Fair Group Summarization with Graph Patterns

Hanchao Ma*, Sheng Guan*, Mengying Wang*, Qi Song† and Yinghui Wu*
*Case Western Reserve University †University of Science and Technology of China
{hxm382,sxg967, mxw767, yxw1650}@case.edu qisong09@ustc.edu.cn

***Abstract*—Given a set of node groups in a graph (*e.g.,* gender or race groups), how to succinctly summarize their neighbors, and meanwhile ensure a "fair" representation to mitigate under- or over-representation of a certain group? We propose a novel framework to compute concise summaries of node groups with fairness guarantees. (1) We introduce a pattern-correction summary structure called r-summaries. An r-summary uses a graph pattern set to specify representative nodes and their connectivity patterns, and an auxiliary edge correction set to losslessly describe their r-hop links. (2) Given a set of node groups, we introduce the fair group summarization problem, which is to compute an r-summary that can select and accurately describe high quality nodes and their neighbors with small edge corrections, and meanwhile guarantee a desirable coverage for each group. The need for generating such summaries is evident in social recommendation, healthcare and graph search. We show that the problem is Σ_2^p-complete with the verification component already NP-complete. (3) We present an algorithm that can generate summaries with (a) guaranteed quality, and (b) a relative approximation on maximum coverage. For large groups, we introduce an efficient algorithm that interleaves node selection and localized pattern discovery to reduce unnecessary computation. In addition, we introduce an algorithm to incrementally maintain the r-summaries over dynamic graphs with evolving edges. Using real-world data, we experimentally verify the efficiency and effectiveness of our algorithms to compute and maintain r-summaries, and verify their applications.**

I. Introduction

Graph summarization has been used to support large-scale graph analysis [25]. Given a graph G, it is to generate compact summary structures $\mathcal{S}$ that (approximately) represent G that also preserves certain properties with queryable structures. A common practice is to follow Minimum Description Length (MDL) principle, which aims to minimize the size of summaries and the corresponding description length of the graph. This is often implemented by frequent pattern mining [45] in favor of subgraphs with a high compression rate, to support downstream tasks such as graph search [28], community detection [6] or influence analysis [23].

Emerging graph analysis with fairness requirements [36], [38], nevertheless, poses new challenges. A common scenario interprets fairness as group coverage constraints [43], [16], [31], [27]. Given a set of node groups, it is desirable to (1) select and concisely describe a set of representative nodes with desirable quality from each group, and (2) ensure a satisfactory "coverage" of each group to prevent under- or over-presentation of certain groups. In practice, such groups may refer to vulnerable social determined by groups *e.g.,* gender, race or professions [13], relevant yet under-represented

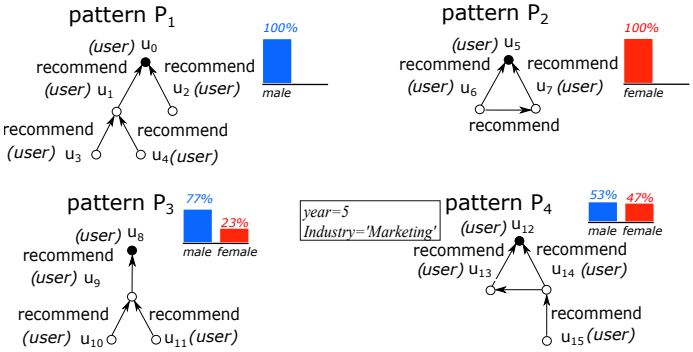


Fig. 1. Summarizing Social Connections in Talent Search.

topics [2], recommendations [15], or designated columns for query benchmarking [4]. Consider the following scenarios.

Example 1: [Talent Search]. Consider a real-world social network G [15] where each node in G denotes a user with attributes such as *title* and *skill*. Each edge indicates a recommendation (recommend) between users. We illustrate two most frequent subgraph patterns P_1 and P_2 (illustrated in Fig. 1), which are separately mined from sub-networks of G that are induced by male-only and female-only users, respectively. They interestingly demonstrate that male and female professional users in general have quite different social connectivity patterns. For example, female professions may favor more active interactions in a small social community (dual networks or "inner circles"), while male users benefit from "high centrality" patterns, as also observed in [46].

A recruiter wants to explore G to promote talent search with "equal opportunity" [15], for which a set of candidates with balanced gender distribution are preferred. She may also want to understand the social connections of these candidates to improve talent search. Neither P_1 or P_2 satisfies such requirement due to their bias to a specific gender. A desirable summary structure with graph patterns should draw almost equal number of male and female users, and describe their neighborhood via graph pattern matching as accurately as possible, to guide the talent search. □

One may consider summarization with frequent subgraphs. However, this may lead to a "skewed" distribution towards majority groups, leading to biased analysis. Another option is to "diversify" these patterns to cover different nodes. Nevertheless, it is not easy to ensure coverage for each group.

Example 2: Consider a graph pattern P_3 in Fig. 1 computed

via frequent pattern mining [10], which is among the ones with the highest support. It indeed covers a large population of the groups. Nevertheless, the users that match u_8 come with a biased gender distribution of 77% male and 23% of female. This is close to the actual distribution of the gender groups in G. It suggests that frequent patterns are sensitive to the skewed gender distribution and over-present the majority groups. Patterns like P_3, if suggested as queries, may lead to both biased search results towards male candidates, but also suggest biased understanding of how talented candidates benefit from social patterns (*e.g.,* "high centrality" only). □

Example 3: [Pandemic Analysis]. In a real-world pandemic spreading network [1], each node denotes a citizen with personal information such as *age groups*, *gender* and (infectious) *history*. Each edge represents routine close contact (contact) between two citizens [47]. Given a budget k of vaccines, a policy maker will need to choose n citizens as "seed" set to apply the vaccines to control the expected spread of the pandemic following the network. That is, she wants to select k nodes that may maximize the spread of the pandemic if no vaccine is given, under a (monotonic submodular) influence maximization function [47] (group immunization). Meanwhile, she wants to investigate the impact of different age distribution to the spread by enforcing configurable coverage constraints to different age groups of the seed set, and to extract their common social connection to better understand the propagation mechanism. Existing frequent pattern discovery and graph summarization methods cannot be used to find summary structures that meanwhile satisfy the configurable coverage requirement over age groups. □

The above examples call for effective graph summary structures that can *simultaneously* support (1) *selection* of a set of high-quality nodes from groups of interests, with guaranteed group coverage that are configurable by users, and (2) *losslessly summarize* their neighbors, with small "reconstruction" effort. The problem has a general form below.

- **Input:** A graph G, a set of groups $\mathcal{V}$ of the same type of nodes in G, where each group $P_i \in \mathcal{V}$ is associated with a range $[l_i, u_i]$ (a pair of integers where $l_i \leq u_i \leq |P_i|$), denoting required coverage;
- **Output:** a summary structure $\mathcal{S}$ with graph patterns that (1) selects ("cover") n_i nodes from each group $V_i \in \mathcal{V}$ via graph pattern matching, such that n_i is in $[l_i, u_i]$, (2) the covered nodes maximize a monotone submodular utility function F, and (3) provides auxiliary structure that can losslessly reconstruct the neighbors of the selected nodes.

Example 4: A better summary structure may present pattern P_4, which "integrates" high centrality and a circle social structure, with informative selection criteria on "work experience" and "industry". This pattern leads to a proper selection of 53% males and 47% females, identified by the pattern node u_{12}. □

Although desirable, *how to characterize and efficiently compute such summaries for configurable utility function and coverage requirements over groups?*

Contributions. This paper investigates group summarization with graph patterns with fairness constraints. We characterize the group fairness as a set of coverage constraints defined on individual groups. We introduce feasible algorithms to compute and maintain summaries with guarantees on user-defined quality and coverage constraints.

(1) We introduce r-*summaries*, a class of "pattern-correction" structures to summarize node groups in graphs (Section II). An r-summary has a set of graph patterns with a designated node type, and a set of edge corrections to guide the reconstruction of neighborhood nodes and edges up to r-hop, for each node that is "covered" by the summary structure.

(2) We introduce quality measures for an r-summary, in terms of conciseness, coverage properties and utility of the nodes (Section III). Based on these quality measures, we formalize the problem of *graph summarization with group fairness* (denoted as FGS) as a min-max optimization problem. Given a group $\mathcal{V}$, our goal is to compute an r-summary with k patterns that selects n nodes in $\mathcal{V}$ that satisfies the coverage constraints, minimizes correction cost, and maximizes the utility.

We establish the hardness result for FGS. We show that it is already NP-complete to verify if a summary structure is an r-summary for a group $\mathcal{V}$, and provide procedures for the verification problem. We further show that FGS is in general Σ_2^P-complete, by establishing a connection to graph reconstruction problem, a known Σ_2^P-complete problem. Here Σ_2^P refers to the class of problems solvable in NP with an oracle for an NP-complete problem.

(3) We introduce an approximation scheme for FGS (Section IV). We represent the min-max form of FGS into a bi-level optimization problem, and use a "select-and-summarize" strategy to compute r-summaries with small accumulated cost at node level, all subject to coverage constraints. We show that this ensures a *relative optimality guarantees* in the form of $(\frac{1}{2}, \ln(n))$-approximation, which computes r-summaries that can (a) approximate optimal node set with $\frac{1}{2}$ ratio, and (b) simultaneously achieves $ln(n)$-approximation of optimal correction cost for the *fixed* node selection. This is a "weaker" form of global approximation guarantee, yet produces desirable summary structures given the guaranteed node quality, coverage requirement, and small accumulated cost that bounds the actual reconstruction cost.

Specifying the approximation scheme, we introduce (1) approximations for FGS with bounded number of patterns (Section V), and (2) an efficient online algorithm that interleaves node selection and summary generation, with a matching guarantee of $(\frac{1}{4}, ln(n) - \frac{1}{k})$, when $\mathcal{V}$ is large (Section VI). These results provide flexible summarization strategies.

(4) We further develop an incremental algorithm to maintain r-summaries upon the arrival of new edges to the groups (Section VII). We incrementalize the computation of the node selection and summarization. Instead of rediscovering new patterns from scratch, we perform an efficient swapping strategy to control the number of r-summaries for conciseness.

(5) Using real-life graphs, we verify the effectiveness and efficiency of our algorithms (Section VIII). Our algorithms can generate summaries with both desired quality and a small amount of edge corrections in covering designated groups. These algorithms are also feasible. For example, it takes up to 700 seconds to generate summaries in real-life graphs with 5 million nodes and 45 million edges. Our case analysis also verifies their applications in supporting talent search and query processing under fair constraints.

Related Work. We categorize the related work as follows.

Graph summarization. Graph summarization has been studied with various optimization goals (see [25] for a survey). Most approaches follow minimum description length (MDL) principle to discover (pre-defined) structural patterns that lead to high compression rate of large graphs [25], [22], [8], [29], [41], leveraging frequent subgraph pattern mining [10] . For example, frequent stars, bipartite graphs, cliques or chains are used as vocabularies to encode succinct descriptions of large social or knowledge graphs [22], or for visual analysis [8]. Sparse patterns are detected to understand and sample community structures [29]. d-summaries [41] construct computationally efficient patterns to approximately describe neighborhood information, which uses an efficient, lossy graph pattern matching process to avoid expensive subgraph isomorphism tests. To avoid information loss, Lossless graph summarization [35], [39], [20] incorporates correction structuresand extends MDL to minimize both summary sizes and the size of (edge) corrections. Unlike conventional graph summarization, our problem aims to compute summaries that are not only concise, lossless, but also ensure group coverage constraints. This is not addressed by prior approaches.

Subset Selection. Subset selection with fairness constraints has been studied [42], [34], [32]. Given a universal set and a set of groups (subsets), it computes a diverse subset that can cover each group with individual cardinality constraints. Approximation algorithms have been studied to generate subsets for max-sum and max-min diversification [34]. Submodular maximization under fairness constraints has been studied for data streams [9], where approximations with constant factors are presented. These methods study set coverage properties and cannot be directly used for graph summarization with fairness constraints. Our formal analysis verifies the hardness of graph summarization with group fairness, and shows the latter is more involved as a general counterpart of these problems. We introduce both feasible approximations and fast heuristics for fair graph summaries.

Diversified Pattern Mining. Diversified subgraph pattern discovery [33] aims to discover subgraph patterns that maximize the coverage of a node set and the pairwise diversity of individual nodes that are covered. A greedy approximation is introduced given the submodular quality measure. The problem is relevant to a special case of our problem when group coverage is consistently defined on a single set. On the other hand, it has been observed that group fairness and diversity may come with conflict [42], [9]. We study a more involved setting and introduce feasible algorithms that compute the summaries under monotone submodular quality measures and explicit group coverage constraints.

II. Graph Patterns and Summaries

Graphs. We consider directed, attributed graphs $G = (V, E, L, T)$, where V is a node set, and $E \subseteq V \times V$ is a set of edges. Each node $v \in V$ (resp. edge $e \in E$) has a label $L(v)$ (resp. $L(e)$). Each node v carries a tuple $T(v) = <(A_1, a_1),\ldots,(A_n, a_n)>$, where A_i $(i \in [1, n])$ from a finite set $\mathcal{A}$ is a node attribute with value a_i.

We use the following notations. The r-hop neighbors (resp. edges) of v, denoted as N_v^r (resp. E_v^r), refers to the nodes (resp. edges) that can be reached from or reach v in r hops. The r-hop neighbors of a node set X, denoted as N_X^r, refers to the set $\bigcup_{v\in X} N_v^r$. The r-hop edge set E_X^r is defined similarly.

Graph patterns. A graph pattern $P(u_o)$ is a connected graph (V_P, E_P, L_P, T_P), where V_P (resp. $E_P \subseteq V_P \times V_P$) is a set of pattern nodes (resp. pattern edges). Each node $u \in V_P$ (resp. edge $e \in E_P$) has a label $L_P(u)$ (resp. $L_P(e)$). Each pattern node u has a set of equality literals $T_P(u)$ in the form of $u.A = a$ $(A \in \mathcal{A})$, where a is a constant.

The node u_o is a designated *focus* of P. In practice, a pattern with a focus captures a "center" of interests and its egocentric structures, as seen in *e.g.,* social network analysis [3], [24].

Coverage. We extend graph pattern matching with *induced subgraph isomorphism* to characterize the coverage of a pattern. Given a pattern P and a graph G, a matching from P to G is a function $h : V_P \to V$, where (a) for each node $u \in V_P$, $L_P(u) = L(h(u))$, and for each literal $u.A = a$ in T_P, $h(u).A = a$; and (b) for each edge $e = (u, u')$ in P, $h(e) = (h(u), h(u'))$ is an edge in G where $L_P(e) = L(h(e))$.

A graph pattern $P(u_o)$ *covers* a node v (resp. edges e) if there exists a matching h such that $v = h(u)$ (resp. $e = h(e_p)$). The set of all the nodes (resp. edges) covered by $P(u_o)$ at the *focus* is denoted as P_V (resp. P_E). Given a set of graph patterns $\mathcal{P}(u_o) = \{P_1(u_o), \ldots P_n(u_o)\}$ with a common focus u_o, the nodes (resp. edges) covered by $\mathcal{P}(u_o)$ in G at u_o, denoted as $\mathcal{P}_V$ (resp. $\mathcal{P}_E$), refers to the set $\bigcup_{P\in\mathcal{P}} P_V$ (resp. $\bigcup_{P\in\mathcal{P}} P_E$), *i.e.,* the union of nodes (resp. edges) covered by the graph patterns in $\mathcal{P}(u_o)$ at u_o.

Groups. A group set $\mathcal{V} = \{V_1, \ldots, V_n\}$ is a set of disjoint node sets in G with a same type, where each group $V_i \in \mathcal{V}$ is a subset of V, and carries a *coverage constraint* $[l_i, u_i]$, where $0 \le l_i \le u_i \le |V_i|$. In practice, users may specify the group set $\mathcal{V}$ as vulnerable social groups (*e.g.,* gender, age, or race groups)for *e.g.,* social search and healthcare [15], [47]; and defines coverage constraints $[l_i, u_i]$ to express fairness constraints such as equal opportunity [15] or disparity constraints [12].

To simplify the presentation, we use the following conventions. (1) We assume a fixed designated focus u_o, and denote graph pattern $P(u_o)$ and graph pattern set $\mathcal{P}(u_o)$ as P and $\mathcal{P}$, and simply refer to them as *patterns* and *pattern set*. (2)

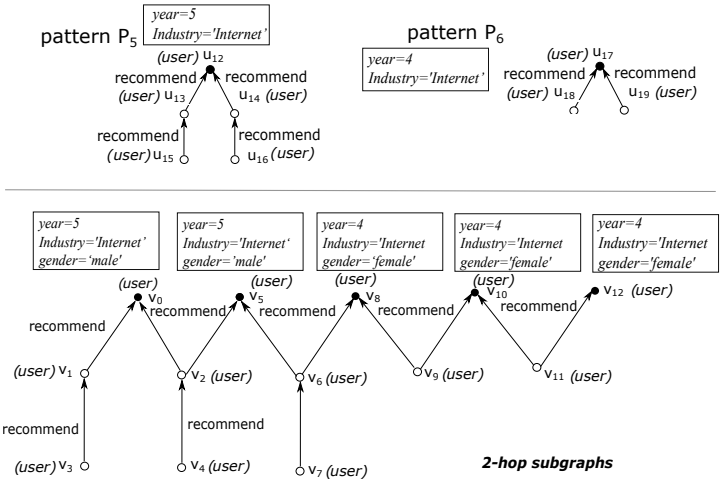


Fig. 2. Graph patterns, groups and r-summaries

Given a set of sets $\mathcal{X}$, we denote the set $\bigcup_{X\in\mathcal{X}} X$ as $\bigcup\mathcal{X}$. (3) We use P_X (resp. $\mathcal{P}_X$) to denote the nodes or edges in a node or edge set X that are also covered by pattern P (resp. pattern set $\mathcal{P}$). The symbol E^r_X refers to the r-hop edges of a node set X.

Based on the patterns, we next introduce *r-summaries*, a class of summary structures for group summarization. .

r-Summaries. Given a graph G with node set V, and a group set $\mathcal{V}$ of n sets duin G, an *r-summary* of $\mathcal{V}$ is a two-part "pattern-correction" structure $\mathcal{S} = (\mathcal{P}, \mathcal{C})$, where

- $\mathcal{P}$ is a pattern set with a common focus u_o, such that $|P_{\mathcal{V}} \cap V_i| \in [l_i, h_i]$; here $\mathcal{P}_{\mathcal{V}} = \mathcal{P}_V \cap \bigcup\mathcal{V}$, *i.e.,* the *group nodes* covered by $\mathcal{P}$; and
- $\mathcal{C}$ refers to a set of *edge corrections*, and is defined as $\mathcal{C} = E^r_{\mathcal{P}_{\mathcal{V}}} \setminus \mathcal{P}_E$, *i.e.,* the edges in r-hop neighbors of $\mathcal{P}_{\mathcal{V}}$ that are not covered by $\mathcal{P}_E$.

An r-summary $(\mathcal{P}, \mathcal{C})$ of group set $\mathcal{V}$ in G ensures to (1) *select* a set of group nodes $\mathcal{P}_{\mathcal{V}}$ from group set $\mathcal{V}$ as the matches of the focus of the patterns $\mathcal{P}$, which satisfies the coverage constraints enforced by each group, and (2) *losslessly summarizes* r-hop neighbors of the selected group nodes $\mathcal{P}_{\mathcal{V}}$, by reconstructing $E^r_{\mathcal{P}_{\mathcal{V}}}$ with $\mathcal{P}_E \cup \mathcal{C}$.

Example 5: Consider a gender group set $\mathcal{V}$, which contains a male group $\{v_0, v_5\}$ with a coverage constraint $[1, 2]$ and female group $\{v_8, v_{10}, v_{12}\}$ with coverage constraint $[2, 3]$. Fig. 2 illustrates a fraction of a profession network G, induced by the 2-hop neighbors of the group nodes $\{v_0, v_5, v_8, v_{10}, v_{12}\}$. A 2-summary $(\mathcal{P}, \mathcal{C})$ for $\mathcal{V}$ is illustrated in Fig. 2, where $\mathcal{P}$ contains two patterns P_5 and P_6. (1) P_5 selects candidates in "Internet" industry with more than 5 years of experience and are recommended by other two users, each further recommended by another user. It only covers the male group $\{v_0, v_5\}$, and misses 5 edges of their 2-hop neighbors. (2) P_6 selects candidates with four years' experience in "Internet" and are recommended by two users. It covers male group $\{v_0, v_5\}$, and two females $\{v_8, v_{10}\}$, missing in total 4 edges of the 2-hop neighbors of the nodes they covered. (3) Putting these together, $\mathcal{P}$ covers all the group nodes except v_{12} ($\mathcal{P}_{\mathcal{V}} = \{v_0, v_5, v_8, v_{10}\}$) and satisfies the coverage constraints, and losslessly describe their 2-hop neighbors with a single edge correction $\mathcal{C} = \{(v_{11}, v_{12})\}$. □

TABLE I
MAJOR NOTATIONS AND SYMBOLS.

Symbol	Description
G, $P(u_o)$ (or P)	graph, (graph) pattern
N^r_V (resp. E^r_V)	r-hop neighbors (resp. edges) of nodes V
$\mathcal{P}(u_o)$ (or $\mathcal{P}$)	a set of patterns with a common focus u_o
$\mathcal{V}$, V_i	a set of groups, and a group in $\mathcal{V}$
$[l, u]$	coverage constraint with lower/upper bounds l/u
P_V, P_E	nodes in V, edges in E that are covered by P
$\mathcal{P}_{\mathcal{V}}$, $\mathcal{P}_E$	group nodes and edges covered by $\mathcal{P}$
$\mathcal{S} = (\mathcal{P}, \mathcal{C})$	an r-summary, with edge corrections $\mathcal{C}$
C	a configuration r, k, n
F	monotone submodular utility function
$\mathcal{C}_l$ (resp. $\mathcal{C}_P$)	edge correction loss of $\mathcal{S}$ (resp. single pattern P)

We summarize the major notations in Table I.

III. GROUP SUMMARIES WITH COVERAGE CONSTRAINT

A. Quality Measurement

Given a set of node groups $\mathcal{V}$ in G, we are interested in finding r-summaries that can select high-quality group nodes from $\mathcal{V}$, and meanwhile accurately describe their neighbors with small correction. These can be characterized by the following three quality measures.

Monotone Submodular Utility. An r-summary $\mathcal{S} = (\mathcal{P}, \mathcal{C})$ should be able to identify a set of high quality nodes $\mathcal{P}(u_o, G)$ that maximizes a utility. This is often determined by a user-specified function F, which typically capture submodular properties such as informativeness [30], diversity [37], or social influence [18]. A utility function F is submodular, if for any two node sets $V_1 \subseteq V_2 \subseteq V$, and any node $v \in V \setminus V_2$, $F(V_1 \cup \{v\})$ - $F(V_1) \geq F(V_2 \cup \{v\})$ - $F(V_2)$.

Conciseness. One also wants to inspect a small number of representative nodes (*e.g.,* candidates in talent search) from a large group $\mathcal{V}$. While the summary structures are often small, it is desirable to find r-summaries that can cover a bounded number of nodes over all the groups $\mathcal{V}$. Moreover, one often wants to inspect a bounded number of patterns.

Edge coverage loss. It is also desirable to ensure a small reconstruction cost to restore the r-hop neighbors of $\mathcal{P}_{\mathcal{V}}$, the group nodes covered by $\mathcal{S}$. This can be determined by the accumulated number of the r-hop edges surrounding $\mathcal{P}_{\mathcal{V}}$ that each pattern P in $\mathcal{P}$ "missess". Let $\mathcal{C}_P = E^r_{P_{\mathcal{V}}} \setminus P_E$, where $P_{\mathcal{V}} \subseteq \bigcup\mathcal{V}$ refers to the group nodes P covers. We define an *accumulated edge coverage loss* as $\mathcal{C}_l = \sum_{P\in\mathcal{P}} |\mathcal{C}_P|$. The smaller $\mathcal{C}_l$ is, the better. Note that $|\mathcal{C}| \leq \mathcal{C}_l$, and smaller $\mathcal{C}_l$ indicates less uncovered edges by $\mathcal{S}$.

Remarks. Another option is to simply define the cost as $|\mathcal{C}|$. We consider accumulated loss as a reasonable upperbound for $|\mathcal{C}|$, and closer to the actual algorithmic reconstruction cost, given the need of pattern-wise inspection in practice.

Problem statement. We now formalize our problem as a min-max optimization problem. Given a graph G, a set of disjoint groups $\mathcal{V}$ with associated coverage constraints, a monotone submodular utility function F, and a user-specified *configuration* $C = \{r, k, n\}$, the *fair group summarization* problem, denoted as FGS, is to compute an r-summary $\mathcal{S} = (\mathcal{P}, \mathcal{C})$ of the group $\mathcal{V}$ with the following general form:

$$(\mathcal{P}, \mathcal{C}) = \min_{|\mathcal{P}| \leq k, \mathcal{C}_l} \max_{|\mathcal{P}_\mathcal{V}| \leq n} F(\mathcal{P}_\mathcal{V})$$

The solution of FGS leads to desirable summary structures $\mathcal{S} = (\mathcal{P}, \mathcal{C})$ as justified by the following properties (also see "Case study" in Section VIII). (1) The group nodes $\mathcal{P}_\mathcal{V}$ covered by $\mathcal{P}$ can be readily suggested as high-quality answers for *e.g.,* talent search and recommendation with fairness constraints [15]. (2) The patterns $\mathcal{P}$ can be directly suggested as meaningful graph queries, to guide query and graph generation with cardinality constraints [5], for *e.g.,* benchmarking. (3) The "pattern-correction" structure $\mathcal{S}$ is *queryable*, where $\mathcal{P}$ naturally serve as (virtual) views to support *e.g.,* view-based query processing [11] with small reconstruction effort. (4) The edge corrections $\mathcal{C}$ also facilitate the interpretation between selected and unselected nodes, by explicitly suggesting their difference via edge corrections.

Example 6: Continuing the example in Fig. 2. Assuming the utility function F quantifies the social influence as the the number of neighbors of the nodes covered by r-summary. Given $r = 2$, $n = 4$ and equal cardinality constraints which is $[2, 2]$ for both male and female groups. The 2-summary $\mathcal{S}$ in Fig. 2 thus covers $\{v_0, v_8, v_5$ and $v_{10}\}$ with P_5, P_6 that achieves a total influence 8. While $|\mathcal{C}| = 1$ with one missing edge (v_{11}, v_{12}), one can verify that $\mathcal{C}_{P_5} = 0$, $\mathcal{C}_{P_6} = 4$, and it takes in total $\mathcal{C}_l = 4$ to reconstruct the 2-hop neighbor of all the covered group nodes. For node v_{12}, the missing edge (v_{11}, v_{12}) can be used to interpret the reason that she is not selected, and be recommended as her new social link. □

B. Verification and Hardness

To understand the hardness of FGS, we first study a *verification problem*. Given G, $\mathcal{V}$, a configuration $C = r, n$, and constants b_c and b_f, it is to determine if a summary structure $\mathcal{S} = (\mathcal{P}, \mathcal{C})$ (1) is *feasible*, *i.e.,* an r-summary of $\mathcal{V}$ that covers at most n nodes $|\mathcal{P}_\mathcal{V}| \leq n$ and also satisfies the coverage constraints for every group, and (2) the covered group nodes at u_o have utility at least b_f, and edge coverage loss $|\mathcal{C}_l| \leq b_c$.

Lemma 1: *The verification problem alone is* NP*-complete.* □

The hardness follows from the reduction from the subgraph isomorphism problem between a single pattern and a graph. Below we outline a procedure to show it's in NP.

Verification. The procedure, denoted as rverify, first checks if $|\mathcal{P}| \leq k$, and performs subgraph isomorphism tests to decide if $\mathcal{P}(u_o, G) \cap \bigcup \mathcal{V} = \emptyset$ (in NP) for at most k patterns. It then verifies if $|\mathcal{P}(u_o, G) \cap \bigcup \mathcal{V}| \leq n$, and $|\mathcal{P}(u_o, G) \cap \mathcal{V}_i| \leq [l_i, h_i]$ for each group $V_i \in \mathcal{V}$. It finally verifies if $\mathcal{C}_l \leq b_c$, and the utility is at least b_f. The above verification process takes $O(k \cdot |\bigcup \mathcal{V}| \cdot T_I + |\bigcup \mathcal{V}|)$ time. Here T_I is the cost of verifying if a single pattern $P \in \mathcal{P}$ covers a group node at u_o, which is typically small in practice. Note that the verification does not require to compute the complete set $\mathcal{P}(u_o, G)$.

We next investigate the hardness of FGS.

Theorem 2: *The* FGS *problem is* Σ_2^p*-complete.* □

Proof sketch: Given G, $\mathcal{V}$, a configuration $\mathcal{C}$=(r, n) and two constants b_c and b_f, the decision problem of FGS is to decide if there exists a feasible r-summaries $\mathcal{S}$ of $\mathcal{V}$ with a utility no less than b_f and edge correction size no more than b_c. The problem can be solved in Σ_2^p. As the verification can be done in NP (Lemma 1), FGS can be solved in Σ_2^p by guessing an r-summary $\mathcal{S}$ and verify its properties with rverify.

To show it's Σ_2^p-complete, we describe a reduction from the Graph Reconstruction (GR) problem [21]. Given two sets $\mathcal{G}^+$ and $\mathcal{G}^-$ of graphs, GR determines whether there exists a graph G_o such that each $G^+ \in \mathcal{G}^+$ is isomorphic to a subgraph of G_o, and each $G^- \in \mathcal{G}^-$ is not isomorphic to any subgraph of G_o. Our reduction constructs G as the union of augmented $\mathcal{G}^+$ and $\mathcal{G}^-$, where each single graph G_i^+ in $\mathcal{G}^+$ (resp. $G_j^- \in \mathcal{G}^-$) is added an augmented edge connecting to a distinct node v_i^+ (resp. v_j^-) with unique label 'positive' (resp. 'negative'). We set $\mathcal{V} = \{V^+, V^-\}$, where $\mathcal{V}^+$ (resp. V^-) contains $|\mathcal{G}^+|$ 'positive' nodes (resp. $|\mathcal{G}^-|$ 'negative' nodes), associated with constraints $[|\mathcal{G}^+|, |\mathcal{G}^+|]$ (resp. $[0, 0]$). Setting a configuration C = $(r_m + 1, |\mathcal{G}^+|, |\mathcal{G}^+|)$, with r_m the largest diameter of graphs in $\mathcal{G}^+$, we show there exists a solution for GR if and only if there is an r-summary for the FGS instance. □

IV. Computing Summaries with Group Fairness

We next introduce practical algorithms to compute r-summaries with coverage and utility guarantees.

A. Approximating Summaries

Given a configuration $\{r, k, n\}$, one wants to compute an optimal r-summary $\mathcal{S}$ with maximized $F(\mathcal{P}_\mathcal{V})$ and smallest edge coverage loss $\mathcal{C}_l$, where $\mathcal{P}_\mathcal{V}$ is the set of group nodes covered by $\mathcal{P}$. A naive approach enumerates and verify all size-k pattern sets, and invokes the verification process to check if each set contributes to an r-summary, and if so, chooses the one with $\mathcal{P}_\mathcal{V}$ that lead to the highest utility. This is, nevertheless, not practical for large G. We thus consider faster algorithms with performance guarantees.

We first represent the min-max problem FGS as the following bi-level optimization problem, in a "weaker" form:

$$\min_{V_p^* \subseteq \mathcal{P}_\mathcal{V}} \mathcal{C}_l(\mathcal{P}, V_p^*), \text{where} \quad (1)$$

$$V_p^* = \arg\max_{|V_p| \leq n; |V_p \cap V_i| \in [l_i, h_i]} F(V_p) \quad (2)$$

where the "lower-level" goal aims to select n group nodes V_p^* that maximizes utility $F(V_p^*)$, and meanwhile satisfies the coverage constraints on $\mathcal{V}$; and an "upper-level" optimization

Algorithm APXFGS

Input: graph G, groups $\mathcal{V}$ with associated coverage constraints, utility function F, *configuration* $\mathcal{C} = \{r, k, n\}$.
Output: a feasible r-summary $\mathcal{S}$ of $\mathcal{V}$.

1. set $V_p := \emptyset$; set $\mathcal{P} := \emptyset$; set $\mathcal{P}_E := \emptyset$; set $E_r := \emptyset$; set $\mathcal{P}_c := \emptyset$; set $\mathcal{P}_u := \emptyset$;
2. $V_p :=$ FairSelect$(\mathcal{V}, F, n)$;
3. **for each** $v \in V_p$ **do**
4. $\quad E_r := E_r \cup E_r(v)$;
5. $\mathcal{P}_c :=$ SumGen(V_p, E_r, r);
6. **while** $V_p \neq \emptyset$ **do**
7. $\quad \mathcal{P}_u := \emptyset$;
8. $\quad$ **for each** $P \in \mathcal{P}_c \setminus \mathcal{P}$ **do**
9. $\quad\quad$ **if** Extendable$(P, \mathcal{P}, \mathcal{V}, n)$ **then**
10. $\quad\quad\quad \mathcal{P}_u := \mathcal{P}_u \cup P$;
11. $\quad P^* := \arg\max_{P_u \in \mathcal{P}_u} \frac{|P_u(u_o, G) \cap V_p|}{\mathcal{C}_{P_u}}$;
12. $\quad \mathcal{P} := \mathcal{P} \cup \{P^*\}$; $V_p := V_p \setminus P^*(u_o, G)$;
13. $\mathcal{S} := (\mathcal{P}, E^r_{V_p} \setminus \mathcal{P}_E)$;
14. **return** $\mathcal{S}$;

Procedure FairSelect $(\mathcal{V}, F, n)$

1. set $V_p := \emptyset$;
2. **while** $|V_p| < n$ **do**
3. $\quad$ set $V_u := \emptyset$
4. $\quad$ **for each** $v \in \mathcal{V} \setminus V_p$ **do**
5. $\quad\quad$ **if** ExtendableM$(v, V_p, \mathcal{V}, n)$
6. $\quad\quad\quad V_u := V_u \cup \{v\}$;
7. $\quad v* := \arg\max_{v' \in V_u} (F(V_p \cup v') - F(V_p))$;
8. $\quad V_p := V_p \cup \{v*\}$;
9. **return** $\mathcal{V}_p$;

Fig. 3. Algorithm APXFGS

is to discover a pattern set $\mathcal{P}^*$ that minimizes $|\mathcal{C}_l(\mathcal{P}^*, V_p^*)|$, subject to cover a *fixed*, desirable set of group nodes V_p^*.

We then resort to compute an r-summary structure $\mathcal{S} = (\mathcal{P}, \mathcal{C})$ of $\mathcal{V}$, that ensures the following: (1) $\mathcal{P}$ covers a set of n nodes V_p ($V_p \subseteq \mathcal{P}_\mathcal{V}$), where V_p satisfies the coverage constraints of $\mathcal{V}$, (2) $F(V_p) \geq \alpha * F(V_p^*)$, and (3) $\mathcal{C}_l(\mathcal{P}, V_p) \leq \beta \mathcal{C}_l(\mathcal{P}^*, V_p)$, for a fixed selected set V_p. We advocate such a solution $\mathcal{P}$ as an (α, β)-*approximation* for FGS. This is a weaker approximation guarantee, as a sub-optimal solution that approximates $\mathcal{P}^*$ subject to V_p. Nevertheless, $\mathcal{S}$ remains to be a desirable solution, treating V_p as a "yardstick" solution that already has a constant approximation ratio to an optimal solution V_p^* of the lower-level optimization, which ensures high utility, guaranteed group coverage constraints, and a relative bound for $\mathcal{C}_l$ (hence a bounded $|\mathcal{C}|$, as $|\mathcal{C}| \leq \mathcal{C}_l$).

Below we present our main result.

Theorem 3: *Given a configuration $C = \{r, n\}$ without cardinality constraint k, there is a $(\frac{1}{2}, ln(n))$-approximation for* FGS. *The algorithm takes $O(n \cdot N \cdot T_I \cdot |\bigcup \mathcal{V}| + n \cdot N^2 + |E|)$ time, where N is the total number of verified patterns.* □

We present a constructive proof for Theorem 3. Our idea is to take a "select-and-summarize" strategy. (1) The selection phase solves the lower-level problem and computes a set of nodes V_p with coverage and quality guarantee. (2) The summarization phase then explores patterns induced from the r-hop neighbors of V_p to ensure the coverage of V_p and its r-hop neighbors with small reconstruction cost, by minimizing accumulated pattern-wise correction $\mathcal{C}_l$.

Procedure Extendable $(P, \mathcal{P}, \mathcal{V}, n)$

1. **if** $P(u_o, G) \cap \bigcup \mathcal{V} = \emptyset$ **then return** false;
2. set $\mathcal{P}_e := \mathcal{P} \cup \{P\}$; integer $cov := 0$;
3. **for each** $V_i \in \mathcal{V}$ **do**
4. $\quad$ **if** $|\mathcal{P}_e(u_o, G) \cap V_i| > h_i$ **then**
5. $\quad\quad$ **return** false;
6. $\quad cov := cov + \max(|\mathcal{P}_e(u_o, G) \cap V_i|, l_i)$;
7. $\quad$ **if** $cov > n$ **return** false;
8. **return** true;

Fig. 4. Procedure Extendable

We next present an algorithm that implements the idea.

Algorithm. The algorithm, denoted as APXFGS (Fig. 3) performs the following.
(1) *Selection phase* (lines 1-4). APXFGS invokes a procedure FairSelect to compute a set of group nodes V_p with high utility $F(V_p)$ and satisfy the coverage constraint (line 2; see Procedure FairSelect). It then initializes an edge set $E_r(V_p)$ to be covered by the patterns.
(2) *Summarization phase* (lines 5-13). It invokes procedure SumGen to perform a constrained graph pattern mining over V_p and their r-hop edges $E_r(V_p)$. The process exploits established graph pattern mining, yet early terminates at patterns with radius up to r from u_o (*i.e.,* those with distance up to r between u_o and any other pattern nodes) (line 5). APXFGS then follows a greedy strategy to dynamically choose a pattern $P*$ that maximize a gain determined by covered nodes $P*_{V_p}$ in V_p (computed as $P^*(u_o, G) \cap V_p$) and uncovered edge counterpart $\mathcal{C}_P$ (lines 6-12). This process is guarded by an "extendable" condition that verifies the coverage constraints (line 9). This ensures the construction of a partial r-summary only. The desired r-summary $\mathcal{S}$ is then constructed as $(\mathcal{P}, E_r \setminus \mathcal{P}_E)$ and returned (line 14).

Procedure Extendable. Given an r-summary $\mathcal{S} = (\mathcal{P}, \mathcal{C})$ of $\mathcal{V}$ and a pattern P, we say $\mathcal{S}$ is *extendable* with a pattern P if $\mathcal{S} = (\mathcal{P} \cup \{P\}, \mathcal{C})$ remains to be feasible. Procedure Extendable determines if a current "partial" r-summary $\mathcal{S}$ is extendable with P, by checking (1) if it violates coverage requirement in terms of upper bound; (2) covers no new nodes (line 3), and (3) covers more than n nodes (line 6).

Procedure FairSelect. Given graph G, groups $\mathcal{V}$, utility function F and integer n, FairSelect selects a set of nodes $V_\mathcal{S} \subseteq \bigcup \mathcal{V}$ such that $V_\mathcal{S}$ maximize F and covers each group $\mathcal{V}_i \in \mathcal{V}$ with desired number of nodes in $[l_i, h_i]$. To this end, it solves a *submodular maximization* problem with group cardinality constraints following [16], which performs an iterative greedy selection strategy over group nodes. (1) In each iteration, FairSelect initializes a candidate set V_u with all the nodes in $\mathcal{V} \setminus V_p$ that can be used to "extend" V_p. This is determined by a procedure ExtendableM (line 5; details omitted) by checking if: (1) for any group V_i in $\mathcal{V}$, $|(V_p \cup v) \cap V_i| < h_i$; (2) $\sum_{V_i \in \mathcal{V}} max(|(V_p \cup v) \cap V_i|, l_i) <= n$, similarly as in Extendable. (2) It then adds a node with

maximal marginal gain of submodular function F (lines 7-8) to the node set V_p, until up to n nodes are selected.

Example 7: Continuing with the example in Fig. 2, we consider a configuration of $r = 2$, $n = 4$, and a same cardinality constraint $[2, 2]$ for both male and female groups.

The selection phase performs a greedy selection of the group nodes. APXFGS identifies a set of promising nodes V_p = $\{v_0, v_5, v_8, v_{10}\}$, which satisfies the coverage requirement of the groups. In the summarization phase, APXFGS firstly select pattern P_5 due to that it introduces a minimal size of edge correction cost 0. As $\mathcal{P} = \{P_5\}$ remains extendable with $P_6 \in \mathcal{P}_c$, APXFGS next verifies P_6, and add it to $\mathcal{P}$.

Consider another pattern $P_7 \in \mathcal{P}_c$ obtained from $E^r_{V_p}$ (not shown). Despite that P_7 needs to perform the same amount of edge corrections as P_6 ($\mathcal{C}_{P_6} = \mathcal{C}_{P_7} = 4$), APXFGS favors P_6 which better covers the group node set $V_P = \{v_0, v_5, v_8, v_{10}\}$. As V_p has been covered by $\{P_5, P_6\}$, APXFGS terminates and returns $\mathcal{S}$ with $\mathcal{P} = \{P_5, P_6\}$, and $\mathcal{C} = \{(v_{11}, v_{12})\}$. □

B. Correctness and Approximability

To see the correctness and quality guarantees of FairSelect, we show that it has the following invariants.

(1) Procedure FairSelect computes a set of nodes V_p such that $F(V_p) \geq \frac{1}{2}F(V_p^*)$, for all subsets of $\bigcup \mathcal{V}$ with size bounded by n that also satisfy the group coverage constraints. This can be verified by an approximation preserving reduction from the lower-level node selection problem to fair submodular maximization [16]. The reduction constructs a base set as $\bigcup \mathcal{V}$ with groups and associated ranges remain intact. It has been verified that a greedy selection process ensures a $\frac{1}{2}$-approximation which is simulated by FairSelect.

(2) Algorithm APXFGS computes a set of summaries that ensures to cover V_P with small accumulated edge cover loss $\mathcal{C}_l$, by solving the upper-level problem as a maximum coverage problem [44]. As each pattern P uniquely determines a set of covered nodes $P(u_o, G)$ and an individual edge cover loss, a reduction treats each P as a subset $P(u_o, G) \cap \bigcup V_p$ with a weight $\mathcal{C}_P$ (recall $\mathcal{C}_l$=$\sum_{P\in\mathcal{P}} \mathcal{C}_P$). It then follows a greedy strategy [44] to select $\mathcal{P}$ with $\mathcal{C}_l \leq ln(|V_p|)\mathcal{C}_l^* \leq ln(n)\mathcal{C}_l^*$. Note that this indicates a provable upper bound for the size of edge correction as well.

Lemma 4: APXFGS *returns an r-summary $\mathcal{S}$ with a size-bounded edge correction $|\mathcal{C}| \leq ln(n)\mathcal{C}_l^*$, where $\mathcal{C}_l^*$ is the optimal edge coverage loss.* □

(3) The two procedures ExtendableM and Extendable correctly implements the verification rverify of r-summaries (Section III-B) into selection and summarization phases. This guarantees the invariant that only feasible r-summaries of $\mathcal{V}$ are correctly returned.

Time cost. We next analyze the time cost. Procedure FairSelect takes $O(n \cdot |\bigcup \mathcal{V}|)$ time to select V_p. Procedure SumGen takes at most $N \cdot T_I |\bigcup \mathcal{V}|$ time to generate and verify the patterns and their covered group nodes, where N is the total number of patterns with radius up to r from u_o, and T_I is the time cost of verifying if a single node is covered by P at u_o. Algorithm APXFGS then takes $O(n \cdot N^2 + |E|)$ to compute $\mathcal{P}$ and $\mathcal{C}$. The total time cost of APXFGS is thus in $O(n \cdot N \cdot T_I \cdot |\bigcup \mathcal{V}| + n \cdot N^2 + |E|)$ time.

The above analysis verifies that APXFGS ensures (1) a solution V_P that approximates a $\frac{1}{2}$ approximation ratio to the optimal solution V_P^*, and (2) an r-summary $\mathcal{S}$ with $\mathcal{C}_l$ that approximates a local optimal solution $\mathcal{C}_l^*$ given V_p, at a ratio $ln(n)$. It thus achieves a $(\frac{1}{2}, \ln(n))$-approximation for FGS. Theorem 3 thus follows.

V. Computing Group Summaries with k Patterns

The approximation scheme APXFGS considers a configuration C = (r, n) without constraints on the number of patterns, and may return an excessive number of patterns. We next consider a variant of FGS, which requires to compute an r-summary with at most k patterns, and minimizes $|\mathcal{C}|$ instead of accumulated correction cost.

$$\min_{|\mathcal{P}|\leq k, V_p^* \subseteq \mathcal{P}_\mathcal{V}} |\mathcal{C}|, \text{where} \tag{3}$$

$$V_p^* = \underset{|V_p|\leq n; |V_p \cap V_i| \in [l_i, h_i]}{\arg\max} F(V_p) \tag{4}$$

We show that a slight revision of algorithm APXFGS achieves the following relative approximation ratio.

Theorem 5: *Given a configuration C=(r, k, n), there exists an $(\frac{1}{2}, 1 + \frac{1}{e\cdot\gamma})$ approximation, where $\gamma = \frac{|E^r_{V_p}|}{|\mathcal{P}_E^* \cap E^r_{V_p}|} - 1$.* □

Here V_p is the approximate solution of V_p^*, which ensures $F(V_p) \geq \frac{1}{2}F(V_p^*)$. Intuitively, a larger γ inherently a better approximation ratio, yet meanwhile indicates a larger correction cost. In other words, it verifies that an optimal solution P^* can be better approximated when it inherently covers a smaller fraction of $E^r_{V_p}$. For example, when γ = 1, there is an $(\frac{1}{2}, 1 + \frac{1}{e})$ approximation, yet under the assumption that even the optimal solution can cover half of the r-hop edges.

Algorithm Outline. We next outline the variant of APXFGS. The algorithm follows the "select-and-summarize" strategy. It first compute V_p with procedure FairSelect (line 2 of APXFGS), and generate patterns with procedure SumGen to be verified (line 5 of APXFGS). The only differences are as follows. (1) It initializes a universal set $E^r_{V_p}$, and for each pattern $P \in \mathcal{P}_c$, a matching edge set $P_E \cap E^r_{V_p}$. (2) It revises the summarization phase (lines 6-13), and selects k patterns by solving a *maximum coverage problem*, which aims to compute k patterns $\mathcal{P}$ with $|\mathcal{P}| \leq k$, such that $\bigcup_{P\in\mathcal{P}} P_E \cap E^r_{V_p}$ is maximized. This equivalently leads to minimizing $|\mathcal{C}|$ for selected $\mathcal{P}$. To this end, it greedily select the pattern P that maximizes a marginal gain as the currently uncovered r-hop edges in $E^r_{V_p}$, *i.e.,* $|E^r_{V_p} \cap (\mathcal{P} \cup \{P\})_E|$, until $|P| = k$. (3) It verifies if the current $\mathcal{P}$ covers V_P and satisfies the coverage constraint, and if so, terminates and returns $\mathcal{S}$ with $\mathcal{P}$ and $\mathcal{C}$. Otherwise, it continues (2) with greedy swapping strategy, until either fails to identify a k pattern set, or early terminates with desirable $\mathcal{P}$ and an r-summary.

Algorithm Online-APXFGS

Input: graph G, groups $\mathcal{V}$ with associated coverage constraints, utility function F, *configuration* $\mathcal{C} = \{r, k, n\}$.
Output: an r-summary $\mathcal{S}$ of $\mathcal{V}$.

1. set $V_p := \emptyset$; set $\mathcal{P} := \emptyset$; set $\mathcal{P}_E := \emptyset$; set $E_r := \emptyset$; set $\mathcal{P}_u := \emptyset$; $\mathcal{S} := \emptyset$; $B_c := \emptyset$;
2. **for each** $v \in \bigcup \mathcal{V}$ **do** */* streaming selection phase */*
3. $w(v) = F(V_p \cup \{v\}) - F(V_p)$;
4. **if** $v \in V_c$ **then** $B_c := B_c \cup v$;
5. **if** ExtendableM$(v, V_p, \mathcal{V}, n)$ **then**
6. $V_p := V_p \cup \{v\}$;
7. **else** */* consult an oracle procedure */*
8. $V_p :=$ UpdateVp $(v, V_p, \mathcal{V}, F, n)$;
9. **if** $v \in V_p$ **then** */* trigger local summarization phase */*
10. $\mathcal{P} :=$ UpdateP $(v, V_p, \mathcal{V}, F, n)$;
/ post processing with bucket B_c */*
11. **while** there is a group $V_c \in \mathcal{V}$ where $|\mathcal{P}_{V_c}| < l_i$ **do**
12. PostSelect$(G, \mathcal{V}, F, \mathcal{C}, B_c, \mathcal{P})$;
13. $\mathcal{S} := (\mathcal{P}, E_r \setminus \mathcal{P}_E)$;
14. **return** $\mathcal{S}$;

Procedure UpdateVp $(v, V_p, \mathcal{V}, F, n)$

1. set $U := \emptyset$;
2. **for each** $v' \in V_p$ **do**
3. $V_p^{'} := V_p \setminus \{v'\}$;
4. **if** ExtendableM$(v, V_p^{'}, \mathcal{V}, n)$ **then** $U := U \cup \{v'\}$;
5. $v^- := \arg\min_{v' \in U}(F(V_u \cup v') - F(V_u))$;
6. $V_u := V_p \setminus \{v^-\}$;
7. **if** $w(v) \geq 2(F(V_u \cup v) - F(V_u))$ **then**
8. $V_p := V_p \setminus \{v'\} \cup \{v\}$;
9. **return** $\mathcal{V}_p$;

Fig. 5. Algorithm Online-APXFGS

Analysis. The correctness and approximation analysis follows the analysis of APXFGS and a reduction from pattern selection to maximum coverage problem with a known approximation ratio $1 - \frac{1}{e}$. Specifically, let $|\mathcal{C}^*|$ be the smallest correction size achieved by optimal solution $\mathcal{P}^*$. As $|\mathcal{C}^*| = |E^r_{V_p}| - |\mathcal{P}^*_E \cap E^r_{V_p}|$, $|\mathcal{C}| = |E^r_{V_p}| - |\mathcal{P}_E \cap E^r_{V_p}|$, and $|\mathcal{P}_E \cap E^r_{V_p}| \geq 1 - \frac{1}{e}|\mathcal{P}^*_E \cap E^r_{V_p}|$, we have $|\mathcal{C}| \leq (1 + \frac{|\mathcal{P}^*_E \cap E^r_{V_p}|}{e \cdot (|E^r_{V_p}| - |\mathcal{P}^*_E \cap E^r_{V_p}|)})|\mathcal{C}^*|$. The algorithm takes the same time cost as APXFGS.

We present the detailed analysis in [**?**].

VI. Online Group Summarization

The algorithm APXFGS requires to compute a set of nodes V_p first, and then generates and verifies patterns from $E^r_{V_p}$. This may be expensive for large $\mathcal{V}$. We next introduce an online algorithm that can process $\mathcal{V}$ as a "stream" of group nodes, without pre-computing V_p. It interleaves node selection and pattern generation to refine the summaries progressively. The algorithm access G as a static graph. We discuss the maintenance of r-summaries over dynamic graphs in Section VII.

Given $\mathcal{V}$ as a node stream, our idea is to (1) streamline the node selection procedure FairSelect with a streaming submodular maximization process [16], and (2) upon a group node v is accepted to V_p, triggers ad-hoc, *localized* pattern generation and verification at smaller, node-rrlevel (which only involves E^r_v) to only perform necessary maintenance of $\mathcal{P}$.

Procedure UpdateP $(v, V_P, \mathcal{P}, \mathcal{V}, n)$

1. $\mathcal{P}_u :=$ SumGen(v, E^r_v, r);
2. **while** $|\mathcal{P}| < k$ **do**
3. P^*:=$\arg\max_{P_u \in \mathcal{P}_u} \frac{|P_u(u_o, G) \cap V_p|}{\mathcal{C}_{P_u}}$;
4. $\mathcal{P} := \mathcal{P} \cup \{P^*\}$;
5. $\mathcal{P}_u := \mathcal{P}_u \setminus \{P^*\}$;
6. $\Delta P := \emptyset$
7. **for each** $P \in \mathcal{P}_u$ **do**
8. **for each** $P' \in \mathcal{P}$ **do**
9. $\mathcal{P}' := \mathcal{P} \setminus \{P'\} \cup \{P\}$;
10. **if** $V_p \subseteq \mathcal{P}'_{\mathcal{V}}$ **then**
11. */* ensuring covering all the nodes in V_p */*
12. $\Delta\mathcal{P}$:= $\Delta\mathcal{P} \cup \{P\}$;
13. P^+:=$\arg\max_{P' \in \Delta\mathcal{P}} \frac{|P'(u_o, G) \cap V_p|}{\mathcal{C}_P}$;
14. P^-:=$\arg\min_{P \in \mathcal{P}} \frac{|P(u_o, G) \cap V_p|}{\mathcal{C}_P}$;
15. $\mathcal{P} := \mathcal{P} \setminus \{P^-\} \cup \{P^+\}$;
16. **return** $\mathcal{P}$;

Fig. 6. Procedure UpdateP

To ensure the correctness, a post processing is performed to ensure coverage properties. Our main result is as follows.

Theorem 6: *Given a configuration C=$\{r, n, k\}$, there is a $(\frac{1}{4}, ln(n) - \frac{1}{k})$-approximation for* FGS. *The online algorithm process each group node v in $O(\log k + N_v \cdot T_I)$ time, where N_v is the number of patterns induced from E^r_v.* □

We next present the online algorithm.

Online Summarization. The online algorithm, denoted as Online-APXFGS, is illustrated in Fig. 5. It maintains, for each group $V_c \in \mathcal{V}$, a bucket B_c, to store the processed nodes in V_c. Upon receiving a group node $v \in \mathcal{V}$, Online-APXFGS performs the following two major steps.

(1) *Streaming selection* (lines 3-8). Online-APXFGS performs streaming submodular maximization selection following [16]. In particular, it first verifies if V_p is extendable (by invoking Procedure ExtendableM; line 5), and if so, either directly accept v to V_p (line 5); otherwise, consults a greedy streaming selection procedure (an "oracle" algorithm; lines 8-16) to decide whether to replace a node $v' \in V_P$ with v, following a greedy strategy, or to reject v, and put it in B_c.

(2) *Local pattern update* (lines 9-10). For each new node v that enters V_p directly or via replacement, it performs a pattern generation and verification. Unlike APXFGS which needs to verify all patterns with radius up to r from u_o induced by $E^r_{V_p}$, the process only need to verify the patterns induced by E^r_v. In particular, for each batch of new patterns P_u derived from E^r_v, and current $\mathcal{P}$, it determines two processes: (a) it iteratively selects k-$|\mathcal{P}|$ nodes from P_u that dynamically maximize the gain determined by current node coverage and correction cost C_P (line 22), to add to $\mathcal{P}$, or (b) dynamically decide a pattern set $P^+ \subseteq \mathcal{P}_u$ to be added to $\mathcal{P}$, and a pattern set $P^- \subseteq \mathcal{P}$ to be replaced out of $\mathcal{P}$, and perform the "swapping" process.

(3) *Post-processing* (lines 11-12). The above process repeats until all the nodes in $\mathcal{V}$ are processed. While ExtendableM ensures no group is "overly covered", it is possible that for

some group V_c with coverage constraint $[l_c, u_c]$, $|V_p \cap V_c| < l_c$ due to the rejection of the nodes. Unlike [16] that simply take random nodes to fill in the gap, for each such group, Online-FGS invokes a procedure PostSelect to enrich both V_p and $\mathcal{P}$ with the nodes in B_c to ensure coverage constraints and guarantees on correction error.

Procedure PostSelect (not shown). For each group $V_c \in \mathcal{V}$ where $|V_p \cap V_c| < l_c$, procedure PostSelect performs another round of "select-and-summarize" process to make $\mathcal{P}$ satisfy the coverage constraint. It (a) dynamically selects the top $(l_i - |V_p \cap V_i|$ nodes from B_c, where each node v maximizes $F(V_p \cup \{v\})$-$F(V_p)$; and (b) follows the swapping strategy (lines 13-14, UpdateP) to update $\mathcal{P}$ with new patterns from the r-hop neighbors of the new nodes added to V_p. This repeats until V_p covers all groups with desired lower bounds.

Analysis. The algorithm Online-APXFGS iteratively process each group node $v \in \bigcup \mathcal{V}$ and ensures the following. (1) The dynamic decisions made by procedure updateVp on accepting or rejecting a node v to V_p follows the greedy streaming submodular maximization [16]. (2) Procedure updateP ensures that $V_p \subseteq \mathcal{P}_\mathcal{V}$ during the swapping strategy; and the procedure ExtendableM ensures that no group is overly covered by $\mathcal{P}$. (3) The procedure PostSelect ensures that no group is insufficiently covered, by enriching both V_p and $\mathcal{P}$. These ensure that Online-FGS correctly maintains an r-summary of "revealed" $\mathcal{V}$ in each iteration and when terminates.

Approximability. The approximation guarantees follow from the $\frac{1}{4}$ approximation ensured by streaming submodular maximization, and online optimization of maximum coverage. In particular, assume $\mathcal{P}^*$ incurs $\mathcal{C}_l^*$ at each round, we show that the swapping strategy in procedure UpdateP, which exchanges P^- with smallest marginal gain with a new counterpart P^+ having the maximized marginal gain, incurs a gap between $\mathcal{C}_l^*$ and $\mathcal{C}_l$ that is bounded by $\frac{1}{k}$ of $\mathcal{C}_l^*$, given that $|\mathcal{P}| \leq k$.

Time cost. There are at most $|\bigcup \mathcal{V}|$ iterations, and in each iteration, (1) it takes procedure UpdateVp $O(\log k)$ time to process each group node $v \in \bigcup \mathcal{V}$; (2) procedure UpdateP takes $O(k \cdot T_I)$ time to verify if v is a match, $O(|E_v^r|)$ time to update the marginal gain, and $O(N_v \cdot T_I)$ time to generate and verify any new patterns from E_v^r, with N_v bounded by N, the total number of patterns verified. The post processing takes $|\sum l_c \cdot N \cdot T_I$ time to enrich $\mathcal{P}$, where $\sum l_c$ is the sum of all the lower bounds. Putting these together, the total cost is in $O(|\bigcup \mathcal{V}| \cdot (\log n + N \cdot T_I) + \sum l_c \cdot N \cdot T_I)$ time.

The above analysis verifies Theorem 6. We present the detailed proofs in [**?**].

VII. Maintenance of Group Summaries

Real graphs are constantly changing. When new nodes join the interested groups $\mathcal{V}$ or new links are formed among group nodes, it is expensive to recompute an r-summary from scratch. We next introduce an incremental algorithm to maintain a feasible r-summary upon the arrival of new nodes and edges. The algorithm dynamically maintains an

Algorithm Inc-FGS

Input: graph G, groups $\mathcal{V}$, utility function F, batch update ΔE; configuration C=$\{r, n\}$, set V_p; r-summary $\mathcal{S}$=$(\mathcal{P}, \mathcal{C})$.
Output: an updated feasible r-summary $\mathcal{S}'$ for $G \oplus \Delta E$.

1. initializes set $\Delta\mathcal{V}$ with $\mathcal{V}$ and ΔE;
2. **if** $\Delta\mathcal{V}$:=$\emptyset$ **return** $\mathcal{S}$;
3. **else** update $\mathcal{V}$; set ΔE_r:=$\emptyset$; set $\mathcal{P}_u$:=$\emptyset$;
   / incrementalized node selection */*
4. set V_p' := IncFairSel $(V_p, \Delta\mathcal{V}, \mathcal{V}, F, n)$; set $\Delta V_p := V_p' \setminus V_p$;
5. **for each** $P \in \mathcal{P}$ **do**
6. **if** $P(u_o, G \oplus \Delta E) \cap \mathcal{V} = \emptyset$ **then** $\mathcal{P} := \mathcal{P} \setminus \{P\}$;
7. **for each** $v \in \Delta V_p$ **do**
8. $\Delta E_r := \Delta E_r \cup E_v^r$;
9. set ΔP_c := SumGen $(V_p', \Delta E_r, r)$;
   / incrementalized summarization */*
10. **while** $\Delta V_p \neq \emptyset$ **do**
11. $\mathcal{P}_u$:=$\emptyset$;
12. **for each** $P \in \Delta P_c \setminus \mathcal{P}$ **do**
13. **if** Extendable $(P, \mathcal{P}, \mathcal{V}, n)$ **then** $\mathcal{P}_u$:=$\mathcal{P}_u \cup \{P\}$;
14. P^*:=$\arg\max_{P_u \in \mathcal{P}_u} \frac{|P_u(u_o, G \oplus \Delta E) \cap V_p'|}{\mathcal{C}_{P_u}}$;
15. $\mathcal{P}$:=$\mathcal{P} \cup \{P^*\}$; ΔV_p:=$\Delta V_p \setminus P^*(u_o, G \oplus \Delta E)$;
16. set $\mathcal{S}' = (\mathcal{P}, \Delta E_r \setminus \mathcal{P}_E)$;
17. **return** $\mathcal{S}'$;

Fig. 7. Algorithm Inc-FGS

r-summary $\mathcal{S}$ of possibly changed $\mathcal{V}$, where the coverage constraints may also be updated, and preserves "anytime" utility and group fairness (coverage) guarantee.

Incrementalization. Our idea is to incrementalize the "selection-and-summarization" phases.

(1) Upon receiving a batch of edge insertions ΔE, it updates the current V_p to V_p' that satisfies both coverage constraints with approximated optimal utility F. This can be performed by invoking a streaming algorithm that process a sequence of new nodes induced from ΔE, following [16].

(2) Once V_p is updated to V_p', the summarization phase finds the new nodes ΔV_p not in V_p, and the r-hop edges of such nodes. This creates a small instance for the summarization task. Due to the strong data locality of subgraph isomorphism, it suffices to generate and verify new patterns from these edges, incrementally update $\mathcal{P}$ to ensure the coverage of ΔV_p, and update $\mathcal{S}$ accordingly.

Algorithm. The algorithm, denoted as Online-FGS and illustrated in Fig. 7, processes edge insertions in batches ΔE. (1) It first verifies if the edge insertions affect group nodes and their neighbors. If not, the original r-summary $\mathcal{S}$ is returned as there is no need to update the summary (lines 1-2). Otherwise, it updates $\mathcal{V}$ and invokes a procedure IncFairSel that follows a streaming fair submodular maximization process [16] to update V_p to V_p' (lines 3-4). It also refines $\mathcal{P}$ by removing any patterns that do not contribute to group coverage in $G \oplus \Delta E$, where $\oplus$ means "applying" the edge insertions to G (lines 5-6). (2) Online-FGS then invokes SumGen only in a (small) bounded fraction of affected nodes and r-hop neighbors, to generate new patterns. It incrementalizes the pattern selection as in APXFGS (lines 10-16) and update $\mathcal{P}$ necessarily with patterns that incurs small $\mathcal{C}_l$. This repeats until ΔV_p is covered

by $\mathcal{P}$ (line 10). The updated $\mathcal{S}'$ is then returned (line 17).

Analysis. It has been verified that an n-set for fair submodularity maximization can be maintained at a competitive ratio $\frac{1}{4}$, with a greedy swapping strategy [16]. At any time, algorithm Online-FGS maintains a feasible r-summary which has the the matching quality guarantees on F and covers the updated $\mathcal{V}$ that satisfies the coverage constraints.

For time cost, the delay time on processing a batch of edge insertion takes (1) $O(\min(n, |\Delta E|) \cdot |\bigcup \mathcal{V}|)$ to update V_p, (2) $O(|\Delta E| \cdot T_I \cdot |\mathcal{P}|)$ to refine $\mathcal{P}$ (lines 5-6), and (3) $O(n \cdot m^2)$ to update $\mathcal{S}$, where m is the number of newly generated patterns from the small fraction of $E^r_{\Delta\mathcal{V}}$. Our tests verified that Online-FGS can process batch update efficiently and significantly outperform APXFGS in efficiency with comparable utility (see Section VIII).

VIII. Experiments

Using real-world attributed graphs, we experimentally verify the effectiveness and efficiency of our algorithms.

Experiment Setting. We used the following setting.

Datasets. We used three real-life graphs. (1) DBP [26] is a movie knowledge graph induced from DBpedia with 1M of nodes and $3.18M$ of edges. Each node has a label (*e.g.,* movie, director, actors; in total 115 types) and attributes such as title, genre (*e.g.,* "Action","Romance"), years and country. Each relation has a label (*e.g.,* directed, collaboration; in total 398 types). (2) LKI [48] is a social network with $3M$ nodes denoting users and organizations, and 26M edges denoting co-review and employment. Each node has attributes such as major, and a synthetic gender attribute with values generated with gender inference tools [7]. (3) Cite [40] has 4.9M nodes with types such as papers and authors, and 46M edges denoting "citations" and "authorship". Each node has attributes such as citation number and topic.

Groups and Utility Functions. We considered the following settings for real-world applications. (1) For fair movie recommendation, we induced 5 groups of movies from DBP based on their genres and countries. We set the utility function F for a set of movies as $F(V_S) = \sum_{v \in V_S} Rating(v)$. (2) For diversified and fair talent search, we induced 6 groups of users in LKI based on their gender and degree, *e.g.,* ({gender:male; degree:BS},{gender:female; degree: BS},{gender:female; degree:MS}). we defined F as $F(V_S) = |\bigcup_{v \in V_S} N(v)|$, where $N(v) = \{u : (u, v) \in E\}$. This function, adapted from social influence maximization [19], [14], favors representative candidates that maximize the professional impact across their peers via "co-reviewed" relation. (3) To recommend collaboration, we induced 4 groups $\mathcal{P}$ of papers with different topics (*e.g.,* "Machine Learning","Networking") from Cite. We used the same function F as in (2), yet induced by relation "citation". In this case, we aimed to summarize the papers of desired coverage of topics along with influenced citations.

We set r in a principled manner to preserve comprehensive information. For each group, we inspected their r-hop neighbors as r grows from 1 (one-hop neighbors) until no new information *e.g.,* node labels are included. For DBP, LKI and Cite, r is set to be 3, 5 and 3, respectively. We set diameter $d = r$ consistently for d-summaries in d-sum. We also investigated the impact of k, the number of patterns in $\mathcal{S}$, and set k=20 by default to allow a large enough coverage of groups $\mathcal{V}$.

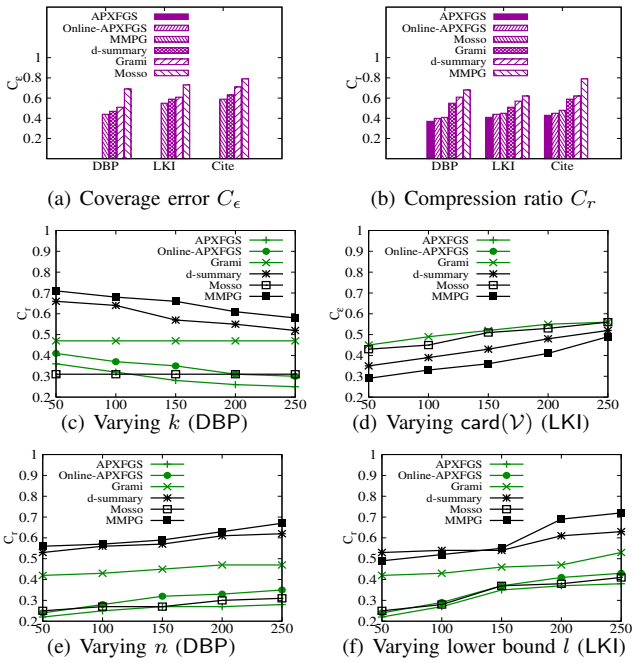


Fig. 8. Effectiveness of Fair Graph Summarization

Algorithms. We implemented the following summarization approaches for FGS, all in Java. (1) Our lossless approaches are APXFGS, Online-APXFGS and Inc-FGS. (2) Grami is adapted from [10]. It mines top-k frequent subgraphs as summary patterns. (3) d-sum is a lossy summary approach adapted from [41]. It efficiently generates k graph patterns that approximately match their counterparts in original graphs. d-sum encourages "larger" patterns to balance the informativeness and frequency of summary structures. (4) MMPG is a lossy summary approach adapting [33]. It computes k reformulated patterns from a specified one to diversify the nodes they cover. (5) Mosso [20] is a lossless graph compression method. It incrementally updates super nodes and edges to summarize a dynamic graph with small edge corrections. APXFGS was compared with Grami, MMPG, and d-sum as all methods can be categorized as pattern-based summarization approaches [25]. We compared Mosso and Inc-FGS (both lossless) over dynamic edge streams to evaluate online performances.

Experimental results. We next presented our findings.

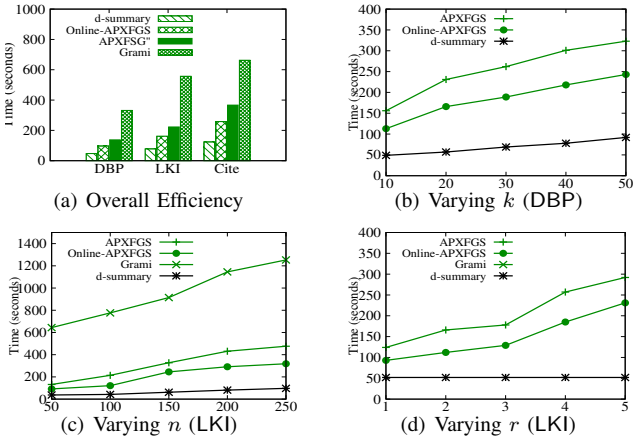
Exp-1: Effectiveness. We first evaluated the coverage and compression rate of the algorithms. Given a summarization algorithm $\mathcal{A}$, groups $\mathcal{V}$ and a summary structure $\mathcal{S}$, we used two normalized measures below. (1) The *coverage error* of

$\mathcal{A}$, adapted from set selection with fairness [16], quantifies the accumulated "gap" between the nodes covered by $\mathcal{S}$ from $\mathcal{A}$ (denoted as $V_{\mathcal{S}}$) and the required coverage of all groups. It is defined as $C_\epsilon(\mathcal{A}) = \frac{\sum_{V_i \in \mathcal{V}} \max\{|V_{\mathcal{S}} \cap V_i| - h_i, l_i - |V_{\mathcal{S}} \cap V_i|, 0\}}{|\mathcal{V}|}$. $C_\epsilon(\mathcal{A}) \in [0, 1]$. The smaller $C_\epsilon(\mathcal{A})$ is, the better. (2) We adapted the *compression ratio* of $\mathcal{A}$ consistently with Mosso [20]. It quantifies the representation size of S (including the edge size of summary patterns $|S|$ and edge correction sizes $|S.C|$), normalized by the edge size of r-hop graphs of $\mathcal{V}$ (denoted as $|G_{\mathcal{V}}|$). It is defined as $C_r(\mathcal{S}) = \frac{|\mathcal{S}| + |\bigcup_{S \in \mathcal{S}} S.C|}{|G_{\mathcal{V}}|}$ ($C_r(\mathcal{S}) \in [0, 1]$). Smaller $C_r(\mathcal{S})$ indicates more "compact" S with smaller reconstruction cost.

Effectiveness. Setting group size $\mathsf{card}(\mathcal{V}) = 2$, $r = 2$, $k = 20$, and $n = 100$, and the cardinality constraint as $[40, 60]$ for both groups, we reported the coverage error and compression ratio of all the algorithms in Figs. 8(a) and 8(b), respectively. Fig. 8(a) verifies the following. (1) Our algorithm APXFGS and Online-APXFGS achieve the optimal coverage with C_ϵ = 0, as they compute the summaries that satisfy the group coverage constraints. (2) Grami discovers summary patterns as frequent subgraphs and is more sensitive to cover major population. Mosso focuses on compressed representation of dense edge connections rather than group coverage. Both have higher coverage errors. (3) d-sum and MMPG have a comparable performance in C_ϵ, and both outperform Mosso and Grami. This is because they both optimize a bi-criteria objective that strikes a balance between pattern size and the diversity. This allows a better coverage of $\mathcal{V}$ compared with Mosso and Grami. Fig. 8(b) tells us the following. (1) While achieving the optimal coverage, APXFGS achieves the smallest (on average 0.39) compression ratio. It outperforms Online-APXFGS,Grami, d-sum and MMPG by 8%, 27%, 41% and 79% in C_r, respectively. (2) Mosso has a comparable performance and achieves 0.44 on average, due to its design for compact summaries. (3) MMPG favors larger patterns (by adding edges) to diversify the covered nodes, leading to larger summary structures. On the other hand, d-sum allows more compact structures to cover more neighbors with approximate pattern matching.

We next evaluated the impact of several factors.

Varying k. Fixing $\mathsf{card}(\mathcal{V}) = 2$, $n = 100$, and $r = 2$, we varied k from 10 to 50 and evaluated its impact to compression ratio over DBP (Fig. 8(c)). (1) Larger k allows APXFGS, d-sum and MMPG to summarize the neighborhood better with more summary patterns and smaller edge errors. On the other hand, using 50 patterns, APXFGS achieves a compression ratio at 0.26 for an underlying graph of $1M$ nodes and $3.2M$ edges, and outperforms Online-APXFGS, d-sum and MMPG. (2) Mosso only generates a single summary graph and is insensitive to k. APXFGS achieves a comparable compression ratio, and outperforms Mosso when $k \geq 20$. This shows that APXFGS can effectively exploit extendable summaries and existing edge corrections to avoid introducing too many new ones. Our results over other datasets are consistent, thus omitted.



(a) Overall Efficiency (b) Varying k (DBP) (c) Varying n (LKI) (d) Varying r (LKI)

Fig. 9. Efficiency of Graph Summarization

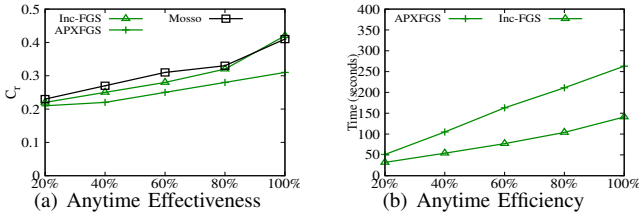
Varying $\mathsf{card}(\mathcal{V})$. Fixing $n = 240$ $r = 2$, and $k = 20$, we varied $\mathsf{card}(\mathcal{V})$ from 2 to 6 and evaluated its impact to C_ϵ over LKI. For all groups, APXFGS ensures optimal group coverage (C_ϵ = 0) as its capability to guarantee the group coverage. It outperforms d-sum, Grami and Mosso by 8%, 17% and 19% on average in C_ϵ, respectively. As $\mathsf{card}(\mathcal{V})$ increases, all other pattern-based summarization methods have degraded performance in coverage, due to that it becomes more difficult for them to maintain the coverage error over more groups.

Varying n. Fixing $\mathsf{card}(\mathcal{V}) = 2$, $k = 20$, and $r = 2$, we varied n (the size of nodes to be summarized) from 50 to 250 and evaluate its impact to compression ratio over LKI. Fig. 8(e) verifies that it is more difficult to maintain the compression ratio when more representative nodes are to be covered even when the group size is fixed. This is due to that larger amount of neighborhood information needs to be summarized, causing more edges to be either missed or added to edge correction.

Varying lower bounds. Fixing $|\mathcal{V}| = 2$, $k = 20$ $r = 2$ and $n = 500$, we varied the lower bound l from 50 to 250, while keeping the upper bound h to be 260, and evaluated impact of coverage requirement to compression ratio over LKI. As shown in Fig. 8(f), all methods perform worse in compression ratio as more nodes and neighbors are required to be summarized. With fixed number of summary patterns, APXFGS responses with larger representation size to ensure optimal coverage, yet still achieves a comparable compactness with Mosso. This verifies its effectiveness under various coverage requirements.

Exp-2: Efficiency. We next evaluated the efficiency of the algorithms. For a fair comparison, we only compared the cost of pattern-based summarization APXFGS, Online-APXFGS, Grami and d-sum.

Efficiency. Using the same setting as in Figs. 8(a) and 8(b), we report the efficiency of APXFGS, Grami and d-sum in Fig. 9(a). APXFGS outperforms Grami by,1.13 times on average. Indeed, APXFGS discovers summary patterns over selected representative nodes and their neighbors, while Grami performs frequent pattern mining over all group nodes. On the other hand, d-sum takes the least time with lossy graph pattern matching, at the cost of high coverage error (see



(a) Anytime Effectiveness (b) Anytime Efficiency

Fig. 10. Online Graph Summarization

Fig. 8(a)). Besides, Online-APXFGS outperforms APXFGS by 1.2 times due to that Online-APXFGS only incrementally evaluates patterns that are generated locally.

Varying k. Using the same setting as in Fig. 8(c), we report the efficiency of APXFGS, and d-sum in Fig. 9(b). Grami always output all the frequent subgraphs and is not sensitive to k (thus not shown). All the algorithms take more time to output the k summaries as k becomes larger. APXFGS outperforms Grami 1.85 times and Online-APXFGS outperforms APXFGS 1.25 times om average. d-sum remains to be the fastest due to lossy matching, yet does not guarantee group coverage.

Varying n and r. Using the same setting as in Fig. 8(e), Fig. 9(c) reports the efficiency of APXFGS, Online-APXFGS, Grami and d-sum. As n increases, all four methods take more time to summarize more nodes from the group, due to larger underlying neighborhood graphs to be covered. APXFGS outperforms Grami by 1.6 times on average.

Fixing $n = 50$, $k = 20$ and $\mathsf{card}(\mathcal{V}) = 2$, we varied the hop constraint r from 1 to 5 and reported the efficiency of APXFGS, Grami and d-sum in Fig. 9(d). As r increases from 1 to 5, APXFGS takes more time, and up to 310 seconds, to generate larger patterns that can cover the r-hop neighbors of the group nodes as needed. Grami (resp. d-sum with d=5) lacks the support to such flexibility and yields same results as top-k most frequent (resp. diversified and lossy) summary patterns without group coverage guarantees.

Exp-3: Online Summarization. We simulated a sequence of edges of LKI by (a) randomly selecting from 2 groups of in total $10K$ nodes, and (b) inducing r-hop neighbor graphs ($r = 2$) of the selected nodes and extract edges. We reported the anytime performance of Inc-FGS and Mosso upon the "seen" fraction of the graph. Fig. 10(a) reports the compression ratio of Inc-FGS, APXFGS and Mosso. As more subgraphs arrive, all the algorithms require larger summary structures that lead to higher compression ratio. APXFGS recomputes the summaries from scratch with smaller error corrections and summary patterns, and outperforms Mosso by 19% in C_r.

On the other hand, Inc-FGS outperforms APXFGS by 1.56 times in processing batches of edge insertions, with summaries of comparable size and optimal coverage ($C_e = 0$), and improves the efficiency of APXFGS better as larger batches arrive (Fig. 10(b)). This verifies the incremental summarization strategy in Inc-FGS is feasible for large graphs.

Exp-4: Case Study. We conducted case analysis to evaluate how r-summaries help with (1) understanding graph search with balanced gender distribution, using LKI; and (2) analyzing

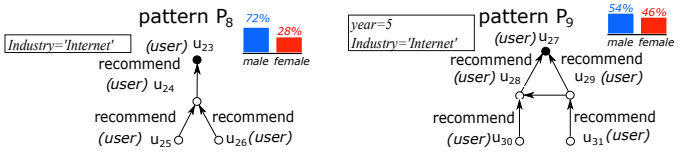


Fig. 11. Case study: Fair Graph Summarization

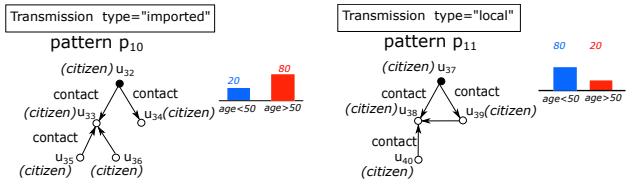


Fig. 12. Summarizing Pandemic Spreading for Immunization Strategies.

the pandemic propagation mechanism with configurable age distribution, using a pandemic spreading network.

Talent Search. A pattern query P_8 aims to search for candidates in Internet industry. Over LKI, it retrieves 15200 candidates, among which 77% are males, and 23% are females, which is not very desirable for the need of equal opportunity. (Given $r = 2$, $n = 100$ and two gender groups $\mathcal{V}$ where each group $V_i \in \mathcal{V}$ is enforced with a equal constraint $[40, 60]$, APXFGS identifies $\mathcal{S}_9$ as a 2-summary with pattern P_9. Treating S_9 as a "materialized view" over LKI, S_8 efficiently retrieves a smaller, representative, high quality candidates with 57% male candidates and 43% female candidates over 90 candidates in "Internet" industry. S_9 helps reduced 82% of the time cost for processing the query P_8. Finally, P_9 suggests revisions to P_8 to understand the results towards new queries.

Pandemic Analysis. Our second case study investigate how r-summaries help pandemic analysis and group immunization. Recall the real-world pandemic spreading network G' [1] in Example 3. Over the spreading network, there are 10000 citizens, among which 58% are young citizens ($age < 50$) and 42% are senior ($age \geq 50$). Given 10 seed nodes, and 100 vaccine budgets, we simulated the group immunization [47] over G'. We tested different configurations for the group immunization, and report two vaccine distributions $[80, 20]$ (by setting the bound (80,80) for age group 1 and (20,20) for age group 2) and $[20, 80]$, respectively. By setting vaccine distribution as $[80, 20]$, 315 citizens are infected while 116 are infected for $[20, 80]$, indicating a better vaccine strategy from the latter case. The patterns P_{10} and P_{11} further suggests frequent social contact patterns from the selected seeds. Both well summarize the spreading network in terms of "individual popularity" (P_{10}) and ""nominations contact" (P_{11}), as consistently observed in [17].

IX. Conclusions

We have introduced a class of r-summaries to summarize node groups and their neighbors with fairness constraints (in terms of group coverage constraints) in graphs. We have verified the hardness of the summarization problem, and provided feasible approximation algorithms and incremental algorithms to compute and maintain r-summaries, with guarantees on coverage and quality properties. As verified analytically and

experimentally, our methods are feasible to support graph summarization among other applications. A future topic is to support more types of fairness constraints.

REFERENCES

[1] covid-19-india-data. https://github.com/imdevskp/covid-19-india-data/, 2018.

[2] Z. Abbassi, V. Mirrokni, and M. Thakur. Diversity maximization under matroid constraints. In *KDD*, 2013.

[3] L. Akoglu, M. McGlohon, and C. Faloutsos. Oddball: Spotting anomalies in weighted graphs. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 410–421, 2010.

[4] G. Bagan, A. Bonifati, R. Ciucanu, G. H. Fletcher, A. Lemay, and N. Advokaat. gmark: Schema-driven generation of graphs and queries. *IEEE Transactions on Knowledge and Data Engineering*, 29(4):856–869, 2016.

[5] A. Bonifati, I. Holubová, A. Prat-Pérez, and S. Sakr. Graph generators: State of the art and open challenges. *ACM Computing Surveys (CSUR)*, 53(2):1–30, 2020.

[6] D. Chakrabarti, S. Papadimitriou, D. S. Modha, and C. Faloutsos. Fully automatic cross-associations. KDD '04, 2004.

[7] Y. Dong, Y. Yang, J. Tang, Y. Yang, and N. V. Chawla. Inferring user demographics and social strategies in mobile social networks. In *KDD*, 2014.

[8] C. Dunne and B. Shneiderman. Motif simplification: improving network visualization readability with fan, connector, and clique glyphs. In *SIGCHI*, 2013.

[9] M. El Halabi, S. Mitrović, A. Norouzi-Fard, J. Tardos, and J. M. Tarnawski. Fairness in streaming submodular maximization: Algorithms and hardness. *NeurIPS*, 2020.

[10] M. Elseidy, E. Abdelhamid, S. Skiadopoulos, and P. Kalnis. Grami: Frequent subgraph and pattern mining in a single large graph. *Proceedings of the VLDB Endowment*, 2014.

[11] W. Fan, X. Wang, and Y. Wu. Answering graph pattern queries using views. In *2014 IEEE 30th International Conference on Data Engineering*, pages 184–195, 2014.

[12] M. Feldman, S. A. Friedler, J. Moeller, C. Scheidegger, and S. Venkatasubramanian. Certifying and removing disparate impact. KDD '15, 2015.

[13] Y. Ge, S. Zhao, H. Zhou, C. Pei, F. Sun, W. Ou, and Y. Zhang. Understanding echo chambers in e-commerce recommender systems. In *SIGIR*, pages 2261–2270, 2020.

[14] S. Gershtein, T. Milo, and B. Youngmann. Multi-objective influence maximization. *algorithms*, 20:33, 2021.

[15] S. C. Geyik, S. Ambler, and K. Kenthapadi. Fairness-aware ranking in search & recommendation systems with application to linkedin talent search. In *KDD*, 2019.

[16] M. E. Halabi, S. Mitrović, A. Norouzi-Fard, J. Tardos, and J. Tarnawski. Fairness in streaming submodular maximization: Algorithms and hardness. *arXiv preprint arXiv:2010.07431*, 2020.

[17] M.-G. Hâncean, J. Lerner, M. Perc, M. C. Ghiță, D.-A. Bunaciu, A. A. Stoica, and B.-E. Mihăilă. The role of age in the spreading of covid-19 across a social network in bucharest. *Journal of Complex Networks*.

[18] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146, 2003.

[19] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, 2003.

[20] J. Ko, Y. Kook, and K. Shin. Incremental lossless graph summarization. In *KDD*, 2020.

[21] K.-I. Ko and W.-G. Tzeng. Three Σ_2^P-complete problems in computational learning theory. *Computational Complexity*, 1(3):269–310, 1991.

[22] D. Koutra, U. Kang, J. Vreeken, and C. Faloutsos. Vog: Summarizing and understanding large graphs. In *SIAM*, 2014.

[23] C.-T. Li and S.-D. Lin. Egocentric information abstraction for heterogeneous social networks. In *2009 International Conference on Advances in Social Network Analysis and Mining*, 2009.

[24] C.-T. Li and S.-D. Lin. Egocentric information abstraction for heterogeneous social networks. In *ASONAM*, 2009.

[25] Y. Liu, T. Safavi, A. Dighe, and D. Koutra. Graph summarization methods and applications: A survey. *ACM Computing Surveys (CSUR)*, 51(3):1–34, 2018.

[26] J. Lu, J. Chen, and C. Zhang. Helsinki Multi-Model Data Repository. https://www2.helsinki.fi/en/researchgroups/unified-database-management-systems-udbms/, 2018.

[27] H. Ma, S. Guan, C. Toomey, and Y. Wu. Diversified subgraph query generation with group fairness. *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, 2022.

[28] A. Maccioni and D. J. Abadi. Scalable pattern matching over compressed graphs via dedensification. KDD '16, 2016.

[29] A. S. Maiya and T. Y. Berger-Wolf. Sampling community structure. In *WWW*, 2010.

[30] R. Mehrotra and E. Yilmaz. Representative & informative query selection for learning to rank using submodular functions. In *Proceedings of the 38th international ACM sigir conference on research and development in information retrieval*, 2015.

[31] B. Mirzasoleiman, A. Badanidiyuru, and A. Karbasi. Fast constrained submodular maximization: Personalized data summarization. In *ICML*, 2016.

[32] B. Mirzasoleiman, A. Badanidiyuru, and A. Karbasi. Fast constrained submodular maximization: Personalized data summarization. In *ICML*, pages 1358–1367, 2016.

[33] D. Mottin, F. Bonchi, and F. Gullo. Graph query reformulation with diversity. In *KDD*, 2015.

[34] Z. Moumoulidou, A. McGregor, and A. Meliou. Diverse data selection under fairness constraints. In *ICDT*, 2021.

[35] S. Navlakha, R. Rastogi, and N. Shrivastava. Graph summarization with bounded error. In *SIGMOD*, 2008.

[36] M. Pan, R.-H. Li, Q. Zhang, Y. Dai, Q. Tian, and G. Wang. Fairness-aware maximal clique enumeration. *arXiv preprint arXiv:2107.10025*, 2021.

[37] A. Prasad, S. Jegelka, and D. Batra. Submodular meets structured: Finding diverse subsets in exponentially-large structured item sets. *Advances in Neural Information Processing Systems*, 2014.

[38] A. Rahmattalabi, P. Vayanos, A. Fulginiti, E. Rice, B. Wilder, A. Yadav, and M. Tambe. Exploring algorithmic fairness in robust graph covering problems. In *NeurIPS*, 2019.

[39] K. Shin, A. Ghoting, M. Kim, and H. Raghavan. Sweg: Lossless and lossy summarization of web-scale graphs. In *The World Wide Web Conference*, pages 1679–1690, 2019.

[40] A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, B.-J. P. Hsu, and K. Wang. An overview of microsoft academic service (mas) and applications. WWW, 2015.

[41] Q. Song, Y. Wu, P. Lin, L. X. Dong, and H. Sun. Mining summaries for knowledge graph search. *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[42] J. Stoyanovich, K. Yang, and H. Jagadish. Online set selection with fairness and diversity constraints. In *Proceedings of the EDBT Conference*, 2018.

[43] A. Tsang, B. Wilder, E. Rice, M. Tambe, and Y. Zick. Group-fairness in influence maximization. *IJCAI*, 2019.

[44] V. V. Vazirani. *Approximation algorithms*. Springer, 2013.

[45] Y. Wu, Z. Zhong, W. Xiong, and N. Jing. Graph summarization for attributed graphs. In *2014 International Conference on Information Science, Electronics and Electrical Engineering*, 2014.

[46] Y. Yang, N. V. Chawla, and B. Uzzi. A network's gender composition and communication pattern predict women's leadership success. *Proceedings of the National Academy of Sciences*, 116(6):2033–2038, 2019.

[47] Y. Zhang, A. Adiga, A. Vullikanti, and B. A. Prakash. Controlling propagation at group scale on networks. In *2015 IEEE International Conference on Data Mining*. IEEE, 2015.

[48] Y. Zhang, J. Tang, Z. Yang, J. Pei, and P. S. Yu. Cosnet: Connecting heterogeneous social networks with local and global consistency. In *KDD*, 2015.