving structures can break previously obtained fairness). Figure 1 illustrates the difficulty of achieving demographic-fair and triangle-preserving clustering on evolving graphs: even if the initial high-order clustering is demographically fair, as structure evolves, the fairness can be broken if the cluster compactness is the only objective along with time. Moreover, the straightforward application of (high-order) spectral clustering methods [5, 41, 51] can be computationally prohibitive, and the additional fairness and frequently updated graph structures may further render the existing computational resources inadequate. The core problem is how to maintain fair and dense clusters effectively and efficiently when the underlying structure is evolving.

Therefore, we need to model the two separate objectives (i.e., demographic fairness and clustering compactness) in a unified spectral clustering framework. Then, efficient and effective techniques are indispensable for adapting this framework to the dynamic setting. In this paper, we start from the static setting and propose a fairness-aware clique-preserving spectral clustering method, which preserves clique connections from graph cuts and guides the clustering results to be fair, i.e., the demographics of each cluster are close to the entire graph. Then, to adapt to the dynamic setting, we propose two core components: (1) *Laplacian Update via Edge Filtering and Searching*, and (2) *Eigen-Pairs Update with Singularity Avoided*. Thus, demographic-fair and clique-dense clusters under new arrival graph structures can be incrementally tracked instead of solving them from scratch. Finally, we combine the proposed techniques into an end-to-end clustering framework named F-SEGA.

Our main contributions are summarized as follows.

- **Problem**: We unify the problem of fairness-aware and clique-preserving spectral clustering, and extend it to dynamic graphs.
- **Method**: We propose a solution, F-SEGA, for demographic-fair and clique-dense dynamic clustering with theoretical analysis.
- **Evaluation**: We design extensive experiments to demonstrate the effectiveness, efficiency, and robustness of F-SEGA.
- **Application**: We identify the connection of F-SEGA to real-world applications by designing a case study of dynamically and proportionally allocating human resources.

The problem of fairness-aware and clique-preserving clustering of temporal graphs is defined in Section 2. In Section 3, we introduce the static clustering method and corresponding techniques to adapt it to the dynamic setting. Then we present our F-SEGA clustering framework in Section 4. Experimental results are provided in Section 5. Finally, we review the related work in Section 6 before we conclude the paper in Section 7.

Table 1: Table of Notation

Symbol	Definition or Description
$\mathcal{G}$	temporal graph $\mathcal{G} = \{G^{(1)}, G^{(2)}, \dots, G^{(T)}\}$
V	set of each ever-appeared node in graph $\mathcal{G}$
h	number of groups in graph $\mathcal{G}$
V_s	set of nodes belonging to the s-th group of $\mathcal{G}$, $s \in \{1, \dots, h\}$
$\boldsymbol{F}$	member-group matrix of graph $\mathcal{G}$
$G^{(t)}$	snapshot graph at time t
$C_l^{(t)}$	the l-th cluster of $G^{(t)}$
$\Delta E^{(t)}$	updated edge set at time t changing $G^{(t)}$ into $G^{(t+1)}$
$\boldsymbol{A}^{(t)}$	standard (edge-based) adjacency matrix of $G^{(t)}$
$\boldsymbol{W}^{(t)}$	clique-weighted adjacency matrix of $G^{(t)}$
$\boldsymbol{D}^{(t)}$	clique-weighted degree matrix of $G^{(t)}$
$\boldsymbol{L}^{(t)}$	clique-weighted Laplacian matrix of $G^{(t)}$
$\boldsymbol{M}^{(t)}$	fairness-constrained clique-weighted Laplacian of $G^{(t)}$

2 PRELIMINARY AND PROBLEM DEFINITION

Throughout this paper, we use lower-case letters for scalars (e.g., α), upper-case letters for sets (e.g., V), bold lower-case letters for column vectors (e.g., $\boldsymbol{x}$), and bold upper-case letters for matrices (e.g., $\boldsymbol{A}$). We follow the matrix indexing, i.e., $A(i,:)$ denotes the i-th row of $\boldsymbol{A}$, and use the parenthesized superscript to denote the timestamp (e.g., $G^{(t)}$). We use $^\top$ to denote the matrix transpose.

Cliques. A k-clique is a complete subgraph consisting of k nodes, and each pair of nodes are connected with an edge. For instance, an edge is a 2-clique, and a triangle is a 3-clique.

Clique-Preserving Normalized Cut. Normalized cut ($Ncut$) [41] measures the compactness of the resulting clusters regarding edge connections. A small $Ncut$ indicates a good partition where many edges are preserved in clusters. Here, we define the Clique-Preserving $Ncut$ ($CPNcut$) to cover any k-cliques.

$$CPNcut(C_1, \dots, C_q, \mathbb{N}) = \sum_{i=1}^{q} \frac{cut(C_i, V \setminus C_i, \mathbb{N})}{\mu(C_i, \mathbb{N})} \tag{1}$$

where $\mathbb{N}$ denotes the target k-clique to be preserved from partitions, $cut(C, V \setminus C_i, \mathbb{N})$ is the number of broken cliques for partitioning graph G into cluster C_i and its complement $V \setminus C_i$, and $\mu(C_i, \mathbb{N})$ denotes the number of instances of cliques within C_i. When $k = 2$, the defined $CPNcut$ is $Ncut$; when $k \geq 3$, $CPNcut$ measures the compactness of high-order clique connections.

Demographic Fairness Constraints. A clustering is fair if the demographics of each cluster are close to the demographics of the whole graph [14, 29], which is expressed as follows.

$$\forall s \in \{1, \dots, h\}, \ \frac{|V_s \bigcap C_i|}{|C_i|} = \frac{|V_s|}{|V|} \tag{2}$$

where V_s stands for the set of nodes of the s-th group when the entire set of nodes $V = \bigcup_{s=1}^{h} V_s$ has h different groups in total, and C_i is the i-th cluster produced by a clustering method.

Graph Arrival Model. We consider a temporal graph as a sequence of snapshots, $\mathcal{G} = \{G^{(1)}, G^{(2)}, \dots, G^{(T)}\}$, where $G^{(t)} = (V^{(t)}, E^{(t)})$ is undirected and unweighted, $V^{(t)}$ and $E^{(t)}$ represent the set of nodes and edges at timestamp t respectively. Following [9, 42], an inserted (or deleted) node at timestamp t is regarded

as an existing dangling node at all previous timestamps (or all future timestamps). Thus, we denote $|V^{(t)}| = |V| = n$, i.e., $V^{(t)}$ consists of each appeared node in the whole life of $\mathcal{G}$, such that the dimension of affinity matrices over time is consistent. Also, we assume the demographic group information of a node remains over time [29], $V_s^{(t)} = V_s^{(t+1)}$, e.g., a male user in the social network always belongs to the male group as the graph structure evolves. Between two consecutive snapshots, we define the set of updated edges as $\Delta E^{(t)} = \Delta E_+^{(t)} \cup \Delta E_-^{(t)} = \{E^{(t+1)} \setminus E^{(t)}\} \cup \{E^{(t)} \setminus E^{(t+1)}\}$, where $\Delta E_+^{(t)}$ and $\Delta E_-^{(t)}$ denote the inserted and deleted edges at timestamp t to change $G^{(t)}$ into $G^{(t+1)}$.

Our goal is to partition the temporal graph $\mathcal{G}$ while largely preserving user-specified k-cliques from fair cuts through all given timestamps. To be specific, at timestamp t, we aim to partition graph $G^{(t)}$ into q disjoint clusters $\bigcup_{i=1}^{q} C_i^{(t)} = V$ and $C_i^{(t)} \cap C_j^{(t)} = \emptyset$ with $i \neq j \in \{1, 2, \ldots, q\}$ via the following objective.

$$\min_{C_i^{(t)}} \sum_{t=1}^{T} CPNcut(C_1^{(t)}, \ldots, C_q^{(t)}, \mathbb{N}) \tag{3}$$

s.t.

$$\forall s \in \{1, \ldots, h\} : \frac{|V_s \bigcap C_i^{(t)}|}{|C_i^{(t)}|} = \frac{|V_s|}{|V|}, t \in \{1, \ldots, T\} \tag{4}$$

Our problem can be formally defined as follows.

Problem. *Fairness-Aware Clique-Preserving Clustering of Temporal Graphs*

Input: *(i) a temporal graph $\mathcal{G} = \{G^{(1)}, \ldots, G^{(T)}\}$, (ii) the number of clusters q, and (iii) a k-clique $\mathbb{N}$.*

Output: *clusters $\{C_1^{(t)}, \ldots C_q^{(t)}\}$ for $t \in \{1, \ldots, T\}$ satisfying clique-dense and demographic-fair objectives (Eq. 3 and Eq. 4).*

3 PROPOSED TECHNIQUES

We start with the static setting and explain how to obtain the fairness-aware clique-preserving clustering (Section 3.1). Then we adapt this static method to the dynamic setting via two components: Laplacian Update via Edge Filtering and Searching (Section 3.2), and Eigen-Pairs Update with Singularity Avoided (Section 3.3).

3.1 Fairness-Aware Clique-Preserving Spectral Clustering: *From Static to Dynamic*

Clique-Preserving Spectral Clustering. Traditional spectral clustering (e.g., [41]) is a special case of minimizing $CPNcut$ if the target clique $\mathbb{N}$ is set to be an edge ($k = 2$). To accommodate high-order ($k \geq 3$) clique patterns, $CPNcut$ (i.e., Eq. 1) is instantiated as follows.

$$\begin{aligned} CPNcut(C_1, \ldots, C_q, \mathbb{N}) &= \sum_{l=1}^{q} \frac{cut(C_l, V \setminus C_l, \mathbb{N})}{\mu(C_l, \mathbb{N})} \text{ w.r.t} \\ cut(C_l, V \setminus C_l, \mathbb{N}) &= \sum_{i \in C_l, j \in V \setminus C_l} W(i, j)\,, \\ \mu(C_l, \mathbb{N}) &= \sum_{i \in C_l, j \in V} W(i, j) \end{aligned} \tag{5}$$

where $\boldsymbol{W} \in \mathbb{R}^{n \times n}$ is the clique-weighted adjacency matrix, n is the number of nodes, and $W(i, j)$ is the number of instances of clique $\mathbb{N}$ containing edge (i, j). For example, if $k = 3$ and edge (i, j) makes up 4 different triangles, then $W(i, j) = 4$.

For a clustering result $\bigcup_{l=1}^{q} C_l$, if we represent it by a node-cluster assignment matrix $\boldsymbol{H} \in \mathbb{R}^{n \times q}$ as follows,

$$H(i, l) = \begin{cases} 1/\sqrt{\mu(C_l, \mathbb{N})} & i \in C_l \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

then we will get $CPNcut(C_1, \ldots, C_q, \mathbb{N}) = Tr(\boldsymbol{H}^\top \boldsymbol{L} \boldsymbol{H})$, where $\boldsymbol{H}^\top$ is the transpose of matrix $\boldsymbol{H}$, $Tr(\cdot)$ denotes the trace of a matrix, $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W} \in \mathbb{R}^{n \times n}$ is the Laplacian matrix, $\boldsymbol{D}$ is the clique-weighted degree matrix, i.e., $\boldsymbol{D} = diag(\boldsymbol{W}\boldsymbol{e})$, and $\boldsymbol{e}$ is the vector with all ones[1].

Therefore, minimizing $CPNcut$ equals to the trace minimization problem, i.e., $\min CPNcut(C_1, \ldots, C_q, \mathbb{N}) = \min Tr(\boldsymbol{H}^\top \boldsymbol{L} \boldsymbol{H})$, with $\boldsymbol{H}$ defined in Eq. 6. Since this minimization problem is discrete and proven NP-complete [41], we relax this problem by allowing the entries of matrix $\boldsymbol{H}$ to take any real values. Then the trace minimization objective is rewritten as follows.

$$\min_{\boldsymbol{H} \in \mathbb{R}^{n \times q}} Tr(\boldsymbol{H}^\top \boldsymbol{L} \boldsymbol{H}) \text{ s.t. } \boldsymbol{H}^\top \boldsymbol{D} \boldsymbol{H} = \boldsymbol{I} \tag{7}$$

Substituting $\boldsymbol{H} = \boldsymbol{D}^{-1/2}\boldsymbol{T}$ with $\boldsymbol{T} \in \mathbb{R}^{n \times q}$, we have

$$\min_{\boldsymbol{T} \in \mathbb{R}^{n \times q}} Tr(\boldsymbol{T}^\top \boldsymbol{D}^{-1/2} \boldsymbol{L} \boldsymbol{D}^{-1/2} \boldsymbol{T}) \text{ s.t. } \boldsymbol{T}^\top \boldsymbol{T} = \boldsymbol{I} \tag{8}$$

Then the trace minimization problem (i.e., Eq. 8) for finding the optimal $\boldsymbol{T}$ can be solved by Rayleigh-Ritz theorem (Section 5.2.2(6) in [33]), which computes the q smallest eigenvalues of $\boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2}$ and stores the corresponding eigenvectors as columns of the optimal $\boldsymbol{T}$. To infer the clustering result from $\boldsymbol{H} = \boldsymbol{D}^{-1/2}\boldsymbol{T}$, we can apply the K-means algorithm [47] on the rows of $\boldsymbol{H}$.

Fairness Constraints on Clique-Preserving Clusters. To make the clique-preserving spectral clustering demographic fair, we need to add the fairness constraint (i.e., Eq. 4) to the clique-preserving clustering. We first derive Eq. 4 as follows.

$$\begin{aligned} &\forall s \in \{1, \ldots, h\} : \frac{|V_s \bigcap C_l|}{|C_l|} = \frac{|V_s|}{|V|} \\ \Leftrightarrow\ &\forall s \in \{1, \ldots, h-1\} : \sum_{i=1}^{n} \left(f_s(i) - \frac{|V_s|}{n} \right) H(i, l) = 0 \end{aligned} \tag{9}$$

where $\boldsymbol{f}_s \in \mathbb{R}^n$ is the group-membership vector: if node i belongs to the group s, then $f_s(i) = 1$; otherwise, $f_s(i) = 0$. Eq. 9 can be proven by replacing $H(i, l)$ with Eq. 6. With the fairness constraint, i.e., Eq. 9, the objective of clique-preserving spectral clustering (i.e., Eq. 7) can be extended as follows.

$$\min_{\boldsymbol{H} \in \mathbb{R}^{n \times q}} Tr(\boldsymbol{H}^\top \boldsymbol{L} \boldsymbol{H}) \text{ s.t. } \boldsymbol{H}^\top \boldsymbol{D} \boldsymbol{H} = \boldsymbol{I} \text{ and } \boldsymbol{F}^\top \boldsymbol{H} = \boldsymbol{0} \tag{10}$$

where $\boldsymbol{F} \in \mathbb{R}^{n \times (h-1)}$ is the group-membership matrix with column vectors $\boldsymbol{f}_s - \frac{|V_s|}{n} \cdot \boldsymbol{e}$, and $\boldsymbol{0}$ stands for the zero matrix.

To solve Eq. 10 by Rayleigh-Ritz theorem [33], we substitute $\boldsymbol{H} = \boldsymbol{Z}\boldsymbol{Q}^{-1}\boldsymbol{X}$ for $\boldsymbol{Z} \in \mathbb{R}^{n \times (n-h+1)}$ with $\boldsymbol{Q} \in \mathbb{R}^{(n-h+1) \times (n-h+1)}$

$$\min_{\boldsymbol{X} \in \mathbb{R}^{(n-h+1) \times q}} Tr(\boldsymbol{X}^\top \boldsymbol{Q}^{-1} \boldsymbol{Z}^\top \boldsymbol{L} \boldsymbol{Z} \boldsymbol{Q}^{-1} \boldsymbol{X}) \text{ s.t. } \boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{I} \tag{11}$$

where $\boldsymbol{Z}$ is the matrix whose columns are the orthonormal basis of the nullspace of $\boldsymbol{F}^\top$, and $\boldsymbol{Q}^2 = \boldsymbol{Z}^\top \boldsymbol{D} \boldsymbol{Z}$. Thus, $\boldsymbol{Z}$ and $\boldsymbol{Q}$ are directly solvable from $\boldsymbol{F}$ and $\boldsymbol{D}$. Similarly, we first need to compute q smallest eigenvalues[2] of matrix $\boldsymbol{Q}^{-1}\boldsymbol{Z}^\top \boldsymbol{L} \boldsymbol{Z} \boldsymbol{Q}^{-1}$, and store the corresponding

[1] $diag(\cdot)$ is defined as the standard diagonalize operation.
[2] We assume $q \leq n - h + 1$ for valid solutions.

eigenvectors as columns of the optimal $\boldsymbol{X}$; then we infer a clique-preserving and demographic-fair clustering from $\boldsymbol{H} = \boldsymbol{Z}\boldsymbol{Q}^{-1}\boldsymbol{X}$ by K-means. For notation clarity, we denote the matrix $\boldsymbol{M}$ as follows and name it *fairness-constrained clique-weighted Laplacian matrix*.

$$\boldsymbol{M} = \boldsymbol{Q}^{-1}\boldsymbol{Z}^{\top}\boldsymbol{L}\boldsymbol{Z}\boldsymbol{Q}^{-1} \in \mathbb{R}^{(n-h+1)\times(n-h+1)} \tag{12}$$

Challenges from the Dynamic Setting. To extend the unified objective (i.e., Eq. 10) to the dynamic setting, two challenges need to be addressed when the input graph is evolving over time.

- Updated edges change the graph structure. Instead of rebuilding $\boldsymbol{M}^{(t+1)}$ from scratch every single timestamp, we need to identify the unchanged structures and update the outdated structures inside $\boldsymbol{M}^{(t)}$ for efficiency (Section 3.2).
- Instead of solving eigenvalues and eigenvectors (i.e., eigen-pairs) of $\boldsymbol{M}^{(t+1)}$ from scratch every single timestamp, we need to compute them in a fast manner. Moreover, we also need to eliminate accumulated tracking errors, if any (Section 3.3).

3.2 Laplacian Update via Edge Filtering and Searching

When the new graph arrives, updated edge set $\Delta E^{(t)}$ changes the fairness-constrained Laplacian matrix $\boldsymbol{M}^{(t)}$ into $\boldsymbol{M}^{(t+1)}$. In Section 3.1, $\boldsymbol{M} = \boldsymbol{Q}^{-1}\boldsymbol{Z}^{\top}\boldsymbol{L}\boldsymbol{Z}\boldsymbol{Q}^{-1}$, where $\boldsymbol{Z}$ and $\boldsymbol{Q}$ depend on the member-group matrix $\boldsymbol{F}$. Since we assume the demographic group membership of each node remains the same, the Laplacian matrix $\boldsymbol{L}^{(t)} = \boldsymbol{D}^{(t)} - \boldsymbol{W}^{(t)}$ is the changing part of $\boldsymbol{M}^{(t)}$. To update $\boldsymbol{L}^{(t)}$, we focus on updating $\boldsymbol{W}^{(t+1)}$, i.e., clique-weighted adjacency matrix.

Updating the corresponding matrices between two consecutive timestamps determines the time complexity of many temporal graph algorithms [17, 19, 21, 25, 44]. According to [21], in the worst case, updating our $\boldsymbol{W}^{(t)}$ for one single updated edge will cost $O(n^{k-2})$, where n is the number of nodes in the graph $\mathcal{G}$, and k is the number of nodes in the target clique $\mathbb{N}$. However, not every updated edge will change the previous spectral clustering result, and we call such an edge *insensitive updated edge*, otherwise, we regard it as a *sensitive updated edge*. Hence, we aim to filter *insensitive updated edges* other than involving them in the update of $\boldsymbol{W}^{(t)}$. First, we model insensitive updated edges as follows.

Definition 3.1 (Insensitive Updated Edge). At timestamp t, given the clustering result $\bigcup_{l=1}^{q} C_l^{(t)}$, an updated edge $e = (i, j) \in \Delta E^{(t)}$ is *insensitive* if it satisfies the following conditions:

$$\begin{aligned} &i, j \in C_l^{(t)},\ e = (i, j) \in \Delta E_+^{(t)}, \\ &D_{ist}(i, V \setminus C_l^{(t)}) > 1,\ D_{ist}(j, V \setminus C_l^{(t)}) > 1 \end{aligned} \tag{13}$$

or

$$\begin{aligned} &i \in C_l^{(t)}, j \in V \setminus C_l^{(t)},\ e = (i, j) \in \Delta E_-^{(t)}, \\ &D_{ist}(i, V \setminus C_l^{(t)}) \leq 1 \end{aligned} \tag{14}$$

where $D_{ist}(v, C^{(t)})$ denotes the shortest distance (i.e., the number of hops) from node v to reach any node within the cluster $C^{(t)}$. The condition $D_{ist}(i, V \setminus C_l^{(t)}) > 1$ indicates that node i does not induce any k-clique on the clustering boundary.

Note that the insensitive edge is independent of its nodes group membership, and it is only related to graph structures. Intuitively, Eq. 13 can be understood as an *"inserted intra-cluster edge"*, and Eq. 14 can be understood as a *"deleted inter-cluster edge"*; whereas a sensitive edge is an *"inserted inter-cluster edge"* or a *"deleted intra-cluster edge"*. It is easy to prove that a single insensitive updated edge will not change the previous spectral clustering (i.e., the optimal partition $\bigcup_{l=1}^{q} C_l^{(t)}$ still achieves the optimal *CPNcut* ratio at $t+1$). Therefore, comparing with updating $\boldsymbol{W}^{(t)}$ for one insensitive edge $e = (i, j)$ costing $O(n^{k-2})$, our proposed edge filtering operation (i.e., Eq. 13 and Eq. 14) only costs $O(max(D^{(t)}(i, i), D^{(t)}(j, j)))$ for identifying that insensitive edge, filtering it out for the current time update, and saving it for the future timestamps. However, even if $\Delta E^{(t)}$ only contains insensitive updated edges, multiple insensitive edges may change the spectral clustering structure under extreme circumstances. In the proof of Proposition 1, we analyze the extreme scenario when many insensitive updated edges affect the previously identified optimal clustering.

Algorithm 1 Laplacian Update via Edge Filtering and Searching

Input:
updated edge set $\Delta E^{(t)}$, matrices $\boldsymbol{A}^{(t)}$, $\boldsymbol{W}^{(t)}$, and $\boldsymbol{D}^{(t)}$
Output:
matrices $\boldsymbol{A}^{(t+1)}$, $\boldsymbol{W}^{(t+1)}$, $\boldsymbol{D}^{(t+1)}$, and $\boldsymbol{L}^{(t+1)}$

```
if each e ∈ ΔE^(t) satisfies Eq. 13 or Eq. 14 then
    Save ΔE^(t) for next timestamp updates
else
    for e = (i, j) ∈ ΔE^(t) do /*Denote i with the larger degree*/
        Update adjacency matrix A^(t) for edge (i, j)
        Mark all the nodes adjacent to i based on A^(t)
        for each node r adjacent to j do
            if node r is marked then
                Input each node pair of i, j, r into Eq. 15
            end if
        end for
        Erase marks
    end for
end if
```

Proposition 1. *Assuming that the clustering $\bigcup_{l=1}^{q} C_l^{(t)}$ is obtained by minimizing CPNcut ratio under the fairness constraint at time t and $\Delta E^{(t)}$ only contains insensitive updated edges, the extreme cases exists (e.g., when the insensitive added edges are heavily localized) that the structure $\bigcup_{l=1}^{q} C_l^{(t)}$ is not guaranteed to have the minimal CPNcut ratio at time $t+1$. (Proof in Appendix.)*

Proposition 1 suggests that even if $\Delta E^{(t)}$ only consists of insensitive updated edges, the previous optimal clustering may not remain optimal when those updates are centralized on a local area of graph $G^{(t)}$. However, in practice, the graph itself and the corresponding updates can be usually sparse [2], and we can assume the extreme cases (e.g., updates are densely localized) in Proposition 1 are rare and obvious against the sparse background. Hence, if extreme cases happen, we may relatively easily observe them (e.g., by identifying the distance among updates or comparing the conductance of local updates with the previously-obtained conductance of the entire graph) and re-run the static algorithm from scratch. Thus, in the dynamic setting, we can filter $\Delta E^{(t)}$ when it only contains insensitive updated edges and save the filtered edges for future updates. But when $\Delta E^{(t)}$ contains a single sensitive updated edge, even if the majority of $\Delta E^{(t)}$ is insensitive, the whole $\Delta E^{(t)}$ need to be

involved into the updating process. Because any sensitive edge will change the previous clustering structure, which will make the currently considered insensitive edges invalid.

After edge filtering, we can then update $\boldsymbol{W}^{(t)}$ efficiently based on edge searching. First, we start from each inserted (or deleted) edge and then incrementally search edges that compose instances (or disappearance) of k-cliques. After that, we add the newly found cliques to $\boldsymbol{W}^{(t)}$. To be specific, the searching process is instanced by the enumeration of k-cliques containing the node i of the updated edge $e = (i, j)$ into the enumeration of $(k-1)$-cliques in the subgraph induced by the neighbors of node i, until $(k-1) = 2$. We illustrate the process for $k = 3$ in Alg. 1 and analyze how a k-clique is decomposed for the fast enumeration in Proposition 2. In Steps 6-8, enumerating an appeared (or disappeared) triangle due to the updated edge (i, j) is decomposed into the enumeration of any edge containing node j in the subgraph induced by the 1-hop neighbors of node i. After detecting an appearing (or disappearing) triangle, Step 9 feeds each involved node into Eq. 15 to update $\boldsymbol{W}^{(t)}$.

$$W^{(t)}(u, v) = W^{(t)}(u, v) + \gamma \tag{15}$$

where u and v denote any two nodes which cause the appearance (or disappearance) of the k-clique $\mathbb{N}$; $\gamma = 1$ (or -1) if the updated edge is an inserted (or deleted) edge. With the recursion rule of decomposing a k-clique into a $(k-1)$-clique [13], the searching time complexity is bounded by the arboricity of the current graph as shown in Proposition 2.

Proposition 2. *Given the graph $G^{(t)}$ and set $\Delta E^{(t)}$ containing sensitive updated edges, updating matrix $\boldsymbol{W}^{(t)}$ in terms of user-specified k-clique ($k \geq 2$) costs time complexity $O(k\alpha^{k-2}m^{(t)})$, where α is the arboricity of graph $G^{(t)}$, and $m^{(t)}$ is the number of edges in graph $G^{(t)}$. (Proof in Appendix.)*

3.3 Eigen-Pairs Update with Singularity Avoided

Next, to infer clustering, we need q smallest eigenvalues $\lambda_1, \ldots, \lambda_q$ and the corresponding eigenvectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_q$ (i.e., eigen-pairs) of the updated $\boldsymbol{M}^{(t+1)}$. We denote $\boldsymbol{\Lambda} \in \mathbb{R}^q$ as the column vector consisting of q smallest eigenvalues, i.e., $\Lambda(i) = \lambda_i$, and $\boldsymbol{U} \in \mathbb{R}^{(n-h+1)\times q}$ as the matrix consisting of th corresponding q eigenvectors, i.e., $U(:, j) = \boldsymbol{u}_j$. However, computing eigen-pairs $(\boldsymbol{\Lambda}^{(t+1)}, \boldsymbol{U}^{(t+1)})$ is costly, typically requiring $O(n^3)$ time [41], where n is the number of nodes at time $t+1$. Therefore, we aim to track them instead of solving them every single time. To be specific, based on matrix perturbation theory in [9], when $\boldsymbol{M}^{(t)}$ changes into $\boldsymbol{M}^{(t+1)}$, the change of eigen-pairs is expressed as follows.

$$(\boldsymbol{M}^{(t)} + \Delta\boldsymbol{M})(\boldsymbol{u}_i^{(t)} + \Delta\boldsymbol{u}_i) = (\lambda_i^{(t)} + \Delta\lambda_i)(\boldsymbol{u}_i^{(t)} + \Delta\boldsymbol{u}_i) \tag{16}$$

where $\lambda_i^{(t)}$ and $\boldsymbol{u}_i^{(t)}$ are the i-th eigenvalue and eigenvector of $\boldsymbol{M}^{(t)}$, $i \in \{1, \ldots, q\}$. Then we can expand Eq. 16 and obtain Eq. 17.

$$\boldsymbol{M}^{(t)}\Delta\boldsymbol{u}_i + \Delta\boldsymbol{M}\boldsymbol{u}_i^{(t)} + \Delta\boldsymbol{M}\Delta\boldsymbol{u}_i = \lambda_i^{(t)}\Delta\boldsymbol{u}_i + \Delta\lambda_i\boldsymbol{u}_i^{(t)} + \Delta\lambda_i\Delta\boldsymbol{u}_i^{(t)} \tag{17}$$

Eigenvalue Update. Based on Eq. 17, we multiply $\boldsymbol{u}_i^{(t)\top}$ on both sides. Suppose matrix $\boldsymbol{M}^{(t)}$ is near symmetric, and the eigenvector has unit length, then we get the following equation.

$$\boldsymbol{u}_i^{(t)\top}\Delta\boldsymbol{M}\boldsymbol{u}_i^{(t)} + \boldsymbol{u}_i^{(t)\top}\Delta\boldsymbol{M}\Delta\boldsymbol{u}_i = \Delta\lambda_i + \boldsymbol{u}_i^{(t)\top}\Delta\lambda_i\Delta\boldsymbol{u}_i^{(t)} \tag{18}$$

Algorithm 2 Eigen-Pairs Update with Singularity Avoided

Input:
eigen-pairs $(\boldsymbol{\Lambda}^{(t)}, \boldsymbol{U}^{(t)})$, perturbation matrix $\Delta\boldsymbol{M}$
Output:
eigen-pairs $(\boldsymbol{\Lambda}^{(t+1)}, \boldsymbol{U}^{(t+1)})$
1: Compute $\boldsymbol{X}^{(t)} = \boldsymbol{U}^{(t)\top}\Delta\boldsymbol{M}\boldsymbol{U}^{(t)}$
/*Eigenvalue Update*/
2: $\Delta\Lambda = diag(\boldsymbol{X}^{(t)})$
3: $\Lambda^{(t+1)} = \Lambda^{(t)} + \Delta\Lambda$
/*Eigenvector Update*/
4: Singularity = False
5: **for** $i = 1 : q$ **do**
6: Form $B^{(t)}(j, j) = \lambda_i^{(t)} + \Delta\lambda_i - \lambda_j^{(t)}$ for $j = 1, \ldots, q$
7: Compute $\boldsymbol{b}_i = (\boldsymbol{B}^{(t)} - \boldsymbol{X}^{(t)})^{-1}X(:, i)^{(t)}$
8: **if** $(\boldsymbol{B}^{(t)} - \boldsymbol{X}^{(t)})$ is singular **then**
9: Singularity == True; Break
10: **end if**
11: $\boldsymbol{u}_i^{(t+1)} = \boldsymbol{u}_i^{(t)} + \sum_{j=1}^{q}\boldsymbol{b}_i(j)\boldsymbol{u}_j^{(t)}$
12: **end for**
13: **if** Singularity == True **then**
14: **for** $i = 1 : q$ **do**
15: $\boldsymbol{u}_i^{(t+1)} = \boldsymbol{u}_i^{(t)}$
16: $\boldsymbol{u}_i^{(t+1)} += \frac{\boldsymbol{u}_j^{(t)\top}\Delta\boldsymbol{M}\boldsymbol{u}_i^{(t)}}{\lambda_i^{(t)} - \lambda_j^{(t)}}\boldsymbol{u}_j^{(t)}$ for $j = 1, \ldots, q$ and $j \neq i$
17: **end for**
18: **end if**

According to the assumption of [9] that $\Delta\lambda_i \ll \lambda_i$ and $\Delta\boldsymbol{u}_i \ll \boldsymbol{u}_i$, then $\boldsymbol{u}_i^{(t)\top}\Delta\boldsymbol{M}\Delta\boldsymbol{u}_i$ and $\boldsymbol{u}_i^{(t)\top}\Delta\lambda_i\Delta\boldsymbol{u}_i^{(t)}$ in Eq. 18 are safely omitted during the computation of $\Delta\lambda_i$. Therefore, we get eigenvalue update equation as follows.

$$\lambda_i^{(t+1)} = \lambda_i^{(t)} + \Delta\lambda_i, \quad \text{s.t.} \quad \Delta\lambda_i = \boldsymbol{u}_i^{(t)\top}\Delta\boldsymbol{M}\boldsymbol{u}_i^{(t)} \tag{19}$$

Eigenvector Update. Since we assume the dimensions of $\boldsymbol{M}^{(t)}$ and $\boldsymbol{M}^{(t+1)}$ are the same, the vector $\Delta\boldsymbol{u}_i$ can be expressed by the weighted sum of current eigenvectors as $\Delta\boldsymbol{u}_i = \sum_{j=1}^{q}\beta_{ji}\,\boldsymbol{u}_j^{(t)}$. Next, we need to estimate each weight β_{ji} to get $\Delta\boldsymbol{u}_i$, and finally obtain $\boldsymbol{u}_i^{(t+1)} = \boldsymbol{u}_i^{(t)} + \Delta\boldsymbol{u}_i$.

To solve for each β_{ji}, again starting from Eq. 17, we replace $\Delta\boldsymbol{u}_i$ with $\sum_{j=1}^{q}\beta_{ji}\,\boldsymbol{u}_j^{(t)}$ and multiply any eigenvector $\boldsymbol{u}_p^{(t)}$ ($1 \leq p \leq q, p \neq i$) on both sides. Then, based on the orthogonality of eigenvectors, we rearrange terms and get the following equation.

$$X(:, i)^{(t)} - \boldsymbol{B}^{(t)}\boldsymbol{b}_i + \boldsymbol{X}^{(t)}\boldsymbol{b}_i = \boldsymbol{0} \tag{20}$$

where $\boldsymbol{X}^{(t)} = \boldsymbol{U}^{(t)\top}\Delta\boldsymbol{M}\boldsymbol{U}^{(t)}$. $\boldsymbol{B}^{(t)} \in \mathbb{R}^{q\times q}$ is the diagonal matrix with $B^{(t)}(j, j) = \lambda_i^{(t)} + \Delta\lambda_i - \lambda_j^{(t)}$, and λ_i is from Eq. 19. $\boldsymbol{b}_i \in \mathbb{R}^q$ is the weight vector to compute Δu_i, i.e., $\boldsymbol{b}_i(j) = \beta_{ji}$. Rearranging Eq. 20, we have the weight computation equation below.

$$\boldsymbol{b}_i = (\boldsymbol{B}^{(t)} - \boldsymbol{X}^{(t)})^{-1}X(:, i)^{(t)} \tag{21}$$

With $\boldsymbol{b}_i$, we can then compute $\boldsymbol{u}_i^{(t+1)}$ as follows.

$$\boldsymbol{u}_i^{(t+1)} = \boldsymbol{u}_i^{(t)} + \Delta\boldsymbol{u}_i, \quad \text{s.t.} \quad \Delta\boldsymbol{u}_i = \sum_{j=1}^{q}\boldsymbol{b}_i(j)\boldsymbol{u}_j^{(t)} \tag{22}$$

However, in Eq. 21, when a certain eigenvalue does not change, then $\boldsymbol{B}^{(t)}$ is a singular matrix. Since the graph and its update are usually sparse, $\boldsymbol{B}^{(t)} - \boldsymbol{X}^{(t)}$ is also highly likely to be singular, then an unchanged eigenvalue will stop other eigenvectors' update. To address this issue, we aim to update eigenvector $\boldsymbol{u}_i$ independent of its $\Delta\lambda_i$. Again, starting from Eq. 17 and omitting trivial terms $\Delta\boldsymbol{M}\Delta\boldsymbol{u}_i$ and $\Delta\lambda_i\Delta\boldsymbol{u}_i^{(t)}$, we obtain the approximation solution $\boldsymbol{M}^{(t)}\Delta\boldsymbol{u}_i + \Delta\boldsymbol{M}\boldsymbol{u}_i^{(t)} = \lambda_i^{(t)}\Delta\boldsymbol{u}_i + \Delta\lambda_i\boldsymbol{u}_i^{(t)}$, where we replace $\Delta\boldsymbol{u}_i$ with $\sum_{j=1}^{q}\beta_{ji}\,\boldsymbol{u}_j^{(t)}$ and get the weight β_{ji} as follows.

$$\beta_{ji} = \frac{\boldsymbol{u}_j^{(t)\top}\Delta\boldsymbol{M}\boldsymbol{u}_i^{(t)}}{\lambda_i^{(t)} - \lambda_j^{(t)}} \tag{23}$$

Then, the alternative eigenvector update operation is as follows.

$$\boldsymbol{u}_i^{(t+1)} = \boldsymbol{u}_i^{(t)} + \Delta\boldsymbol{u}_i, \text{ s.t. } \Delta\boldsymbol{u}_i = \sum_{j=1}^{q}\frac{\boldsymbol{u}_j^{(t)\top}\Delta\boldsymbol{M}\boldsymbol{u}_i^{(t)}}{\lambda_i^{(t)} - \lambda_j^{(t)}}\boldsymbol{u}_j^{(t)} \tag{24}$$

Note that, comparing with Eq. 22, Eq. 24 is less accurate by omitting two mentioned trivial terms, but the matrix singularity issue is avoided. Now, we are ready to present Alg 2. to track eigenvalues and eigenvectors of $\boldsymbol{M}^{(t+1)}$. In Alg. 2, Steps 2–3 update the eigenvalue based on Eq. 19, Steps 5–12 update the eigenvector according to Eq. 22 if singularity issue does not happen, otherwise Steps 13–18 update the eigenvector according to Eq. 24. The total time complexity of Alg. 2 is bounded, as shown in Proposition 3.

Proposition 3. *Given the perturbation matrix $\Delta\boldsymbol{M}$ from time t to $t+1$, using Alg. 2 to track $(\boldsymbol{\Lambda}^{(t+1)}, \boldsymbol{U}^{(t+1)})$ costs time complexity $O(q^4 + nq^2)$, where q stands for the number of tracked eigenvalues, and n is the number of nodes. (Proof in Appendix.)*

4 F-SEGA ALGORITHM

In this section, we combine all proposed techniques and introduce the end-to-end clustering framework, F-SEGA, in Alg. 3. Given the user-specified k-clique $\mathbb{N}$, the desired number of clusters q, the group-membership matrix $\boldsymbol{F}$, the clique-weighted adjacency matrix $\boldsymbol{W}^{(0)}$ at $t = 0$, and the updated edge set $\{\Delta E^{(0)}, \ldots, \Delta E^{(T-1)}\}$, F-SEGA algorithm outputs the clique-preserving and fairness-aware clustering associated with each timestamp t. In Alg. 3, the initial clustering at $t = 0$ can be obtained through the proposed static algorithm (Section 3.1). Then, Step 2 updates Laplacian matrix $\boldsymbol{L}^{(t)}$ into $\boldsymbol{L}^{(t+1)}$ to obtain $\boldsymbol{M}^{(t+1)}$ through Eq. 12. After that, Step 3 tracks eigen-pairs of the obtained $\boldsymbol{M}^{(t+1)}$ by leveraging the perturbation between $\boldsymbol{M}^{(t+1)}$ and $\boldsymbol{M}^{(t)}$ in Alg. 2. Finally, Step 4 returns the clique-preserving and fair clustering result.

Theorem 4.1 (Time Complexity of F-SEGA). *The time complexity of the proposed F-SEGA algorithm is bounded by $O(k\alpha^{k-2}m^{(t)} + q^4 + q^2n)$ at each timestamp t, where k is the number of nodes in user-defined clique $\mathbb{N}$, α is the arboricity of graph $G^{(t)}$, and $m^{(t)}$ is the number of edges in graph $G^{(t)}$. (Proof in Appendix.)*

5 EXPERIMENTS

We evaluate the effectiveness, efficiency, and parameter sensitivity of F-SEGA [3]through comparison with baselines and ablation studies. We also provide a case study of using F-SEGA to design a proportional allocation of human resources.

[3]https://github.com/DongqiFu/F-SEGA

Algorithm 3 Fairness-Aware Clique-Preserving Spectral Clustering of Temporal Graphs (F-SEGA)

Input:
k-clique $\mathbb{N}$, number of clusters q, matrices $\boldsymbol{A}^{(0)}, \boldsymbol{W}^{(0)}, \boldsymbol{F}$, and updated edge set $\{\Delta E^{(0)}, \ldots, \Delta E^{(T-1)}\}$.
Output:
clusters $\{C_1^{(t)}, \ldots C_q^{(t)}\}$, where $t \in \{1, 2, \ldots, T\}$
1: **for** $t = 0 : T-1$ **do**
2: Update graph Laplacian matrix $\boldsymbol{L}^{(t)}$ into $\boldsymbol{L}^{(t+1)}$ through Alg. 1 to obtain $\boldsymbol{M}^{(t+1)}$ by Eq. 12.
3: Track $(\boldsymbol{\Lambda}^{(t+1)}, \boldsymbol{U}^{(t+1)})$ through Alg. 2 by leveraging $\boldsymbol{M}^{(t)}$ and $\boldsymbol{M}^{(t+1)}$.
4: Set $\boldsymbol{U}^{(t+1)}$ as $\boldsymbol{X}^{(t+1)}$ in Eq. 11 to form $\boldsymbol{H}^{(t+1)}$ for obtaining clustering $\{C_1^{(t+1)}, \ldots C_q^{(t+1)}\}$
5: **end for**

5.1 Experiment Setup

Real-World Temporal Graphs. *HighSchool-2011* [16] stores dynamic human contacts, where 126 nodes denote students, and the 28,561 timestamped edges denote the face-to-face contacts between students during 4 days. The groups of students include males and females. *HighSchool-2013* [35] is another dynamic human contact graph of 327 high school students during 5 days. The nodes denote students grouped by males and females, and the 188,509 timestamped edges denote contacts. *PrimarySchool* [22] is a dynamic interaction graph of 232 students and 10 teachers grouped by males and females, and 125,773 timestamped edges denote the contacts among them. *ASA* [40] stands for a dynamic inter-personal graph, where 5,767 nodes represent male and female employees from 384 limited companies, and 873,716 timestamped edges represent the affiliation among employees in 10 years. *Hospital* [46] is a dynamic human contact graph among patients and healthcare workers in a hospital ward for 5 days. 75 nodes in the Hospital graph are divided into four groups: patients, nurses, medical doctors, and administrative staff. 32,424 timestamped edges denote the contacts.

Data Pre-processing. Due to different time granularities (e.g., seconds and months) and different durations (e.g., 4 days and 10 years) among real-world temporal graphs, we unify all timestamps into 11 snapshots, where the snapshot $G^{(0)}$ occupies $80\% - 90\%$ of the entire observed graph in terms of the volume (i.e., number of edges), and each updated edge set $\Delta E^{(t)}, t \in \{0, \ldots, 9\}$ contains updated edges with the number of $1\% - 2\%$ volume of the whole observed graph.

5.2 Baseline Algorithms

We include spectral clustering algorithms from 3 aspects, i.e., **fair and unfair**, **low-order and high-order**, and **static and dynamic**. SC [41] is the standard (i.e., edge-based) spectral clustering aiming for minimizing $Ncut$. TripSC [9] tracks eigen-pairs of Laplacian matrix to meet the dynamic spectral clustering baseline. MSC [5] stands for the motif spectral clustering, which is a high-order algorithm proposed to minimize the number of broken motifs due to the graph cuts. Different from cliques, motifs do not have to be fully connected. FSC [29] represents the fair spectral clustering, which

Table 2: Comparison of Effectiveness and Efficiency

Data	HighSchool-2011 (Small Number of Clusters)				HighSchool-2011 (Large Number of Clusters)			
Method \ Metric	*Ncut*	*CPNcut*	Avg. Balance	Time (cs)	*Ncut*	*CPNcut*	Avg. Balance	Time (cs)
SC	3.1389 ± 0.8599	3.0331 ± 0.9046	0.4596 ± 0.0454	9.5270 ± 2.4491	11.7701 ± 1.0063	11.7578 ± 0.9956	0.2679 ± 0.0245	29.7407 ± 4.3385
TripSC	3.9756 ± 1.0791	3.9507 ± 1.1274	0.4519 ± 0.0669	4.7160 ± 0.2390	12.6620 ± 1.2201	12.4800 ± 1.0953	0.3458 ± 0.0268	5.8290 ± 0.0238
MSC	3.1443 ± 0.8973	2.9554 ± 0.8900	0.3888 ± 0.0850	17.0819 ± 2.1950	12.1444 ± 0.9648	12.1707 ± 0.9624	0.2686 ± 0.0503	47.5360 ± 4.0390
FSC	3.4110 ± 0.7931	3.3047 ± 0.8312	0.4457 ± 0.0185	23.8289 ± 2.3470	11.8866 ± 0.9955	11.9571 ± 1.0047	0.2473 ± 0.0228	55.2460 ± 4.5900
F-SEGA	4.4525 ± 0.9885	4.4435 ± 0.9947	0.6281 ± 0.0851	15.4022 ± 0.9090	13.4689 ± 1.5127	13.4726 ± 1.4659	0.4023 ± 0.0413	15.1860 ± 1.1020
Data	HighSchool-2013 (Small Number of Clusters)				HighSchool-2013 (Large Number of Clusters)			
Method \ Metric	*Ncut*	*CPNcut*	Avg. Balance	Time (cs)	*Ncut*	*CPNcut*	Avg. Balance	Time (cs)
SC	1.4866 ± 0.4334	0.6458 ± 0.2347	0.4708 ± 0.0135	33.2589 ± 2.3160	8.3264 ± 0.9598	6.9722 ± 0.9852	0.3596 ± 0.0276	55.7780 ± 5.3360
TripSC	1.7915 ± 0.2823	1.0755 ± 0.5641	0.4531 ± 0.0182	27.5309 ± 0.4920	12.4063 ± 0.7526	12.2279 ± 0.8568	0.4568 ± 0.1048	29.5580 ± 7.2800
MSC	1.4664 ± 0.4205	0.6483 ± 0.1829	0.4695 ± 0.0139	63.0641 ± 2.2970	8.4577 ± 1.0816	7.0812 ± 1.1676	0.3971 ± 0.0432	84.5730 ± 4.9880
FSC	1.5203 ± 0.4895	0.6620 ± 0.2860	0.5160 ± 0.0466	52.8459 ± 2.5430	8.3129 ± 0.9832	6.9370 ± 0.9918	0.3430 ± 0.0264	76.1430 ± 5.9880
F-SEGA	1.5296 ± 0.3493	0.6728 ± 0.1800	0.4620 ± 0.0058	23.1415 ± 0.1730	11.1068 ± 1.1258	10.4639 ± 1.1930	0.4481 ± 0.0827	23.2780 ± 0.5130
Data	PrimarySchool (Small Number of Clusters)				PrimarySchool (Large Number of Clusters)			
Method \ Metric	*Ncut*	*CPNcut*	Avg. Balance	Time (cs)	*Ncut*	*CPNcut*	Avg. Balance	Time (cs)
SC	3.2948 ± 0.7586	3.1690 ± 0.7504	0.7346 ± 0.0621	26.5813 ± 2.9490	7.6632 ± 0.9263	7.4345 ± 0.8838	0.6385 ± 0.0145	33.1950 ± 4.3430
TripSC	3.3368 ± 0.7193	3.1903 ± 0.7137	0.7090 ± 0.0319	17.1575 ± 0.2420	9.0180 ± 0.9117	8.8868 ± 0.9408	0.6235 ± 0.0432	17.2640 ± 0.4350
MSC	3.3528 ± 0.7852	3.1730 ± 0.7744	0.7098 ± 0.0232	74.5010 ± 2.7000	7.4506 ± 0.5170	7.4875 ± 0.8808	0.6775 ± 0.0331	90.0930 ± 4.0000
FSC	3.2830 ± 0.7488	3.1894 ± 0.7390	0.7430 ± 0.0566	49.4491 ± 3.2140	7.6606 ± 0.9217	7.4255 ± 0.8698	0.6747 ± 0.0331	67.6080 ± 4.4381
F-SEGA	3.4018 ± 0.7412	3.2126 ± 0.7205	0.6717 ± 0.0143	21.8349 ± 0.6260	8.7635 ± 1.8539	8.4676 ± 1.7872	0.6338 ± 0.0500	22.1440 ± 0.2928

adds the fairness constraints on the standard spectral clustering to make the demographics of each cluster close to the whole graph. For static baselines, we report the clustering result that is solely obtained on the last snapshot. *For dynamic baselines, we report the tracked clustering result until the last snapshot, whose tracking is started from the first snapshot.* We provide ablations by removing different components of F-SEGA individually in Appendix E.

5.3 Effectiveness Comparison

Evaluation Metrics. (1) *Ncut*, which measures the compactness of clustering through broken edges, low *Ncut* score indicates that clusters are densely connected by edges; (2) *CPNcut*, which measures the compactness of clusters in terms of broken k-cliques, in experiments we set $k = 3$, low *CPNcut* score indicates that clusters are densely connected by k-cliques; (3) Time, which records the consumed time of each algorithm; (4) Average Balance [29], which measures the fairness of clustering by calculating the demographics of each cluster, higher balance $bal(\cdot)$ indicates a cluster is fairer. *Ncut* and *CPNcut* are realized by Eq. 1 by changing the order of $\mathbb{N}$, and Average Balance is realized as follows.

$$\frac{1}{|C_l|}\sum_{l} bal(C_l) = \frac{1}{|C_l|}\sum_{l} \min_{s\neq s'\in\{1,\ldots,h\}} \frac{|V_s \cap C_l|}{|V_{s'} \cap C_l|} \in [0,1]$$

Quantitative Analysis. The six settings (i.e., three datasets with a small number of clusters $q = \{5, 6, 7\}$ and a large number of clusters $q = \{14, 15, 16\}$) in Table 2 reflects two scenarios in the real world. **First, the initial distribution of the input graph is not demographic fair**. As shown in HighSchool-2011(Large), when all other baselines find roughly similar sub-optimal compact and fair clustering, our F-SEGA identifies a fairer (i.e., higher average balance) clustering while sacrificing very little compactness (e.g., competitive low *CPNcut* ratio). In Highschool-2011(Small) and HighSchool-2013(Large), the input data is not fairly distributed either. Our F-SEGA still achieves very competitive fairness scores with only a little decreases in the compactness metric. **Second, the initial distribution of the input graph is already demographic fair**. For example, no matter with Highschool-2013(Small), PrimarySchool(Small), or PrimarySchool(Large), the initial distribution is fair as all baselines fall into the same performance level. Despite some random tracking errors, our F-SEGA still achieves competitive performance in a very efficient manner.

F-SEGA achieves good overall performance among high-order density, demographic fairness, and time complexity. Taking HighSchool-2011 dataset as an example shown in Figure 2, F-SEGA is the closest one to the comprehensiveness, i.e., the line ($x = y = z$).

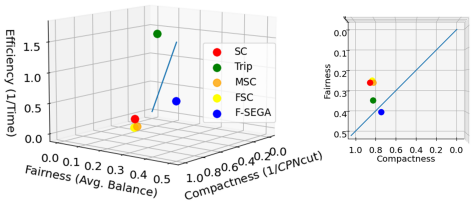


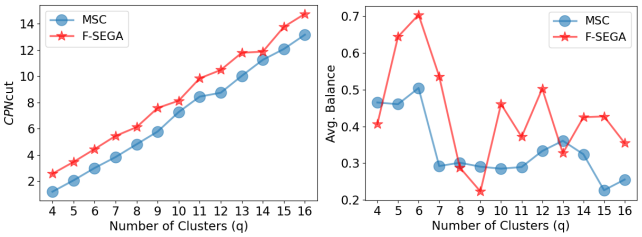
Figure 2: Comprehensiveness in HighSchool-2011 (Large).

5.4 Parameter Sensitivity

Different choices of the number of clusters (i.e., q) affect the structure of clustering. Here, we aim to investigate the performance consistency of F-SEGA in different clustering settings. To this end, we evaluate the compactness performance (i.e., *CPNcut*) and the fairness performance (i.e., Average Balance) for different q values. We report the performance on HighSchool-2011. Comparing with MSC [5] (designed for the high-order but unfair spectral clustering) at the final snapshot, we have the following observations. (1) In Figure 3a, *CPNcut* increases as q increases, because more cliques are cut as the number of clusters increases. (2) In Figure 3b, as q increases, the fairness performance (i.e., Average Balance) does not have a clear increasing or decreasing pattern because it is based on the distribution of the given data; (3) We can see adding fairness constraints produces much fairer clustering but costs compactness.

5.5 Case Study

In the case study, we use the proposed F-SEGA to design a proportional strategy for allocating human resources in a hospital ward.



(a) $CPNcut$ **w.r.t.** q **(b) Balance w.r.t.** q

Figure 3: Performance on Different Clustering Sizes.

In Hospital graph data, there are four groups of people: patients, nurses, doctors, and administrative staff. Given their existing connections, it is critical to design an efficient communication and resource allocation strategy [50]. Ideally, we want healthcare workers and patients to be densely connected in each cluster such that patients can obtain timely communication and care. Also, each cluster should contain similar proportions of people from all four groups to ensure a balanced workload for healthcare workers.

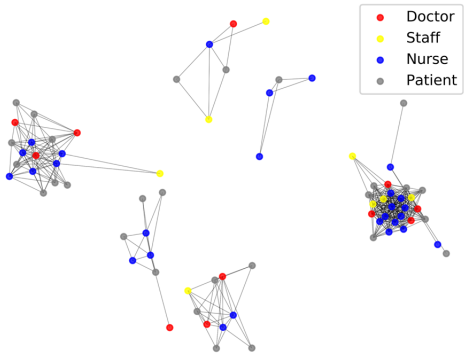


Figure 4: Proportional human resource allocation in Hospital graph. Grey nodes denote patients, blue nodes denote nurses, red nodes denote medical doctors, and yellows nodes denote administrative staff.

To this end, we use F-SEGA algorithm to track the clustering of Hospital graph and show the clustering result on the last snapshot in Figure 4. Here, we set the triangle as the target clique being preserved from graph cuts for representing efficient communication among people and $q = 6$. We observe that F-SEGA produces a set of clusters that are not only triangle-dense but also contain similar proportions of people from all four groups. To be specific, we discover that healthcare workers are assigned densely and proportionally in each cluster, which suggests that F-SEGA could help dynamically design the human resource allocation in a proportional and efficient manner given the original connections.

6 RELATED WORK

Fair Clustering. Fairness constraints in the clustering problems receive a surge of research interests. Different fair clustering algorithms [6, 11, 14, 24, 26, 29, 32, 49] are designed for different fair objectives, like protecting minority groups or hiding sensitive attributes. Fair clustering algorithms reduce bias in many applications, like computer vision [32]. In [14], researchers aim to protect groups such that the demographic distribution in every cluster should be approximately equal to the entire dataset. Different from [14] who always produces fair clustering results no matter how much cost increases compared with unfair clustering methods, researchers in [29] add the fairness constraints on the spectral clustering algorithm to guide a fair clustering result if such result exists but only pay a little price for the compactness loss of the returned clusters.

High-Order Clustering. For the graph clustering problems, the standard spectral clustering algorithms [36, 41] study the graph structure by investigating the eigenvalues and eigenvectors of the graph Laplacian matrix. To preserve high-order connection patterns from cuts, tensor-based spectral clustering methods [4, 51] are proposed. In [5, 45], a re-weighting method is introduced to model the high-order patterns, this method represents the input graph into a weighted two-dimensional matrix where each entry stands for the number of high-order patterns that edge occupies, then the eigen-decomposition solution of that two-dimensional matrix can be used for indicating clustering results to preserve high-order patterns. This re-weighting method is also adopted to solve the high-order local clustering problem [53]. In the high-order local clustering algorithms [21, 53–55], the local cluster is produced by only exploring a small portion of the entire graph given the seed node. Recently, authors in [7] consider the heterogeneity among nodes and edges and propose the high-order spectral clustering method in heterogeneous graphs.

Dynamic Clustering. For dynamic graphs, early dynamic clustering methods [3, 8] obtain the static clustering results independently at each timestamp, then matching or mapping them to investigate the evolutionary pattern of dynamic graphs. In the dynamic setting, some methods are proposed for the temporal smoothness like [12] which prefers the clustering result not only fitting the current data well but also stable and less sensitive to short-term updates. And some other dynamic clustering methods are proposed for fast clustering solutions in the dynamic setting, like [9, 10, 34, 37, 38, 52]. For example, in [38], authors propose the incremental spectral clustering method, which incrementally updates the eigenvalues and eigenvectors of the Laplacian matrix for indicating clustering results at each timestamp. Most dynamic graph clustering algorithms focus on edges (i.e., low-order structures) connectivity information.

To the best of our knowledge, F-SEGA is the first attempt to access high-order clustering under the group fairness constraints in the dynamic setting, which reconciles the two clustering constraints with the clustering efficiency.

7 CONCLUSION

In this paper, we propose F-SEGA for fairness-aware and clique-preserving spectral clustering of temporal graphs. In addition to the theoretical derivation of F-SEGA and the analysis of its time complexity, we empirically evaluate the effectiveness, efficiency, and robustness of F-SEGA. We also provide a case study to show the real-world application of F-SEGA.

ACKNOWLEDGMENTS

This work is supported by National Science Foundation under Award No. IIS-1947203, IIS-2117902, and IIS-2137468. The views and conclusions are those of the authors and should not be interpreted as representing the official policies of the funding agencies or the government.

REFERENCES

[1] Balázs Adamcsek, Gergely Palla, Illés J. Farkas, Imre Derényi, and Tamás Vicsek. 2006. CFinder: locating cliques and overlapping modules in biological networks. *Bioinformatics* (2006).

[2] Charu C. Aggarwal and Karthik Subbian. 2014. Evolutionary Network Analysis: A Survey. *ACM Comput. Surv.* (2014).

[3] Sitaram Asur, Srinivasan Parthasarathy, and Duygu Ucar. 2007. An event-based framework for characterizing the evolutionary behavior of interaction graphs. In *KDD*.

[4] Austin R. Benson, David F. Gleich, and Jure Leskovec. 2015. Tensor Spectral Clustering for Partitioning Higher-order Network Structures. In *SDM*.

[5] Austin R. Benson, David F. Gleich, and Jure Leskovec. 2016. Higher-order organization of complex networks. *Science* (2016).

[6] Suman Kalyan Bera, Deeparnab Chakrabarty, Nicolas Flores, and Maryam Negahbani. 2019. Fair Algorithms for Clustering. In *NeurIPS*.

[7] Aldo G. Carranza, Ryan A. Rossi, Anup Rao, and Eunyee Koh. 2020. Higher-order Clustering in Complex Heterogeneous Networks. In *KDD*.

[8] Deepayan Chakrabarti, Ravi Kumar, and Andrew Tomkins. 2006. Evolutionary Clustering. In *KDD*.

[9] Chen Chen and Hanghang Tong. 2015. Fast Eigen-Functions Tracking on Dynamic Graphs. In *SDM*.

[10] Chen Chen and Hanghang Tong. 2017. On the eigen-functions of dynamic graphs: Fast tracking and attribution algorithms. *Stat. Anal. Data Min.* (2017).

[11] Xingyu Chen, Brandon Fain, Liang Lyu, and Kamesh Munagala. 2019. Proportionally Fair Clustering. In *ICML*.

[12] Yun Chi, Xiaodan Song, Dengyong Zhou, Koji Hino, and Belle L. Tseng. 2007. Evolutionary spectral clustering by incorporating temporal smoothness. In *KDD*.

[13] Norishige Chiba and Takao Nishizeki. 1985. Arboricity and Subgraph Listing Algorithms. *SIAM J. Comput.* (1985).

[14] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. 2017. Fair Clustering Through Fairlets. In *NeurIPS*.

[15] Imre Derényi, Gergely Palla, and Tamás Vicsek. 2005. Clique Percolation in Random Networks. *Physical Review Letters* (2005).

[16] Julie Fournet and Alain Barrat. 2014. Contact Patterns among High School Students. *PLoS ONE* (2014).

[17] Dongqi Fu, Yikun Ban, Hanghang Tong, Ross Maciejewski, and Jingrui He. 2022. DISCO: Comprehensive and Explainable Disinformation Detection. In *CIKM 2022*.

[18] Dongqi Fu, Liri Fang, Ross Maciejewski, Vetle I. Torvik, and Jingrui He. 2022. Meta-Learned Metrics over Multi-Evolution Temporal Graphs. In *KDD 2022*.

[19] Dongqi Fu and Jingrui He. 2021. SDG: A Simplified and Dynamic Graph Neural Network. In *SIGIR*.

[20] Dongqi Fu and Jingrui He. 2022. Natural and Artificial Dynamics in Graphs: Concept, Progress, and Future. *Frontiers in Big Data* (2022).

[21] Dongqi Fu, Dawei Zhou, and Jingrui He. 2020. Local Motif Clustering on Time-Evolving Graphs. In *KDD*.

[22] Valerio Gemmetto, Alain Barrat, and Ciro Cattuto. 2014. Mitigation of infectious disease at school: targeted class closure vs school closure. *BMC infectious diseases* (2014).

[23] Mark S Granovetter. 1977. The strength of weak ties. In *Social networks*. Elsevier.

[24] Lingxiao Huang, Shaofeng H.-C. Jiang, and Nisheeth K. Vishnoi. 2019. Coresets for Clustering with Fairness Constraints. In *NeurIPS*.

[25] Ling Huang, Donghui Yan, Michael I. Jordan, and Nina Taft. 2008. Spectral Clustering with Perturbed Data. In *NeurIPS*.

[26] Jian Kang, Jingrui He, Ross Maciejewski, and Hanghang Tong. 2020. InFoRM: Individual Fairness on Graph Mining. In *KDD*.

[27] Jian Kang, Tiankai Xie, Xintao Wu, Ross Maciejewski, and Hanghang Tong. 2022. InfoFair: Information-Theoretic Intersectional Fairness. In *IEEE BigData*.

[28] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. 2020. Representation Learning for Dynamic Graphs: A Survey. *J. Mach. Learn. Res.* (2020).

[29] Matthäus Kleindessner, Samira Samadi, Pranjal Awasthi, and Jamie Morgenstern. 2019. Guarantees for Spectral Clustering with Fairness Constraints. In *ICML*.

[30] Gueorgi Kossinets and Duncan J Watts. 2006. Empirical analysis of an evolving social network. *Science* (2006).

[31] Liangyue Li and Hanghang Tong. 2020. Computational Approaches to the Network Science of Teams. (2020).

[32] Peizhao Li, Han Zhao, and Hongfu Liu. 2020. Deep Fair Clustering for Visual Learning. In *CVPR*.

[33] Helmut Lütkepohl. 1996. *Handbook of matrices*. Vol. 1. Wiley Chichester.

[34] Lionel Martin, Andreas Loukas, and Pierre Vandergheynst. 2018. Fast Approximate Spectral Clustering for Dynamic Networks. In *ICML*.

[35] Rossana Mastrandrea, Julie Fournet, and Alain Barrat. 2015. Contact Patterns in a High School: A Comparison between Data Collected Using Wearable Sensors, Contact Diaries and Friendship Surveys. *PLOS ONE* (2015).

[36] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. 2001. On Spectral Clustering: Analysis and an algorithm. In *NeurIPS*.

[37] Huazhong Ning, Wei Xu, Yun Chi, Yihong Gong, and Thomas S. Huang. 2007. Incremental Spectral Clustering With Application to Monitoring of Evolving Blog Communities. In *SDM*.

[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* (2010).

[39] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. 2005. Uncovering the overlapping community structure of complex networks in nature and society. *nature* (2005).

[40] Cathrine Seierstad and Tore Opsahl. 2011. For the few not the many? The effects of affirmative action on presence, prominence, and social capital of women directors in Norway. *Scandinavian Journal of Management* (2011).

[41] Jianbo Shi and Jitendra Malik. 2000. Normalized Cuts and Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* (2000).

[42] Hanghang Tong, Spiros Papadimitriou, Philip S. Yu, and Christos Faloutsos. 2008. Proximity Tracking on Time-Evolving Bipartite Graphs. In *SDM*.

[43] Amanda L. Traud, Peter J. Mucha, and Mason A. Porter. 2011. Social Structure of Facebook Networks. *CoRR* (2011).

[44] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2019. DyRep: Learning Representations over Dynamic Graphs. In *ICLR*.

[45] Charalampos E. Tsourakakis, Jakub Pachocki, and Michael Mitzenmacher. 2017. Scalable Motif-aware Graph Clustering. In *WWW*.

[46] Philippe Vanhems, Alain Barrat, Ciro Cattuto, Jean-François Pinton, Nagham Khanafer, Corinne Régis, Byeul-a Kim, Brigitte Comte, and Nicolas Voirin. 2013. Estimating Potential Infection Transmission Routes in Hospital Wards Using Wearable Proximity Sensors. *PLoS ONE* (2013).

[47] Ulrike von Luxburg. 2007. A tutorial on spectral clustering. *Stat. Comput.* (2007).

[48] Jianxin Wang, Xiaoqing Peng, Min Li, and Yi Pan. 2013. Construction and application of dynamic protein interaction network based on time course gene expression data. *Proteomics* (2013).

[49] Yian Wang, Jian Kang, Yinglong Xia, Jiebo Luo, and Hanghang Tong. 2022. iFiG: Individually Fair Multi-view Graph Clustering. In *IEEE BigData*.

[50] Nimnath Withanachchi, Yasuo Uchida, Shyama Nanayakkara, Dulani Samaranayake, and Akiko Okitsu. 2007. Resource allocation in public hospitals: Is it effective? *Health Policy* (2007).

[51] Tao Wu, Austin R. Benson, and David F. Gleich. 2016. General Tensor Spectral Co-clustering for Higher-Order Data. In *NeurIPS*.

[52] Donghui Yan, Ling Huang, and Michael I. Jordan. 2009. Fast approximate spectral clustering. In *KDD*.

[53] Hao Yin, Austin R. Benson, Jure Leskovec, and David F. Gleich. 2017. Local Higher-Order Graph Clustering. In *KDD*.

[54] Dawei Zhou, Jingrui He, Hasan Davulcu, and Ross Maciejewski. 2018. Motif-Preserving Dynamic Local Graph Cut. In *IEEE BigData*.

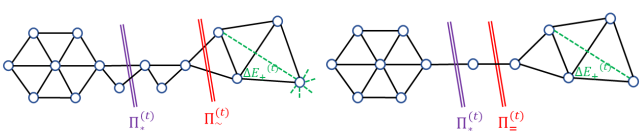
[55] Dawei Zhou, Si Zhang, Mehmet Yigit Yildirim, Scott Alcorn, Hanghang Tong, Hasan Davulcu, and Jingrui He. 2017. A Local Algorithm for Structure-Preserving Graph Cut. In *KDD*.

A PROOF OF PROPOSITION 1

To prove Proposition 1, we need to show the **optimal graph cuts** $\Pi_*^{(t)}$ at time t may not achieve the optimized *CPNcut* ratio at $t+1$, which implies the **tie graph cuts** $\Pi_=^{(t)}$ and **inferior graph cuts** $\Pi_\sim^{(t)}$ (w.r.t *CPNcut* ratio) may be chosen as the optimum for time $t+1$. To give a relatively full analysis, we first discuss that in a general case that $\Pi_*^{(t)}$ is still the optimum at $t+1$, and give some extreme cases that it will not be the optimum.

For an **insensitive inserted edge**, according to Eq. 5, it will only increase or remain the volume of each cluster of $\Pi_*^{(t)}$ and remain the number of broken cliques. Therefore, the *CPNcut* ratio of $\Pi_*^{(t)}$ remains or in-depth minimizes under the fairness constraint.

- However, extreme cases can happen that $\Pi_\sim^{(t)}$ can be the optimum at $t+1$ for only insensitive inserted edges. For example, as shown in Figure 5a (for 3-clique), ΔE only contains insensitive inserted edges w.r.t $\Pi_*^{(t)}$, but when the huge number of insensitive updated edges arrives intensively and locally on the right complement of $\Pi_\sim^{(t)}$, the volume of the right cluster will increase dramatically and $\Pi_\sim^{(t)}$ can be $\Pi_*^{(t+1)}$. In the real world, the graph is usually sparse, and each time update is also much smaller compared to the volume of cluster [21], such that we based on this assumption and infer $\Pi_\sim^{(t)}$ will not be $\Pi_*^{(t+1)}$.
- Extreme cases can also happen that $\Pi_=^{(t)}$ is qualified for $\Pi_*^{(t+1)}$ when the inserted insensitive edges w.r.t $\Pi_*^{(t)}$ are also insensitive to $\Pi_=^{(t)}$ as shown in Figure 5b (for 3-clique). In this scenario, temporal smoothness [12] will choose $\Pi_*^{(t)}$ as $\Pi_*^{(t+1)}$, i.e., the previously selected graph cuts have a preference when tied.



(a) Extreme Case I. **(b) Extreme Case II.**

Figure 5: Inserted Insensitive Edges Analysis.

For a **deleted insensitive edge**, if that inter-cluster deleted edge occupies s k-cliques $\mathbb{N}$, then s k-cliques will disappear and the volume of graph $G^{(t)}$, $\mu(G^{(t)}, \mathbb{N})$, will decrease $s \cdot \frac{k(k-1)}{2}$, because the weights of edges in each of s k-cliques are decreased. According to Eq. 5, for a cluster $C_l^{(t)}$, $cut(C_l, V \setminus C_l, \mathbb{N})$ will decrease at most $\eta \cdot s \cdot \frac{k(k-1)}{2}$ and $\mu(C_l, \mathbb{N})$ will decrease at most $(1-\eta) \cdot s \cdot \frac{k(k-1)}{2}$, $\eta = \frac{\kappa}{k} \in (0,1)$, and $\kappa \in \{1, \ldots, k-1\}$. If $\eta \geq 0.5$, since $\mu(C_l, \mathbb{N}) > cut(C_l, V \setminus C_l, \mathbb{N})$, the *CPNcut* ratio of $\Pi_*^{(t)}$ will decrease at $t+1$. Even $\eta \to 0$, we consider $\mu(C_l, \mathbb{N}) \gg cut(C_l, V \setminus C_l, \mathbb{N})$ then the *CPNcut* ratio of $\Pi_*^{(t)}$ will remain at $t+1$. Thus, the *CPNcut* ratio of $\Pi_*^{(t)}$ remains or in-depth minimizes under the fairness constraint.

- An extreme case as shown in in the Figure 6, $\Pi_=^{(t)}$ achieves the same *CPNcut* ratio (k=3) as $\Pi_*^{(t)}$ in terms of insensitive deleted edges. An insensitive deleted edge to $\Pi_*^{(t)}$ is also insensitive to $\Pi_=^{(t)}$, which means $\Pi_=^{(t)}$ has the change to be $\Pi_*^{(t+1)}$. If that happens, temporal smoothness [12] will still choose $\Pi_*^{(t)}$ as $\Pi_*^{(t+1)}$, i.e., the previous selected graph cuts have preference when tied.

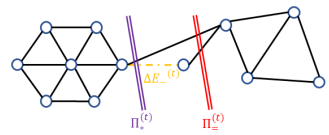


Figure 6: Extreme Case III.

B PROOF OF PROPOSITION 2

We first prove that the proposed Fast Update of Graph Laplacian method updates Laplacian matrix $\boldsymbol{L}^{(t)}$ in terms of the 3-clique requires $O(\alpha m^{(t)})$ time. Then we generalize the proof to arbitrary k. In the outermost for-loop of Alg. 1, node i denotes the node with the larger degree in the updated edge (i,j). Step 3 and Step 9 require $O(d^{(t)}(i))$ time, and Step 4 requires $O(\sum_{(i,j)\in\Delta E^{(t)}} d^{(t)}(j))$ time, where $d^{(t)}(u)$ denotes the standard (i.e., edge-based) degree of the node u and $d^{(t)}(u) = \sum_{v\in V^{(t)}} A^{(t)}(u,v)$. Therefore, for the total running time $O(updating)$, we have

$$\begin{aligned} O(updating) &= \sum_{i\in V_{\Delta E^{(t)}}} O(\, d^{(t)}(i) + \sum_{(i,j)\in\Delta E^{(t)}} d^{(t)}(j)) \\ &\leq \sum_{i\in V^{(t)}} O(\, d^{(t)}(i) + \sum_{(i,j)\in\Delta E^{(t)}} d^{(t)}(j)) \\ &\leq O(m^{(t)}) + O\,(\sum_{(i,j)\in\Delta E^{(t)}} d^{(t)}(j)) \\ &\leq O(m^{(t)}) + O\,(\sum_{(u,v)\in\Delta E^{(t)}} \min\{d^{(t)}(u),\ d^{(t)}(v)\}) \end{aligned} \tag{25}$$

where $V_{\Delta E^{(t)}}$ stands of the set of nodes in $\Delta E^{(t)}$. We assume that $|\Delta E^{(t)}| < m^{(t)}$, then according to [13], we have

$$\sum_{(u,v)\in\Delta E^{(t)}} \min\{d^{(t)}(u),\ d^{(t)}(v)\} \leq 2\alpha m^{(t)} \tag{26}$$

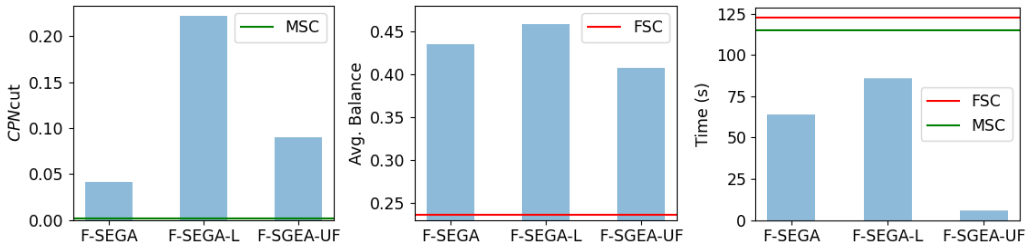
where α is the arboricity of the graph $G^{(t)}$ and stands for the minimum number of edge-disjoint spanning forests of $G^{(t)}$. Note that the bounded time complexity is independent of the number of updated edges due to the summation. Thus, we have the total running time $O(updating) \leq O(\alpha m^{(t)})$. With the recursion rule of listing k-clique [13], we can detect the appearance (or disappearance) of k-cliques at each timestamp by recursively applying Alg. 1 until k-1=2, and the total time complexity is bounded by $O(k\alpha^{k-2}m^{(t)})$.

C PROOF OF PROPOSITION 3

First of all, for Step 1 of Alg. 2, we have

$$\boldsymbol{u}_i^{(t)\prime} \Delta \boldsymbol{M} \boldsymbol{u}_j^{(t)} = \sum_{(s,r)} \Delta M(s,r)\, u_i^{(t)}(s)\, u_j^{(t)}(r) \tag{27}$$

where (s,r) stands for the non-zero entry of $\Delta\boldsymbol{M}$. We denote the number of non-zero entries of $\Delta\boldsymbol{M}$ as o, then Step 1 costs $O(q^2 o)$ time complexity. Then, compared with the second for-loop (i.e., Steps 14–17), it is easy to prove that the dominant part of Alg.2 is



(a) Ablation in ASA (w.r.t. $CPNcut$). **(b) Ablation in ASA (w.r.t. Avg. Balance).** **(c) Ablation in ASA (w.r.t. Time).**

Figure 7: Ablation Study of F-SEGA.

the first for-loop (i.e., Steps 5–12) [10], where the inverse operation of Step 7 costs $O(q^3)$, and the multiplication operation of Step 11 costs $O(nq)$. Therefore, the first for-loop costs $O(q^4 + nq^2)$. Since $o < n$, Alg.2 will cost $O(q^4 + nq^2)$ in the worst case.

D PROOF OF THEOREM 4.1

F-SEGA consists of two sequential parts, i.e., Laplacian update and eigen-pairs update. Therefore, the time complexity is easy to be proved by Proposition 2 and Proposition 3.

E ABLATION STUDIES

Here, we provide ablations by removing different components of F-SEGA individually. (1) By removing the fairness constraint (i.e., Eq. 9), F-SEGA-UF solves the clique-preserving spectral clustering in the dynamic setting. (2) By replacing the clique-weighted adjacency matrix $\boldsymbol{W}$ with the standard adjacency matrix $\boldsymbol{A}$, F-SEGA-L targets to the low-order and fair clustering in the dynamic setting.

Regarding the ablation study in ASA data as shown in Figure 7a-7c, where the dynamics setting is the same as other datasets, and the number of clusters q is set to be 2. We can see each proposed technique plays its own role. For example, F-SEGA-L fails to find high-order dense clustering, and F-SEGA-UF could not identify a fair clustering. For the time complexity, since the high-order connections make matrices more sparse, F-SEGA runs faster than F-SEGA-L, and F-SEGA-UF runs fastest by avoiding intensive inverse operations for fairness constraints.

F LIMITATIONS

In Eq. 6, matrix $\boldsymbol{H}$ is a node-cluster assignment matrix, where each node belongs to only one cluster. Given the format of $\boldsymbol{H}$ in Eq. 6, the trace of $\boldsymbol{H}^\top \boldsymbol{L}\boldsymbol{H}$ exactly equals to $CPNcut$ in Eq. 5. That's why minimizing $CPNcut$ equals to minimizing the trace of $\boldsymbol{H}^\top \boldsymbol{L}\boldsymbol{H}$.

Then, if we can obtain $\boldsymbol{H}$ as Eq. 6 expressed (i.e., 0 and $1/\sqrt{\mu(C_l, \mathbb{N})}$ valued), the clusters can be directly read out from $\boldsymbol{H}$. However, in a more general case, we need to relax $\boldsymbol{H}$ by letting its entries take any real values. That's where the approximation originates, and also the reason we need K-means to infer clusters from the real-valued matrix $\boldsymbol{H}$.

Using K-means is an effective approximation for classic spectral clustering [36, 47]. In our setting, i.e., adding fairness constraints to clustering (i.e., Eq. 3 + Eq. 4), using K-means also has good empirical performance as shown in our experiments or in [29]. To the best of our knowledge, the theoretical bound of compactness and fairness for using K-means is still an open problem. In [29], the authors proved that using K-means (in the static setting) can partition an SBM-generated synthetic graph with a bounded accuracy, but without fairness error analysis. Currently, K-means is a commonly used and necessary way to infer clustering, and analyzing the exact compactness and fairness error bound in the general case is a very interesting topic.