2023 IEEE 39th International Conference on Data Engineering (ICDE)

Fairness-aware Maximal Biclique Enumeration on Bipartite Graphs

Ziqi Yin†, Qi Zhang†, Wentao Zhang‡, Rong-Hua Li†, Guoren Wang†
†*Beijing Institute of Technology, Beijing, China;* ‡*Peking university, Beijing, China*
ZIQI003@e.ntu.edu.sg; {qizhangcs,rhli,wanggr}@bit.edu.cn; wentao.zhang@pku.edu.cn

2023 IEEE 39th International Conference on Data Engineering (ICDE) | 979-8-3503-2227-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICDE55515.2023.00131

***Abstract*—Maximal biclique enumeration is a fundamental problem in bipartite graph data analysis. Existing biclique enumeration methods mainly focus on non-attributed bipartite graphs and also ignore the *fairness* of graph attributes. In this paper, we introduce the concept of fairness into the biclique model for the first time and study the problem of fairness-aware biclique enumeration. Specifically, we propose two fairness-aware biclique models, called single-side fair biclique and bi-side fair biclique respectively. To efficiently enumerate all single-side fair bicliques, we first present two non-trivial pruning techniques, called fair α-β core pruning and colorful fair α-β core pruning, to reduce the graph size without losing accuracy. Then, we develop a branch and bound algorithm, called FairBCEM, to enumerate all single-side fair bicliques on the reduced bipartite graph. To further improve the efficiency, we propose an efficient branch and bound algorithm with a carefully-designed combinatorial enumeration technique. Note that all of our techniques can also be extended to enumerate all bi-side fair bicliques. We also extend the two fairness-aware biclique models by constraining the ratio of the number of vertices of each attribute to the total number of vertices and present corresponding enumeration algorithms. Extensive experimental results on five large real-world datasets demonstrate our methods' efficiency, effectiveness, and scalability.**

I. Introduction

A bipartite graph $G(U, V, E)$ contains two disjoint vertex sets U and V and one edge set E in which each edge links a node in U and a node in V. Many real-world networks, such as online user-item networks [8], [28], [30], [35], [43] and gene co-expression networks [7], [9], [37], [41] can be modeled as bipartite graphs. Recently, the problems of analysis of bipartite graphs have attracted much attention due to numerous real-world applications, such as maximal biclique enumeration [1], [6], [17], [41], butterfly counting [24], [29], [31], [42], and maximum biclique search [5], [18], [22], [34].

In recent years, the concept of fairness has also been widely investigated in data analysis related areas [10], [11], [13], [27]. Many existing studies reveal that a biased machine learning model may result in discrimination upon a discrimination group, such as the gender bias and the racial bias [3], [16], [26], [39]. Various methods (e.g., group fairness and individual fairness [4], [25], [27], etc.) are proposed to tackle this problem. Despite their effectiveness in data analysis applications, the fairness in graph data analysis [21] is still under-explored. A notable example is that Pan *et al.* proposed two fairness-aware maximal clique models to find fair communities in attributed graphs [21]. Their models, however, are mainly tailored for traditional attributed graphs, and they cannot be directly generalized to other types of graphs, such as bipartite graphs studied in this paper.

In this work, we focus mainly on attributed bipartite graphs, motivated by the fact that many real-life graphs, such as online customer-product networks, can be modeled as attributed bipartite graphs. We introduce the concept of fairness into the classic biclique model and investigate the problem of mining fairness-aware bicliques on attributed bipartite graphs. Here a biclique is a subgraph of the bipartite graph in which every pair of nodes belonging to two different sides has an edge. Note that nodes at the upper side and lower side of the attributed bipartite graph are often with different types of attributes. The fairness property can be defined on one side of nodes, and also can be defined on two sides of nodes. Therefore, we propose two new models to characterize the fairness of bicliques in bipartite graphs called single-side fair biclique and bi-side fair biclique respectively. A single-side fair biclique is a biclique that requires one side nodes satisfying the fairness property and also it is a maximal subgraph satisfying such a property. That is, the number of vertices for each attribute is no less than a threshold β and the maximum difference between the number of vertices of every attribute is no greater than a threshold δ. Similarly, a bi-side fair biclique is a biclique that guarantees fairness on both sides, and also it is the maximal subgraph that meets such a property. In a bi-side fair biclique, the number of vertices in the upper side and the lower side for each attribute is no less than the thresholds α and β, and the maximum difference between the number of vertices of every attribute is no greater than a threshold δ. Notably, both single-side fair biclique and bi-side fair biclique can be extended to the proportion fair biclique models by introducing a fairness ratio θ. In particular, the threshold θ requires that on the fair side, the ratio of the number of vertices of each attribute to the total number of vertices is no less than θ.

Mining fair bicliques in bipartite graphs has a variety of applications. For instance, in scientific collaboration networks (e.g., DBLP), we may wish to find a team of experts that includes a similar number of junior and senior experts and also with different research areas. Such teams can be identified by mining the bi-side fair biclique in author-publication networks, as the bi-side fair biclique can ensure the team contains a similar number of junior and senior researchers and also with different research areas. In job recommendation systems (e.g., Jobs), there may exist nationality bias. That is, foreigners may be recommended for less popular jobs even if they have a better degree and working experience. The same problem lies in movie recommendation systems (e.g., Movies), in which exposure bias exists. The intuition is that already popular movies typically get more recommendation chances than relatively new movies even if they are of equal mass. To eliminate the biases, we can mine one-side fair bicliques by defining the fairness on the job side and movie side, to ensure the recommendation results are not nationality or time sensitive.

Although the practical significance of our fair biclique models, there are no existing solutions that can be used to mine all single-side fair bicliques or bi-side fair bicliques in bipartite graphs. Moreover, we show that the problem of enumerating all single-side fair bicliques or bi-side fair bicliques on bipartite graphs is NP-hard. To solve this problems, we first propose a branch and bound algorithm, called FairBCEM, with two carefully-designed pruning techniques to enumerate all single-side fair bicliques. To further improve the efficiency, we propose a novel FairBCEM++ algorithm which first enumerates all maximal bicliques and then uses a carefully-designed combinatorial enumeration technique to enumerate all results in the set of all maximal bicliques, instead of in the original bipartite graph. We show that all our techniques can also be extended to solve the bi-side fair biclique enumeration problem. To summarize, we make the following contributions.

New models. We propose a single-side fair biclique and a bi-side fair biclique models to characterize the fairness of cohesive bipartite subgraphs. Additionally, we also propose proportion single-side fair biclique and proportion bi-side fair biclique models which take account of the ratio of the number of vertices of each attribute to the total number of vertices. To the best of our knowledge, we are the first to introduce the concept of fairness into bipartite graphs for

2375-026X/23/$31.00 ©2023 IEEE
DOI 10.1109/ICDE55515.2023.00131

biclique mining tasks.

Novel algorithms. To enumerate all single-side fair bicliques, we first propose a fair α-β core pruning technique to prune unpromising nodes in the original bipartite graph. Then, we develop a pruning technique, called colorful α-β core pruning, by first constructing a 2-hop graph on the fair-side vertices and then applying the colorful core pruning technique to reduce the fair-side vertices. A branch and bound algorithm, namely, FairBCEM, is proposed to enumerate all single-side fair bicliques. To further boost the performance, we develop a new algorithm called FairBCEM++ which makes use of maximal bicliques as the candidates, and then enumerates all single-side fair bicliques in such candidates by using a carefully-devised combinatorial enumeration technique. Besides, we also extend the proposed pruning techniques and the enumeration algorithms to handle the bi-side fair biclique enumeration problem, which results in a basic enumeration algorithm BFairBCEM and an improved algorithm BFairBCEM++. Additionally, we also present the algorithms, called FairBCEMPro++ and BFairBCEMPro++, to enumerate all proportion single-side fair bicliques and proportion bi-side fair bicliques.

Extensive experiments. We conduct extensive experiments to evaluate the efficiency and effectiveness of our algorithms using five real-world networks. The results show that: (1) the pruning techniques for single-side fair biclique enumeration and bi-side fair biclique enumeration can significantly prune unpromising vertices; (2) for single-side fair biclique enumeration, FairBCEM++ is at least two orders of magnitude faster than that FairBCEM; (3) for bi-side fair biclique enumeration, BFairBCEM++ is around 3-100 times faster than BFairBCEM; (4) both our improved algorithms can process a large bipartite graph with 7,577,304 nodes and 12,282,059 edges. In addition, we conduct three case studies on DBLP, Jobs and Movies, to evaluate the effectiveness of our solutions. The results show that both single-side fair biclique and bi-side fair biclique can find meaningful and interesting fair communities in DBLP and fair recommendation results in Jobs and Movies. For reproducibility purposes, the source code of this paper is released at https://github.com/Heisenberg-Yin/fairnesss-biclique.

II. Preliminaries

Let $G = (U, V, E, A)$ be an undirected, unweighted, and attributed bipartite graph, where $U(G)$ and $V(G)$ are two disjoint vertex sets, and $E(G) \subseteq U(G) \times V(G)$ denotes the edge set of G. Generally, we call the vertex sets $U(G)$ and $V(G)$ the upper side and lower side of G, respectively. $A(G) = \{A_U, A_V\}$ is the attribute set of G in which A_U is the attribute of vertices in $U(G)$ and A_V is that of vertices in $V(G)$. For an arbitrary vertex u, we use $u.val$ to indicate the value of its attribute. Let $A(U)$ be the set of all attribute values of A_U, i.e., $A(U) = \{u.val | u \in U(G)\}$. Analogously, we denote $A(V) = \{u.val | u \in V(G)\}$. The cardinalities of $A(U)$ and $A(V)$ are A_n^U and A_n^V, respectively. We mainly focus on the case of two-dimensional attribute for each side of G, i.e., $A_n^U = A_n^V = 2$. Without loss of generality, we denote $A(U) = \{a_i^U | 0 \leq i < A_n^U\}$ and $A(V) = \{a_i^V | 0 \leq i < A_n^V\}$. The set of neighbors of vertex u in graph G is denoted as $N(u, G) = \{v | (u, v) \in E(G)\}$, and the degree of u in G is represented as $D(u, G) = |N(u, G)|$. Given a vertex set S, we use $N(S) = \{v | v \in N(u), \forall u \in S\}$ to indicate the set of neighbors of S. The number of vertices with attribute value a_i^* in the set S is $S_{a_i^*} = \{v | v.val = a_i^*\}$ where the symbol "$*$" is either U or V. We omit the symbol G in the above notations when the context is clear.

Definition 1: (Biclique) Given an bipartite graph $G(U, V, E)$, a subgraph C is a biclique if: (1) $E(C) = U(C) \times V(C)$; (2) $U(C) \subseteq U(G)$; (3) $V(C) \subseteq V(G)$.

Definition 2: (Maximal biclique) Given a bipartite graph $G(U, V, E)$ and a subgraph C, C is a maximal biclique if: (1) C is a biclique; (2) there is no other biclique $C' \supset C$ satisfies (1).

Below, we introduce two novel fairness-aware biclique models, namely, Single-Side Fair BiClique (SSFBC) and Bi Side Fair BiClique (BSFBC). Without losing generality, we consider V as the fair side in the SSFBC model and both U and V as the fair sides in BSFBC.

Definition 3: (Single-side fair biclique) Given an attributed bipartite graph $G(U, V, E, A)$ and three integers α, β, δ, a biclique $C(U, V, E, A)$ of G is a single-side fair biclique if (1) $|C(U)| \geq \alpha$; (2) $\forall a_i^V \in A(V), |C(V)_{a_i^V}| \geq \beta$ and $\forall a_i^V, a_j^V \in A(V), ||C(V)_{a_i^V}| - |C(V)_{a_j^V}|| \leq \delta$; (3) there is no biclique $C' \supset C$ satisfying (1) and (2).

Definition 4: (Bi-side fair biclique) Given an attributed bipartite graph $G(U, V, E, A)$ and three integers α, β, δ, a biclique $C(U, V, E, A)$ of G is a bi-side fair biclique if (1) $\forall a_i^U \in A(U), |C(U)_{a_i^U}| \geq \alpha$ and $\forall a_i^U, a_j^U \in A(U)$, $||C(U)_{a_i^U}| - |C(U)_{a_j^U}|| \leq \delta$; (2) $\forall a_i^V \in A(V), |C(V)_{a_i^V}| \geq \beta$ and $\forall a_i^V, a_j^V \in A(V)$, $||C(V)_{a_i^V}| - |C(V)_{a_j^V}|| \leq \delta$; (3) there is no biclique $C' \supset C$ satisfying (1) and (2).

Example 1: Consider an attributed bipartite graph $G = (U, V, E, A)$ in Fig. 1(a). For the upper side $U(G)$, the values of attribute A_U are represented as a and b in a square, respectively. And the attribute values of A_V are a and b in a circle for the lower side $V(G)$. Suppose that $\alpha = 1, \beta = 2$ and $\delta = 1$. By Definition 3, the subgraph C_S induced by the vertex set $\{u_3, u_4, v_2, v_4, v_6, v_9\}$ is a SSFBC of G and the subgraph C_B induced by $\{u_3, u_4, v_2, v_4, v_6, v_9\}$ is a BSFBC Clearly, C_B is a subgraph of C_S, which means that a BSFBC must be contained in SSFBCs. □

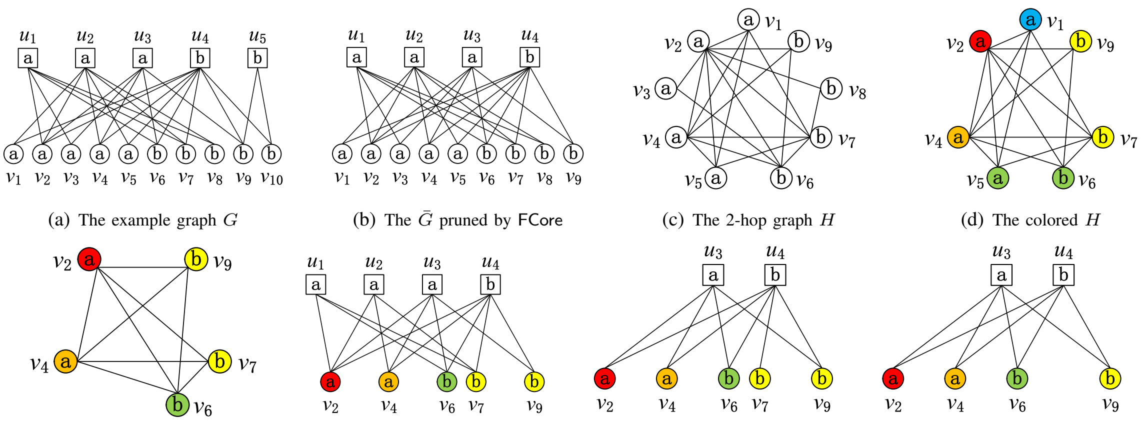
In addition, fairness considers not only the number of vertices with each attribute but also the ratio of the number of vertices of each attribute to the total number of vertices on the fair side. Below, we propose two extended models of SSFBC and BSFBC namely, Proportion Single-Side Fair BiClique (PSSFBC) and Proportion Bi-Side Fair BiClique (PBSFBC), to further guarantee the fairness by introducing a fairness radio threshold. θ

Definition 5: (Proportion single-side fair biclique) Given an attributed bipartite graph $G(U, V, E, A)$, three integers α, β, δ and a float θ, a biclique $C(U, V, E, A)$ of G is a proportion single-side fair biclique if (1) $|C(U)| \geq \alpha$; (2) $\forall a_i^V \in A(V), |C(V)_{a_i^V}| \geq \beta$ and $\forall a_i^V, a_j^V \in A(V)$, $||C(V)_{a_i^V}| - |C(V)_{a_j^V}|| \leq \delta$; (3) $\forall a_i^V \in A(V), |C(V)_{a_i^V}|/|C(V)| \geq \theta$; (4) there is no biclique $C' \supset C$ satisfying (1), (2) and (3).

Definition 6: (Proportion bi-side fair biclique) Given an attributed bipartite graph $G(U, V, E, A)$, three integers α, β, δ and a float θ, a biclique $C(U, V, E, A)$ of G is a proportion bi-side fair biclique if (1) $\forall a_i^U \in A(U), |C(U)_{a_i^U}| \geq \alpha$ and $\forall a_i^U, a_j^U \in A(U)$, $||C(U)_{a_i^U}| - |C(U)_{a_j^U}|| \leq \delta$; (2) $\forall a_i^V \in A(V), |C(V)_{a_i^V}| \geq \beta$ and $\forall a_i^V, a_j^V \in A(V)$, $||C(V)_{a_i^V}| - |C(V)_{a_j^V}|| \leq \delta$; (3) $\forall a_i^V \in A(V), |C(V)_{a_i^V}|/|C(V)| \geq, \theta$ $\forall a_i^U \in A(U), |C(U)_{a_i^U}|/|C(U)| \geq \theta$; (4) there is no biclique $C' \supset C$ satisfying (1), (2) and (3).

Problem statement. Given an attributed bipartite graph $G(U, V, E, A)$, three integers α, β, δ, and a float θ, our goal is to find all SSFBCs, PSSFBCs, BSFBCs, PBSFBCs in G.

Hardness. We first discuss the hardness of the single-side fair biclique enumeration problem. Considering a special case: $\alpha = 0, \beta = 0, \delta = n$, where n is the graph size. Clearly, with these parameters, the single-side fair biclique enumeration problem degenerates to the traditional maximal biclique enumeration problem, which is NP-hard. Thus, finding all single-side fair bicliques is also an NP-hard problem. The bi-side fair biclique enumeration problem is more challenging than enumerating all single-side fair bicliques because the number of bi-side fair bicliques is often much larger than that of single-side fair bicliques. By definition, we can see



(a) The example graph G (b) The $\bar{G}$ pruned by FCore (c) The 2-hop graph H (d) The colored H

(e) The ego colorful 2-core $\bar{H}$ (f) The $\bar{G}$ pruned by $\bar{H}$ (g) The $\bar{G}$ pruned by FCore (h) The $\bar{G}$ pruned by CFCore

Fig. 1. The pruning process of FCore and CFCore on the example graph G.

that a bi-side fair biclique is always contained in a single-side fair biclique. On the contrary, a single-side fair biclique is not necessarily a bi-side fair biclique.

Compared to the traditional biclique enumeration problem, the fairness-aware biclique enumeration problem is harder. First, both single-side fair biclique and bi-side fair biclique-models do not satisfy the hereditary property. That is, subgraphs of a SSFBC or BSFBC are not always fair subgraphs due to the attribute constraint. As a result, it is more difficult to check the maximally for both single-side fair biclique and bi-side fair biclique. Second, the number of fairness-aware bicliques is generally larger than that of traditional maximal bicliques, resulting in a higher time cost to enumerate all fairness-aware bicliques. For example, on IMDB, with the parameters $\alpha = 8, \beta = 10, \delta = 2$, the number of maximal bicliques and single-side fair bicliques are 12,614 and 3,502,746, respectively. In the case of $\alpha = 4, \beta = 6, \delta = 2$, we can find 42,023 maximal bicliques and 11,091,721 bi-side fair bicliques.

Below, we analyze the lower bounds of time complexity for finding all SSFBCs and BSFBCs. We first introduce an important theorem which is proved in [23].

Theorem 2.1: Every bipartite graph with n vertices contains at most $2^{n/2}$ bicliques [23].

In the worst case, all bicliques can satisfy the α and β constraints of Definition 3, and thus we only consider the parameter $delta$. Given a biclique $C(U, V, E, A)$, without loss of generality, we assume that $|C(V)_{a_1^V}| = |C(V)_{a_2^V}| + n_1$ and $|C(U)_{a_1^U}| = |C(U)_{a_2^U}| + n_2$ hold, where $n_1 > \delta$ and $n_2 > \delta$. Then, the number of SSFBCs is $\binom{|C(V)_{a_2^V}+\delta|}{|C(V)_{a_1^V}|}$, whose maximum value is $\binom{\lfloor C(V)_{a_1^V}/2\rfloor}{|C(V)_{a_1^V}|}$. Similarly, the maximum number of BSFBCs is equal to $\binom{\lfloor C(V)_{a_1^V}/2\rfloor}{|C(V)_{a_1^V}|}\binom{\lfloor C(U)_{a_1^U}/2\rfloor}{|C(U)_{a_1^U}|}$. Since there are $2^{n/2}$ bicliques (Theorem 2.1) and $C(V), C(u) \leq n$ holds, finding all SSFBCs and BSFBCs take at least $O(C_n^{\lfloor n/2\rfloor} * 2^{n/2})$ and $O(C_n^{\lfloor n/2\rfloor})^2 * 2^{n/2})$ time respectively as algorithms need to output these fair bicliques.

For enumerating all PSSFBCs and PBSFBCs, the lower bound of time complexity can be easily derived by analogous methods of finding SSFBCs and BSFBCs, we omit the analysis due to the space limit.

III. Single-side fair biclique enumeration

In this section, we first introduce two non-trivial pruning techniques, called fair α-β core pruning and colorful fair α-β core pruning, to reduce the scale of a graph. Then, two branch-and-bound enumeration algorithms, called FairBCEM and FairBCEM++, are proposed to enumerate all single-side fair bicliques. Finally, we develop the FairBCEMPro++ algorithm to solve the PSSFBC enumeration problem.

A. Fair α-β core pruning

Below, we first give the definition of *attribute degree* which is important to derive the fair α-β core pruning technique.

Definition 7: (Attribute degree) Given an attributed bipartite graph $G = (U, V, E, A)$ and an attribute value $a_i \in A(U) \cup A(V)$. The attribute degree of vertex u, denoted by $D_{a_i}(u, G)$, is the number of vertices of u's neighbors whose attribute value is a_i, i.e., $D_{a_i}(u, G) = |\{v | v \in N(u), v.val = a_i\}|$.

Definition 8: (Fair α-β core) Given an attributed bipartite graph $G = (U, V, E, A)$, a subgraph $H = (L, R, E, A)$ is a fair α-β core if (1) $D_{a_i}(u, H) \geq \beta, u \in L, a_i \in A(V)$; (2) $D(v, H) \geq \alpha, v \in R$; (3) there is no subgraph $H' \supset H$ that satisfies (1) and (2) in G.

With Definition 8, we have the following lemma. Due to the space limit, all the proofs in this paper are omitted.

Lemma 1: Given an attributed bipartite graph $G = (U, V, E, A)$ and two integers α, β, any single-side fair biclique must be contained in a fair α-β core.

According to Lemma 1, we propose a fair α-β core computation algorithm, namely, FCore, to prune unpromising vertices that definitely do not belong to any single-side fair biclique. The pseudo-code of FCore is outlined in Algorithm 1, which is a variant of the classic core decomposition algorithm [2], [19]. Specifically, a priority queue Q is used to maintain the vertices which will be removed during the peeling procedure (line 1). FCore first calculates the attribute degrees and degrees for vertices in the upper side and lower side, respectively, to initialize Q (lines 2-10). Based on Definition 8, for a vertex $u \in U$ (i.e., the upper side), FCore removes u from G once its minimum attribute degree $D_{min}(u)$ is less than β; and for $v \in V$, (i.e., the lower side), it removes v from G once its degree $D(v)$ is less than α. After that, the algorithm computes the fair α-β core of G by iteratively peeling vertices from the remaining graph based on their degrees and attribute degrees (lines 11-24). Finally, FCore returns the remaining graph $\hat{G}$ as the fair α-β core. It is easy to show that FCore consumes $O(E + V)$ time using $O(U \times A_n^V + V)$ space.

B. Colorful fair α-β core pruning

The fair α-β core pruning may not be very effective as it only employs the constraint of attribute degree and ignores the property of cliques. To this end, we present a more powerful pruning technique, called Colorful Fair α-β core (CFCore) pruning, by establishing an interesting connection between our problem and the weak fair clique model proposed in [21].

Recall that by Definition 3, in a single-side fair biclique C, any two vertices in $C(V)$ share at least α common neighbors.

Algorithm 1: FCore

Input: $G = (U, V, E, A)$, two integers α, β
Output: The fair α-β core $\hat{G}$

Let $\mathcal{Q}$ be a priority queue; $\mathcal{Q} \leftarrow \emptyset$;
for $u \in U$ **do**
  for $v \in N(u)$ **do** $D_{v.val}(u)$++;
  $D_{\min}(u) \leftarrow \min\{D_{a_i^V}(u) | a_i^V \in A(V)\}$;
for $u \in U$ **do**
  if $D_{\min}(u) < \beta$ **then** $\mathcal{Q}.push(u)$; Remove u from G;
for $v \in V$ **do**
  for $u \in N(v)$ **do** $D(v)$++;
for $v \in V$ **do**
  if $D(v) < \alpha$ **then** $\mathcal{Q}.push(v)$; Remove v from G;
while $\mathcal{Q} \neq \emptyset$ **do**
  $u \leftarrow \mathcal{Q}.pop()$;
  for $v \in N(u)$ **do**
    if *v is not removed* **then**
      if $v \in U$ **then**
        $D_{u.val}(v)$--;
        $D_{\min}(v) \leftarrow \min\{D_{a_i^V}(v) | a_i^V \in A(V)\}$;
        if $D_{\min}(v) < \beta$ **then**
          $\mathcal{Q}.push(v)$; Remove v from G;
      if $v \in V$ **then**
        $D(v)$--;
        if $D(v) < \alpha$ **then**
          $\mathcal{Q}.push(v)$; Remove v from G;
        $\mathcal{Q}.push(v)$; Remove v from G;
$\hat{G} \leftarrow$ the remaining graph of G;
return $\hat{G}$;

Algorithm 2: CFCore

Input: $G = (U, V, E, A)$, two integers α, β
Output: The pruned graph $\hat{G}$

1 $\bar{G}(U, V, E, A) \leftarrow$ FCore(G, α, β);
2 Let $\mathcal{Q}$ be a priority queue; $\mathcal{Q} \leftarrow \emptyset$;
3 $H(V, E, A_V) \leftarrow$ Construct2HopGraph$(\bar{G}, \alpha, \bar{G}(V))$;
4 **for** $u \in H(V)$ **do**
5   **if** $D(u, H) < A_n^V \times \beta - 1$ **then** Remove u from H;
6 Color all vertices in H by invoking a degree based greedy coloring algorithm;
7 **for** $u \in H(V)$ **do**
8   **for** $v \in N(u) \cup \{u\}$ **do**
9   **if** $M_u(v.val, color(v)) = 0$ **then** $ED_{v.val}(u)$++;
10     $M_u(v.val, color(v))$++;
11   $ED_{\min}(u) \leftarrow \min\{ED_{a_i^V}(u) | a_i^V \in A(V)\}$;
12 **for** $u \in H(V)$ **do**
13   **if** $ED_{\min}(u) < \beta$ **then** $\mathcal{Q}.push(u)$; Remove u from H;
14 **while** $\mathcal{Q} \neq \emptyset$ **do**
15   $u \leftarrow \mathcal{Q}.pop()$;
16   **for** $v \in N(u, H)$ **do**
17   **if** *v is not removed* **then**
18       $M_v(u.val, color(u))$--;
19   **if** $M_v(u.val, color(u)) \leq 0$ **then**
20         $ED_{u.val}(v) \leftarrow ED_{u.val}(v) - 1$;
21         $ED_{\min}(v) \leftarrow \min\{ED_{a_i^V}(v) | a_i^V \in A(V)\}$;
22   **if** $ED_{\min}(v) < \beta$ **then**
23           $\mathcal{Q}.push(v)$; Remove v from H;
24 The ego colorful β-core $\bar{H} \leftarrow$ the remaining graph of H;
25 **for** $u \in \bar{G}(V) - \bar{H}(V)$ **do**
26   Remove u from $\bar{G}(V)$;
27 $\hat{G} \leftarrow$ FCore$(\bar{G} = (U, V, E, A), \alpha, \beta)$;
28 **return** $\hat{G}$;

Thus, we can construct a 2-hop graph $H(V, E, A)$ on the fair side of G as follows. We keep the vertices of H as those in the lower side of G, i.e., $H(V) = G(V)$ and $A = A_V$. Given two vertices $v_i, v_j \in V(H)$, if the number of common neighbors of v_i and v_j in G is no less than α, we connect v_i and v_j in H as v_i and v_j may appear in the same single-side fair biclique. With the 2-hop graph H, we have the following observation.

Observation 1: Given an attributed bipartite graph G and its 2-hop graph H. For an arbitrary single-side fair biclique C, the vertices in $C(V)$ form a clique $\hat{C}$ in H in which the number of vertices whose attribute value equals a_i^V is no less than β.

With Observation 1, the clique $\hat{C}$ satisfies the fairness restriction of the weak fair clique model in [21]. As a weak fair clique is maximal, $\hat{C}$ must be contained in a weak fair clique. Thus, we can apply the colorful core pruning technique proposed in [21] to prune unpromising vertices in H that cannot form a weak fair clique. However, the colorful core pruning in [21] does not consider the attribute value of the vertex itself. Below, we give the variants of colorful degree and colorful core, called *ego colorful degree* and *ego colorful core* by incorporating the vertex attribute.

Definition 9: (Ego colorful degree) Given an attributed graph $G = (V, E, A)$ and an attribute value $a_i \in A$. The ego colorful degree of vertex u, denoted by $ED_{a_i}(u, G)$, is the number of colors of u and u's neighbors whose attribute value is a_i, i.e., $ED_{a_i}(u, G) = |\{color(v) | v \in N(u) \cup \{u\}, v.val = a_i\}|$.

In Definition 9, the color of each node can be obtained by the classic greedy graph coloring algorithm [14], which ensures that two adjacent nodes have different colors. Let $ED_{\min}(u, G)$ denotes the minimum ego colorful degree of a vertex u, i.e., $ED_{\min}(u, G) = \min\{ED_{a_i}(u, G) | a_i \in A\}$. We omit the symbol G in $ED_{a_i}(u, G)$ and $ED_{\min}(u, G)$ when the context is clear.

Definition 10: (Ego colorful k-core) Given an attributed graph $G = (V, E, A)$ and an integer k, a subgraph $H = (V_H, E_H, A)$ of G is an ego colorful k-core if: (1) for each vertex $u \in V_H, ED_{\min}(u, H) \geq k$; (2) there is no subgraph H' that satisfies (1) and $H' \supset H$.

Based on Definition 10, we have the following lemma.

Lemma 2: Given an attributed bipartite graph G, its 2-hop graph H, and the parameters α, β, δ. For an arbitrary single-side fair biclique C, the vertices in $C(V)$ must be contained in the ego colorful β-core of H.

With Lemma 2, we can construct a 2-hop graph H based on the fair side V and prune the vertices in $G(V)$ that cannot form a single-side fair biclique by calculating the ego colorful β-core of H. Obviously, the scale of ego colorful β-core is smaller than that of H. That means that some vertices in the lower side can be removed from G, and thus we can further apply the FCore to prune the vertices in both the upper side and lower side of G. Based on this idea, we propose a colorful fair α-β core pruning algorithm, namely, CFCore, as shown in Algorithm 2. The CFCore algorithm works as follows. It first performs FCore (Algorithm 1) to calculate the fair α-β core $\bar{G}$ according to Lemma 1 (line 1). The CFCore algorithm then constructs a 2-hop graph H on the fair (lower) side $G(V)$ (Algorithm 3), and deletes the vertices whose degree is less than $A_n^V \times \beta - 1$ as such vertices clearly cannot form a single-side fair biclique (lines 3-5). After that, CFCore uses the greedy coloring for H which colors vertices based on the order of degree [2], [19], and computes the ego colorful β-core $\bar{H}$ by iteratively peeling vertices from the remaining graph based on their ego colorful degrees (lines 6-24). According to Lemma 2, the CFCore safely removes the vertices that are not contained in the ego colorful β-core $\bar{H}$ from $\bar{G}$. It further performs FCore (Algorithm 1) again to reduce the vertices for both the upper side and lower side of $\bar{G}$ (lines 25-27). Finally, CFCore returns the pruned graph $\hat{G}$ which contains all single-side fair bicliques. Algorithm 2 consumes $O(E + V + \sum_{u \in U} d(u, G)^2 + \sum_{v \in V} d(v, G)^2)$ time using $O(V \times A_n^V \times color)$ space.

Example 2: Consider the bipartite graph $G = (U, V, E, A)$ in Fig. 1(a). Suppose that we set $\alpha = 2, \beta = 2$. The CFCore first performs FCore to calculate fair α-β core denoted by $\bar{G}$ as shown in Fig. 1(b). Then it constructs 2-hop graph H for the fair side V of $\bar{G}$ (i.e., the vertices in circle), which is illustrated

Algorithm 3: Construct2HopGraph

Input: $G = (U, V, E, A)$, a integer α, the fair side V
Output: The 2-hop graph H based on the fair side V

Let $H = (V = G(V), E = \emptyset, A = A_V)$ be an attributed graph;
for $v \in G(V)$ **do**
    Initialize an array C with $C[i] = 0, 1 \leq i \leq |G(V)|$;
    for $u \in N(v, G)$ **do**
        for $w \in N(u, G)$ **do**
            if $w \neq v$ **then** $C[w] \leftarrow \mathcal{C}[w] + 1$;
    for $u \in G(V)$ **do**
        if $C[u] \geq \alpha$ *and* $u < v$ **then** $E(H) \leftarrow E(H) \cup (u, v)$;
return H;

Algorithm 4: MFSCheck

Input: The sets $S, \hat{S}$, the set of attribute values A, two integers k, δ
Output: true: $\hat{S}$ is a maximal fair subset; false: $\hat{S}$ is not a maximal fair subset

if $\exists a_i \in A, \hat{S}_{a_i} < k$ **then return false**;
$C \leftarrow S - \hat{S}$;
if $\forall a_i \in A, |C_{a_i}| > 0$ **then return false**;
for $a_i \in A$ **do**
    if $|C_{a_i}| > 0$ **then**
        if $\exists u \in C_{a_i}, \hat{S} \cup \{u\}$ is a fair set **then return false**;
return true;

in Fig. 1(c). The vertex v_3 in Fig. 1(c) with two neighbors cannot form a single-side fair biclique, and we remove it from H. This is because a single-side fair biclique C contains at least $A_n^V \times \beta$ vertices in the lower side V, which requires that the vertices in $V(C)$ should have at least $A_n^V \times \beta - 1 = 2 \times 2 - 1 = 3$ neighbors in the 2-hop graph H. Analogously, vertex v_8 in Fig. 1(c) is not included in a single-side fair biclique and we also remove v_8 from H. After the degree pruning, we color H using a greedy coloring algorithm [14] as shown Fig. 1(d), and computes the ego colorful 2-core $\bar{H}$. Taking v_1 as an example, we derive the ego colorful degrees of v_1, i.e., $ED_a(v_1, H) = 4$ and $ED_b(v_1, H) = 1$. Further, we have $ED_{\min}(v_1, H) = 1 \leq \beta = 2$. Thus, v_1 can be safely removed, since it is not in the ego colorful 2-core and also not in a single-side fair biclique by Lemma 2. Fig. 1(e) shows the ego colorful 2-core $\bar{H}$. We use $\bar{H}$ to prune the bipartite graph $\bar{G}$. The remaining graph is illustrated in Fig. 1(f). Clearly, in the lower side, the pruned $\bar{G}$ only has 5 vertices while the previous $\bar{G}$ in Fig. 1(b) has 9 vertices. Further, CFCore performs FCore again to remove the vertices in $\bar{G}$ as depicted in Fig. 1(g) and Fig. 1(h). The final graph pruned by CFCore is shown in Fig. 1(h), which is significantly small than the original graph in Fig. 1(a). ☐

C. *The* FairBCEM *algorithm*

Before introducing the FairBCEM algorithm, we first give two important definitions, i.e., *fair set* and *maximal fair subset*.

Definition 11: (Fair set) Given an attributed set S with attribute values in A and two integers k, δ, we call S is a fair set if (1) $\forall a_i \in A, |S_{a_i}| \geq k$; (2) $\forall a_i, a_j \in A, ||S_{a_i}| - |S_{a_j}|| \leq \delta$.

Definition 12: (Maximal fair subset) Given an attributed set S with attribute values in A and two integers k, δ, $\hat{S} \subseteq S$ is a maximal fair subset if (1) $\hat{S}$ is a fair set based on k, δ; (2) there is no fair set $\bar{S} \subset S$ satisfying $\hat{S} \subset \bar{S}$.

Here we propose an efficient algorithm to identify whether a set $\hat{S}$ is the maximal fair subset of the set S as shown in Algorithm 4. Clearly, $\hat{S}$ is a maximal fair subset when it satisfies there is no subset of $S - \{\hat{S}\}$ could be added into $\hat{S}$ without harming its fairness.

Equipped with CFCore pruning techniques, we propose the FairBCEM algorithm which enumerates all single-side fair bicliques based on a branch and bound search method. In FairBCEM, there are four important sets: L, R, P, Q which control the generation of the search tree. Specifically, we use R to denote the currently-found vertices in the lower side V which may be extended to a single-side fair biclique. L is the vertex set in the upper side U in which every vertex is a neighbor of all vertices in R. P is the candidate set in V that can be used to extend R in the search tree. Q is the set of vertices in which every vertex can be used to expand R but has already been visited in previous search paths. Below, we give some observations to explain our FairBCEM algorithm.

Observation 2: If $\forall a_i^V \in A(V)$, we can find that at least one vertex $v \in Q$ with $v.val = a_i^V$ satisfying $\forall u \in L, (u, v) \in E$, R is not a maximal and thus we can end the current search and all deeper searches.

Observation 3: Given a fair set R, if there is no vertex set $S \subseteq P \cup Q$ which is fully connected to L and could be added into R without breaking the fairness, then (L, R) is a single-side fair biclique.

Observation 4: If all nodes in P are fully connected to R, and $R \cup P$ is a fair set, then we can add all vertices in P into R without losing solution.

Observation 5: If $|L| < \alpha$ or $\exists a_i^V \in A(V), |R_{a_i^V}| + |P_{a_i^V}| < \beta$, we can terminate the current search branch.

Based on above observations, the FairBCEM algorithm for single-side fair biclique enumeration is outlined in Algorithm 5. It first employs the CFCore pruning to remove vertices that cannot be in a single-side fair biclique and initializes four sets L, R, P, Q, and then invokes the BackTrackFBCEM procedure to find all single-side fair bicliques with the branch-and-bound technique. In BackTrackFBCEM, each vertex x in P is used to extend the current-found R. With the adding of x, L must be updated to keep out those vertices that are not adjacent to x, as each vertex in L is a neighbor of all vertices in R (lines 7-8). A variable $flag$, initialized as true, indicates that whether there is a single-side fair biclique in the current branch. We denote Q^{FC} and P^{FC} the vertices in Q and P that are fully connected to L respectively, which are used to check the maximality of R. Clearly, if $|L'| < \alpha$, we cannot find a single-side fair biclique because it violates the restriction on the number of vertices in the upper side in Definition 3, and thus we set $flag$ to false (line 9). Then, the BackTrackFBCEM procedure identifies whether R is maximal with the set Q based on Observation 2 and maintains the value of $flag$ and the set Q' (lines 10-15). Once $flag$ equals false, there is no single-side fair biclique in the current branch and we move x from P to Q to indicate that x has been searched (lines 29-30). Otherwise, the BackTrackFBCEM computes the sets P' and P^{FC} with the candidate set P (lines 17-20). If $P' = P^{FC}$, all vertices in P are fully connected to R' and we can directly check if $(L', R' \cup P^{FC})$ is a single-side fair biclique according to Observation 4. If so, BackTrackFBCEM adds the biclique $(L', R' \cup P^{FC})$ into the result set Res and updates P' and P^{FC} as empty sets (lines 21-23). After that, the procedure identifies whether R' is a maximal fair set of $R' \cup P^{FC} \cup Q^{FC}$ by Algorithm 4 and adds (L', R') into Res by Observation 3 (lines 24-26). Subsequently, If $P' \neq \emptyset$ and $\forall a_i^V \in A(V), |R'_{a_i^V}| + |P'_{a_i^V}| \geq \beta$, BackTrackFBCEM performs the next backtracking with the new L', R', P', Q' (lines 27-28). The final set Res maintains all single-side fair bicliques in G (line 4).

Correctness analysis. Clearly, we enumerate all possible R based on the sets P, Q and all single-side fair bicliques lie in the enumeration tree, thus the completeness of our algorithm is satisfied. The fairness and maximality of a biclique are satisfied at line 22 and line 25 of Algorithm 5. Besides, the set Q can guarantee that each single-side fair biclique only be enumerated once, thus our algorithm also satisfy the non-redundancy property. In conclusion, our FairBCEM algorithm can correctly output all single-side fair bicliques.

Algorithm 5: FairBCEM

Input: A bipartite graph $G = (U, V, E, A)$, three integers α, β, δ
Output: The set of all single-side fair bicliques Res

$\hat{G} = (\hat{U}, \hat{V}, \hat{E}, A) \leftarrow \mathsf{CFCore}(G, \alpha, \beta)$;
$L \leftarrow \hat{U}$; $R \leftarrow \emptyset$; $P \leftarrow \hat{V}$; $Q \leftarrow \emptyset$;
$\mathsf{BackTrackFBCEM}(L, R, P, Q)$;
return Res;

Procedure $\mathsf{BackTrackFBCEM}(L, R, P, Q)$
while $P \neq \emptyset$ **do**
- $x \leftarrow$ a vertex in P; $flag \leftarrow true$;
- $R' \leftarrow R \cup \{x\}$; $L' \leftarrow \{u \in L | (u, x) \in \hat{E}\}$;
- **if** $|L'| < \alpha$ **then** $flag \leftarrow false$;
- **for** $u \in Q$ **do**
  - $N(u) = \{v \in L' | (u, v) \in \hat{E}\}$;
  - **if** $|N(u)| = |L'|$ **then** $Q^{FC} \leftarrow Q^{FC} \cup \{u\}$;
  - **if** $|N(u)| \geq \alpha$ **then** $Q' \leftarrow Q' \cup \{u\}$;
- **if** $\forall a_i^V \in A(V), Q^{FC}_{a_i^V} > 0$ **then**
  - $flag \leftarrow false$;
- **if** *flag* **then**
  - **for** $v \in P, v \neq x$ **do**
    - $N(v) = \{u \in L' | (u, v) \in \hat{E}\}$;
    - **if** $|N(v)| = |L'|$ **then** $P^{FC} \leftarrow P^{FC} \cup \{v\}$;
    - **if** $|N(v)| \geq \alpha$ **then** $P' \leftarrow P' \cup \{v\}$;
  - **if** $P^{FC} = P'$ **then**
    - **if** $(L', R' \cup P^{FC})$ *is a fair one-side biclique* **then**
      - $R' \leftarrow R' \cup P^{FC}$; $P^{FC} \leftarrow \emptyset$; $P' \leftarrow \emptyset$;
  - **if** R' *is a fair set* **then**
    - **if** R' *is maximal fair subset of* $R' \cup P^{FC} \cup Q^{FC}$ **then**
      - $Res \leftarrow Res \cup (L', R')$;
  - **if** $P' \neq \emptyset$ *and* $\forall a_i^V \in A(V), |R'_{a_i^V}| + |P'_{a_i^V}| \geq \beta$ **then**
    - $\mathsf{BackTrackFBCEM}(L', R', P', Q')$;
- $P \leftarrow P - \{x\}$;
- $Q \leftarrow Q \cup \{x\}$;

Algorithm 6: FairBCEM++

Input: A bipartite graph $G = (U, V, E, A)$, three integers α, β, δ
Output: The set of all single-side fair bicliques Res

1 $\hat{G} = (\hat{U}, \hat{V}, \hat{E}, A) \leftarrow \mathsf{CFCore}(G, \alpha, \beta)$;
2 $L \leftarrow \hat{U}$; $R \leftarrow \emptyset$; $P \leftarrow \hat{V}$; $Q \leftarrow \emptyset$;
3 $\mathsf{BackTrackFBCEM{+}{+}}(L, R, P, Q)$;
4 **return** Res;

5 **Procedure** $\mathsf{BackTrackFBCEM{+}{+}}(L, R, P, Q)$
6 **while** $P \neq \emptyset$ **do**
7 $x \leftarrow$ a vertex in P; $flag \leftarrow true$;
8 $R' \leftarrow R \cup \{x\}$; $L' \leftarrow \{u \in L | (u, x) \in \hat{E}\}$;
9 **if** $|L'| < \alpha$ **then** $flag \leftarrow false$;
10 **for** $u \in Q$ **do**
11 $N(u) = \{v \in L' | (u, v) \in \hat{E}\}$;
12 **if** $|N(u)| = |L'|$ **then** $flag \leftarrow false$; **break**;
13 **if** $|N(u)| > 0$ **then** $Q' \leftarrow Q' \cup \{u\}$;
14 $C \leftarrow C \cup \{u\}$;
15 **if** *flag* **then**
16 **for** $v \in P, v \neq x$ **do**
17 $N(v) = \{u \in L' | (u, v) \in \hat{E}\}$;
18 **if** $|N(v)| = |L'|$ **then**
19 $R' \leftarrow R' \cup \{v\}$;
20 $N^{lap}(v) = \{u | u \in L / L', (u, v) \in \hat{E}\}$;
21 **if** $|N^{lap}(v)| = 0$ **then** $C \leftarrow C \cup \{v\}$;
22 **if** $|N(v)| \geq \alpha$ **then** $P' \leftarrow P' \cup \{v\}$;
23 **if** (L', R') *is a single-side fair biclique* **then**
24 $Res \leftarrow Res \cup (L', R')$;
25 **else**
26 $\mathcal{R}' \leftarrow \mathsf{Combination}(R', A(V), \beta, \delta)$;
27 **for** $r' \in \mathcal{R}'$ **do**
28 **if** $N(r') = L$ **then** $Res \leftarrow Res \cup (L', r')$;
29 **if** $P' \neq \emptyset$ *and* $\forall a_i^V \in A(V), |R'_{a_i^V}| + |P'_{a_i^V}| \geq \beta$ **then**
30 $\mathsf{BackTrackFBCEM{+}{+}}(L', R', P', Q')$;
31 $P = P - C$;
32 $Q = Q \cup C$;

Algorithm 7: Combination

Input: A set S, the set of attribute value A, two integers k, δ
Output: The set of all combinations $\mathcal{C}an\mathcal{S}et$

if $\exists a_i \in A, S_{a_i} < k$ **then**
- **return** $\emptyset$;

$msize = \min_{a_i \in A} S_{a_i}$;
for $a_i \in A$ **do**
- $csize = \min(S_{a_i}, msize + \delta)$;
- $\mathcal{R}es(a_i) \leftarrow$ all subsets of S_{a_i} that with size equals $csize$;

$\mathcal{C}an\mathcal{S}et \leftarrow \mathcal{R}es(a_0)$;
for $a_i \in A, i \neq 0$ **do**
- $\mathcal{C}an\mathcal{S}et = \mathcal{C}an\mathcal{S}et \times \mathcal{R}es(a_i)$;

return $\mathcal{C}an\mathcal{S}et$;

D. *The* FairBCEM++ *algorithm*

The FairBCEM algorithm may suffer from large search space due to enormous single-side fair bicliques. To further improve the efficiency, we propose a new algorithm, called FairBCEM++, which first enumerates all maximal bicliques and then uses a combinatorial enumeration technique to find all single-side fair bicliques in the set of all maximal bicliques. Our algorithm is based on the key observation that any single-side fair biclique must be contained in a biclique.

More specifically, FairBCEM++ first find all maximal bicliques satisfying $|L| \geq \alpha$ and $R_{a_i^V} \geq \beta, \forall a_i^V \in A(V)$, and then enumerates all single-side fair bicliques among them. The pseudo-code of FairBCEM++ is depicted in Algorithm 6. Similar to FairBCEM, FairBCEM++ uses the CFCore pruning to remove unpromising vertices and then performs the BackTrackFBCEM++ procedure to find all single-side fair bicliques (lines 1-3). In each iteration of BackTrackFBCEM++, we find all maximal bicliques based on the idea of the MBEA++ algorithm [41] which adds a set of vertices (i.e., the set C) into R once. Specifically, it first extends R by adding x and obtain the set L' in which vertices are linked to x (lines 7-8). Then, it determines whether (L', R') is a maximal biclique by trying to add each vertex u in Q to the current biclique. Clearly, if not, we can terminate the current search as any single-side fair biclique must be in a biclique (lines 10-13). Otherwise, we move the vertices connected to all vertices in L' from P to R' once and update the sets C and P' (lines 16-22). We consider two cases for (L', R'): (1) R' is a fair set then (L', R') is a single-side fair biclique (lines 23-24); (2) R' is not a fair set then we calculate all maximal fair subsets of R' to further enumerate single-side fair bicliques (lines 25-28). The maximal fair subsets can be obtained by a combinatorial enumeration method as illustrated in Algorithm 7. Let $r' \in \mathcal{R}'$ be a maximal fair subset of R'. If $N(r')$ equals L, we obtain a single-side fair bicliqueand the BackTrackFBCEM++ procedure adds (L', r') into the result set Res (line 28). Similar to FairBCEM, BackTrackFBCEM++ invokes the next backtracking procedure if $P' \neq \emptyset$ and $\forall a_i^V \in A(V), |R'_{a_i^V}| + |P'_{a_i^V}| \geq \beta$ hold (lines 29-30). Finally, the set Res maintains all single-side fair bicliques in G (line 4).

Correctness analysis. The bicliques with $|L| \geq \alpha, |R_{a_i}| \geq \beta$ are enumerated due to the correctness of MBEA++ [41]. For any maximal biclique $B(L, R')$, the algorithm enumerates all single-side fair bicliques in B. Since every single-side fair biclique is contained in a maximal bilcique, FairBCEM++ satisfies completeness. In line 26, we find all maximal fair subsets of R' by the Combination algorithm and identify whether they form a biclique with L. Thus, the fairness constraint is satisfied. As L is shrinking during the search process, the maximality is also met due to the line 28. Meanwhile, each single-side fair biclique $B'(L, R')$'s L is the L of a maximal biclique $B(L, R')$ and every maximal biclique has different L, thus every single-side fair biclique only be enumerated in one maximal biclique, which avoids repeated enumeration.

Extending to finding all PSSFBCs. We propose an algorithm, called FairBCEMPro++, to enumerate all PSSFBCs by slightly modifying FairBCEM++ (Algorithm 6). Specifically, in line 23 of Algorithm 6, FairBCEMPro++ replaces the inspection for a single-side fair biclique with the inspection for a proportion single-side fair biclique which can be easily implemented. Additionally, in line 26 of Algorithm 6, we use a different algorithm, called CombinationPro, instead of Combination, to enumerate proportion single-side fair bicliques. The workflow of CombinationPro is similar to that of Combination, and the difference is that CombinationPro calculates $csize$ by $min(S_{a_i}, msize+\delta, msize * \frac{(1-\theta)}{\theta})$ (line 5 in Algorithm 7). The third item comes from the proportion constraint which can be easily derived by the inequality $\frac{msize}{msize+csize} \geq \theta$. Due to the space limit, we omit the pseudo-codes of FairBCEMPro++ and CombinationPro.

IV. Bi-side fair biclique enumeration

This section first revises the pruning techniques for solving the single-side fair biclique enumeration problem to fit into our bi-side fair biclique enumeration problem. Then, we propose an algorithm, called BFairBCEM, by extending FairBCEM to enumerate all fair bi-side fair bicliques. Similarly, we also propose an algorithm called BFairBCEM++ by extending the FairBCEM++ algorithm. Finally, we present the BFairBCEMPro++ algorithm to solve PBSFBC enumeration problem by adapting the BFairBCEM++ algorithm.

A. The pruning techniques

In single-side fair biclique enumeration, we derive two pruning techniques by considering the attribute degrees of vertices on the fair side (i.e., the lower side V). In the bi-side fair biclique model, the attribute constraint is expanded to both the upper side and lower side, thus a natural idea is to employ the attribute degrees of vertices in U and V to design the pruning methods. Below, we give two pruning techniques, namely, BFCore and BCFCore, which are variants of FCore and CFCore, respectively.

Bi-fair α-β core pruning (BFCore). Similar to FCore, we introduce the concept of bi-fair α-β core as Definition 13 and derive the Lemma 3 to prune vertices in both U and V that are definitely not in any bi-side fair biclique.

Definition 13: (Bi-fair α-β core) Given an attributed bipartite graph $G=(U,V,E,A)$, a subgraph $H=(L,R,E,A)$ is a bi-fair α-β core if (1) $D_{a_i}(u,H) \geq \beta, u \in L, a_i \in A(V)$; (2) $D_{a_i}(v,H) \geq \alpha, v \in R, a_i \in A(U)$; (3) there is no subgraph $H' \supset H$ that satisfies (1) and (2) in G.

Lemma 3: Given an attributed bipartite graph $G = (U,V,E,A)$ and two integers α, β, any bi-side fair biclique must be contained in a bi-fair α-β core.

With Lemma 3, a question is how to calculate the bi-fair α-β core of a bipartite graph G. We devise a peeling algorithm, called BFCore, by slightly modifying FCore (Algorithm 1), as Definition 13 is also a variant of the classic k-core [2], [19]. Specifically, for each vertex v in V, BFCore calculates the attribute degree $D_{a_i^U}(v)$ instead of the degree $D(v)$ (lines 2-6). When a vertex u is removed, the algorithm updates the attribute degrees for its neighbors and maintains the priority queue Q. If a neighbor v is in the lower side V, BFCore calculates the new attribute degree $D_{a_i^U}(v)$ as it is in the upper side U (lines 16-19). The other steps of BFCore are similar to those of FCore and thus we omit the pseudo-code of BFCore.

Bi-colorful fair α-β core pruning (BCFCore). In CFCore, we construct the 2-hop graph on the fair side V by adding an edge for two vertices with at least α common neighbors (i.e., the condition (1) in Definition 3). While the bi-side fair biclique model considers the fairness on both U and V. Thus, when building the 2-hop graph on V, we only add an edge for two vertices if they share at least α common neighbors for each attribute value $a_i^U \in A(U)$ (i.e., the condition (1) in Definition 4). Here, we revise the 2-hop graph algorithm to fit the bi-side fair biclique enumeration problem, which is outlined in Algorithm 8. In the graph constructed by BiConstruct2HopGraph, we can still calculate the ego colorful β-core to prune the unpromising vertices in V.

Algorithm 8: BiConstruct2HopGraph

Input: $G=(U,V,E,A)$, a integer α, the fair side V
Output: The 2-hop graph H based on the fair side V

```
Let H = (V = G(V), E = ∅, A = A_V) be an attributed graph;
for v ∈ G(V) do
    C is an array with C[i][j] = 0, 1 ≤ i ≤ |G(V)|, 1 ≤ j ≤ |A(U)|;
    for u ∈ N(v, G) do
        for w ∈ N(u, G) do
            if w ≠ v then C[w][w.val] ← C[w][w.val] + 1;
    for u ∈ G(V) do
        if ∀a_i^U ∈ A(U), C[u][a_i^U] ≥ α and u < v then
            E(H) ← E(H) ∪ (u, v);
return H;
```

Algorithm 9: BFairBCEM

Input: A bipartite graph $G=(U,V,E,A)$, three integers α, β, δ
Output: The set of all bi-side fair bicliques Res

```
1 Ĝ = (Û, V̂, Ê, A) ← BCFCore(G, α, β);
2 L ← Û; R ← ∅; P ← V̂; Q ← ∅;
3 Enumerate all single-side fair bicliques by FairBCEM(Ĝ, α, β, δ);
4 for each single-side fair biclique B(L', R') do
5     L' ← Combination(L', A(U), α, δ);
6     for l' ∈ L' do
7         if R' is a maximal fair subset of N(l') then
8             Res ← Res ∪ (l', R');
9 return Res;
```

In addition, the bi-side fair biclique model also requires fairness on the upper side U, and thus we can prune the vertices in U like handling the lower side V. Based on this idea, we propose the BCFCore algorithm which is similar to FCore and we only make the following minor changes. In particular, for the lower side V, BCFCore constructs the 2-hop graph by BiConstruct2HopGraph instead of Construct2HopGraph (line 3 in Algorithm 2), and computes the ego colorful β-core to prune the vertices in V. And for the upper side U, BCFCore again builds the 2-hop graph by BiConstruct2HopGraph with parameters (G, β, U), and calculates the ego colorful α-core to prune the unpromising vertices in U. Due to the space limitation, we omit the pseudo-code of BCFCore.

B. The BFairBCEM *algorithm*

Before introducing our BFairBCEM algorithm, we first give the following observation.

Observation 6: A bi-side fair biclique must be contained in single-side fair bicliques.

With Observation 6, we present the BFairBCEM algorithm as shown in Algorithm 9. We first search all single-side fair bicliques and then enumerate all bi-side fair bicliques by combination of the upper side. Specifically, BFairBCEM invokes FairBCEM to search all single-side fair bicliques (line 3). Given a single-side fair biclique $B(L', R')$, it satisfies the fairness restriction on the lower side, and we enumerate all maximal fair subsets of L' in the upper side to ensure fairness by the Combination algorithm (line 5). For a maximal fair subset of l' in $\mathcal{L}'$, the BFairBCEM algorithm determines whether R' is a maximal subset of $N(l')$ (line 7). Clearly, if yes, (l', R') is a bi-side fair biclique and we add it into Res. As all bi-side fair bicliques are contained in all single-side fair bicliques based on Observation 6. The BFairBCEM algorithm correctly returns all bi-side fair bicliques.

Correctness analysis. All single-side fair bicliques are correctly enumerated by FairBCEM and any bi-side fair biclique must be included in a single-side fair biclique, so the completeness is satisfied. The maximality is met by the line 7 of Algorithm 9, since l' is a maximal fair subset of $N(R)$ and R' is a maximal fair subset of $N(l')$, which also verifies the fairness restriction. For non-redundancy, it is obviously that any bi-side fair biclique enumerated in a single-side fair biclique has the same R', and there is no two different single-side fair bicliques has the same R, thus any bi-side fair biclique is enumerated once.

C. The BFairBCEM++ algorithm

Based on Observation 6, we can also invoke the FairBCEM++ algorithm to search all single-side fair bicliques and then enumerate all bi-side fair bicliques by the combinatoral enumeration method. Hence, we propose the BFairBCEM++ algorithm which can be easily devised by slightly modifying Algorithm 9. That is, we use FairBCEM++ instead of FairBCEM in line 3 to find all single-side fair bicliques. Due to the space limitation, we omit the pseudo-code of BFairBCEM++.

Extending to finding all PBSFBCs. We can slightly adapt the BFairBCEM++ algorithm to solve PBSFBC enumeration problem, which is called BFairBCEMPro++. That is, we replace Combination with CombinationPro (line 5 in Algorithm 9), and use the inspection for a PBSFBC instead of that for a BSFBC (lines 3-4 in Algorithm 9). It is worth noting that we also need to check whether the ratio constraint is satisfied for maximal fair subset checking (line 7 in Algorithm 9). We omit the details of BFairBCEMPro++ due to the space limit.

V. Experiments

A. Experimental setup

For single-side fair biclique enumeration problem, we implement FairBCEM (Algorithm 5) and FairBCEM++ (Algorithm 6) equipped with the pruning techniques FCore (Algorithm 1) and CFCore (Algorithm 2). To enumerate all bi-side fair bicliques, the BFairBCEM (Algorithm 9) and BFairBCEM++ are implemented armed with the BFCore and BCFCore pruning techniques. For comparison, we implement two naive search algorithms, i.e., NSF and BNSF, to find all SSFBCs and BSFBCs, which reserve the pruning techniques such as Algorithm 1 and Algorithm 2 and drop off all pruning techniques in the search process such as Observation 2, Observation 4 and Observation 5. We also implement the above enumeration algorithms with two different vertex selection orderings, i.e., DegOrd and IDOrd, which are obtained by sorting the vertices based on a non-increasing manner of their degrees and IDs respectively. All algorithms are implemented in C++. We conduct all experiments on a PC with a 2.10GHz Inter Xeon CPU and 256GB memory. We set the time limit for all algorithms to 24 hours, and use the symbol "INF" to denote that the algorithm cannot terminate within 24 hours.

Datasets. We evaluate the efficiency of the proposed algorithms in five real-world graphs. Specifically, Wiki-catis a feature network. Youtube, IMDB are affiliation networks, Twitter is an interaction network and DBLP is an authorship network. All datasets can be downloaded from http://konect.cc/. Note that all these datasets are non-attributed bipartite graphs, thus we randomly assign an attribute to each vertex to generate attributed graphs for evaluating the efficiency of all algorithms.

Parameters. There are four parameters in our algorithms: α, β, δ and θ. α and β are used to restrict the size of fair bicliques. If α and β are too small, we will obtain too many small bicliques which are not meaningful. When α and β are too large, most of the vertices will be pruned during the pruning processing and the remaining graph will miss much structural information, resulting in few bicliques being outputted. We carefully fine-tune them to extract meaningful fair bicliques based on the biclique numbers in real-life datasets. δ represents the maximum difference between the number of vertices of every attribute. With δ increases, the fairness between different attributes in vertex set decreases. Therefore, δ should not be set to be too large or the problem will degenerate to the maximal biclique enumeration problem. The parameter θ is the fairness ratio threshold and we can easily derive that θ is no larger than 0.5. Thus, θ also should not be set to be too large. Since different datasets have various scales, the parameter α and β is set within different integers. For SSFBC (PSSFBC) and BSFBC (PBSFBC) enumeration problems, we also set parameters within different integers. The detailed parameter settings can be found on the website https://github.com/Heisenberg-Yin/fairnesss-biclique.

TABLE I
DATASETS AND PARAMETERS

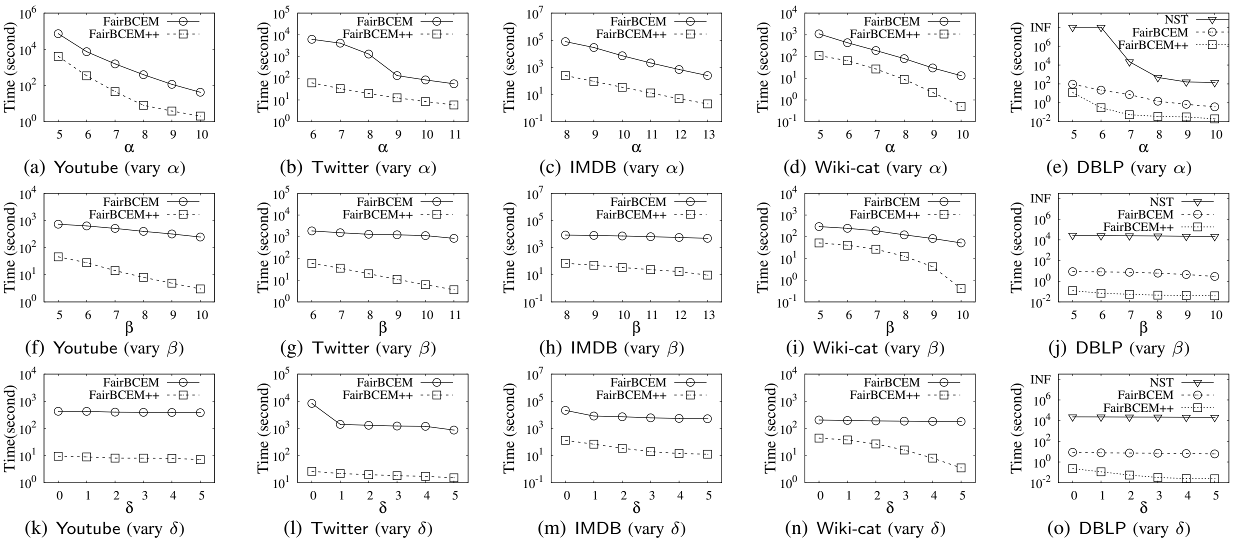
Dataset	$\|U\|$	$\|V\|$	$\|E\|$	Density	α^{*s}	β^{*s}	α^{*b}	β^{*b}	δ^*	θ^*
Youtube	$94,238$	$30,087$	$293,360$	1.0×10^{-4}	8	8	5	5	2	0.4
Twitter	$175,214$	$530,418$	$1,890,661$	2.0×10^{-5}	8	8	6	7	2	0.4
IMDB	$303,617$	$896,302$	$3,782,463$	1.4×10^{-5}	10	10	6	6	2	0.4
Wiki-cat	$1,853,493$	$182,947$	$3,795,796$	1.1×10^{-5}	7	7	6	6	2	0.4
DBLP	$1,953,085$	$5,624,219$	$12,282,059$	1.1×10^{-6}	7	7	4	4	2	0.4

Note: α^{*s}, β^{*s} and α^{*b}, β^{*b} are the default values of α, β for SSFBC (PSSFBC) and BSFBC (PBSFBC) models respectively, δ^*, θ^* are the default values of δ and θ.

B. Efficiency testing

Exp-1: Evaluation of the pruning techniques. For single-side fair biclique enumeration problem, both FairBCEM and FairBCEM++ algorithms can use FCore and CFCore to prune unpromising nodes. For bi-side fair biclique enumeration problem, the pruning techniques BFCore and BCFCore can reduce the graph size in BFairBCEM and BFairBCEM++. In this experiment, we evaluate these pruning techniques by comparing the number of remaining vertices after pruning and the consuming time with varying α and β. Fig. 3 and Fig. 4 illustrate the results for single-side fair biclique and bi-side fair biclique enumeration on IMDB, respectively. The results on the other datasets are consistent. Fig. 3 (a)-(b) show that both FCore and CFCore can significantly reduce the number of vertices compared to the original graph as expected. Moreover, the number of remaining vertices decreases with larger α or β. In general, CFCore outperforms FCore in terms of the pruning performance, especially for relatively small α or β values. As shown in Fig. 3 (c)-(d), the running time of FCore and CFCore decreases as α or β increases and CFCore takes more time than FCore to prune unpromising vertices. This is because CFCore performs FCore first and further reduces the graph by ego fair α-β core pruning in 2-hop graph (Algorithm 2). For example, in Fig. 3(a) with $\alpha = 8$, FCore reduces the number of vertices from 9,266,649 to 12,507; and CFCore further reduces the number of vertices to 1,318. When β equals 8, the number of remaining vertices after FCore and CFCore are 13,757 and 1,490 respectively as shown in Fig. 3(b). As a result, the CFCore pruning can achieve superior pruning effect over the FCore with slightly time consuming. Besides, similar results can also be found in Fig. 4 for bi-side fair biclique enumeration. To sum up, the above experimental results validate the effectiveness and efficiency of the FCore, CFCore, BFCore and BCFCore pruning techniques.

Exp-2: Evaluation of SSFBC enumeration algorithms. Here we evaluate FairBCEM and FairBCEM++ algorithms equipped with descending DegOrd by varying α, β and δ. The results are depicted in Fig. 2. As expected, the runtime of FairBCEM and FairBCEM++ decreases with increasing α, β, δ on all datasets. This is because for a large α, β, many vertices can be pruned by the FCore and CFCore pruning techniques and the search space can also be correspondingly reduced during the branch and bound procedure. For a large δ, the number of single-side fair bicliques decreases with increasing δ due to the maximality constraint, thus resulting in a trend of decreasing time. Moreover, we can also see that the runtime of FairBCEM++ is at least two orders of magnitude lower than that of FairBCEM within all parameter settings over all datasets. For instance, when $\alpha = 10$ with default β and δ, FairBCEM consumes 29,192 seconds to find all single-side fair bicliques on IMDB, while FairBCEM++ takes only 91 seconds to output the results, which is almost three orders of magnitude faster than the FairBCEM algorithm. These results validate the efficiency of the proposed FairBCEM and FairBCEM++ algorithms.

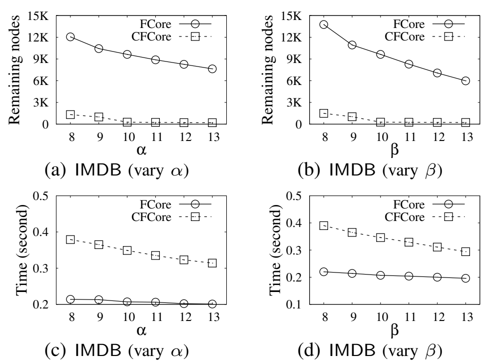


(a) Youtube (vary α) (b) Twitter (vary α) (c) IMDB (vary α) (d) Wiki-cat (vary α) (e) DBLP (vary α)

(f) Youtube (vary β) (g) Twitter (vary β) (h) IMDB (vary β) (i) Wiki-cat (vary β) (j) DBLP (vary β)

(k) Youtube (vary δ) (l) Twitter (vary δ) (m) IMDB (vary δ) (n) Wiki-cat (vary δ) (o) DBLP (vary δ)

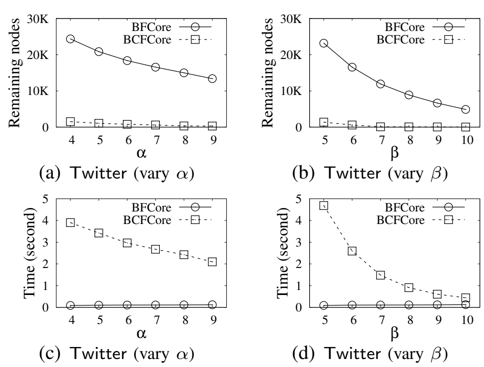
Fig. 2. The running time of the NSF, FairBCEM and FairBCEM++ algorithms in different datasets.



(a) IMDB (vary α) (b) IMDB (vary β)

(c) IMDB (vary α) (d) IMDB (vary β)

Fig. 3. The pruning time and remaining nodes of FCore and CFCore.



(a) Twitter (vary α) (b) Twitter (vary β)

(c) Twitter (vary α) (d) Twitter (vary β)

Fig. 4. The pruning time and remaining nodes of BFCore and BCFCore.

In FairBCEM and FairBCEM++ algorithms, a vertex is selected from the candidate set to the current biclique for performing a backtracking search procedure. Since the search spaces with various orderings are significantly different, we also evaluate the two algorithms with IDOrd and DegOrd orderings. Table.II depicts the runtime of FairBCEM and FairBCEM++ equipped with IDOrd and DegOrd in the case of default α, β, δ over all datasets. As shown in Table.II, the FairBCEM with DegOrd is significantly faster than that with IDOrd. For example, in IMDB, the FairBCEM algorithms with IDOrd and DegOrd consume 4,378 seconds and 2,098 seconds to output all single-side fair bicliques. Clearly, the latter is almost 2 times faster than the former. Similar results can also be found for FairBCEM++ algorithms with IDOrd and DegOrd. Again, the FairBCEM++ algorithm outperforms FairBCEM on all datasets, which is consistent with our previous founding. The results indicate that the DegOrd ordering is more efficient that the IDOrd ordering during the search procedure.

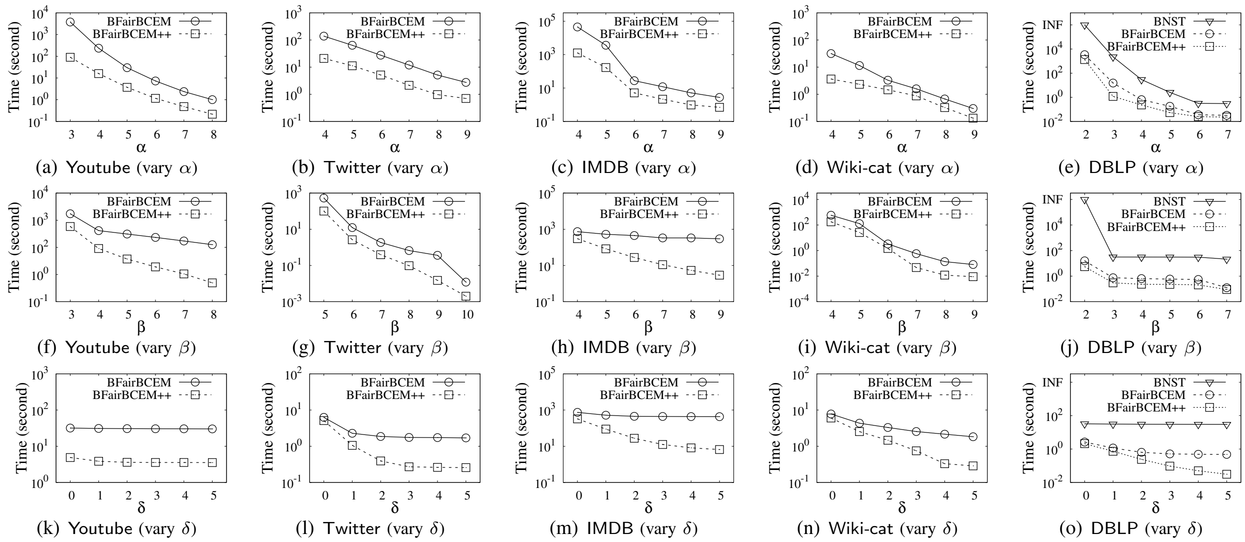
TABLE II
THE RUNTIME OF DIFFERENT ALGORITHMS WITH IDOrd AND DegOrd.

Algorithm (s)	Ordering	IMDB	Youtube	Twitter	Wiki-cat	DBLP
FairBCEM	IDOrd	7,022.7	157.1	854.2	90.6	6.3
	DegOrd	1,612.9	43.6	611.8	45.9	2.6
FairBCEM++	IDOrd	78.6	16.1	72.5	13.2	0.6
	DegOrd	61.9	8.3	65.1	12.4	0.5
BFairBCEM	IDOrd	174.2	2.3	76.8	0.9	1.5
	DegOrd	68.1	1.4	69.1	0.4	1.1
BFairBCEM++	IDOrd	19.8	7.4	63.8	0.3	0.7
	DegOrd	17.2	1.7	59.7	0.2	0.6

In addition, We compare NSF with the proposed FairBCEM and FairBCEM++ on all datasets. We only show the results on DBLP in Fig. 2 as NSF runs out of time on other datasets with most parameter settings. As can be seen, FairBCEM is at least two orders of magnitude faster than NSF. These results confirm that our proposed algorithms significantly outperform the NSF algorithm.

Exp-3: Evaluation of BSFBC **enumeration algorithms.** We evaluate the runtime of BFairBCEM and BFairBCEM++ with DegOrd by varying α, β, δ. The results are depicted in Fig. 5. As expected, the runtime of BFairBCEM and BFairBCEM++ decreases as α, β, δ increases, which is similar to that of single-side fair biclique enumeration algorithms. Moreover, we also observe that the BFairBCEM++ algorithm is almost 3-100 times faster than the BFairBCEM algorithm within all parameter settings on all datasets. For example, when $\beta = 7$ with default α and δ, the runtime of BFairBCEM and BFairBCEM++ take 17 seconds and 1 second to output all bi-side fair bicliques on Youtube, respectively. Obviously, the former is significantly faster than the latter. These results validate the efficiency of the proposed BFairBCEM and BFairBCEM++ algorithms.

In addition, we compare the running time of BFairBCEM and BFairBCEM++ algorithms armed with IDOrd and DegOrd under default α, β, δ. As seen in Table.II, the BFairBCEM with DegOrd significantly outperforms IDOrd by a large margin. For example, in IMDB, the BFairBCEM algorithm with IDOrd takes 253 seconds to find all bi-side fair bicliques, while the algorithm with DegOrd only needs 169 seconds. Similar results can also be found for BFairBCEM++ algorithms with IDOrd and DegOrd. Again, the BFairBCEM++ algorithm

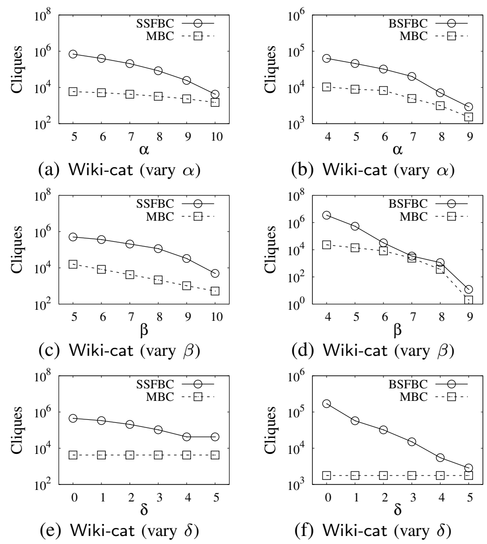


(a) Youtube (vary α) (b) Twitter (vary α) (c) IMDB (vary α) (d) Wiki-cat (vary α) (e) DBLP (vary α)

(f) Youtube (vary β) (g) Twitter (vary β) (h) IMDB (vary β) (i) Wiki-cat (vary β) (j) DBLP (vary β)

(k) Youtube (vary δ) (l) Twitter (vary δ) (m) IMDB (vary δ) (n) Wiki-cat (vary δ) (o) DBLP (vary δ)

Fig. 5. The running time of the BNSF, BFairBCEM and BFairBCEM++ algorithms on different datasets.

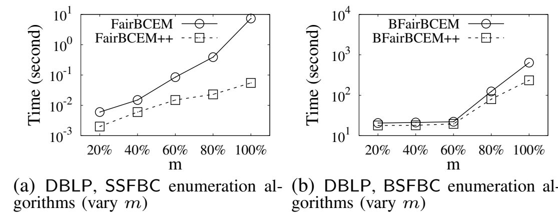


(a) Wiki-cat (vary α) (b) Wiki-cat (vary α)

(c) Wiki-cat (vary β) (d) Wiki-cat (vary β)

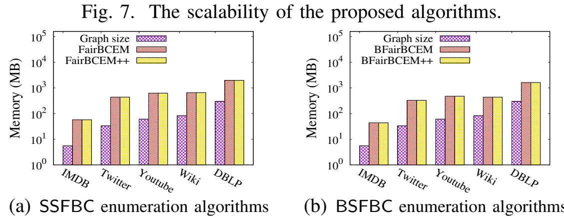
(e) Wiki-cat (vary δ) (f) Wiki-cat (vary δ)

Fig. 6. The numbers of the maximal bicliques, SSFBCs and BSFBCs.



(a) DBLP, SSFBC enumeration algorithms (vary m) (b) DBLP, BSFBC enumeration algorithms (vary m)

Fig. 7. The scalability of the proposed algorithms.



(a) SSFBC enumeration algorithms (b) BSFBC enumeration algorithms

Fig. 8. The memory overhead.

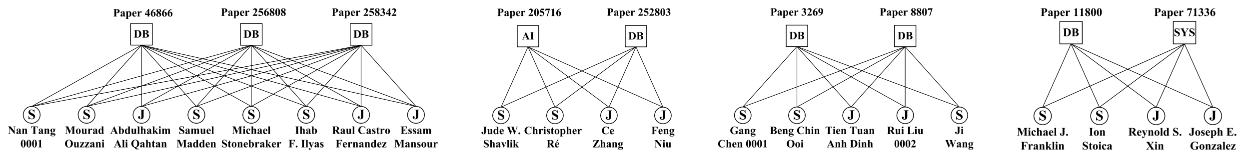
is faster than BFairBCEM over all datasets. These results also demonstrate the efficiency of DegOrd ordering which is consistent with our previous findings.

Besides, we also evaluate the running time of BNSF with BFairBCEM++ and BFairBCEM++ on all datasets. We show the results on DBLP in Fig. 5 as NSF cannot terminate with limited time on other datasets under parameter settings. We can see that BFairBCEM is at least two orders of magnitude faster than BNSF. These results confirm that our algorithms are significantly faster than the algorithm.

Exp-4: The number of SSFBCs and BSFBCs. Fig. 6 reports the number of single-side fair bicliques and bi-side fair bicliques with varying α, β, δ on Wiki-cat. Note that we find the maximal biclique $B(L, R)$ satisfying $|L| \geq \alpha$ and $|R| \geq 2 \times \beta$ for comparison with single-side fair biclique. To compare with bi-side fair biclique, we search the maximal biclique $B(L, R)$ with $|L| \geq 2 \times \alpha$ and $|R| \geq 2 \times \beta$. Clearly, there are significant numbers of single-side fair bicliques and bi-side fair bicliques on Wiki-cat. For example, in the case of $\alpha = 6, \beta = 6, \delta = 2$ for single-side fair biclique enumeration problem, there are 9,548 maximal bicliques, 346,411 single-side fair bicliques. As the case of $\alpha = 3, \beta = 6, \delta = 2$ for bi-side fair biclique enumeration problem, there are 546,411 bi-side fair bicliques, and 9,548 maximal biclique. In general, the number of single-side fair bicliques and bi-side fair bicliques is larger than that of maximal bicliques. This finding is consistent with our analysis in Section II, because any single-side fair biclique or bi-side fair biclique must be included in a maximal biclique. Additionally, we can see that the number of maximal bicliques, single-side fair bicliques and bi-side fair bicliques decreases as α, β, δ increases. This is because with a larger $\alpha/\beta/\delta$, the fairness constraint and size constraint become stricter for single-side fair biclique/single-side fair biclique models and maximal biclique model respectively.

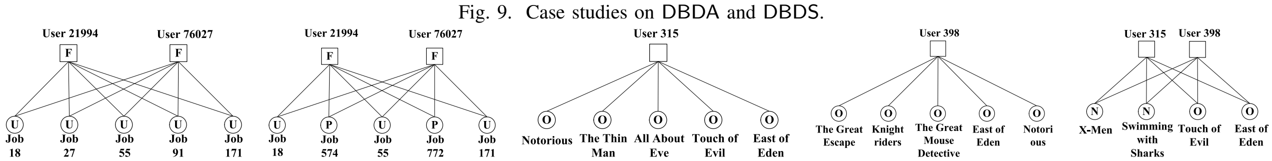
Exp-5: Scalability testing. Here we evaluate the scalability of the proposed algorithms. To this end, we generate four subgraphs for each dataset by randomly picking 20%-80% of the edges, and evaluate the runtime of the algorithms for single-side fair biclique enumeration and bi-side fair biclique enumeration. Fig. 7 illustrates the results on DBLP and the results on the other datasets are similar. For the SSFBC enumeration algorithms, as show in Fig. 7(a), the runtime of FairBCEM increases smoothly as the graph size increases. while the runtime of FairBCEM++ keeps relatively stable with different values of m. Again, FairBCEM++ is at least 10 times faster than FairBCEM with all parameter settings, which is consistent with our previous findings. For the SSFBC enumeration algorithms, as can be seen from Fig. 7(b), the runtime of BFairBCEM++ increases more smoothly w.r.t. the graph size than that of BFairBCEM. These results demonstrate the high scalability of the proposed algorithms.

Exp-6: Memory overhead. Fig. 8 shows the memory over-



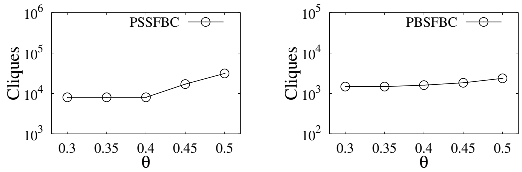
(a) DBDA, SSFBC, ($\alpha = 3, \beta = 3, \delta = 2$) (b) DBDA, BSFBC, ($\alpha = 1, \beta = 2, \delta = 2$) (c) DBDS, SSFBC, ($\alpha = 2, \beta = 2, \delta = 2$) (d) DBDS, BSFBC, ($\alpha = 1, \beta = 2, \delta = 2$)

Fig. 9. Case studies on DBDA and DBDS.



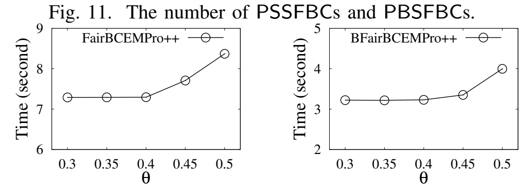
(a) Jobs, the CF algorithm (b) Jobs, SSFBC, ($\alpha = 2, \beta = 2, \delta = 1$) (c) Movies, the CF algorithm (d) Movies, the CF algorithm (e) Movies, SSFBC, ($\alpha = 2, \beta = 2, \delta = 1$)

Fig. 10. Case studies on Jobs and Movies.



(a) Youtube (vary θ) (b) Youtube (vary θ)

Fig. 11. The number of PSSFBCs and PBSFBCs.



(a) Youtube, FairBCEMPro++ algorithm (vary θ) (b) Youtube, BFairBCEMPro++ algorithm (vary θ)

Fig. 12. The running time of FairBCEMPro++ and BFairBCEMPro++.

heads of the enumeration algorithms on all datasets. Note that the memory costs of different algorithms do not include the size of the graph. From Fig. 8, we can see that the memory usages of FairBCEM and FairBCEM++ are almost equal and are always larger than the original graph size. This is because they both perform the CFCore pruning technique and enumerate single-side fair bicliques following a depth-first manner, thus the space overhead mainly depends on the data structures in CFCore. These results are consistent with our analysis in Section III-C. Similar results can also be found for BFairBCEM and BFairBCEM++ algorithms.

Exp-7: Evaluation of PSSFBC and BSFBC enumeration algorithms. Here we evaluate the FairBCEMPro++ and BFairBCEMPro++ algorithms by varying the additional parameter θ. Fig. 11 and Fig. 12 illustrate the number of PSSFBCs and PBSFBCs and the running time of FairBCEMPro++ and BFairBCEMPro++ on Youtube. The results on the other datasets are similar. As can be seen, the number of proportion fair bicliques and the runtime increase with the increasing θ. When $\theta = 0.5$, the PSSFBC enumeration problem degenerates to the SSFBC enumeration problem with $\delta = 0$. Therefore, solving the PSSFBC enumeration problem takes a similar time as the SSFBC enumeration problem. The case is also similar to the PBSFBC enumeration problem. When θ approaches 0.5, more bicliques satisfy the definitions of proportion fair bicliques, thus the number of PSSFBCs and PBSFBCs increases, and the running time of algorithms also increases.

C. Case study

Case study on DBLP. We conduct a case study on a collaboration network DBLP to show the effectiveness of our algorithms. The DBLP dataset is downloaded from dblp.uni-trier.de/xml/. We construct a bipartite graph on DBLP by defining two type nodes, that is, the papers are on the upper side and the scholars are on the lower side. When a scholar is an author of a paper, there is an edge between them. Based on DBLP, We further construct two attributed bipartite subgraphs: DBDA and DBDS as follows. For DBDA, we keep the scholars that have published at least one paper on the database (DB), and artificial intelligence (AI) related conferences. Each scholar has an attribute A_V with $A(V) = \{S, J\}$ where S represents a senior scholar and J indicates a junior scholar. We assign the attribute value for a scholar v by identifying whether he/she has published papers for over 10 years. If yes, we set $v.val$ to S otherwise the $v.val$ is J. Every paper is associated with an attribute A_U with $A(U) = \{DB, AI\}$ to indicate that this paper is published in DB and AI related conferences. For DBDS, we only remain the scholars that have published at least one paper on the database (DB), and system (SYS) related conferences. Each scholar also has an attribute A_V with $A(V) = \{S, J\}$ and We assign the attribute value for scholars by the method for DBDA. Each paper has an attribute A_U with $A(U) = \{DB, SYS\}$ to indicate that this paper is published in DB and SYS related conferences. Finally, the DBDA has 260,605 papers and 240,420 scholars with 781,378 edges, i.e., $|U| = 240,420$ and $|V| = 260,605$. And the DBDS contains 163,545 papers and 139,703 scholars with 433,928 edges, i.e., $|U| = 163,545$ and $|V| = 139,703$. We perform FairBCEM++ and BFairBCEM++ algorithms to find all single-side fair bicliques and bi-side fair bicliques.

As examples, Fig. 9 (a)-(b) and Fig. 9 (c)-(d) show one single-side fair biclique and one bi-side fair biclique on DBDA and DBDS respectively. We do not illustrate the title of papers since the title is too long. In Fig. 9(a), we can see that there are five senior scholars and three junior scholars, which is clearly a single-side fair biclique of DBDA with $\alpha = 3, \beta = 3, \delta = 2$. From their homepages, all scholars in Fig. 9(a) are interested in database-related areas, which is consistent with the attributes of papers they connected. The senior authors, such as Michael Stonebraker and Samuel Madden are indeed well-known scholars in the field of the database. This result indicates that our FairBCEM++ can find single-side fair bicliques which guarantee the fairness of one side in real-world applications. The bipartite in 9(b) is a bi-side fair bicliquewhich contains two senior scholars and two junior scholars in the lower side and one AI paper [40] and one DB paper [20] in the upper side. Moreover, the professors Christopher Ré and Jude W. Shavlik are databases and artificial intelligence scientists, and Ce Zhang is relatively young compared with the former two scholars who are students of Christopher Ré. This result confirms that the proposed BFairBCEM++ indeed can find bi-side fair bicliques to ensure the fairness of two sides in real-world graphs. Similar results can also be found on DBDS. Fig. 9(c) depicts a single-side fair biclique with five senior scholars and three junior scholars. In Fig. 9(d), there are two senior scholars and two junior scholars who have co-authored one DB paper [36] published in SIGMOD and one SYS paper [12] published in OSDI. Among all scholars, the professors Michael Frankin and

Ion Stoica are also well-known in data science and distributed systems areas. These results demonstrate the effectiveness of single-side fair biclique and bi-side fair biclique models and our proposed algorithms.

Case study on Jobs**.** We use a job recommendation dataset Jobs to conduct a case study which can be downloaded from https://www.kaggle.com/competitions/job-recommendation. The dataset consists of 7 windows, and we consider window 1 for simplicity as each window is independent. We construct a bipartite graph G by defining two type nodes, i.e., the user on the upper side and the job on the lower side. The attribute of jobs is popularity, which is set based on the number of applications for this position. In order to avoid cold start problem, we only reserve the top-1000 jobs with the highest number of applications and assign the top-500 jobs as more popular jobs (the attribute is P) and the others as less popular jobs (the attribute is U). We also assign each user an attribute value A or F to represent he/she is American or foreigner. Therefore, the bipartite graph G contains 63,412 users and 1,000 jobs with $A_V = \{P, U\}$ and $A_U = \{A, F\}$. We use the Collaborative Filtering (CF) algorithm to calculate recommendation results which is shown in Fig. 10(a). In Fig. 10(a), there is an edge between a user and a job if the job lies in the top-5 recommendation jobs with the CF algorithm. From the information of Jobs, we can find that user 21,994 comes from India and has a master's degree with 9 years of work experience, and user 76,027 is a Canadian and has a master's degree with 23 years of work experience. Clearly, the two foreigners have similar education and work experience, but all the jobs recommended for them are less popular jobs. To eliminate the biases, we construct a bipartite graph G' in which each edge represents that the job has the top-10 highest recommendation score computed by CF, i.e, G contains 63,412 users, 1,000 jobs and 63,4120 edges. Then we perform FairBCEM++ to find SSFBCs by setting the jobs as the fair side. A SSFBC containing user 21,994 and user 76,027 is depicted in Fig. 10(b). As expected, both more popular jobs and less popular jobs are recommended to the two foreigners. These results demonstrate the effectiveness of our fair biclique models and proposed algorithms.

Case study on Movies**.** We also conduct a case study on a movie recommendation dataset Movies which can be downloaded from https://www.kaggle.com/code/rounakbanik/movie-recommender-systems. We construct a bipartite graph including the user on the upper side and the movies on the lower side. For each movie, we assign its attribute to O to represent an old movie which is published before 1990, and otherwise, its attribute is set to N to indicate a new movie. The bipartite graph consists of 9,000 movies and 700 users, i.e., $|U| = 700$ and $|V| = 9,000$. The recommendation result by the traditional CF algorithm is shown in Fig. 10(c) and Fig. 10(d), an edge means that a movie lies in the top-5 recommendation answers for a user. As can be seen, for two users of similar interests, all five movies in Fig. 10(c) and Fig. 10(d) are old movies. The CF algorithm suffers from explosion bias, that is, already popular movies will get more chance to be recommended and relatively new movies get less recommendation chance even if they are of comparable quality, which is generally called cold start problem. To solve this problem, We connect each user with top-10 movies according to the personalized recommendation scores computed by CF and invoke FairBCEM++ to find SSFBCs. A SSFBC containing user 310 and user 512 is shown in Fig. 10(e). By introducing fairness into the movie recommendation task, the new recommended movie "X-men" is more desirable and famous compared with old movies. This result indicates that fair biclique models can relieve the problem of explosion bias.

VI. Related Work

Cohesive bipartite subgraph mining. Our work is related to cohesive subgraph mining in bipartite graphs which has attracted much attention in recent years. For example, Zhang *et al.* [41] proposed a branch and bound algorithm, i.e., MBEA, to search all maximal bicliques. To accelerate the search efficiency, Abidi *et al.* [1] further presented a pivoting enumeration algorithm called PMBE which is based on the Containment Directed Acyclic Graph (CDAG). Yang *et al.* [38] investigated the problem of (p, q)-clique counting and proposed BCList and BCList++ algorithm which applies a layer-based exploring strategy and cost model to accelerate the searching process. Lyu *et al.* [15] presented a new algorithm to search maximum bi-clique which can be used to process bipartite graphs of billion scale. Wang *et al.* [33] developed a novel index structure to help finding the (α, β)-community which is a minimum edge weight (α, β)-core. Wang *et al.* [32] proposed a vertex-priority-based paradigm BFC-VP to accelerate butterfly counting by a large margin. All the algorithms mentioned above do not consider the fairness of cohesive subgraphs and they are mainly tailored to non-attributed bipartite graphs. To the best of our knowledge, the definition of fairness-aware biclique is proposed for the first time, and also our work is the first to study the problem of finding fairness-aware biclique in bipartite graphs.

Fairness-aware data mining. Our work is inspired by a concept called fairness which has been widely studied in machine learning communities. Verma *et al.* [27] proposed many concepts to better measure fairness. Zehlike *et al.* [39] presented a method to generate a ranking with guaranteed group fairness, which can ensure the proportion of protected elements in the rank is no less than a given threshold. Serbos *et al.* [26] investigated a problem of fairness in the package-to-group recommendation, and propose a greedy algorithm to find approximate solutions. Beutel *et al.* [3] also studied the fairness in recommendation systems and presented a set of metrics to evaluate algorithmic fairness. Another line of research on fairness is studied in classification tasks. Some notable works include demographic parity [11] and equality of opportunity [13]. For instance, Hardt *et al.* [13] proposed a framework that can optimally adjust any learned predictor to reduce bias. Our definition of fairness which requires the equality of different attribute values in a group is different from those in the above studies in the machine learning literature. In the field of data mining, Pan *et al.* [21] introduced the fairness into clique model and proposed several algorithms to find fair cliques. Unlike their work, we focus on studying the fairness-aware biclique enumeration problem on bipartite graphs, and our techniques are significantly different from their techniques.

VII. Conclusion

In this paper, we study the problem of enumerating fairness-aware bi-cliques in bipartite graphs. We propose a single-side fair biclique model and a bi-side fair biclique model to introduce fairness to bipartite graphs. To enumerate all single-side fair bicliques, we first present the FCore and CFCore pruning techniques to prune unpromising vertices, and then develop a branch and bound algorithm FairBCEM to enumerate all single-side fair bicliques in the pruned graph. To improve the efficiency, we present the FairBCEM++ algorithm to search all single-side fair bicliques by using maximal cliques as candidates to reduce search space. For the bi-side fair biclique enumeration problem, we also propose BFCore and BCFCore pruning techniques and develop the BFairBCEM algorithm with a branch and bound technique. The improved algorithm, i.e., BFairBCEM++, is also presented to find all bi-side fair bicliques. We also consider the ratio of the number of vertices of each attribute to the total number of vertices and propose the proportion single-side fair biclique and proportion bi-side fair biclique models and enumeration algorithms. We conduct extensive experiments using five large real-life graphs, and the results demonstrate the efficiency, effectiveness, and scalability of the proposed solutions.

Acknowledgement

This work was partially supported by (i) National Key RD Program of China 2021YFB3301300, (ii) NSFC Grants U2241211, 62072034, U1809206, and (iii) CCF-Huawei Populus Grove Fund. Rong-Hua Li is the corresponding author of this paper.

References

[1] Aman Abidi, Rui Zhou, Lu Chen, and Chengfei Liu. Pivot-based maximal biclique enumeration. In *IJCAI*, 2020.

[2] Vladimir Batagelj and Matjaz Zaversnik. An o(m) algorithm for cores decomposition of networks. *CoRR*, cs.DS/0310049, 2003.

[3] Alex Beutel, Jilin Chen, Tulsee Doshi, et al. Fairness in recommendation ranking through pairwise comparisons. In *SIGKDD*, 2019.

[4] Alex Beutel, Jilin Chen, Tulsee Doshi, Hai Qian, Li Wei, Yi Wu, Lukasz Heldt, Zhe Zhao, Lichan Hong, Ed H. Chi, and Cristos Goodrow. Fairness in recommendation ranking through pairwise comparisons. In *KDD*, 2019.

[5] Lu Chen, Chengfei Liu, Jiajie Xu, and Jianxin Li. Efficient exact algorithms for maximum balanced biclique search in bipartite graphs. In *SIGMOD Conference*, 2021.

[6] Lu Chen, Chengfei Liu, Rui Zhou, Jiajie Xu, and Jianxin Li. Efficient maximal biclique enumeration for large sparse bipartite graphs. *Proc. VLDB Endow.*, 15(8):1559–1571, 2022.

[7] Calvin Chi, Yuting Ye, Bin Chen, and Haiyan Huang. Bipartite graph-based approach for clustering of cell lines by gene expression-drug response associations. *Bioinform.*, 37(17):2617–2626, 2021.

[8] Francesco Colace, Massimo De Santo, Luca Greco, Vincenzo Moscato, and Antonio Picariello. A collaborative user-centered framework for recommending items in online social networks. *Comput. Hum. Behav.*, 51:694–704, 2015.

[9] Eduardo Corel, Raphaël Méheust, Andrew Keith Watson, James Oscar McInerney, Philippe Lopez, and Eric Bapteste. Bipartite network analysis of gene sharings in the microbial world. *Molecular Biology and Evolution*, 35:899 – 913, 2018.

[10] Yushun Dong, Jing Ma, Chen Chen, and Jundong Li. Fairness in graph mining: A survey. *CoRR*, abs/2204.09888, 2022.

[11] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *ITCS*, 2012.

[12] Joseph E. Gonzalez, Reynold S. Xin, Ankur Dave, Daniel Crankshaw, Michael J. Franklin, and Ion Stoica. Graphx: Graph processing in a distributed dataflow framework. In *OSDI*, pages 599–613. USENIX Association, 2014.

[13] Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. In *NIPS*, 2016.

[14] William Hasenplaugh, Tim Kaler, Tao B. Schardl, and Charles E. Leiserson. Ordering heuristics for parallel graph coloring. In *SPAA*, 2014.

[15] Bingqing Lyu, Lu Qin, Xuemin Lin, Ying Zhang, Zhengping Qian, and Jingren Zhou. Maximum biclique search at billion scale. *Proc. VLDB Endow.*, 13(9):1359–1372, 2020.

[16] Hanchao Ma, Sheng Guan, Christopher Toomey, and Yinghui Wu. Diversified subgraph query generation with group fairness. In *WSDM*, pages 686–694. ACM, 2022.

[17] Ziyi Ma, Yuling Liu, Yikun Hu, Jianye Yang, Chubo Liu, and Huadong Dai. Efficient maintenance for maximal bicliques in bipartite graph streams. *World Wide Web*, 25(2):857–877, 2022.

[18] Pasin Manurangsi. Inapproximability of maximum biclique problems, minimum k-cut and densest at-least-k-subgraph from the small set expansion hypothesis. *Algorithms*, 11(1):10, 2018.

[19] David W. Matula and Leland L. Beck. Smallest-last ordering and clustering and graph coloring algorithms. *J. ACM*, 30(3):417–427, 1983.

[20] Feng Niu, Ce Zhang, Christopher Ré, and Jude W. Shavlik. Scaling inference for markov logic via dual decomposition. In Mohammed Javeed Zaki, Arno Siebes, Jeffrey Xu Yu, Bart Goethals, Geoffrey I. Webb, and Xindong Wu, editors, *ICDM*, 2012.

[21] Minjia Pan, Rong-Hua Li, Qi Zhang, Yongheng Dai, Qun Tian, and Guoren Wang. Fairness-aware maximal clique enumeration. *ICDE*, 2021.

[22] J Pardalos and M Resende. On maximum clique problems in very large graphs. *DIMACS series*, 50:119–130, 1999.

[23] Erich Prisner. Bicliques in graphs i: Bounds on their number. *Combinatorica*, 20(1):109–117, 2000.

[24] Seyed-Vahid Sanei-Mehri, Ahmet Erdem Sariyüce, and Srikanta Tirthapura. Butterfly counting in bipartite networks. In *KDD*, 2018.

[25] Govind S. Sankar, Anand Louis, Meghana Nasre, and Prajakta Nimbhorkar. Matchings with group fairness constraints: Online and offline algorithms. *CoRR*, abs/2105.09522, 2021.

[26] Dimitris Serbos, Shuyao Qi, Nikos Mamoulis, Evaggelia Pitoura, and Panayiotis Tsaparas. Fairness in package-to-group recommendations. In *WWW*, 2017.

[27] Sahil Verma and Julia Rubin. Fairness definitions explained. In *FairWare*, 2018.

[28] Haibo Wang, Chuan Zhou, Jia Wu, Weizhen Dang, Xingquan Zhu, and Jilong Wang. Deep structure learning for fraud detection. In *ICDM*, 2018.

[29] Jia Wang, Ada Wai-Chee Fu, and James Cheng. Rectangle counting in large bipartite graphs. In *IEEE International Congress on Big Data*, 2014.

[30] Jun Wang, Arjen P. de Vries, and Marcel J. T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *SIGIR*, 2006.

[31] Kai Wang, Xuemin Lin, Lu Qin, Wenjie Zhang, and Ying Zhang. Vertex priority based butterfly counting for large-scale bipartite networks. *Proc. VLDB Endow.*, 12(10):1139–1152, 2019.

[32] Kai Wang, Xuemin Lin, Lu Qin, Wenjie Zhang, and Ying Zhang. Vertex priority based butterfly counting for large-scale bipartite networks. *Proc. VLDB Endow.*, 12(10):1139–1152, 2019.

[33] Kai Wang, Wenjie Zhang, Xuemin Lin, Ying Zhang, Lu Qin, and Yuting Zhang. Efficient and effective community search on large-scale bipartite graphs. In *ICDE*, 2021.

[34] Yiyuan Wang, Shaowei Cai, and Minghao Yin. New heuristic approaches for maximum balanced biclique problem. *Information Sciences*, 432:362–375, 2018.

[35] Shiwen Wu, Wentao Zhang, Fei Sun, and Bin Cui. Graph neural networks in recommender systems: A survey. *CoRR*, abs/2011.02260, 2020.

[36] Reynold S. Xin, Joseph E. Gonzalez, Michael J. Franklin, and Ion Stoica. Graphx: a resilient distributed graph system on spark. In *GRADES*, page 2. CWI/ACM, 2013.

[37] Xiaohan Xing, Fan Yang, Hang Li, Jun Zhang, Yu Zhao, Mingxuan Gao, Junzhou Huang, and Jianhua Yao. Multi-level attention graph neural network based on co-expression gene modules for disease diagnosis and prognosis. *Bioinform.*, 38(8):2178–2186, 2022.

[38] Jianye Yang, Yun Peng, and Wenjie Zhang. (p, q)-biclique counting and enumeration for large sparse bipartite graphs. *Proc. VLDB Endow.*, 15(2):141–153, 2021.

[39] Meike Zehlike, Francesco Bonchi, Carlos Castillo, Sara Hajian, Mohamed Megahed, and Ricardo Baeza-Yates. Fa* ir: A fair top-k ranking algorithm. In *CIKM*, 2017.

[40] Ce Zhang, Feng Niu, Christopher Ré, and Jude W. Shavlik. Big data versus the crowd: Looking for relationships in all the right places. In *ACL*, 2012.

[41] Yun Zhang, Charles A. Phillips, Gary L. Rogers, Erich J. Baker, Elissa J. Chesler, and Michael A. Langston. On finding bicliques in bipartite graphs: a novel algorithm and its application to the integration of diverse biological data types. *BMC Bioinform.*, 15:110, 2014.

[42] Alexander Zhou, Yue Wang, and Lei Chen. Butterfly counting on uncertain bipartite networks. *Proc. VLDB Endow.*, 15(2):211–223, 2021.

[43] Xingquan Zhu, Haicheng Tao, Zhiang Wu, Jie Cao, Kristopher Kalish, and Jeremy Kayne. *Fraud Prevention in Online Digital Advertising*. Springer Briefs in Computer Science. 2017.