Efficient Maximum Fair Clique Search over Large Networks

Qi Zhang[†], Rong-Hua Li[†], Zifan Zheng[†], Hongchao Qin[†], Ye Yuan[†], Guoren Wang[†]

[†]*Beijing Institute of Technology, Beijing, China;*
qizhangcs@bit.edu.cn; lironghuabit@126.com; stevenzzf0926@gmail.com;
qhc.neu@gmail.com; yuan-ye@bit.edu.cn; wanggrbit@gmail.com

***Abstract*—Mining cohesive subgraphs in attributed graphs is an essential problem in the domain of graph data analysis. The integration of fairness considerations significantly fuels interest in models and algorithms for mining fairness-aware cohesive subgraphs. Notably, the relative fair clique emerges as a robust model, ensuring not only comprehensive attribute coverage but also greater flexibility in distributing attribute vertices. Motivated by the strength of this model, we for the first time pioneer an investigation into the identification of the maximum relative fair clique in large-scale graphs. We introduce a novel concept of colorful support, which serves as the foundation for two innovative graph reduction techniques. These techniques effectively narrow the graph's size by iteratively removing edges that do not belong to relative fair cliques. Furthermore, a series of upper bounds of the maximum relative fair clique size is proposed by incorporating consideration of vertex attributes and colors. The pruning techniques derived from these upper bounds can significantly trim unnecessary search space during the branch-and-bound procedure. Adding to this, we present a heuristic algorithm with a linear time complexity, employing both a degree-based greedy strategy and a colored degree-based greedy strategy to identify a larger relative fair clique. This heuristic algorithm can serve a dual purpose by aiding in branch pruning, thereby enhancing overall search efficiency. Extensive experiments conducted on six real-life datasets demonstrate the efficiency, scalability, and effectiveness of our algorithms.**

I. Introduction

Graph, consisting of a collection of vertices and edges connecting these vertices, has gained widespread use in representing intricate real-world networks. Graph analysis stands as a crucial tool for understanding network structures and revealing underlying relationships. One fundamental task of graph analysis is cohesive subgraph computation, which aims to identify locally well-connected structures in graphs [1]. A clique, which requires that every pair of vertices within it must be connected by an edge, represents the most basic form of a cohesive subgraph. The computation of cohesive subgraph related to clique has drawn extensive attention in both academia and industry spheres, resulting in many notable research outcomes such as those highlighted in [2]–[6].

Attributed graphs are a specialized type of graph that enhance the modeling of real-world networks by incorporating additional attribute information into vertices and edges. A lot of studies focus on identifying communities within attributed graphs [7]–[14]; however, these studies often prioritize communities with the highest number of desired attributes, disregarding the fairness of communities and thus failing to meet the needs of specific real-world applications. For example, in collaborative networks where each vertex has a research direction, users may aim to find a small, closely collaborating group to achieve specific interdisciplinary project goals. Such goals require a balanced representation of individuals with diverse research interests to ensure that the team possesses sufficient expertise in every area. Similarly, in social networks where each vertex includes a gender attribute, users might seek to identify a team with a balanced representation of males and females to reduce gender bias.

To address this problem, Pan *et al.* introduced the concept of fairness from the field of artificial intelligence [15]–[22] into data mining and proposed both weak fair clique and strong fair clique models to characterize fair communities [23]. Specifically, a weak fair clique is a maximal clique ensuring that the number of vertices for each attribute is at least k. On the other hand, a strong fair clique not only requires that the number of vertices with different attributes no less than k but also must be strictly equal. Subsequently, various works on fair cliques are investigated, including the relative fair clique [24], absolute fair clique [25], fair clique for bipartite graphs [26], and fair community for heterogeneous graphs [27]. The relative fair clique, in particular, mandates that the number of vertices for each attribute is at least k, with the difference in the vertex number for different attributes not exceeding δ. Clearly, this model strikes a balance between a weak fair clique and a strong fair clique, ensuring comprehensive attribute coverage while allowing for a more flexible distribution of vertices among attributes. With this robust cohesive subgraph model, we embark on the inaugural investigation of finding the maximum relative fair clique in large-scale graphs.

Identifying the maximum relative fair clique holds significant applications across diverse domains in graph analysis. For example, in collaboration networks, finding the largest team with a small difference in the number of males and females can enhance project creativity by leveraging the distinct strengths that different genders bring to problem-solving, decision-making, and various domains. Similarly, when a project needs the convergence of distinct research domains, it is often imperative to assemble a team that encompasses all areas in a balanced manner, while also being of the maximum size. In social networks, the pursuit of larger and well-connected teams, including both local and foreign members, can significantly enhance product promotion, facilitating the attainment of global brand exposure and influence. In the domain of film, discovering and investing in a substantial team comprising both young talent and seasoned actors is likely to yield higher returns, given that such a team typically possesses a high level of experience and creativity, among other valuable attributes.

To address the problem of maximum fair clique search, an intuitive approach is to enumerate all relative fair cliques and output the one with the largest number of vertices. Nevertheless, this approach is computationally expensive, especially for large graphs. Given our goal of finding the relative fair clique with the largest size, a more efficient approach is typically developed by focusing on three crucial aspects: (i) introducing efficient graph reduction techniques to narrow the size of the graph before performing the branch-and-bound search; (ii) designing effective upper bounds on the size of relative fair clique, enabling the pruning of branches that are unlikely to contain the maximum relative fair clique; (iii) developing heuristic algorithms that quickly identify a larger relative fair clique, further aiding branch pruning. In line with these three aspects, we make the following contributions.

Novel graph reduction techniques. We introduce a novel concept called "colored support" and use it to define a specific subgraph, which is demonstrated to encompass all relative fair cliques. To compute this subgraph, the ColorfulSup algorithm is presented with a peeling strategy to iteratively remove edges that are not permissible within relative fair cliques. Additionally, the enhanced colorful support based reduction is provided to further reduce the graph size.

A series of upper bounds for branch pruning. We focus on the colors and attributes of vertices, leading to the development of intuitive upper bounds with low computational complexity, such as the attribute-color-based upper bound and enhanced-attribute-color-based upper bound. To further improve pruning effectiveness, we develop the colorful-degeneracy-based upper bound, colorful-h-index-based upper bound, and colorful-path-based upper bound. Despite the potential for slightly increased computational costs, the superior pruning performance of these advanced upper bounds ultimately contributes to the search efficiency of the maximum relative fair clique.

Efficient heuristic search algorithms. We present a heuristic algorithm that combines the degree-based and colorful-degree-based greedy strategies. This algorithm can produce a larger relative fair clique with linear time complexity, contributing to pruning the search branches.

Extensive experiments. We conduct comprehensive experimental studies to evaluate the proposed algorithms using six real-world datasets. The results demonstrate that: (i) the colorful-support-based reduction and its enhanced version significantly remove edges not contained in relative fair cliques; (ii) the proposed upper bounds markedly reduce the runtime for the maximum fair clique search; (iii) the relative fair clique size yielded by our heuristic algorithm closely align with the size of the maximum relative fair clique. In most datasets, the difference does not exceed 6. Additionally, we conduct five case studies on real-life graphs with different attributes. The results show that: (i) the fair clique model effectively maintains a balanced number of vertices with different attributes within a community; and (ii) our algorithms can identify the maximum relative fair clique, making it a versatile tool applicable in various domains including product marketing, team formation, business investment, and more.

II. PRELIMINARIES

We focus on an undirected and unweighted attributed graph $G = (V, E, A)$, where V represents the set of vertices, E stands for the set of edges, and A is the set of vertex attributes. Let $n = |V|$, $m = |E|$ be the number of vertices and edges, respectively. We specifically concentrate on the scenario of two-dimensional attributes, i.e., $A = \{a, b\}$, and the number of attributes is $A_n = 2$. Given a vertex v, its attribute is denoted as $A(v)$. The set of v's neighbors is denoted as $N_G(v)$, i.e., $N_G(v) = \{u \in V|(u, v) \in E\}$, and $deg_G(v) = |N_G(v)|$ represents the degree of v. Denote by d_{max} the maximum degree of the vertices in G. For a subset $S \subseteq V$, the subgraph of G induced by S is defined as $G_S = (V_S, E_S)$ where

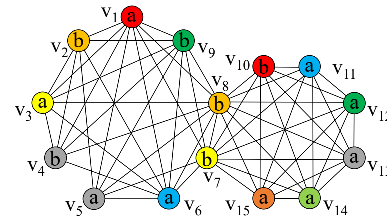


Fig. 1. The example graph G

$V_S = S$ and $E_S = \{(u, v)|u, v \in S, (u, v) \in E\}$. Given an attribute a (resp., b), we use $cnt_S(a)$ (resp., $cnt_S(b)$) to indicate the number of vertices in S whose attribute is a (resp., b), i.e., $cnt_S(a) = |\{v \in S|A(v) = a\}|$ (resp., $cnt_S(b) = |\{v \in S|A(v) = b\}|$). The subscript G, S in the notations $N_G(v), deg_G(v)$, $cnt_S(a)$ and $cnt_S(b)$ are omitted when the context is self-evident.

Definition 1: (Relative fair clique) [24] Given an attributed graph $G = (V, E, A)$ with $A = \{a, b\}$ and two integers k, δ, a clique C of G is a (k, δ)-relative fair clique satisfying:

(i) Fairness: The number of vertices associated with attribute a and attribute b is no less than k, and the difference in their vertex counts is no more than δ, i.e., $cnt_C(a) \geq k$, $cnt_C(b) \geq k$ and $|cnt_C(a) - cnt_C(b)| \leq \delta$.
(ii) Maximal: There is no clique $C' \supset C$ in G satisfying (i).

Below, we present the problem formulation of the maximum relative fair clique search, followed by an illustrative example. For brevity, we refer to a relative fair clique as a fair clique and use them interchangeably throughout the rest of the paper.

Problem formulation. Given an attributed graph $G = (V, E, A)$ with $A = \{a, b\}$, and two integers k, δ, our goal is to identify a relative fair clique in G with the maximum number of vertices.

Example 1: Consider a graph G shown in Fig. 1, and suppose the parameters $k = 3$ and $\delta = 1$. Given a vertex set $S = \{v_7, v_8, v_{10}, v_{11}, v_{12}, v_{13}, v_{14}, v_{15}\}$, then the answer to the maximum relative fair clique search problem is $S - v_{11}$ (or $S - v_{12}, S - v_{13}, S - v_{14}, S - v_{15}$).

Challenges. We begin by discussing the complexity of the maximum fair clique search problem. In a special case where $k = 0$ and $\delta = +\infty$, the maximum fair clique search problem becomes to the maximum clique search problem, which is known to be NP-hard [3]. Consequently, the maximum fair clique search problem is also NP-hard. To address the maximum fair clique search problem, a straightforward approach is to identify all fair cliques and then output the one with the largest number of vertices. However, this approach is fraught with inefficiency, particularly when dealing with large-scale graphs. The problem presents several challenges: (i) How to devise effective graph reduction techniques to shrink the size of graphs before initiating the branch-and-bound search; (ii) How to design upper bounding techniques that minimize the exploration of undesirable branches during the branch-and-bound search procedure; (iii) How to develop efficient heuristic algorithms that can rapidly identify a larger fair clique, enabling the efficient pruning of search branches. To tackle these challenges, we introduce novel colorful-support-based reduction techniques, leveraging insights from truss decomposition. These techniques are capable of significantly reducing the size of the graph by excluding vertices and edges that cannot form a fair clique. Additionally, a series of powerful upper bound based pruning techniques are developed to steer clear of needless branch exploration in the branch-and-bound search process. To further improve efficiency, a heuristic algorithm with linear time complexity is presented, efficiently computing a larger fair clique to facilitate more vigorous branch pruning.

Discussion. This paper builds on the fair clique model, which is also applicable to community search. Specifically, given a set of vertices, the goal is to enumerate all fair clique containing these vertices or to find the maximum fair clique that includes them. Both tasks can be accomplished by using the subgraph induced by the query vertices and their common neighbors as the input graph, then applying the fair clique enumeration algorithms [23], [24] and our maximum fair clique search algorithms.

III. The Graph Reduction Techniques

This section emphasizes graph reduction techniques as a preliminary step to performing the branch-and-bound search for the maximum fair clique. We initially introduce existing graph reduction methods, and subsequently, explore novel techniques based on the concept of "colorful support" to effectively reduce the graph's size.

A. Existing techniques

Existing graph reduction techniques stem from graph coloring, which aims to assign colors to vertices to ensure that connected vertices have distinct colors [28], [29]. Given a graph $G = (V, E)$, we denote the color of a vertex $u \in V$ by $color(u)$. With graph coloring, Pan *et al.* introduced two essential concepts: the *colorful degree* and *colorful k-core*, forming the basis of their graph reduction techniques.

Definition 2: (Colorful degree) [23], [24] Given an attributed graph $G = (V, E, A)$ with $A = \{a, b\}$. For an attribute a (or b), the colorful degree of vertex u based on a (or b), denoted by $D_a(u, G)$ (or $D_b(u, G)$), refers to the count of distinct colors among u's neighbors associated with attribute a (or b), i.e., $D_a(u, G) = |\{color(v)|v \in N(u), A(v) = a\}|$ (or $D_b(u, G) = |\{color(v)|v \in N(u), A(v) = b\}|$).

Definition 3: (Colorful k-core) [23], [24] Given an attributed graph $G = (V, E, A)$ with $A = \{a, b\}$ and an integer k, a subgraph $H = (V_H, E_H, A)$ of G is a colorful k-core if: (i) for each $u \in V_H$, $D_{min}(u, H) = \min\{D_a(u, H), D_b(u, H)\} \geq k$; (ii) there is no subgraph $H' \subseteq G$ satisfying (i) and $H \subset H'$.

With these concepts, the colorful k-core based graph reduction, namely, ColorfulCore, is shown in Lemma 1 [23], [24].

Lemma 1: Given an attributed graph $G = (V, E, A)$ and an integer k, any relative fair clique must be contained in the colorful $(k-1)$-core of G [23], [24].

The ColorfulCore reduction considers the attributes of u's neighbors individually, potentially assigning the same color to vertices with attributes a and b. However, this scenario is improbable in a fair clique. Addressing this, Zhang *et al.* [24] proposed the enhanced colorful k-core based reduction, known as EnColorfulCore, by allocating each color to a specific attribute. Before introducing EnColorfulCore, we give the following important concepts.

Definition 4: (Enhanced colorful degree) Given a colored attributed graph $G = (V, E, A)$ with $A = \{a, b\}$, the enhanced colorful degree of u, denoted as $ED(u)$, is defined as the minimum number of colors assigned exclusively to either attribute a or attribute b.

Definition 5: (Enhanced colorful k-core) Given an attributed graph $G = (V, E, A)$ with $A = \{a, b\}$ and an integer k, a subgraph $H = (V_H, E_H, A)$ of G is an enhanced colorful k-core if: (i) for each $u \in V_H$, $ED(u) \geq k$; (ii) there is no subgraph $H' \subseteq G$ that satisfies (i) and $H \subset H'$.

Lemma 2 details the reduction technique based on the enhanced colorful k-core, i.e., EnColorfulCore [24].

Lemma 2: Given an attributed graph $G = (V, E, A)$ with $A = \{a, b\}$ and an integer k, any relative fair clique must be contained in the enhanced colorful $(k-1)$-core of G.

B. The colorful support based reduction

Existing graph reduction techniques focus on eliminating unpromising vertices, offering limited capability for reducing graph size. Here we introduce the novel concept of "colorful support" and develop a reduction technique that iteratively deletes edges unlikely to form fair cliques. The *colorful support* of an edge (u, v) is defined as follows.

Definition 6: (Colorful support) Given an attributed graph $G = (V, E, A)$, an edge (u, v), and an attribute $a_i \in A$. The colorful support of (u, v) based on a_i, denoted by $\overline{sup}_{a_i}(u, v)$, is the number of distinct colors within the common neighbors of u and v having attribute a_i, i.e., $\overline{sup}_{a_i}(u, v) = |\{color(w)|w \in N(u) \cap N(v), A(w) = a_i\}|$.

Below, we introduce the colorful support based reduction technique, namely, ColorfulSup, elaborated in Lemma 3.

Lemma 3: Given an attributed graph $G = (V, E, A)$ with $|A| = A_n$ and an integer k, let G' be the maximal subgraph of G satisfying:

(i) $\forall (u, v) \in E_{G'}$ with $A(u) = A(v)$, $\overline{sup}_{A(u)}(u, v) \geq k-2$ and $\overline{sup}_{a_i \in (A - \{A(u)\})}(u, v) \geq k$;

(ii) $\forall (u, v) \in E_{G'}$ with $A(u) \neq A(v)$, $\overline{sup}_{A(u)}(u, v) \geq k-1$, $\overline{sup}_{A(v)}(u, v) \geq k-1$, and $\overline{sup}_{a_i \in (A - \{A(u) \cup A(v)\})}(u, v) \geq k$;

then, any fair clique C in G that adheres to the size constraint of k is encompassed within G'.

Proof: Let's consider an attributed graph $G = (V, E, A)$ with $A = \{a, b\}$ and an edge (u, v) in the fair clique C with $A(u) = A(v) = a$. According to Definition 1, u and v must have at least $k-2$ common neighbors with attribute a and at least k common neighbors with attribute b in C. Since vertices with the same color cannot be adjacent, it follows that $\overline{sup}_a(u, v) \geq k-2$ and $\overline{sup}_b(u, v) \geq k$. Similar arguments apply to (u, v) in C with $A(u) = A(v) = b$, or $A(u) = a, A(v) = b$, or $A(u) = b, A(v) = a$. Due to space limitations, we omit the proofs for these cases. Hence, it can be concluded that C must be included in the maximal subgraph G'. □

Algorithm 1 depicts the pseudo-code of the colorful support reduction technique ColorfulSup, a variant of the truss decomposition. The main idea is to iteratively delete edges failing to satisfy any of the two conditions in Lemma 3 to reduce the graph size. Specifically, it first performs graph coloring by degree-based greedy method, thereby calculating the colorful support for each edge (lines 1-5). A priority queue $\mathcal{Q}$ maintains edges that violate one of the two conditions in Lemma 3, which will be removed during the peeling procedure (line 6). The data structure $M_{(u,v)}$ keeps track of the count of common neighbors of u and v with identical attributes and colors (lines 7-16). Subsequently, ColorfulSup iteratively peels edges from the remaining graph according to Lemma 3 (lines 17-25). Finally, the algorithm outputs the remaining graph G' as the maximal subgraph defined in Lemma 3 (lines 26-27).

Algorithm 1: ColorfulSup(G, k)

Input: $G = (V, E, A)$, an integer k
Output: The maximal subgraph G'

1 Color all vertices with a degree-based greedy coloring algorithm;
2 **for** $(u, v) \in E$ **do**
3   **for** $w \in N(u) \cap N(v)$ **do**
4   **if** $M_{(u,v)}(A(w), color(w)) = 0$ **then** $\overline{sup}_{A(w)}(u, v)$++;
5     $M_{(u,v)}(A(w), color(w))$++;
6 Let $\mathcal{Q}$ be a priority queue; $\mathcal{Q} \leftarrow \emptyset$;
7 **for** $(u, v) \in E$ **do**
8   **if** $A(u) = a$ *and* $A(v) = a$ **then**
9   **if** $\overline{sup}_a(u, v) < k - 2$ *or* $\overline{sup}_b(u, v) < k$ **then**
10       $\mathcal{Q}.push(u, v)$; Remove (u, v) from G;
11   **else if** $A(u) = b$ *and* $A(v) = b$ **then**
12   **if** $\overline{sup}_a(u, v) < k$ *or* $\overline{sup}_b(u, v) < k - 2$ **then**
13       $\mathcal{Q}.push(u, v)$; Remove (u, v) from G;
14   **else**
15   **if** $\overline{sup}_a(u, v) < k - 1$ *or* $\overline{sup}_b(u, v) < k - 1$ **then**
16       $\mathcal{Q}.push(u, v)$; Remove (u, v) from G;
17 **while** $\mathcal{Q} \neq \emptyset$ **do**
18   $(u, v) \leftarrow \mathcal{Q}.pop()$;
19   **for** $w \in N(u) \cap N(v)$ **do**
20   **if** (u, w) *is not removed* **then**
21       $M_{(u,w)}(A(v), color(v))$--;
22   **if** $M_{(u,w)}(A(v), color(v)) \leq 0$ **then**
23         $\overline{sup}_{A(v)}(u, w) \leftarrow \overline{sup}_{A(v)}(u, w) - 1$;
24   Perform the operations as lines 8-16 for edge (u, w);
25   Perform the operations as lines 20-24 for edge (v, w);
26 $G' \leftarrow$ the remaining graph of G;
27 **return** G';

Example 2: Consider a graph G in Fig. 1, and suppose that $k = 3$ and $\delta = 1$. It is evident that G qualifies as a colorful 2-core as $D_{min}(u, G) \geq 2$ for every vertex u in G. Meanwhile, G is also an enhanced colorful 2-core. For edge (v_2, v_5), the common neighbors with attribute a are v_1 and v_6, while the remaining v_9 is associated with attribute b. Therefore, we have $\overline{sup}_a(v_2, v_5) = 2$ and $\overline{sup}_b(v_2, v_5) = 1$. Clearly, (v_2, v_5) violates condition (ii) in Lemma 3 because of $A(v_2) = b$, $A(v_5) = a$ and $\overline{sup}_b(v_2, v_5) < 3 - 1 = 2$, thus it cannot form a fair clique and can be safely removed from G. Following this deletion, the remaining graph satisfies Lemma 3, containing all fair cliques in G with the size constraint k.

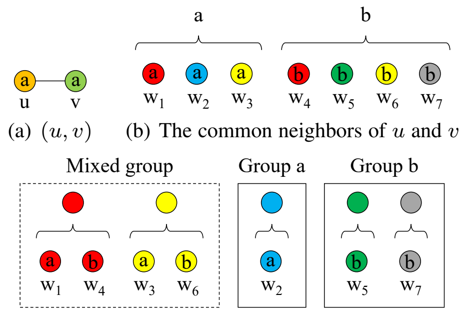
Below, we analyze the complexity of Algorithm 1.

Theorem 1: Algorithm 1 consumes $O(\alpha \times |E| + |V|)$ time using $O(|E| \times |A| \times color(G))$ space, where α is the arboricity of graph G, and $color(G)$ denotes the number of colors in G.

Proof: In line 1, the greedy coloring procedure takes $O(|E| + |V|)$ time [30]. In lines 2-5, it is clear that the algorithm takes $O(\sum_{(u,v)\in E} \min\{deg(u), deg(v)\}) = O(\alpha \times |E|)$ time. Regarding lines 17-25, the algorithm can update $M_{(u,w)}$ and $M_{(v,w)}$ for each $w \in N(u) \cap N(v)$ in $O(1)$ time. For each triangle (u, v, w), the update operator only performs once, thus the total time complexity of Algorithm 1 is bounded by $O(\alpha \times |E| + |V|)$. In terms of space complexity, the algorithm maintains $M_{(u,v)}$ for each edge, resulting in a total space requirement bounded by $O(|E| \times |A| \times color(G))$. $\square$

C. The enhanced colorful support based reduction

However, the ColorfulSup technique still exhibits limitations in graph reduction. Consider, for instance, an edge (u, v) in Fig. 2(a), where $k = 4$. The common neighbors of u and v are depicted in Fig. 2(b). According to Definition 3, we determine $\overline{sup}_a(u, v) = 3$ and $\overline{sup}_b(u, v) = 4$, implying that (u, v) is



(a) (u, v) (b) The common neighbors of u and v

(c) The groups of common neighbors

Fig. 2. The shortcoming of ColorfulSup

preserved after executing ColorfulSup. Nevertheless, it is worth noting that neighbors with attribute a share colors with those bearing attribute b. Thus, these seven neighbors are unlikely to coexist within a fair clique. Given these limitations, we draw inspiration from the enhanced colorful degree and propose an alternative: the enhanced colorful support as presented below.

Definition 7: (Enhanced colorful support) Given an attributed graph $G = (V, E, A)$ with $A = \{a, b\}$, an edge (u, v), and an attribute value $a_i \in A$. The enhanced colorful support of (u, v) based on a_i, denoted as $\widetilde{sup}_{a_i}(u, v)$, is the count of colors designated with attribute a_i.

The enhanced colorful support is determined by associating each color with a specific attribute. For instance, when considering an edge (u, v) with $A(u) = A(v) = a$, the process unfolds as follows. The common neighbors of u and v are partitioned into three groups based on their colors: Group a, Group b and Mixed group. Let c_a, c_b and c_m be the number of colors within these three respective groups. In case $c_m = 0$, we set $\widetilde{sup}_a(u, v) = c_a$ and $\widetilde{sup}_b(u, v) = c_b$. On the other hand, when $c_a < k-2$, we select $\gamma = \min\{(k-2-c_a), c_m\}$ colors from the Mixed group and assign them to attribute a, resulting in $\widetilde{sup}_a(u, v) = c_a + \gamma$; otherwise, we set $\widetilde{sup}_a(u, v) = c_a$. Next, we update the remaining $\hat{c}_m = c_m - \gamma$ and repeat the color assignment process for attribute b. Thus, $\widetilde{sup}_b(u, v) = c_b + \min\{(k - c_b), \hat{c}_m\}$ holds when $c_b < k$, while it remains at c_b otherwise. The calculation of $\widetilde{sup}_a(u, v)$ and $\widetilde{sup}_b(u, v)$ in the scenario where the edge's endpoints possess other attributes can be inferred similarly, although not elaborated due to space constraints. With the definition and calculation method of enhanced colorful support established, we proceed to the subsequent lemma, which contributes to further reducing the graph size.

Lemma 4: Given an attributed graph $G = (V, E, A)$ with $A = \{a, b\}$ and an integer k, let G' be the maximal subgraph of G satisfying:

(i) $\forall (u, v) \in E_{G'}$ with $A(u) = A(v) = a$, $\widetilde{sup}_a(u, v) \geq k - 2$ and $\widetilde{sup}_b(u, v) \geq k$;
(ii) $\forall (u, v) \in E_{G'}$ with $A(u) = A(v) = b$, $\widetilde{sup}_a(u, v) \geq k$ and $\widetilde{sup}_b(u, v) \geq k - 2$;
(iii) $\forall (u, v) \in E_{G'}$ with $A(u) = a, A(v) = b$ or $A(u) = b, A(v) = a$, $\widetilde{sup}_a(u, v) \geq k-1$ and $\widetilde{sup}_b(u, v) \geq k-1$;

then, every fair clique C in G that satisfies the size constraint with k is contained in G'.

Example 3: Consider the edge (u, v) with $A(u) = A(v) = a$ in Fig. 2(a) as an illustration. The common neighbors of u and v can be divided into three groups as shown in Fig. 2(c). Here, attribute a is uniquely associated with blue, and attribute b is exclusive to dark green and grey. The colors red and yellow, on the other hand, are common to both attributes a and b.

Thus, we have $c_a = 1$, $c_b = 2$ and $c_m = 2$. For a fair clique with a size constraint of $k = 4$ that includes (u, v), it needs to be supplemented with at least 2 vertices with a and 4 vertices with b. Consider the first attribute a. As a is exclusively blue, we must choose $\gamma = \min\{(4-2-1), 2\} = 1$ color from the Mixed group to assign to attribute a, which is assumed to be red. For attribute b, only yellow remains in the Mixed group at this point, so we assign it to b. Thus, we have $\widetilde{sup}_a(u, v) = 2$ and $\widetilde{sup}_b(u, v) = 3$. Evidently, (u, v) obey condition (i) in Lemma 4, indicating it must not form a fair clique and can therefore be safely removed.

To compute the maximal subgraph G' in Lemma 4, we employ the peeling strategy and make the following simple adaptation of Algorithm 1. Specifically, in lines 2-5, instead of calculating the colorful support for each edge, we compute the enhanced colorful support. Then, we initialize the priority queue $\mathcal{Q}$ and eliminate unpromising edges based on Lemma 4 in lines 7-25. This adapted version, utilizing enhanced colorful support, is named EnColorfulSup and its pseudo-code is omitted due to space limit. Theorem 2 shows the complexity of EnColorfulSup.

Theorem 2: The EnColorfulSup algorithm's time complexity is $O(\alpha \times |E| \times color(G))$, utilizing $O(|E| \times color(G))$ space.

Proof: As mentioned, the greedy coloring procedure takes $O(|E| + |V|)$ time [30]. The algorithm takes $O(\sum_{(u,v)\in E} \min\{deg(u), deg(v)\} + |E| \times color(G)) = O((\alpha + color(G)) \times |E|)$ time to initialize $Group_{(u,v)}$ and calculate $\widetilde{sup}_a(u, v)$ and $\widetilde{sup}_b(u, v)$ for each edge. For each triangle (u, v, w), the update cost is bounded by $O(color)$. Thus the total time complexity amounts to $O(\alpha \times |E| \times color(G))$. Regarding space complexity, the algorithm maintains the structure $Group_{(u,v)}$ for each color, resulting in a total space requirement $O(|E| \times color(G))$. □

Remark. The colorful support based reduction technique is independent of attribute dimensionality, making it applicable for solving the maximum relative fair clique search problem in graphs with arbitrary attribute dimensions. Furthermore, this technique is adaptable to various fair clique enumeration problems, including weak fair clique enumeration, strong fair clique enumeration, and relative fair clique enumeration [23], [24]. By contrast, our enhanced colorful support based reduction technique is tailored for 2-dimensional attribute graphs and remains applicable to these fair clique enumeration problems within the 2-dimensional attribute context.

IV. A BRANCH-AND-BOUND FRAMEWORK

This section introduces the basic framework for identifying the maximum fair clique, i.e., MaxRFC. Following this, we introduce a series of simple yet effective upper-bound techniques designed to curtail the search space. Additionally, we propose more stringent upper bounds aimed at further enhancing the efficiency of the maximum fair clique search algorithm.

A. The basic framework

Here, we present a basic framework, namely, MaxRFC, for the maximum fair clique search problem. The main idea of MaxRFC involves employing a branch-and-bound framework along with a simple upper bound derived from set size to prune unpromising branches.

Algorithm 2: MaxRFC(G, k, δ)

Input: $G = (V, E, A)$ with $A = \{a, b\}$, two integers k, δ
Output: The fair clique with the largest size R^*

1 $\ddot{G} = (\ddot{V}, \ddot{E}) \leftarrow$ EnColorfulCore(G, k);
2 $\hat{G} = (\hat{V}, \hat{E}) \leftarrow$ ColorfulSup$(\ddot{G}, k)$;
3 $\bar{G} = (\bar{V}, \bar{E}) \leftarrow$ EnColorfulSup$(\hat{G}, k)$;
4 Initialize an array B with $B(i) = false, 1 \leq i \leq \bar{V}$;
5 $R^* \leftarrow \emptyset$;
6 **for** $u \in \bar{V}$ **do**
7   **if** $B(u) = false$ **then**
8         $C \leftarrow$ ConnectedGraph(u, B);
9         $\mathcal{O} \leftarrow$ CalColorOD(C);
10         $R \leftarrow \emptyset$;
11   Branch$(R, C, \mathcal{O}, a, -1)$;
12 **return** R^*;

The workflow of MaxRFC is detailed in Algorithm 2. R represents an identified clique with the potential for expansion into a fair clique. C denotes a candidate set with $C \cap R = \emptyset$, containing vertices used to extend set R. R^* signifies the maximum fair clique discovered thus far. Algorithm 2 initially performs EnColorfulCore, ColorfulSup, and EnColorfulSup sequentially to exclude vertices and edges that are unlikely to be included in fair cliques, thus reducing the graph size (lines 1-3). Then, the algorithm invokes the Branch procedure to find the maximum fair clique in the reduced graph $\bar{G}$ (lines 6-11). Since $\bar{G}$ may be disconnected, we perform Branch on each connected component. For vertex selection order, in line with the method outlined in [23], [24], the algorithm utilizes the colorful core based ordering, i.e., CalColorOD (line 9). Finally, MaxRFC outputs R^* as a result (line 12). The Branch procedure, described in Algorithm 3, alternatively picks a vertex of a particular attribute during the backtracking process to find a fair clique. When the candidate set C becomes empty, it signifies the discovery of a fair clique. At this point, Branch compares the current clique R with the existing optimal solution R^*, determining whether an update to R^* is warranted (line 11). Additionally, a basic upper bounding pruning technique, expressed as $|\hat{C}| + |\hat{R}|$, is integrated into Branch to reduce the number of branches (line 19).

It is noteworthy that in Algorithm 2 and Algorithm 3, we abstain from using a set, often denoted as X, to keep track of vertices that could be added to R and have been traversed in earlier search paths. This choice is made due to the fact that X is utilized to prevent redundant enumerations of fair cliques. Its absence does not impact the determination of the maximum fair clique, and the operations on X even introduce an additional time cost. Additionally, the MaxRFC algorithm employs an alternating attribute selection strategy, selecting vertices with different attributes in each round to form partial solutions, thereby maintaining a small attribute gap and accelerating solution discovery.

B. The intuitive and effective upper bounds

In this subsection, our goal is to establish upper bounds for the size of fair cliques within the search instance (R, C). Let $MRFC(R, C)$ denote the size of the maximum fair clique in the instance (R, C), and (R, C) can be entirely pruned if the upper bounds are no larger than $2 \times k + \delta$ or $|R^*|$. An intuitive upper bound of $MRFC(R, C)$ asserts that a fair clique contains all the vertices in the instance (R, C), i.e., Lemma 5, which is applied in the basic framework MaxRFC (line 19 in Algorithm 2).

Lemma 5: (Size-based upper bound) Given an instance (R, C), $ub_s = |R| + |C|$ is an upper bound of $MRFC(R, C)$.

The size-based upper bound is straightforward. By factoring in the constraint regarding the number of attributes

Algorithm 3: Branch($R, C, \mathcal{O}, attr\_choose, attr\_max$)

```
Procedure Branch(R, C, O, attr_choose, attr_max)
for u ∈ C do C_{A(u)} ← C_{A(u)} ∪ u;
for u ∈ R do R_{A(u)} ← R_{A(u)} ∪ u;
if C_{attr_choose} = ∅ and a_max = −1 then
    a_min ← |R_{attr_choose}|;
    a_max ← a_min + δ;
if |R_a| = a_max then C ← C − C_a; C_a ← ∅;
if |R_b| = a_max then C ← C − C_b; C_b ← ∅;
if C = ∅ then
    if |R*| < |R| then
        R* ← R; return;
if C_{attr_choose} = ∅ then
    Branch(R, C, O, A − attr_choose, a_max); return;
for u ∈ C_{attr_choose} do
    R̂ ← R ∪ u; Ĉ ← ∅; flag ← false;
    for v ∈ C do
        if v ∈ N(u) and O(v) > O(u) then
            Ĉ ← Ĉ ∪ v; cnt_Ĉ(A(v))++;
    if |Ĉ| + |R̂| < |R*| then continue;
    if |Ĉ| + |R̂| < 2k then continue;
    for v ∈ R̂ do cnt_R̂(A(v))++;
    if cnt_R̂(a) + cnt_Ĉ(a) < k or cnt_R̂(b) + cnt_Ĉ(b) < k then
        continue;
    Branch(R̂, Ĉ, O, A − attr_choose, a_max);
```

TABLE I
NOTATION FOR DIFFERENT UPPER BOUNDS (UB: UPPER BOUND)

Notation	Definition	Formulation
ub_s	size-based UB	$ub_s = \|R\| + \|C\|$
ub_a	attribute-based UB	$ub_a = \sum_{a_i \in A} cnt_{R\cup C}(a_i)$; or $ub_a = A_n(\min_{a_i \in A} cnt_{R\cup C}(a_i)) + (A_n - 1)\delta$
ub_c	color-based UB	$ub_c = color(R \cup C)$
ub_{ac}	attribute-color-based UB	$ub_{ac} = \sum_{a_i \in A} color_{R\cup C}(a_i)$; or $ub_{ac} = A_n(\min_{a_i \in A} color_{R\cup C}(a_i)) + (A_n - 1)\delta$
ub_{eac}	enhanced-attribute-color-based UB	$ub_{eac} = 2 \times \min\{c_a, c_b\} + c_m + \delta$ (2-dimension)
$ub_{\triangle}$	degeneracy-based UB	$ub_{\triangle} = \triangle(G')$
ub_h	h-index-based UB	$ub_h = h(G')$
ub_{cd}	colorful-degeneracy-based UB	$ub_{cd} = \sum_{a_i \in A} D_{a_i}(u, G')$; or $ub_{cd} = A_n(\min_{a_i \in A} D_{a_i}(u, G')) + (A_n - 1)\delta$
ub_{ch}	colorful-h-index-based UB	$ub_{ch} = \sum_{a_i \in A} D_{a_i}(u, G')$; or $ub_{ch} = A_n(D_{a_i}(u, G')) + (A_n - 1)\delta$
ub_{cp}	colorful-path-based UB	$ub_{cp} = \|CP(G')\|$

within a fair clique, we can derive a tighter upper bound of $MRFC(R, C)$, as demonstrated in Lemma 6.

Lemma 6: (Attribute-based upper bound) Given an instance (R, C), if $|cnt_{R\cup C}(a) - cnt_{R\cup C}(b)| < \delta$ holds, then $ub_a = cnt_{R\cup C}(a) + cnt_{R\cup C}(b)$ is an upper bound of $MRFC(R, C)$; otherwise, $ub_a = 2 \times \min\{cnt_{R\cup C}(a), cnt_{R\cup C}(b)\} + \delta$ is an upper bound of $MRFC(R, C)$.

On the other hand, we employ the graph coloring technique to deduce upper bounds for $MRFC(R, C)$. Let G' represent the subgraph induced by the vertices in $R \cup C$. We apply a degree-based greedy coloring approach to assign colors to the vertices of G' and denote the number of colors in G' as $color(R \cup C)$. By leveraging the vertex coloring, the ensuing upper bounds can be established.

Lemma 7: (Color-based upper bound) Given an instance (R, C), $ub_c = color(R \cup C)$ serves as an upper bound of $MRFC(R, C)$.

Lemma 6 and Lemma 7 individually focus on either the vertices' attributes or their colors. To achieve a more comprehensive approach, we integrate both attributes and colors to derive a tighter upper bound for $MRFC(R, C)$. Denote $color_{R\cup C}(a)$ (resp., $color_{R\cup C}(b)$) as the count of colors assigned to vertices with attribute a (resp., b) within G'. The attribute-color-based upper bound is outlined as follows.

Lemma 8: (Attribute-color-based upper bound) Given an instance (R, C), if $|color_{R\cup C}(a) - color_{R\cup C}(b)| < \delta$, then $ub_{ac} = color_{R\cup C}(a) + color_{R\cup C}(b)$ stands as an upper bound for $MRFC(R, C)$; otherwise, $ub_{ac} = 2 \times \min\{color_{R\cup C}(a), color_{R\cup C}(b)\} + \delta$ serves as an upper bound of $MRFC(R, C)$.

In Lemma 8, it is possible for color intersections between vertices with attribute a and those with attribute b. Drawing inspiration from the concept of enhanced colorful support, we introduce a tighter upper bound ub_{eac} for $MRFC(R, C)$ by categorizing vertices based on their colors.

Lemma 9: (Enhanced-attribute-color-based upper bound) Given an instance (R, C), if $\min\{c_a, c_b\} + c_m < \max\{c_a, c_b\} - \delta$, then $ub_{eac} = 2 \times \min\{c_a, c_b\} + c_m + \delta$ serves as an upper bound of $MRFC(R, C)$. Here c_a, c_b and c_m are the number of colors in the Group a, Group b and Mixed group, respectively.

Theorem 3: Computing ub_s has a time complexity of $O(1)$, and computing ub_a/ub_c/ub_{ac}/ub_{eac} carries a time complexity of $O(|V(G')|)$.

Beyond the mentioned upper bounds, those bounds for the maximum clique size can also serve as constraints for the maximum fair clique size because a fair clique represents a specific instance of a clique. The upper bounds of the maximum clique size typically encompass the degeneracy of a graph [31], [32], and the h-index of a graph [33], as illustrated in Lemma 10 and Lemma 11.

Lemma 10: (Degeneracy-based upper bound [34]) Given an instance (R, C), $ub_{\triangle} = \triangle(G')$ is an upper bound of $MRFC(R, C)$ where $\triangle(G')$ denotes the degeneracy of G' (i.e., the maximum core number of G').

Lemma 11: (h-index-based upper bound [34]) Given an instance (R, C), $ub_h = h(G')$ is an upper bound of $MRFC(R, C)$ where $h(G')$ is the maximum value of h such that there exist h vertices with degree no less than h in G'.

It is proved that $MRFC(R, C) \leq ub_{\triangle} \leq ub_h$, with the computation of degeneracy having a higher time complexity compared to that of h-index of a graph (i.e., Theorem 4).

Theorem 4: The time complexity of computing $ub_{\triangle}$ and ub_h are $O(|E(G')|)$ and $O(|V(G')|)$, respectively [34].

C. The non-trivial upper bounds

In this subsection, we present three novel concepts: "colorful degeneracy", "colorful h-index", and "colorful path". These concepts provide corresponding upper bounds to bound the size of the maximum fair clique within the search branch (R, C). We introduce each of these three non-trivial upper bounds in turn below.

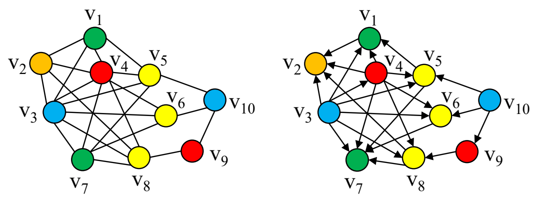
Colorful-degeneracy-based upper bound. Building upon the colorful k-core concept, the colorful core number and colorful degeneracy are defined as follows.

Definition 8: (Colorful core number) Given a colored graph G, the colorful core number of a vertex v in G, denoted as $ccore(v)$, is the largest k such that the colorful k-core of G contains v.

Definition 9: (Colorful degeneracy) The color degeneracy of G is the maximum value among colorful core numbers of vertices in G, i.e., $\overline{\triangle}(G) = \max_{v \in G} ccore(v)$.

With Definition 9, the upper bound of the maximum fair clique derived by colorful degeneracy is given in Lemma 12.

Lemma 12: (Colorful-degeneracy-based upper bound) Given an instance (R, C), let u be the vertex with the largest



(a) The colored graph G' (b) The DAG graph $\vec{G}'$

Fig. 3. Running example of the colorful-path-based upper bound

colorful core number, i.e., $u = \arg\max_{v \in G'} ccore(v)$. If $|D_a(u, G') - D_b(u, G')| < \delta$, then $ub_{cd} = D_a(u, G') + D_b(u, G')$ is an upper bound of $MRFC(R, C)$; otherwise, $ub_{cd} = 2 \times \min\{D_a(u, G'), D_b(u, G')\} + \delta$ stands as an upper bound of $MRFC(R, C)$.

Theorem 5 outlines the time complexity for computing the colorful-degeneracy-based upper bound, which aligns its proof with the complexity of computing the colorful k-core, as detailed in [23], [24].

Theorem 5: The time complexity of computing ub_{cd} is $O(|E(G')| + |V(G')|)$.

Colorful-h-index-based upper bound. Here, we introduce the definition of the colorful h-index, followed by the derived upper bound governing the size of a maximum fair clique within (R, C).

Definition 10: (Colorful h-index) Given a colored graph G, for a vertex v in G, let $D_{min}(v, G) = \min\{D_a(v, G), D_b(v, G)\}$. We construct a sequence $L = D_{min}(v_1, G), D_{min}(v_2, G), ..., D_{min}(v_t, G)$. The colorful h-index of G, denoted as $\overline{h}(G)$, is the maximum integer h such that there exist at least h vertices with $D_{min}(v, G) \geq h$.

Utilizing Definition 10, we establish an upper bound for $MRFC(R, C)$ through the colorful h-index, as shown in Lemma 13.

Lemma 13: (Colorful-h-index-based upper bound) Given an instance (R, C), coloring the subgraph G' induced by the vertices in $R \cup C$. Let u represent the vertex with $\overline{h}(G') = D_{min}(u, G')$. If $|D_a(u, G') - D_b(u, G')| < \delta$, then $ub_{ch} = D_a(u, G') + D_b(u, G')$ is an upper bound of $MRFC(R, C)$; else, $ub_{ch} = 2 \times \min\{D_a(u, G'), D_b(u, G')\} + \delta$ is an upper bound of $MRFC(R, C)$.

Theorem 6: The time complexity of computing ub_{ch} is $O(|E(G')| + |V(G')|)$.

Proof: The calculation of $D_{min}(u, G)$ for each vertex u in G requires $O(\sum_{u \in V(G')} deg(u) + |V(G')|) = O(|E(G')| + |V(G')|)$ time. Following this, the computation of the h-index consumes $O(|V(G')|)$ time. Thus, the time complexity for computing ub_{ch} amounts to $O(|E(G')| + |V(G')|)$. □

As mentioned earlier, the degeneracy-based upper bound is not greater than the h-index-based upper bound. Similarly, by Definition 9 and Definition 10, we have that the colorful-degeneracy-based upper bound is not greater than the colorful-h-index-based upper bound, i.e., $ub_{cd} \leq ub_{ch}$.

Colorful path based upper bound. Given an instance (R, C) and the colored subgraph G'. Let $CL(G') = \{c_1, c_2, ..., c_p\}$ denote the color set of G', and V_{c_i} represent the vertices with color c_i, i.e., $V_{c_i} = \{v \in R \cup C | color(v) = c_i\}$. By utilizing the color ID and vertex ID, a total ordering $\prec$ on $R \cup C$ can be defined with the following rule. For any two vertices u and v in $R \cup C$, $u \prec v$ if and only if: (i) $color(u) < color(v)$; or (ii) $color(u) = color(v)$ and $u_{ID} < v_{ID}$ [35]. Based on this total ordering, each edge (u, v) can be oriented from the low-ranked vertex to the high-ranked vertex, resulting in a Directed Acyclic Graph (DAG) $\vec{G}'$. Below, we provide the definition of a colorful path.

Algorithm 4: ColorfulPathDP(G, R, C)

```
Input: The graph G = (V, E), an instance (R, C)
Output: The largest length of colorful paths in (R, C)
1  Color all vertices in R ∪ C by a degree-based greedy coloring algorithm;
2  Construct the DAG G⃗' = (V' = (R ∪ C), E⃗') of G';
3  Let f(i) be the number of vertices in a colorful path ending in i and with the
   maximum size;
4  Let B be an array of size |V'| constructed according to the total ordering ≺;
5  for u ∈ V' do
6      f(u) ← 1;
7  for i from 0 to |V'| − 1 do
8      u ← B(i);
9      for v ∈ N⁻_G⃗'(u) do
10         f(u) ← max{f(u), f(v) + 1};
11     maxlen ← max{maxlen, f(u)};
12 return maxlen;
```

Definition 11: (Colorful path) Given a colored graph $G = (V, E)$, a colorful path $P = \{v_1, v_2, ..., v_p\}$ is a path where each vertex possesses a unique color, i.e., $\forall v_i \in P, \nexists v_j \in P - \{v_i\}, color(v_i) = color(v_j)$.

Within the (fair) clique, every pair of vertices is connected by edges. Due to the principle of graph coloring, the vertices in the (fair) clique hold different colors, thereby forming a colorful path. It is evident that the largest colorful path can be used to establish an upper bound for the size of the maximum (fair) clique, as detailed in Lemma 14.

Lemma 14: (Colorful-path-based upper bound) Given an instance (R, C), coloring the subgraph G' induced by the vertices in $R \cup C$. We construct its DAG $\vec{G}'$ using the total ordering and let $CP(G')$ be the largest colorful path in G'. Then, $ub_{cp} = |CP(G')|$ is an upper bound of $MRFC(R, C)$.

Example 4: Consider a colored graph G' shown in Fig. 3(a). We can easily check that $CL = \{c_1 = blue, c_2 = red, c_3 = yellow, c_4 = green, c_5 = orange\}$. Assuming $k = 3$ and $\delta = 1$, let's consider the edge (v_3, v_5). Since $color(v_3) = c_1 < color(v_5) = c_3$, we conclude that $v_3 \prec v_5$ based on the total ordering, resulting in the directed edge $\langle v_3, v_5 \rangle$ in $\vec{G}'$. The DAG $\vec{G}'$ of G' is depicted in Fig. 3(b). Within $\vec{G}'$, there exists a 5-colorful path $P = \{v_3, v_4, v_5, v_1, v_2\}$ and nine 4-colorful paths. It is evident that $CP(G') = P$, thus rendering $ub_{cp} = |CP(G')| = 5$ as an upper bound for $MRFC(R, C)$.

To calculate the longest length of colorful paths in a DAG $\vec{G}'$, we can employ the Dynamic Programming (DP) approach. In particular, let $N^+_{\vec{G}'}(u)$ and $N^-_{\vec{G}'}(u)$ represent the outgoing neighbors and incoming neighbors of u in $\vec{G}'$. The notation $f(i)$ indicates the number of vertices in a colorful path ending in i with the maximum size. Initially, the value of $f(i)$ is set to 1 for every vertex $i \in R \cup C$. Then, $f(i)$ can be calculated using the transition equation: $f(i) = (\max_{u \in N^-_{\vec{G}'}(i)} f(u)) + 1$.

The DP-based algorithm for calculating the largest size of colorful paths, referred to as ColorfulPathDP, is detailed in Algorithm 4. It commences by employing the degree-based greedy coloring algorithm to assign colors to the vertices in the graph G'. Subsequently, it constructs the DAG $\vec{G}'$ using a total ordering $\prec$ (lines 1-2). Following this, the algorithm initializes $f(i)$ to 1 for each vertex (line 3, lines 5-6) and computes $f(i)$ using a DP approach to yield the length of the longest colorful path ending at vertex i within $\vec{G}'$ (lines 7-11). During the DP process, ColorfulPathDP uses a variable $maxlen$ to maintain the number of vertices in a colorful path with the largest size in $\vec{G}'$, i.e., $maxlen = |CP(\vec{G}')|$. Finally, the algorithm outputs $maxlen$ as an upper bound of $MRFC(R, C)$ (line 12). The complexity of Algorithm 4 is presented in Theorem 7.

Theorem 7: The ColorfulPathDP algorithm requires $O(|V(G')| + |E(G')|)$ time for calculating ub_{cp}.

Remark. Table I provides details of the upper bounds. Notably, ub_{eac} is specifically designed for two-dimensional attribute graphs. For the other upper bounds, which can be extended to high-dimensional attribute graphs, we present generalized formulations in Table I that align with those in this section when $A_n = 2$.

V. Heuristic Algorithms

This section introduces a heuristic framework, namely, HeurRFC, to identify a larger fair clique within linear time. The framework relies on two key procedures: the degree-based greedy procedure, referred to as DegHeur, and the colorful-degree-based greedy procedure, known as ColorfulDegHeur. We begin by detailing DegHeur and ColorfulDegHeur before outlining the heuristic framework HeurRFC.

The degree-based greedy procedure. The degree-based greedy algorithm, i.e., DegHeur, computes a larger fair clique by iteratively selecting the vertex with the highest degree to augment R until further extension is not feasible. The pseudo-code of DegHeur is depicted in Algorithm 5. To ensure attribute fairness to the greatest extent feasible, DegHeur adopts an alternating attribute selection strategy similar to MaxRFC. However, a fundamental disparity exists: while MaxRFC endeavors to extend R for every vertex in C, DegHeur incorporates only the vertex with the highest degree to R. Specifically, during the iteration when a vertex with attribute a is chosen, DegHeur adds to R the vertex $v \in C$ that satisfies $v \leftarrow \max_{v \in C, A(v)=a} deg(v)$ (line 20). The algorithm terminates when C is empty, yielding R^* as a larger fair clique.

The colorful-degree-based greedy procedure. We introduce the colorful-degree-based greedy algorithm ColorfulDegHeur. Similar to DegHeur, ColorfulDegHeur employs a greedy strategy to extend the set R based on the colorful degree (as defined in Definition 2). To implement the ColorfulDegHeur algorithm, we make a slight modification to Algorithm 5. Specifically, we replace line 2 with $v \leftarrow \max_{v \in V} \min\{D_a(v), D_b(v)\}$ and line 20 with $v \leftarrow \max_{v \in C, A(v)=attr\_choose} \min\{D_a(v), D_b(v)\}$.

The heuristic framework. Algorithm 6 outlines the heuristic framework HeurRFC, encompassing both the degree-based and colorful-degree-based procedures. The main idea is to compute two fair cliques by invoking DegHeur and ColorfulDegHeur and then select the one with a larger cardinality. It is important to note that upon obtaining a fair clique R^*, its size can aid in graph pruning, as a larger fair clique is guaranteed to be within the $(|R^*| - 1)$-core subgraph (line 3 and line 8 in Algorithm 6). After performing DegHeur and ColorfulDegHeur, the HeurRFC algorithm recolors the remaining graph and establishes the upper bound of the maximum fair clique as the number of colors (lines 9-10). Finally, it outputs R^*, ub, and $color$ and terminates.

Remark. HeurRFC can be integrated into the branch-and-bound search algorithm MaxRFC to improve the efficiency for finding the maximum fair clique. Specifically, after MaxRFC performs EnColorfulSup for graph reduction, it can invoke the

Algorithm 5: DegHeur(G, k, δ)

```
   Input: G = (V, E, A), two integers k and δ
   Output: The fair clique R*
1  R* ← ∅;
2  v ← max_{v∈V} deg(v);
3  attr_choose ← a ∈ A − A(v);
4  HeurBranch({v}, N(v), attr_choose, R*, −1);
5  return R*;
6  Procedure HeurBranch(R, C, attr_choose, R*, a_max)
7  for u ∈ C do C_{A(u)} ← C_{A(u)} ∪ {u};
8  for u ∈ R do R_{A(u)} ← R_{A(u)} ∪ {u};
9  if C_{attr_choose} = ∅ and a_max = −1 then
10     a_min ← |R_{attr_choose}|;
11     a_max ← a_min + δ;
12 if |R_a| = a_max then C ← C − C_a; C_a ← ∅;
13 if |R_b| = a_max then C ← C − C_b; C_b ← ∅;
14 if C = ∅ then
15     R* ← R; return;
16 if C_{attr_choose} = ∅ then
17     attr_choose ← a ∈ A − attr_choose;
18     HeurBranch(R, C, attr_choose, R*, a_max);
19     continue;
20 v ← max_{v∈C, A(v)=attr_choose} deg(v);
21 attr_choose ← a ∈ A − A(v);
22 R̂ ← R ∪ {v};
23 Ĉ ← C ∩ N(v);
24 if |Ĉ| + |R̂| < k * 2 then return;
25 for v ∈ R̂ do cnt_R̂(A(v))++;
26 for v ∈ Ĉ do cnt_Ĉ(A(v))++;
27 if cnt_R̂(a) + cnt_Ĉ(a) < k or cnt_R̂(b) + cnt_Ĉ(b) < k then return;
28 HeurBranch(R̂, Ĉ, attr_choose, R*, a_max);
```

Algorithm 6: HeurRFC(G, k, δ)

```
   Input: G = (V, E, A), two integers k and δ
   Output: The fair clique R*, the upper bound ub, the color array color
1  R* ← the fair clique by performing DegHeur on G;
2  k* ← |R*| − 1;
3  G ← the k*-core of G;
4  R̂ ← the fair clique by performing ColorfulDegHeur on G;
5  if |R̂| > |R*| then
6      R* ← R̂;
7      k* ← |R*| − 1;
8      G ← the k*-core of G;
9  Color the graph G;
10 ub ← the number of colors in G;
11 return (R*, ub, color(·));
```

HeurRFC algorithm to yield a larger fair clique R^*. Then R^* can be utilized to prune the branch (R, C) during the processing of Branch when the upper bound of $MRFC(R, C)$ does not exceed $|R^*|$. Undoubtedly, a high-quality solution from HeurRFC significantly prunes search branches, thereby reducing the time consumption of the MaxRFC algorithm. In the experiments, we will compare the sizes of fair cliques found by HeurRFC and MaxRFC to demonstrate the effectiveness of our heuristic framework. In addition, although HeurRFC is designed for two-dimensional attribute graphs, the underlying idea can clearly be extended to the high-dimensional attribute graphs.

Theorem 8: The HeurRFC algorithm takes $O(|E| + |V|)$ time to output a fair clique with a larger size.

VI. Experiments

A. Experimental setup

Algorithms. We implement the colorful support based pruning algorithms, ColorfulSup (Algorithm 1) and EnColorfulSup, for graph reduction. We categorize the upper bounds ub_s, ub_a, ub_c, ub_{ac} and ub_{eac} into a group, denoted by ub_{AD}, called the advanced upper bound of $MRFC(R, C)$. For the maximum

TABLE II
DATASETS

Dataset	$n = \|V\|$	$m = \|E\|$	d_{max}	Description
Themarker	69,414	3,289,686	8,930	Social network
Google	875,713	8,644,102	6,332	Web network
DBLP	1,843,615	16,700,518	2,213	Collaboration network
Flixster	2,523,387	15,837,602	1,474	Social network
Pokec	1,632,803	44,603,928	14,854	Social network
Aminer	423,469	2,462,224	712	Collaboration network

fair clique search problem, we implement the basic framework MaxRFC (Algorithm 2) equipped with the following upper bounds to prune unpromising branches: (1) ub_{AD}; (2) $ub_{AD} + ub_{\triangle}$; (3) $ub_{AD} + ub_h$; (4) $ub_{AD} + ub_{cd}$; (5) $ub_{AD} + ub_{ch}$; (6) $ub_{AD} + ub_{cp}$. Furthermore, the heuristic framework HeurRFC is implemented (Algorithm 6) integrating both the degree-based greed method (Algorithm 5) and colorful-degree-based greed method. Additionally, we implement the versions of MaxRFC equipped with HeurRFC and the aforementioned upper bounds. 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 to 12 hours for all algorithms, and use the symbol "INF" to denote cases where the algorithm cannot terminate within 12 hours or run out of memory. For reproducibility, the source code of this paper is released on GitHub: https://github.com/fan2goa1/MaximumFairClique.

Remark. To determine the maximum fair clique, one strategy involves finding all relative fair cliques using the methods in [24] and subsequently selecting the clique with the highest vertex count. However, this approach is computationally intensive [24]. Furthermore, the graph reduction capability of the colorful core based technique presented in [24] is notably inferior to that of our colorful support based techniques implemented in MaxRFC. Additionally, in MaxRFC, we improve the efficiency by integrating the size-based upper bound to prune unpromising branches. Consequently, in our experiments, we employ the MaxRFC algorithm as the baseline.

Datasets. We utilize six real-world graphs to evaluate the efficiency of the proposed algorithms and the dataset statistics are summarized in Table II. Among these datasets, Aminer is an attributed graph where the attribute indicates the gender of scholars, available for download from https://github.com/SotirisTsioutsiouliklis/FairLaR/. The remaining datasets consist of non-attributed graphs accessible from networkrepository.com/ and snap.stanford.edu. For these non-attributed graphs, we generate attribute graphs by randomly assigning attributes to vertices with approximately equal probability to evaluate the efficiency of all algorithms.

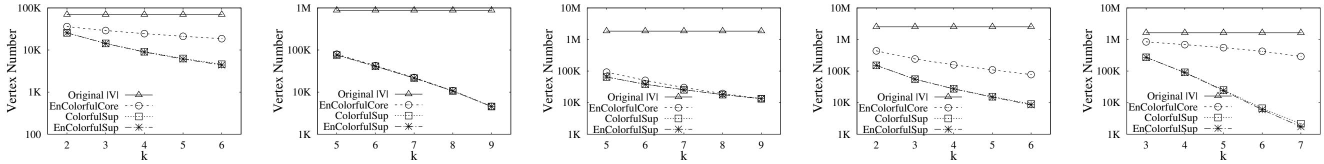
Parameters. In the maximum fair clique search problem, two parameters, k and δ, require consideration. Due to variations in dataset scales, we adjust the parameter k to different integers for each dataset. Specifically, for Aminer, k is chosen in the range of $[4, 8]$ with a default value of $k = 6$. For Themarker, we select k from the interval $[2, 6]$ with a default value of $k = 6$. For Google and DBLP, k ranges between $[5, 9]$, and the default value is $k = 7$. For Flixster, we consider k from $[2, 6]$, setting the default value as $k = 3$. Regarding Pokec, k varies within $[3, 7]$, with the default set to $k = 4$. As for the parameter δ, integer values within the range of $[1, 5]$ are considered, with a default value assigned as $\delta = 4$. In particular, for Themarker and Flixster, we set the default value of δ to be 3. During the variation of one parameter, the value of another parameter is maintained at its default setting.

B. Performance studies

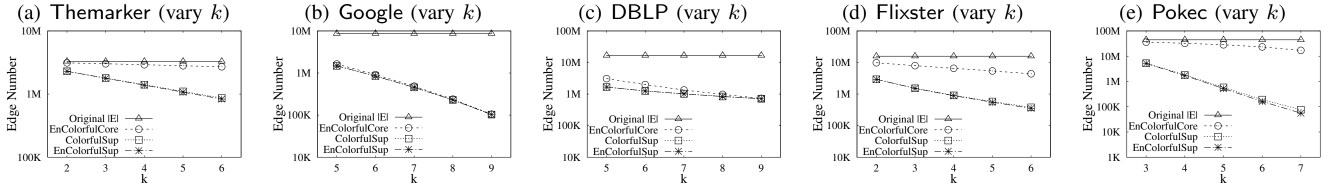
Evaluation of the graph reduction techniques. In this experiment, we evaluate the graph reduction techniques, namely, EnColorfulCore, ColorfulSup, and EnColorfulSup, by varying the value of k. The counts of remaining vertices and edges on datasets with generated attributes are depicted in Fig. 4. Notably, as the value of k increases, the number of vertices and edges left in the graph decreases across all reduction techniques. This is because, with larger values of k, the requirements for the enhanced colorful degree (resp., colorful support, enhanced colorful support) of vertices (resp., edges) within fair cliques become more rigorous. Consequently, only a few vertices and edges are able to fulfill these stringent requirements. Moreover, with a fixed k, EnColorfulCore, ColorfulSup and EnColorfulSup significantly reduce the number of vertices and edges compared to the initial graph. Both ColorfulSup and EnColorfulSup exhibit more robust graph reduction capabilities compared to EnColorfulCore, and EnColorfulSup outperforms ColorfulSup. This is owing to the fact that ColorfulSup builds upon EnColorfulCore by incorporating a constraint on the number of common neighbors with a specific attribute at the endpoints of an edge, i.e., the constraint on the colorful support of an edge. EnColorfulSup further extends ColorfulSup by assigning colors to specific attributes, imposing more stringent conditions on edges, and resulting in a more pronounced reduction in nodes and edges. For example, on the Pokec dataset with $k = 7$, sequentially applying EnColorfulCore, ColorfulSup and EnColorfulSup leaves 290,258, 2,155, and 1,735 vertices, with remaining edges numbering 17,004,374, 75,652, and 55,536, respectively. In contrast, the original graph contains 1,632,803 vertices and 44,603,928 edges. Additionally, we evaluate the performance of these three reductions using the Aminer dataset with real attributes, and the results shown in Fig. 5 align consistently with the previous findings.

We also investigate the impact of different coloring methods on reduction effectiveness. Since finding the optimal coloring is NP-hard and graph coloring is a preliminary step in our methods, we limit our comparison to approaches based on various greedy strategies that can provide a coloring solution in linear time. Specifically, we apply five greedy strategies to color the original graph: (1) largest-degree-first (LF), (2) dynamically largest-degree-first (DLF), (3) smallest-degree-last (SL), (4) smallest-ID-first (SIDF), and (5) random coloring (RD). We then record the number of colors used, as well as the number of remaining vertices and edges after applying the EnColorfulSup reduction technique. The experimental results are presented in Table IV. For the datasets Themarker, Flixster, and Pokec, the number of colors produced by the naive strategies (e.g., SIDF and RD) is significantly higher than that produced by the non-trivial strategies (e.g., LF, DLF, and SL). Consequently, the number of vertices and edges remaining after EnColorfulSup with the naive strategies is greater than that obtained using the non-trivial strategies, as expected. Among the non-trivial strategies, the number of colors obtained is similar, resulting in only minor differences in the scale of remaining graphs after applying EnColorfulSup. Therefore, using non-trivial strategies to color the original graphs is sufficient, and in this paper, we adopt the LF strategy, a degree-based coloring method mentioned earlier.

Evaluation of different upper bounds. We evaluate the runtime of the MaxRFC algorithms equipped with different upper



(a) Themarker (vary k) (b) Google (vary k) (c) DBLP (vary k) (d) Flixster (vary k) (e) Pokec (vary k)



(f) Themarker (vary k) (g) Google (vary k) (h) DBLP (vary k) (i) Flixster (vary k) (j) Pokec (vary k)

Fig. 4. Comparison of graph reduction techniques: EnColorfulCore, ColorfulSup and EnColorfulSup

TABLE III
RUNNING TIMES OF THE MaxRFC ALGORITHMS WITH DIFFERENT UPPER BOUNDS

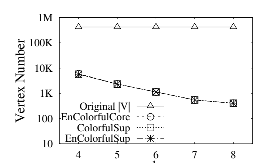
Dataset	k	The MaxRFC algorithms with different upper bounds (μs)					
		ub_{AD}	$ub_{AD}+ub_{\triangle}$	$ub_{AD}+ub_h$	$ub_{AD}+ub_{cd}$	$ub_{AD}+ub_{ch}$	$ub_{AD}+ub_{cp}$
Themarker	2	164,020,093	164,222,230	**163,785,612**	164,051,886	164,191,208	164,073,654
	3	156,447,185	156,455,589	**155,891,523**	156,514,092	156,114,447	156,206,675
	4	**133,397,225**	133,598,283	133,501,854	133,536,721	133,408,072	133,555,517
	5	111,368,194	**111,170,802**	111,552,467	111,195,109	111,248,057	111,198,219
	6	95,690,748	95,773,560	95,812,487	95,818,086	95,825,326	**95,608,156**
Google	5	13,296,055	13,221,049	13,219,447	**13,197,031**	13,200,569	13,207,587
	6	8,438,944	8,418,007	8,400,184	8,408,693	8,410,402	**8,399,664**
	7	5,608,029	5,594,834	**5,593,969**	5,595,033	5,599,214	5,598,307
	8	3,963,008	3,952,311	3,953,677	3,951,426	3,951,837	**3,951,291**
	9	3,112,872	3,109,023	3,108,917	**3,108,193**	3,108,959	3,108,725
DBLP	5	79,231,788	79,242,860	79,298,483	**79,215,098**	79,324,955	79,278,494
	6	65,693,405	65,693,550	65,717,597	**65,671,062**	65,693,812	65,691,195
	7	57,826,719	57,838,782	57,834,688	**57,806,109**	57,825,486	57,830,394
	8	52,304,894	52,316,988	52,316,524	**52,300,741**	52,310,454	52,304,072
	9	48,244,249	**48,224,779**	48,231,096	48,232,376	48,224,881	48,239,952
Flixster	2	116,217,884	113,383,872	**111,973,906**	114,089,180	113,714,458	114,033,281
	3	51,747,574	51,890,463	51,798,280	**51,559,466**	51,740,465	51,574,520
	4	40,146,859	40,173,220	40,170,887	**40,135,284**	40,155,296	40,163,524
	5	33,427,015	33,424,487	33,438,979	33,415,927	33,419,604	**33,413,852**
	6	28,680,932	28,699,177	28,688,229	**28,678,719**	28,685,011	28,690,077
Pokec	3	383,185,110	382,281,393	383,224,567	392,558,892	380,385,663	**379,412,432**
	4	179,717,984	179,859,519	180,397,350	179,407,754	180,891,601	**179,011,512**
	5	133,645,808	133,629,147	133,627,799	133,626,269	133,628,578	**133,623,669**
	6	123,720,463	123,714,946	123,714,386	**123,713,901**	123,714,644	123,716,296
	7	96,308,417	96,306,924	96,307,518	96,306,542	96,307,212	**96,306,375**
Aminer	4	1,740,023	**1,735,615**	1,736,197	1,735,819	1,735,884	1,736,148
	5	1,468,401	1,466,590	1,466,970	**1,466,422**	1,466,673	1,466,564
	6	1,399,833	1,398,941	**1,398,720**	1,398,862	1,399,046	1,398,811
	7	1,282,391	1,282,148	1,282,173	**1,281,954**	1,281,991	1,282,156
	8	1,251,681	1,251,316	**1,251,297**	1,251,460	1,251,342	1,251,482

Dataset	δ	The MaxRFC algorithms with different upper bounds (μs)					
		ub_{AD}	$ub_{AD}+ub_{\triangle}$	$ub_{AD}+ub_h$	$ub_{AD}+ub_{cd}$	$ub_{AD}+ub_{ch}$	$ub_{AD}+ub_{cp}$
Themarker	1	90,597,328	89,731,778	91,511,428	91,809,020	89,826,042	**89,395,928**
	2	**94,772,436**	96,119,426	95,119,312	94,986,264	98,534,905	95,162,120
	3	95,690,748	95,773,560	95,812,487	95,818,086	95,825,326	**95,608,156**
	4	94,292,236	**94,244,774**	97,198,799	99,452,775	98,857,373	101,292,596
	5	106,183,433	104,451,294	105,621,450	107,220,150	103,967,715	**103,817,481**
Google	1	5,615,173	5,595,760	5,598,803	5,596,462	**5,588,777**	5,596,590
	2	5,615,501	**5,592,032**	5,596,900	5,594,423	5,597,983	5,597,964
	3	5,614,339	5,597,891	**5,595,395**	5,599,553	5,596,092	5,595,905
	4	5,608,029	5,594,834	**5,593,969**	5,595,033	5,599,214	5,598,307
	5	5,610,155	5,596,797	5,598,475	5,597,520	5,595,962	**5,594,828**
DBLP	1	57,813,767	57,814,992	57,820,346	**57,798,765**	57,813,522	57,809,781
	2	57,817,797	57,817,311	57,821,456	**57,801,427**	57,815,387	57,805,805
	3	57,820,374	57,831,164	57,835,896	57,831,067	57,833,149	**57,816,605**
	4	57,826,719	57,838,782	57,834,688	**57,806,109**	57,825,486	57,830,394
	5	57,837,625	57,827,627	57,840,896	**57,819,094**	57,834,107	57,824,004
Flixster	1	51,498,237	51,532,023	51,582,030	**51,486,834**	51,531,231	51,529,883
	2	51,540,428	51,613,976	51,612,262	**51,534,845**	51,579,454	51,621,994
	3	51,747,574	51,890,463	51,798,280	**51,559,466**	51,740,465	51,574,520
	4	**51,651,691**	51,821,367	51,919,363	51,658,008	51,745,928	51,771,211
	5	**51,530,114**	51,589,462	51,586,067	51,536,750	51,782,115	51,568,905
Pokec	1	133,659,904	133,659,538	133,658,234	**133,647,298**	133,655,196	133,652,383
	2	133,653,055	133,649,730	133,646,114	**133,640,817**	133,649,393	133,641,592
	3	133,682,815	133,672,986	133,673,620	**133,666,725**	133,671,610	133,671,002
	4	133,645,808	133,629,147	133,627,799	133,626,269	133,628,578	**133,623,669**
	5	133,638,610	133,629,536	133,632,658	**133,619,445**	133,625,968	133,624,136
Aminer	1	1,399,201	1,398,504	1,398,299	**1,398,291**	1,398,325	1,398,557
	2	1,399,097	**1,398,257**	1,398,417	1,398,430	1,398,261	1,398,443
	3	1,399,251	1,399,251	1,398,751	1,398,520	**1,398,238**	1,398,703
	4	1,399,833	1,398,941	**1,398,720**	1,398,862	1,399,046	1,398,811
	5	1,399,996	**1,399,182**	1,399,489	1,399,286	1,399,864	1,399,603

TABLE IV
COMPARISON OF DIFFERENT ORDERING-BASED GREEDY COLORINGS

Dataset	Size	LF	DLF	SL	SIDF	RD
Themarker	*color*	57	59	**54**	86	84
	n	**4,403**	4,542	4,478	4,601	4,623
	m	**835,042**	861,474	853,114	874,676	879,276
Google	*color*	45	44	**44**	44	44
	n	21,520	21,520	21,520	21,520	21,520
	m	456,322	456,298	**456,272**	456,308	456,312
DBLP	*color*	280	280	280	280	280
	n	25,609	25,609	25,609	25,609	25,609
	m	1,013,924	**1,013,914**	1,013,926	1,013,926	1,013,930
Flixster	*color*	47	50	**41**	54	56
	n	**55,252**	55,362	55,378	55,429	55,394
	m	**1,512,264**	1,525,642	1,523,930	1,529,690	1,525,176
Pokec	*color*	33	32	**29**	43	43
	n	**89,534**	90,956	90,889	91,440	91,470
	m	**1,712,360**	1,802,134	1,791,550	1,826,926	1,830,668
Aminer	*color*	45	45	45	45	45
	n	1,131	1,131	1,131	1,131	1,131
	m	23,772	23,772	23,772	23,772	23,772

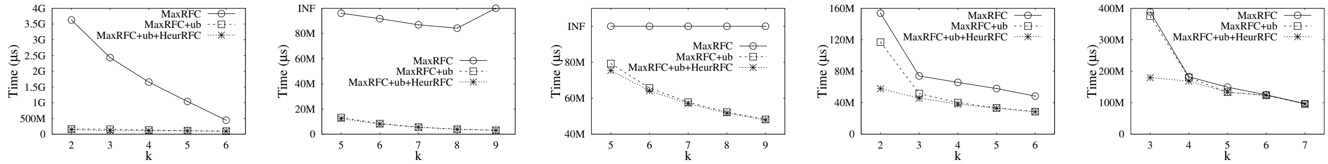
bounds with varying k and δ. These upper bounding pruning techniques are applied in MaxRFC when selecting vertices to be added to R for the first time. The running times of MaxRFC using various upper bounds are presented in Table III, with the minimum time highlighted. It can be observed that diverse datasets exhibit distinct characteristics, resulting in varying optimal upper bounds. Notably, the colorful-degeneracy-based upper bound and colorful-path-based upper bound achieve superior performance across a broader range of experimental settings. Although the running times of MaxRFC with different upper bounds do not exhibit considerable differences within the same dataset, employing these upper bounds in MaxRFC significantly reduces the runtime for the maximum fair clique search, as demonstrated in the subsequent experiments.



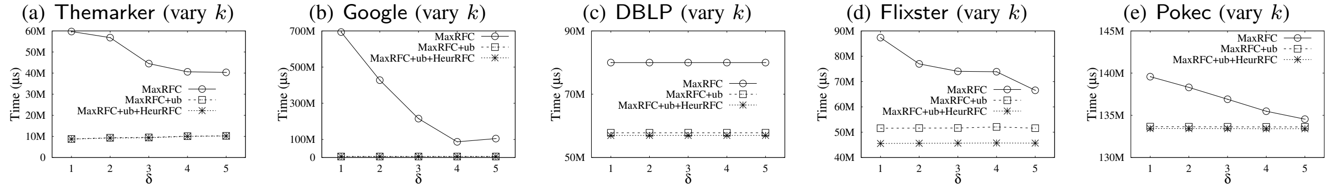
(a) Aminer (vary k) (b) Aminer (vary k)

Fig. 5. Comparison of graph reduction techniques on Aminer

Additionally, we present non-trivial upper bounds for each dataset in Table II after peforming graph reduction with $k = 2$ and $\delta = 5$. The results are depicted in Table V. Since the reduced graph may consist of multiple connected components, a specific upper bound for a dataset may correspond to multiple values. It can be observed that across these datasets, ub_{cd} is consistently no larger than ub_{ch}, while the relative size of ub_{cd} and ub_{cp} varies depend-



(a) Themarker (vary k) (b) Google (vary k) (c) DBLP (vary k) (d) Flixster (vary k) (e) Pokec (vary k)



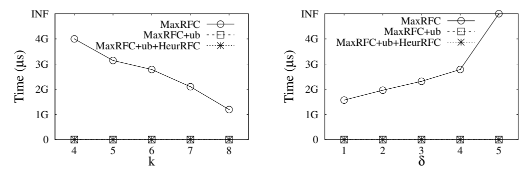
(f) Themarker (vary δ) (g) Google (vary δ) (h) DBLP (vary δ) (i) Flixster (vary δ) (j) Pokec (vary δ)

Fig. 6. Comparison of the MaxRFC algorithms

ing on the case. This aligns with the conclusion from our previous experiments that ub_{cd} and ub_{cp} generally perform better. We also conduct experiments on very dense graphs, which are commonly used in the evaluation of maximum clique search algorithms and represent more challenging cases. The datasets include: brock800_4, p_hat1500_3, MANN_a81, keller6, hamming10_4 and C4000.5, all of which can be downloaded from http://lcs.ios.ac.cn/~caisw/graphs.html. We generate two-dimensional attributes for these graphs as described earlier and then compute non-trivial upper bounds, with the results also shown in Table V. As observed, ub_{cp} is significantly smaller than ub_{cd} in these dense graphs, and once again, ub_{cd} is no larger than ub_{ch}. Based on these results, we suggest using either ub_{cd} or ub_{cp} for relatively sparse graphs, and only ub_{cp} for relatively dense graphs.

TABLE V
THE NON-TRIVIAL UPPER BOUNDS AFTER APPLYING GRAPH REDUCTION

Dataset	ub_{cd}	ub_{ch}	ub_{cp}	Dataset	ub_{cd}	ub_{ch}	ub_{cp}
Themarker	61	107	**57**	brock800_4	215	233	**142**
Google	**29**, 35, 37, **31**, 39	35, 43, 39, 31, 39	35, **33**, **33**, 45, **38**	p_hat1500_3	425	475	**284**
DBLP	**181**	181	280	MANN_a81	1081	1097	**830**
Flixster	59	85	**47**	keller6	877	1003	**781**
Pokec	**25**, 43	25, 61	27, **33**	hamming10_4	223	233	**128**
Aminer	**41**	41	45	C4000.5	665	703	**396**



(a) Aminer (vary k) (b) Aminer (vary δ)

Fig. 7. Comparison of the MaxRFC algorithms on Aminer

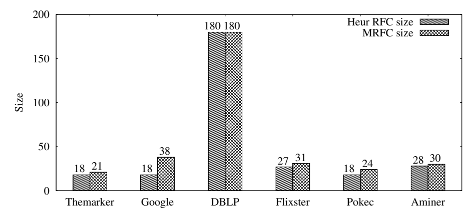
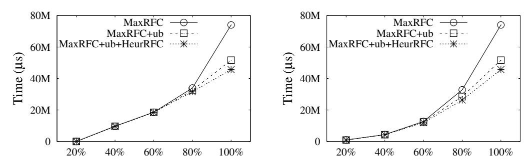


Fig. 8. The sizes of fair cliques found by MaxRFC and HeurRFC



(a) Flixster (vary m) (b) Flixster (vary n)

Fig. 9. Scalability test

Evaluation of the maximum fair clique search algorithms. We establish MaxRFC as the baseline and conduct a comparative analysis against two variations: MaxRFC with upper bounding technique, and MaxRFC with both upper bounding technique and HeurRFC. For each dataset, we select the optimal upper bound from Table III to apply as the upper bound in MaxRFC. Specifically, for Themarker, Google and Pokec, MaxRFC uses "$ub_{AD} + ub_{cp}$" as the upper bound, while for the other datasets, it employs "$ub_{AD} + ub_{cd}$" as the upper bound. The runtime of MaxRFC, MaxRFC+ub, and MaxRFC+ub+HeurRFC for finding the maximum fair clique is shown in Fig. 6 and Fig. 7. Note that in Fig. 6(b), "INF" indicates "Out of memory", while in Fig. 6(c) and Fig. 7, "INF" represents that the algorithm exceeds the predefined time limit. As can be seen, the running time of MaxRFC, MaxRFC+ub, and MaxRFC+ub+HeurRFC tends to decrease with increasing k due to fewer cliques satisfying fair clique constraints, expediting the identification of the maximum fair clique. Changes in δ do not exhibit a consistent trend in the runtime of these algorithms; rather, this seems to be influenced by the characteristics of the specific dataset. Notably, both MaxRFC+ub and MaxRFC+ub+HeurRFC exhibit significantly faster execution times compared to MaxRFC. This performance enhancement can be credited to the use of the upper-bound-based pruning techniques and the integration of the heuristic-result-based pruning. Concerning the MaxRFC+ub+HeurRFC algorithm, although its runtime is marginally lower than that of MaxRFC+ub, these results suggest the contribution of HeurRFC to the efficiency of the maximum fair clique search process. For instance, on the Flixster, when $k = 2$, MaxRFC+ub and MaxRFC+ub+HeurRFC run approximately 15 and 20 times faster than MaxRFC, respectively. These results underscore the efficiency of the proposed upper bound pruning techniques and the heuristic algorithm.

The effectiveness of the heuristic algorithm. We evaluate the effectiveness of HeurRFC by comparing the size of the fair clique it finds with the size of the maximum fair clique. The results are depicted in Fig. 8. Clearly, across most datasets, the fair clique discovered by HeurRFC is very close in size to the maximum fair clique, with differences of no more than 6. Notably, on DBLP, the HeurRFC algorithm outputs a fair clique of the same size as the maximum fair clique. These results demonstrate that our HeurRFC algorithm can indeed yield a fair clique of larger size within linear time, making it a valuable tool for pruning the search space in MaxRFC.

Scalability testing. We create four subgraphs for each dataset by randomly selecting 20%-80% of vertices and edges to evaluate the scalability of the maximum fair clique search algorithms. The results on Flixster are presented in Fig. 9. Similar outcomes are expected for the other datasets, though they are not shown here due to space limits. As can be seen, MaxRFC exhibits a steep rise in running time with increasing m or n, whereas MaxRFC+ub and MaxRFC+ub+HeurRFC show a more gradual increase. Again, the runtime of MaxRFC is notably longer compared to MaxRFC+ub and MaxRFC+ub+HeurRFC. These results confirm the superior scalability of the MaxRFC+ub and MaxRFC+ub+HeurRFC algorithms in handling large-scale graphs.

C. Case study

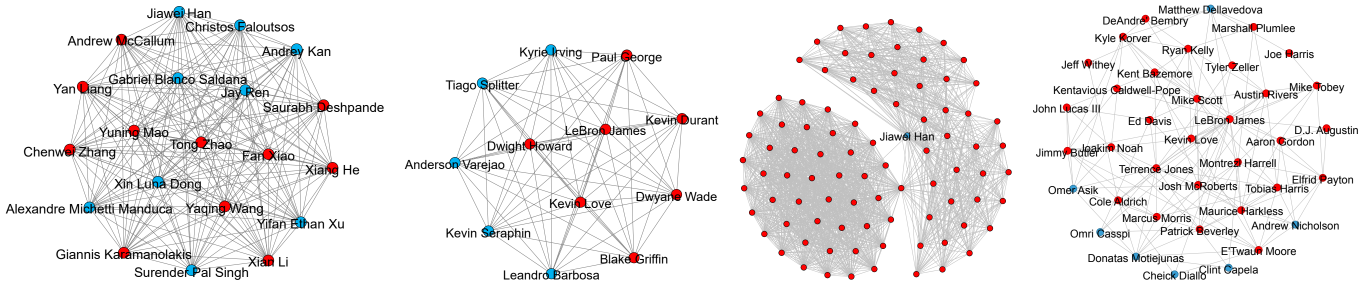
Case study on DBAI. We conduct a case study on a collaboration network DBAI. The DBAI dataset is a subgraph of DBLP downloaded from dblp.uni-trier.de/xml/, which contains the authors who had published at least one paper in the database (DB) and artificial intelligence (AI) related conferences. The subgraph contains 139,675 vertices and 975,722 undirected edges. The attribute A represents the author's main research area, i.e., $A = \{DB, AI\}$. We assign the attribute for each vertex based on the maximum number of papers an author published in the related conferences. Performing our algorithms with $k = 5$ and $\delta = 3$, the maximum fair clique is depicted in Fig. 10(a), which includes 9 scholars specializing in DB (colored blue) and 11 in AI (colored red), maintaining a difference within δ between the scholar counts of each research field. These scholars have garnered considerable recognition within databases and artificial intelligence. For instance, Prof. Jiawei Han focuses on knowledge discovery, data mining, and database systems, boasting an impressive h-index of 200. Similarly, Prof. Andrew McCallum's expertise lies in statistical machine learning, natural language processing, and information retrieval, reflected in his h-index of 117. When embarking on a research project that demands a blend of database and machine learning expertise, our algorithms come to the fore. They identify the largest and most specialized cohort, ensuring equilibrium in participant numbers across the two distinct research directions.

Additionally, the maximum fair clique size can illuminate the intersecting degree between these two different research directions. The minuscule size of the maximum fair clique implies limited linkage between the two directions, while a larger maximum fair clique suggests a robust interconnection. Insights derived from our algorithms can guide interdisciplinary collaborations and research initiatives.

Case study on NBA. The NBA dataset, sourced from https://github.com/yushundong/PyGDebias, contains 403 basketball players and 21,242 relationships. Players' nationalities serve as attributes, i.e., $A = \{U.S., Oversea\}$. Invoking specified parameters of $k = 5$ and $\delta = 3$, our algorithms determine a maximum fair clique, illustrated in Fig. 10(b). Red vertices represent 7 U.S. players, while blue vertices denote 5 players from overseas. All these individuals are widely renowned NBA stars, connected either through shared team histories or robust personal friendships. For instance, LeBron James, Kyrie Irving, and Kevin Love were core players for the Cavaliers, contributing to their 2016 NBA championship win. Dwyane Wade and LeBron James formed a dynamic partnership while playing together for the Miami Heat, securing two NBA championships. Anderson Varejao, Leandro Barbosa, and Tiago Splitter, representing Brazil, have collectively competed in prestigious international basketball events like the Olympics and World Cup, fostering a strong camaraderie through national team participation. Discovering a dense organization with a large size that encompasses a nearly equivalent count of foreign and local stars by our algorithms holds significant potential for sports clubs, athletes, and brands. This potential extends to attracting a broader fan base, expanding exposure, enhancing brand recognition, and ultimately amplifying the impact of their social media marketing endeavors.

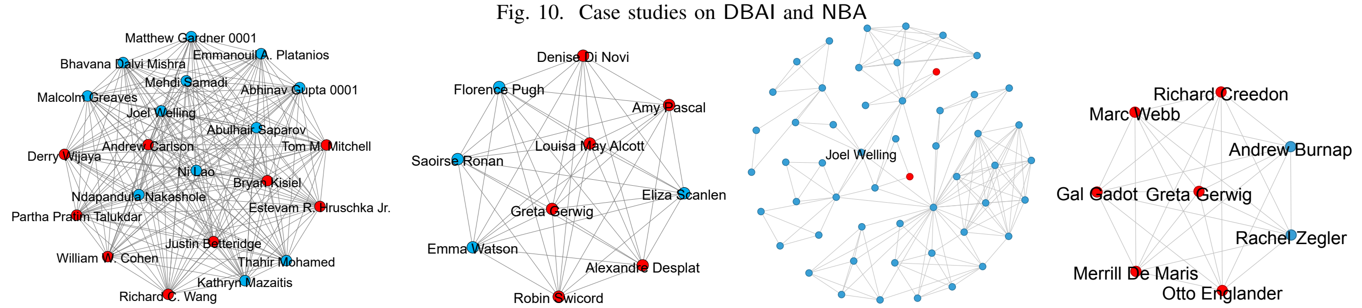
Case study on Aminer. We conduct a case study on Aminer to evaluate the effectiveness of our algorithms. The attribute A in Aminer indicates the gender of the author, i.e., $A = \{male, female\}$. With $k = 5$ and $\delta = 3$, we invoke the proposed algorithms to find the maximum fair clique. Fig. 11(a) shows the result with 13 males (colored blue) and 16 females (colored red). It maintains a balance, ensuring the count of males and females is not less than k, with a difference between them not exceeding δ. The scholars in Fig. 11(a) primarily affiliate with two establishments: the smart HCI lab of the ICxT Innovation center at the University of Turin and Telecom Italy Company. Their focus areas span human-computer interaction, information visualization, and multimodal interaction. Notably, five scholars boast a Google Scholar impact exceeding 2,000. Further validation through the HCI Lab's official website confirms a longstanding partnership with Telecom Italy, involving collaborative projects like Personalised Television Services, E-Tourism-Context-Aware Systems, and ICT Converging Technologies 2008-PIEMONTE, among others. These findings underscore the effectiveness of our algorithms in identifying large, well-connected teams renowned in the field of human-computer interaction. Within these collectives, scholars of diverse genders leverage their individual expertise, culminating in a robust and adept collaborative force.

Case study on IMDB. We conduct a case study on a movie dataset obtained from https://developer.imdb.com. Filtering out movies categorized as $titleType = movie$ and $isAdult = 0$, we create a graph IMDB. This graph comprises 583,933 vertices representing actors, directors, writers, and others, connected by 29,332,894 edges indicating their collaborations. Each vertex is associated with an attribute from $A = \{S, J\}$, where S represents a senior artist and J denotes a junior artist. This categorization is based on birth year: with individuals born before 1990 classified as S and those born after as J. Using our algorithms with parameters $k = 5$ and $\delta = 3$, we identify the maximum fair clique as depicted in Fig. 11(b). The team connected to the film "Little Women" intricately combines 4 junior artists (colored blue) and 6 senior artists (colored red). Among them, Louisa May Alcott is the novelist behind the film's source material, and Greta Gerwig takes on the directorial role. Denise Di Novi, Robin Swicord, and Amy Pascal manage production aspects. Alexandre Desplat contributes his musical talents to compose the soundtrack, and the others are accomplished actors. This movie boasted an IMDB rating of 7.8 and earned a place among the top 10 movies of the year according to the American Film Institute. It also secured nominations at esteemed award ceremonies like the Academy Awards, BAFTAs, and Golden Globes. This serves as evidence that a diverse team comprising both young and seasoned artists can blend creativity, expertise, and experience to elevate the quality of cinematic production.



(a) DBAI, MRFC (b) NBA, MRFC (c) DBAI, ATC (d) NBA, CDE

Fig. 10. Case studies on DBAI and NBA



(a) Aminer, MRFC (b) IMDB, MRFC (c) Aminer, ATC (d) IMDB, ATC

Fig. 11. Case studies on Aminer and IMDB

Identifying such a team through our algorithms and investing in it can yield substantial returns.

Comparison with other attributed community models. We compare the maximum fair clique model with three attributed community models: CDE, ATC, and AC. The CDE model is used for community detection, and we output the largest of the resultant communities for comparison. Since the CDE model is embedding-based and struggles to handle large-scale graph data, we apply it only on the small NBA dataset. In contrast, the ATC and AC models are used for community queries and rely on a query vertex and an attribute set. For this reason, we compute the communities based on ATC and AC models by selecting the query vertex as one from the maximum fair cliques of DBAI, Aminer, and IMDB in Fig. 10(a), Fig. 11(a) and Fig. 11(b), and setting the input attribute set to include all possible values of an attribute. The experimental results are shown in Fig. 10(c), Fig. 11(c) and Fig. 11(d).

As seen, the ATC model fails to identify fair communities. For instance, in the Aminer dataset (Fig. 11(c)), the ATC-based community containing "Joel Welling" comprises 53 nodes, but only 2 are female, revealing a clear gender bias. Similar patterns of unfairness are observed in the DBAI and IMDB datasets (Fig. 10(c) and Fig. 11(d)). The CDE model is also unable to ensure fairness in the detected communities. For example, in the NBA dataset (Fig. 10(d)), the largest community consists of 37 vertices, but only 7 are overseas basketball players, highlighting a significant nationality bias. For the AC model, we do not present the results; instead, we analyze its lack of fairness by using gender as an illustrative attribute. The AC model maximizes the intersection of shared attributes among vertices and the set of input attributes, with each vertex possessing only a single attribute (e.g., male or female). If a community contains two vertices with different genders, there are no shared attributes, resulting in a community score of 0. Therefore, the community based on AC is limited to vertices with the same attribute (i.e., all male or all female), and fairness cannot be guaranteed. In contrast, the communities identified by the fair clique model ensure that the number of vertices with different attributes within a community remains balanced. These results illustrates the effectiveness of the fair clique model.

D. Discussion

Guidelines for parameter settings in the fair clique model. The fair clique is based on an attributed graph $G = (V, E, A)$ and two parameters k and δ. The attribute A is specified by the user according to their practical requirements, i.e., determining which attribute should be used to ensure fairness. For selecting suitable values for k and δ, we provide the following guidelines. For the parameter k, since the size of the fair clique is clearly no larger than the size of maximum clique in a graph, denoted as C_{max}, a binary search method can be applied to identify an appropriate value of k within the interval $[1, \lfloor C_{max}/A_n \rfloor]$. If C_{max} is unknown, we recommend, based on experimental observations, that k should not be set too large. It is preferable to choose k within the interval $[3, 7]$. Regarding the parameter δ, which represents the allowable difference in the number of vertices with different attributes, we suggest keeping δ small to maintain community fairness. Therefore, we recommend setting δ within the range $[1, 6]$.

Guidelines for selecting pruning upper bounds. Appropriate upper bound techniques are essential for ensuring the effectiveness of the maximum fair clique search algorithm. To support this process, we outline guidelines for selecting the most suitable bounds. Among the trivial bounds, the size-based upper bound ub_s is the easiest to compute, taking only $O(1)$ time, though it is also the least tight. For the upper bounds ub_a, ub_c and ub_{ac}, each with $O(|V(G')|)$ time complexity, we have $ub_a \geq ub_c$ and $ub_a \geq ub_{ac}$. Additionally, when there is a significant difference between the color counts of distinct attributes, $ub_{ac} \leq ub_c$. Notably, the enhanced-attribute-color-based upper bound ub_{eac} applies only to 2-dimensional attribute graphs. The degeneracy-based upper bound $ub_{\triangle}$ and the h-index-based upper bound ub_h are applied to the maximum clique search problem without incorporating attribute or color information, which typically makes them very large and less effective for our problem. Based on this analysis, we recommend using ub_s initially, followed by a combination of ub_c and ub_{ac} for the trivial bounds. For non-trivial bounds, we derive that the colorful-h-index-based upper bound ub_{ch} is no less than the colorful-degeneracy-based upper bound ub_{cd}. However, it remains unclear whether ub_{cd} is larger than the colorful-path-based upper bound ub_{cp}. Based on our experimental results, we suggest using either ub_{cd} or ub_{cp} for relatively sparse graphs, and only ub_{cp} for relatively dense graphs.

VII. Related work

Maximum clique computation. Our work is closely related to the Maximum Clique Computation (MCC) problem, aiming to find the clique with the largest number of nodes. The MCC problem falls into the domain of NP-hard problems [36]. Existing research primarily centers on devising heuristic algorithms that approximate solutions close to the maximum clique size. These heuristic algorithms iteratively augment the partial clique R by adding vertices from the candidate set C based on specific greedy strategies until C is empty. For example, the maximum degree-based heuristic greedily selects the vertex with the highest degree to extend R in each iterative step [37], while the degeneracy order-based heuristic prioritizes vertices with the largest degeneracy for inclusion into R [38]. The ego-centric degeneracy-based heuristic extends the degeneracy order-based approach to each vertex's ego network and identifies the largest one as the result [3], [39]. On the other hand, effective exact methods for the MCC problem are also extensively studied, primarily based on the branch and bound framework. These exact methods consider every possible vertex addition to the partial clique R to form a new search branch and often employ upper bound-based pruning techniques to improve search efficiency [3], [5], [39]–[42]. Chang *et al.* presented a state-of-the-art algorithm for the MCC problem, transforming the MCC problem on sparse graphs into multiple dense graphs. They also provided a branch-reduce-bound framework to compute the maximum clique on dense graphs [3], [39]. Due to the inherent differences between clique and fair clique concepts, all the aforementioned algorithms cannot be directly applied to address our problem.

Fairness-aware data mining. Our work is motivated by the concept of fairness. It has attracted much attention in the machine learning research area, such as the classification task [15]–[18], [43], [44] and the recommendation task [19]–[22], [45], [46]. Within the field of data mining, Pan *et al.* blazed a trail by introducing fairness into the clique model, proposing both weak and strong fair clique models, as well as a suite of enumeration algorithms [23]. Based on this foundation, Zhang *et al.* introduced the relative fair clique model, offering a compromise between weak and strong fair clique models [24]. Hao *et al.* defined the absolute fair clique model and studied the problem of finding absolute fair cliques from attributed social networks [25]. Qiao *et al.* incorporated fairness into the KPcore model, formulating the maximum core mining problem on heterogeneous information networks [27]. In addition, Yin *et al.* focused on fairness within bipartite graphs, introducing the single-side and bi-side fair bicliques, and studied the problem of fairness-aware biclique enumeration [26]. This paper, for the first time, investigates the problem of finding the relative fair clique with the largest size. Among the mentioned studies, only the relative fair clique enumeration algorithms introduced in [24] possess adaptability for solving our problem. However, these algorithms tend to exhibit inefficiency, especially when dealing with large graphs. In light of this, we propose efficient graph reduction techniques and deploy a series of upper-bounding pruning techniques to enhance the efficiency of finding the maximum fair clique.

VIII. Conclusion

This paper studies the problem of finding the maximum fair clique in large graphs. Two novel graph reduction techniques grounded in colorful support are presented, aimed at shrinking graph size. Then, we propose a series of upper-bounding techniques to prune needless search space during the branch-and-bound procedure. Adding to this, a linear time complexity heuristic algorithm based on degree and colorful degree greedy strategies is presented for finding a larger fair clique, which can also be used to prune branches to further improve search efficiency. Comprehensive experiments on six real-life graphs demonstrate the efficiency, scalability and effectiveness of the proposed algorithms.

References

[1] L. Chang and L. Qin, *Cohesive subgraph computation over large sparse graphs: algorithms, data structures, and programming techniques*, 2018.

[2] C. Bron and J. Kerbosch, "Finding all cliques of an undirected graph (algorithm 457)," *Commun. ACM*, vol. 16, no. 9, pp. 575–576, 1973.

[3] L. Chang, "Efficient maximum clique computation over large sparse graphs," in *KDD*, 2019, pp. 529–538.

[4] D. Eppstein, M. Löffler, and D. Strash, "Listing all maximal cliques in large sparse real-world graphs," *Journal of Experimental Algorithmics*, vol. 18, pp. 3–1, 2013.

[5] C.-M. Li, H. Jiang, and F. Manyà, "On minimization of the number of branches in branch-and-bound algorithms for the maximum clique problem," *Computers & Operations Research*, vol. 84, pp. 1–15, 2017.

[6] P. San Segundo, A. Lopez, and P. M. Pardalos, "A new exact maximum clique algorithm for large and massive sparse graphs," *Computers & Operations Research*, vol. 66, pp. 81–94, 2016.

[7] Y. Li, C. Sha, X. Huang, and Y. Zhang, "Community detection in attributed graphs: An embedding approach," in *AAAI*, 2018, pp. 338–345.

[8] Y. Fang, R. Cheng, S. Luo, and J. Hu, "Effective community search for large attributed graphs," *Proc. VLDB Endow.*, vol. 9, no. 12, pp. 1233–1244, 2016.

[9] A. Khan, L. Golab, M. Kargar, J. Szlichta, and M. Zihayat, "Compact group discovery in attributed graphs and social networks," *Inf. Process. Manag.*, vol. 57, no. 2, p. 102054, 2020.

[10] C. Pizzuti and A. Socievole, "A genetic algorithm for community detection in attributed graphs," in *Applications of Evolutionary Computation - 21st International Conference, EvoApplications 2018, Parma, Italy, April 4-6, 2018, Proceedings*, ser. Lecture Notes in Computer Science, vol. 10784, 2018, pp. 159–170.

[11] Z. Xu, Y. Ke, Y. Wang, H. Cheng, and J. Cheng, "A model-based approach to attributed graph clustering," in *SIGMOD*. ACM, 2012, pp. 505–516.

[12] J. Yang, J. McAuley, and J. Leskovec, "Community detection in networks with node attributes," in *ICDM*, 2013, pp. 1151–1156.

[13] Y. Wu, Z. Zhong, W. Xiong, and N. Jing, "Graph summarization for attributed graphs," in *ISEEE*, vol. 1, 2014, pp. 503–507.

[14] X. Huang and L. V. S. Lakshmanan, "Attribute-driven community search," *Proc. VLDB Endow.*, vol. 10, no. 9, pp. 949–960, 2017.

[15] A. Cotter, H. Jiang, and K. Sridharan, "Two-player games for efficient non-convex constrained optimization," in *ALT*, ser. Proceedings of Machine Learning Research, vol. 98, 2019, pp. 300–332.

[16] H. Narasimhan, "Learning with complex loss functions and constraints," in *AISTATS*, ser. Proceedings of Machine Learning Research, vol. 84, 2018, pp. 1646–1654.

[17] B. E. Woodworth, S. Gunasekar, M. I. Ohannessian, and N. Srebro, "Learning non-discriminatory predictors," in *COLT*, ser. Proceedings of Machine Learning Research, vol. 65, 2017, pp. 1920–1953.

[18] R. S. Zemel, Y. Wu, K. Swersky, T. Pitassi, and C. Dwork, "Learning fair representations," in *ICML*, ser. JMLR Workshop and Conference Proceedings, vol. 28, 2013, pp. 325–333.

[19] A. Singh and T. Joachims, "Fairness of exposure in rankings," in *KDD*, 2018, pp. 2219–2228.

[20] Ashudeep Singh and Thorsten Joachims, "Policy learning for fairness in ranking," in *NeurIPS*, 2019, pp. 5427–5437.

[21] A. Asudeh, H. V. Jagadish, J. Stoyanovich, and G. Das, "Designing fair ranking schemes," in *SIGMOD*, 2019, pp. 1259–1276.

[22] A. Beutel, J. Chen, T. Doshi, H. Qian, L. Wei, Y. Wu, L. Heldt, Z. Zhao, L. Hong, E. H. Chi, and C. Goodrow, "Fairness in recommendation ranking through pairwise comparisons," in *KDD*, 2019, pp. 2212–2220.

[23] M. Pan, R. Li, Q. Zhang, Y. Dai, Q. Tian, and G. Wang, "Fairness-aware maximal clique enumeration," in *ICDE*, 2022, pp. 259–271.

[24] Q. Zhang, R.-H. Li, M. Pan, Y. Dai, Q. Tian, and G. Wang, "Fairness-aware maximal clique in large graphs: Concepts and algorithms," *IEEE TKDE*, 2023.

[25] F. Hao, Y. Yang, J. Shang, and D.-S. Park, "Afcminer: Finding absolute fair cliques from attributed social networks for responsible computational social systems," *IEEE TCSS*, 2023.

[26] Z. Yin, Q. Zhang, W. Zhang, R. Li, and G. Wang, "Fairness-aware maximal biclique enumeration on bipartite graphs," *CoRR*, vol. abs/2303.03705, 2023.

[27] L. Qiao, H. Hou, and G. Wang, "Community search algorithm on heterogeneous information networks based on attribute fairness," *Journal of Software*, vol. 34, no. 3, pp. 0–0, 2022.

[28] D. W. Matula, G. Marble, and J. D. Isaacson, "Graph coloring algorithms," in *Graph theory and computing*, 1972, pp. 109–122.

[29] T. R. Jensen and B. Toft, *Graph coloring problems*, 2011, vol. 39.

[30] W. Hasenplaugh, T. Kaler, T. B. Schardl, and C. E. Leiserson, "Ordering heuristics for parallel graph coloring," in *SPAA*, 2014, pp. 166–177.

[31] D. R. Lick and A. T. White, "k-degenerate graphs," *Canadian Journal of Mathematics*, vol. 22, no. 5, pp. 1082–1096, 1970.

[32] S. B. Seidman, "Network structure and minimum degree," *Social networks*, vol. 5, no. 3, pp. 269–287, 1983.

[33] J. E. Hirsch, "An index to quantify an individual's scientific research output," *Proc. Natl. Acad. Sci. USA*, vol. 102, no. 46, pp. 16 569–16 572, 2005.

[34] J. Wang, J. Cheng, and A. W. Fu, "Redundancy-aware maximal cliques," in *KDD*, 2013, pp. 122–130.

[35] T. Eden, D. Ron, and C. Seshadhri, "On approximating the number of k-cliques in sublinear time," in *STOC*, 2018, pp. 722–734.

[36] R. M. Karp, "Reducibility among combinatorial problems," in *CCC*, ser. The IBM Research Symposia Series, 1972, pp. 85–103.

[37] B. Pattabiraman, M. M. A. Patwary, A. H. Gebremedhin, W. Liao, and A. N. Choudhary, "Fast algorithms for the maximum clique problem on massive graphs with applications to overlapping community detection," *Internet Math.*, vol. 11, no. 4-5, pp. 421–448, 2015.

[38] R. A. Rossi, D. F. Gleich, and A. H. Gebremedhin, "Parallel maximum clique algorithms with applications to network analysis," *SIAM J. Sci. Comput.*, vol. 37, no. 5, 2015.

[39] Lijun Chang, "Efficient maximum clique computation and enumeration over large sparse graphs," *VLDB J.*, vol. 29, no. 5, pp. 999–1022, 2020.

[40] C. Li, Z. Fang, and K. Xu, "Combining maxsat reasoning and incremental upper bound for the maximum clique problem," in *ICTAI*, 2013, pp. 939–946.

[41] E. Tomita, "Efficient algorithms for finding maximum and maximal cliques and their applications," in *WALCOM*, ser. Lecture Notes in Computer Science, vol. 10167, 2017, pp. 3–15.

[42] E. Tomita, Y. Sutani, T. Higashi, S. Takahashi, and M. Wakatsuki, "A simple and faster branch-and-bound algorithm for finding a maximum clique," in *WALCOM*, ser. Lecture Notes in Computer Science, vol. 5942, 2010, pp. 191–203.

[43] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. S. Zemel, "Fairness through awareness," in *ITCS*, 2012, pp. 214–226.

[44] M. Hardt, E. Price, and N. Srebro, "Equality of opportunity in supervised learning," in *NeurIPS*, 2016, pp. 3315–3323.

[45] A. J. Biega, K. P. Gummadi, and G. Weikum, "Equity of attention: Amortizing individual fairness in rankings," in *SIGIR*, 2018, pp. 405–414.

[46] M. Zehlike and C. Castillo, "Reducing disparate exposure in ranking: A learning to rank approach," in *WWW*, 2020, pp. 2849–2855.