Density Personalized Group Query

Chih-Ya Shen
Department of Computer Science
National Tsing Hua University, Taiwan
chihya@cs.nthu.edu.tw

Shao-Heng Ko
Institute of Information Science
Academia Sinica, Taiwan
shaohengko,gslee@iis.sinica.edu.tw

Guang-Siang Lee
Institute of Information Science
Academia Sinica, Taiwan
gslee@iis.sinica.edu.tw

Wang-Chien Lee
Department of Computer Science and Engineering
The Pennsylvania State University, USA
wlee@cse.psu.edu

De-Nian Yang
Institute of Information Science
Research Center for Information Technology Innovation
Academia Sinica, Taiwan
dnyang@iis.sinica.edu.tw

ABSTRACT

Research on new queries for finding dense subgraphs and groups has been actively pursued due to their many applications, especially in social network analysis and graph mining. However, existing work faces two major weaknesses: i) incapability of supporting personalized neighborhood density, and ii) inability to find sparse groups. To tackle the above issues, we propose a new query, called Density-Customized Social Group Query (DCSGQ), that accommodates the need for *personalized density* by allowing individual users to flexibly configure their social tightness (and sparseness) for the target group. The proposed DCSGQ is general due to flexible in configuration of neighboring social density in queries. We prove the NP-hardness and inapproximability of DCSGQ, formulate an Integer Program (IP) as a baseline, and propose an efficient algorithm, `FSGSel-RR`, by relaxing the IP. We then propose a fixed-parameter tractable algorithm with a performance guarantee, named `FSGSel-TD`, and further combine it with `FSGSel-RR` into a hybrid approach, named `FSGSel-Hybrid`, in order to strike a good balance between solution quality and efficiency. Extensive experiments on multiple large real datasets demonstrate the superior solution quality and efficiency of our approaches over existing subgraph and group queries.

PVLDB Reference Format:
Chih-Ya Shen, Shao-Heng Ko, Guang-Siang Lee, Wang-Chien Lee, and De-Nian Yang. Density Personalized Group Query. PVLDB, 16(4): 615 - 628, 2022.
doi:10.14778/3574245.3574249

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://bit.ly/36QhrvV.

Chih-Ya Shen and Shao-Heng Ko contributed equally to this work.

Proceedings of the VLDB Endowment, Vol. 16, No. 4 ISSN 2150-8097.
doi:10.14778/3574245.3574249

1 INTRODUCTION

Dense subgraph queries (e.g., [22, 27, 59]) and group queries (e.g., [7, 21, 54, 74]) are important for addressing application needs in social network analysis and graph mining. These queries aim to extract a set of users or entities which are modeled as vertices in a graph by considering different relationships among them, such as social tightness, spatial proximity, and diversity. However, we observe two major weaknesses in the existing work:

i) *Incapability of supporting personalized neighborhood density.* Existing subgraph and group queries usually consider the *unified lower bound* of the density (or sparseness for a sparse group query), e.g., k-core requires *each selected individual* to have a degree *at least* k, and a k-plex of size p asks *each individual* to have *at least* $p-k-1$ friends within the subgraph. This implicitly encourages us to find groups that are denser (or sparser for a sparse group query) than the specified lower bound. However, in many cases, individuals usually need the *personalized* and *appropriate* neighborhood density, rather than the *densest (sparsest)* neighborhood. For example, in a social event, Alice may require at least 3 and at most 5 friends in the group, in order to make new friends while meeting some old ones; while at the same event, Bob may need at least 2 and at most 7 friends. Please note that setting such a range on neighborhood density is a more general form and can cover dense/sparse queries by setting the lower bound to 0 or upper bound to ∞. Therefore, it is more desirable to allow each individual to specify the upper and lower bounds of her neighborhood density.

While personalized recommendation systems [20, 37] have been studied extensively to address each user's personal preference, very little effort on personalization is made in subgraph and group queries. By considering this personalization factor, users may freely configure the desired density of their social neighborhoods in the target group. For example, in a social event, introverts may require more friends than other attendees to provide sufficient social support.[1]

ii) *Inability to find sparse groups.* Most subgraph and group queries focus on extracting *dense* (i.e., socially tight) groups, with

[1]Please note that such preference on neighborhood density can be learned from people's interaction history or manually configured. We revised previous work [15] and build a machine learning model that is able to accurately infer users' preferences on neighborhood density (detailed in [56]).

little effort spent on finding sparse groups that require only a small number of social interactions within the group. Indeed, sparse groups find many applications, e.g., forming psychoeducational groups and reviewer selection [19, 51, 53]. Studies also show the strengths of sparse groups. For example, the members of a sparse group are less emotionally attached, leading to more interpersonal comfort with less social complications [65]. Also, the loose acquaintances of sparse group members can help generate creative ideas [13], and facilitate the exchange of novel information with diverse others [28]. In addition, in some friending events, some participants may prefer to have few acquaintances in the activity to increase interactions with new friends (social sparseness requirement). However, existing dense subgraph and group queries do not meet this need.

Addressing these two weaknesses of the existing subgraph and group queries all together is very challenging. To tackle these issues, we propose a novel notion of *personalized (local) density requirement*, which allows *individual* users to configure the desired lower and upper bounds of their neighboring social densities in terms of certain social tightness metrics. With this personalized density requirement, we are able to extract a group in which *each* member's social density requirement is satisfied, e.g., an extrovert member requires *at most* 4 friends in the group (social sparseness), another introvert member requires *at least* 6 friends (social tightness), and other members require 3 to 8 friends in the group. Facilitating density personalization in a group may boost a satisfactory atmosphere in the group. Moreover, the idea of personalized density *has not been pursued in research on group queries before.*

In this paper, we formulate a new research problem, named *Density-Customized Social Group Query (DCSGQ)*, which facilitates personalized (local) density requirements for each vertex as well as the social tightness and sparseness requirements, such that the total vertex weight of the selected group is maximized.[2] Here, DCSGQ maximizes the total vertex weight because this objective function is widely adopted in subgraph queries, such as [26, 52, 69].[3] Answering DCSGQ is very challenging because all the selected vertices' local density requirements (either tightness or sparseness) need to be satisfied simultaneously. Please note that by considering *merely the social tightness*, many subgraph and group queries are already NP-hard and inapproximable problems [27, 38, 60].

We show that DCSGQ is NP-hard and inapproximable within any factor. We first formulate an integer program (IP), solvable by any commercial solver (e.g., Gurobi[4]), to serve as a baseline in this study. Moreover, we propose an efficient algorithm, Flexible Subgroup Selection with Randomized Rounding (`FSGSel-RR`), that relaxes the above IP to solve DCSGQ in large-scale general graphs very efficiently. We then propose a fixed-parameter tractable (FPT) algorithm[5] that achieves a *guaranteed performance bound*, named Flexible Subgroup Selection on Tree Decompositions (`FSGSel-TD`), based on the idea of *tree decomposition*[6] [6, 49] that efficiently obtains the solutions with guaranteed performance (the concept of tree decomposition is to be introduced in Section 5.1). Furthermore, we exploit the strengths of `FSGSel-RR` and `FSGSel-TD` to propose a hybrid approach, named `FSGSel-Hybrid`, that strikes a good balance between solution quality and efficiency. The contributions of this paper are summarized as follows.

- We propose a new query, named Density-Customized Social Group Query (DCSGQ), with the new notion of *personalized density* that allows *individual* users to configure the desired lower and upper bounds of their neighboring social densities (to address both social tightness and sparseness). To our best knowledge, this is the first work that addresses the personalized density on group queries.
- We formally formulate the DCSGQ problem with IP and prove its NP-hardness and inapproximability. We design algorithm `FSGSel-RR` for efficiently solving DCSGQ and then devise algorithm `FSGSel-TD` with a guaranteed performance bound. We also integrate the ideas of `FSGSel-TD` and `FSGSel-RR` into `FSGSel-Hybrid`, which strikes an excellent balance between solution quality and efficiency.
- We evaluate the performance of the proposed approaches on multiple large-scale networks. The results show that, in addition to the guaranteed performance proved in this paper, our proposed approaches obtain good solutions very efficiently and outperform the other baselines.

In the following, Section 2 introduces the DCSGQ problem. Section 3 reviews the related work. Sections 4-6 detail the proposed three algorithms and theoretical results. Section 7 reports the experimental results, and Section 8 concludes this paper.

2 PROBLEM FORMULATION AND ANALYSIS

The notation table is presented in the online full version [56]. Let $G = (V, E)$ be a social graph with a vertex set V of n users and a set E of m edges denoting their social relationships, where each user $v \in V$ is associated with a nonnegative vertex weight $\mathrm{w}(v) \geq 0$ of v,[7] and each edge $e = (u, v) \in E$ is associated with a nonnegative weight $\mathrm{f}(u, v) \geq 0$, indicating the strength of friendship. To plan a social event, following [18, 70], an *initiator* $v_0 \in V$ is specified in order to select a group of users from her h-neighborhood, i.e., the h-hop friends of v_0 in the network, which is formally defined as follows.

Definition 1 (h-neighborhood ($\mathrm{N}_J^h(u)$) in subgraph J)**.** Given a positive integer h, a user u, and a subgraph $J \subseteq G$, a vertex $v \in V$ is said to *belong in u's h-neighborhood in J* if and only if there exists a path of at most h edges from u to v in J. We denote $\mathrm{N}_J^h(u)$ as the set containing all vertices in u's h-neighborhood.[8]

[2] Vertex weights can represent willingness, preference, spatial distance, etc.

[3] We can employ a simple approach to transform a problem that minimizes total vertex weight to one that maximizes total vertex weight.

[4] http://www.gurobi.com.

[5] FPT algorithms can solve the problems efficiently for small values of the fixed parameter appearing in the exponent of the time complexity.

[6] Tree decomposition has been employed to solve various NP-hard graph optimization problems due to its fixed-parameter tractability and good performance [8, 9, 47].

[7] The meaning of $\mathrm{w}(v)$ depends on the application scenario, e.g., the user preference of an activity (inviting users who are interested in an art exhibition), or the importance of the user to a social event. Such information can be learned with machine learning approaches, e.g., [63, 73].

[8] When the context is clear, e.g., $J = G$, we omit the subscript J and use the simplified notation $\mathrm{N}^h(u)$. Also, when $h = 1$, we omit the superscript h and use $\mathrm{N}_J(u) = \mathrm{N}_J^1(u)$

Aiming to meet users' personalized preferences on social tightness, a set of *personalized local density constraints* are imposed on the targeted subgraph $H \subseteq G$.

Definition 2 (Personalized local density constraint)**.** Given a subgraph $H \subseteq G$, a metric of local density $\mathrm{d}_H(\cdot)$, and two non-negative numbers $\bar{d}(u)$ and $\underline{d}(u)$ denoting the acceptable range of local density for each vertex $u \in H$.[9] We say that H satisfies u's *personalized local density constraint* if and only if $\bar{d}(u) \geq \mathrm{d}_H(u) \geq \underline{d}(u)$. For example, for core-based groups that target on vertex degree (or weighted degree), $\mathrm{d}_H(u) = \sum_{v \in \mathrm{N}_H(u)} \mathrm{f}(u, v)$, where $\mathrm{f}(u, v)$ is the aforementioned edge weight function, and $\mathrm{N}_H(u) = \mathrm{N}^1_H(u)$ is the direct neighborhood of u in H.

To address the two weaknesses as stated in Section 1, we propose a new query, named *Density-Customized Social Group Query (DCSGQ)*, which considers both the upper and lower bounds of the local density constraints. Note that the local density constraint in DCSGQ is very general in that i) the density requirements are completely personalized for each vertex (individual)'s ego network; and ii) the local density is regulated by an upper bound and a lower bound. We formally introduce the DCSGQ problem as follows.

Problem: Density-Customized Social Group Query.
Given: a social network $G = (V, E)$, an initiator $v_0 \in V$, vertex weight $\mathrm{w}(u) \geq 0$, edge weight $\mathrm{f}(u, v) \geq 0$, hop count $h \in \mathcal{N}$, a local density metric $\mathrm{d}_H(u)$ (i.e., $deg_H(u)$), and personalized local density thresholds $\bar{d}(u)$ and $\underline{d}(u)$, for all $u, v \in V$.
Find: a subgraph $H^* \subseteq G$ to maximize the total vertex weight of vertices in H^*, i.e., $\max_{H^*} \sum_{v \in H^*} \mathrm{w}(v)$, such that

- $v_0 \in H^*$ (initiator constraint).
- $H^* \subseteq \mathrm{N}^h_G(v_0)$ (distance constraint).
- $\bar{d}(u) \geq \mathrm{d}_{H^*}(u) \geq \underline{d}(u), \forall u \in H^*$ (local density constraint).[10]

Here, the initiator v_0 could be viewed as the *activity organizer*, which commonly appears in activity organization services, such as *Meetup* and *Facebook Event* and is included in many previous research works [24, 29, 30, 40, 44]. In some applications such as social event matching, it might not be necessary to include the initiator in the resulted query. In such cases, one can lift the initiator constraint to create a *no-initiator* variation of DCSGQ. This variant can be easily reduced to the original problem. As many existing subgraph and group queries do not specify an initiator, we do not explicitly distinguish between these two variants in the following discussions. The personalized local density requirements that cover both the lower and upper bounds significantly raise the algorithmic challenges of handling DCSGQ. We first analyze the hardness and inapproximability of the proposed DCSGQ below.

Theorem 1. *DCSGQ is* NP*-hard to approximate within any ratio.*

Proof. We prove the theorem via a gap-introducing reduction from Independent Set. Please see [56] for details. □

3 RELATED WORK

Dense and sparse subgraph queries. Dense subgraph discovery [22, 27, 59] is a fundamental problem in graph data mining with numerous applications such as community detection [24, 60], reachability in databases [34], and subgraph search [32, 35, 58]. An excellent survey [23] is presented for a complete discussion on community search. Among various measures for modeling subgraph cohesiveness, cliques (and their relaxations) attract much research attention [12, 45, 50, 52, 64, 76], while k-truss structures [3, 33] and k-core structures [21, 61, 74] are also important tightness measurements. In contrast to the dense subgraph discovery, finding *sparse* (or *tenuous*) subgraphs is an emerging research direction. The sparse k-connected graph problem aims to find the k-connected spanning subgraph with the minimum number of edges [48], the L2DS problem extracts a 2-spanner of an input graph that minimizes the maximum degree [19], while $c_{>t}$-free graph problem aims at extracting subgraphs without any induced cycle on more than t vertices [25]. In addition, MkTG [53] and others [31, 51] target on extracting sparse (or tenuous) groups with the minimized interactions. However, there is no existing work on personalizing the local density with respect to individual vertices in graphs.
Group queries with multiple dimensions of attributes. Group queries can be viewed as extensions of dense subgraph queries, where many research works focus on extracting socially tight groups while reducing the spatial distances [16, 21, 26, 36, 40, 52, 57, 61, 76]. In addition, the task completion [68], potential friendships [54, 55], dissimilarity [67, 74], or profiles [17] are also considered. Although these group queries fit their respective application scenarios well, however, they do not explore the issues of personalization. In recommendation systems, methods and issues for personalized recommendation have been extensively studied [20, 37, 43]. However, for dense subgraph and group queries, the idea of allowing users to personalize their local densities in the selected group is totally new and its research challenges have not been explored.
Summary. The proposed DCSGQ is new and important in the following aspects: i) personalization of social tightness in users' social neighborhoods, ii) consideration of both social tightness and sparseness, and iii) flexibility of specifying neighborhood density.

4 ALGORITHM WITH ROUNDING (FSGSEL-RR)

In this paper, we propose three effective algorithms to tackle DCSGQ from different perspectives. We first develop a scalable approach, called Flexible Subgroup Selection with Randomized Rounding (FSGSel-RR), that retrieves a promising solution very efficiently. FSGSel-RR first formulates DCSGQ as an integer program (IP) and relaxes it as a linear program with the same objective and constraints, where a *fractional* solution can be retrieved by a fully optimized solver. Finally, FSGSel-RR constructs a promising final subgroup from the fractional solutions, based on the ideas of *pipage rounding [2]* and *randomized rounding* regimes.

Although FSGSel-RR tackles DCSGQ efficiently with good solution quality, it does not have a performance guarantee. Therefore, in Section 5, we devise a fixed-parameter tractable (FPT) algorithm,

---

to refer to u's *direct neighborhood* in J. Note that $u \in \mathrm{N}^h_J(u)$ for any $h \geq 0$; also, $\mathrm{N}(u) = \mathrm{N}^1_G(u)$ is u's ego network.
[9] We can leverage the data log of users' past social activities to implicitly learn the suitable personalized local density constraints for each user, e.g., [15, 42, 75], with graph neural networks or collaborative filtering.
[10] We can leverage the data log of users' past social activities to implicitly learn the suitable personalized local density constraints for each user, e.g., [15, 42, 75].

FSGSel-TD, that achieves a good *guaranteed performance bound*. FSGSel-TD is based on our observation that DCSGQ becomes more tractable in *tree-like* structures. By leveraging the notion of *tree decomposition*, we identify the hidden *tree-like* structure of general graphs. The proposed FSGSel-TD aims to operate on the obtained tree decomposition, to cope with the challenging personalized local density requirements. Leveraging the tree decomposition for general graphs, FSGSel-TD adheres to the social tightness and sparseness constraints to achieve a good performance guarantee in graphs with low treewidths (to be formally defined later in Section 5).[11]

While FSGSel-TD achieves a good theoretical guarantee, its time complexity depends on the treewidth and thus can be computationally expensive in some extreme cases. To address this issue, in Section 6, we propose a hybrid approach, FSGSel-Hybrid, that combines the merits of FSGSel-TD (i.e., good solution guarantee) and FSGSel-RR (i.e., good scalability), to strike a good balance between solution quality and efficiency.

By exploring the idea of *pipage rounding*[12] [2], we propose *Bicriteria Attendee Selection* (BAS) in FSGSel-RR to iteratively enhance the solution in each iteration, aiming to reduce the number of fractional decision variables while maximizing the total vertex weight and minimizing the total *violation* of the personalized local density constraints. With BAS tailoring the intermediate solution towards an integral solution at each iteration, FSGSel-RR successfully finds the solution after $O(n)$ iterations, where n is the number of fractional variables. Experimental results in Section 7 show that FSGSel-RR obtains good solutions very efficiently.

Integer Program. We formulate DCSGQ with IP such that the optimal solution of DCSGQ can be acquired with commercial solvers, e.g., CPLEX or Gurobi, both as a baseline and as a building block of FSGSel-RR. Note that solving IP is NP-hard; therefore, this baseline does not guarantee efficiency. For simplicity, we assume that $\mathrm{N}_G^h(v_0)$, i.e., the h-neighborhood of the initiator v_0, is given as input to the IP.[13] Let $x_u \in \{0,1\}$ be a binary variable representing whether vertex $u \in V$ is selected in H. Further, let M be a large constant, i.e., an upper bound of $\mathrm{d}_H(u)$ for all $u \in V$. The objective function is therefore max $\sum_{u\in V} w(u) \cdot x_u$, subject to the following constraints:

$$x_{v_0} = 1, \tag{1}$$

$$x_u = 0, \quad \forall u \notin \mathrm{N}_G^h(v_0) \tag{2}$$

$$\mathrm{d}_H(u) \le \overline{d}(u) + M(1 - x_u), \quad \forall u \in \mathrm{N}_G^h(v_0) \tag{3}$$

$$\mathrm{d}_H(u) \ge \underline{d}(u) \cdot x_u, \quad \forall u \in \mathrm{N}_G^h(v_0) \tag{4}$$

$$x_u \in \{0,1\}, \quad \forall u \in V. \tag{5}$$

Constraint 1 requires the initiator v_0 to be selected in H. Constraint 2 prohibits the users outside $\mathrm{N}_G^h(v_0)$ to be selected in H. Constraint 3 states that the local density of u does not exceed $\overline{d}(u)$ if u is selected, i.e., $x_u = 1$. It is automatically satisfied if $x_u = 0$ due to the additional large term $M(1 - x_u)$. If this term is not imposed, the constraint could prohibit the legit assignment to select a lot of u's friends (a large $\mathrm{d}_H(u)$) when u is not selected ($x_u = 0$). Constraint 4 requires that the local density of u is at least $\underline{d}(u)$ if u is selected. Similarly, this is automatically satisfied if $x_u = 0$. Constraint 5 regulates every x_u to be binary. Let binary variable $y_{uv} \in \{0,1\}$ represent whether vertices u and v are both selected in H, regulated by the following constraint:

$$x_u + x_v - 1 \le y_{uv} \le \frac{x_u + x_v}{2}, \quad \forall u, v \in V. \tag{6}$$

If $x_u = x_v = 1$, both the lower and upper bounds of y_{uv} in Constraint 6 are 1, leading to $y_{uv} = 1$. On the other hand, if at least one of x_u and x_v is 0, the upper bound is either 0 or $\frac{1}{2}$, eliminating the possibility of setting $y_{uv} = 1$, whereas the lower bound does not exceed 0 (i.e., holds trivially). Constraint 6 ensures that $y_{uv} = 1$ if and only if $x_u = x_v = 1$. The local density setting is represented by a function of the y variable as follows.[14]

Linear relaxation. Following the standard linear relaxation technique [2], the LP relaxation of DCSGQ is formulated by replacing Constraint 5 with a linear constraint $0 \le x_u \le 1$, and relaxing the y variable as $0 \le y_{u,v} \le 1$. The optimal fractional solution of the relaxed problem can thus be acquired in polynomial time with a commercial solver. As the x-variables are sufficient to represent the solution of DCSGQ (i.e., x_u denotes whether individual u is selected in the subgroup), the optimal solution can be fully represented by X^* (the set of optimal x variables).

Significance to Flexibility Indicator (SFI). Given an intermediate fractional solution X, which is the set containing all $x_u, \forall u \in V$, the idea of BAS is to iteratively tailor X such that i) the number of fractional elements in X gradually decreases, and ii) the corresponding solution quality gradually improves. To strike a balance between the goals of improving the vertex weight and complying with the personalized local density constraints, we introduce the *Significance to Flexibility Indicator* (SFI) $Q(X)$ to measure the solution quality defined as follows.

$$Q(X) := \sum_{u\in V} w(u)x_u - \lambda \cdot \sum_{u\in V} g(u)x_u = \sum_{u\in V} (w(u) - \lambda \cdot g(u))x_u,$$

where $\sum_{u\in V} w(u) \cdot x_u$ is the total vertex weight achieved by the fractional solution, $\lambda > 0$ is a relative balance parameter between the above two goals, and $g(u) = \max\{0, \mathrm{d}_H(u) - \overline{d}(u), \underline{d}(u) - \mathrm{d}_H(u)\}$ is the total amount of violation imposed on the personalized local density constraint of u.[15] Therefore, a larger λ in SFI puts more emphasis on complying with the personalized local density constraints in the final solution, where a smaller λ prioritizes maximizing the objective. We empirically set λ in $[10, 100]$ according to the sensitivity tests in the online version of this paper [56]. Note that the objective value of the solution does not contain the part of λ. Suppose the λ value is adjusted (e.g., increased) between different executions of FSGSel-RR. It does not "improve" the SFI values of

[11] Based on our experimental results in Section 7, many real social networks have small treewidths after partial tree decomposition.

[12] Obtaining an integral solution of at least the value of the fractional solution.

[13] $\mathrm{N}_G^h(v_0)$ can be retrieved by a simple graph search in $O(|V| + |E|)$-time.

[14] Please note that Constraint 7 considers *weighted* degree, which also supports *unweighted degree* without any modification.

$$\mathrm{d}_H(u) = \sum_{v\in N_H(u)} \mathrm{f}(u,v) = \sum_{v\in N_G(u)} y_{uv} \cdot \mathrm{f}(u,v). \tag{7}$$

[15] Note that the computation of $\mathrm{d}_H(u)$ here takes into account both the edge weights and fractional variables, i.e., an edge with weight $\frac{1}{2}$ to a neighbor vertex v with fractional variable $x_v = \frac{1}{3}$ will contribute $\frac{1}{6}$ to $\mathrm{d}_H(u)$.

all solutions equally; instead, solutions with low violations to the constraints are favored.

Bicriteria Attendee Selection (BAS). In each iteration, BAS randomly selects a fractional element x_i from X and examines two alternative solutions X_i^+ and X_i^-, which correspondingly rounds x_i to 1 (i.e., selecting v_i into H) and 0 (i.e., not selecting v_i).[16]

Algorithm 1 FSGSel-RR

Input: X^*
Output: $H \subseteq G$
1: $X \leftarrow X^*$
2: $H \leftarrow \emptyset$
3: **while** some element in X is fractional **do**
4: **for** $i = 1$ to σ **do** (sample for σ times)
5: Sample x_i u.a.r. from all fractional elements
6: **if** $Q(X + (1 - x_i) \cdot \mathbf{1}_i) \geq Q(X)$ **then**
7: $X \leftarrow X + (1 - x_i) \cdot \mathbf{1}_i$
8: **Break**
9: **else if** $Q(X - x_i \cdot \mathbf{1}_i) \geq Q(X)$ **then**
10: $X \leftarrow X - x_i \cdot \mathbf{1}_i$
11: **Break**
12: Sample x_i u.a.r. (first σ samples failed)
13: $X \leftarrow X + (1 - x_i) \cdot \mathbf{1}_i$ with probability x_i
14: **for** $u \in V$ **do**
15: **if** $x_u = 1$ **then**
16: $H \leftarrow H \cup \{u\}$
17: **return** H

Time complexity and solution quality. Let n_{frac} denote the number of fractional variables in X^*, and let $t(Q)$ denote the complexity of evaluating $Q(X)$. As each round of FSGSel-RR samples at most σ elements, and the total time complexity of FSGSel-RR is $O(n_{\text{frac}} \cdot \sigma \cdot t(Q))$, efficiently finding a promising solution. Please see [56] for more analysis on the solution quality and efficiency of FSGSel-RR, as well as a sensitivity test on the parameters.

5 ALGORITHM ON TREE DECOMPOSITION (FSGSEL-TD)

Although FSGSel-RR tackles DCSGQ efficiently with good solution quality, it does not have a performance guarantee. Therefore, we propose a fixed-parameter tractable (FPT) algorithm, FSGSel-TD, that exploits the notion of *tree decomposition* to achieve a *guaranteed performance bound*. We first introduce the idea of tree decomposition and then detail FSGSel-TD.

[16] Let $\mathbf{1}_i$ denote the vector where only the i-th element is 1 and all other elements are 0, then $X_i^+ = (X + (1 - x_i) \cdot \mathbf{1}_i)$ and $X_i^- = X - x_i \cdot \mathbf{1}_i$. Please note that this process does not violate the local density constraint. This is because if a vertex u satisfies the local density constraint, $g(u) = 0$ holds; otherwise, $g(u) > 0$, which reduces $Q(X)$. BAS verifies whether this improves SFI, i.e., whether $Q(X_i^+) \geq Q(X)$ or $Q(X_i^-) \geq Q(X)$; if the former holds, X is replaced by X_i^+, and vice versa. If none of the above holds, BAS proceeds by re-sampling another fractional element x_i for at most σ times. In the rare case when $Q(X)$ cannot be improved within the consecutive σ samples of fractional elements (i.e., trapped in local optimum), BAS randomly rounds the $(\sigma + 1)$-th sampled element x_i to 1 with probability x_i, and to 0 with probability $1 - x_i$. BAS terminates when X becomes a 0-1 vector and thus corresponds to a subgraph $H \subseteq G$, and outputs H as the final solution.

Note that BAS with $\sigma = 0$ reduces to a straightforward randomized rounding scheme where each element is rounded with probability proportional to its fractional weight. As such, by BAS, FSGSel-RR incorporates pipage rounding (local improvement) and randomized rounding (to escape from local optimum) to capture the merits of both approaches. By contrast, without the additional randomized rounding step, BAS with pipage rounding becomes vulnerable to being trapped at a local optimum, as the solution space might not be weakly convex with respect to the criteria $Q(X)$. The pseudo code of FSGSel-RR is summarized in Algorithm 1.

5.1 Tree Decomposition and Treewidth

Group queries on social network databases, due to complicated constraints on graph structures, are often proven NP-hard, and thus computationally intractable. However, many NP-hard problems become tractable when the underlying input graph belongs to some special graph classes with good structural properties, e.g., *trees*. Take the maximum independent set problem as an example. It is NP-hard on general graphs, but it admits a simple, polynomial-time algorithm on trees [9].

Trees are a special kind of graphs. While it is usually unreasonable to limit the input graphs to trees, tree-based algorithms remain effective on graphs that are *tree-like*, i.e., similar to trees. Intuitively, for graphs with only a few additional edges more than a tree, previous research demonstrated that it is still promising to solve some hard problems efficiently and find good solutions via the tree-based algorithms augmented with a few extra steps, to address the parts in a graph that violate tree properties. Inspired by this insight, we first leverage the notion of *treewidth* [6, 49], which measures how similar an arbitrary graph is to a tree, to characterize the input graphs and devise suitable algorithms. A smaller treewidth of a graph represents that the graph is more similar to a tree. For example, a tree has a treewidth of 1, whereas the densest graphs, i.e., cliques of size n, have treewidths $n - 1$.

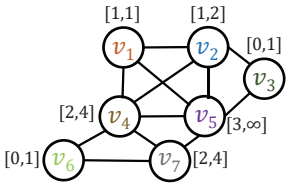
For graphs with bounded treewidths, many classical NP-hard problems become tractable, such as Independent Set, Vertex Coloring, Vertex Cover, and Steiner Tree [8]. Many social networks are inclined to have small treewidths (the real datasets in Section 7 also show this tendency) due to the small-world property, i.e., each vertex in a social network usually connects to only a small portion of other vertices. Motivated by the insights above, we leverage the notions of tree decomposition and treewidth as important building blocks to devise an efficient algorithm for DCSGQ, named FSGSel-TD, that has a *guaranteed performance bound*. We then combine FSGSel-TD with the proposed FSGSel-RR, into a hybrid approach, named FSGSel-Hybrid for general graphs to efficiently solve the problem in general graphs.

The concept of treewidths is tied to *tree decomposition* [9]. In the following, we provide essential background on treewidths and tree decompositions of graphs.

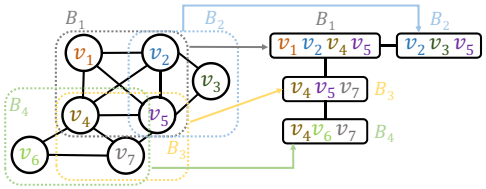
Definition 3 (Tree decomposition, width, and treewidth)**.** Given a graph $G = (V, E)$, a tree decomposition T of G is a tree structure consisting of a set of tree nodes (*bags*), where each bag $B \subseteq V$ includes a set of graph vertices, satisfying:

- for each graph edge $e = (u, v) \in E$, there exists a tree node (bag) B containing both u and v.
- for each graph vertex v, all tree nodes (bags) containing v form a connected subtree in T.

To avoid confusion, we use the term *node* interchangeably with *bags*, and use uppercase-B notations (such as B_r and B_x) *only* to refer to them, where B stands for a "bag", which is itself *a set of graph vertices*, whereas a lowercase variable (e.g., u, v, or v_0) always refers to a *vertex* (element of the original graph G) instead of a bag. For example, Figure 1(b) presents a tree decomposition of Figure 1(a). Here, B_1, B_2, B_3, B_4 each is a tree node (bag), where each bag includes a set of vertices in the graph.



(a) **Input graph.**



(b) **One tree decomposition of Figure 1(a).**

Figure 1: Illustrative examples.

Let $\mathcal{B}_T$ denote the set containing all tree nodes (bags) in T. For an arbitrary tree node (bag) B, $|B|$ denotes the *size* of B, i.e., the number of graph vertices contained in B. The *width* of a tree decomposition T is $\max_{B \in \mathcal{B}_T} |B| - 1$, and its *size* is $|T| = |\mathcal{B}_T|$. The *treewidth* of a graph G, denoted by $\omega(G)$, is the minimum width among all possible tree decompositions of G.

Treewidth measures the similarity of tree decomposition of G to a tree. A tree has a treewidth of exactly 1, while an n-vertex clique has a treewidth of $(n-1)$. Treewidth is an important measure for tree decomposition because a more succinct decomposition (i.e., with a smaller width to be more tree-like) is usually preferred. In general, determining the *treewidth* of G is NP-hard [5] because each vertex can be included in multiple bags. However, if the exact treewidth is given as a fixed constant, the corresponding decomposition can be computed in linear time [10] (fixed-parameter tractable).[17]

Example 1. Figure 1(b) illustrates a tree decomposition of the social graph G in Figure 1(a). It consists of four bags: $B_1 = \{v_1, v_2, v_4, v_5\}$, $B_2 = \{v_2, v_3, v_5\}$, $B_3 = \{v_4, v_5, v_7\}$, and $B_4 = \{v_4, v_6, v_7\}$. Note that i) B_1 to B_4 form a tree; ii) both criteria in Definition 3 are satisfied, e.g., edge (v_5, v_7) is contained in B_3, and all tree bags containing v_4 (i.e., B_1, B_3, and B_4) form a connected subtree. The width of this tree decomposition is $4 - 1 = 3$. Please note that another decomposition with only a single bag $\widehat{B} = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$ is also a valid tree decomposition. In this case, its width is $7 - 1 = 6$. □

Tree decompositions facilitate the systematic design of optimization algorithms that examine one tree bag at a time (similar to examining one vertex at a time when the input graph is a tree). However, it is still tedious to handle the relations between neighboring bags due to the many possible types of bags involved. To further simplify the bag types, it necessitates a set of additional *nice* properties on the tree decompositions detailed below.

[17] Practically, a number of heuristic strategies are often used in practice [1, 46] since the linear time algorithm is computationally expensive empirically.

Definition 4 (Nice tree decomposition [9])**.** A tree decomposition T is *nice* if, regarding some bag B_r as the root, every bag $B_x \in \mathcal{B}_T$ belongs to the following four types:

- *Leaf*: B_x has no children bags, and $|B_x| = 1$.
- *Introduce*: B_x has exactly one child bag B_y, $|B_x| = |B_y| + 1$, and $B_x = B_y \cup \{v\}$ for some graph vertex v.
- *Forget*: B_x has exactly one child bag B_y, $|B_x| = |B_y| - 1$, and $B_y = B_x \cup \{v\}$ for some graph vertex v.
- *Join*: B_x has exactly two children bags B_y, B_z, while $B_x = B_y = B_z$ (i.e., they are exactly the same vertex sets).

5.2 Design of FSGSel-TD

FSGSel-TD is based on a bottom-up approach that starts from a singleton vertex (as the smallest possible subgraph, i.e., leaf bag) and gradually builds up the *profile* (i.e., a function that describes the local densities of vertices in a group) of appropriate candidate social groups according to the personalized local density requirements.

Conceptually, FSGSel-TD includes two major components:

i) **Generating bags with nice tree decomposition and building subproblems based on bags in a bottom-up fashion.** FSGSel-TD operates in a reverse topological order on a nice tree decomposition and carefully examines the subproblems with the four types of bags, i.e., {Leaf, Introduce, Forget, Join}, of the input graph. Specifically, for a *Leaf* bag, since the subgraph contains only one vertex without any children bags, FSGSel-TD directly decides whether to combine this subgraph with other subgraphs in its sibling subproblems. For *Introduce* and *Forget* bags with exactly one child bag, since the child bag of a *Forget* bag has one additional vertex, FSGSel-TD decides whether to include the additional vertex when combining the subgraphs in sibling subproblems. In contrast, since the child bag of an *Introduce* bag has fewer vertices, FSGSel-TD examines the subproblem in the child bag with a stricter density requirement to ensure the feasibility of the final solution. Finally, *Join* bags have two identical children bags, and FSGSel-TD combines the two subgraphs in two subproblems to maximize the objective value while satisfying the constraints.

ii) **Employing the notion of *partial local density profile* to systematically solve subproblems while satisfying the constraints.** FSGSel-TD leverages the notion of *partial local density profiles* (detailed below) to derive the optimal objective value of the subproblems generated above. Since the subproblem considers a subgraph of the original graph G, the local density constraint of the subproblem needs to be revised to ensure the feasibility of the candidate solutions after combining other subproblems.[18]

In the following, we first introduce *partial local density profile* to keep track of the local densities of the vertices in a tree bag.

Definition 5 (Partial local density profile)**.** For a tree bag B, a *partial local density profile* $\mathrm{P}_S : S \mapsto \mathcal{R}^+$ of a vertex subset $S \subseteq B$ is

[18] As an example, conceptually, observe that a $(k+1)$-clique C is a legitimate k-core, but any proper subgraph $\widehat{C} \subset C$ of the $(k+1)$-clique is not a k-core by definition, i.e., the degree of every vertex in $\widehat{C}$ must be smaller than k. Therefore, the partial local density profile is derived for each candidate solution to ensure the feasibility of the final solution. The partial density profile restricts $\mathrm{d}_H(u)$ of a subset S in a tree bag to be a fixed value based on the property of the subproblem. For the rest vertices in the tree bag, we then decide to keep or remove them to satisfy the local density constraint. FSGSel-TD carefully examines different partial local density profiles and subset S of each subproblem to gradually build the final solution with the *backtracking procedure*.

a function of S specifying the local density (defined on G) of *every* vertex in S.[19] Moreover, a subgroup $H \subseteq G$ is *consistent* with P_S if and only if i) $S \subseteq H$, i.e., every vertex specified in P_S is contained in H, and ii) for each vertex $v \in S$, the local density of v on H is exactly $\mathrm{P}_S(v)$, i.e., $\mathrm{d}_H(v) = \mathrm{P}_S(v), \forall v \in S$.

The partial local density profile also allows FSGSel-TD to explicitly keep track of the local density $\mathrm{d}_H(v)$ when the solution is gradually constructed. Conceptually, H is the solution we aim to extract, and by deriving the partial local density profile of S, we are able to gradually build the final solution from the candidate solutions without violating the local density constraint.

Note that DCSGQ does not enforce *every* vertex in a bag to satisfy its local density requirement; instead, only the vertices selected by the backtracking procedure (detailed later in this section) are required to meet the requirement, because it allows us to explore more candidate solutions to improve the solution. Moreover, to ensure the final solution to be feasible, i.e., satisfying all the personalized local density requirements, we further introduce the *feasible* partial local density profiles as follows.

Definition 6 (Feasible partial local density profile)**.** A partial local density profile P_S is *feasible* if all local densities specified in P_S satisfy the personalized local density constraints in DCSGQ.

Given a rooted nice tree decomposition T, for a tree bag $B_x \in \mathcal{B}_T$, let V_x be the union of all bags in the subtree rooted by B_x.[20] For a *feasible* P_S of $S \subseteq B_x$, let $\mathrm{W}_{\max}(B_x, S, \mathrm{P}_S)$ denote the maximum total vertex weight among a subgraph $H_x \subseteq V_x$ such that:

- $H_x \subseteq \mathrm{N}_G^h(v_0)$, *(distance constraint)*
- $\overline{d}(v) \geq \mathrm{d}_{H_x}(v) \geq \underline{d}(v), \forall v \in H_x$, *(feasibility)*
- H_x is consistent with P_S. *(consistency)*

Conceptually, when building a solution $H \subseteq G$, we exploit S to identify the vertices required to be included in H (consistent with P_S). In other words, (B_x, S, P_S) specifies a subproblem in FSGSel-TD to construct the final solution iteratively. For a vertex $v \in S$, P_S exerts a stricter constraint $\mathrm{d}_H(v)$ than the original local density constraint $[\underline{d}(v), \overline{d}(v)]$, because $\mathrm{d}_H(v)$ needs to be a fixed value (i.e., $\mathrm{P}_S(v)$).[21] Intuitively, $\mathrm{W}_{\max}(B_x, S, \mathrm{P}_S)$ captures the essential goal of DCSGQ and thus serves as the key subproblem in FSGSel-TD. When FSGSel-TD breaks the original problem into subproblems, FSGSel-TD specifies the partial local density profiles on the subgraphs generated to answer the subproblems. Because the inequality above, i.e., $\overline{d}(v) \geq \mathrm{d}_{H_x}(v) \geq \underline{d}(v), \forall v \in H_x$, ensures the feasibility of H_x (the solution to a subproblem on V_x), and different P_S are evaluated to ensure the feasibility of the final solution.

By leveraging the structural properties of T, and regarding (B_x, S, P_S) as a subproblem, FSGSel-TD recursively constructs candidate solutions (also the associated density profiles) according to the descendent bags in the tree decomposition. That is, when we specify an $S \subseteq B_x$ and a feasible partial local density profile P_S, $\mathrm{W}_{\max}(B_x, S, \mathrm{P}_S)$ is a feasible solution. To achieve the approximation ratio, it is necessary to carefully examine $S \subseteq B_x$ and the corresponding feasible partial local density profile P_S.

As bags in T are not necessarily disjoint, it is necessary for FSGSel-TD to carefully examine the overlapping vertices and monitor their local densities. Another key challenge is that a *feasible* local density profile may need to be decomposed into *infeasible* local density sub-profiles in order to achieve the performance bound. As a conceptual example, observe that a $(k+1)$-clique C is a legitimate k-core (aforementioned as a special case of DCSGQ), but any proper subgraph $\widehat{C} \subset C$ of the $(k+1)$-clique is not a k-core by definition, i.e., the degree of some vertex in $\widehat{C}$ must be smaller than k. Moreover, the local density $\mathrm{d}(v)$ of a vertex v may decrease to below the lower bound $\underline{d}(v)$ when H is decomposed into smaller subgraphs, and it is necessary for FSGSel-TD to examine these type of subgraphs. Therefore, FSGSel-TD also maintains and optimizes the subproblems with *infeasible* partial density profiles for the subqueries to construct the final optimal solution. For the subproblem not requiring feasibility, let $\mathrm{W}'_{\max}(B_x, S, \mathrm{P}_S)$ denote the maximum total vertex weight among a subgroup $H_x \subseteq V_x$ such that:

- $H_x \subseteq \mathrm{N}_G^h(v_0)$, *(distance constraint)*
- H_x is consistent with P_S, *(consistency)*
- The vertices in $H_x \setminus S$ satisfy the local density constraints, i.e., $\overline{d}(v) \geq \mathrm{d}(v) \geq \underline{d}(v), \forall v \in H_x \setminus S$. *(feasibility overrided by the partial profile)*

In other words, vertices in S may *violate* the local density constraints to comply with the (infeasible) P_S, but other vertices in $H_x \setminus S$ are still regulated by personalized local density constraints. Equipped with the above two types of subproblems, FSGSel-TD iteratively visits the bags in T to build the subgraph.

The objective of DCSGQ can be achieved by setting the parameters carefully, as indicated in the following observation.

Observation 1. Assume that the root bag B_r in the nice tree decomposition contains v_0 (i.e., the initiator).[22] Then, for each $S \subseteq B_r$ containing v_0 and each feasible P_S defined on S, $\mathrm{W}_{\max}(B_r, S, \mathrm{P}_S)$ corresponds to the objective of a feasible solution of DCSGQ, where all the vertices of $S \subseteq H$ are selected into H with local density consistent with P_S. Thus, the optimal objective of DCSGQ is $\max_{S, \mathrm{P}_S} \mathrm{W}_{\max}(B_r, S, \mathrm{P}_S)$.[23]

5.3 Derivation of Four Bag Types in FSGSel-TD

Equipped with the two types of subqueries, FSGSel-TD iteratively visits the bags in T to build up the subgraph. As the bags are of four major categories in a nice tree decomposition, it is necessary to derive different computation steps respective to i) different settings of local density metrics, and ii) different types of tree bags.

5.3.1 Leaf bag. For a leaf bag $B_x = \{v\}$, the only nonempty partial vertex set is $S = B_x = \{v\}$. As $V_x = \{v\}$ consists of only one vertex, the local density of any vertex in any subgraph of V_x is zero. Therefore, for the feasible-type subqueries, it suffices to check the feasibility of the following partial local density profile.

[19] An example of local density of each vertex $v \in S$ is its degree.
[20] As such, V_x is a union of (not necessarily disjoint) vertex sets, and thus is also a set of vertices. Note that this subtree includes B_x.
[21] Note that the two criteria do not conflict because P_S is required to be feasible.
[22] This can be guaranteed by always including the initiator in the root in the nice tree decomposition. For simplicity, here we assume that every tree bag $B \subseteq \mathrm{N}_G^h(v_0)$, i.e., vertices outside the feasible region are filtered out and not considered.
[23] Here P_S is required to be feasible. Note that the optimal subgroup H could correspond to multiple $\mathrm{W}_{\max}(B_r, S, \mathrm{P}_S)$; however, the optimal objective is unique as all such subproblems share the same value.

Case I: $\overline{d}(v) \geq 0 \geq \underline{d}(v)$, or equivalently, $\underline{d}(v) = 0$. In this case,

$$\begin{cases} \mathrm{W}_{\max}(B_x, B_x, \mathrm{P}_{B_x}) = \mathrm{W}'_{\max}(B_x, B_x, \mathrm{P}_{B_x}) = \mathrm{w}(v), & \mathrm{P}_{B_x} = 0 \\ \mathrm{W}_{\max}(B_x, B_x, \mathrm{P}_{B_x}) = \mathrm{W}'_{\max}(B_x, B_x, \mathrm{P}_{B_x}) = -\infty, & \text{otherwise.} \end{cases}$$

Here a value of $-\infty$ is given to infeasible subqueries.
Case II: $\underline{d}(v) > 0$**.** In this case, no profile is feasible. For the second type of subqueries, we have

$$\begin{cases} \mathrm{W}_{\max}(B_x, B_x, \mathrm{P}_{B_x}) = -\infty,\ \mathrm{W}'_{\max}(B_x, B_x, \mathrm{P}_{B_x}) = \mathrm{w}(v), & \mathrm{P}_{B_x} = 0 \\ \mathrm{W}_{\max}(B_x, B_x, \mathrm{P}_{B_x}) = \mathrm{W}'_{\max}(B_x, B_x, \mathrm{P}_{B_x}) = -\infty, & \text{otherwise.} \end{cases}$$

For the empty partial vertex set $S = \emptyset$, P_S does not take any input and is a null function. In this case, $\mathrm{W}_{\max}(B_x, \emptyset, \emptyset) = \mathrm{W}'_{\max}(B_x, \emptyset, \emptyset) = 0$.

5.3.2 Introduce bag. For an introduce bag B_x, let its single child bag be $B_y = B_x \setminus \{u\}$. A subgroup $S_x \subseteq B_x$ either contains u or not containing u. **Case I:** $u \notin S_x$**.** Then we have $S_x \subseteq B_y$, and for all possible partial local density profiles P_{S_x}, $\mathrm{W}_{\max}(B_x, S_x, \mathrm{P}_{S_x}) = \mathrm{W}_{\max}(B_y, S_x, \mathrm{P}_{S_x})$ and $\mathrm{W}'_{\max}(B_x, S_x, \mathrm{P}_{S_x}) = \mathrm{W}'_{\max}(B_y, S_x, \mathrm{P}_{S_x})$.
Case II: $u \in S_x$**.** From the property of tree decomposition,

$$\sum_{v \in S_x} \mathrm{f(u,v)} = \mathrm{P}_{S_x}(u)\ (u\text{'s degree is consistent with profile}). \tag{8}$$

To find $\mathrm{W}_{\max}(B_x, S_x, \mathrm{P}_{S_x})$, we need to ensure that P_{S_x} is feasible. For P_{S_x} to be a feasible profile, we check whether the following condition holds.

$$\overline{d}(v) \geq \mathrm{P}_{S_x}(v) \geq \underline{d}(v),\ \forall v \in S_x\ (\text{profile is feasible}) \tag{9}$$

- **Case II-(1): Condition (8) does not hold for** u**.** Then P_{S_x} is impossible (not consistent with S_x). In this case, we have $\mathrm{W}_{\max}(B_x, S_x, \mathrm{P}_{S_x}) = \mathrm{W}'_{\max}(B_x, S_x, \mathrm{P}_{S_x}) = -\infty$.

- **Case II-(2): Condition (8) holds for** u**, and Condition (9) holds (for all** $v \in S_x$**).** It means S_x is feasible. Let $S_y = S_x \setminus \{u\}$. We define another partial local density profile P_{S_y} on S_y as follows: $\mathrm{P}_{S_y}(v) = \mathrm{P}_{S_x}(v) - \mathrm{f}(v, u), \forall v \in S_y$. If there exists a feasible subgroup H consistent with P_{S_x}, by deleting vertex u from H, the resulting subgroup $H \setminus \{u\}$ must be consistent with P_{S_y}. Note that P_{S_y} might not be feasible since u is deleted from H. Fortunately, deleting u can only violate the core constraints for vertices in S_x, i.e., it can be proved by the properties of tree decomposition. Therefore, for any feasible H consistent with P_{S_x}, not only is $H \setminus \{u\}$ consistent with P_{S_y}, but all core constraints for vertices in $H \setminus S_x = (H \setminus \{u\}) \setminus S_y$ are satisfied. Thus, $\mathrm{W}_{\max}(B_x, S_x, \mathrm{P}_{S_x}) = \mathrm{W}'_{\max}(B_y, S_y, \mathrm{P}_{S_y}) + \mathrm{w}(u)$.

- **Case II-(3): Condition (8) holds for** u**, but Condition (9) does not hold (for some** $v \in S_x$**).** Since S_x is not a feasible partial profile, $\mathrm{W}_{\max}(B_x, S_x, \mathrm{P}_{S_x}) = -\infty$.

To find $\mathrm{W}'_{\max}(B_x, S_x, \mathrm{P}_{S_x})$, S_x is not required to be feasible. Thus, $\mathrm{W}'_{\max}(B_x, S_x, \mathrm{P}_{S_x}) = \mathrm{W}'_{\max}(B_y, S_y, \mathrm{P}_{S_y}) + \mathrm{w}(u)$ for both cases II-(2) and II-(3).

5.3.3 Forget bag. For a forget bag B_x, let its single child bag be $B_y = B_x \cup \{u\}$. Assume a subgraph $H \subseteq V_x$ is consistent with P_{S_x}. Then H either does not contain u or does indeed contain u. In the former case, H is consistent with P_{S_x} w.r.t. B_y (note that P_{S_x} is a valid partial local density profile in B_y because $S_x \subseteq B_x \subseteq B_y$). For the latter case, let $S_y = S_x \cup \{u\}$. Then i) $H \cap B_y = S_y$, and H should be consistent to some partial local density profile defined on S_y. ii) $\sum_{w \in H} \mathrm{f}(v, w) = \mathrm{P}_{S_x}(v)$, and the local densities specified P_{S_x} remain unchanged, since H is consistent with P_{S_x}. iii) $\sum_{v \in H} \mathrm{f}(u, v) \geq \sum_{v \in S_x} \mathrm{f}(u, v)$, since $S_x \subseteq H$. In other words, assume that H is consistent with some P_{S_y} in B_y, then $\mathrm{P}_{S_y}(v) = \mathrm{P}_{S_x}(v)\ \forall v \in S_x$, and P_{S_y} is decided solely by $\mathrm{P}_{S_y}(u)$. For convenience, we denote $\mathrm{P}_{S_y}^{d(u)}$ to describe the partial profile where u's local density is $d(u)$. More specifically, we have i) $\mathrm{P}_{S_y}^{d(u)}(v) = \mathrm{P}_{S_x}(v)$, $\forall v \in S_x$ and ii) $\mathrm{P}_{S_y}^{d(u)}(u) = d(u)$. Note that $d(u) \geq \sum_{v \in S_x} \mathrm{f}(u, v)$ (since these vertices are guaranteed to contribute to u's local density in H). Furthermore, for H to be feasible, we have $\underline{d}(u) \leq d(u) \leq \overline{d}(u)$. Therefore, let $\underline{d}'(u) = \max\{\sum_{v \in S_x} \mathrm{f}(u, v), \underline{d}(u)\}$, we have the following relations:

$$\begin{aligned} \mathrm{OPT}^{\mathrm{I}}_{\mathrm{W}_{\max}} &= \mathrm{W}_{\max}(B_y, S_x, \mathrm{P}_{S_x}) \\ \mathrm{OPT}^{\mathrm{II}}_{\mathrm{W}_{\max}} &= \max_{d(u) \in [\underline{d}'(u), \overline{d}(u)]} \mathrm{W}_{\max}(B_y, S_y, \mathrm{P}_{S_y}^{d(u)}) \\ \mathrm{W}_{\max}(B_x, S_x, \mathrm{P}_{S_x}) &= \max\{\mathrm{OPT}^{\mathrm{I}}_{\mathrm{W}_{\max}}, \mathrm{OPT}^{\mathrm{II}}_{\mathrm{W}_{\max}}\} \end{aligned}$$

To find $\mathrm{W}'_{\max}(B_x, S_x, \mathrm{P}_{S_x})$, it needs to identify some subgraph H that satisfies the CORE constraints for all vertices except for those in S_x. The local density for u thereby should fall between $\underline{d}(u)$ and $\overline{d}(u)$ if $u \in H$. Therefore, we have a similar set of relations:

$$\begin{aligned} \mathrm{OPT}^{\mathrm{I}}_{\mathrm{W}'_{\max}} &= \mathrm{W}'_{\max}(B_y, S_x, \mathrm{P}_{S_x}) \\ \mathrm{OPT}^{\mathrm{II}}_{\mathrm{W}'_{\max}} &= \max_{d(u) \in [\underline{d}'(u), \overline{d}(u)]} \mathrm{W}'_{\max}(B_y, S_y, \mathrm{P}_{S_y}^{d(u)}) \\ \mathrm{W}'_{\max}(B_x, S_x, \mathrm{P}_{S_x}) &= \max\{\mathrm{OPT}^{\mathrm{I}}_{\mathrm{W}'_{\max}}, \mathrm{OPT}^{\mathrm{II}}_{\mathrm{W}'_{\max}}\} \end{aligned}$$

5.3.4 Join bag. For a join bag B_x, let its two children bags be B_y and B_z. Recall that V_y and V_z denote the union of all tree bags in the subtrees rooted by B_y and B_z, and $V_x = V_y \cup V_z$. A key observation is as follows: if there exists vertices $y \in V_y$ and $z \in V_z$ such that $(y, z) \in E$, then at least one of y and z is contained in B_x. Otherwise, neither y nor z are contained in B_x. Recall that there exists at least one tree bag B containing both y and z (the first requirement of the tree decomposition). Let p_y denote the path from B to B_y in the tree decomposition, and let p_z denote the path from B to B_z similarly. Then at least one of p_y and p_z goes through B_x, which violates the second requirement of the tree decomposition that all bags containing y (or z) should form a connected subtree.

With the above observation, when merging two subgroups $H_y \subseteq V_y$ and $H_z \subseteq V_z$ such that $H_y \cap B_x = H_z \cap B_x = S_x$, we need to check only the local densities of vertices in B_x.

$$\sum_{v \in H_y \cup H_z} \mathrm{f}(u, v) = \sum_{v \in H_y} \mathrm{f}(u, v) + \sum_{v \in H_z} \mathrm{f}(u, v) - \sum_{v \in S_x} \mathrm{f}(u, v), \quad \forall u \in S_x.$$

Therefore, if H_y is consistent with some partial local density profile P'_{S_x} (in B_x), and H_z is consistent with some partial local density profile P''_{S_x} (in B_x), then P_{S_x} is consistent with the subgraph $H_y \cup H_z$ (in B_x) if and only if

$$\mathrm{P}_{S_x}(u) = \mathrm{P}'_{S_x}(u) + \mathrm{P}''_{S_x}(u) - \sum_{v \in S_x} \mathrm{f}(u, v), \quad \forall u \in S_x.$$

Let $\mathrm{P} = \mathrm{P}' + \mathrm{P}''$ represent that the above relation holds for all $u \in S_x$. To find $\mathrm{W}_{\max}(B_x, S_x, \mathrm{P}_{S_x})$, similar to that in *Introduce* bags, we first require P_{S_x} to be feasible as follows:

$$\overline{d}(v) \geq \mathrm{P}_{S_x}(v) \geq \underline{d}(v),\ \forall v \in S_x. \tag{10}$$

- **Case I: Condition (10) holds (for all $v \in S_x$).** Then $\mathrm{W}_{\max}(B_x, S_x, \mathrm{P}_{S_x}) = \max_{\mathrm{P}=\mathrm{P}'+\mathrm{P}''} \{\mathrm{W}'_{\max}(B_y, S_x, \mathrm{P}'_{S_x}) + \mathrm{W}'_{\max}(B_z, S_x, \mathrm{P}''_{S_x}) - \sum_{v \in S_x} \mathrm{w}(v)\}$.

- **Case II: Condition (10) does not hold (for some $u \in S_x$).** Then $\mathrm{W}_{\max}(B_x, S_x, \mathrm{P}_{S_x}) = -\infty$.

Again, similar to that in *Introduce* bags, to find $\mathrm{W}'_{\max}(B_x, S_x, \mathrm{P}_{S_x})$, S_x is not required to be feasible; thus, $\mathrm{W}'_{\max}(B_x, S_x, \mathrm{P}_{S_x}) = \max_{\mathrm{P}=\mathrm{P}'+\mathrm{P}''} \{\mathrm{W}'_{\max}(B_y, S_x, \mathrm{P}'_{S_x}) + \mathrm{W}'_{\max}(B_z, S_x, \mathrm{P}''_{S_x}) - \sum_{v \in S_x} \mathrm{w}(v)\}$ for both cases I and II.

5.4 FSGSel-TD and Performance Analysis

The pseudo code of FSGSel-TD is presented in Algorithm 2. FSGSel-TD includes a backtracking procedure, which retrieves the corresponding subgraph H after the optimal objective is found. Recall that we find the optimal objective $\max_{S,\mathrm{P}_S} \mathrm{W}_{\max}(B_r, S, \mathrm{P}_S)$ in the original query, where P_S needs to be feasible. Let S^* be the subset of B_r and $P^*_{S^*}$ be the local density profile that achieve the above objective, i.e., $\mathrm{W}_{\max}(B_r, S^*, \mathrm{P}^*_{S^*}) = \max_{S,\mathrm{P}_S} \mathrm{W}_{\max}(B_r, S, \mathrm{P}_S)$ for vertex subset S^* and some partial local density profile $\mathrm{P}^*_{S^*}$, where $S^* \subseteq H$. To retrieve other vertices in H, the backtracking procedure traces back the derivation of the optimal objective value by examining the four bag types, as summarized in Algorithm 2.

FSGSel-TD *discretizes* the edge weights $\mathrm{f}(u,v)$ to significantly reduce the computational overhead while introducing only a small bounded error. In other words, FSGSel-TD finds a solution with a good performance guarantee to strike a good balance between solution quality and efficiency as follows.

Theorem 2. ***(Performance bound and time complexity).*** *Given an error factor ϵ, let $\rho = \frac{\epsilon \underline{d}}{\Delta(G)}$, where $\Delta(G) = \max_{u \in V} deg_G(u)$ and $\underline{d} = \min_{u \in V} \underline{d}(u)$. Then,* FSGSel-TD *returns a solution $\tilde{H}$ such that i) $\sum_{v \in \tilde{H}} w(v) \geq (1-\epsilon) \cdot OPT$, where OPT is the objective value of the optimal solution, and ii) $(1+\epsilon) \cdot \overline{d}(v) \geq \mathrm{d}_H(v) \geq \underline{d}(v), \forall v \in \tilde{H}$, within $O(\omega |T| \cdot 2^{\omega} \cdot M^{2\omega})$ time,[24] where ω and $|T|$ are the treewidth and the number of tree nodes, respectively. Here, $M = Q + 1$ after discretizing the edge weights $f(u,v)$ into $Q+1$ different values.*

Proof. Please see [56] for details. □

Please note that the treewidth ω is usually small for social networks and is viewed as a *constant* [8, 9].

Algorithm 2 FSGSel-TD

Input: input graph $G = (V, E)$, tree decomposition T on G, initiator $v_0 \in V$, vertex weight $\mathrm{w}(u) \geq 0$, edge weight $\mathrm{f}(u,v) \geq 0$, hop count $h \in \mathcal{N}$, local density metric $\mathrm{d}(u)$ personalized local density thresholds $\overline{d}(u)$ and $\underline{d}(u)$, for all $u, v \in V$.
Output: A subgraph $\tilde{H}$

1: Let $T^h_{v_0} \subseteq T$ denote the nice tree decomposition on G w.r.t. $N^h_G(v_0)$; Pick bag B_r containing v_0 as the root bag
2: Traverse $T^h_{v_0}$ from B_r to retrieve a traversal order $\widehat{O}$; Apply *rounding-and-scaling* (detailed in [56])
3: For each bag B in the reversed order of O, compute $\mathrm{W}_{\max}(B, \cdot, \cdot)$ and $\mathrm{W}'_{\max}(B, \cdot, \cdot)$ values (detailed in [56])
4: Invoke the following Backtrack procedure for each tree bag B and its vertex subset $S_x \subseteq B_x$ to find solution $\tilde{H}$
5: **return** $\tilde{H}$
6:
7: **procedure** Backtrack(tree bag B_x, vertex subset $S_x \subseteq B_x$, partial local density profile P_{S_x} defined on S_x, type TYPE $\in \{\mathrm{W}_{\max}, \mathrm{W}'_{\max}\}$)
8: **if** B_x is *Leaf* **then**
9: **return** S_x
10: **if** B_x is *Introduce* with children B_y, where $B_x = B_y \cup \{u\}$ **then**
11: **if** $u \notin S_x$ **then**
12: **return** Backtrack(B_y,S_x,P_{S_x},TYPE)
13: **else**
14: Create S_y and P_{S_y} (Case II-(1) for *Introduce* bags [56])
15: **return** $\{u\} \cup$ Backtrack(B_y,S_y,P_{S_y},$\mathrm{W}'_{\max}$)
16: **if** B is *Forget* with children B_y, where $B_x = B_y \setminus \{u\}$ **then**
17: $S'_x \leftarrow S_x \cup \{u\}$
18: $S_{\mathrm{TYPE}}(B_x, S_x, \mathrm{P}_{S_x}) \leftarrow \mathrm{TYPE}(B_x, S_x, \mathrm{P}_{S_x}) - \mathrm{TYPE}(B_y, S'_x, \mathrm{P}_{S'_x})$
19: **if** $S_{\mathrm{TYPE}}(B_x, S_x, \mathrm{P}_{S_x}) \neq 0$ **then**
20: **return** Backtrack(B_y,S_x,P_{S_x},TYPE)
21: **else**
22: $d(u) \leftarrow \mathrm{P}_{S'_x}(u)$; Create S_y and $\mathrm{P}^{d(u)}_{S_y}$
23: **return** Backtrack(B_y,S_y,$\mathrm{P}^{d(u)}_{S_y}$,TYPE)
24: **if** B is *Join* with children B_y and B_z **then**
25: $\mathrm{P}' \leftarrow \arg\max_{\mathrm{P}_{S_x}} \mathrm{TYPE}(B_y, S_x, \mathrm{P}_{S_x})$
26: $\mathrm{P}'' \leftarrow \arg\max_{\mathrm{P}_{S_x}} \mathrm{TYPE}(B_z, S_x, \mathrm{P}_{S_x})$
27: **return** Backtrack(B_y,S_x,P',TYPE) ∪ Backtrack(B_z,S_x,P'',TYPE)
28: **return** H

6 HYBRID APPROACH (FSGSEL-HYBRID)

In the following, we present FSGSel-Hybrid to combine the merits of FSGSel-TD and FSGSel-RR. FSGSel-Hybrid is based on the unique *core-fringe* phenomenon [1, 11, 14, 46] of networks, i.e., many real-world networks exhibit a densely connected *core* (here, the core is not the mathematical notion of k-core) and multiple *fringes* with low treewidths. More specifically, given a width threshold parameter $\overline{\omega}$,[25] the core-fringe structure can be efficiently found by degree-based tree decomposition strategies, such as DEGREE and FILLIN [46]. This process, termed *partial tree decomposition*, creates a forest of bags with the width at most $\overline{\omega}$ interconnected via a dense core. Partial tree decomposition is also beneficial to a social network with small treewidth, because by controlling $\overline{\omega}$, we are able to reduce the width of the fringe, which allows the proposed FSGSel-Hybrid to gain better performance.

Our idea of FSGSel-Hybrid is to leverage the core-fringe structure to divide the initial problem into two parts: i) a *core subproblem*, focusing on the vertices in the dense core and exploiting the efficient FSGSel-RR; ii) a *fringe subproblem*, employing FSGSel-TD to find a good subgroup on the fringe tree structure. As FSGSel-RR is scalable on large graphs, and the widths of the fringe are significantly lower than the original treewidth, FSGSel-Hybrid is able to find better subgraphs than FSGSel-RR efficiently. However, it is important to effectively combine the subgraphs retrieved by FSGSel-RR and FSGSel-TD such that the personalized local density constraints on all vertices are satisfied. For the *boundary vertices*, i.e., vertices appearing in the dense core but having edges connecting to the fringe (and vice versa), it is necessary for FSGSel-Hybrid to carefully monitor their local densities to meet the constraints, and the details are presented in [56].

[24] Please note that the time complexity looks unrelated to ϵ at the first glance. However, as ϵ and M are inversely proportional, M (and time complexity) becomes larger with a smaller ϵ.
[25] We analyze different settings of $\overline{\omega}$ in [56].

Table 1: Summary of datasets.

Dataset	$\|V\|$	$\|E\|$	Avg. Deg.	CC
LiveJ	3,997,962	34,681,189	17.35	0.2843
Orkut	3,072,441	117,185,083	76.28	0.1666
Wiki	1,791,489	25,446,040	28.41	0.2746
Email	36,692	183,831	10.02	0.497
FB	4,039	88,234	43.7	0.61
Youtube	1,134,890	2,987,624	5.27	0.0808

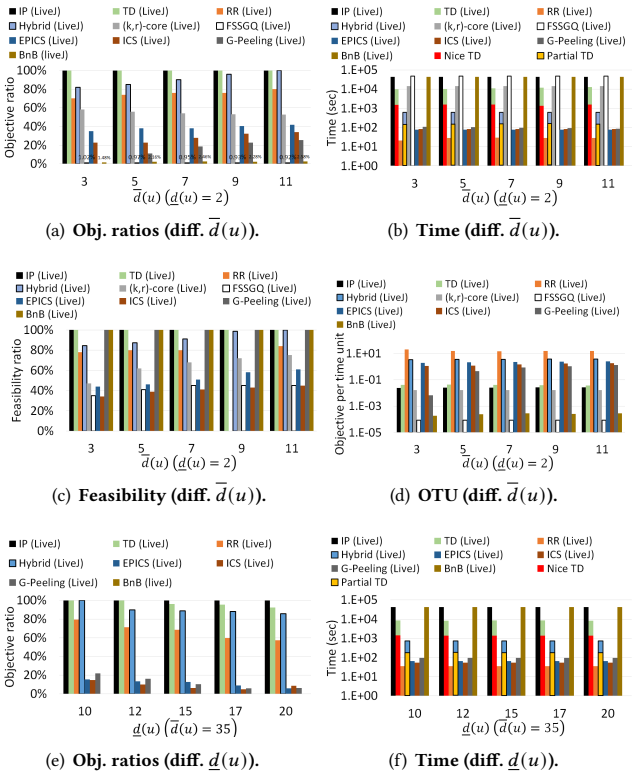
7 EXPERIMENTAL RESULTS

Datasets. We conduct experiments on a variety of real datasets. The datasets and the codes are available publicly [56]. *LiveJournal (LiveJ)* and *Orkut* are both online social networks with approximately 4M vertices with 34M edges and 3M vertices with 117M edges, respectively [71]. *Wiki* is extracted from the web graph of Wikipedia hyperlinks [72] and has approximately 1.8M vertices and 25M edges. *Email* is an e-mail social network with 36K vertices and 183K edges [39], and *Youtube* is a video sharing dataset with 1.1M vertices and approximately 3M edges [71]. Table 1 summarizes the characteristics of the above datasets.

Performance metrics. We compare the proposed algorithms with the baselines using the following performance metrics: i) *Objective value*; ii) *Objective ratio*: the ratio of the objective value to that of the optimal solution; iii) *Feasibility ratio*: the ratio of the number of vertices *satisfying the personalized local density requirement* to the total number of vertices for the returned subgraph; iv) *Computation time*. All the algorithms are implemented on an HP DL580 server with Quadcore Intel X5450 3.0 GHz CPUs and 1TB RAM. Each experimental result is averaged over 50 data samples.

Baselines. Since no existing algorithm was proposed for DCSGQ, we evaluate the proposed algorithms against three baselines: i) `IP`, the Integer Program detailed in Section 4. It obtains the optimal solution to DCSGQ, thus acting as a benchmark for DCSGQ; ii) (k, r)-`core` [74], aiming to identify the maximum subgraph by jointly considering the social tightness and similarity, which could be viewed as solving a subproblem of DCSGQ; iii) *Flexible Socio Spatial Group Query (*`FSSGQ`*)* [26], a flexible query that considers the group size, spatial closeness, and social connectivity simultaneously. Please note that by setting the vertex weight to 1, DCSGQ indeed maximizes the selected subgraph size while satisfying the social constraints. As `FSSGQ` considers similar factors, we chose it as a baseline for comparison;[26] iv) *Influential Community Search (*`ICS`*)* [41], a community search approach that extracts dense subgraphs, which considers the social density factor as DCSGQ does; v) *Personalized Influential Community Search (*`EPICS`*)* [66], an extension of `ICS` that requires the specified query vertex to be in the returned subgraph, which also considers the social factor as DCSGQ does; vi) `BnB`, a branch-and-bound approach that employs effective pruning and branching strategies, which acts as a benchmark approach here; vii) `G-Peeling`, a greedy approach that iteratively removes the vertex that violates the local density constraints, which is included to demonstrate that a trivial approach does not perform well for DCSGQ. For a fair comparison, these baselines are executed on the same $\mathrm{N}_G^h(v_0)$ as our approaches do.

[26]To achieve the best solution quality, we implemented the Exact approach [26].



(a) **Obj. ratios (diff. $\overline{d}(u)$).** (b) **Time (diff. $\overline{d}(u)$).**

(c) **Feasibility (diff. $\overline{d}(u)$).** (d) **OTU (diff. $\overline{d}(u)$).**

(e) **Obj. ratios (diff. $\underline{d}(u)$).** (f) **Time (diff. $\underline{d}(u)$).**

Figure 2: Comparisons with baselines in *LiveJ*.

7.1 Performance Evaluation

Comparisons with baseline approaches. We first compare various approaches on two large-scale datasets, *LiveJ* and *Orkut*, where $\mathrm{f}(u, v) = \mathrm{w}(v) = 1, \forall u, v \in V$. For (k, r)-`core`, we set $k = \underline{d}(u)$ since both k and $\underline{d}(u)$ are requirements on the minimum degree. Following [74], the similarity score of each pair of vertices (u, v) is assigned with the Jaccard similarity on their 1-hop neighbors, and r is set as the top-1% similarity value. For `FSSGQ`, we set its minimum acquaintance constraint c to require each selected user having at least $\underline{d}(u)$ friends in the group. We set weighting parameters as $\alpha = \gamma = 0.5$, and $\beta = 0$, to let `FSSGQ` focus on the social tightness and group size. For `BnB`, we let it run the same amount of time as `IP`. If `BnB` does not terminate when the time is up, `BnB` returns the best solution obtained so far as its solution. For `ICS`, we set $r = 1$ and $k = \underline{d}(u)$; For `EPICS`, we set $k = \underline{d}(u)$ and $q = v_0$.

Figures 2(a) and 2(e) present the objective ratios with different $\overline{d}(u)$ and $\underline{d}(u)$, and $h = \{1, 2\}$ in *LiveJ*. All the results in *Orkut* are with similar trends, and are presented in [56]. `IP` and `TD` achieve 100% objective ratios because `IP` is designed to extract the optimal solution, and `TD` is designed to achieve the guaranteed performance. The objective ratios of `Hybrid` is close to 100% in *LiveJ*, indicating that this proposed approach effectively obtains the solutions very close to the optimal ones. In contrast, (k, r)-`core` and `FSSGQ` both perform very poorly because their problem formulations do not aim to satisfy the personalized density requirements. `G-Peeling` and

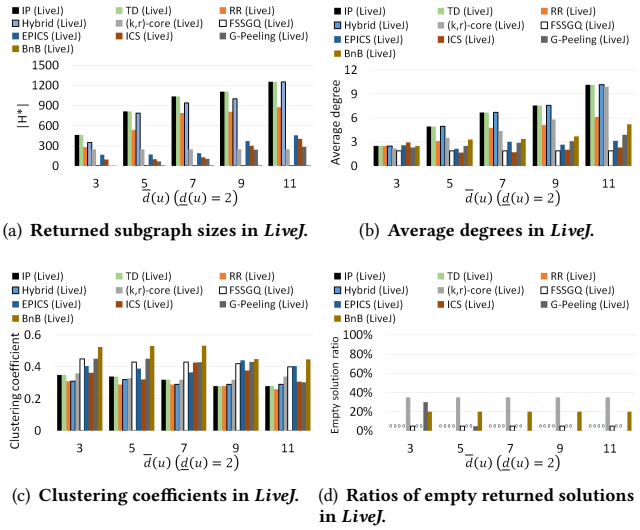
BnB both achieve poor solution quality. G-Peeling usually removes a very large number of extra vertices. When BnB uses up the time, it only explores a tiny portion of the solution space, resulting in very poor solutions. However, as the results of the approaches are averaged over multiple test cases, in some cases BnB indeed performs better than FSGSel-TD. The baselines ICS and EPICS also achieve very poor objective and feasibility ratios (as will be shown), because they do not consider the personalized local density and thus usually return infeasible solutions.

Figures 2(b) and 2(f) compare the efficiency. IP and TD achieve the perfect objective and feasibility ratios, while TD is much more computationally efficient because TD well leverages the notion of partial local density profile and tree decomposition to break the original problem into subproblems and effectively combine them. Hybrid achieves very high objective and feasibility ratios within a reasonable computation time, which achieves a good balance on efficiency and solution quality, much better than RR. The computation time of the baselines, (k,r)-core and FSSGQ is huge, because they need to examine different groups but do not leverage the structural information summarized by tree decomposition. G-Peeling is efficient, but its solution quality is also very poor. While the baselines ICS and EPICS incur more computation time than the proposed RR, their objective and feasibility ratios are much inferior to RR. Please note that the time complexity of G-Peeling is $O(|E|+|V|^2)$. For RR, by carefully controlling the sampled elements to be much smaller than $|E|$, the time complexity of RR does not exceed G-Peeling.

Figure 2(c) shows the feasibility ratios in *LiveJ* for various $\overline{d}(u)$. For varying $\underline{d}(u)$, the results show very similar trends and thus are presented in [56]. IP and TD both achieve 100% feasibility. Hybrid is able to achieve very high feasibility ratios, outperforming RR, because Hybrid combines the strengths of TD and RR to strike a good balance on efficiency and solution quality. In contrast, the baselines (k,r)-core and FSSGQ both show very poor feasibility because they do not consider the personalized density requirement $\overline{d}(u)$. G-Peeling and BnB both achieve 100% feasibility, because they are designed to return only feasible solutions. However, their poor solution quality makes them impractical for deployment in real-world scenarios. Figure 2(d) presents the *Objective per Time Unit (OTU)*, which is the objective value acquired per time unit. The proposed RR and Hybrid significantly outperform the other baselines. It indicates that these two approaches are able to acquire very good solutions efficiently.

Analyses of returned subgraphs. Figure 3 presents community-related measures of the resulting subgraphs in the dataset *LiveJ*, such as *subgraph size*, *average degree*, *clustering coefficient*, and *ratio of empty returned subgraphs*, i.e., the ratio of empty subgraphs that are returned by each approach. Additional results, i.e., *diameter* and the *quantity of distinct subgraphs*, are presented in the full version [56].

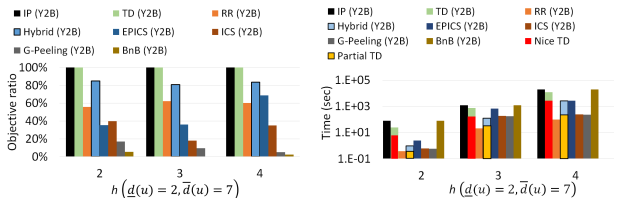
Figure 3(a) shows the returned subgraph sizes. The returned graph sizes of the proposed TD, RR, and Hybrid outperform the other baselines, and are close to the optimal solution IP. This is because these proposed approaches pay special attention to the local density requirements, enabling them to find large and feasible subgraphs. Figure 3(b) presents the average degree of the subgraphs. For IP, TD, RR, and Hybrid, the average degrees increase when $\overline{d}(u)$ grows. This



(a) **Returned subgraph sizes in *LiveJ*.** (b) **Average degrees in *LiveJ*.**

(c) **Clustering coefficients in *LiveJ*.** (d) **Ratios of empty returned solutions in *LiveJ*.**

Figure 3: Community scores in *LiveJ*.



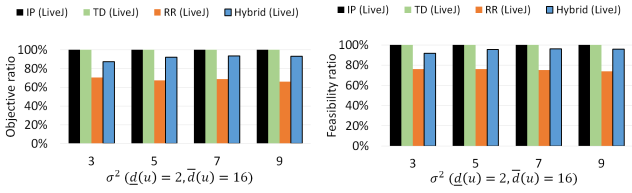
(a) **Obj. ratios in *Y2B*.** (b) **Time in *Y2B*.**

Figure 4: Diff. h in *Y2B*.

is because these approaches tend to find larger subgraphs in order to maximize the objective function. Figure 3(c) presents the clustering coefficients, where IP, TD, and Hybrid slightly decline when $\overline{d}(u)$ increases, because these three algorithms find larger subgraphs with a larger $\overline{d}(u)$. Figure 3(d) presents the ratios of solutions that are empty. The results show that the proposed approaches almost return no empty solution, indicating that they all are able to handle the personalized local density requirement well and extract feasible solutions.

Sensitivity tests: different h and tree decomposition. We conducted the experiments on the *Youtube (Y2B)* dataset with $h = \{2,3,4\}$. Here, we set $h = \{2,3,4\}$ because *Y2B* exhibits the small-world phenomenon, i.e., the 90-percentile effective diameter of *Y2B* is 6.5. Since the trends of feasibility ratios on *Y2B* are very similar to those on other datasets, we present them in [56]. As shown in Figures 4(a) and 4(b), the proposed Hybrid is able to obtain very good solutions that are close to the optimal ones. Also, the computation time of TD is similar to EPICS and ICS but outperforms BnB, a branch-and-bound approach. In contrast, the other baseline approaches all achieve very poor objective ratios. Please note that, since BnB is a branch-and-bound algorithm that examines the huge amount of combinations of the solutions, BnB

takes a large amount of time to return the solution when $h = \{3, 4\}$ (here, we set the time limit of BnB as 2 hours for $h = \{2\}$ and 24 hours for $h = \{3, 4\}$). In contrast, Hybrid is able to acquire much better solutions efficiently.



(a) **Obj. ratios with various $w(u)$.** (b) **Fea. ratios with various $w(u)$.**

Figure 5: Diff. $w(u)$ assignments.

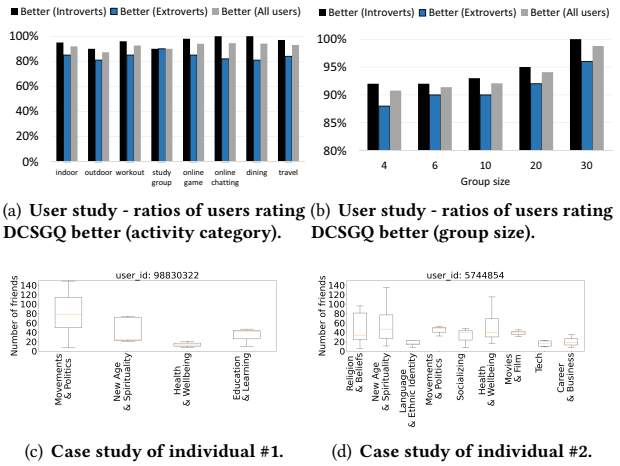
Sensitivity tests: $w(u)$ and $f(u, v)$ assignments. Figure 5 presents the results on various vertex weights, i.e., $w(u)$, on *LiveJ*. We present the results of varying $f(u, v)$ assignments in [56] since the results of varying $f(u, v)$ are very similar to those of $w(u)$. Following related works [4, 62], we assign the vertex weights (and edge weights) with Gaussian distribution with different variance values (σ^2). Figures 5(a) and 5(b) present the objective and feasibility ratios with different $w(u)$. Hybrid is very efficient and its feasibility ratios are all very close to 100%, indicating that it can effectively tackle the proposed DCSGQ under different vertex weight settings. We also observe that the feasibility ratios of Hybrid slightly increase when σ^2 grows. This is because a larger σ^2 results in more diverse local density requirements. In this case, Hybrid is able to find larger subgraphs that satisfy the density requirements.

7.2 User Study and Case Study

User study. We conduct a user study with 359 users to validate the proposed problem formulation with its key feature, *personalized local density constraint*. In this study, we aim to answer the following research questions by surveying users' opinions.

- **RQ1.** *Does the proposed DCSGQ with personalized local density constraint fit users' need better?*
- **RQ2.** *What are the preferred $\underline{d}(u)$ and $\overline{d}(u)$ values in different scenarios?*

To answer RQ1, we design 8 scenarios to ask for users' opinions, i.e., whether they consider setting a personalized local density constraint in an activity would better fit their needs. For each scenario, each user is asked to answer RQ1 with different group sizes, i.e., groups with 4, 6, 10, 20, and 30 users. Figures 6(a) and 6(b) present the results for RQ1. Figure 6(a) presents the ratios of users who consider DCSGQ is *better*. For the 8 categories, around 90% of users consider DCSGQ better because of its personalized local density constraint. Moreover, *introverts* (identified by the big five personality test) prefer DCSGQ more compared to *extroverts* because the introverts can freely adjust the social atmosphere in the activity. Figure 6(b) presents users' opinions with different group sizes. When the group size increases, the users desire the personalized local density more because people feel safe to participate in activities with friends, and the DCSGQ allows users flexibly adjust



(a) **User study - ratios of users rating DCSGQ better (activity category).** (b) **User study - ratios of users rating DCSGQ better (group size).**

(c) **Case study of individual #1.** (d) **Case study of individual #2.**

Figure 6: User study and case study results.

and personalize the number of friends in the activity. We provide more detailed discussions and the answers to RQ2 in [56].

Case study. Figures 6(c) and 6(d) show the ground-truth numbers of friends in various categories of groups for these two individuals in the large-scale real activity planning dataset *Meetup NY+SF* (the dataset is detailed in [56]). The results manifest that they both have quite diverse numbers of friends in these categories. Moreover, for the three categories, {*Movements & Politics*, *New Age & Spirituality*, *Health & Wellbeing*}, the individual in Figure 6(c) has median numbers of friends of 80, 23, and 16, respectively. In contrast, for the individual in Figure 6(d), these median numbers become 48, 43, and 39. It suggests that an individual may have various social tightness in different types of activities. We also provide more comparisons with our proposed DCSGQ and conventional dense subgraph queries in [56].

We present additional results of sensitivity tests in other datasets mentioned in Table 1, sensitivity tests for different λ and σ values, and treewidths of popular online social networks in [56].

8 CONCLUSION

We propose DCSGQ, a new query with personalized local density requirements. We show that DCSGQ is NP-hard and propose three effective algorithms from different perspectives and prove the guaranteed performance bound. Experimental results on multiple datasets demonstrate the effectiveness and efficiency of the proposed algorithms, which outperform the other baselines.

ACKNOWLEDGMENTS

This work was supported in part by the National Science and Technology Council (NSTC), Taiwan, under Grant NSTC 111-2636-E-007-019-.

REFERENCES

[1] A. B. Adcock et al. 2016. Tree decompositions and social graphs. *Internet Mathematics* 12, 5 (2016), 315–361.
[2] A. A. Ageev and M. Sviridenko. 2004. Pipage rounding: A new method of constructing algorithms with proven performance guarantee. *J. Comb. Optim.* 8,

3 (2004), 307–328.
[3] E. Akbas and P. Zhao. 2017. Truss-based community search: a truss-equivalence based indexing approach. *PVLDB* 10, 11 (2017), 1298–1309.
[4] A. Al-Baghdadi and X. Lian. 2020. Topic-based community search over spatial-social networks. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2104–2117.
[5] S. Arnborg, D. Corneil, and A. Proskurowski. 1987. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic Discrete Methods* 8, 2 (1987), 277–284.
[6] S. Arnborg and A. Proskurowski. 1989. Linear time algorithms for NP-hard problems restricted to partial k-trees. *Discret. Appl. Math.* 23, 1 (1989), 11–24.
[7] F. Bi et al. 2018. An optimal and progressive approach to online search of top-k influential communities. *PVLDB* 11, 9 (2018), 1056–1068.
[8] H. Bodlaender and A. Koster. 2008. Combinatorial optimization on graphs of bounded treewidth. *Comput. J.* 51, 3 (2008), 255–269.
[9] H. L. Bodlaender. 1988. Dynamic programming on graphs with bounded treewidth. In *ICALP*, Vol. 317. 105–118.
[10] H. L. Bodlaender. 1996. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.* 25, 6 (1996), 1305–1317.
[11] B. Bollobás and O. Riordan. 2003. Robustness and vulnerability of scale-free random graphs. *Internet Mathematics* 1, 1 (2003), 1–35.
[12] D. Boob, Y. Gao, R. Peng, S. Sawlani, C. Tsourakakis, D. Wang, and J. Wang. 2020. Flowless: Extracting densest subgraphs without flow computations. In *Proceedings of The Web Conference 2020*. 573–583.
[13] R. Burt. 2004. Structural holes and good ideas. *American journal of sociology* 110, 2 (2004), 349–399.
[14] D. S. Callaway et al. 2000. Network robustness and fragility: Percolation on random graphs. *Physical review letters* 85, 25 (2000), 5468.
[15] H.-W. Chen, H.-H. Shuai, D.-N. Yang, W.-C. Lee, C. Shi, P. Yu, and M.-S. Chen. 2021. Structure-aware parameter-free group query via heterogeneous information network transformer. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2075–2080.
[16] L. Chen, C. Liu, R. Zhou, J. Xu, J. Yu, and J. Li. 2020. Finding effective geo-social group for impromptu activities with diverse demands. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 698–708.
[17] Y. Chen et al. 2018. Exploring communities in large profiled graphs. *IEEE TKDE* 31, 8 (2018), 1624–1629.
[18] Y.-L. Chen, D.-N. Yang, C.-Y. Shen, W.-C. Lee, and M.-S. Chen. 2018. On efficient processing of group and subsequent queries for social activity planning. *IEEE Transactions on Knowledge and Data Engineering* 31, 12 (2018), 2364–2378.
[19] E. Chlamtac et al. 2012. Everywhere-sparse spanners via dense subgraphs. In *IEEE FOCS*. 758–767.
[20] J. Ding et al. 2018. Improving implicit recommender systems with view data. In *IJCAI*.
[21] Y. Fang et al. 2018. On spatial-aware community search. *IEEE TKDE* 31, 4 (2018), 783–798.
[22] Y. Fang et al. 2019. Efficient algorithms for densest subgraph discovery. *PVLDB* 12, 11 (2019), 1719–1732.
[23] Y. Fang, X. Huang, L. Qin, Y. Zhang, W. Zhang, R. Cheng, and X. Lin. 2020. A survey of community search over big graphs. *The VLDB Journal* 29, 1 (2020), 353–392.
[24] Y. Fang, Y. Yang, W. Zhang, X. Lin, and X. Cao. 2020. Effective and efficient community search over large heterogeneous information networks. *Proceedings of the VLDB Endowment* 13, 6 (2020), 854–867.
[25] P. Gartland, D. Lokshtanov, M. Pilipczuk, M. Pilipczuk, and P. Rzazewski. 2021. Finding large induced sparse subgraphs in c> t-free graphs in quasipolynomial time. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*. 330–341.
[26] B. Ghosh et al. 2018. The flexible socio spatial group queries. *PVLDB* 12, 2 (2018), 99–111.
[27] A. Gionis and C. E. Tsourakakis. 2015. Dense subgraph discovery: KDD 2015 tutorial. In *ACM SIGKDD*. 2313–2314.
[28] M. Granovetter. 1973. The strength of weak ties. *American journal of sociology* 78, 6 (1973), 1360–1380.
[29] F. Guo, Y. Yuan, G. Wang, L. Chen, X. Lian, and Z. Wang. 2019. Cohesive group nearest neighbor queries over road-social networks. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 434–445.
[30] F. Guo, Y. Yuan, G. Wang, X. Zhao, and H. Sun. 2021. Multi-attributed community search in road-social networks. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 109–120.
[31] B.-Y. Hsu, Y.-F. Lan, and C.-Y. Shen. 2018. On automatic formation of effective therapy groups in social networks. *IEEE Transactions on Computational Social Systems* 5, 3 (2018), 713–726.
[32] B.-Y. Hsu, C.-Y. Shen, and M.-Y. Chang. 2020. WMEgo: willingness maximization for ego network data extraction in online social networks. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 515–524.
[33] X. Huang et al. 2015. Approximate closest community search in networks. *PVLDB* 9, 4 (2015), 276–287.
[34] R. Jin et al. 2009. 3-HOP: a high-compression indexing scheme for reachability query. In *ACM SIGMOD*. 813–826.
[35] H. Kim, Y. Choi, K. Park, X. Lin, S.-H. Hong, and W.-S. Han. 2021. Versatile equivalences: Speeding up subgraph query processing and subgraph matching. In *Proceedings of the 2021 International Conference on Management of Data*. 925–937.
[36] J. Kim, T. Guo, K. Feng, G. Cong, A. Khan, and F. Choudhury. 2020. Densely connected user community and location cluster search in location-based social networks. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2199–2209.
[37] Y. Koren and J. Sill. 2013. Collaborative filtering on ordinal user feedback. In *IJCAI*.
[38] V. E. Lee et al. 2010. A survey of algorithms for dense subgraph discovery. In *Managing and Mining Graph Data*. Vol. 40. Springer, 303–336.
[39] J. Leskovec et al. 2009. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics* 6, 1 (2009), 29–123.
[40] Q. Li, Y. Zhu, and J. Yu. 2020. Skyline cohesive group queries in large road-social networks. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 397–408.
[41] R.-H. Li et al. 2015. Influential community search in large networks. *PVLDB* 8, 5 (2015), 509–520.
[42] Y. Liao, W. Lam, L. Bing, and X. Shen. 2018. Joint modeling of participant influence and latent topics for recommendation in event-based social networks. *ACM Transactions on Information Systems (TOIS)* 36, 3 (2018), 1–31.
[43] K.-P. Lin, Y.-W. Chang, C.-Y. Shen, and M.-C. Lin. 2018. Leveraging online word of mouth for personalized app recommendation. *IEEE Transactions on Computational Social Systems* 5, 4 (2018), 1061–1070.
[44] Q. Liu, Y. Zhu, M. Zhao, X. Huang, J. Xu, and Y. Gao. 2020. VAC: vertex-centric attributed community search. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 937–948.
[45] C. Ma, Y. Fang, R. Cheng, L. Lakshmanan, W. Zhang, and X. Lin. 2020. Efficient algorithms for densest subgraph discovery on large directed graphs. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1051–1066.
[46] S. Maniu et al. 2019. An experimental study of the treewidth of real-world graph data. In *ICDT*, Vol. 127. 12:1–12:18.
[47] K. Meeks and A. Scott. 2011. The parameterised complexity of list problems on graphs of bounded treewidth. *arXiv preprint arXiv:1110.4077* (2011).
[48] H. Nagamochi and T. Ibaraki. 1992. A linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph. *Algorithmica* 7, 1-6 (1992), 583–596.
[49] N. Robertson and P. D. Seymour. 1991. Graph minors. X. Obstructions to tree-decomposition. *J. Comb. Theory* 52, 2 (1991), 153–190.
[50] A. Sarıyüce. 2021. Motif-driven dense subgraph discovery in directed and labeled networks. In *Proceedings of the Web Conference 2021*. 379–390.
[51] C.-Y. Shen et al. 2015. Forming online support groups for internet and behavior related addictions. In *ACM CIKM*. 163–172.
[52] C.-Y. Shen et al. 2015. Socio-spatial group queries for impromptu activity planning. *IEEE TKDE* 28, 1 (2015), 196–210.
[53] C.-Y. Shen et al. 2017. On finding socially tenuous groups for online social networks. In *ACM SIGKDD*. 415–424.
[54] C.-Y. Shen et al. 2020. Activity organization for friend-making optimization in online social networks. *IEEE TKDE* (2020).
[55] C.-Y. Shen, C. Fotsing, D.-N. Yang, Y.-S. Chen, and W.-C. Lee. 2018. On organizing online soirees with live multi-streaming. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.
[56] C.-Y. Shen, S.-H. Ko, G.-S. Lee, W.-C. Lee, and D.-N. Yang. 2022. Density personalized group query - online full version. https://bit.ly/36QhrvV.
[57] C.-Y. Shen, D.-N. Yang, W.-C. Lee, and M.-S. Chen. 2016. Spatial-proximity optimization for rapid task group deployment. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 10, 4 (2016), 1–36.
[58] S. Sun, X. Sun, Y. Che, Q. Luo, and B. He. 2020. RapidMatch: a holistic approach to subgraph query processing. *Proceedings of the VLDB Endowment* 14, 2 (2020), 176–188.
[59] C. E. Tsourakakis. 2015. The k-clique densest subgraph problem. In *ACM WWW*. 1122–1132.
[60] C. E. Tsourakakis et al. 2013. Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. In *ACM SIGKDD*. 104–112.
[61] K. Wang et al. 2018. Efficient computing of radius-bounded k-cores. In *IEEE ICDE*. 233–244.
[62] K. Wang, W. Zhang, X. Lin, Y. Zhang, L. Qin, and Y. Zhang. 2021. Efficient and effective community search on large-scale bipartite graphs. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 85–96.
[63] W. Wang, H. Yin, X. Du, W. Hua, Y. Li, and Q. Nguyen. 2019. Online user representation learning across heterogeneous social networks. In *Proceedings*

of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval. 545–554.

[64] Y. Wang et al. 2017. Query optimal k-plex based community in graphs. *Data Science and Engineering* 2, 4 (2017), 257–273.

[65] K. Wright and C. Miller. 2010. A measure of weak-tie/strong-tie support network preference. *Communication Monographs* 77, 4 (2010), 500–517.

[66] Y. Wu, J. Zhao, R. Sun, C. Chen, and X. Wang. 2021. Efficient personalized influential community search in large networks. *Data Science and Engineering* 6, 3 (2021), 310–322.

[67] C.-H. Yang and C.-Y. Shen. 2022. Enhancing Machine Learning Approaches for Graph Optimization Problems with Diversifying Graph Augmentation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2191–2201.

[68] Chen-Hsu Yang, Hong-Han Shuai, Chih-Ya Shen, and Ming-Syan Chen. 2021. Learning to solve task-optimized group search for social internet of things. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[69] D.-N. Yang et al. 2012. On socio-spatial group query for location-based social networks. In *ACM SIGKDD*. 949–957.

[70] D.-N. Yang, Y.-L. Chen, W.-C. Lee, and M.-S. Chen. 2011. On social-temporal group query with acquaintance constraint. *Proceedings of the VLDB Endowment* 4, 6 (2011).

[71] J. Yang and J. Leskovec. 2015. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems* 42, 1 (2015), 181–213.

[72] H. Yin et al. 2017. Local higher-order graph clustering. In *ACM SIGKDD*. 555–564.

[73] D. Zhang, J. Yin, X. Zhu, and C. Zhang. 2018. Network representation learning: A survey. *IEEE Transactions on Big Data* 6, 1 (2018), 3–28.

[74] F. Zhang et al. 2017. When engagement meets similarity: efficient (k, r)-core computation on social networks. *PVLDB* 10, 10 (2017), 998–1009.

[75] T. Zhao, J. McAuley, and I. King. 2014. Leveraging social connections to improve personalized ranking for collaborative filtering. In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*. 261–270.

[76] Q. Zhu et al. 2017. Geo-social group queries with minimum acquaintance constraints. *VLDBJ* 26, 5 (2017), 709–727.