Prerequisite-driven Fair Clustering on Heterogeneous Information Networks

JUNTAO ZHANG, School of Computer Science, Wuhan University, China
SHENG WANG*, School of Computer Science, Wuhan University, China
YUAN SUN, School of Business, La Trobe University, Australia
ZHIYONG PENG*, School of Computer Science & Big Data Institute, Wuhan University, China

This paper studies the problem of fair clustering on heterogeneous information networks (HINs) by considering constraints on structural and sensitive attributes. We propose a Prerequisite-driven Fair Clustering (**PDFC**) algorithm to solve this problem. Specifically, we define the structural constraint on the connection among nodes in HINs by combining meta-paths and prerequisite meta-paths and introduce *Fairlets* as the balance constraint. Under two constraints, we learn node embeddings based on graph models and perform the *Cholesky decomposition* to obtain their orthogonal embeddings. We fuse node embeddings under constraints, define the loss function of **PDFC**, and perform k-means to achieve clustering. In addition, we design an update strategy of the adjacency matrix to achieve dynamic **PDFC** over time. Compared with several fair clustering algorithms on three real-world datasets, our experimental results verify the effectiveness and efficiency of **PDFC**.

CCS Concepts: • **Information systems → Clustering**.

Additional Key Words and Phrases: fair clustering, structural constraints, balance constraints, dynamic clustering

ACM Reference Format:
Juntao Zhang, Sheng Wang, Yuan Sun, and Zhiyong Peng. 2023. Prerequisite-driven Fair Clustering on Heterogeneous Information Networks. *Proc. ACM Manag. Data* 1, 2, Article 122 (June 2023), 27 pages. https://doi.org/10.1145/3589267

1 INTRODUCTION

Biases related to sensitive attributes such as gender, race, and prestige, are well known to drive differences in scientific output and impact, and these inequalities are widespread in science [40]. Fortunately, fair algorithms have received extensive attention in machine learning and data science, such as clustering [8, 19, 20, 38, 50], classification [29, 49, 70], ranking [6, 21], outlier detection [46, 53], among others, which help us eliminate data-inherent bias.

Currently, most studies [2, 3, 7, 8, 19, 24, 38, 72] follow the *Fairlets* in [20] to deal with the fair clustering problem of sensitive attributes in Euclidean space. These studies integrate the *Fairlets* into the objective of different clustering algorithms to ensure each cluster has a proportional representation of sensitive attributes. Fair clustering algorithms on graphs require all sensitive

*Corresponding authors.

Authors' addresses: Juntao Zhang, School of Computer Science, Wuhan University, Wuhan, China, juntaozhang@whu.edu.cn; Sheng Wang, School of Computer Science, Wuhan University, Wuhan, China, swangcs@whu.edu.cn; Yuan Sun, School of Business, La Trobe University, Melbourne, Australia, yuan.sun@latrobe.edu.au; Zhiyong Peng, School of Computer Science & Big Data Institute, Wuhan University, Wuhan, China, peng@whu.edu.cn.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
2836-6573/2023/6-ART122 $15.00
https://doi.org/10.1145/3589267

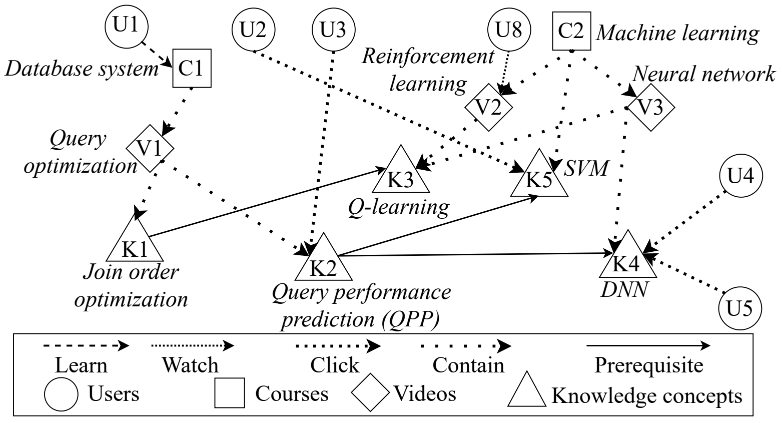


Fig. 1. A heterogeneous information network.

subgroups to be proportionally represented by nodes in each cluster [22, 37]. For example, fair graph clustering [37, 39, 48] requires sensitive attributes to be covered in each cluster, which has been applied to sensitive attribute prediction [11] and friendship recommendations [47]. Several studies [12, 15] address the fair graph partitioning problem, which ensures that each cluster forms a connected subgraph and applies it to subgraph partitioning [15].

Although fair algorithms have attracted much attention, only a few studies have considered eliminating biases toward sensitive attributes via heterogeneous information networks (HINs) [69]. HINs can model real-world data through entity types and their relationships [54]. For example, Figure 1 shows a HIN that models users (U,◯), courses (C,□), videos (V,◇), and knowledge concepts (K,△) as different entity types and describes various relations among entities. Zeng et al. [69] proposed a fair algorithm to mitigate gender bias in automated career counseling, but they did not consider users' structural or collaboration fairness in HINs. Inequalities are common in scientific communities, and most efforts to understand them treat scientists as isolated individuals and ignore the network effects of collaboration [40]. Different from [69], we study the problem of fair clustering on HINs, which aims to form k clusters that require structures (e.g., connectivity) and sensitive attributes to be proportionally represented by nodes and their relations.

Networks (or graphs) may contain latent background knowledge, for example, two nodes may be cannot-link (in different clusters) or must-link (in the same cluster) [36], and prerequisite or citation relations. Latent knowledge cannot be captured or directly utilized by vanilla clustering algorithms. Recently, several studies have utilized them to improve clustering algorithms. For example, Chatziafratis et al. [17] have implemented clustering with the important prior cannot-link and must-link information as constraints. Jong et al. [30] utilize prerequisite relations between knowledge concepts as the prior constraint rule to group users to achieve their knowledge complementarity. Additionally, meta-paths [56] in HINs can construct the connection among target nodes via other nodes, which has been applied in clustering [42, 43] and classification [41] tasks. Inspired by the successful application of prerequisite relations and meta-paths, we consider setting relevant constraints to ensure a connection between target nodes in HINs.

As shown in Figure 1, we take *artificial intelligence (AI) driven database optimization* as an example to describe our study problem. For instance, users want to learn the knowledge of *AI-driven query performance prediction (*QPP*)*, but they are not familiar with the knowledge of *AI*, so *AI* is the subsequent knowledge that users need. How can they find latent users with *AI* knowledge to form learning groups, or existing algorithms that assign appropriate learning partners to them? In reality, we can help users find learning partners or groups via potential path information in HINs. In Figure

1, user U3 has the database knowledge of *QPP*, and user U4 or U5 knows *deep neural networks* (*DNN*). Thus, users U4 and U5 can establish the connection via common knowledge of *DNN*, and they have the social principle of homophily [31] when participating in activities of groups [65], which will help them communicate more easily during their learning process. In addition, users U3 and U4 (U5) can establish another connection through the prerequisite relationship between *QPP* and *DNN* to form learning groups, and they have heterogeneity [28] when participating in the activities of groups, which helps them to cooperate in the learning process. To represent these two types of connections among users in HINs, we define meta-paths (Definition 3) and prerequisite meta-paths (Definition 4), respectively. According to these two definitions, we define structural constraints (Definition 5) that require each cluster to contain these two types of path information. In addition, to avoid the unfairness of sensitive attributes, we also follow the *Fairlets* in [20] to define balance constraints (Definition 6). Thus, we aim to achieve fair clustering on HINs under structural and balance constraints in this paper.

Motivated by the above observations, we introduce the objective of *Normalized Cut* [52] (*NCut*) as the objective of our fair clustering under structural and balance constraints. The *NCut* problem achieves a *balance* of clusters by edge weights [58]. Unfortunately, introducing the *balance* condition makes the *NCut* problem become NP-hard[14]. Our fair clustering becomes a bi-objective minimization problem that forms k-fair clusters over a set of N target nodes in HINs, which is also NP-hard [23] when two constraints are considered simultaneously. Therefore, we propose a Prerequisite-drive Fair Cluster (**PDFC**) algorithm to solve this problem. Specifically, we utilize an attention-based embedding learning model (Section 4.2) and graph convolutional networks [35] to learn the embeddings of nodes under two constraints, respectively. Finally, we transform the bi-objective minimization into a single-objective minimization problem by unifying the embeddings of target nodes in HINs, which is used as the loss function of our fair clustering.

The network (graph) data may change over time, such as the node addition, deletion, and update, which makes previous clustering results invalid. Nevertheless, existing fair clustering algorithms cannot capture changes in the overall structure of network (graph) data, hence inapplicable to online fair clustering. Currently, several studies [16, 27, 32, 33, 66] of unfair clustering algorithms deal with time-varying or streaming data through sliding window models or evolutionary clustering to achieve dynamic clustering. Consequently, we design an updating strategy of the adjacency matrix based on sliding window models to set the importance of graph structure updates. Then we utilize **PDFC** learning on the updated adjacent matrix to obtain fair clustering. The source code and data, and other artifacts have been made available [1].

To summarize, this paper makes the following contributions:

- We study the problem of fair clustering on HINs by considering the balanced distribution of structures and sensitive attributes in clusters. We design a novel specific similarity path, denoted as the *prerequisite meta-path*, to construct the connection of target nodes in HINs. We define the loss function of fair clustering as the objective minimization and explain it is NP-hard (Section 3).
- We propose **PDFC**. **PDFC** learns potential embeddings of structural information and sensitive attribute between nodes, then obtains their orthogonal embeddings as node embeddings under structural and balance constraints by performing *Cholesky decomposition*. We convert the bi-objective into single-objective minimization by concatenating node embeddings (Section 4).
- We design an updating strategy of the adjacency matrix to implement the dynamic **PDFC** over time (Section 5).
- We conduct extensive experiments on three real-world datasets to verify the effectiveness and efficiency of **PDFC** (Section 6).

[1]https://github.com/zhang-juntao/PDFC

2 RELATED WORK

Fair Clustering. Vanilla clustering algorithms aim to partition unlabeled data into similar groups, such as k-means for clustering trajectory [60] and DBSCAN for clustering heterogeneous networks [18]. Compared with vanilla clustering algorithms, fair clustering not only ensures that unlabeled data are in similar groups but also focuses on counteracting biases towards sensitive attributes in clustering results. On the one hand, most studies [2, 3, 7, 8, 19, 24, 38, 72] integrate the *Fairlets* of [20] into their clustering objectives (e.g., k-means [61] and k-median [13]) to eliminate biases toward sensitive attributes in Euclidean space. For example, Bera et al. [8] extended the model of [20] to get fair clustering algorithms for any p-norm objective to reduce the cost of clustering if allowing for small additive violations to the fairness constraint. Abraham et al. [3] focused on fair clustering for multiple sensitive attributes and incorporated a novel fairness loss term, which nudges the clustering towards fairness on the set of sensitive attributes, into the k-means, called Fair k-means. Ziko et al. [72] proposed a general framework of fair clustering, which integrates a Kullback-Leibler fairness term with clustering objectives.

On the other hand, several studies [11, 12, 15, 37, 39, 47, 48] focus on fair clustering on graphs to eliminate biases inherent against sensitive attributes. For instance, Kleindessner et al. [37] first tried to incorporate the *Fairlets* of [20] into the graph-based objective (e.g., spectral clustering [58]) by embedding linear constraints on the graph Laplacian matrix to achieve fair clustering on graph data. Li et al. [39] proposed FairAdj to empirically learn a fair adjacency matrix by updating the normalized adjacency matrix while keeping the original graph structure unchanged.

Dynamic Clustering. With the increase of streaming or time-varying data, dynamic clustering or incremental clustering has attracted extensive attention [18, 66]. Several current studies employ the sliding window model [10, 32, 33, 44] and evolutionary clustering [5, 16, 27] to realize dynamic or incremental clustering. Kim et al. [33] presented the *DISC*, which can carry out the clustering tasks for streaming data on time without compromising the quality of clustering results or consuming excessive computational resources. Subsequently, Kim et al. [32] proposed the *DenForest*, which can accurately examine whether the underlying graph is being split or not by a new data structure *DenTree*, which then determines efficiently and accurately whether a cluster is to be split by a point removed from the window in logarithmic time.

The evolutionary clustering processing timestamped data to produce a sequence of clusters [16]. Gu et al. [27] proposed DynamicC for clustering in high-velocity dynamic scenarios (continuously updated, inserted, and deleted.) by previous clustering results. DynamicC observes the cluster evolution patterns by an existing batch clustering algorithm in the training phase and makes clustering decisions in reaction to data operations in the prediction phases. You et al. [66] proposed a Robust Temporal Smoothing Clustering that implements dynamic graph clustering of temporal networks to overcome the problem of considerable noise during temporal information smoothing, high time complexity, etc.

Remarks. (1) Most of the existing studies consider the fair clustering of sensitive attributes in Euclidean space. However, except for [69], few studies consider eliminating biases toward sensitive attributes using the rich information of HINs. In addition, there is a lack of relevant research on the structural fairness of graphs or networks; (2) Inspired by the successful application of sliding window models, we consider capturing the dynamic changes of HINs at different times using the time-based sliding window model. Then we recalculate the adjacency matrix formed by the target node at the corresponding time and design an update strategy of the adjacency matrix to expand PDFC for fair clustering over time.

3 DEFINITIONS

To represent entities (e.g., user, video) and their relations (e.g., users watch videos), we model them as a heterogeneous information network (HIN) [26], as shown in Fig. 1. Before defining the fair clustering problem, we introduce several preliminaries about HINs. Table 1 lists the notations used.

Table 1. Descriptions of notations

Notations	**Description**
$\mathcal{G}$	The heterogeneous information network
$\mathcal{V}, \mathcal{E}$	The set of nodes and the set of edges
$\mathcal{T}, \mathcal{R}$	Nodes and edges correspond to the set of entity and relation types
V_i, E_j	Set of node and edge in the type $\mathcal{T}_i$ and $\mathcal{R}_j$
$\mathcal{G}_S$	The network schema of HINs
ϕ, ψ	The node and relation mapping function
$\mathcal{P}, \mathcal{PP}$	Set of meta-paths and prerequisite meta-paths
R, R_p	The composite relation on $\mathcal{P}$ and $\mathcal{PP}$
$G = (V, E, X)$	A graph G is formed by $\mathcal{P}$ and $\mathcal{PP}$, V is the node set of target entity, E is the edge between V, and X is the feature of V
$\boldsymbol{A}$	The adjacency matrix of G
$\boldsymbol{A}_{K_{i'} \mapsto K_{j'}}$	Prerequisite meta-path connection structure
$\boldsymbol{A}_{K_{i'}}$, $\boldsymbol{A}_{K_{j'}}$	Meta-path connection structure
$C = \{C_1, ..., C_k\}$	k disjoint clusters
V_s	A sensitive attribute group

3.1 Preliminaries

Definition 1. *(**Heterogeneous Information Networks**) [26, 55] An HIN $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{V} = \bigcup_{i=1}^{n} V_i$ is a set of nodes, $\mathcal{T} = \{T_1, ..., T_n\}$ is a set of n entity types in $\mathcal{V}$, and V_i is the node set about the entity type T_i. $\mathcal{E} = \bigcup_{j=1}^{m} E_j$ is a set of edges, $\mathcal{R} = \{R_1, ..., R_m\}$ is a set of m relation types between entities in $\mathcal{T}$, and E_j is the edge set about the relation types R_j. An HIN requires that $|\mathcal{T}| + |\mathcal{R}| > 2$.*

Definition 2. *(**Network Schema**) [26, 55] The network schema is a meta template of an HIN $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, called $\mathcal{G}_S = (\mathcal{T}, \mathcal{R})$, which shows the relations between entity types via the relation types in $\mathcal{R}$.*

The $\mathcal{G}_S = (\mathcal{V}, \mathcal{E})$ has two mappings: (1) an entity-type mapping ϕ: $\mathcal{V} \longrightarrow \mathcal{T}$ maps an entity of $\mathcal{V}$ into its types in $\mathcal{T}$; (2) a relation-type mapping ψ: $\mathcal{E} \longrightarrow \mathcal{R}$ maps an edge in $\mathcal{E}$ into its relation types in $\mathcal{R}$. Figure 2 is the network schema of Figure 1.

Definition 3. *(**Meta-Paths**) [55] A meta-path is a path that connects two entities of the same type via other entity types on the network schema $\mathcal{G}_S = (\mathcal{T}, \mathcal{R})$, denoted as $\mathcal{P} : T_1 \xrightarrow{R_1} T_2 \xrightarrow{R_2} ... \xrightarrow{R_l} T_{l+1}$. Meta-path $\mathcal{P}$ describes a composite relation $R = R_1 \circ R_2 \circ ... \circ R_l$ between T_1 and T_{l+1}, where $\circ$ is the composition operator on relations, and T_1 and T_{l+1} are the same entity types.*

Example 1. *If two nodes v_i and v_j are related by the composite relation R in $\mathcal{G}$, there is a path between them, called a **path instance** $p_{v_i \rightsquigarrow v_j}$ of $\mathcal{P}$. Therefore, we acquire all path instances based on the meta-path $\mathcal{P}$, denoted as $p_{v_i \rightsquigarrow v_j} \vdash \mathcal{P}$. In Figure 1, we select three types of meta-paths to model relatedness between the target entity (U, users) in $\mathcal{G}$ by clicking on the entity (K, knowledge concepts), including $\mathcal{P}_1$ (U $\xrightarrow{1}$ K $\xleftarrow{1}$ U), $\mathcal{P}_2$ (U $\xrightarrow{1}$ V $\xrightarrow{1}$ K $\xleftarrow{1}$ V $\xleftarrow{1}$ U), and $\mathcal{P}_3$ (U $\xrightarrow{1}$ C $\xrightarrow{1}$ K $\xleftarrow{1}$ C $\xleftarrow{1}$ U), where 1 represents that there is a relation between entities in Figure 2. In Figure 1, a specific instance of*

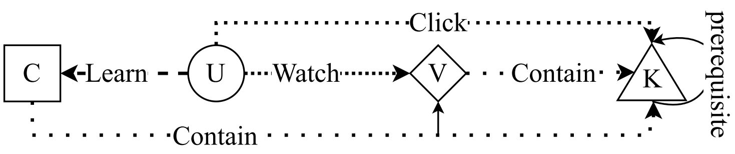


Fig. 2. Network schema of the HIN.

the meta-path $\mathcal{P}_1$ *is* U4 $\xrightarrow{1}$ K4 $\xleftarrow{1}$ U5, *which represents* U4 *and* U5 *connected by clicking and learning the knowledge concept* K4.

Definition 4. ***(Prerequisite Meta-Paths)*** *A prerequisite meta-path denotes a path that connects two entities of the same type via the dependency relationship of other entity types on the network schema* $\mathcal{G}_S = (\mathcal{T}, \mathcal{R})$, *denoted as* $\mathcal{PP}$: $T_1 \xrightarrow{R_1} ...T_i \overset{PR_i}{\longmapsto} T_j ... \xrightarrow{R_l} T_{l+1}$. $\mathcal{PP}$ *connects the same type of entities* T_1 *and* T_{l+1} *based on a composite relation* $R_p = R_1 \circ ... \circ PR_i \circ ... \circ R_l$, *where* $\circ$ *is the composition operator on relations.* PR_i *is the prerequisite relationship between* T_i *and* T_j.

Example 2. *We suppose that the nodes* v_i *and* v_j *click on the knowledge concept* $K_{i'}$ *and* $K_{j'}$, *respectively, and* $K_{i'}$ *is a prerequisite knowledge concept of* $K_{j'}$. *Thus,* v_i *and* v_j *are related by the composite relation* R_p *in* $\mathcal{G}$. *As there is a path between them, we say this path is a* ***prerequisite path instance*** $p_{v_i \rightsquigarrow ... K_{i'} \mapsto K_{j'} ... \rightsquigarrow v_j}$ *of* $\mathcal{PP}$. *We obtain all prerequisite path instances based on prerequisite meta-paths* $\mathcal{PP}$, *denoted as* $p_{v_i \rightsquigarrow ... K_{i'} \mapsto K_{j'} ... \rightsquigarrow v_j} \vdash \mathcal{PP}$. *In Figure 1, we also select three prerequisite meta-paths to denote the connections among the entity (U) in the HIN by clicking on the entity (K), including* $\mathcal{PP}_1$ (U $\xrightarrow{1}$ $K_{i'}$ $\mapsto$ $K_{j'}$ $\xleftarrow{1}$ U), $\mathcal{PP}_2$ (U $\xrightarrow{1}$ V $\xrightarrow{1}$ $K_{i'}$ $\mapsto$ $K_{j'}$ $\xleftarrow{1}$ V $\xleftarrow{1}$ U), *and* $\mathcal{PP}_3$ (U $\xrightarrow{1}$ C $\xrightarrow{1}$ $K_{i'}$ $\mapsto$ $K_{j'}$ $\xleftarrow{1}$ C $\xleftarrow{1}$ U), *where* $\mapsto$ *represents the prerequisite relation between* $K_{i'}$ *and* $K_{j'}$. *A specific instance of the prerequisite meta-path* $\mathcal{PP}_1$ *is* U3 $\xrightarrow{1}$ K2 $\mapsto$ K5 $\xleftarrow{1}$ U2, *which denotes* U3 *and* U2 *connected by clicking and learning these two knowledge concepts* K2 *and* K5.

In HIN $\mathcal{G}$, we give the entity set K with the prerequisite relationships and the target entity set V. Therefore, we use connections among target entities in HIN $\mathcal{G}$ via *meta-paths* and *prerequisite meta-paths* to construct an undirected graph, denoted as $G = (V, E, X)$. $V = \{v_1, ..., v_N\}$ is the node set formed by the target entity (user), E is the edge among nodes, $X \in \mathbb{R}^{N \times F}$ is the feature of nodes, N is the number of nodes, and F is the dimension of features. The structure of graph G contains two types, i.e., the *meta-path connection structure* and the *prerequisite meta-path connection structure.* Each edge between nodes v_i and v_j in G carries a weight $\boldsymbol{A}_{ij} > 0$, which means v_i and v_j are connected. The *adjacency matrix* of G is $\boldsymbol{A} \in \mathbb{R}^{N \times N} = (\boldsymbol{A}_{ij})_{i,j \in \{1,...,N\}}$, where $\boldsymbol{A}_{ij} = \boldsymbol{A}_{ji}$.

Assume that $\boldsymbol{A}_{K_{i'} \mapsto K_{j'}} = \{p_{\underset{\rightsquigarrow}{K_{i'} \mapsto K_{j'}}} : p_{\underset{\rightsquigarrow}{K_{i'} \mapsto K_{j'}}} \vdash \mathcal{PP}\}$ (calculated in Section 4.1.2) is the weight of the *prerequisite meta-path connection structure* formed on $K_{i'} \mapsto K_{j'}$, and $\boldsymbol{A}_{K_{i'}} = \{p_{\underset{\rightsquigarrow}{K_{i'}}} : p_{\underset{\rightsquigarrow}{K_{i'}}} \vdash \mathcal{P}\}$ and $\boldsymbol{A}_{K_{j'}} = \{p_{\underset{\rightsquigarrow}{K_{j'}}} : p_{\underset{\rightsquigarrow}{K_{j'}}} \vdash \mathcal{P}\}$ (calculated in Section 4.1.1) are the weight of the *meta-path connection structure* formed on $K_{i'}$ and $K_{j'}$, respectively. Consequently, we utilize $\boldsymbol{A}_{K_{i'}; \mapsto; K_{j'}} = \boldsymbol{A}_{K_{i'}} \cup \boldsymbol{A}_{K_{i'} \mapsto K_{j'}} \cup \boldsymbol{A}_{K_{j'}}$ to denote the overall structure of G.

Definition 5. ***(Structural Constraints)*** *Given* k *disjoint clusters* $C = \{C_1, ..., C_k\}$ *of* G, *we require that each cluster has a similar structure to* G *under the constraints of* $\boldsymbol{A}_{k_{i'}}$, $\boldsymbol{A}_{k_{i'} \mapsto k_{j'}}$, *and* $\boldsymbol{A}_{k_{j'}}$. *The definition of structural constraints in cluster* C_l *is as follows:*

$$structural(C_l) = min \sum_{K_{i'}, K_{j'} \in K_{C_l}} \frac{\boldsymbol{A}_{K_{i'} \mapsto K_{j'}}}{\boldsymbol{A}_{K_{i'}} + \boldsymbol{A}_{K_{j'}}}, \tag{1}$$

where K_{C_l} *is the entity set with prerequisite relationships in* C_l.

Example 3. *In Figure 1, we assume that users U3, U4, and U5 form a cluster (called C_l) that contains the* meta-path connection structure *and* prerequisite meta-path connection structure. *Thus, this cluster satisfies the fairness of the structural constraint. If it is similar to [42] that only the meta-path is considered, then U3 cannot form clusters with U4 (U5) and does not meet the structural constraint.*

Definition 6. *(**Balance Constraints**) [20, 37] Assume that nodes have h different demographic groups relating to sensitive attributes, nodes denoted as $V = \bigcup_{s \in [h]} V_s$ (the feature denoted as $X_h \in \mathbb{R}^{N \times h}$). Given k disjoint clusters $C = \{C_1, ..., C_k\}$ of G, we require that each cluster has a proportional representation of different demographic groups. The definition of balance constraints satisfies:*

$$balance(C_l) = \min_{s \neq s' \in [h]} \frac{|V_s \cap C_l|}{|V_{s'} \cap C_l|} \in [0, 1], \tag{2}$$

where the higher the balance constraint value of cluster C_l, the fairer C_l.

Example 4. *We refer to Example 3 to describe the balance constraint. If one of the users U3, U4, and U5 in the cluster C_l is female and two of the users are male, we can get $balance(C_l) = 0.5$ according to Equation (2).*

3.2 Problem Definitions

In this section, we define the problem of fair clustering under structural and balance constraints. We introduce the *Normalized Cut* [52] (*NCut*) objective, which normalizes the cut by the total number of connections between each cluster to the rest of the graph, as the loss function of our fair clustering. In reality, we achieve fair clustering by learning the embedding of target nodes (users) in HINs. First, we learn latent embeddings of nodes through information dissemination between them in the adjacency matrix $\boldsymbol{A}$ and ensure each column of learned latent embeddings is constrained to be orthogonal. Then, we concatenate the learned orthogonal embeddings of nodes under these two constraints. Next, we calculate the orthogonal eigenvectors of nodes corresponding to the k smallest eigenvalues. Finally, we combined the learned orthogonal embeddings and orthogonal eigenvectors of nodes as the input of k-means to generate clustering. More details will be introduced in Section 4.

Definition 7. *(**NCut Objective**) [52, 58] Given an undirected graph G, the adjacency matrix $\boldsymbol{A}$, an integer k, the goal is to partition G into k disjoint clusters by minimizing the NCut objective.*

$$NCut(C_1, ..., C_k) = \sum_{l=1}^{k} \frac{Cut(C_l, \bar{C_l})}{vol(C_l)}, \tag{3}$$

where $Cut(C_l, \bar{C_l}) = \sum_{i \in C_l, j \in \bar{C_l}} \boldsymbol{A}_{ij}$, $\bar{C_l}$ is the complement of C_l, $vol(C_l) = \sum_{i \in C_l} d_i$ denotes the size of C_l by the sum of weights among nodes in C_l, and d_i is the degree of the ith *node.*

To approximate the objective of *NCut*, we introduce the indicator matrix $H \in \mathbb{R}^{N \times k} = \{H_1, ..., H_k\}$, which includes k indicator vectors $H_l = \{H_{1l}, ..., H_{il}, ...H_{Nl}\}^T (i \in \{1, ..., N\}, l \in \{1, ..., k\})$ as columns to encode clusters $C = \{C_1, ..., C_k\}$, T is the transpose. We denote H_{il} as follows:

$$H_{il} = \begin{cases} \frac{1}{\sqrt{vol(C_l)}}, & \text{if } v_i \in C_l \\ 0, & \text{if } v_i \notin C_l \end{cases}. \tag{4}$$

We set $NCut(C_1, ..., C_k) = Tr(H^T LH)$ according to the calculation of [58], H_{il} of Equation (4) satisfies $H^T DH = I_k$, and D is the *degree matrix* of $\boldsymbol{A}$, $L = D - \boldsymbol{A}$ is the unnormalized graph Laplacian matrix, and Tr is the trace of a matrix. We modify Equation (3) as the problem of minimizing $Tr(H^T LH)$ over H by

$$\min_{H \in \mathbb{R}^{N\times k}} Tr(H^T LH), \quad s.t.\ H^T DH = I_k. \tag{5}$$

Inspired by using the learned node spectral embeddings instead of H in [71], we also consider utilizing the embedding of nodes in G to replace H. We let X and X_h be the original feature of nodes under structural and balance constraints and use them to replace H in Equation (5) to form the bi-objective of fair clustering by

$$\min_{X, X_h \in \mathbb{R}^{N\times k}} Tr(X^T LX),\ Tr(X_h^T LX_h),\ s.t.\ X^T DX,\ X_h^T DX_h = I_k. \tag{6}$$

According to [14], the problem of minimizing the objective of the *NCut* is NP-hard. The bi-objective minimization problem of our fair clustering is also NP-hard [23] when two constraints are considered simultaneously. The complexity of our fair clustering lies within the size of users of HINs. One way to solve the bi-objective minimization problem is to transform it into a single-objective problem. We modify Equation (6) as the single-objective minimization to define the loss function of our fair clustering.

Definition 8. ***(The Loss Function of Fair Clustering)*** *Given an undirected graph G, the adjacency matrix $\boldsymbol{A}$, an integer k, and the original feature X and X_h under structural and balance constraints to replace H. The loss function of our fair clustering is as follows:*

$$\begin{aligned} \mathcal{L}(\Theta) = Tr(f_\Theta(X)^T L f_\Theta(X)) + Tr(f_\Theta(X_h)^T L f_\Theta(X_h)), \\ s.t.\ f_\Theta(X)^T D f_\Theta(X),\ f_\Theta(X_h)^T L f_\Theta(X_h) = I_k, \end{aligned} \tag{7}$$

where f is a graph model, the weight Θ is the parameters of ***PDFC****.*

Equation (7) fuses Tr under two constraints into one to become the loss function of fair clustering. In Section 4, we learn the latent node embedding based on the original features X and X_h by using information dissemination of the adjacency matrix $\boldsymbol{A}$ to train Θ.

4 ALGORITHM

In this section, we describe a Prerequisites-driven Fair Clustering (**PDFC**) algorithm on HINs. Specifically, we first construct an *adjacency matrix* $\boldsymbol{A}$ of G based on meta-paths and prerequisite meta-path, which contain two path types of connection structures. Then, we learn the embedding information of target nodes along different path information in HINs. Finally, we transform the bi-objective under structural and balance constraints into a single-objective minimization problem and give the loss function of **PDFC**.

4.1 Adjacency Matrix Improvement

The two types of connection structures in G determine the closeness of relationships among nodes, while the quality of graph structure directly affects node embedding learning. Therefore, we form the *adjacency matrix* $\boldsymbol{A}$ of G via connections among target nodes based on *meta-paths* and *prerequisite meta-paths* in the HIN $\mathcal{G}$.

4.1.1 Connections Based on Meta-paths. In HINs, meta-paths have been effectively used to capture connections between the same entity types [41, 43]. Given a meta-path $\mathcal{P}_p$ of $\mathcal{G}$, we compute the connection weight between two nodes v_i and v_j by *PathSim* [55]:

$$S_{\mathcal{P}_p(v_i,v_j)} = \frac{2 \times |\{p_{v_i \rightsquigarrow v_j} : p_{v_i \rightsquigarrow v_j} \vdash \mathcal{P}\}|}{|\{p_{v_i \rightsquigarrow v_i} : p_{v_i \rightsquigarrow v_i} \vdash \mathcal{P}\}| + |\{p_{v_j \rightsquigarrow v_j} : p_{v_j \rightsquigarrow v_j} \vdash \mathcal{P}\}|}. \tag{8}$$

Given a set of meta-paths $\mathcal{P}$ of $\mathcal{G}$, we can derive a similarity matrix $S_{\mathcal{P}_p}$ for each meta-path $\mathcal{P}_p \in \mathcal{P}$ according to Equation (8). Then, we construct the *adjacency matrix* $\boldsymbol{A}_{\mathcal{P}}$, which represents the meta-path connection structure among nodes, based on similarity matrices sum of meta-paths $\mathcal{P}$:

$$\boldsymbol{A}_{\mathcal{P}} = \sum_{p=1}^{|\mathcal{P}|} \alpha_p S_{\mathcal{P}_p}, \tag{9}$$

where α_p denotes the importance of the meta-path $\mathcal{P}_p$ in $\mathcal{P}$. We apply the k-nearest neighbor algorithm [4] to select the number of neighbors (δ) of each node in the *adjacency matrix* $\boldsymbol{A}_{\mathcal{P}}$, i.e., the number of neighbors v_i under meta-paths does not exceed δ.

4.1.2 Connections Based on Prerequisite Meta-paths. Given a prerequisite meta-path $\mathcal{PP}_p$ of $\mathcal{G}$, v_i and v_j are nodes of the same type connected through $K_{i'}$ and $K_{j'}$ ($K_{i'} \mapsto K_{j'}$) on $\mathcal{PP}_p$. According to Definition 4, we refer to Equation (8) to design a prerequisite meta-path, called *PrePathSim*. We compute the connection weight between two nodes v_i and v_j by *PrePathSim* as follows:

$$S_{\mathcal{PP}_p}(v_i, v_j) = \frac{\{p_{v_i \rightsquigarrow \dots K_{i'} \mapsto K_{j'} \dots \rightsquigarrow v_j} \vdash \mathcal{PP}\}}{\sum_{v_s \in \mathcal{N}_{v_j}^{\mathcal{PP}_p}, K_{i'} \mapsto K_{j'} \in \mathcal{G}_K} \{p_{v_i \rightsquigarrow \dots K_{i'} \mapsto K_{j'} \dots \rightsquigarrow v_s} \vdash \mathcal{PP}\}}, \tag{10}$$

where $\mathcal{N}_{v_j}^{\mathcal{PP}_p}$ is a set of nodes connected to node v_j by the prerequisite meta-path $\mathcal{PP}_p$, but $\mathcal{N}_{v_j}^{\mathcal{PP}_p}$ does not include v_i, $\mathcal{G}_K$ is a set of prerequisite relationships.

Given a set of prerequisite meta-paths $\mathcal{PP}$ of $\mathcal{G}$, we can derive an *adjacency matrix* $S_{\mathcal{PP}_p}$ for the prerequisite meta-path $\mathcal{PP}_p \in \mathcal{PP}$ according to Equation (10). We construct the *adjacency matrix* $\boldsymbol{A}_{\mathcal{PP}}$, which is the prerequisite meta-path connection structure among nodes, based on *adjacency matrix* sum of prerequisite meta-paths $\mathcal{PP}$:

$$\boldsymbol{A}_{\mathcal{PP}} = \sum_{p=1}^{|\mathcal{PP}|} \beta_p S_{\mathcal{PP}_p}, \tag{11}$$

where β_p is the importance of each prerequisite meta-path $\mathcal{PP}_p$ in $\mathcal{PP}$.

As the prerequisite relationship is directional, $\boldsymbol{A}_{\mathcal{PP}}$ is an asymmetric matrix. Thus, we let $S_{\mathcal{PP}_p}(v_i, v_j)$ and $S_{\mathcal{PP}_p}(v_j, v_i)$ be equal on prerequisite meta-paths to ensure $\boldsymbol{A}_{\mathcal{PP}}$ is a symmetric matrix. Next, we add $\boldsymbol{A}_{\mathcal{PP}}$ and its transpose to modify $\boldsymbol{A}_{\mathcal{PP}}$.

$$\hat{\boldsymbol{A}}_{\mathcal{PP}} = \boldsymbol{A}_{\mathcal{PP}} + \boldsymbol{A}_{\mathcal{PP}}^T. \tag{12}$$

We also apply the k-nearest neighbor algorithm to select the number of neighbors (δ) of nodes in the *adjacency matrix* $\hat{\boldsymbol{A}}_{\mathcal{PP}}$. Finally, we construct the *adjacency matrix* $\boldsymbol{A}$ by a weighted sum of $\boldsymbol{A}_{\mathcal{P}}$ and $\hat{\boldsymbol{A}}_{\mathcal{PP}}$.

$$\boldsymbol{A} = \boldsymbol{A}_{\mathcal{P}} + \eta \hat{\boldsymbol{A}}_{\mathcal{PP}}. \tag{13}$$

where η is a trade-off hyper-parameter that decides the importance of $\hat{\boldsymbol{A}}_{\mathcal{PP}}$ for learning the structure of graph G. We set $\eta \in [0, 1]$, which not only verifies the effectiveness of prerequisite meta-paths but also improves the structure of the adjacency matrix.

Example 5. *We give a simple HIN that contains meta-path* ($\mathrm{U} \xrightarrow{1} \mathrm{K} \xleftarrow{1} \mathrm{U}$) *and prerequisite meta-path* ($\mathrm{U} \xrightarrow{1} \mathrm{K}_{i'} \mapsto \mathrm{K}_{j'} \xleftarrow{1} \mathrm{U}$). *According to Equations (8) and (10), we can obtain the node's meta-path connection weight and the prerequisite meta-path connection weight, respectively. Then, we construct the adjacency matrices $\boldsymbol{A}_{\mathcal{P}}$ and $\hat{\boldsymbol{A}}_{\mathcal{PP}}$ among nodes based on Equations (9) and (12). Finally, we set $\eta = 1$ to ensure that $\boldsymbol{A}_{\mathcal{P}}$ and $\hat{\boldsymbol{A}}_{\mathcal{PP}}$ are equally important in Equation (13). The whole construction process of adjacency matrix $\boldsymbol{A}$ is shown in Figure 3. Compared with existing approaches [41, 43] considering*

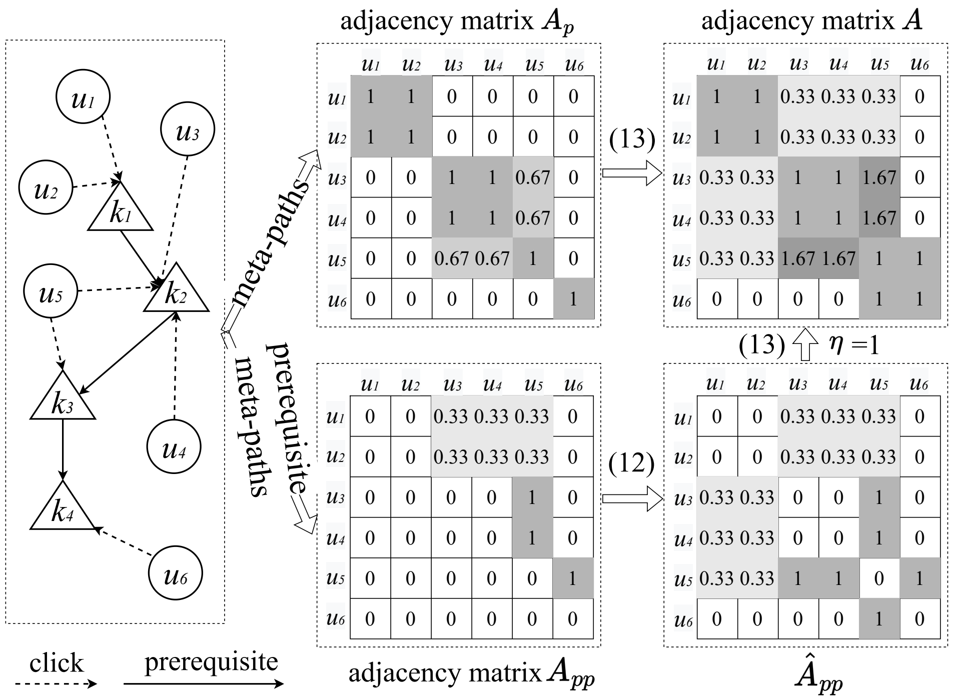


Fig. 3. An example of adjacency matrix improvement.

only meta-paths, the new adjacency matrix $\boldsymbol{A}$ *is improved according to the connection structures of meta-paths and prerequisite meta-paths, which enhances the connection among nodes.*

4.2 Attention-based Embedding Learning

In reality, as nodes have different neighbors in *adjacency matrix* $\boldsymbol{A} \in \mathbb{R}^{N \times N}$, their importance in meta-paths and prerequisite meta-paths are also different for specific tasks. Therefore, we present an attention-based embedding learning model to learn the potential embedding information among nodes. First, we introduce a node-level attention mechanism to learn the weights between the node and its connected neighbors under meta-paths and prerequisite meta-paths and aggregate them as node embeddings. Then, we design a path-level attention network to identify the importance of meta-paths and prerequisite meta-paths and fuse the structural information of these paths into node embeddings. The algorithm process is shown in Algorithm 1.

4.2.1 Node-Level Attention. Due to the heterogeneity of nodes in HIN, we need to map node features into the same feature space before aggregating the weights between nodes and their connected neighbors under meta-paths and prerequisite meta-paths. The process of mapping is as follows:

$$x_i' = M_{\phi_i} \cdot x_i, \tag{14}$$

where x_i and x_i' are the features of node i (replace v_i) before and after mapping, M_{ϕ_i} is the mapping matrix of type ϕ_i.

Let meta-paths $\mathcal{P} = \{\mathcal{P}_1, ..., \mathcal{P}_{|\mathcal{P}|}\}$ and prerequisite meta-paths $\mathcal{PP} = \{\mathcal{PP}_1, ..., \mathcal{PP}_{|\mathcal{PP}|}\}$ uniformly denote PATHS $\{\theta_1, ..., \theta_{|\mathcal{MP}|}\}$, where $|\mathcal{MP}|$ is the sum of $|\mathcal{P}|$ and $|\mathcal{PP}|$. We can calculate the weight between the node i and its PATHS-based neighbors $\mathcal{N}_i^{\theta}$ (include i) by performing an attention mechanism, $\mathcal{N}_i^{\theta}$ represents the structural information centered on node i on different PATHS. Given a node pair (i, j) connected via *Paths* θ, we obtain the normalized attention coefficient α_{ij}^{θ} between i and j as follows:

$$\alpha_{ij}^{\theta} = \frac{exp(\sigma(\mathbf{a}_{\theta}^{T} \cdot [x_i' || x_j']))}{\sum_{l \in \mathcal{N}_i^{\theta}} exp(\sigma(\mathbf{a}_{\theta}^{T} \cdot [x_i' || x_l']))}, \tag{15}$$

where σ is the ReLU nonlinearity function, $||$ is the concatenation operation, $\mathbf{a}_{\theta}$ is the node-level attention vector for *Paths* θ, and T represents transposition. Since *Paths*-based neighbors of i are different from those of node j, the importance of j to i is different from i to j, resulting in α_{ij}^{θ} being asymmetric.

We obtain the *Paths*-based embedding of node i (denoted as u_i^{θ}) by aggregating the attention coefficient α_{ij}^{θ} and the mapping features of node i's neighbors. To ensure that the training process is more beneficial, we repeat the node-level attention for L times and concatenate the learned embeddings of nodes. Thus, the node embedding u_i^{θ} through the *Paths* θ is calculated as follows:

$$u_i^{\theta} = \overset{L}{\underset{l=1}{||}} \sigma\Big(\sum_{j \in \mathcal{N}_i^{\theta}} \alpha_{ij}^{\theta} \cdot x_j'\Big). \tag{16}$$

For the unified set *Paths* $\{\theta_1, ..., \theta_{|\mathcal{MP}|}\}$ of meta-paths and prerequisite meta-paths, we obtain node embeddings along the *Paths*, denoted as $\{u^{\theta_1}, ..., u^{\theta_{|\mathcal{MP}|}}\}$.

4.2.2 Path-Level Attention. We know that the connections among nodes can reflect different structural information according to different *Paths*. To obtain more comprehensive node embeddings, we need to integrate the structure information of *Paths* into node embeddings. We design the path-level attention to identify the importance of *Paths* and fuse them as a part of node embeddings.

Let node embeddings $\{u^{\theta_1}, ..., u^{\theta_{|\mathcal{MP}|}}\}$ learned from the node-level attention be used as the input of path-level attention. Following [25, 62], we transform node embeddings $\{u^{\theta_1}, ..., u^{\theta_{|\mathcal{MP}|}}\}$ through nonlinear networks (e.g., fully connected networks). The importance of the path θ_p in *Paths* is called w_{θ_p}, which is calculated as follows:

$$w_{\theta_p} = \frac{1}{|\mathcal{MP}|} \sum_{p=1}^{|\mathcal{MP}|} q^T \cdot Tanh(W \cdot u_i^{\theta_p} + b), \tag{17}$$

where q is the path-level attention vector, $|\mathcal{MP}|$ is the number of *Paths*, W and b are the parameters on a nonlinear network with the *Tanh* activation function. Similar to node-level attention, we need to normalize the importance of all paths in *Paths* via *softmax function.* The normalized weight of path θ_p, denoted as β_{θ_p}:

$$\beta_{\theta_p} = \frac{exp(w_{\theta_p})}{\sum_{p=1}^{|\mathcal{MP}|} exp(w_{\theta_p})}, \tag{18}$$

where the size of β_{θ_p} represents the importance in *Paths*, and the importance of each path is different. According to the β_{θ_p}, we fuse it with node embeddings $\{u^{\theta_1}, ..., u^{\theta_{|\mathcal{MP}|}}\}$ to obtain the final embeddings of nodes containing the connection structure of *Paths*, denoted as $Z \in \mathbb{R}^{N \times LF'}$, as follows:

$$Z = \sum_{p=1}^{|\mathcal{MP}|} \beta_{\theta_p} u^{\theta_p}. \tag{19}$$

Algorithm 1: Attention-based embedding learning

```
Input: The heterogeneous information network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the PATHS $\{\theta_1, \theta_2, ..., \theta_{|\mathcal{MP}|}\}$, the feature
       of nodes $X$, the number of attention head $L$ ;
Output: The nodes embedding $Z$.
1  for each path $\theta_p \in \{\theta_1, \theta_2, ..., \theta_{|\mathcal{MP}|}\}$ do
2      for each attention head $l \in \{1, ..., L\}$ do
3          for target node $i \in \mathcal{V}$ do
4              The mapped feature $x_i'$ of node $i$;
5              Extract target nodes' neighbors $\mathcal{N}_i^{\theta_p}$ based on meta-paths and prerequisite meta-paths;
               /* According to Equations (8) and (10).                                        */
6              for $j \in \mathcal{N}_i^{\theta_p}$ do
7                  Calculate the attention weight coefficient $\alpha_{ij}^{\theta_p}$;
8          Concatenate the embedding for each attention head $u_i^{\theta_p} = \|\sigma(\sum_{j \in \mathcal{N}_i^{\theta_p}} \alpha_{ij}^{\theta_p} \cdot x_j')$;
9      Calculate the importance weights of nodes on each path $\beta_{\theta_p}$;
10     Obtain the node's final embedding $Z+ = \beta_{\theta_p} u^{\theta_p}$;
11 return $Z$;
```

4.3 Fair Clustering

In this section, we inject node embeddings Z containing structural information into Equation (6) to replace X as *structural constraints*. We also add the sensitive attribute embeddings of nodes into Equation (6) to replace X_h as *balance constraints*. Then, we learn the orthogonal embedding of nodes under two constraints. Next, we concatenate the orthogonal embedding of nodes to compute their orthogonal eigenvectors. Finally, we modify Equation (7) by combining the orthogonal embeddings and eigenvectors of nodes, and we perform k-means algorithm on them to achieve clustering.

4.3.1 Structural Constraints. To inject node embeddings Z into Equation (6), we leverage a fully connected network to map Z into k-dimensional space, denoted as $Z_1 \in \mathbb{R}^{N \times k}$:

$$Z_1 = ReLU(Z \cdot W_1 + b_1), \tag{20}$$

where $W_1 \in \mathbb{R}^{LF' \times k}$, $b_1 \in \mathbb{R}^k$ are learnable parameters, k is the dimension of features, and $ReLU$ is the activation function.

Next, we replace X with Z_1 and have $Z_1^T D Z_1 = I_k$ according to Equation (6). However, we need to ensure that the mapped node embeddings Z_1 are orthogonal to meet the requirements of spectral clustering. Inspired by this work [51] and $rank(I_k) = k$ is full rank, we obtain the QR decomposition by performing the *Cholesky decomposition* on $Z_1^T D Z_1 = Q_1 Q_1^T$, where $Q_1 \in \mathbb{R}^{k \times k}$ is a lower triangular matrix. Therefore, we set the orthogonal embeddings of nodes $\hat{Z}_1 \in \mathbb{R}^{N \times k}$, and the calculation form is as follows:

$$\hat{Z}_1 = D^{\frac{1}{2}} Z_1 (Q_1^{-1})^T, \tag{21}$$

where $Q_1^{-1} \in \mathbb{R}^{k \times k}$ is a learnable matrix.

4.3.2 Balance Constraints. Now, we consider how to incorporate Equation (2) into Equation (6) to minimize the *NCut* objective. Suppose that sensitive attributes of nodes in G contain h different demographic groups such that $\bigcup_{s \in [h]} V_s$. Let $f^{(s)} \in \{0, 1\}^N$ be the group-membership vector of V_s, $f_i^{(s)} = 1$ if $i \in V_s$ and $f_i^{(s)} = 0$ otherwise. Following [37], we integrate $C = \{C_1, ..., C_k\}$ of G into

$H \in \mathbb{R}^{N\times k}$ of Equation (4),

$$\begin{aligned} \forall s \in [h-1] : \sum_{i=1}^{N}(f_i^{(s)} - \frac{|V_s|}{N})H_{il} = 0 \Leftrightarrow \\ \forall s \in [h] : \frac{|V_s \cap C_l|}{|C_l|} = \frac{|V_s|}{N}. \end{aligned} \tag{22}$$

To ensure that the clustering results are as fair as possible, we add constraint condition $F^T H = 0_{(h-1)\times k}$ to Equation (5), where $F \in \mathbb{R}^{N\times(h-1)}$ is the matrix that has the vectors $f^{(s)} - (|V_s|/N)\cdot \mathbf{1}_N$. We know that *rank*(F) = *rank*(F^T) = h-1, and the number of clusters k satisfies: $k \leq N-h+1$. We compute the singular value decomposition (SVD) of F^T to construct a matrix $\tilde{Z} \in \mathbb{R}^{N\times(N-h+1)}$ whose columns form the orthonormal basis of the null space of F^T. Then, we feed $\tilde{Z}$ into a two-layer graph convolutional network to obtain the embeddings of k-dimensional space. Finally, we learn potential associations between nodes and their neighbors by the layer-wise propagation rule, which captures the spatial information of sensitive attributes of nodes on the graph, denoted as Z_2:

$$Z_2 = \sigma(\boldsymbol{A}\sigma(\boldsymbol{A}\tilde{Z}W^{(0)})W^{(1)}), \tag{23}$$

where $Z_2 \in \mathbb{R}^{N\times k}$ is the embeddings of k-dimensional, $W^{(0)} \in \mathbb{R}^{(N-h+1)\times k}$ and $W^{(1)} \in \mathbb{R}^{k\times k}$ are learnable parameter matrices, and σ is the Tanh activation function.

Similar to structural constraints, we replace X_h of Equation (6) with Z_2 and have $Z_2^T D Z_2 = I_k$, $Z_2^T D Z_2$ is positive definite. Then, we also perform the *Cholesky decomposition* on $Z_2^T D Z_2 = Q_2 Q_2^T$, where $Q_2 \in \mathbb{R}^{k\times k}$ is a lower triangular matrix. Therefore, we set the sensitive attribute embedding of nodes, denoted as $\hat{Z}_2 \in \mathbb{R}^{N\times k}$, and the calculation form is as follows:

$$\hat{Z}_2 = D^{\frac{1}{2}} Z_2 (Q_2^{-1})^T, \tag{24}$$

where $Q_2^{-1} \in \mathbb{R}^{k\times k}$ is a learnable matrix.

4.3.3 Loss Fusion of Fair Clustering. $\hat{Z}_1$ and $\hat{Z}_2$ ($f_\Theta(X)$ and $f_\Theta(X_h)$ in Definition 8) are the latent embeddings of nodes corresponding to X and X_h under two constraints, respectively. In reality, Equation (7) transforms Equation (6) into a single-objective minimization and serves as the loss function of fair clustering. However, it is difficult to trade-off the loss function under our two constraints. Thus, we concatenate $\hat{Z}_1$ and $\hat{Z}_2$, denoted as $\mathcal{Z} = [\hat{Z}_1 || \hat{Z}_2]$, and use $\mathcal{Z}$ to replace $f_\Theta(X)$ and $f_\Theta(X_h)$. We compute orthonormal eigenvectors corresponding to the k smallest eigenvalues of $\mathcal{Z}L\mathcal{Z}$, denoted as $E \in \mathbb{R}^{N\times k}$. We modify Equation (7) as follows:

$$\mathcal{L} = Tr(E^T \mathcal{Z}^T L \mathcal{Z} E), \quad s.t.\ E^T E = I_k. \tag{25}$$

Finally, we follow [58, 71] to perform the k-means algorithm on $\mathcal{Z}E$ to partition nodes of G into k disjoint clusters. As a result, our bi-objective becomes a single-objective minimization problem. Equation (25) is the loss function in the training process of the **PDFC** and is the optimized expression of Equation (7). The process of the **PDFC** algorithm is shown in Algorithm 2.

5 ONLINE ALGORITHM

The dynamic change of HINs over time may update the adjacency matrix $\boldsymbol{A}$ of G, and the update of the adjacency matrix $\boldsymbol{A}$ includes three situations: 1) old nodes are removed; 2) new nodes are added; 3) the status of nodes is updated. In this section, we design an updating strategy of the adjacency matrix $\boldsymbol{A}$ according to sliding window models. Then we integrate the dynamic adjacency matrix $\boldsymbol{A}$ into **PDFC** for dynamic clustering.

Algorithm 2: PDFC

```
Input: The target node set $V = \{v_1, ..., v_N\}$, the adjacency matrix $\boldsymbol{A}$, node embeddings $Z \in \mathbb{R}^{N \times k}$, the
       group-membership vector $f^{(s)} \in \{0,1\}^N, s \in [h]$, the number of clusters $k$.
Output: Clusters $C = \{C_1, C_2, ..., C_k\}$.
1  Compute the Laplacian matrix $L = D - \boldsymbol{A}$ by the degree matrix $D$ ;
2  Map $Z$ into $k$-dimensional space $Z_1 \in \mathbb{R}^{N \times k}$;
3  Perform the Cholesky decomposition on $Z_1^T D Z_1 = Q_1 Q_1^T$ and obtain $\hat{Z}_1 = D^{\frac{1}{2}} Z_1 (Q_1^{-1})^T$;
4  Obtain a matrix $F \in \mathbb{R}^{N \times (h-1)}$ that has the vectors $f^{(s)} - \frac{|V_s|}{N} \cdot \mathbf{1}_N$;
5  Compute the SVD of $F^T$ to obtain the matrix $\tilde{Z} \in \mathbb{R}^{N \times (N-h+1)}$;
6  Aggregate neighbor information $Z_2 = \sigma(\boldsymbol{A}\sigma(\boldsymbol{A}\tilde{Z}\mathbf{W}^{(0)})\mathbf{W}^{(1)})$;
7  Perform the Cholesky decomposition on $Z_2^T D Z_2 = Q_2 Q_2^T$ and obtain $\hat{Z}_2 = D^{\frac{1}{2}} Z_2 (Q_2^{-1})^T$;
8  Concatenate $\hat{Z}_1$ and $\hat{Z}_2$, $\mathcal{Z} = [\hat{Z}_1 || \hat{Z}_2]$;
9  Apply $k$-means on $\mathcal{Z}E$ to partition nodes into $k$ disjoint clusters ;
10 return $C$;
```

5.1 Sliding Window Models

The sliding window model consists of two parameters, called *window* and *stride*, and their descriptions are introduced below.

Window. The *window* $\mathcal{W}$ contains the latest interaction information of nodes in the adjacency matrix $\boldsymbol{A}$, and $|\mathcal{W}|$ is the size of the *window*, which determines the size of the adjacency matrix $\boldsymbol{A}$. There are two ways: a *count-based window* (select the number of nodes of the adjacency matrix in the *window*) and a *time-based window* (denote a range for the *window* duration) for us to choose the type of sliding window.

Stride. The *stride* $\mathcal{S}$ denotes the range of *window* sliding when updating the adjacency matrix $\boldsymbol{A}$ to ensure that the clustering results change over time. The size of the *stride* is denoted as $|\mathcal{S}|$, representing the range of new data added for the *window*.

Taking $\mathcal{G}$ of Figure 1 as an example, we set the type of *window* $\mathcal{W}$ as the *time-based window*, $\mathcal{W}$ duration range is three months, and the *stride* is one month. Whenever *window* $\mathcal{W}$ slides forward by one month, the $\mathcal{W}$ will remove nodes, add new nodes, and update several nodes in the adjacency matrix $\boldsymbol{A}$ during this period to derive a new adjacency matrix. Then new adjacency matrix is processed by clustering algorithms to generate new clustering results.

5.2 Fair Clustering via Sliding Window Models

Let $\{1, 2, ..., \tau\}$ be a sequence of time points, and the interval between any two-time points represents the *stride* $\mathcal{S}$ of the sliding window. $G = \{\mathbf{G}_1, \mathbf{G}_2, ..., \mathbf{G}_\tau\}$ of $\mathcal{G}$ is a set of states evolving over $\{1, 2, ..., \tau\}$, where the state of $\mathbf{G}_t$ is derived from $\mathbf{G}_{t-1}$ and will subsequently evolve into $\mathbf{G}_{t+1}$, $\mathbf{G}_{t-1} \bigcap \mathbf{G}_t \neq \emptyset$, and $\mathbf{G}_t \bigcap \mathbf{G}_{t+1} \neq \emptyset$. The adjacency matrix $\boldsymbol{A} = \{\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_\tau\}$ of G on $\{1, 2, ..., \tau\}$ denotes the relationship weights among nodes when the sliding window slides at different time points.

In the static state, fair clustering aims to obtain k clusters with fairness from the node set V of G, denoted as $\{C_l\}_{l=1}^k$, where $V = \bigcup\{C_l\}_{l=1}^k$, $C_l \bigcap C_j = \emptyset$ for $l \neq j$, C_l is the l-th cluster, and k is the number of clusters. However, when $G = \{\mathbf{G}_1, \mathbf{G}_2, ..., \mathbf{G}_\tau\}$ changes with the time point sequence $\{1, 2, ..., \tau\}$, the fair clustering algorithm needs to dynamically capture the adjacency matrix of G at each time point and obtain the corresponding clusters, denoted as $\{C_{lt}\}_{l=1}^{k_t}$, where C_{lt} is the l-th cluster at time point t.

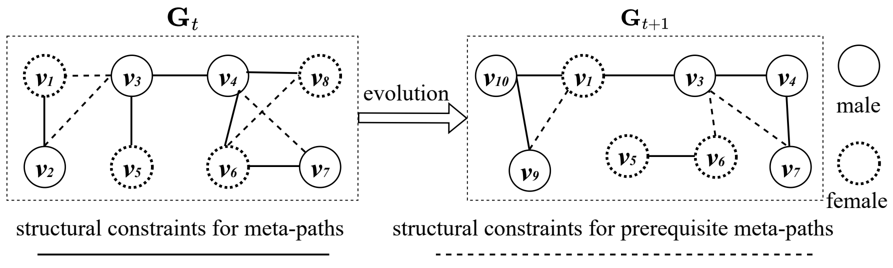


Fig. 4. An example of a dynamic fair clustering process.

As shown in Figure 4, we take the evolution process of graph G at time points t and t+1 as an example and analyze the state from $\mathbf{G}_t$ to $\mathbf{G}_{t+1}$. At time point t, we divide graph $\mathbf{G}_t$ by a fair clustering algorithm to form two clusters $C_{1t} = \{v_1, v_2, v_3, v_5\}$ and $C_{2t} = \{v_4, v_6, v_7, v_8\}$, C_{1t} and C_{2t} satisfy the structural and balance constraints. $\mathbf{G}_t$ evolves into $\mathbf{G}_{t+1}$ when the sliding window slides from time point t to t+1, and we find that the information and structure of $\mathbf{G}_{t+1}$ have changed compared to $\mathbf{G}_t$. For example, v_2 and v_8 are deleted, v_9 and v_{10} are added, and the connections between other nodes are updated. At time point t+1, we obtain clusters $C_{1t+1} = \{v_1, v_9, v_{10}\}$ and $C_{2t+1} = \{v_3, v_4, v_5, v_6, v_7\}$ in graph $\mathbf{G}_{t+1}$, and they satisfy the structural and balance constraints.

In practice, the changes from $\mathbf{G}_t$ to $\mathbf{G}_{t+1}$ are updates of their corresponding adjacency matrices $\mathcal{A}_t$ and $\mathcal{A}_{t+1}$. We let $\Delta\mathcal{A}_t$ represent the structural information in the sliding window at time points t and t+1, which is the common part of $\mathcal{A}_t$ and $\mathcal{A}_{t+1}$. Inspired by the common embedding matrix learning in [66], we decompose $\Delta\mathcal{A}_t$ into two parts: one part is the adjacency matrix (denoted as $\mathcal{A}_t^r$) formed by the nodes that will be removed at the time point t+2, and the other represents the adjacency matrix (denoted as $\mathcal{A}_t^i$) formed by the inherent structure of graphs $\mathbf{G}_t$ and $\mathbf{G}_{t+1}$. In addition, we let $\tilde{\mathcal{A}}_{t+1}$ be the adjacency matrix of nodes that will add when $\mathbf{G}_t$ evolves into $\mathbf{G}_{t+1}$ at the time point t+1. Therefore, we present an updated strategy of the adjacency matrix with time point sequence $\{1, ..., \tau\}$ changes, as shown in Algorithm 3.

Algorithm 3: Adjacency matrix update

```
Input: The initialized sliding window $\mathcal{W}_t$, the stride $\mathcal{S}$, and the adjacency matrix $\mathcal{A}_t$, the
       hyper-parameters $\lambda$ and $\mu$;
Output: The updated adjacency matrix $\mathcal{A}_{t+1}$.
1 for each stride $\mathcal{S}_i \in \{1, ..., \tau\}$ do
2     $\mathcal{W}_{t+1} \xleftarrow{\mathcal{S}_i} \mathcal{W}_t$;
3     $\mathcal{A}_{t+1} \xleftarrow{\mathcal{S}_i} \mathcal{A}_t$;
4     $\Delta\mathcal{A}_t = \mathcal{A}_t \cap \mathcal{A}_{t+1}$;
5     $\mathcal{A}_t^r, \mathcal{A}_t^i \Leftarrow Decompose(\Delta\mathcal{A}_t, |\mathcal{S}|)$;
6     $\tilde{\mathcal{A}}_{t+1} = \mathcal{A}_{t+1} \setminus \Delta\mathcal{A}_t$;
      /* Update the adjacency matrix.                                  */
7     $\mathcal{A}_{t+1} = \lambda\mathcal{A}_t^r + \mathcal{A}_t^i + \mu\tilde{\mathcal{A}}_{t+1}$ ;
8 return $\mathcal{A}_{t+1}$;
```

Note that both $\mathcal{A}_t$ and $\mathcal{A}_{t+1}$, as well as the partial adjacency matrix $\Delta\mathcal{A}_t$, $\mathcal{A}_t^r$, etc., have a dimension $\mathbb{R}^{N\times N}$. We set the hyper-parameters $\lambda \in (0, 1]$ and $\mu \in [1, 2)$, which are attached to the graph structure formed by meta-paths and prerequisite meta-paths, and control the importance of the corresponding nodes when updating the adjacency matrix $\mathcal{A}_{t+1}$.

6 EXPERIMENTS

6.1 Setup

Datasets. We conduct experiments with three real-world datasets: `MOOCCube`, `DBLP`, and `Movielens`. The detailed statistics of the three datasets are shown in Table 2.

- `MOOCCube`. `MOOCCube` is an online education dataset, and it was collected and published on *XuetangX*[2] by Yu et al. [67]. We get four entities (courses (C), videos (**V**), users (U), and knowledge concepts (K)) and their relationships related to computer science from January 1, 2018, to May 31, 2019, as our experimental data and name it as `MOOCCube_cs`. The sensitive attributes of MOOCCube are contained in its extended dataset MOOCCubeX [68], and we match the corresponding user information.
- `DBLP`. `DBLP` is a citation network dataset, and it was collected and published on *AMiner*[3] by Tang et al. [57]. The citation relationship among papers indicates their sequential dependency, which can be considered one paper is a "prerequisite" for another. Therefore, we uniformly express the citation relationship as a prerequisite relationship in this work. We extracted three entities (authors (A), papers (P), and venues (**V**)) and their relationships from journals or conferences in three fields of *database*, *data mining*, and *information retrieval* from 2009 to 2018. Since `DBLP` does not involve sensitive attributes such as gender or race, we investigated several studies [9, 45, 63] on the gender ratio of authors who published papers in computer science and found that the proportion of female authors has been 15%-22% for the past two decades. Therefore, we simulate that the proportion of female authors in our extracted dataset is 20%. We set the parameter 4:1 in the random function to indicate the ratio of male authors (denoted as 1) to female authors (denoted as 0), then randomly allocate 1 and 0 to the authors in `DBLP`.
- `Movielens`. It is a rating dataset about movies collected by *GroupLens*. We get two entities (users (U) and movies (M)) and their relations in `Movielens` [1]. The potential relationship between movies is their dependence, which indicates the "prerequisite" relationship of our work description. We require dependency values between movies in [1] greater than 0.5. Inspired by age as a sensitive attribute in [64], we divide users into two groups based on whether the age of user reviews is greater than 4.

We selected the data of `MOOCCube_cs` from January 1, 2018, to March 31, 2018, the `DBLP` data from January 2009 to December 2011, and the entire `MOOCCube_cs` as the offline data of **PDFC**. We set meta-paths and prerequisite meta-paths on `DBLP` as {APA, A**V**PVA} and {APPA, A**V**PPVA}, and the two kinds of paths on `MOOCCube_cs` are described as EXAMPLE 1 and EXAMPLE 2. We set meta-paths and prerequisite meta-paths in `Movielens` as UMU and UMMU, respectively. We implement dynamic **PDFC** by the sliding window model on the remaining data of `MOOCCube_cs`.

Baselines. We evaluate the performance of our **PDFC** by comparing it with several state-of-the-art fair clustering algorithms.

- Fair *k*-medians (**FKMEDI**) [7]. This fair clustering is a scalable algorithm for computing (*r, b*)-fairlet decompositions with a running time be nearly linear by any integer values of *r, b*. Note that it only implements comparison under balance constraints.
- Fair Spectral Clustering (**FSC**) [37]. This fair algorithm tries to incorporate the fairness notion in [20] into spectral clustering for partitioning graph data. We implement unnormalized and normalized spectral clustering algorithms with fairness constraints, denoted as **UFSC** and **NFSC**, respectively.

[2] http://www.xuetangx.com

[3] https://www.aminer.cn/citation

Table 2. Statistics of datasets.

Datasets	Entities	Count	Relations	Count
MOOCCube_cs	C	150	C-V	7745
	V	6730	V-K	26224
	K	4884	C-K	10692
	U	106834	U-C	250931
			U-V	3700129
DBLP	A	24964	A-P	48534
	P	24916	P-V	24916
	V	20	A-V	36091
Movielens	U	943	U-M	100000
	M	1682	M-M	6183

- Variational Fair Clustering (**VFC**) [72]. It is a general and bound-optimization framework of fair clustering, which integrates a Kullback-Leibler fairness term with clustering objectives. We implement the *Ncut* objective of **VFC**, denoted as **NVFC**.

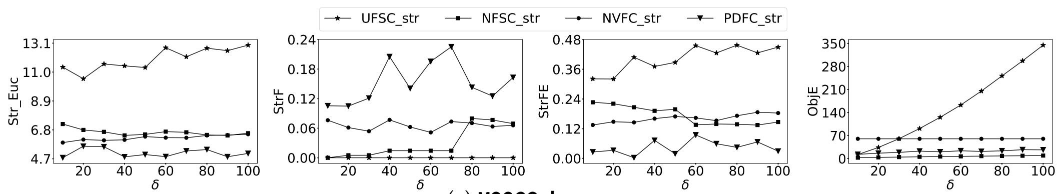
Implementations. We set the hyper-parameters of **PDFC** and its variants to our experiment. For α and β of connections based on meta-paths and prerequisite meta-paths, we set their values are 0.5, 0.25, and 0.25 in MOOCCube_cs, 0.6 and 0.4 in DBLP, and 1 in Movielens. The number of attention heads L is 4 in attention-based embedding learning. We select Adam [34, 59] as the optimizer of **PDFC** and its variants with the epoch being 50, the learning rate being 0.05, and weight decay being 0.01. We implement **PDFC** and its variants with PyTorch 1.12 in Python 3.8. All experiments run on the PC with 11th Gen Intel(R) Core(Tm) i5-11300H@3.10GHz.
Evaluation Metrics. We design several metrics to evaluate our method and baselines according to Equations (1) and (2).

- **BalF**. We utilize min *balance*(C_l) as the overall *balance fairness* (BalF) [72] of clustering results according to Equation (2).
- **BalFE**. Inspired by defining the average of Equation (2) over all clusters as "Balance" [37], we denote the difference between the average BalF and the balance of sensitive attributes in the original data as the *balance fairness error* (BalFE).
- **Bal_Euc**. We compute the Euclidean distance between the ratio of sensitive attributes in clusters and the ratio of the original data to represent the closeness of overall clustering, called Bal_Euc.
- **StrF, StrFE, Str_Euc**. Similar to balance constraints, we also define the *structural fairness* (StrF), *structural fairness error* (StrFE), and Str_Euc (the Euclidean distance between the structure of meta-paths and prerequisite meta-paths in clusters and the original dataset to denote the closeness of the overall clustering results) according to Equation (1).

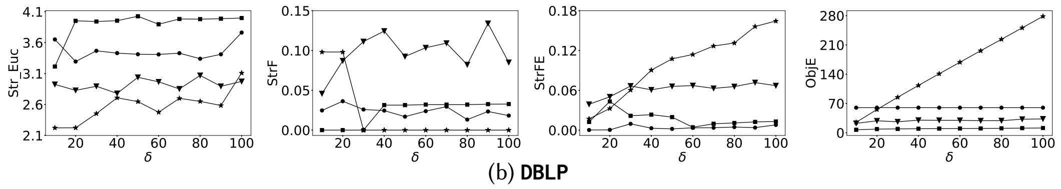
We take the value as the error between k clusters when the objective of methods converge, called ObjE, which is the optimal loss function value in Equation (25). We also record the *computational time* of methods as a metric. Note that except for the metrics StrF and BalF, the smaller values of others are better.

6.2 Effectiveness

In this section, we investigate the effect of the number of neighbors (δ) of the adjacency matrix $\boldsymbol{A}$ and the parameter (η) of meta-paths and prerequisite meta-paths on different methods. Under structural, balance, and two constraints, we denote methods as **METHOD_str**, **METHOD_bal**,



(a) `MOOCCube_cs`



(b) `DBLP`

Fig. 5. The evaluation metrics of methods change with increasing δ under structural constraints.

Table 3. Fairness comparisons of PDFC_str with other methods under structural constraints.

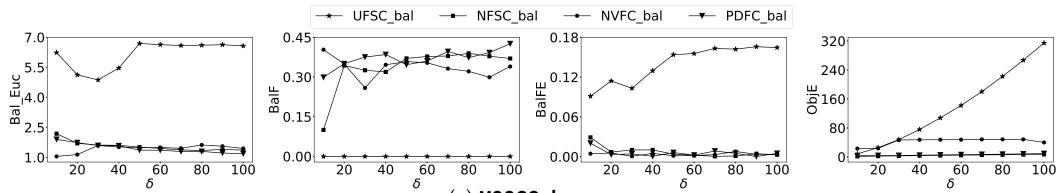
Methods	`MOOCCube_cs`				`DBLP`				`Movielens`			
	Str_Euc	**StrF/StrFE**	**ObjE**	**Time(s)**	**Str_Euc**	**StrF/StrFE**	**ObjE**	**Time(s)**	**Str_Euc**	**StrF/StrFE**	**ObjE**	**Time(s)**
UFSC_str	10.5046	0.0/0.3214	33.6848	**11.5669**	**2.2226**	0.0981/0.0171	25.5069	**27.111**	10.6225	0.0/0.0754	40.6206	**0.5255**
NFSC_str	6.4913	0.0751/0.1428	**8.4768**	52.6022	3.2139	0.0/0.0124	**7.1465**	126.9949	3.9492	0.0/0.006	11.5141	1.0167
NVFC_str	5.8706	0.0762/0.1350	60.0	95.6576	3.2943	0.0363/**0.0006**	60.0	158.7442	**1.3076**	**0.0231**/0.0096	**2.0**	28.6134
PDFC_str	**4.8314**	**0.2011**/**0.0710**	22.6061	67.5309	2.7789	**0.1244**/0.0613	30.0963	94.6937	3.0156	0.0011/**0.0043**	40.2717	10.7954

and **METHOD**, respectively. Then we obtain the average results of running ten times for methods and analyze their effectiveness.

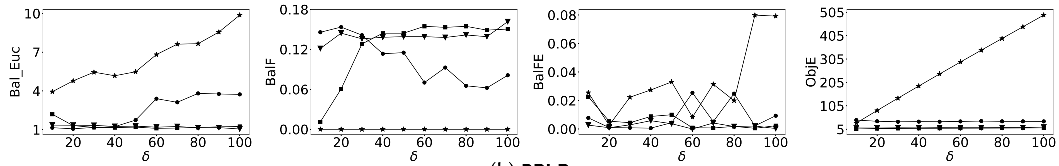
6.2.1 Analysis of Structural Fairness. Under structural constraints, we set η as 1 in Equation (13), the number of clusters as 30, and eliminate sensitive attributes to analyze structural fairness.

As shown in Figure 5, we exhibit the change of evaluation metrics of **UFSC_str**, **NFSC_str**, **NVFC_str**, and **PDFC_str** methods with the increase of δ. As δ increases, we find that Str_Euc of **UFSC_str** and **NVFC_str** gradually increase on two datasets. However, the Str_Euc values of **NFSC_str** and **PDFC_str** gradually decrease on `MOOCCube_cs`, while the Str_Euc of **NFSC_str** increases and the Str_Euc of **PDFC_str** oscillates on `DBLP`. On `MOOCCube_cs`, we can see that the Str_Euc of **PDFC_str** outperforms other methods in all δ, and **PDFC_str** gets the best Str_Euc when $\delta = 40$. The Str_Euc of **PDFC_str** is not optimal among all methods on `DBLP`, and it reaches the optimal when $\delta = 40$. For the `StrF`, **PDFC_str** increases with δ, while the other methods perform poorly on two datasets. We find that the `StrF` of **PDFC_str** outperforms other methods on two datasets, and **PDFC_str** achieves the optimal value on `MOOCCube_cs` and `DBLP` when $\delta = 70$ and $\delta = 90$, respectively. The `StrFE` of **NFSC_str** gradually decreases as δ increases, while the `StrFE` of **UFSC_str** and **NVFC_str** increases. Interestingly, the `StrFE` of **PDFC_str** oscillating changes on `MOOCCube_cs` and is better than other methods, while its `StrFE` gradually increases on `DBLP` and is not optimal in all methods. The `StrFE` of **PDFC_str** is optimal on `MOOCCube_cs` and `DBLP` when $\delta = 30$ and $\delta = 40$, respectively. Except for **NVFC_str**, the `ObjE` of the other methods increases with δ on two datasets, especially the `ObjE` of **UFSC_str** has the fastest growth rate. We find that the `ObjE` of **PDFC_str** is only weaker than **NFSC_str** on two datasets.

We take evaluation metrics `StrF` and Str_Euc as the key metrics to determine the optimal δ of all methods. On `MOOCCube_cs`, we select the δ of **UFSC_str**, **NFSC_str**, **NVFC_str**, and **PDFC_str** are 20, 90, 10, and 40, respectively. On `DBLP`, we select the δ of **UFSC_str**, **NFSC_str**, **NVFC_str**, and **PDFC_str** are 10, 10, 20, and 40, respectively. On `Movielens`, we also select the optimal δ of each method. Then we report the best metrics of methods for comparison, as shown in Table 3. On `MOOCCube_cs`, although the `ObjE` of **PDFC_str** is weaker than **NFSC_str** and elapsed time of



(a) MOOCCube_cs



(b) DBLP

Fig. 6. The evaluation metrics of methods change with increasing δ under balance constraints.

Table 4. Fairness comparisons of PDFC_bal with other methods under balance constraints.

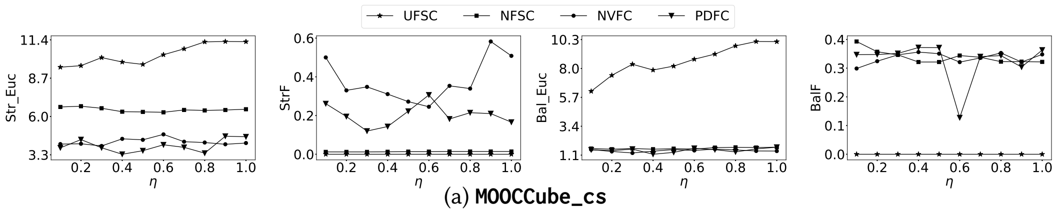
Methods	MOOCCube_cs				DBLP				Movielens			
	Bal_Euc	BalF/BalFE	ObjE	Time(s)	Bal_Euc	BalF/BalFE	ObjE	Time(s)	Bal_Euc	BalF/BalFE	ObjE	Time(s)
FKMEDI	4.9348	**0.4397**/0.3293	983.0837	**9.5735**	12.0396	**0.2176**/0.6840	3267.8235	30.6973	7.6002	**0.2877**/0.5036	9065.1912	2.5057
UFSC_bal	4.8716	0.0/0.1035	47.9633	11.6157	3.9372	0.0/0.0255	31.4785	**14.9498**	3.1597	0.0769/**0.005**	24.3162	**0.2244**
NFSC_bal	1.3433	0.3901/**0.0011**	**6.0789**	50.5786	1.0997	0.1546/0.0009	**9.2289**	118.7103	2.7892	0.0833/0.006	14.1697	0.712
NVFC_bal	**1.0276**	0.4092/0.0022	23.7719	240.7116	1.0655	0.1535/0.0010	37.8243	202.7759	3.3828	0.128/0.0384	N/A	28.3343
PDFC_bal	1.1831	0.4245/0.0052	9.3919	30.1671	**1.0512**	0.1617/**0.0003**	14.4626	37.3380	**2.445**	0.1763/0.0159	**12.8835**	2.8213

PDFC_str is weaker than **UFSC_str**, the Str_Euc, StrF, and StrFE of **PDFC_str** are better than other methods, especially StrF which indicates the structural fairness of clustering results. On DBLP, we find that although the metrics of **PDFC_str** are not much better than that of **UFSC_str**, its StrF is the best. Compared with the other two methods, the Str_Euc, StrF, and time of **PDFC_str** have advantages. Since **PDFC_str** contains more matrix operations, its elapsed time is higher than that of **UFSC_str** (there are fewer matrix operations in the four methods). Unexpectedly, we find the **NVFC_str** is optimal in the Str_Euc, StrF, and ObjE metrics on Movielens. However, its elapsed time is the highest. The StrFE of our **PDFC_str** is optimal, and its Str_Euc, StrF, and ObjE metrics are better than the other two methods.

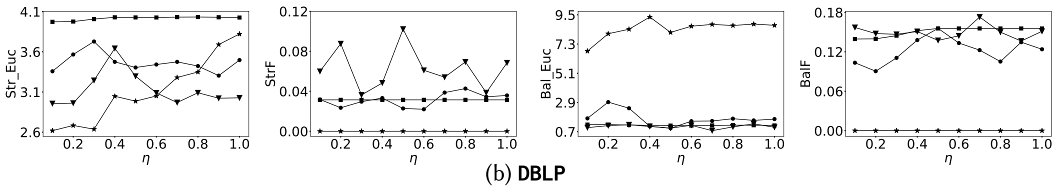
Insight 1: In DBLP, Str_Euc and elapsed time of **UFSC_str** are better than **PDFC_str**. One reason is that the matrix calculation of **UFSC_str** is less. The other is that clustering results of **UFSC_str** are disequilibrium, and a large cluster appears. In Movielens, the **NVFC_str** obtains the optimal solution by constraining the range of solutions, but its cost is a long-elapsed time.

6.2.2 Fairness Analysis of Sensitive Attributes. We set η to be 0 in Equation (13) to ensure that the adjacency matrix $\boldsymbol{A}$ does not contain the structural information of prerequisite meta-paths. The number of clusters is still 30, and we evaluate the effectiveness of methods under balance constraints.

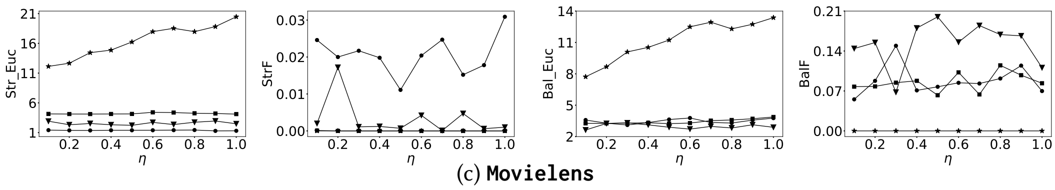
As shown in Figure 6, we display the change of evaluation metrics of the **UFSC_bal**, **NFSC_bal**, **NVFC_bal**, and **PDFC_bal** methods as δ increases. On two datasets, we can see that the Bal_Euc of **NFSC_bal** and **PDFC_bal** decreased with the increase of δ, but that of **UFSC_bal** and **NVFC_bal** increased gradually. On two datasets, the Bal_Euc of **PDFC_bal** is optimal when $\delta = 100$. The BalF of **UFSC_bal** remains 0 as δ changes on both datasets, while the BalF of the **NVFC_bal** decreases and the other two methods increase. The BalF of **PDFC_bal** outperforms other methods on MOOCCube_cs, while it is weaker than the BalF of **NFSC_bal** when $\delta \in [40, 90]$ on DBLP. The BalF of **PDFC_bal** is optimal when $\delta = 100$. Except for the BalFE of **UFSC_bal** increases with δ, BalFE of other methods decreases gradually, and the overall change is small. However, the BalFE of **NVFC_bal** oscillates slightly on DBLP. The BalFE of **PDFC_bal** outperforms other methods when



(a) **MOOCCube_cs**

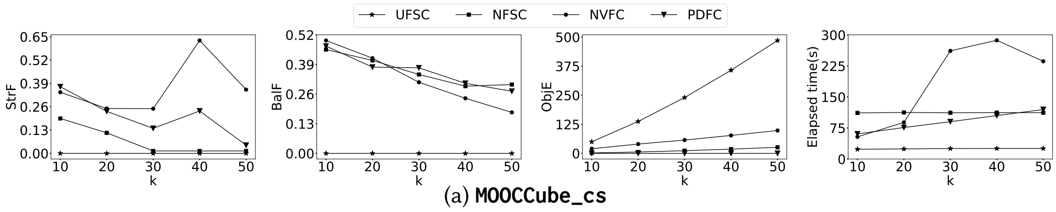


(b) **DBLP**

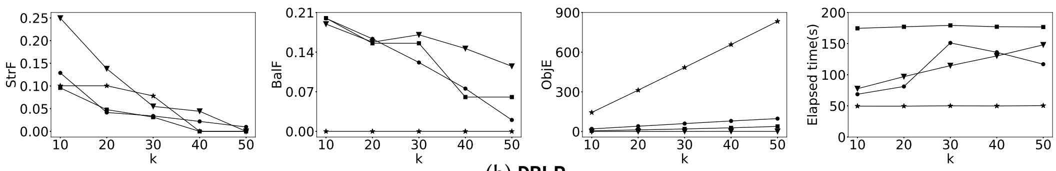


(c) **Movielens**

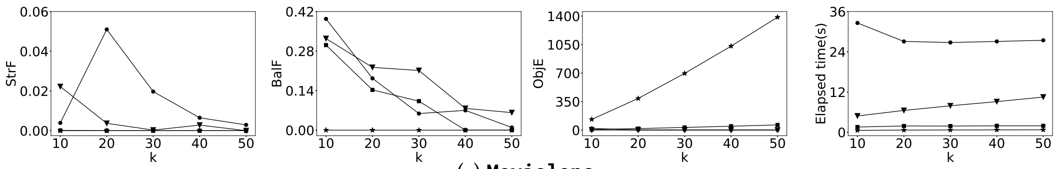
Fig. 7. The metric of methods varies with increasing η.



(a) **MOOCCube_cs**



(b) **DBLP**



(c) **Movielens**

Fig. 8. The metric of methods varies with increasing k.

δ >= 60 on two datasets, and the BalFE of **PDFC_bal** BalFE is optimal when δ = 100. The ObjE of the four methods increases gradually with the increase of δ, while the ObjE of **UFSC_bal** increases fastest with δ. The ObjE of **UFSC_bal** is still optimal due to its fewer matrix calculation, similar to structural constraints.

We take BalF and Bal_Euc as the key metrics to obtain the optimal δ of methods. On MOOCCube_cs, we select the δ of **UFSC_bal**, **NFSC_bal**, and **NVFC_bal** are 30, 80, and 10, respectively. On DBLP, we select the δ of **UFSC_bal**, **NFSC_bal**, and **NVFC_bal** are 10, 60, and 20, respectively. As for the δ of **PDFC_bal** is 100 on two datasets. On Movielens, we also select the optimal δ of each method. In addition, we add the **FKmedi** method for comparison. Then we report their best metrics for

Table 5. Fairness comparisons of PDFC with other methods on two constraints.

Metrics	`MOOCCube_cs`				`DBLP`				`Movielens`			
	UFSC	NFSC	NVFC	PDFC	UFSC	NFSC	NVFC	PDFC	UFSC	NFSC	NVFC	PDFC
Str_Euc	9.4735	6.2954	3.9229	**3.3381**	**2.6186**	4.0282	3.4046	2.9687	12.1295	4.2493	**1.3617**	2.2253
StrF	0.0	0.0135	**0.3481**	0.1432	0.0468	0.0314	0.0228	**0.0541**	0.0	0.0	**0.0217**	0.0007
StrFE	0.3053	0.1926	0.1613	**0.0104**	0.0459	0.0215	**0.0058**	0.0563	0.0738	0.0216	0.0175	**0.0084**
Bal_Euc	6.1871	1.5857	1.2722	**1.1703**	6.7707	1.1595	0.9589	**0.7793**	7.7193	3.5872	3.1073	**2.8952**
BalF	0.0	0.3439	0.3463	**0.3718**	0.0	0.1556	0.1557	**0.173**	0.0	0.1149	0.1492	**0.1999**
BalFE	0.1561	0.0113	**0.0003**	0.0013	0.0455	0.0043	**0.0007**	0.0067	0.167	0.026	0.024	**0.0236**
ObjE	240.2098	11.5153	58.4773	**0.1079**	484.2649	19.443	60.0	**0.5428**	696.6013	32.446	2.0	**0.0129**
Time(s)	**21.9898**	103.2949	276.3581	91.1281	**44.5793**	174.3751	172.6359	117.4046	**0.6383**	1.8151	28.3476	7.833

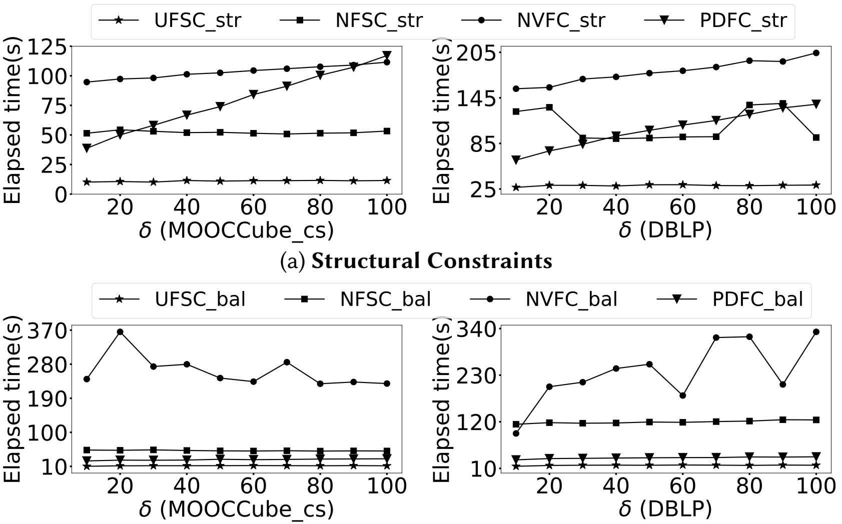
comparison, as shown in Table 4. Compared with other methods on `MOOCCube_cs`, we find that none of the evaluation metrics of **PDFC_bal** is optimal, but it still has advantages. For example, although Bal_Euc of **NVFC_bal** is superior than **PDFC_bal**, its elapsed time is eight times that of **PDFC_bal**. The BalF of **FKMEDI** is superior than **PDFC_bal**, but its ObjE is nearly 105 times that of **PDFC_bal**. The Bal_Euc and BalF of the **PDFC_bal** method are superior to **UFSC_bal** and **NFSC_bal**. On `DBLP`, the Bal_Euc and BalFE of **PDFC_bal** are best. The Bal_Euc and ObjE of **PDFC_bal** are optimal on `Movielens`. The situation of other metrics on these two datasets is similar to that of `MOOCCube_cs`.

Insight 2: On `MOOCCube_cs`, although each evaluation metric of the **PDFC_bal** method is not optimal, it has advantages when all evaluation metrics are considered comprehensively.

6.2.3 Fairness Analysis under Two Constraints. In this section, we set δ as 50, the number of clusters (k) from 10 to 50, and let η range from 0.1 to 1 to investigate the influence of hyper-parameter η and k on methods by considering two constraints.

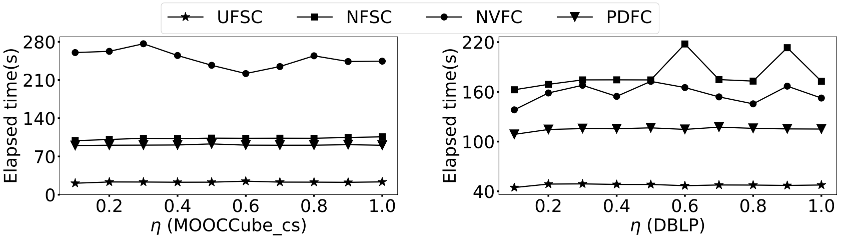
We set the number of clusters as 30 and investigate the influence of hyper-parameter η on methods. As shown in Figure 7, we display the change of evaluation metrics Str_Euc, StrF, Bal_Euc, and BalF of the **UFSC**, **NFSC**, **NVFC**, and **PDFC** methods with η. On `MOOCCube_cs`, we find that the Str_Euc of **PDFC** is better than other methods, while only its StrF is weaker than **NVFC**. The Bal_Euc and BalF of **PDFC** are slightly better than other methods. We find that the Str_Euc, Bal_Euc, and BalF of **PDFC** are optimal when $\eta = 0.4$, while its StrF is best when $\eta = 0.6$. We select 0.4 as the optimal η of **PDFC**. Compared with other methods on `DBLP`, we can see that the Str_Euc and StrF of **PDFC** are better than other methods, but its Str_Euc is weaker than **UFSC** when $\eta <= 0.6$ on `DBLP`. Although the BalF of **PDFC** is not as stable as the **NFSC** method, its Bal_Euc and BalF are better than other methods. We find that the Bal_Euc and BalF of **PDFC** are optimal when $\eta = 0.7$, the Str_Euc value is best when $\eta = 0.1$, and the StrF value is best when $\eta = 0.5$. We select 0.7 as the optimal η of **PDFC**. On `Movielens`, we find that the Str_Euc and StrF of **NVFC** are optimal, while the Bal_Euc and BalF of **PDFC** are optimal. The Str_Euc and Bal_Euc of **UFSC** increase with η, while its StrF and BalF are always 0. We set the η value of **UFSC** as 0.1. The BalF of **NVFC** and **PDFC** are optimal when $\eta = 0.3$ and $\eta = 0.5$, and we set their η value as 0.3 and 0.5, respectively. The BalF of **NFSC** is optimal when $\eta = 0.8$, and we set its η value as 0.8.

We set η of all methods as the optimal value in Figure 7 and investigate the influence of hyper-parameter k on them. As shown in Figure 8, we display the change of evaluation metrics StrF, BalF, ObjE, and elapsed time of the **UFSC**, **NFSC**, **NVFC**, and **PDFC** methods with k. On three datasets, the StrF and BalF of the four methods decrease with the increase of k. The elapsed time of four methods increases gradually with k, the ObjE of **UFSC**, **NFSC**, and **NVFC** is also increasing, while the ObjE of **PDFC** remains unchanged. However, the StrF and BalF of **UFSC** are always 0 with the increase of k. Interestingly, as the k increases, the ObjE of **PDFC** is the best in all methods and very small. Except for the StrF on `MOOCCube_cs` and `Movielens`, other metrics of **PDFC** have advantages over other methods on three datasets.



(a) **Structural Constraints**

(b) **Balance Constraints**



(c) **Two Constraints**

Fig. 9. The elapsed time of methods.

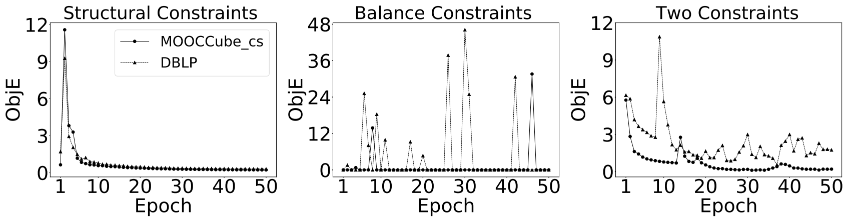


Fig. 10. Convergence analysis of PDFC and its variants.

Without losing generality, we set the number of clusters (k) as 30 and η of methods as the optimal value in Figure 7. We obtain the values of other evaluation metrics of methods according to the best results of Str_Euc, StrF, Bal_Euc, and BalF metrics in Figure 7, as shown in Table 5. On three datasets, we find that although **PDFC** is weaker than one of the methods in individual evaluation metrics, more than half of its evaluation metrics are optimal. Therefore, these experimental results prove that we concatenate node embeddings of structural and balance constraints that are effective. ***Insight 3***: After we concatenate node embeddings of structural and balance constraints, the ObjE improves at least 87, 26, and 988 times on `MOOCCube_cs`, `DBLP`, and `Movielens`. One reason is we concatenate node embeddings of structural and sensitive attributes, which can help us alleviate the trouble when optimizing the bi-objective. Another is that **PDFC** can learn rich information via meta-paths and prerequisite meta-paths to ensure a close connection among nodes and provide a basis for fair clustering.

6.3 Efficiency

6.3.1 Elapsed Time. As shown in Figure 9, we exhibit the elapsed time of methods varies with δ and η on MOOCCube_cs and DBLP. Figure 9(a) shows the elapsed time change with the increase of δ under structural constraints. The elapsed time of **PDFC_str** and **NVFC_str** gradually increases as δ increases on the two datasets, and **UFSC_str** does not change much. Strangely, the elapsed time of **NFSC_str** is unchanged on MOOCCube_cs, and it changes dynamically with δ on DBLP. **PDFC_str** has a long-elapsed time because it contains more matrix operations, **NVFC_str** has a small threshold (Figures 9(b) and (c) also show the high time cost.), and **UFSC_str** and **NFSC_str** are only affected by the dimensions of the adjacency matrix. In Figure 9(b), except for the elapsed time of **NVFC_bal** changes with δ, other methods have little overall change. We can see that the elapsed time of **NVFC_bal** decreases with the increase of δ on MOOCCube_cs, but the opposite is true on DBLP. Since the **PDFC_bal** has fewer matrix operations, its elapsed time is reduced and even lower than that of **NFSC_bal**. In Figure 9(c), we find that the elapsed time of **PDFC**, **UFSC**, and **NFSC** under two constraints is roughly the sum of the elapsed time in Figure 9(a) and (b). However, **NVFC** is affected by the set threshold, and its elapsed time changes dynamically on different η, especially on DBLP. Compared with the **NFSC** and **NVFC** methods, the elapsed time of **PDFC** is acceptable under two constraints.

6.3.2 Convergence Analysis. Figure 10 shows the convergence of **PDFC** on MOOCCube_cs and DBLP with the increase of iteration. We reduce the ObjE value by 100 and 100000 times under structural and balance constraints, respectively. We observe that the convergence of **PDFC** is the best under structural constraints, and the overall convergence is still achieved under the balance constraint, although there are several outliers. Under the two constraints, we find that the convergence of **PDFC** on MOOCCube_cs is better than that on DBLP. In addition, the convergence of **PDFC** experiences a slight oscillation after 25 iterations on DBLP. The reason is that there is a large scale of entities in DBLP, but their interaction relationships are sparse.

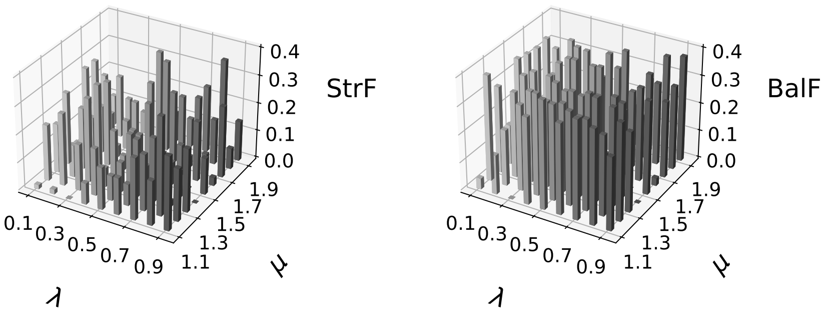
6.4 Dynamic Clustering Analysis

On MOOCCube_cs, we set the sliding window size to be 3 and the sliding stride to be 1, where its unit of measure is the month.

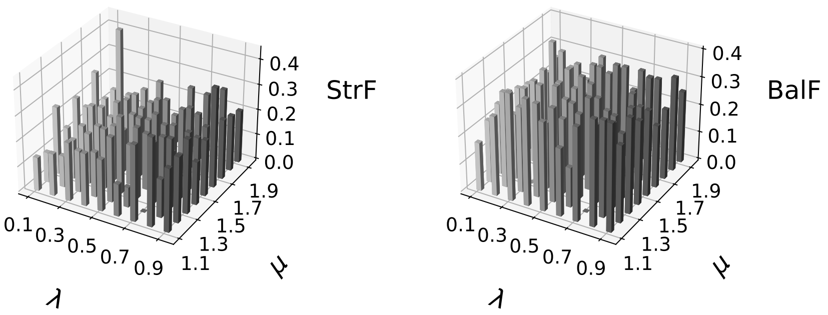
6.4.1 Parameter Sensitivity. The update of the adjacency matrix $\boldsymbol{A}$ contains two parameters: λ and μ, where λ controls the importance of the graph structure among nodes that will be deleted at the next time, and μ represents the importance of the graph structure among nodes added at the current time.

We observed how λ and μ affect the performance of **PDFC** by varying λ from 0.1 to 0.9 and μ from 1.1 to 1.9 for MOOCCube_cs, as shown in Figure 11. At time t and $t+1$, we find that BalF is overall better than StrF in the clustering results. The reason is that we update the adjacency matrix through parameters λ and μ leading to the structure change of the graph, which has a tremendous influence on StrF and a small impact on BalF. These values indicate that the StrF and BalF are better when $\mu \in [1.5, 1.9]$. Therefore, μ increases the importance of graph structure among newly added nodes. We can see that the overall values of StrF are better while the BalF is slightly poor when $\lambda \in [0.5, 0.9]$. However, the values of these two metrics are just the opposite when $\lambda \in [0.1, 0.5]$. The reason is that the correlation of the graph structure between nodes may be destroyed when λ is small, resulting in weak connectivity among nodes and poor results.

6.4.2 Time Change under $\mathcal{W}$. As shown in Figure 12, the elapsed time of **PDFC** as the sliding window slides with the stride. Because the number of captured nodes is different when the sliding window strides, we find that the elapsed time of **PDFC** is also dynamic. The reason is that the



(a) The clustering results of the sliding window at time t.



(b) The clustering results of the sliding window at time t+1.

Fig. 11. Effect of hyperparameters λ and μ on **PDFC** performance when the adjacency matrix is updated.

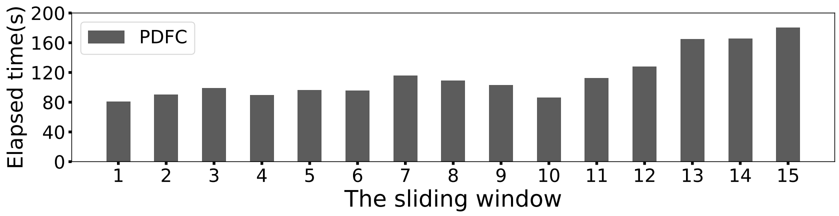


Fig. 12. The elapsed time of **PDFC** as the sliding window slides with the stride.

number of nodes determines the dimensions of the adjacency matrix, which affects the matrix operation and leads to different elapsed times of **PDFC** under the sliding window.

7 CONCLUSIONS

We investigated fair clustering on HINs to ensure structural fairness and eliminate biases inherent toward sensitive attributes. To achieve this goal, we proposed a Prerequisite-driven Fair Clustering algorithm (**PDFC**). In addition, we designed an adjacency matrix update strategy to expand **PDFC** and dynamically capture fair clustering results to deal with stream or time-varying data. Our experiments on three real-world datasets verified the effectiveness and efficiency of **PDFC**. In future research, we will improve fair results for different objectives from the perspective of multi-objective optimization, since we found that the evaluation metrics cannot reduce simultaneously under structural and balance constraints.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (No. U1811263 and No. 62202338). Besides, we also thank anonymous reviewers for their helpful reports.

REFERENCES

[1] Github.com/librahu. *Movielens.* https://github.com/librahu/HIN-Datasets-for-Recommendation-and-Network-Embedding/tree/master/Movielens

[2] Mohsen Abbasi, Aditya Bhaskara, and Suresh Venkatasubramanian. 2021. Fair Clustering via Equitable Group Representations. In *FAccT*. 504–514.

[3] Savitha Sam Abraham, Deepak P, and Sowmya S. Sundaram. 2020. Fairness in Clustering with Multiple Sensitive Attributes. In *EDBT*. 287–298.

[4] N S Altman. 1992. An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *Source: The American Statistician* 46 (1992), 175–185.

[5] Natalia M. Arzeno and Haris Vikalo. 2021. Evolutionary Clustering via Message Passing. *IEEE Trans. Knowl. Data Eng.* 33, 6 (2021), 2452–2466.

[6] Abolfazl Asudeh, H. V. Jagadish, Julia Stoyanovich, and Gautam Das. 2019. Designing Fair Ranking Schemes. In *SIGMOD*. 1259–1276.

[7] Arturs Backurs, Piotr Indyk, Krzysztof Onak, Baruch Schieber, Ali Vakilian, and Tal Wagner. 2019. Scalable Fair Clustering. In *ICML*. 405–413.

[8] Suman Kalyan Bera, Deeparnab Chakrabarty, Nicolas Flores, and Maryam Negahbani. 2019. Fair Algorithms for Clustering. In *NeurIPS*. 4955–4966.

[9] Angela Bonifati, Michael J. Mior, Felix Naumann, and Nele Sina Noack. 2021. How Inclusive are We? *SIGMOD Rec.* 50, 4 (2021), 30–35.

[10] Michele Borassi, Alessandro Epasto, Silvio Lattanzi, Sergei Vassilvitskii, and Morteza Zadimoghaddam. 2020. Sliding Window Algorithms for k-Clustering Problems. In *NeurIPS*. 8716–8727.

[11] Avishek Joey Bose and William L. Hamilton. 2019. Compositional Fairness Constraints for Graph Embeddings. In *ICML*. 715–724.

[12] Sylvain Bouveret, Katarína Cechlárová, Edith Elkind, Ayumi Igarashi, and Dominik Peters. 2017. Fair Division of a Graph. In *IJCAI*. 135–141.

[13] Paul S. Bradley, Olvi L. Mangasarian, and W. Nick Street. 1996. Clustering via Concave Minimization. In *NeurIPS*. 368–374.

[14] Thang Nguyen Bui and Curt Jones. 1992. Finding Good Approximate Vertex and Edge Partitions is NP-Hard. *Inf. Process. Lett.* 42, 3 (1992), 153–159.

[15] Ioannis Caragiannis, Evi Micha, and Nisarg Shah. 2022. A Little Charity Guarantees Fair Connected Graph Partitioning. In *AAAI*. 4908–4916.

[16] Deepayan Chakrabarti, Ravi Kumar, and Andrew Tomkins. 2006. Evolutionary clustering. In *KDD*. 554–560.

[17] Vaggos Chatziafratis, Rad Niazadeh, and Moses Charikar. 2018. Hierarchical Clustering with Structural Constraints. In *ICML*. 773–782.

[18] Lu Chen, Yunjun Gao, Yuanliang Zhang, Christian S. Jensen, and Bolong Zheng. 2019. Efficient and Incremental Clustering Algorithms on Star-Schema Heterogeneous Graphs. In *ICDE*. 256–267.

[19] Xingyu Chen, Brandon Fain, Liang Lyu, and Kamesh Munagala. 2019. Proportionally Fair Clustering. In *ICML*. 1032–1041.

[20] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. 2017. Fair Clustering Through Fairlets. In *NeurIPS*. 5029–5037.

[21] Yushun Dong, Jian Kang, Hanghang Tong, and Jundong Li. 2021. Individual Fairness for Graph Neural Networks: A Ranking based Approach. In *KDD*. 300–310.

[22] Yushun Dong, Jing Ma, Chen Chen, and Jundong Li. 2022. Fairness in Graph Mining: A Survey. *CoRR* abs/2204.09888 (2022).

[23] Mohammadreza Esfandiari, Dong Wei, Sihem Amer-Yahia, and Senjuti Basu Roy. 2019. Optimizing Peer Learning in Online Groups with Affinities. In *KDD*. 1216–1226.

[24] Mehrdad Ghadiri, Samira Samadi, and Santosh S. Vempala. 2021. Socially Fair k-Means Clustering. In *FAccT*. 438–448.

[25] Jibing Gong, Yao Wan, Ye Liu, Xuewen Li, Yi Zhao, Cheng Wang, Qing Li, Wenzheng Feng, and Jie Tang. 2022. Reinforced MOOCs Concept Recommendation in Heterogeneous Information Networks. *CoRR* abs/2203.11011 (2022).

[26] Jibing Gong, Shen Wang, Jinlong Wang, Wenzheng Feng, Hao Peng, Jie Tang, and Philip S. Yu. 2020. Attentional Graph Convolutional Networks for Knowledge Concept Recommendation in MOOCs in a Heterogeneous View. In *SIGIR*. 79–88.

[27] Binbin Gu, Saeed Kargar, and Faisal Nawab. 2022. Efficient Dynamic Clustering: Capturing Patterns from Historical Cluster Evolution. In *EDBT*. 2:351–2:363.

[28] Sumyea Helal, Jiuyong Li, Lin Liu, Esmaeil Ebrahimie, Shane Dawson, Duncan J. Murray, and Qi Long. 2018. Predicting academic performance by considering student heterogeneity. *Knowl. Based Syst.* 161 (2018), 134–146.

[29] Maliha Tashfia Islam, Anna Fariha, Alexandra Meliou, and Babak Salimi. 2022. Through the Data Management Lens: Experimental Analysis and Evaluation of Fair Classification. In *SIGMOD*. 232–246.

[30] Bin Shyan Jong, LuYung Wu, and Te Yi Chan. 2006. Dynamic Grouping Strategies Based on a Conceptual Graph for Cooperative Learning. *IEEE Trans. Knowl. Data Eng.* 18, 6 (2006), 738–747.

[31] Kazi Zainab Khanam, Gautam Srivastava, and Vijay Mago. 2020. The Homophily Principle in Social Network Analysis. *CoRR* abs/2008.10383 (2020).

[32] Bogyeong Kim, Kyoseung Koo, Undraa Enkhbat, and Bongki Moon. 2022. DenForest: Enabling Fast Deletion in Incremental Density-Based Clustering over Sliding Windows. In *SIGMOD*. 296–309.

[33] Bogyeong Kim, Kyoseung Koo, Juhun Kim, and Bongki Moon. 2021. DISC: Density-Based Incremental Clustering by Striding over Streaming Data. In *ICDE*. 828–839.

[34] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[35] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[36] Dan Klein, Sepandar D. Kamvar, and Christopher D. Manning. 2002. From Instance-level Constraints to Space-Level Constraints: Making the Most of Prior Knowledge in Data Clustering. In *ICML*. 307–314.

[37] Matthäus Kleindessner, Samira Samadi, Pranjal Awasthi, and Jamie Morgenstern. 2019. Guarantees for Spectral Clustering with Fairness Constraints. In *ICML*. 3458–3467.

[38] Bo Li, Lijun Li, Ankang Sun, Chenhao Wang, and Yingfan Wang. 2021. Approximate Group Fairness for Clustering. In *ICML*. 6381–6391.

[39] Peizhao Li, Yifei Wang, Han Zhao, Pengyu Hong, and Hongfu Liu. 2021. On Dyadic Fairness: Exploring and Mitigating Bias in Graph Connections. In *ICLR*.

[40] Weihua Li, Sam Zhang, Zhiming Zheng, Skyler J Cranmer, and Aaron Clauset. 2022. Untangling the network effects of productivity and prominence among scientists. *Nature communications* 13, 1 (2022), 1–11.

[41] Xiang Li, Danhao Ding, Ben Kao, Yizhou Sun, and Nikos Mamoulis. 2021. Leveraging Meta-path Contexts for Classification in Heterogeneous Information Networks. In *ICDE*. 912–923.

[42] Xiang Li, Ben Kao, Zhaochun Ren, and Dawei Yin. 2019. Spectral Clustering in Heterogeneous Information Networks. In *AAAI*. 4221–4228.

[43] Xiang Li, Yao Wu, Martin Ester, Ben Kao, Xin Wang, and Yudian Zheng. 2017. Semi-supervised Clustering in Attributed Heterogeneous Information Networks. In *WWW*. 1621–1629.

[44] Jiali Mao, Qiuge Song, Cheqing Jin, Zhigang Zhang, and Aoying Zhou. 2016. TSCluWin: Trajectory Stream Clustering over Sliding Window. In *DASFAA*. 133–148.

[45] Sandra Mattauch, Katja Lohmann, Frank Hannig, Daniel Lohmann, and Jürgen Teich. 2020. A bibliometric approach for detecting the gender gap in computer science. *Commun. ACM* 63, 5 (2020), 74–80.

[46] Deepak P and Savitha Sam Abraham. 2021. FairLOF: Fairness in Outlier Detection. *Data Sci. Eng.* 6, 4 (2021), 485–499.

[47] Tahleen A. Rahman, Bartlomiej Surma, Michael Backes, and Yang Zhang. 2019. Fairwalk: Towards Fair Graph Embedding. In *IJCAI*. 3289–3295.

[48] Aida Rahmattalabi, Phebe Vayanos, Anthony Fulginiti, Eric Rice, Bryan Wilder, Amulya Yadav, and Milind Tambe. 2019. Exploring Algorithmic Fairness in Robust Graph Covering Problems. In *NeurIPS*. 15750–15761.

[49] Babak Salimi, Luke Rodriguez, Bill Howe, and Dan Suciu. 2019. Interventional Fairness: Causal Database Repair for Algorithmic Fairness. In *SIGMOD*. 793–810.

[50] Roy Schwartz and Roded Zats. 2022. Fair Correlation Clustering in General Graphs. In *APPROX/RANDOM*. 37:1–37:19.

[51] Uri Shaham, Kelly P. Stanton, Henry Li, Ronen Basri, Boaz Nadler, and Yuval Kluger. 2018. SpectralNet: Spectral Clustering using Deep Neural Networks. In *ICLR*.

[52] Jianbo Shi and Jitendra Malik. 2000. Normalized Cuts and Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* 22, 8 (2000), 888–905.

[53] Hanyu Song, Peizhao Li, and Hongfu Liu. 2021. Deep Clustering based Fair Outlier Detection. In *KDD*, Feida Zhu, Beng Chin Ooi, and Chunyan Miao (Eds.). 1481–1489.

[54] Yizhou Sun and Jiawei Han. 2012. Mining heterogeneous information networks: principles and methodologies. *Synthesis Lectures on Data Mining and Knowledge Discovery* 3, 2 (2012), 1–159.

[55] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. *Proc. VLDB Endow.* 4, 11 (2011), 992–1003.

[56] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2022. Heterogeneous Information Networks: the Past, the Present, and the Future. *Proc. VLDB Endow.* 15, 12 (2022), 3807–3811.

[57] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. ArnetMiner: extraction and mining of academic social networks. In *KDD*. 990–998.

[58] Ulrike von Luxburg. 2007. A tutorial on spectral clustering. *Stat. Comput.* 17, 4 (2007), 395–416.

[59] Hanchen Wang, Rong Hu, Ying Zhang, Lu Qin, Wei Wang, and Wenjie Zhang. 2022. Neural Subgraph Counting with Wasserstein Estimator. In *SIGMOD*. 160–175.

[60] Sheng Wang, Zhifeng Bao, J. Shane Culpepper, Timos Sellis, and Xiaolin Qin. 2019. Fast Large-Scale Trajectory Clustering. *Proc. VLDB Endow.* 13, 1 (2019), 29–42.

[61] Sheng Wang, Yuan Sun, and Zhifeng Bao. 2020. On the Efficiency of K-Means Clustering: Evaluation, Optimization, and Algorithm Selection. *Proc. VLDB Endow.* 14, 2 (2020), 163–175.

[62] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. 2019. Heterogeneous Graph Attention Network. In *WWW*. 2022–2032.

[63] Samuel F. Way, Daniel B. Larremore, and Aaron Clauset. 2016. Gender, Productivity, and Prestige in Computer Science Faculty Hiring Networks. In *WWW*. 1169–1179.

[64] Dong Wei, Md Mouinul Islam, Baruch Schieber, and Senjuti Basu Roy. 2022. Rank Aggregation with Proportionate Fairness. In *SIGMOD*. 262–275.

[65] Hongzhi Yin, Lei Zou, Quoc Viet Hung Nguyen, Zi Huang, and Xiaofang Zhou. 2018. Joint Event-Partner Recommendation in Event-Based Social Networks. In *ICDE*. 929–940.

[66] Jingyi You, Chenlong Hu, Hidetaka Kamigaito, Kotaro Funakoshi, and Manabu Okumura. 2021. Robust Dynamic Clustering for Temporal Networks. In *CIKM*. 2424–2433.

[67] Jifan Yu, Gan Luo, Tong Xiao, Qingyang Zhong, Yuquan Wang, Wenzheng Feng, Junyi Luo, Chenyu Wang, Lei Hou, Juanzi Li, Zhiyuan Liu, and Jie Tang. 2020. MOOCCube: A Large-scale Data Repository for NLP Applications in MOOCs. In *ACL*. 3135–3142.

[68] Jifan Yu, Yuquan Wang, Qingyang Zhong, Gan Luo, Yiming Mao, Kai Sun, Wenzheng Feng, Wei Xu, Shulin Cao, Kaisheng Zeng, Zijun Yao, Lei Hou, Yankai Lin, Peng Li, Jie Zhou, Bin Xu, Juanzi Li, Jie Tang, and Maosong Sun. 2021. MOOCCubeX: A Large Knowledge-centered Repository for Adaptive Learning in MOOCs. In *CIKM*. 4643–4652.

[69] Ziqian Zeng, Rashidul Islam, Kamrun Naher Keya, James R. Foulds, Yangqiu Song, and Shimei Pan. 2021. Fair Representation Learning for Heterogeneous Information Networks. In *ICWSM*. 877–887.

[70] Hantian Zhang, Xu Chu, Abolfazl Asudeh, and Shamkant B. Navathe. 2021. OmniFair: A Declarative System for Model-Agnostic Group Fairness in Machine Learning. In *SIGMOD*. 2076–2088.

[71] Xiaotong Zhang, Han Liu, Xiao-Ming Wu, Xianchao Zhang, and Xinyue Liu. 2021. Spectral embedding network for attributed graph clustering. *Neural Networks* 142 (2021), 388–396.

[72] Imtiaz Masud Ziko, Jing Yuan, Eric Granger, and Ismail Ben Ayed. 2021. Variational Fair Clustering. In *AAAI*. 11202–11209.

Received October 2022; revised January 2023; accepted February 2023