Effective Individual Fairest Community Search over Heterogeneous Information Networks

Taige Zhao
Deakin University
Geelong, Australia
zhaochr@deakin.edu.au

Jianxin Li*
Deakin University
Geelong, Australia
jianxin.li@deakin.edu.au

Ningning Cui
Chang'an University
Xi'an, China
csnncui@chd.edu.cn

Wei Luo
Deakin University
Geelong, Australia
wei.luo@deakin.edu.au

ABSTRACT

Community search over heterogeneous information networks has been applied to wide domains, such as activity organization and team formation. From these scenarios, the members of a group with the same treatment often have different levels of activity and workloads, which causes unfairness in the treatment between active members and inactive members (called individual unfairness). However, existing works do not pay attention to individual fairness and do not sufficiently consider the rich semantics of HINs (e.g., high-order structure), which disables complex queries. To fill the gap, we formally define the issue of individual fairest community search over HINs (denoted as IFCS), which aims to find a set of vertices from the HIN that own the same type, close relationships, and small difference of activity level and has been demonstrated to be NP-hard. To do this, we first develop an exploration-based filter that reduces the search space of the community effectively. Further, to avoid repeating computation and prune unfair communities in advance, we propose a message-based scheme and a lower bound-based scheme. At last, we conduct extensive experiments on four real-world datasets to demonstrate the effectiveness and efficiency of our proposed algorithms, which achieve at least $\times 3$ times faster than the baseline solution.

CCS CONCEPTS

• **Do Not Use This Code** → **Generate the Correct Terms for Your Paper**; *Generate the Correct Terms for Your Paper*; Generate the Correct Terms for Your Paper; Generate the Correct Terms for Your Paper.

---

*Both authors contributed equally to this research.

---

1 INTRODUCTION

Heterogeneous information networks [8] (HINs) are networks involving interconnected objects with different types (aka *labels*). Compared to the traditional homogeneous network with the same type, HIN enables to storing more rich semantic information and has become prevalent in various domains, such as citation networks [38], social networks [21] and human-resource networks [36]. Fig. 1 illustrates an HIN modeling a Database System and Logic Programming (DBLP) network, which contains four types of vertices: *author* (A), *paper* (P), *venue* (V), and *topic* (T). The edges between different types of vertices have different semantic relationships, such as authorship (A-P) and publication (P-V).

In recent years, community search over HIN has attracted much attention due to its importance in many applications, such as recommendation [10, 17], team formation [37] and identification of protein functions [7]. Existing works like [8, 14–16, 31] extend the traditional cohesive community models such as k-core [1], k-truss [35] and k-clique [9], and these works require users to customize query requests like meta-path [8, 16, 31], relational constraints [15] and motif [14]. However, they did not fully consider the rich semantics of the HIN, which can not handle the complex customized query request. The other important factor in the community search problem is the fairness. As revealed in [10, 18, 28], the notion of fairness was proposed to mitigate the bias and systematic discrimination for disadvantaged people in terms of sensitive features (e.g., gender, age and race) in communities. For example, assume that people are almost composed of a specific gender in a community. In this case, members of the community are generally inclined to communicate with people of a specific gender, which causes discrimination for people of the other gender. To mitigate the discrimination in the community search problem, Matth et al. [19] considered the notion of fairness as the difference in the proportion of vertex types between communities and proposed a fairness-based clustering method, which guarantees the similarity proportion of vertex type in each cluster. Similarly, Zhang et al. [34] considered fairness as the difference in vertices quantity between different types of vertex in a community. However, these works only focus on group-based fairness, i.e., keeping the similarity of certain metrics between groups,

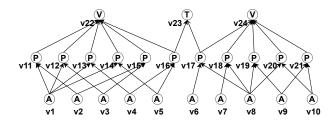Taige Zhao, Jianxin Li, Ningning Cui, and Wei Luo

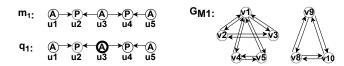

Figure 1: An example of HIN

Figure 2: An example of fairness community search

which can not handle the fairness at a fine granularity level, e.g., mitigating the discrimination between members (called individual fairness) in a group.

IFCS problem. In this paper, we study the problem of **I**ndividual **F**airest **C**ommunity **S**earch over HINs (IFCS) to find a set of vertices with the target type hold (1) each vertex satisfies the customized query request and has close relationships; (2) the vertices have small difference of level of activities. To formulate the IFCS problem, we face two key questions: (1) How to model the customized query requests and relationships for vertices in a community? (2) How to measure the active level of a member and the difference of active level among members in a community?

For the former question, we extend the well-known concept of motif [14] to model the customized query request and relationship. A motif, also known as a *higher-order structure* or *graphlet*, is a small subgraph pattern. For example, Figure 2 shows two motifs m_1, m_2. To specify the target type, we select a vertex with user preference in a motif and use its type as the target type, and call such motif the *target-aware motif*. For instance, Figure 2 illustrates a target-aware motif q_1, where the vertex u_3 with a wider border is the selected vertex. It describes that each author should collaborate on a paper with the other two authors in the community, respectively. In this case, the author v_1 in Figure 1 has relationships with author v_2 and v_3 through the instance of target-aware motif $\{v_3, v_{12}, v_1, v_{11}, v_2\}$.

For the latter question, we follow the idea in [10] that the active level of a member is related to the number of motif instances around it. In addition, we adopt the concept of the Gini coefficient [11] to measure the difference in active levels among members of a community. As the difference in active level among members in a community becomes small, the gini coefficient will approach zero. By carefully considering the above problems, given an HIN and a target-aware motif with a target type, the objective of IFCS is to find the community that contains vertices of the target type connected via instances of the target-aware motif and has the lowest Gini coefficient. Example 1 describes a scenario where the fairest community can be found.

Example 1. *Assume an institute wants to recruit a group of researchers for a development position such that each researcher must*

collaborate on at least a paper with the other two researchers of the group, respectively. This customized query request can be modeled as the target-aware motif q_1 in Figure 2. Intuitively, if a researcher collaborates more papers with more researchers, his/her level of activity should be higher. In this case, highly active researchers in the group may be more suitable for better treatment because they have more project experience and the capability to do more work. In addition, the existence of low-activity researchers is unfair to others, because they did fewer projects but have the same treatment as others. From a fairness perspective, a group is better if researchers in the group have a small difference in level of activity. Our proposed fairest community search problem can support such a scenario.

In this case, we enumerate the instances of motif q_1 from the HIN in Figure 1, and get two groups $T_{c1} = \{v_1, v_2, v_3, v_4, v_5\}$, $T_{c2} = \{v_8, v_9, v_{10}\}$ shown in Figure 2 G_{M_1}. We can see that all the authors in T_{c2} collaborate on a paper with the other two authors, but the authors v_2, v_3, v_4, v_5 in T_{c1} collaborate a paper with other two authors and the author v_1 in T_{c1} collaborate four papers with other five authors. Obviously, the activity level of v_1 in T_{c_1} is different from others, and the activity levels of members in T_{c2} are the same. So T_{c2} have a small difference in activity level and we can identify the community T_{c2} as the fairest community.

Challenges. To find the fairest communities, a basic method is to enumerate all the communities and calculate the fairness score of each community, then return the maximal community has the lowest fairness score. However, there are two computational challenges of the basic method: (1) we need to re-enumerate an instance of motif around each vertex of the target type to verify the satisfaction of the customized query request once a vertex of the target type is removed; (2) we need to enumerate all motif instances around each member of the community to calculate their active level.

To conquer the first challenge, we propose an exploration-based filter strategy to reduce the potential target vertices that need to be checked and a message-passing based optimization strategy to avoid redundant computation. To solve the second challenge, we derive the lower bound of the fairness score to prune the unfair communities in advance.

Contributions. We state our main contributions as follows:

- To the best of our knowledge, this is the first work to formalize the problem of *individual fairest community search* (IFCS) over HINs, which introduces individual fairness for the community model.
- As the IFCS problem is NP-hardness, we develop a *Filter-Verify* algorithm to solve the IFCS problem.
- We further propose an exploration-based and a message-passing based optimization strategy to reduce redundant computation, then we provide a lower bound based optimization strategy to identify and prune the unfair community in advance during the process of community search.
- We conducted extensive experiments on four real-world datasets to demonstrate the effectiveness of the proposed fair community model and the efficiency of the proposed optimization strategies.

The rest of this paper is organized as follows. First, we present the definition of the IFCS problem in Section 2. Then we describe

Table 1: The summary of notations

Notation	Description
$G = (V_G, E_G)$	an HIN with vertex set V_G and edge set E_G
$\psi_G(v)$	the type of vertex v in HIN G
$\mathcal{A}$	the vertex type set in HIN
$q = (V_q, E_q, v_t)$	a target-aware motif with vertex set V_q, edge set E_q and target vertex v_t
ξ_q	the bijective function of motif q
g_m^q	an instance of motif q
$p(v)$	the active level of target vertex instance v
T_c, C	the target-aware community and the candidate target-aware community
$FS(T_c)$	the fairness score of target-aware community T_c

the detailed procedure for the baseline solution of the problem in Section 3. Next, we discuss the techniques in Section 4 to speed up finding the fairest communities. In Section 5, we report the experimental setting and the evaluation results. Finally, we review the related work in Section 6, and conclude our work in Section 7.

2 PROBLEM DEFINITION

2.1 Preliminaries

We model an HIN as a directed graph $G = (V_G, E_G)$ with a vertex-type mapping function $\psi : V_G \rightarrow \mathcal{A}$, where each vertex $v \in V_G$ has a vertex type $\psi(v) \in \mathcal{A}$. We use $|\cdot|$ to denote the number of elements in a set, for example, $|V_G|$ and $|E_G|$ denote the number of vertices and edges in G, respectively. Here, we summarize the most important notations in Table 1.

We first introduce the notion of motif, which has been widely used to describe subgraph patterns.

Definition 1 (Motif [25]). Given an HIN with its vertex-type set $\mathcal{A}$, a motif $q = (V_q, E_q)$ is a small connected HIN with vertex-type mapping functions $\psi_q : V_q \rightarrow \mathcal{A}_q$, where $\mathcal{A}_q \subseteq \mathcal{A}$.

Definition 2 (Instance of Motif [25]). Given an HIN G with its vertex-type mapping functions ψ and a motif $q = (V_q, E_q)$ with its vertex-type mapping functions ψ_q, a subgraph $g_m^q = (V_m^q, E_m^q)$ of G is an instance of motif q, if $\exists$ bijection function $\xi_q : V_q \rightarrow V_m$ satisfies (1) $\forall v \in V_q, \psi_q(v) = \psi(\xi_q(v))$ holds; (2) $\forall (v, v') \in E_q$, $(\xi_q(v), \xi_q(v')) \in E_m$ holds.

2.2 Target-aware Community

To model the customized query request and formulate the target-aware community, we introduce the following new concepts.

Definition 3 (Target-aware Motif). Given a motif $q = (V_q, E_q)$ with its vertex-type mapping function ψ_q, we can specify a vertex $v_t \in V_q$ as the *target vertex*. We also use $q = (V_q, E_q, v_t)$ to represent and call it *target-aware motif*.

For simplicity, all the motifs mentioned in the following are target-aware motifs.

Definition 4 (Instances of Target Vertex). Given an HIN G, a motif $q = (V_q, E_q, v_t)$ with its bijection function ξ_q and the instances of motif q in G, the instances of target vertex v_t are the corresponding vertex $\xi_q(v_t)$ of the instances of motif q.

Definition 5 (Instances of Motif around Instance of Target Vertex). Given an HIN G, a motif $q = (V_q, E_q, v_t)$ with its bijection function ξ_q and an instance v of target vertex v_t in G, the instances of motif around v are the instances of motif in G whose corresponding vertex $\xi_q(v_t) = v$.

Here, we regard the instance of target vertex v as *M-neighbor* of the instance of target vertex u if there exists an instance of motif around u containing v. In addition, we regard these two instances of target vertex v, u as *M-connected* if there exists a chain of vertices from v to u such that one vertex of any two adjacent vertices in the chain is a M-neighbor of the other vertex.

Definition 6 (Active Level of Target Vertex Instance). Given an HIN G, a motif $q = (V_q, E_q, v_t)$ and a set of M-connected target vertex instances I, the active level s_v of a target vertex instance $v \in I$ is the number of motif instances $g_m^q = (V_m^q, E_m^q)$ which satisfies $\forall v' \in V_m^q \wedge \psi(v') = \psi_q(v_t), v' \in I$ around v in G.

Definition 7 (Target-aware Community). Given an HIN G with its vertex-type mapping function ψ and a motif $q = (V_q, E_q, v_t)$ with its bijection function ξ_q, the target-aware community T_c is a set of M-connected target vertex instances in G that satisfies $\forall v \in T_c$, $s_v \geqslant 1$.

Based on definition 7, we call a target-aware community is maximal if it is not contained in any other target-aware community. For a motif q, we can also induce a homogeneous graph G_M (defined in the following) from an HIN G, called M-graph, to record the M-neighbors of each member of the target-aware community. Essentially, the maximal target-aware communities are the vertices of the weakly connected subgraphs of G_M.

Definition 8 (M-graph). Given an HIN G and a motif q, the M-graph is a directed homogeneous network $G_M = (V_M, E_M)$ such that (1) it contains all the M-connected target vertex instances whose active level is not less than 1; (2) for each vertex $v_M \in V_M$, it has an edge linked to each of its M-neighbors.

2.3 Problem Statement

In this paper, we invoke a widely-accepted fairness measurement, i.e., Gini coefficient [11] to measure the similarity of active level among members in a target-aware community. It can be defined as follows.

Definition 9 (Fairness Score of Target-Aware Community). Given a target-aware community T_c and the list of active levels S of each member in T_c, the fairness score of T_c can be measured as follows:

$$FS = \frac{\sum_{i=1}^{|S|} \sum_{j=1}^{|S|} |s_i - s_j|}{2|S| \sum_{m=1}^{|S|} s_m} \tag{1}$$

where s_i, s_j, s_m are active levels in S. Based on the property of Gini coefficient, the fairness score is in $(0, 1]$, where 0 represents perfect equality while 1 means maximal inequality. Intuitively, it

is directly proportional to the difference of active levels among instances of target vertex and inversely proportional to the number of target vertex instance.

Example 2. *Consider the HIN G in Figure 1 and the motif q_1 in Figure 2. We can get the M-graph G_{M1} and two maximal target-aware communities $T_{c1} = \{v_1, v_2, v_3, v_4, v_5\}$, $T_{c2} = \{v_8, v_9, v_{10}\}$ shown in Figure 2. For T_{c1}, the active levels of v_1, v_2, v_3, v_4, v_5 are 12, 2, 2, 2, 2, respectively, and the fairness score of T_{c1} is 0.4. For T_{c2}, the active levels of v_8, v_9, v_{10} is 2, 2, 2 and the fairness score of T_{c2} is 0.*

Based on the above definitions, the target of this work is to find the maximal target-aware communities owning the minimum fairness score. Next, we formalize the problem as Individual Fairest Community Search (IFCS) over HIN.

Problem Statement (Individual Fairest Community Search). *Given an HIN $G = (V_G, E_G)$ and a motif $q = (V_q, E_q, v_t)$, the problem of Individual Fairest Community Search (IFCS) over HIN is to find a maximal target-aware community T_c from G satisfying:*

$$\arg \min FS(T_c)$$

Problem Complexity: Now, we demonstrate that the IFCS problem is NP-hard by reducing the motif instance enumeration problem to it. Given an HIN G that has a vertex of type b and all other vertices with type a, we want to find a target-aware community that each vertex of type b in the community should connect at least two vertices of type a, that is, the motif q is a small HIN graph that contains a vertex with type b connected with two vertices of type a. This is equivalent to enumerating all instances of q from G because the active level of each community member is calculated by enumerating all instances of q in G. Apparently, the feasible solution of IFCS corresponds to the motif instance enumeration problem. As discussed in [13], enumerating the instance of motif from a graph is an NP-complete problem. Therefore, the IFCS problem is also an NP-hard problem.

3 THE FILTER-VERIFY SOLUTION

To address the IFCS problem, a basic solution is to enumerate all the maximal target-aware communities and calculate their fairness scores. Then, the fairest target community can be returned by selecting the one that has the minimum fairness score. In specific, we follow the same paradigm of the filter-verify algorithm [3]. It consists of three steps: (1) filter the unsatisfied vertices that are not in M-graph; (2) build the M-graph and calculate the active level of each vertex of M-graph; (3) for each weakly connected subgraph g_M in M-graph, calculate its fairness score, then return the vertices in g_M that has the lowest fairness score. The process is presented in Algorithm 1.

We first initialize a set N_R to store the edges of M-graph and a dictionary D to store the active level of each vertex of target-aware communities (Line 1). Next, we filter the unsatisfied vertices by enumerating a motif instance around the vertices of the target type (Lines 2-7). We first initialize a list V_N to store the candidate vertices of M-graph and add vertices with type $\psi_q(v_t)$ in G to V_N (Line 3). For each iteration, we enumerate a motif instance around each vertex v in V_N using the state-of-the-art subgraph matching algorithm [27]. If there is no motif instance g_m^q around v, we can

---

Algorithm 1: Basic Solution

Input: An HIN $G = (V_G, E_G)$, a motif $q = (V_q, E_q, v_t)$ with vertex type mapping function ψ_q.
Output: A list of maximal individual fairness communities $maxC$

1   $N_R \leftarrow \varnothing$, $D \leftarrow$ empty dictionary ;
2   **repeat**
3      $V_N \leftarrow$ vertices with type $\psi_q(v_t)$ in G ;
4   **for each** *vertex* $v \in V_N$ **do**
5   **if** *no motif instance around v found by an existing subgraph isomorphism algorithm from G* **then**
6   Delete v from G ;
7   **until** $V_N \setminus$ vertices with type $\psi_q(v_t)$ in $G = \varnothing$;
8   **for each** *instance $g_m^q = (V_m^q, E_m^q)$ of motif q around v in G found by an existing subgraph isomorphism algorithm* **do**
9   **if** $D[v] = \varnothing$ **then** $D[v] \leftarrow 1$;
10   **else** $D[v] \leftarrow D[v] + 1$;
11   **for** *each vertex v' with type $\psi_q(v_t)$ in $V_m^q \setminus v$* **do**
12        $N_R \leftarrow N_R \cup \{(v', v)\}$
13   $G_M = (V_M, E_M) \leftarrow$ generate graph using N_R ;
14   $maxC \leftarrow [\,]$, $maxFS \leftarrow 0$;
15   **for each** *weakly connected subgraph $g_w = (V_w, E_w)$ of G_M* **do**
16   **if** $|V_w| \geq k$ **then**
17        $P \leftarrow [\,]$;
18   **for** $v \in V_w$ **do** $P.add(D[v])$;
19        $FS \leftarrow$ calculate the fairness score using active levels in P ▷ Equation 1 ;
20   **if** $FS < maxFS$ **then**
21           $maxC$.removeAll();
22           $maxC$.add(V_w), $maxFS = FS$;
23   **else if** $FS = maxFS$ **then**
24           $maxC$.add(V_w);
25   **Return** $maxC$;

---

know v is not included in M-graph and delete it from G (Lines 4-6). We repeat the above process until no vertex is removed in this iteration.

Next, we build the M-graph and calculate the active level of each target vertex instance (Lines 8-13). For each vertex v with type $\psi_q(v_t)$ in G, we enumerate the rest of motif instances around v using the state-of-the-art subgraph matching algorithm [27]. Once a motif instance g_m^q around v is enumerated, we update the active level of v in D and add the edges that connect v with other vertices of target type in g_m^q to N_R (Lines 8-12). After enumerating all the motif instances around vertices in V_N, we can build the M-graph G_M using edges in N_R (Line 13).

Finally, we get the target-aware communities and calculate their fairness scores (Lines 14-24). We initialize $maxC$ and $maxFS$ to store the target-aware communities and the fairness score of the target-aware communities (Line 14). Then we get the maximal target-aware communities by returning the weakly connected subgraphs of G_M. For each weakly connected subgraph, we calculate its fairness score using active levels stored in D. Once the fairness score of a community is smaller than the existing fairest communities, we remove all communities in $maxC$ and put this community to $maxC$ and the corresponding fairness score to $maxFS$ (Lines 15-24). In the end, the fairest communities in $maxC$ are the final results.

Example 3. *Take the HIN G in Figure 1, the motif $q_1 = (V_q, E_q, v_t)$ in Figure 2 as an example. Firstly we enumerate motif instances around $v_1 - v_{10}$ and get a motif instances around $v_1 - v_5, v_8 - v_{10}$. So we delete the vertices v_6, v_7. Due to the leaving of v_6, v_7 may cause the*

$v_1 - v_5, v_8 - v_{10}$ do not exist a motif instance around them, we re-enumerate motif instances around $v_1 - v_5, v_8 - v_{10}$ and get a motif instance around them, respectively. Next, we enumerate the rest of the motif instances around $v_1 - v_5, v_8 - v_{10}$ and record the active levels $s_{v_1} = 12, s_{v_2} = 2, s_{v_3} = 2, s_{v_4} = 2, s_{v_5} = 2, s_{v_8} = 2, s_{v_9} = 2, s_{v_{10}} = 2$. After that, we generate the M-graph G_{M1} shown in Figure 2 and get two target-aware community $T_{c1} = \{v_1, v_2, v_3, v_4, v_5\}$, $T_{c2} = \{v_8, v_9, v_{10}\}$ from G_{M1}, and calculate the fairness score 0.4, 0 of T_{c1} and T_{c2}, respectively. Finally, we can conclude that the fairest target-aware community is T_{c2}.

Complexity Analysis. The time complexity analysis of Algorithm 1 consists of the following steps. We first construct the index discussed in [27] to enumerate instances of motif, which takes $O(|E_G| \cdot |E_q|)$ time. For each iteration of target vertex identification process, it takes $O(d^{\frac{3}{2}d_t} \cdot |V_t|)$ to enumerate a motif instance for each vertex in V_t using the algorithm in [12], where d is the average degree of HIN G, d_t is the length of the shortest path between target vertex and its most distanced node in motif q, and $|V_t|$ is the number of vertices of target type. In the worst case, the number of iteration could be $|V_t|$, so the total time complexity of the target vertex identification process is $O(d^{\frac{3}{2}d_t} \cdot V_t^2)$. In the process of active level calculation, the time complexity is $|V_t||V_q|$ because the number of motif instances could be $|V_t||V_q|$. In the process of maximal target-aware communities generation, the time cost is $O(|V_t| + |E_G|)$ by returning the weakly connected subgraphs of G_M. Thus, the total time complexity of Algorithm 1 is $O(d^{\frac{3}{2}d_t} \cdot V_t^2 + |V_t||V_q| + |E_G| \cdot |E_q|)$ in total.

It is obvious that the complexity of basic solution is high. The main drawbacks lie in: (1) all the vertices of target type need to be identified for each iteration in target vertex identification process; (2) all the vertices in M-graph need to enumerate the motif instances around them to calculate their active level.

4 OPTIMIZATION

To overcome the above drawbacks, in this section, we first propose an exploration-based filter to prune the ineligible vertices of the target type. Then, we develop a message-passing based optimization strategy to avoid redundant computation. Finally, we propose a lower bound-based to filter the unfair community in advance by using the derived lower bound of fairness score.

4.1 Reducing Potential Target Vertices

In this subsection, we propose an exploration-based filter to further reduce potential instances of target vertex. Before introducing the details, we first introduce a query vertex filtering strategy, called Neighborhood Label Frequency (NLF) filter [2]. It aims to find the candidate vertices of a query vertex in a motif that may be contained in instances of motif.

Definition 10 (Neighborhood Label Frequency (NLF) Filter [2]). Given a query vertex u in a motif q and a vertex v in an HIN G, v is the candidate vertex of u if v satisfies $\forall l \in L_N(u), d_i(v, l) < d_i(u, l) \land d_o(v, l) < d_o(u, l)$.

where $L_N(v)$ is the set of unique labels of u's in-neighbors and out-neighbors, $d_i(v, l)$ is the number of in-neighbors of v with label

l, and $d_o(v, l)$ is the number of out-neighbors of v with label l. In addition, we also introduce the other candidate vertex filter method, which supports our exploration-based filter search.

Definition 11 (Exact Star Isomorphism Constraint [27]). Given an HIN G, a query vertex u of a motif q and a candidate vertex v of u pass the NLF filter in G, v satisfies the exact star isomorphism constraint if $\forall u' \in N(u), \exists v' \in N(v)$ such that $v' \in u'.C$.

where $N(u)$ is the in-neighbors and out-neighbors of u, $u'.C$ is a set of candidate vertices of u' in HIN G. Intuitively, if the candidate vertex v of u satisfies the exact star isomorphism constraint, there exists at least one candidate vertex of u' in neighbors of v for each neighbor u' of u. Here, we adopt the breadth-first search (BFS) order starting from the vertex v_t in q to find the candidate vertices of each query vertex. Based on the exact star isomorphism constraint and BFS order, we have the following corollary.

Corollary 1. Given an HIN G, a query vertex u of a motif q, the BFS order π of q and a vertex v in G. If v satisfies the exact star isomorphism constraint, it must hold the following two conditions: (1) $\forall u' \in N(u) \land idx_{u'}(\pi) < idx_u(\pi), \exists v' \in N(v)$ such that $v' \in u'.C$; (2) $\forall u' \in N(u) \land idx_{u'}(\pi) > idx_u(\pi), \exists v' \in N(v)$ such that $v' \in u'.C$;

where $idx_u(\pi)$ is the position (i.e., index) of u in the searching order π. Based on the NLF filter and corollary 1, we can find the candidate target vertex instances and explore the candidate regions around each candidate target vertex instance that may contain motif instances around it. Intuitively, the candidate region is composed of the candidate vertices of each query vertex. In this case, the candidate M-neighbors of a candidate target vertex instance v are the vertices of the target type in the candidate region. We use a directed homogeneous network, denoted as CM-graph, to store the candidate M-connected target vertex instances.

Definition 12 (CM-graph). Given a HIN $G = (V_G, E_G)$ and a motif $q = (V_q, E_q, v_t)$, the CM-Graph is a directed homogeneous graph $G_{CM} = (V_{CM}, E_{CM})$ such that (1) it contains all the vertices of target type $v_{CM} \in V_{CM}$ passing the NLF filter and satisfying the constraints in corollary 1; (2) for each vertex $v_{CM} \in V_{CM}$, it has an edge linked to each of its candidate M-neighbors.

We process the exploration-based filter in two steps: (1) generate the candidate regions and CM-graph to explore the candidate M-connected vertices using condition 1 of Corollary 1 in the forward candidate exploration; (2) refine the candidate regions and CM-graph to prune ineligible candidate M-connected vertices using condition 2 of Corollary 1 in the backward candidate refinement; **Forward Candidate Exploration**. We first initialize S to store the edges of candidate regions, E_{CM} to store the edges in CM-graph, and π to store the BFS order of motif q (Line 1). Then we get the candidate target vertex instances C from HIN G by selecting the vertices of the target type passing the NLF filter (Lines 2-3), and delete the vertices of type $\psi_q(v_t)$ in V_G but not contained in C to reduce the searching space (Line 4). Next, we explore the candidate region around each candidate target vertex instance $c \in C$ following the BFS-order π in the forward candidate exploration process (Lines 5-31).

Specifically, we initialize S' to store the set of edges in the candidate region around c, u.C' to store the candidate vertices of a query

(a) An example of motif
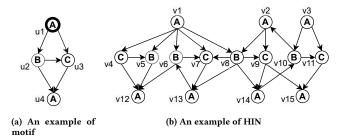
(b) An example of HIN

Figure 3: Example of exploration-based filtering search

vertex u, and use a boolean variable i-add to denote whether v has a candidate region around it (Line 6). Intuitively, i-add is false if a query vertex has no candidate vertex, which means there is no candidate region around v. Then we get the candidate vertex of each vertex $u \in \pi \setminus v_t$ in the candidate region around c (Lines 7-22). We first get the neighborhoods N'_u of u whose candidate vertices have been found (Line 9). Then we randomly select a vertex u_b in N'_u and use its candidate vertices to find the candidate vertex of u (Lines 10-22). To achieve this, for each candidate vertex v of u_b, we first find the neighbours $N(v)$ of v satisfying: (1) have the same type as u; (2) pass the NLF filter, and (3) have the same edges direction $dir(v, v')$ between v and $v' \in N(v)$ as edges direction $dir(u_b, u)$ between vertices u_b and u (Line 12). If there does not exist such neighbor, the i-add is set to be false (Line 22). Otherwise, we verify whether v' satisfies condition 1 of Corollary 1 (Lines 13-21).

We use E_t to store the edges between v' and its neighbors that are contained in the candidate vertices of query vertices $N'_u \setminus u_b$ (Line 13), and use a boolean variable c-add to denote whether v' satisfies the condition 1 of Corollary 1 (Line 14). For each query vertex $\hat{u} \in N'_u \setminus u_b$, we add the edges between candidate vertices of $\hat{u}$ and v' into E'_t (Line 16). If E'_t is empty, we know that there is no candidate vertex of $\hat{u}$ around v', i.e., v' does not satisfy the condition 1 of Corollary 1 (Lines 17-18). Otherwise, we add edges in E'_t to E_t (Line 19). If v' is eligible, we add edges in E_t to S' (Lines 20-21). Once v is an eligible candidate vertex instance, we add the edges in S' to S and add candidate vertices of query vertex u in motif to the candidate target vertex set $\hat{u}.C$ (Lines 23-24). Finally, we get the candidate M-neighbors of c by selecting the vertices with type $\psi_q(v_t)$ in the candidate region around c except c, then add edges between c and its candidate M-neighbors into E_{CM} (Lines 25-26). After that, we filter the ineligible M-connected candidate target vertex instance whose quality is less than k (Lines 27-29) and delete the vertices with type $\psi_q(v_t)$ in V_G but not included in $v_t.C$ to reduce the searching space in backward exploration process (Lines 30-31).

Backward Candidate Refinement. In the backward processing, we refine each candidate region based on the unexploited neighbors of each query vertex u in the forward candidate exploration process, i.e., the in-neighbors and out-neighbors $N(u)$ of u whose index is larger than u in BFS order (Lines 32-41). In contrast to the forward processing, now we find the candidate vertices of each query vertex following the in reverse order of π and get the refined candidate region of each vertex of target type in $u_f.C$.

Algorithm 2: Exploration-based Filter(G, q)

Input: An HIN $G = (V_G, E_G)$, a motif $q = (V_q, E_q, v_t)$ with vertex type mapping function ψ_q
Output: a CM-Graph and a refined G

1   $S \leftarrow \varnothing, E_{CM} \leftarrow \varnothing, C \leftarrow \varnothing, \pi \leftarrow$ BFS order of q ;
2   **for each** *vertices* $v \in G$ **do**
3     $\lfloor$ $C \leftarrow C \cup \{v \mid \psi(v) = \psi_q(v_t), v$ pass the NLF filter $\}$;
4   Delete vertices $\{v \in V_G \mid \psi(v) = \psi_q(v_t)\} \setminus C$ from G ;
   // Lines 5-31: Forward candidate Exploration
5   **for each** *vertices* $c \in C$ **do**
6     $S' \leftarrow \varnothing, v_t.C' \leftarrow \{c\}$, i-add $\leftarrow$ True ;
7   **for each** *query vertex* $u \in \pi \setminus v_t$ **do**
8       $u.C' \leftarrow \varnothing$;
9       $N'_u \leftarrow \{u' \in N(u) \mid idx_u(\pi) < idx_{u'}(\pi)\}$;
10      $u_b \leftarrow$ Random select a vertex from N'_u ;
11   **for each** *vertex* $v \in u_b.C'$ **do**
12   **for each** *vertex* $v' \in \{\hat{v} \in N(v) \mid \psi(\hat{v}) = \psi_q(u) \wedge v$ *pass the NLF filter* $\wedge dir(v, \hat{v}) = dir(u_b, u)\}$ **do**
13          $E_t \leftarrow$ edges between v and v' with the same direction as edges between u_b and u;
14   c-add $\leftarrow$ True ;
15   **for each** *vertex* $\hat{u} \in N'_u \setminus u_b$ **do**
16            $E'_t \leftarrow \{(\bar{v}, v') \in E(v') \setminus (v, v') \mid \bar{v} \in \hat{u}.C'\}$;
17   **if** $E'_t = \varnothing$ **then**
18              $\lfloor$ c-add $\leftarrow$ *False*, **Break** ;
19   **else** $E_t \leftarrow E_t \cup E'_t$;
20   **if** *c-add = True* **then**
21            $\lfloor$ $S' \leftarrow S' \cup E_t, u.C' \leftarrow u.C' \cup \{v'\}$;
22   **if** $u.C' \leftarrow \varnothing$ **then** i-add $\leftarrow$ false, **Break**;
23   **if** *i-add = true* **then**
24       $S \leftarrow S \cup S'$, **for each** $u \in \pi$ **do** $u.C \leftarrow u.C'$;
25       $G_c = (V_c, E_c) \leftarrow$ Induce graph from G using S' ;
26       $E_{CM} \leftarrow E_{CM} \cup \{(c, \hat{v}) \mid \hat{v} \in V_c \setminus c \wedge \psi(\hat{v}) = \psi_q(v_t)\}$;
27   $G_{CM} = (V_{CM}, E_{CM}) \leftarrow$ Induce graph using E_{CM} ;
28   Delete vertices not include in $v_t.C$ from G_{CM} ;
29   Remove the connected subgraphs whose number of vertices is smaller than k from G_{CM};
30   $G = (V_G, E_G) \leftarrow$ Induce graph from G using S ;
31   Delete vertices $\{v \in V_G \mid \psi(v) = \psi_q(v_t)\} \setminus V_{CM}$ from G ;
   // Lines 32-40: Backward candidate refinement
32   $u_f \leftarrow$ last vertex in π ;
33   **for** *each vertices* $v \in u_f.C \cap V_{G'}$ **do**
34     $S' \leftarrow \varnothing, v_f.C' \leftarrow \{c\}$, i-add $\leftarrow$ True ;
35   **for each** *query vertex* $u \in \pi \setminus u_f$ *in reverse order* **do**
36   Same as Lines 8 ;
37       $N'_u \leftarrow \{u' \in N(u) \mid idx_u(\pi) > idx_{u'}(\pi)\}$;
38   Same as Lines 10-22 ;
39   Same as Lines 23-24 ;
40   Delete vertices not include in $v_t.C$ from G_{CM} ;
41   Same as Line 27-31 ;
42   **Return** G_{CM}, G ;

Example 4. *Consider the motif q in Figure 3a and the HIN in Figure 3b. Firstly, we get the BFS order $\{u_1, u_2, u_3, u_4\}$ of q and the candidate target vertex instances $C = \{v_1, v_2, v_3\}$. Next we explore the candidate region around each candidate target vertex instance in C.*

For v_1, (1) in the forward processing, we first find the candidate vertex of u_2 with $N'_u = \{u_1\}$ and $u_b = u_1$. Then we process each vertex v' in $\{v_6, v_8\}$. Note that although v_5 is the neighbor of v_1 with the same label as u_2, it is pruned because it does not pass the NLF filter. Due to the $N'_u \setminus u_b$ is empty, we conduct that v_6, v_8 are candidate vertex of u_2 and add the edges $(v_1, v_6), (v_1, v_8)$ in E_t to the S'. Next, we find that v_7, v_9 are the candidate vertex of u_2, but v_9 will be pruned because it does not satisfy the condition 1 of Corollary 1, i.e., there is no candidate vertex of u_1 in the neighbors of v_9. Then we add $(v_6, v_7), (v_8, v_7), (v_1, v_7)$ into S'. Finally we find v_{13} as the candidate vertex of u_4 and add $(v_{13}, v_7), (v_{13}, v_8)$ into S'. The candidate region around v_1 is the subgraph induced by edges $S' =$

(a) An example of HIN
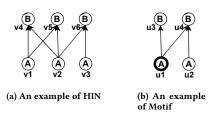
(b) An example of Motif

Figure 4: An example of message-passing strategy

$\{(v_1, v_6), (v_1, v_8), (v_6, v_7), (v_8, v_7), (v_1, v_7), (v_{13}, v_7), (v_{13}, v_8)\}$ and the candidate vertex C' of each query vertex is $u_1 = \{v_1\}$, $u_2 = \{v_6, v_8\}$, $u_3 = \{v_7\}$ and $u_4 = \{v_{13}\}$.

(2) In the backward processing, we first process the candidate vertex of u_3, i.e., v_7, with $N'_u = \{u_4\}$ and $u_b = u_4$. Then we find the edges $\{(v_{13}, v_7)\}$ connected with v_7 and candidate vertices of u_b i.e., { v_{13} }, and add them to E'_t. Next we process the u_2's candidate vertices $\{v_6, v_8\}$. We pruned v_6 because it does not pass the NLF filter, so we add edge $\{(v_{13}, v_8), (v_8, v_7)\}$ into E'_t. Finally we process the u_1's candidate v_1 and add the edges $\{(v_1, v_7), (v_1, v_8)\}$ to E'. The candidate region around v_1 after backward refinement is the subgraph induced by edges $S' = \{(v_1, v_7), (v_1, v_8), (v_8, v_7), (v_{13}, v_7), (v_{13}, v_8)\}$.

The time complexity of Algorithm 2 consists of the following steps. In the forward candidate exploration process, we need to take $O(|E_G| \cdot d_q(u))$ to check the candidate vertices around query vertex u, where $d_q(u)$ is the degree of vertex u in motif q. Thus the total running time of the forward candidate exploration process is $O(\sum_{u \in V_q} |E_G| \cdot d_q(u))$, i.e., $O(|E_G| \cdot |E_q|)$. In backward candidate refinement, we have the same process to check the candidate vertices around each query vertex u in motif q. So the total complexity of Algorithm 2 is $O(|E_G| \cdot |E_q|)$.

4.2 Message-passing based Optimization Strategy

By reviewing the process of filter-verify solution, we can see that all the vertices of the target type need to re-enumerate a motif instance around them in each iteration to check whether they are in M-graph. For example, considering the motif in Figure 4b and the HIN in Figure 4a, if we remove v_3 by NLF filter, we need to re-enumerate a motif around v_1, v_2 in the next iteration. However, v_1 do not need to re-enumerate after removing v_3 because the deleting of v_3 does not affect the structure around v_1. To solve this challenge, we propose a message-passing based strategy to avoid unnecessary re-enumeration.

The key idea of our strategy is that the target vertex instance does not need to re-enumerate a motif around it if none of its M-neighbors is removed. Recall that the candidate target vertex instances and their candidate M-neighbors have been stored in the CM-graph. Thus, if a candidate target vertex instance is removed from CM-graph, we can send a message to its in-neighbors in CM-graph to re-identify whether there exists a motif instance around them in the next iteration.

The detailed procedure is presented in Algorithm 3. We first initialize V_e to store the candidate vertex instances that need to check whether it has a motif instance around it (Line 1), then we

---

Algorithm 3: Message-Passing(G, q, G_{CM})

Input: An HIN $G = (V_G, E_G)$, a motif $q = (V_q, E_q, v_t)$ with vertex type mapping function ψ_q and a CM graph $G_{CM} = (V_{CM}, E_{CM})$
Output: A list of individual fairness community $maxC$

1   $V_e \leftarrow$ vertices with type $\psi_q(v_t)$ in G ;
2   **while** $V_e \neq \varnothing$ **do**
3     $V'_e \leftarrow \varnothing$;
4   **for each** *vertex v_e in V_e* **do**
5   **if** *no motif instance around v_e found by an existing subgraph isomorphism algorithm from G'* **then**
6         $V'_e \leftarrow V'_e \cup \{$in-neighbors of v_e in $G_{CM}\}$;
7   Delete v_e from G_{CM} and G ;
8     $V_e \leftarrow V'_e$
9   **Return** G;

---

iteratively check the instances in V_e until V_e is empty (Lines 2-8). We first initialize V'_e to record the candidate target vertices whose candidate M-neighbors are removed (Line 3). For each vertex instance v_e in V_e, once a motif instance around v_e is enumerated by an existing subgraph isomorphism, we immediately check the next vertex instance. Otherwise, we add the in-neighbors of v_e into V'_e and delete v_e from G_{CM} and G (Lines 4-7). Once the checking process is finished, if V'_e is not empty, we continue to check the candidate vertex in V'_e in the next iteration (Line 8). Finally, we return the HIN G (Lines 9).

4.3 The Lower Bound of Fairness Score

In this subsection, we propose a lower bound of fairness score to filter the unfair communities in advance. The key idea is that if we have calculated the active levels of some M-connected target vertex instances and know the lower bound of the fairness score is not lower than the fairness score of the existing fairest communities, we immediately know that the M-connected target vertex instances are not fairest and do not need to enumerate motif instances around the rest M-neighbor target vertex instances. Before providing a detailed proof of the lower bound, we provide an equivalent formula of the Equation (1) proved in [33].

$$FS = \frac{2\left(s_1 + 2s_2 + \ldots + |S|s_{|S|}\right)}{|S| \sum_{j=1}^{|S|} s_j} - \frac{|S| + 1}{|S|} \qquad (2)$$

subject to

$$s_1 < s_2 < \ldots < s_{|S|}$$

which can be transferred into the following form:

$$FS = \frac{s_1 + 3s_2 + \ldots + (2|S| - 1)s_{|S|}}{|S| \sum_{j=1}^{|S|} s_j} - 1 \qquad (3)$$

subject to

$$s_1 < s_2 < \ldots < s_{|S|}$$

Based on the Equation (3), we can get the following property.

Property 1. *Given a candidate target-aware community C and a list of active levels S' of n M-connected target vertex instances C' in C, the active levels S'' of each target vertex instance in $C \setminus C'$ should be smaller than the maximum value S'_{max} of S' if we intend to minimize the fairness score FS of C.*

PROOF. We prove the correctness of this property by contradiction. Assume $S = S' \cup S''$ has been sorted in ascending order. Through using the Equation 3, fairness score of C can be written as $FS = \frac{\sum_{i=1}^{|C|}(2i-1)s_i}{|C|\sum_{i=1}^{|C|}s_i} - 1$, where $s_i \in S$. It's a multivariate equation where the elements in S'' are independent variables. The first-order partial derivatives of FS is $\frac{\partial FS}{\partial s_p} = \frac{(2p-1)B-A}{(s_p+B)^2}$, where $s_p \in S''$, A, B are constants and $A = \left(\sum_{j=1}^{|C|}(2j-1)s_j\right) - (2p-1)s_p$, $B = (\sum_{j=1}^{|C|} s_j) - s_p$. In this case, if $s_{|C|} > S'_{max}$, $s_{|C|}$ must in S'', and the first-order partial derivatives of FS w.r.t $s_{|C|}$ is positive because $(2|C|-1)B > A$. Therefore, $s_{|C|}$ must be less than or equal to S'_{max} if we want to minimise FS because FS will become larger as $s_{|C|}$ becomes larger when $s_{|C|}$ is greater than S'_{max}. Through the above inference, we can determine that other active levels in S'' must be less than or equal to S'_{max} iteratively.   □

In addition, we proved the following property to support the lower bound of the fairness score.

Property 2. *Given a list of numbers $L = l_1, \ldots, l_m$, each number in $X = x_1, \ldots, x_n$ should be equal to the median value of L if we intend to minimize $z = \sum_{i=1}^{|S|}\sum_{j=1}^{|S|}|s_i - s_j|$, where $S = L \cup X$ and $s_i, s_j \in S$.*

PROOF. We rewrite z to $z = (2\sum_{j=1}^{|L|}\sum_{i=1}^{|X|}|l_j-x_i|)+\sum_{i=1}^{|X|}\sum_{j=1}^{|X|}|x_i-x_j|$. As proved in [26], $\sum_{j=1}^{|L|}|l_j - x|$ is minimal if x is equal to the median of S. Thus, $(2\sum_{j=1}^{|L|}\sum_{i=1}^{|X|}|l_j - x_i|)$ can be minimized when $x_i \in X$ is the median of L. In addition, $\sum_{j=1}^{|X|}|x_i - x_j|$ can be minimized when $x_i \in X$ are the same. Thus, we can conclude z can be minimized when each number in X equals the median value of L.   □

Now, we show a lower bound of the fairness score in Property 3 based on Property 1, 2 and Equation 1.

Property 3. *Given a candidate target-aware community C and a list of active levels $S' = \{s_1 \ldots s_m\}$ of m M-connected target vertex instances C' in C, the lower bound FS^{LB} of the fairness score can be calculated as:*

$$FS^{LB} = \frac{\sum_{i=1}^{|S|}\sum_{j=1}^{|S|}|s_i - s_j|}{2|C|\left(\sum_{m=1}^{|S'|}s_m + (|S| - |S'|)S'_{max}\right)} \quad (4)$$

where S contains the active levels in S' and $|C| - |S'|$ median value of S', $s_i, s_j \in S$ and $s_m \in S'$.

PROOF. We get the lower bound of FS by minimize the numerator and maximize the denominator of Equation 1 separately. Based the property 2, we can get the minimum value of numerator $\sum_{j=1}^{|S|}|s_i - s_j|$ when the active levels of target vertex instances $C \setminus C'$ equal to the medium value of S'. To get the maximum value of the denominator $2|S|\left(\sum_{=1}^{|S|}s_m\right)$, we need to get the maximum value of $|S|$ and $\sum_{=1}^{|S|}s_m$. Recall that the target-aware community is contained in the candidate candidate community, thus the maximum value of $|S|$ are the size of candidate community $|C|$. As discussed in Property 1, $\sum_{=1}^{|S|}s_m$ have the maximum value when the active levels

of target vertex instances $C \setminus C'$ are equal to the maximum value of S'. Based on the maximum and minimum value of the denominator and numerator, we can get the lower bound of fairness score.   □

4.4   The optimization Algorithm

Algorithm 4 presents the optimization algorithm by reducing potential target vertices and utilizing message-passing based strategy and the lower bound of fairness score. Initially, we remove the ineligible vertices of the target type using Algorithm 2 (Line 1) and find the vertices in M-graph using Algorithm 3 (Line 2). Then we initialize $D, maxC, FS_m$ to store active levels, the fairest communities and their fairness score (Line 3). Next, we search the target-aware community in each candidate target-aware community g, i.e., each weakly connected subgraph of G_{CM} using the lower bound of fairness score (Lines 4-35). For each candidate target-aware community, we first initialize Vst, IV, UV, VIV to store the visited M-connected target vertex instances, in-neighbors of visited target vertex instances, unvisited M-connected vertices of visited target vertex instances and visited target vertex instances but not M-connected by target vertex instances in Vst (Line 5). Then we randomly add a vertex from g to UV (Line 6), and start to find the target-aware community by visiting the vertex in UV(Lines 7-35). For each vertex v in UV, we add it to Vst, enumerate the motif instances around it to calculate its active level, and add the M-neighbors of v which is not visited to UV (Lines 8-12). Based on the active levels of visited target vertex instances, we calculate the lower bound of the fairness score. If the lower bound is higher than the fairness score of existing fairest communities, we remove this candidate target-aware community; else, we continue to implement the above process (Lines 13-15). If UV is empty, we consider whether the in-neighbors OV of vertices in Vst but not contained in Vst and the vertices in Vst are M-connected (Lines 16-35). We first get the M-neighbors of each vertex $\hat{v}$ in OV (Line 17). If $\hat{v}$ is contained in VIV, we can get the M-neighbors of $\hat{v}$ by collecting the out-neighbors of $\hat{v}$ in g (Lines 33-35); else, we get its M-neighbors and active level by enumerating the motif instances around $\hat{v}$ (Lines 18-26). If M-neighbors of $\hat{v}$ are contained in Vst, we know that $\hat{v}$ is M-connected to the vertices in Vst (Line 27-29); if not, we record the M-neighbors and active level of $\hat{v}$ (Lines 30-32). We repeat the above process until UV and OV are empty. Finally, we return the fairest communities stored in $MaxC$.

The time complexity of Algorithm 4 is $O(d^{\frac{3}{2}d_t} \cdot V_t^2 + |V_t|^{|V_q|} + |E_G| \cdot |E_q|)$ in total. But the exploration-based filter and message-passing based optimization strategy help to filter out ineligible vertices in V_t, and the lower bound based pruning rules filter the unfair communities so as to improve the efficiency.

5   EXPERIMENTS

5.1   Experimental Setup

Dataset. We performed extensive experiments on four real-world HIN datasets: IMDB[1], DBLP[1], Freebase[1], and Amazon[1]. Their statistics, such as the number of vertices, edges, vertex types, average degree of vertices and number of distinct motifs, are presented in

---

Algorithm 4: The Optimization Algorithm

Input: An HIN $G = (V_G, E_G)$, a motif $q = (V_q, E_q, v_t)$ with vertex type mapping function ψ_q

Output: A list of individual fairness communities $maxC$

1   $G_{CM} \leftarrow$ Exploration-based Filter(G, q) ;
2   $G \leftarrow$ Message-Passing(G, q, G_{CM}) ;
3   $D \leftarrow$ empty dictionary, $FS_m = 1$, $maxC = []$;
4   **for each** *weakly connected graph* $g = (V_g, E_g)$ *of* G_{CM} *in ascending order of* $|g|$ **do**
5     $Vst \leftarrow \varnothing$, $i_f \leftarrow$ False, $IV \leftarrow \varnothing$, $UV \leftarrow \varnothing$ $VIV \leftarrow \varnothing$;
6     $UV \leftarrow$ Random select a vertex from $V_g \setminus Vst$;
7   **while** $UV \neq \varnothing$ **do**
8   Random pop a vertex v from UV to Vst ;
9       $IV \leftarrow IV \cup \{$ in-neighbors of v in g $\}$;
10   **for each** *instance* $g_m^q = (V_m^q, E_m^q)$ *of motif* q *around* v *in* G **do**
11   Same as Line 9-10 of Algorithm 1 ;
12         $UV \leftarrow UV \cup \{v' \in V_m^q \mid \psi(v') = \psi_q(v_t) \wedge v' \notin Vst\}$;
13   **if** $i_f =$ *False* **then**
14         $FS^{LB} \leftarrow$ calculate the lower bound using active levels of Vst ;
15   **if** $FS^{LB} > FS_m$ **then** $i_f \leftarrow$ True ;
16   **if** $UV = \varnothing$ **then**
17         $OV \leftarrow IV \setminus Vst$, $IV \leftarrow \varnothing$;
18   **if** $OV = \varnothing$ **then**
19   Same as Line 19-24 of Algorithm 1 ;
20   **else**
21   **for each** *vertex* $\hat{v} \in OV$ **do**
22   **if** $\hat{v} \notin VIV$ **then**
23               $MN \leftarrow \varnothing$;
24   **for each** *instance* $g_m^q = (V_m^q, E_m^q)$ *of motif* q *around* $\hat{v}$ *in* G **do**
25   Same as Line 9-10 of Algorithm 1 ;
26                 $MN \leftarrow MN \cup \{v' \in V_m^q \mid \psi(v') = \psi_q(v_t)\}$;
27   **if** $MN \cap Vst \neq \varnothing$ **then**
28               $UV \leftarrow UV \cup (MN \setminus Vst)$;
29               $Vst \leftarrow Vst \cup \{\hat{v}\}$;
30   **else**
31               $VIV \leftarrow VIV \cup \{\hat{v}\}$;
32   Delete out-edges of $\hat{v}$ in g and add out-edges between $\hat{v}$ and vertices in MN to g ;
33   **else**
34             $MN \leftarrow$ out-neighbors of $\hat{v}$ in g ;
35   Same as Line 27-29 ;

36   **Return** $maxC$;

Table 2: Dataset Statistics

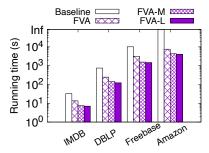| Dataset	Vertices	Edges	Vertex types	motifs
IMDB	11,616	34,212	3	6
DBLP	26,128	239,566	4	7
Freebase	180,098	1,057,688	8	16
Amazon	1,569,960	264,339,468	107	76

Figure 5: Time cost of Target-aware Community Search on Four Datasets under Default Parameter Setting

a query as infinite (**Inf**) if the query set cannot be finished in 24 hours.

Algorithms. We evaluate four algorithms in our experiments, namely Baseline, FVA, FVA-M, FVA-L. Baseline is the filter-verify solution discussed in Section 3. FVA is the filter-verify algorithm with the reducing potential target vertices strategy. FVA-M considers the reducing potential target vertices strategy and message-passing based optimization strategy. FVA-L utilize the reducing potential target vertices strategy, message-passing based strategy and the lower-bound to search community, i.e., the Algorithm 4.

Environment. All the experiments are implemented in Python 3.7 programming language and are conducted on a Linux system that has an Intel Core i5 CPU @ 2GHz and 8GB of memory. For subgraph isomorphism, we use Grand-Iso [24], which is a state-of-the-art subgraph isomorphism algorithm.

5.2 Evaluation of Efficiency

In this subsection, we present the performance of our proposed three algorithms compared with the baseline solution. Figure 5 demonstrates the time cost of four methods over four datasets under the default parameter settings. Clearly, FVA-L runs much faster than the other three methods for every dataset. In specific, FVA-L reduces the time cost by ×2.41, ×3.18, and ×11.15 compared to the baseline method for IMDB, DBLP, Freebase, and Amazon dataset, respectively.

To show the impact of each parameter, we also evaluate the efficiency of the proposed algorithms over four datasets by varying the motif size $|V_q|$.

Varying motif size $|V_q|$. Figure 6 shows the average time cost of the four algorithms when motif size $|V_q|$ varies from 3 to 7. We obverse all the algorithms consume higher time costs when the motif size increases. This is because the cost of active level calculation (i.e., motif enumeration) increases with the motif size. In addition, the three proposed algorithms consume significantly

---

Table 2. IMDB is an online dataset of movies and television programs, which consists of three types of vertices (movies, directors, and actors). DBLP is a website for computer science bibliography, which has four vertex types containing authors, papers, terms, and publication venues after data preprocessing and extraction. Freebase is a huge collaborative knowledge graph, which contains 8 genres of entities. Amazon is a co-purchase graph. Its nodes represent goods and edges indicate that two goods are frequently bought together.

Parameters. We randomly create motifs to test different situations and reported the average running time, space cost and effectiveness metrics. In particular, we first generate a small HIN by conducting a random walk on the data graph following [2]. After that, we randomly choose a vertex in the small HIN as the target vertex and select the small HIN as a motif if it has at least two vertices with the type of target vertex. For performance evaluation, we create motifs with varying sizes from 3 to 7 (default size is 5) because the size of the motifs is bounded from 3 to 7 in real applications [14, 19, 25]. We randomly create 5 motif sets, each of which contains 100 motifs of the same size. We treat the running time of

Figure 6: Efficiency evaluation over motif size $|V_q|$ on four datasets

Figure 7: Time cost of four methods with different sampling ratios on four datasets

Figure 8: Effectiveness Analysis

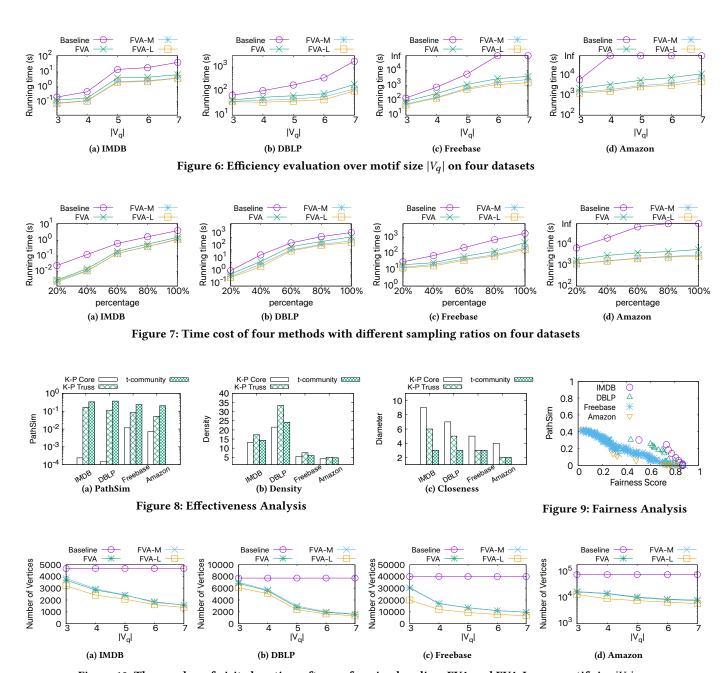Figure 9: Fairness Analysis

Figure 10: The number of visited vertices after performing baseling, FVA and FVA-L over motif size $|V_q|$

less time than the baseline method and consume decreasing time in a similar trend as the results under the default parameter settings. Thus, we conclude that our proposed methods effectively enhance the performance of target-aware community search under different motif sizes.

Memory overheads analysis. Table 3 shows the total memory overheads of the methods on four datasets. Note that the memory overheads of algorithms do not contain the memory overhead of the subgraph isomorphism algorithm. Obviously, the memory

overheads of FVA, FVA-M, and FVA-L are significantly higher than that of baseline. This is because FVA, FVA-M, and FVA-L store the candidate vertices of each query vertex in the exploration-based filter stage. Additionally, we also see the memory overheads of FVA, FVA-M, and FVA-L are almost the same. This is because the message-passing based postponing enumeration search and lower bound-based approach changed the method of searching, which does not incur additional memory overhead. Furthermore, we can see the memory overheads of same method in Amazon dataset

Table 3: Evaluation of memory overheads (MB)

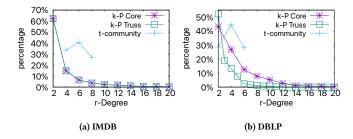| Dataset	Baseline	FVA	FVA-M	FVA-L
IMDB	4.15	11.10	11.11	11.13
DBLP	8.22	34.41	34.44	34.43
Freebase	7.16	27.38	27.41	27.47
Amazon	4.71	9.31	9.32	9.32

(a) IMDB　　　　**(b) DBLP**

Figure 11: r-degree distribution of communities

is significant lower than other three datasets. This is because the number of motif instances and target-aware community stored in memory is less than other three datasets.

Efficiency of pruning strategies. Figure 10 evaluates the pruning efficiency of Baseline, FVA, FVA-M and FVA-L by comparing the number of remaining vertices that need to calculate their active levels on four datasets with varying $|V_q|$. As can be seen from Figure 10, FVA, FVA-M, and FVA-L can significantly reduce the number of vertices compared to the baseline method as expected. Moreover, the number of remaining vertices decreases as $|V_q|$ increases. For instance, in the IMDB dataset, FVA reduces the number of vertices from 4670 to 1592; FVA-M reduces the number of vertices from 3683 to 1587; and FVA-L further reduces the number of vertices from 3198 to 1372. Compared with the baseline method, FVA and FVA-M have nearly same number of remaining vertices. These results confirm that the pruning effect of FVA and FVA-M mainly comes from the exploration-based filter. In addition, we can find that FVA-L substantially reduces the number of vertices compared to FVA. This is because FVA-L not only prunes the unpromising vertices in the exploration-based filter but also prunes the vertices of the candidate target-aware communities that do not pass the lower bound of fairness score.

5.3 Evaluation of Scalability

We evaluate the scalability of four proposed algorithms over datasets IMDB, DBLP, Freebase and Amazon. For each dataset, we generate four small datasets with different sizes by randomly sampling 20%, 40%, 60%, and 80% vertices from the datasets, respectively. Note that the dataset itself is considered with the 100% data size. Figure 7 shows the time cost of four algorithms on the size-varying datasets. With the increase of the dataset size, we can observe that the running time of Baseline, FVA, FVA-M and FVA-L has a linear increasing trend. This implies that our proposed three algorithms are easily applied to large-scale networks. However, the baseline method can not be finished within 24 hours in large datasets (i.e.,

Amazon and Freebase). Thus, we conclude that the baseline solution has limitations in scalability.

5.4 Evaluation of Effectiveness

To show the effectiveness of community search in HIN, we compare our fairest target-aware community (t-community) with the k-$\mathcal{P}$ Core [8] and k-$\mathcal{P}$ Truss [31]. To achieve this, we generate five symmetric meta-paths $\mathcal{P}$ discussed in [8] that are the sequences of vertex types between two given target vertex types, and can be seen as the motifs in our work. We calculate the quality metrics of each k-$\mathcal{P}$ Core and k-$\mathcal{P}$ Truss, and report the highest metrics as result. We utilize the following metrics to analyze the quality of communities.

Relational Degree of Community Members. Conventionally, the degree of a vertex is the number of edges connecting it. To adapt it for communities in HINs, we redefine it as the number of M-neighbors of a vertex and call it relational degree (r-degree). For each community, we count the percentages of vertices whose r-degree varies from 1 to 20. Due to the space limitation, we only report the average percentage values on IMDB and DBLP datasets in Figures 11. Clearly, compared to k-$\mathcal{P}$ Core and k-$\mathcal{P}$ Truss, the variances of the r-degree of target-aware community are smaller. Thus, the engagement between vertices in t-communities is more similar than the engagement between vertices in k-$\mathcal{P}$ Core and k-$\mathcal{P}$ Truss.

Similarity of Community Members. We measure the similarity of community members by using PathSim [29]. Specifically, we first find communities of k-P core, k-P Truss and target-aware community, then compute the PathSim value for each pair of vertices in these communities. Figure 8a shows the average PathSim values on four datasets. Clearly, target-aware communities achieve higher similarity values than those of k-P core and k-P Truss, so their members are more similar to each other.

Density of link relationships. To measure the density of link relationships, we extend the traditional density and redefine it as the number of vertex pairs that are connected by meta-path over the number of vertices in community. The average densities for communities of each community model are depicted in Figure 8b. We observe that the densities of t-communities are higher than the k-P core but lower than the k-P Truss. Thus, the target-aware community model achieves stronger cohesiveness than the k-P core model but less cohesiveness than the k-P Truss model.

Closeness of Community. To measure the closeness of communities, a commonly-used metric is the diameter [22], which is the largest shortest distance between any pair of vertices in the community. To adapt it for communities in HINs, we redefine the distance as motif-constrained distance, i.e., M-distance; that is, the M-distance between two target vertex instances linked by an instance of the motif is 1. In our experiment, we first calculate the shortest M-distance of each pair of target vertex instances in these communities and report the largest distance in Figure 8c. Clearly, the t-communities have smaller diameter than k-P cores and k-P Trusses on all datasets, which means that the community members tend to have closer relationships.

Evaluation of Fairness. We conduct the study on the relationship between fairness scores and the similarity between members

in t-communities. Specifically, we first find t-communities on each dataset, then compute the fairness score of each t-communiy and the PathSim value for each pair of vertices in each community. Figure 9 shows the average PathSim values against the fairness scores of t-communities on four datasets. Clearly, the average PathSim consumes higher when the fairness score decreases. For instance, we can find 14 t-communities in DBLP, the fairness scores are 0.16, 0.19, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3, 0.33, 0.36, 0.37, 0.38, 0.4, 0.56, and the average PathSim values are 0.36, 0.25, 0.23, 0.22, 0.20, 0.18, 0.18, 0.16, 0.14, 0.06, 0.05, 0.04, 0.04, 0.005, respectively. Thus, we conclude that the community members will be more similar if the target-aware community has a higher fairness score.

6 RELATED WORK

Community search: Community search aims to query cohesive subgraphs that satisfy the customized query request. To measure cohesiveness of a subgraph, existing works develop different community models such as k-core [1, 30], k-truss [5], k-clique [20], k-edge-connected component [4], and the k-plex [6]. However, these works focus on searching communities over homogeneous graphs, which cannot be directly used in a heterogeneous network because the relation between vertex types is different. Recently, researchers attempted to find cohesive communities from HINs. For instance, [8, 14–16, 31] utilize different customized query requests such as meta-path [8, 16, 31], relational constraint [15] and motif [14] to extend traditional community models. However, these studies did not consider the notion of fairness, which may lead systematic discrimination for disadvantaged people in communities.

Fairness-aware Community Mining: The notion of fairness graph retrieval has received much attention in recent years. It is used to alleviate the bias problem caused by the tendency of retrieval. For example, [19] proposed a fair spectral clustering algorithm to generate communities, which ensures that each cluster contains roughly the same number of group elements. [17, 32] provided a parity-based fairness measurement to distinguish the differences in behavior between dominant and disadvantaged users. [10] proposed a heuristic reordering based fairness algorithm to reduce the influence of active users' history on inactive users' recommendations. [23] refined the attributes that need to be considered fairly. It applied the fairness measures to user-defined attributes and allowed other attributes to bias in the recommendation. However, the current works focus on how to make unbiased recommendations through keeping the similarity of certain metrics between groups, which can not make sure the fairness in fine granularity level (e.g., keep fairness between members in a group).

7 CONCLUSIONS

In this paper, we first discussed the necessity of individual fairness in community search problem, then formalized the novel problem of individual fairest community search, which considers the similarity of active level between members in a community. To model the relationships between members of communities and customized query requests of users, we extended the well-known concept of motif. To tackle this problem, we first proposed an filter-verify algorithm, then we propose an exploration-based filter strategy to reduce the potential target vertices. Based on the filter, we developed a

message-passing based postponing enumeration search method to reduce redundant computation. We further boosted the query efficiency by identifying and pruning the unfair community in advance during the process of community search. Our experimental results on four real HINs show the efficiency of our proposed filter and algorithms, and the effectiveness of our proposed community.

REFERENCES

[1] Vladimir Batagelj and Matjaz Zaversnik. 2003. An O (m) algorithm for cores decomposition of networks. *arXiv preprint cs/0310049* (2003).

[2] Fei Bi, Lijun Chang, Xuemin Lin, Lu Qin, and Wenjie Zhang. 2016. Efficient subgraph matching by postponing cartesian products. In *Proceedings of the 2016 International Conference on Management of Data*. 1199–1214.

[3] Taotao Cai, Jianxin Li, Nur Al Hasan Haldar, Ajmal Mian, John Yearwood, and Timos Sellis. 2020. Anchored vertex exploration for community engagement in social networks. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 409–420.

[4] Lijun Chang, Jeffrey Xu Yu, Lu Qin, Xuemin Lin, Chengfei Liu, and Weifa Liang. 2013. Efficiently computing k-edge connected components via graph decomposition. In *Proceedings of the 2013 ACM SIGMOD international conference on management of data*. 205–216.

[5] Jonathan Cohen. 2008. Trusses: Cohesive subgraphs for social network analysis. *National security agency technical report* 16, 3.1 (2008), 1–29.

[6] Alessio Conte, Donatella Firmani, Caterina Mordente, Maurizio Patrignani, and Riccardo Torlone. 2017. Fast enumeration of large k-plexes. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 115–124.

[7] Marcus T. Dittrich, Gunnar W. Klau, Andreas Rosenwald, Thomas Dandekar, and Tobias Müller. 2008. Identifying functional modules in protein-protein interaction networks: an integrated exact approach. In *ISMB*. 223–231.

[8] Yixiang Fang, Yixing Yang, Wenjie Zhang, Xuemin Lin, and Xin Cao. 2020. Effective and efficient community search over large heterogeneous information networks. *Proceedings of the VLDB Endowment* 13, 6 (2020), 854–867.

[9] Uriel Feige. 2004. Approximating Maximum Clique by Removing Subgraphs. *SIAM J. Discret. Math.* 18, 2 (2004), 219–225.

[10] Zuohui Fu, Yikun Xian, Ruoyuan Gao, Jieyu Zhao, Qiaoying Huang, Yingqiang Ge, Shuyuan Xu, Shijie Geng, Chirag Shah, Yongfeng Zhang, et al. 2020. Fairness-aware explainable recommendation over knowledge graphs. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 69–78.

[11] Corrado Gini. 1921. Measurement of inequality of incomes. *The economic journal* 31, 121 (1921), 124–126.

[12] Myoungji Han, Hyunjoon Kim, Geonmo Gu, Kunsoo Park, and Wook-Shin Han. 2019. Efficient Subgraph Matching: Harmonizing Dynamic Programming, Adaptive Matching Order, and Failing Set Together. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, Peter A. Boncz, Stefan Manegold, Anastasia Ailamaki, Amol Deshpande, and Tim Kraska (Eds.). ACM, 1429–1446. https://doi.org/10.1145/3299869.3319880

[13] Juris Hartmanis. 1982. Computers and intractability: a guide to the theory of np-completeness (michael r. garey and david s. johnson). *Siam Review* 24, 1 (1982), 90.

[14] Jiafeng Hu, Reynold Cheng, Kevin Chen-Chuan Chang, Aravind Sankar, Yixiang Fang, and Brian YH Lam. 2019. Discovering maximal motif cliques in large heterogeneous information networks. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 746–757.

[15] Xun Jian, Yue Wang, and Lei Chen. 2020. Effective and Efficient Relational Community Detection and Search in Large Dynamic Heterogeneous Information Networks. *Proc. VLDB Endow.* 13, 10 (2020), 1723–1736.

[16] Yangqin Jiang, Yixiang Fang, Chenhao Ma, Xin Cao, and Chunshan Li. 2022. Effective Community Search over Large Star-Schema Heterogeneous Information Networks. *Proc. VLDB Endow.* 15, 11 (2022), 2307–2320. https://www.vldb.org/pvldb/vol15/p2307-jiang.pdf

[17] Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. 2012. Enhancement of the Neutrality in Recommendation.. In *Decisions@ RecSys*. Citeseer, 8–14.

[18] Jian Kang, Jingrui He, Ross Maciejewski, and Hanghang Tong. 2020. Inform: Individual fairness on graph mining. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 379–389.

[19] Matthäus Kleindessner, Samira Samadi, Pranjal Awasthi, and Jamie Morgenstern. 2019. Guarantees for spectral clustering with fairness constraints. In *International Conference on Machine Learning*. PMLR, 3458–3467.

[20] Jussi M Kumpula, Mikko Kivelä, Kimmo Kaski, and Jari Saramäki. 2008. Sequential algorithm for fast clique percolation. *Physical review E* 78, 2 (2008), 026109.

[21] Jianxin Li, Xinjue Wang, Ke Deng, Xiaochun Yang, Timos Sellis, and Jeffrey Xu Yu. 2017. Most influential community search over large social networks. In *2017 IEEE 33rd international conference on data engineering (ICDE)*. IEEE, 871–882.

[22] Jianxin Li, Xinjue Wang, Ke Deng, Xiaochun Yang, Timos Sellis, and Jeffrey Xu Yu. 2017. Most Influential Community Search over Large Social Networks. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. 871–882. https://doi.org/10.1109/ICDE.2017.136

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

[24] Jordan K. Matelsky, Elizabeth P. Reilly, Erik C. Johnson, Jennifer Stiso, Danielle S. Bassett, Brock A. Wester, and William Gray-Roncal. 2021. DotMotif: an open-source tool for connectome subgraph isomorphism search and graph queries. *Scientific Reports* 11, 1 (Jun 2021). https://doi.org/10.1038/s41598-021-91025-5

[25] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. 2002. Network motifs: simple building blocks of complex networks. *Science* 298, 5594 (2002), 824–827.

[26] Sunny Garlang Noah. 2007. The median of a continuous function. (2007).

[27] Shixuan Sun and Qiong Luo. 2022. Subgraph Matching With Effective Matching Order and Indexing. *IEEE Trans. Knowl. Data Eng.* 34, 1 (2022), 491–505.

[28] Sahil Verma and Julia Rubin. 2018. Fairness definitions explained. In *2018 ieee/acm international workshop on software fairness (fairware)*. IEEE, 1–7.

[29] Yue Wang, Zhe Wang, Ziyuan Zhao, Zijian Li, Xun Jian, Lei Chen, and Jianchun Song. 2020. HowSim: A General and Effective Similarity Measure on Heterogeneous Information Networks. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 1954–1957.

[30] Cynthia I Wood and Illya V Hicks. 2015. The minimal k-core problem for modeling k-assemblies. *The Journal of Mathematical Neuroscience (JMN)* 5, 1 (2015), 1–19.

[31] Yixing Yang, Yixiang Fang, Xuemin Lin, and Wenjie Zhang. 2020. Effective and Efficient Truss Computation over Large Heterogeneous Information Networks. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. 901–912. https://doi.org/10.1109/ICDE48307.2020.00083

[32] Sirui Yao and Bert Huang. 2017. Beyond parity: Fairness objectives for collaborative filtering. *Advances in neural information processing systems* 30 (2017).

[33] Don Zagier. 1983. Inequalities for the Gini coefficient of composite populations. *Journal of Mathematical Economics* 12, 2 (1983), 103–118.

[34] Qi Zhang, Rong-Hua Li, Minjia Pan, Yongheng Dai, Qun Tian, and Guoren Wang. 2023. Fairness-aware Maximal Clique in Large Graphs: Concepts and Algorithms. *IEEE Transactions on Knowledge and Data Engineering* (2023).

[35] Feng Zhao and Anthony KH Tung. 2012. Large scale cohesive subgraphs discovery for social network visual analysis. *Proceedings of the VLDB Endowment* 6, 2 (2012), 85–96.

[36] Taige Zhao, Ningning Cui, Yunliang Chen, and Man Li. 2020. Efficient strategy mining for football social network. *Complexity* 2020 (2020), 1–11.

[37] Taige Zhao, Ningning Cui, Yunliang Chen, and Man Li. 2020. Efficient Strategy Mining for Football Social Network. *Complex.* 2020 (2020), 8823189:1–8823189:11.

[38] Taige Zhao, Xiangyu Song, Man Li, Jianxin Li, Wei Luo, and Imran Razzak. 2023. Distributed Optimization of Graph Convolutional Network Using Subgraph Variance. *IEEE Transactions on Neural Networks and Learning Systems* (2023).