Social-Spatial Group Queries with Keywords

SAJID HASAN APON, Purdue University, USA
MOHAMMED EUNUS ALI, Bangladesh University of Engineering and Technology, Bangladesh
BISHWAMITTRA GHOSH, National University of Singapore, Singapore
TIMOS SELLIS, Swinburne University of Technology, Australia

Social networks with location enabling technologies, also known as geo-social networks, allow users to share their location-specific activities and preferences through check-ins. A user in such a geo-social network can be attributed to an associated location (spatial), her preferences as keywords (textual), and the connectivity (social) with her friends. The fusion of social, spatial, and textual data of a large number of users in these networks provide an interesting insight for finding meaningful geo-social groups of users supporting many real-life applications, including activity planning and recommendation systems. In this article, we introduce a novel query, namely, Top-k Flexible Socio-Spatial Keyword-aware Group Query (SSKGQ), which finds the best k groups of *varying sizes* around different points of interest (POIs), where the groups are ranked based on the social and textual cohesiveness among members and spatial closeness with the corresponding POI and the number of members in the group. We develop an efficient approach to solve the SSKGQ problem based on our theoretical upper bounds on distance, social connectivity, and textual similarity. We prove that the SSKGQ problem is NP-Hard and provide an approximate solution based on our derived relaxed bounds, which run much faster than the exact approach by sacrificing the group quality slightly. Our extensive experiments on real data sets show the effectiveness of our approaches in different real-life settings.

CCS Concepts: • **Information systems** → **Information retrieval query processing**; *Location-based services;*

Additional Key Words and Phrases: Geo-social networks, spatial database, group query, group query with keywords, spatial networks with keywords

ACM Reference format:
Sajid Hasan Apon, Mohammed Eunus Ali, Bishwamittra Ghosh, and Timos Sellis. 2021. Social-Spatial Group Queries with Keywords. *ACM Trans. Spatial Algorithms Syst.* 8, 1, Article 1 (October 2021), 32 pages.
https://doi.org/10.1145/3475962

1 INTRODUCTION

With the recent advancement of mobile and location acquisition technologies, users in all major social network sites are now able to share their locations and activities through check-ins. These

Sajid Hasan Apon work done while the author was a student at Bangladesh University of Engineering and Technology.
Authors' addresses: S. H. Apon, Purdue University, West Lafayette, Indiana, USA; email: sapon@purdue.edu; M. E. Ali, Bangladesh University of Engineering and Technology, Bangladesh; email: eunus@cse.buet.ac.bd; B. Ghosh, National University of Singapore, Singapore; T. Sellis, Swinburne University of Technology, Australia.

2374-0353/2021/10-ART1 $15.00
https://doi.org/10.1145/3475962

social networks with location-enabled features are commonly known as location-based social networks or geo-social networks, where along with the connectivity with friends or acquaintances, each user has an associated location. Each user in such geo-social networks also has textual attributes representing her preferences or expertise derived from the posts and activities. The fusion of social (connectivity), spatial (location), and textual (preferences) data in such social network users gives an immense opportunity to find interesting and meaningful groups of users that can support real-life applications such as activity planning [18, 32], ride-sharing [9], travel recommendation, and product promotion.

Existing works on group queries in geo-social networks mainly focus on finding the *best* group of users in terms of social connectivity among members and spatial closeness w.r.t. single or multiple **points of interest (POIs)** [14, 32]. More specifically, Yang et al. [32] find the best group of users of *fixed size* who has the highest combined score in terms of social connectivity among members and spatial closeness to a given POI. Recently, Ghosh et al. [14] offered the flexibility of processing socio-spatial group queries while finding the highest-scoring groups of different sizes w.r.t. multiple POIs, where the score of a group is computed as a combination of group size, social connectivity among members, and spatial closeness of members w.r.t. the corresponding POI.

The main drawback of the above query construction [14, 32] is that the group cohesiveness is measured *only* based on the social connectivity among members and the spatial closeness of members to the meeting point, which may fail to capture useful information such as preferences of group members, which is a key to supporting many in real-life applications. For instance, the decision-making of a holiday dinner at a restaurant among a group of friends depends on an individual's choice of foods and the availability of similar foods in the restaurant. In another instance, a pair of socially disconnected individuals may be interested in visiting the same restaurant based on their intersection of common favorite foods. In such examples, the set of favorite foods are the preferences (represented as keywords) of an individual and the set of available foods are the keywords of the host restaurant. Considering the above contexts, *keyword-match* between a pair of individual members adds *another dimension to the social ties (cohesiveness)* in addition to the social connection (e.g., friendship), hence is expected to be maximized while ranking different groups. However, *keyword mismatch* between a member and a POI can act as *another form of distance measure* in addition to the spatial distance, which has to be minimized while searching for the best group. Therefore, in this article, in addition to traditional social connectivity and spatial distance, we consider keyword match as an important attribute while scoring the groups in terms of cohesiveness among members, and the distance between members and the meeting point. Intuitively, in such a socio-spatial-keyword group formulation, a high scoring group exhibits a higher keyword match between each pair of members, and between members and the POI, while at the same time maintains tighter social connectivity among members and close spatial proximity to the corresponding POI.

Based on the above observation, in this article, we introduce a new type of group query, namely, the flexible **Socio-Spatial Keyword Aware Group Query (SSKGQ)**, which can be formally defined as follows. Given a socio-spatial graph of socially connected users with each user having a set of keywords denoting their preferences or expertise, a set of POIs with each POI also having a set of keywords describing the POI attributes, the minimum social connectivity constraint c of the group, the maximum allowable spatial distance d_{max} of a user for the meetup, the minimum n_{min} and the maximum group size n_{max}, and the number of desired groups k, the top-k-SSKGQ returns the top-most k groups ranked using a score function (discussed in Section 3), where each group is associated with a POI, and members of the group satisfy the required constraints.

A Motivating Example: Let us consider the following scenario: the owner of a restaurant chain wants to find the k best groups of users to announce *special-discount* offers for groups of attendees

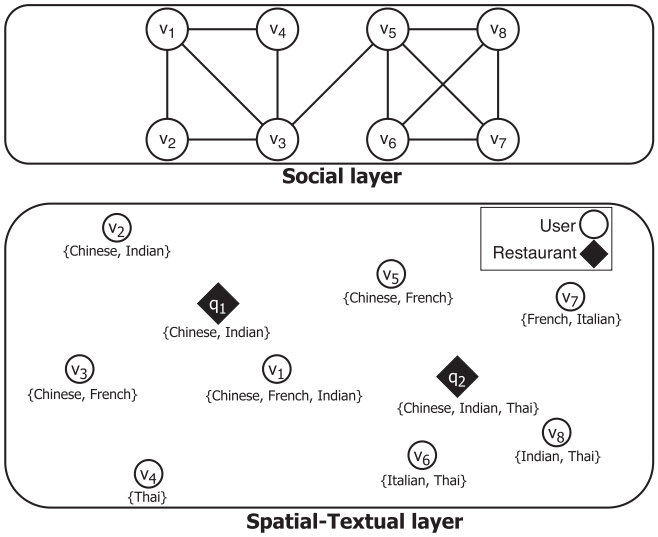


Fig. 1. A set of users $V = \{v_1, \ldots, v_8\}$ and a set of POIs $Q = \{q_1, q_2\}$.

Table 1. Distance Between Members in V and POIs in Q

Member	q_1	q_2
v_1	2	12
v_2	4	11
v_3	5	8
v_4	7	5
v_5	5	3
v_6	8	2
v_7	10	4
v_8	11	3

at different branches of the facility. The intention of the restaurant owner is to identify groups of friends that are most likely to take the offers and maximize the profit.

For illustration, we consider a socio-spatial network with keywords in Figure 1, where the upper rectangle represents the social layer and the lower rectangle represents the spatio-textual layer. Here $V = \{v_1, v_2, \ldots, v_8\}$ represents a set of users in the given graph, where every edge in the social layer represents the social connectivity between two individuals, and $Q = \{q_1, q_2\}$ is the location set of restaurants (meeting points) in the spatio-textual layer.

Suppose that the restaurant owner is interested in finding the best two groups of size three or four, where no member in the group is located more than 5 units of distance away from the meeting point. The resulting group must satisfy a minimum acquaintance constraint $c = 2$, that is, each person is socially connected with at least two other people in a group. We see that group $G_1 = \{v_1, v_3, v_5\}$ is not a feasible group as it does not comply with the minimum acquaintance constraint. Group $G_2 = \{v_1, v_2, v_3, v_4\}$ satisfies the minimum acquaintance constraint but there is no restaurant that is located at a convenient distance to *every* member in this group. Specifically, q_1 is located too far away from v_4, and q_2 is located too far away from v_1, v_2, and v_3. Thus, G_2 cannot be a good group either.

Now, if we look at $G_3 = \{v_1, v_2, v_3\}$ and $G_4 = \{v_5, v_6, v_7, v_8\}$, then we see that both groups the satisfy familiarity constraint, and the members of G_3 and G_4 are located at a convenient distance from q_1 and q_2, respectively. We observe that the members in G_3 have common food-preferences while the members in G_4 have varied preferences. Therefore, the members in G_3 are comparatively much *closer* to one another in terms of keyword-match among members and our score function gives a higher inter-member keyword-similarity score to this group (0.56 of 1.0) compared to that of G_4 (0.17). Also, the average keyword similarity between the members in G_3 with q_1 is higher than that of the members in G_4 and q_2 leading to a *member-POI keyword score* of 0.67 (of maximum 1.0) in $\langle G_3, q_1 \rangle$ and 0.354 in $\langle G_4, q_2 \rangle$.

Based on our social, spatial, and textual combined scores, two groups $[\langle G_3, q_1 \rangle, \langle G_4, q_2 \rangle]$ are returned as the result of the restaurant owner's query. Here, according to social, spatial, and textual criteria, G_3 outranks G_4 despite having an equal social score, almost similar spatial distance

score, and a lower *size-score* (because of the smaller size). The detailed method of calculating the scores is discussed in Section 3.2, and the scores of all the feasible groups are provided in Table 3, where we can see that $\langle\{v_1, v_2, v_3\}, q_1\rangle$ is the best combination, followed by $\langle\{v_5, v_6, v_7, v_8\}, q_2\rangle$. A straightforward approach to solve the Top-k SSKGQ is to enumerate all feasible groups of allowable group-size and evaluate them against each POI using the score function. This combinatorial approach incurs high query processing costs and hence it is not suitable for practical applications. Earlier approaches such as those proposed in References [14, 32] for socio-spatial group queries do not consider the textual dimension and hence their connectivity and distance-based pruning methods cannot be applied directly to process SSKGQ efficiently. In particular textual inclusion invalidates all bounds and pruning required for efficient processing of SSKGQ, and hence we need to devise the bounds for effective pruning and termination strategies in the search process for SSKGQ by including all involving factors of queries.

To answer the Top-k SSKGQ efficiently, we first propose an *exact* algorithm based on the *branch-and-bound* technique. The key ideas of our approach are (1) to derive *theoretical bounds* on the scores of candidate solution groups to effectively prune the search space and (2) to devise an efficient ordering scheme of the users utilizing the spatial, textual (keyword), and social properties with respect to a POI to retrieve the *potential* groups early on to avoid evaluating a large number of candidate groups that cannot be included in the final solution list. We adapt the spatial-textual index IR-Tree [22], which allows searches with different weights on textual and spatial domains at run time to facilitate our strategy for socio-spatial-keyword ordering for retrieving the members sequentially. We prove that the SSKGQ is NP-Hard, and thus, to process the SSKGQ in large geo-social networks, we propose an approximate algorithm that runs significantly faster than our proposed exact solution by sacrificing the accuracy slightly. Unlike the exact algorithm in which we independently consider each member in the network for group formation, the approximate algorithm utilizes a heuristic that can include multiple members in a single pass. This enables the approximate algorithm to execute even faster than the exact algorithm, and the trade-off is the possible retrieval of sub-optimal groups in certain cases. We also derive theoretical estimations for the runtime of the approximate algorithm and the worse-case performance guarantee. On real datasets, both the exact algorithm and the approximate algorithm outperform the baseline algorithm based on Reference [32]. The exact method executes *two orders* of magnitude faster and the approximate algorithm executes *three orders* of magnitude faster in networks with around 50K to 200K users (Section 7). The relative *goodness*, i.e., the quality of the solution provided by the approximate solution compared to the optimal solution is also presented in our experimental evaluation (Section 6).

The contribution of this article can be summarized as follows.

- We introduce two new measures for group cohesiveness: textual preference similarity among members, and textual (di-)similarity between members and the POI. Based on these new textual cohesiveness measures and traditional socio-spatial measures, we propose a novel socio-spatio-textual group query that finds interesting groups w.r.t. different POIs from a given geo-social network.
- We develop theoretical bounds combining social, spatial, and textual attributes of members and POIs, which enable us to propose an efficient algorithm for solving the SSKGQ.
- We prove that the SSKGQ is NP-Hard, and propose an approximation algorithm that runs significantly faster than our proposed exact algorithm by sacrificing the group quality slightly.
- We conduct extensive experiments on real datasets to evaluate the performance of our proposed algorithms and empirically show performance improvement over existing methods.

2 RELATED WORKS

Group queries in social networks have been studied extensively covering various aspects of group attributes that include social connectivity, spatial distance, and keyword similarity. We discuss the recent related works in the following subsections.

2.1 Social Query Processing

Social connectivity-based group queries find a user-group with certain social relationships. A social group is a cohesive subgraph that is formed by users with acquaintance relations. Group and team queries have been studied in the context of social networks [26, 34], including socio-temporal queries [31] and expert collaboration queries [18, 19]. Yang et al. [31] introduce SGSelect and STGSelect to effectively prune search space wherein they employ the idea of pivot time slots to find a group with minimum total social distance in a socio-temporal group query. Chen et al. [7] focus on socio-temporal group queries and propose the TGQ query by highlighting the historical view in the context of group forming and changing. Lappas et al. [18] and Li et al. [19] study the problem of finding a group of experts in a social graph with required skills while minimizing communication costs within that group.

2.2 Spatial Group Query Processing

Spatial group queries find a group of points while keeping the aggregate spatial distance to a minimum. Earlier, the importance of the spatial factor in group formation was studied by Barthélemy [2]. R-tree and its variants have been extensively used in the study of spatial query processing over the past years in processing different forms of spatial group queries.

Roussopoulos et al. [24] propose a branch-and-bound algorithm to search the **nearest-neighbor (NN)** object to a query point by traversing an R-tree. They further extend the algorithm for k-nearest neighbors. Katayama et al. [15] propose a similar work based on SR-tree. Papadias et al. [23] exploit the concept of NN query and find a point with the smallest aggregate spatial distances to a set of query points. Li et al. [21] proposed a **flexible aggregate nearest neighbor (FANN)** algorithm that finds the nearest data object of a fixed subgroup size. Ali et al. [1] propose a consensus query, which finds a POI that minimizes the travel distance for at least a specified number of groups.

2.3 Socio-Spatial Query Processing

A recent body of work in group queries merges both spatial and social dimensions in query processing. Li et al. [20] propose a **social-aware ridesharing group (SaRG)** query, which retrieves a group of riders by taking into account their social connections and spatial proximity. Shi et al. [25] show how a density-based clustering paradigm can be extended to apply to places that are visited by users in a geosocial network.

Zhu et al. [33] propose a **geo-social group query with minimum acquaintance constraint (GSGQs)** to guarantee the worst-case per user acquaintance in a group. They devise two socio-spatial index structures, namely, SaR-tree and SaR*-tree. The latter improves on the former by considering both spatial and social distances while clustering objects.

Yang et al. [32] propose a **socio-spatial group query (SSGQ)** to select a group of nearby attendees with tight social relations. A similar study by Ghosh et al. [14] finds the top-k socio-spatial groups where the size of a group is flexible and members maintain a minimum social acquaintance within a group.

2.4 Spatial Keyword Query Processing

Spatial-keyword queries exploit both location and textual descriptions in group formation. This body of work finds objects that are near to a specified location with a set of specific keywords. In the simplest form, a spatial keyword query takes a user location and user-supplied keywords as arguments and returns objects that are spatially and textually relevant to these arguments. These queries can be extended in finding groups of similar users/objects. Felipe et al. [10] present an efficient method to answer top-k spatial keyword queries by introducing an indexing structure called Information Retrieval R-Tree, which combines an R-Tree with superimposed text signatures. Chen et al. [5] discuss various indexing techniques for spatial keyword query processing and compare them. Reference [4] focuses on web querying with local intent, covering different kinds of functionality as well as the ideas underlying their definition.

2.5 Limitations of the Existing Studies

None of the existing work in the literature considers all four key factors, i.e., social connectivity, spatial proximity, keyword-matching, and flexible group size together, while processing group queries with respect to a set of points of interest in the geo-social graph. For example, References [6, 12, 17] are among the many other works that process group queries based on social connectivity and spatial proximity; however, they overlooked keywords. However, Reference [13] does consider keywords but the resulting groups are of a predefined fixed size, and the groups are not formed around meeting points of the querist's preference. To fill up the above research gaps, we consider all four factors that can support many novel applications.

Although the methods in References [14, 32] seem apparently closely related to those of ours in this article, the direct extension of their approaches to process SSKQG is either too inefficient to be applied in practical usage or inapplicable. This is proved by the performance of our baseline model outlined in Section 4.2. Reference [14] uses R-Tree for indexing but that is inadequate for our cause, since we incorporate keywords, and hence we use IR-Tree. Our ordering technique is also unique, since we combine both spatial and keyword factors in the ordering. Moreover, we use a two-step approach to build the intermediate solution group as opposed to the straightforward distance ordering in Reference [14], and our process for the inclusion of a member in the intermediate solution group is more complex and depends on multiple factors. However, Reference [32] only considers groups of fixed size and has no provision for incorporating keyword match, and hence, unique ordering and search techniques need to be devised to process SSKQG. Even though Reference [32] proposed a technique utilizing *intermediate solution groups*, the process of incrementally forming the solution group is different from ours. Reference [32] iteratively merges smaller clusters to arrive at a solution, whereas we inspect the unexplored members by following a distinctive ordering technique and include one (in the *exact* algorithm) or multiple (in the *fast approximate* algorithm) at a time, and then backtrack.

3 PROBLEM FORMULATION

3.1 Notations

Let $G = (V, E)$ be a socio-spatial-textual network (graph), where V is a set of users (vertices) and E is a set of connections (edges) among users. Each edge $\{u, v\} \in E$ represents a social connection between two members $u, v \in V$. Each user $v \in V$ is defined as a pair $(v.\lambda, v.\psi)$ where $v.\lambda$ is the location point of v and $v.\psi$ is a set of keywords associated with member v.

Let Q be a spatial-textual dataset that represents a set of candidate meeting points, i.e., POIs. Each object $q \in Q$ is a pair $(q.\lambda, q.\psi)$, where $q.\lambda$ is the location of q and $q.\psi$ is a set of keywords associated with q.

Table 2. Description of Notations

Notation	Definition
G	the socio-spatial-textual graph of members
V	the set of all members in G
E	the set of all edges (i.e., social connections) in G
$v.\lambda$	the location of a member $v \in V$
$v.\psi$	the set of keywords of a member $v \in V$
$q.\lambda$	the location of a POI $q \in Q$
$q.\psi$	the set of keywords of a POI $q \in Q$
$DIST(v.\lambda, q.\lambda)$	Spatial distance of member v to meeting point q
n_{min}	minimum group size
n_{max}	maximum group size
c	minimum acquaintance constraint
d_{max}	maximum allowable distance between a member and a POI
L	Rank list
k	Number of entries in rank-list

In addition, we consider several application-specific constraints to define our queries: minimum group size n_{min}, maximum group size n_{max}, minimum acquaintance constraint c, and maximum spatial range d_{max}. In particular, the size of a solution group must be between n_{min} and n_{max}, and each member in the group must be socially connected with at least c other members, the spatial distance of a member in the solution group to the corresponding POI is at most d_{max}. The notations are summarized in Table 2.

3.2 Problem Statement

Based on the above notations, we now formally define the **Top-k Flexible Social-Spatial-Keyword Aware Group Query (Top-k SSKGQ)**.

Definition 3.1. Given a socio-spatial-textual graph $G = (V, E)$, set of POIs Q, minimum acquaintance constraint c, minimum group size n_{min}, maximum group size n_{max}, maximum spatial range d_{max}, and the number of groups to be ranked k, the result of *Top-k SSKGQ* is a list L.

$$L = \{\langle G_1, q_1, \text{score}(G_1, q_1)\rangle, \langle G_2, q_2, \text{score}(G_2, q_2)\rangle, \ldots, \langle G_k, q_k, \text{score}(G_k, q_k)\rangle\}$$

In list L, each tuple $\langle G_i, q_i, \text{score}(G_i, q_i)\rangle, G_i \subseteq G, q_i \in Q$ satisfies the following conditions:

(1) $n_{min} \leq |V_i| \leq n_{max}$,
(2) $\forall v \in V_i, \deg(v_i) \geq c$, where $\deg(v)$ is the undirected degree/social connection of v,
(3) $\forall v \in V_i,\ \text{dist}(v.\lambda, q_i.\lambda) \leq d_{max}$,
(4) $\text{score}(G_i, q_i) \geq \text{score}(G_{i+1}, q_{i+1}),\ 1 \leq i \leq k-1$,
(5) $\text{score}(G_k, q_k) \geq \text{score}(G_j, q_j),\ \forall j > k$.

Since the function score in the above definition is evaluated based on individual scores of the social, spatial, and textual attributes of the group and the corresponding POIs, and the size of the group, we now formally defined the score function as follows.

3.3 The Score Function

Given a socio-spatial-textual network $G = (V, E)$, a group $G_g = (V_g, E_g)$, which is a sub-graph of G, where $V_g \subseteq V$ and E_g is the set of edges in the induced sub-graph $G[V_g]$, and a meeting point q, the socio-spatial-textual group score of G_g with respect to q is defined as $\text{score}(G_g, q) : (G_g, q) \rightarrow [0, 1]$:

$$SCORE(G_g, q) = w_1 S_{sc} + w_2 S_{sp} + w_3 S_{kw(u,v)} + w_4 S_{kw(v,q)} + w_5 S_{sz}. \quad (1)$$

Here, S_{sc}, S_{sp}, $S_{kw(u,v)}$, $S_{kw(v,q)}$, and S_{sz} are the social score, spatial score, inter-member keyword-similarity score, group-POI keyword-similarity score, and group-size score of G_g w.r.t. q, respectively. w_i denotes the weight of individual scores, where $w_i \in [0, 1]$, $\forall i \in \{1, \ldots, 5\}$ and $\sum_i w_i = 1$. We use similar definitions of social S_{sc} and spatial S_{sp} scores as defined in Reference [14] and define the other three new scores in this article. The details of the scores are as follows.

3.3.1 Social Connectivity Score: S_{sc}. The *Social connectivity score* S_{sc} of a socio-spatial graph $G = (V, E)$ is defined as the graph density of G, which can be formally defined as follows:

$$S_{sc} = \frac{2|E_g|}{|V_g|(|V_g| - 1)} = \frac{\sum_{v \in V_g} \left|N_v^{V_g}\right|}{|V_g|(|V_g| - 1)}. \quad (2)$$

Here, N_v^V denotes the set of acquainted neighbors of member v.

3.3.2 Spatial Score: S_{sp}. The *Spatial score* S_{sp} of a graph $G = (V, E)$ with respect to a meeting point q measures the spatial closeness of its members to q. We can formally define the spatial score of a group as follows:

$$S_{sp} = 1 - \frac{\sum_{v \in V_g} DIST(v.\lambda, q.\lambda)}{|V_g| \times d_{max}}. \quad (3)$$

Here $\text{dist}(v.\lambda, q.\lambda)$ is the spatial distance (i.e., Euclidean distance) between member v and meeting point q.

3.3.3 Keyword Scores: $S_{kw(mem-mem)}$ and $S_{kw(mem-poi)}$. In a group of people meeting at a POI, two types of keyword-similarity are important: (1) keyword-similarity among the members in the group and (2) keyword-similarity of each member with the POI. Hence, to capture these similarities, we define two types of keyword-scores: (1) the *Inter-member Keyword-similarity Score* $S_{kw(mem-mem)}$ and (2) the *Group-POI Keyword-similarity Score* $S_{kw(mem-poi)}$.

We define $S_{kw(mem-mem)}$ as the *mean Jaccard Index*[1] of the keyword-sets for every member-pair (u, v) in a candidate solution group, given by the following equation:

$$\begin{aligned} S_{kw(mem-mem)} &= \sum_{u,v \in V_g, u \neq v} J(u.\psi, v.\psi) \div \binom{|V_g|}{2} \\ &= \sum_{u,v \in V_g, u \neq v} \frac{|u.\psi \cap v.\psi|}{|u.\psi \cup v.\psi|} \div \binom{|V_g|}{2}. \end{aligned} \quad (4)$$

Similarly, we define $S_{kw(mem-poi)}$ regarding POI q as the *mean Jaccard Index* of the keyword-set of q with every member$v$ in the candidate solution group, which is formally defined as follows:

$$S_{kw(mem-poi)} = \sum_{v \in V_g} J(v.\psi, q.\psi) \div |V_g| = \sum_{v \in V_g} \frac{|v.\psi \cap q.\psi|}{|v.\psi \cup q.\psi|} \div |V_g|. \quad (5)$$

[1]The Jaccard Index, or Jaccard Similarity of two sets A and B, denoted by $J(A, B)$ is defined as the ratio of the cardinality of $A \cap B$ and the cardinality of $A \cup B$; i.e., $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$.

3.3.4 Group-size Score: S_{sz}. We define the group size score S_{sz} of $G_g = (V_g, E_g)$ such that a group with a disallowed size has a score of 0. For allowable group size, S_{sz} is monotonically increasing with respect to $|V|$:

$$S_{sz} = \begin{cases} 0 & |V_g| < n_{min}, \\ \frac{|V_g| - n_{min} + 1}{n_{max} - n_{min} + 1} & n_{min} \leq |V_g| \leq n_{max}, \\ 0 & |V_g| > n_{max}. \end{cases} \tag{6}$$

3.3.5 Free Text Instead of Keyword Set. It is safe to assume that in most real-world scenarios, the POIs will have a definite set of keywords. The users in the social network, however, may not have so. In such cases, we can use techniques such as a *bag of words* to compute the member-POI keyword score $S_{kw(mem-poi)}$. However, to compute the inter-member keyword score $S_{kw(mem-mem)}$ in such a scenario, or in certain cases where even the POIs have descriptive (free) text instead of a *set* of keywords, computation of the keyword score(s) will require determining the textual similarity and semantic similarity, and advanced language models such as BERT [11] and GPT-3 [3] can be used for such purposes.

Note that, in such cases, we can use these NLP models and normalize the similarity score returned by the models on a scale from 0 to 1 and use that score in Equation (1). Such datasets are not readily available for our purpose and will require a significant amount of preprocessing. We defer the implementation of such techniques for future work and focus only on keywords in this article.

3.4 A Running Example

The above query formation and the computation of scores can be realized easily from the following example. Let us refer to Figure 1 in Section 1 and the motivating example discussed there. There are eight users and two meeting points in this network. The distances between the users and the meeting points are shown in Table 1. In this example, we set $c = 2, d_{max} = 10, n_{min} = 3, n_{max} = 4, k = 2$, and $\mathrm{w}_1 = \mathrm{w}_2 = \mathrm{w}_3 = \mathrm{w}_4 = \mathrm{w}_5 = 0.2$. We consider $G_3 = \{v_1, v_2, v_3\}$ according to our score function regarding POI q_2 at first. Although this group fulfills the minimum social acquaintance constraint and the minimum (and maximum) group size constraint, this group is not a candidate solution w.r.t. q_2, since v_1 and v_2 are located more than d_{max} distance away from q_2. However, the members of G_3 are all close to POI q_1 and hence, is a candidate solution w.r.t. q_1 Using Equations (2) through (6), we calculate S_{sc}, S_{sp}, $S_{kw(mem-mem)}$, $S_{kw(mem-poi)}$, and S_{sz} to be 1.0, 0.63, 0.56, 67, and 0.5, respectively. Substituting these values into our score function (Equation (1)), we get 0.671, which is the overall score of G_3 regarding POI q_1. We calculate the scores of the groups that satisfy the minimum acquaintance and the maximum distance constraints in a similar fashion. The scores of different w.r.t. POIs are shown in Table 3. From this table, we can see that the result of the query in this social network for the aforementioned configuration will be

$$L = \{\langle\{v_1, v_2, v_3\}, q_1, 0.671\rangle, \langle\{v_5, v_6, v_7, v_8\}, q_2, 0.644\rangle\}.$$

In a real-world scenario, the weights $\mathrm{w}_i, i \in [1, 5]$ will be set at the querist's discretion and depending on the relative weights, the result of the query may be quite different in the same social network.

4 NP-HARDNESS AND THE BASELINE

In this section, we will prove that our proposed problem, Top-k SSKGQ, is NP-hard, and then give a straightforward solution by modifying a state-of-the-art technique, which will act as our baseline.

Table 3. Scores of the Possible Groups

Group	Score re q_1	Score re q_2
$\{v_1, v_2, v_3\}$	0.671	—
$\{v_1, v_3, v_4\}$	0.517	—
$\{v_5, v_6, v_7\}$	0.413	0.562
$\{v_5, v_6, v_8\}$	—	0.547
$\{v_5, v_7, v_8\}$	—	0.506
$\{v_6, v_7, v_8\}$	—	0.562
$\{v_1, v_2, v_3, v_4\}$	0.632	—
$\{v_5, v_6, v_7, v_8\}$	—	0.644

4.1 NP-hardness

Theorem 4.1. *Top-k SSKGQ is NP-hard.*

Proof. We prove that SSKGQ is NP-hard by reducing the problem into finding n-clique that is a well-known NP-hard problem. Decision problem n-clique states that, given a graph G_c whether the graph contains a clique wherein a n-clique is a complete graph of n vertices with an edge connecting every two vertices. In SSKGQ, we let $G = G_c$, $n = n_{min} = n_{max}$, $c = n - 1$, $Q = \{q\}$, $d_{max} = \infty$, $\forall v \in V, DIST(v.\lambda, q.\lambda) = 1$, $\forall v \in V, q.\psi = v.\psi$, and $\forall (u, v) \in V \times V, u.\psi = v.\psi$. We first prove the necessary condition. If G_c contains an n-clique, then there must exist a group with the same vertices in the n-clique such that every member has social relations with all the other attendees of the group, and the total spatial distance is n. We then prove the sufficient condition. If G in SSKGQ contains a group with the size n and c as $n - 1$, then G_c in problem n-clique must contain a solution with size n too. Hence, the theorem follows. □

4.2 The Baseline Method

A straightforward way to solve Top-k SSKGQ is to enumerate all groups of size between n_{min} and n_{max} w.r.t. all POIs that satisfy all query constraints, evaluate each group using the score function, and take the best k of them. This combinatorial naive approach cannot scale to large networks. Thus, we develop a baseline by extending and reshaping the SSGQ algorithm proposed in Reference [32], which is the most closely related work to our study. The SSGQ algorithm finds the best group of a fixed size w.r.t. a single meeting point based on the socio-spatial criteria. It prunes the search tree using distance ordering and socio-spatial ordering. However, this algorithm does not consider keyword dimension or variable group sizes. Moreover, the SSGQ method imposes an *average* minimum acquaintance constraint among members in the resultant group(s) and an *average* maximum spatial distance constraint of the members from the meeting point. In contrast, we impose a minimum acquaintance constraint on *each* member in the resultant group(s), and a maximum spatial distance constraint on *each* member. Therefore, to answer Top-k-SSKGQ, we design the baseline method in the following way. We find groups of each meeting point for each allowable group size (between n_{min} and n_{max}) such that each group follows the social and spatial constraints. Since the SSGQ method is designed for finding only the best group (only one group), we maintain a priority queue of length k in our modified baseline method and keep updating the queue as the search continues. As for the ordering method, the SSGQ method uses the R-tree (the SR-tree in the enhanced version) and has no means of incorporating keywords in ordering and pruning the search space for using only distance ordering. Therefore, we also omit the keyword dimension in our ordering technique in the baseline method. Only in the score calculation part, we consider the keyword dimension and evaluate the groups using our score function.

5 AN EFFICIENT ALGORITHM FOR PROCESSING TOP-*K* SSKGQ

In this section, we propose an efficient algorithm, namely, **Efficient Flexible Group Selection (EFGS)** for answering Top-k SSKGQ. In our proposed algorithm, we first design an exact pruning strategy to avoid unnecessary explorations of the search space. Later, we propose an ordering technique for member selection effectively while exploring the search space for candidate members in a group.

5.1 Solution Overview

We apply the best first search with branch-and-bound strategy for generating all feasible groups. To construct a group, we consider two sets of members: an intermediate solution set (group) and a remaining/unexplored set. At each step, we first check if the search can be terminated by computing the upper bound on the scores of each possible solution group growing from the intermediate set. Otherwise, we include a member from the remaining set into the intermediate group based on a socio-spatial-keyword ordering.

We maintain a priority queue of size k to track top k groups. For each meeting point, we start with an empty set as the intermediate solution group and the set of all members in the graph as the set of remaining members. As the search progresses, we include users one by one into that intermediate set from the remaining set by following a socio-spatial-keyword ordering. The ordering ensures that the best solution groups are found early in the search process. Thus, we can stop exploring a branch of the search tree earlier by computing bounds on the groups that are already found. The computed bounds also help us in determining whether going further down the branch yields better groups than the ones we already found. When we find an intermediate group of size between the minimum and maximum allowable group size, we evaluate it using our score function and insert it into the solution list if the score is better than that of the kth group in the list. The search process is similar to the *finding all subsets of a fixed size using backtracking* problem with branch and bound for pruning out unnecessary branches, where the search is conducted for every POI. Thus, one of the key contributions of our approach is to develop bounds (see Section 5.2) that allow us to prune the search space efficiently.

Let V_I and V_R denote the intermediate and remaining set of members, respectively, regarding meeting point q. Initially, $V_I = \{\}$ and $V_R = V$. At each step, a member is extracted from V_R and included into V_I. When $n_{min} \leq |V_I| \leq n_{max}$ and V_I satisfies the familiarity constraint, V_I becomes a candidate solution group. We then compute its score and update the top k list. Finally, the algorithm backtracks to the previous state of V_I to complete a full tree search.

Procedure FindGroup($V_I, V_R, L, q, n_{min}, n_{max}$)

```
1  if (|V_I| + |V_R| < n_min) or |V_I| ≥ n_max then
2  |   return
3  if |L| = k then
4  |   if Prune(V_I, V_R, L, q, n_min, n_max) = true then
5  |   |   return                                          // prune the branch
6  u ← Extract(V_I, V_R, q)
7  if n_min ≤ |V_I + {u}| ≤ n_max then          // a candidate solution group found
8  |   UpdateList(L, V_I ∪ {u})                // evaluate the group, update ranklist
9  FindGroup(V_I ∪ {u}, V_R − {u}, L, q, n_min, n_max)     // continue exploring the branch
10 FindGroup(V_I, V_R − {u}, L, q, n_min, n_max)                          // backtrack
```

The selection of which member to extract from V_R at each iteration is crucial, and good ordering of the unexplored members will result in significant speedup. For example, distance ordering in V_R returns a member with minimum spatial distance to the meeting point q. However, keyword-similarity ordering returns a member with the maximum similarity of keywords with the meeting point q. In this problem, we use the IR tree [22] to implement spatial-keyword ordering that considers both spatial distance and keyword-similarity of each member in V_R regarding meeting point q. Since spatial-keyword ordering does not consider social connectivity and keyword-similarity within the intermediate group, each member from spatial-keyword ordering is passed through a social filter before including into V_I. With this Socio-Spatial-Keyword Ordering, we are now able to develop an algorithm that efficiently prunes unnecessary search space. We have detailed our ordering techniques in Section 5.3.

Procedure FindGroup() summarizes the core part of our algorithm, i.e., finding groups regarding a meeting point q. On line 1, we check the allowable group size. On lines 3 and 4, we check whether we can prune the branch based on our derived bounds to terminate the search (Section 5.2). In line 6, we retrieve a user from V_R to include it into V_I using our socio-spatio-keyword ordering described in Section 5.3. If a new candidate group is found, then the result list L is updated on line 8. Lines 9 and 10 are used to recursively continue the search with the newly retrieved vertex u and without the vertex u, respectively. For each meeting point, the search begins with calling FindGroup() with $V_I = V$ (set of all vertices in G), $V_R = \{\}$, and $L = \{\}$.

5.2 Advanced Termination of Search

At any point of exploring the socio-spatial network $G = (V, E)$ regarding POI q, for any intermediate solution group V_I and set of remaining members V_R, we compute an upper bound on the score of the best group that can grow from V_I. We backtrack to the previous state of the search if this upper bound is not higher than the score of the k^{th} group in our solution list, assuming that we have already evaluated k or more candidate solutions.

We compute upper bounds of S_{sc}, S_{sp}, $S_{kw(mem-poi)}$, and $S_{kw(mem-mem)}$ (which are defined in Section 3.1), denoted by $S^{\uparrow}_{sc}$, $S^{\uparrow}_{sp}$, $S^{\uparrow}_{kw(mem-poi)}$, and $S^{\uparrow}_{kw(me-mem)}$, respectively, which individually represent maximum possible social score, spatial score, member-POI-keyword score and member-member-keyword score of a candidate solution group can achieve. Since the total score of a group is the linear combination of these scores (plus the group-size score, which is fixed for a group of fixed size), a group that has the corresponding components of its score function equal to these upper bounds is the best group that we can hope to form by keep exploring further down the branch of the intermediate solution group in the search tree. Hence, we substitute these values into Equation (1) to determine the maximum possible score $S^{\uparrow}_n$ of a solution group $V_S = V_I \cup V_A$ formed by joining V_I with a subset of unexplored members $V_A \subseteq V_R$ of size $n - |V_I|$ for every possible value of $n = |V_S|$ where $n_{min} \leq n \leq n_{max}$. Evidently, no candidate solution group of size n growing from V_I can have a score higher than $S^{\uparrow}_n$. Hence, when $S^{\uparrow}_n$ is less than the score of the k^{th} group in our solution list, we can safely prune that branch, since it is not possible to find a better group by keeping exploring. Otherwise, there is a possibility that we can find at least one group that can make it on the rank list, and so we keep exploring further down the branch.

5.2.1 Upper Bound for the Social Score S_{sc}. From Equation (2), we see that the social score S_{SC} of a group is directly proportional to the total number of social connections in that group. Now, when we add new members to an intermediate solution group with the goal of forming a

candidate solution group, there will be the following three categories of edges (connections) in the group:

(1) edges that connect any two members in the intermediate solution group,
(2) edges that connect any two *newly added* members, and
(3) edges that connect a newly added member to a member in the intermediate solution group.

The number of edges in Category (1) is fixed for a particular intermediate solution group V_I, and is known prior to adding new members. However, the number of edges of Categories (2) and (3) depends on which new members we choose to add from the unexplored member-list V_R. Now, instead of forming and evaluating every possible candidate solution group $V_S = V_I \cup V_A$ growing from V_I by joining a set of additional unexplored members $V_A \subseteq V_R$ with V_I, we can compute the theoretical *maximum possible* number of edges (an upper bound) of Categories 2 and (3) given V_I and V_R, and form the group based on the upper bound. Essentially, we determine the maximum possible total number of edges in any candidate solution group growing from V_I and subsequently, the maximum possible social score that any candidate solution group formed by exploring further down the search tree in branch V_I can have. This gives us an upper bound $S_{SC}^{\uparrow}$ of the social score S_{SC}. Let $V_S = V_I \cup V_A$ be a candidate solution group formed by joining a subset of unexplored members V_A to the intermediate solution group V_I where $|V_S| = n$. Evidently, $|V_A| = n - |V_I|$, since V_I and V_A are disjoint. Now, let E_S be the set of edges among all the members in V_S, E_I be the set of edges among the members in V_I, and E_A be the set of edges connecting any two members in V_A. More specifically, E_S, E_I, and E_A are the set of edges in the induced sub-graphs $G[V_S]$, $G[V_I]$, and $G[V_A]$, respectively. Also, let E_{IA} bet the set of edges in G that connect a member in V_I to a member in V_A, i.e., $E_{IA} = \{(u, v) : u \in V_I, v \in V_A, (u, v) \in G.E\}$. Then, E_I, E_A, and E_{IA} are sets of edges in Categories (1), (2), and (3), respectively, and $|E_S| = |E_I| + |E_A| + |E_{IA}|$. Here, E_I is known a V_I is known. We need to determine the maximum possible number (upper bound) of *new edges* introduced when we join a subset of new members to V_I.

It is apparent that

$$|E_I| = \frac{1}{2} \sum_{v \in V_I} \left| N_v^{V_I} \right|, \tag{7}$$

where $N_v^{V_I}$ is the set of acquainted neighbors of member v in V_I.

Now, an upper bound on $|E_A|$ is obtained by

$$|E_A|^{\uparrow} = \frac{1}{2}|V_A| \sum_{v \in V_R} \max_{v \in V_R} \left| N_v^{V_R} \right| = \frac{1}{2}(n - |V_I|) \max_{v \in V_R} \left| N_v^{V_R} \right|. \tag{8}$$

$|E_A|^{\uparrow}$ is an upper bound on $|E_A|$, because the vertex with the maximum degree in V_R is identified, and $|V_A|(= n - |V_I|)$ vertices are selected from V_R, which guarantees $|E_A|^{\uparrow} \geq |E_A|$.

Finally, an upper bound on $|E_{IA}|$ is obtained by

$$|E_{IA}|^{\uparrow} = \sum_{v \in V_I} \left| N_v^{V_R} \right|. \tag{9}$$

$|E_{IA}|^{\uparrow}$ is an upper bound on $|E_{IA}|$, because, for every member $v \in V_I$, we count all the members in V_R that are connected to v. Since $V_A \subseteq V_R, |N_v^{V_R}| \geq |N_v^{V_A}|$ and the relation $|E_{IA}|^{\uparrow} \geq |E_{IA}|$ follows.

Thus, an upper bound on $|E_S|$ is obtained by $|E_S|^{\uparrow} = |E_I| + |E_A|^{\uparrow} + |E_{IA}|^{\uparrow}$.

Using these values, we obtain the upper bound on S_{sc} from Equation (2) as follows:

$$\begin{aligned} S_{sc}^{\uparrow} &= \frac{2|E_S|^{\uparrow}}{|V_S|(|V_S|-1)} \\ &= \frac{2(|E_I| + |E_A|^{\uparrow} + |E_{IA}|^{\uparrow})}{n(n-1)} \\ &= \frac{\sum_{v \in V_I} \left|N_v^{V_I}\right| + (n - |V_I|) \max\limits_{v \in V_R} \left|N_v^{V_R}\right| + 2\sum_{v \in V_I} \left|N_v^{V_R}\right|}{n(n-1)}. \end{aligned} \tag{10}$$

5.2.2 Lower Bound for the Spatial Score S_{sp}. Let us assume that we have an intermediate solution group V_I and we form a candidate solution group $V_S = V_I \cup V_A$ of size n (i.e., $|V_S| = n$) w.r.t. meeting point q by joining a subset V_A of unexplored members to the intermediate solution list V_I. The spatial score of V_S w.r.t. meeting point q depends on the aggregate distance of the members in V_S from q. We derive a lower bound on the aggregate distance of the members in V_A, which will give us an overall lower bound on the aggregate distance of the members in V_S. Using this lower bound, we can derive the upper bound on the spatial score that a candidate solution group growing from V_I can have.

Let, D_S, D_I, and D_A be the aggregate distance from q to the members in V_S, V_I, and V_A, respectively. Then, from Equation (3), the spatial score of V_S is given by

$$\begin{aligned} S_{SP} &= 1 - \frac{D_S}{|V_S| \times d_{max}} \\ &= 1 - \frac{D_I + D_A}{(|V_I| + |V_A|) \times d_{max}} && \text{(since } V_I \text{ and } V_A \text{ are disjoint)} \\ &= 1 - \frac{\sum_{u \in V_I} DIST(u.\lambda, q.\lambda) + \sum_{v \in V_A} DIST(v.\lambda, q.\lambda)}{n \times d_{max}}. \end{aligned} \tag{11}$$

Let d_{min} be the minimum distance from meeting point q to any member in V_R, i.e.,

$$d_{min} = \min_{\forall v \in V_R} [DIST(v.\lambda, q.\lambda)]. \tag{12}$$

Hence, when we form a solution group, the minimum distance that any member in V_A can have to the meeting point q is d_{min}. Therefore, the aggregate distance of the newly added members cannot be lower than $|V_A| \times d_{min}$. More specifically, $\sum_{v \in V_A} DIST(v.\lambda, q.\lambda) \geq |V_A| \times d_{min}$, thus $D_A^{\downarrow} = |V_A| \times d_{min}$ is a lower bound on D_A.

Substituting the value of $D_A^{\downarrow}$ for D_A in Equation (11), an upper bound for S_{sp} can thus be computed as follows:

$$S_{sp}^{\uparrow} = 1 - \frac{\sum_{v \in V_I} DIST(v.\lambda, q.\lambda) + (n - |V_I|) \times d_{min}}{n \times d_{max}}. \tag{13}$$

5.2.3 Upper Bounds for Member-Member Keyword Score $S_{kw(mem-mem)}$ and Member-POI Keyword Score $S_{kw(mem-poi)}$. Recalling the argument from Section 5.2.1, we see that there are three categories of member-member pair (u, v) in a candidate solution group $V_S = V_I \cup V_A$:

(1) $u \in V_I, v \in V_I$,
(2) $u \in V_A, v \in V_A$,
(3) $u \in V_I, v \in V_A$.

Here, V_I is the intermediate solution group and $V_A \subseteq V_R$ is a subset of all the unexplored member that is joined with V_I to form the candidate solution group. Now, from Equation (4), we see that the member-member keyword score of V_S is directly proportional to the sum of Jaccard indices of the keyword-sets of its member-pairs, i.e., the sum of the ratio of the cardinality of the intersection and the cardinality of the union of the keyword sets of every member-member pair in V_S. To derive the upper bounds on the member-member keyword score, we will compute the maximum possible sum of Jaccard indices for each of these three kinds of pairs.

Let, $J(u.\psi, v.\psi)$ be the Jaccard index between sets $u.\psi$ and $v.\psi$. Then,

$$\begin{aligned}&\sum_{\substack{u,v\in V_S\\u\neq v}} J(u.\psi, v.\psi)\\ &= \sum_{\substack{u,v\in V_I\\u\neq v}} J(u.\psi, v.\psi) + \sum_{\substack{u,v\in V_A\\u\neq v}} J(u.\psi, v.\psi) + \sum_{\substack{u\in V_I\\v\in V_A}} J(u.\psi, v.\psi).\end{aligned} \tag{14}$$

The three terms on the right-hand side of Equation (14) represent the sum of Jaccard indices of Categories (1), (2), and (3) pairs described above. The sum of Jaccard indices for Category (1) pairs is fixed and known prior to forming V_S, since V_I is known. However, $\sum_{u,v\in V_A, u\neq v} J(u.\psi, v.\psi)$ and $\sum_{u\in V_I, v\in V_A} J(u.\psi, v.\psi)$ depend on the choice of V_A. Our goal is to derive upper bounds on these two for an arbitrary choice of V_A such that $V_S = V_I \cup V_A$ and $|V_S| = n$.

Let, P be the union of the keywords of all the members in V_R, i.e.,

$$P = \bigcup \{v.\psi | v \in V_R\}. \tag{15}$$

Then, $\sum_{u,v\in V_A, u\neq v} J(P, P)$ is an upper bound for $\sum_{u,v\in V_A; u\neq v} J(u.\psi, v.\psi)$, and $\sum_{u\in V_I} J(u.\psi, P)$ is an upper bound for $\sum_{u\in V_I, v\in V_A} J(u.\psi, v.\psi)$, because $\forall v \in V_A\ v.\psi \subseteq P$.

Thus, we obtain an upper bound for $\sum_{\substack{u,v\in V_S\\u\neq v}} J(u.\psi, v.\psi)$ as

$$\begin{aligned}&\sum_{\substack{u,v\in V_S\\u\neq v}} J^{\uparrow}(u.\psi, v.\psi)\\ &= \sum_{\substack{u,v\in V_I\\u\neq v}} J(u.\psi, v.\psi) + \sum_{\substack{u,v\in V_A\\u\neq v}} J(P, P) + \sum_{\substack{u\in V_I\\v\in V_A}} J(u.\psi, P)\\ &= \sum_{\substack{u,v\in V_I\\u\neq v}} \frac{u.\psi \cap v.\psi}{u.\psi \cup v.\psi} + \sum_{\substack{u,v\in V_A\\u\neq v}} (1) + \sum_{u\in V_I} \frac{u.\psi \cap P}{u.\psi \cup P}\\ &= \sum_{\substack{u,v\in V_I\\u\neq v}} \frac{u.\psi \cap v.\psi}{u.\psi \cup v.\psi} + \binom{n - |V_I|}{2} + \sum_{u\in V_I} \frac{u.\psi \cap P}{u.\psi \cup P}.\end{aligned} \tag{16}$$

Hence, from Equation (4), we get an upper bound for $S_{kw(mem-mem)}$ as follows:

$$S^{\uparrow}_{kw(mem-mem)} = \frac{\sum_{u,v\in V_S, u\neq v} J^{\uparrow}(u.\psi, v.\psi)}{\binom{|n|}{2}}, \tag{17}$$

where the value of the numerator is obtained from Equation (16).

Similarly, we get an upper bound for $S_{kw(mem-poi)}$ regarding POI q from Equation (5) as

$$S^{\uparrow}_{kw(mem-poi)} = \frac{\sum_{v \in V_I} \frac{q.\psi \cap v.\psi}{q.\psi \cup v.\psi} + (n - |V_I|)\frac{q.\psi \cap P}{q.\psi \cup P}}{n}. \tag{18}$$

5.2.4 Advanced Termination. We substitute the values of S_{sc}, S_{sp}, $S_{kw(mem-mem)}$, and $S_{kw(mem-poi)}$ in Equation (1) with $S^{\uparrow}_{sc}$, $S^{\uparrow}_{sp}$, $S^{\uparrow}_{kw(mem-mem)}$, and $S^{\uparrow}_{kw(mem-poi)}$, respectively, to compute the score upper bound $SCORE^{\uparrow}(G, q)$ of candidate solution groups growing from V_I for every allowable group size $n \in \{n_{min}, \ldots, n_{max}\}$. Let $S^{\uparrow}_n$ denote this upper bound for groups having size n. If $S^{\uparrow}_n$ is less than the score of $L[k]$ (kth group in the solution list L that we have up until this point) for every possible value of n, then we reject that branch and backtrack. Otherwise, we expand on V_I. Function Prune() illustrates this pruning technique from a more systematic point of view. This function returns `true` if pruning on a branch is possible, and returns `false` otherwise. On line 3, we compute the score upper bound for every allowable group size. For any group size n, if there is a possibility of finding a better group than $L[k]$ by exploring further down V_I, Function Prune() immediately returns `false` (lines 5 and 6). If there is no prospect of finding a better group than $L[k]$ (kth group in the solution list), then it returns `true` on line 8, which indicates the decision to prune the branch.

It is important to note that when we say *substitute the values*, we do not necessarily mean to compute the values from the ground up at each call of the procedures. It is possible to reuse the computational results while backtracking. For example, in Equation (11), the term $\sum_{u \in V_I} DIST(u.\lambda, q.\lambda)$ need not be recomputed and can be passed as a parameter to the subsequent recursive calls of FindGroup(). At each step, this allows us to add only the distance of the newly added member(s) to the sum of the distances passed down the recursion tree. Computational overhead for the other values can be similarly reduced.

Function Prune($V_I.V_R, L, q$)

1 **for** *every possible group-size n* **do**
2 Compute $S^{\uparrow}_{sc}, S^{\uparrow}_{sp}, S^{\uparrow}_{kw(mem-mem)}, S^{\uparrow}_{kw(mem-poi)}$, and S_{sz} using Equations (10), (13), (17), (18), and (6)
3 $S^{\uparrow}_n \leftarrow w_1 S^{\uparrow}_{sc} + w_2 S^{\uparrow}_{sp} + w_3 S^{\uparrow}_{kw(mem-mem)} + w_4 S^{\uparrow}_{kw(mem-poi)} + w_5 S_{sz}$
`// compute score upper bound` $S^{\uparrow}_n$ `from Equation (1)`
4
5 **if** $S^{\uparrow}_n > L[k].score$ **then** `// may find groups better than` $L[k]$
6 **return** false `// do not prune`
7
8 **return** true `// no better groups possible, prune this branch`

5.3 Social-Spatial-Keyword Ordering

The order in which we explore the unvisited members in the search tree is important as it reduces the run time of the algorithm significantly. Exploring the suitable members earlier ensures the formation of the better groups earlier, which in turn enables us to prune larger portions of the search tree. We employ two ordering techniques in our algorithm. First, we order the members

with respect to the meeting point (POI), and then we follow this ordering to further order the unexplored members in V_R with respect to the intermediate solution group V_I so that we can extract the most suitable member early.

5.3.1 Ordering regarding the POI. At each step, we extract a member v from V_R and include in V_I. To decide which member to extract first, we propose an ordering (namely, spatial-keyword ordering) of members that is a linear combination of both spatial distance and keyword-match of v regarding q. To mitigate the effect of different numerical values, we normalize both spatial distance and keyword-match in the ordering function. The spatial-keyword distance $dist_{sp-kw}$ of member v regarding meeting point q is defined as follows:

$$dist_{sp-kw} = \frac{DIST(v.\lambda, q.\lambda)}{d_{max}} + \left(1 - \frac{|v.\psi \cap q.\psi|}{|v.\psi \cup q.\psi|}\right). \tag{19}$$

5.3.2 Ordering regarding V_I. To ensure that a member extracted from V_R to be included in V_I has good social connectivity and inter-member keyword match with the members in the intermediate solution-group, we pass the members of V_R through a socio-keyword filter, following the order enforced by Equation (19). To determine which user to extract first from the set of unexplored users, we can either consider only the social connectivity of a new user with the intermediate solution group or consider both the social connectivity and inter-member keyword similarity. Both techniques are discussed below.

Considering only social connectivity for ordering: For each member $v \in V_R$, let us denote the set of members in V_I who are socially connected with v by $N_v^{V_I}$. Thus, a greater value of $|N_v^{V_I}|$ indicates better social connectivity within the intermediate group. To retrieve the best possible member earlier, we explore the members in V_R according to the increasing value of $dist_{sp-kw}$, extracting the first member v having $|N_v^{V_I}| = |V_I|$, since $|V_I|$ is the maximum possible value for $|N_v^{V_I}|$. However, if no member from V_R achieves $|N_v^{V_I}| = |V_I|$, it does not imply that every solution growing from V_I does not have sufficient social connectivity. It is possible to find a vertex v in V_R and a solution growing from $V_I \cup \{v\}$ when other members added later can form a good enough solution group. In that case, we include the member with the maximum value of $|N_v^{V_I}|$.

Considering both social connectivity and keyword similarity for ordering: Other than good social connectivity, we also want a member to have good keyword similarity with the members in V_I. The *total* keyword-similarity of v with V_I is obtained from $\sum_{u \in V_I} J(u.\psi, v.\psi) = \sum_{u \in V_I} \frac{|v.\psi \cap u.\psi|}{|v.\psi \cup u.\psi|}$. The maximum possible value for both $|N_v^{V_I}|$ and $\sum_{u \in V_I} J(u.\psi, v.\psi)$ is $|V_I|$. Therefore, we introduce another ordering metric $\Gamma_v \in [0, 1]$ for each members $v \in V_R$ defined as

$$\Gamma_v = \frac{\left|N_v^{V_I}\right| + \sum_{u \in V_I} J(u.\psi, v.\psi)}{2|V_I|}. \tag{20}$$

The $2|V_I|$ introduced in the right-hand side of Equation (20) is a normalization factor, ensuring a value from 0 to 1 for Γ_v. Similar to social ordering, we explore the members in V_R according to increasing value of $dist_{sp-kw}$. As soon as we encounter a member v in V_R with $\Gamma_v = 1$, we include that member into V_I. In the situation where no such member is found, we extract the member with the maximum value of Γ_v. If there is more than one such member, then we retrieve the one appearing earliest in the ordering defined by Equation (19). Whether ordering based only on $|N_v^{V_I}|$ performs better than ordering based on Γ_v depends on the structure of the social network. Intuitively, using only $|N_v^{V_I}|$ might require exploring a lower number of members at each step. However, Γ_v ensures that not only the social connectivity but also the keyword-similarity is taken

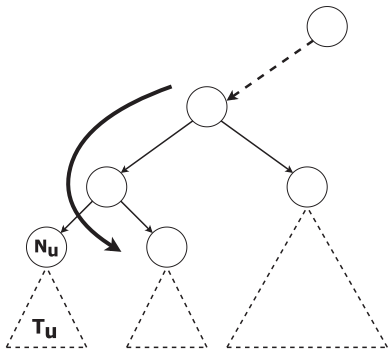


Fig. 2. Pruning the search space.

into consideration in determining the best member to extract. This strategy may require us to evaluate a larger number of candidates for extraction at each step. However, this procedure is also expected to result in an overall lower number of steps as promising members are apparently extracted early on.

5.4 Optimality Proof for the EFGS Algorithm

EFGS performs a complete search in the search space. Therefore, it guarantees to return a list of the best k groups in the social network. In this section, we prove that EFGS is indeed optimal by the *Method of Contradiction*.

If EFGS is not optimal, then there must be at least one unexplored group G_u with the following two properties:

(1) $SCORE(G_u, q) > L[k].score$, and
(2) $\langle G_u, q, SCORE(G_u, q)\rangle \notin L$,

where q is any arbitrary POI and L is the solution list returned by EFGS.

Now, without loss of generality, let us assume that G_u is contained in the sub-tree T_u, which was pruned off by EFGS at the node N_u (Figure 2). Let S_{N_u} denote the value of $S_n^{\uparrow}$ as defined in Section 5.2.4 computed at node N_u on line 4 of the function `Prune`. Since G_u is fully contained within T_u, S_{N_u} must be at least as large as $SCORE(G_u, q)$. Therefore, our initial assumption of $SCORE(G_u, q) > L[k].score$ leads us to the relation $S_{N_u} > L[k].score$, which is a contradiction, because T_u will be pruned off iff $S_{N_u} < L[k].score$. Otherwise, the `FindGroup` method will continue to explore further down the search-tree. Hence, there can be no group $G_u \notin L$ that is better than the k^{th} (last) group in L, and our claim that EFGS is optimal is thereby proven.

6 A FASTER APPROXIMATE APPROACH

Our proposed EFGS algorithm gives an optimal solution to SSKGQ. However, it might still not be scalable enough for extremely large social networks. For such networks, we need an even faster algorithm that, at the expense of the possibility of providing a list of sub-optimal groups, will be suitable for deploying at scale. In other words, we want an approximate method that may not be optimal but is significantly faster and less resource-intensive. The main idea of our approximate method is to include a set of multiple members $V_C \subseteq V_R$ at a time into the intermediate solution group V_I from the set of unexplored members V_R to incrementally build a candidate solution group, as opposed to the EFGS method where at each step we extract only one member from V_R

according to socio-spatial-keyword ordering. We call our approximate algorithm *EFGS-FA* (EFGS-Fast Approximate).

In other words, in this approximate approach, instead of the ExtractMulti method, which returns a single member, we use the ExtractMulti method, which returns a set V_C of multiple members. The members in V_C are considered for inclusion as a cluster, i.e., we either include all of them into the intermediate solution group or do not consider any of them for inclusion. Procedure FindGroupsFast(), which is the counterpart of Procedure FindGroup() for the faster algorithm, illustrates this approximate approach from a high-level standpoint. In FindGroupsFast(), we follow the same ordering techniques described in Section 5.3 to include members into V_I, starting with $V_C = \{\}$ at each call of ExtractMulti. The number of new members to extract (i.e., $|V_C|$) changes as V_I grows. At the ith level of the recursive call to FindGroup (which corresponds to the depth of the node at which FindGroup is called), we consider *exactly* i members for extraction from V_R, i.e., ExtractMulti returns a set of i members so that the number of members to extract at a time increases as the size of the intermediate solution group increases. This is because smaller intermediate groups are more sensitive to the inclusion of multiple members at a time. At the top level of recursion when $V_I = \{\}$, we extract only 1 member from V_R. At the next level where $|V_I| = 1$, we extract two members and so on. The major differences of FindGroupsFast() from FindGroup() are on lines 6 and 7, where we extract and add more than one member, opposed to exactly one.

Procedure FindGroupsFast($V_I, V_R, q, step$)

1 **if** $(|V_I| + |V_R| < n_{min})$ **or** $|V_I| \geq n_{max}$ **then**
2 | **return**
3 **if** *the solution list already contains k no. of groups* **then**
4 | **if** Prune($V_I, V_R, L, q, n_{min}, n_{max}$) = *true* **then**
5 | | **return** // prune the branch
6 $V_C \leftarrow$ ExtractMulti($V_I, V_R, q, step$)
// *step* is the number of members to extract; i.e., $step = |V_C|$
7 **if** $n_{min} \leq |V_I \cup V_C| \leq n_{max}$ **then** // a candidate solution group found
8 | UpdateList($L, V_I \cup V_C$) // evaluate the group, update ranklist
9 FindGroupsFast($V_I \cup V_C, V_R - V_C, q, min(step + 1, n_{max} - |V_I| - |V_C|)$ // continue exploring the branch
10 FindGroupsFast($V_I, V_R - V_C, q, step$) // backtrack

6.1 Time Complexity of EFGS-FA

To compute the time-complexity for the approximate algorithm, we observe that if we require m steps to reach a candidate group of size n_{max} (maximum possible group-size), then

$$\sum_{1}^{m-1} i < n_{max} \leq \sum_{1}^{m} i, \tag{21}$$

from which we get

$$-0.5 + \sqrt{1 + 8 \times n_{max}} \leq m \leq 0.5 + \sqrt{1 + 8 \times n_{max}}, \tag{22}$$

i.e., at the last level of recursion, we add $\Omega(\lfloor 1.4\sqrt{n_{max}} \rfloor)$ members to V_I and then backtrack, removing those m members, and then add m more unexplored members from V_R again, continuing

this until V_R is exhausted. If $n_{max} << N$, then the average number of steps to completely explore a single branch of the search tree can indeed be approximated by $N/\lfloor 1.4\sqrt{n_{max}} \rfloor$. In other words, the height of the search tree reduces to $N/\lfloor 1.4\sqrt{n_{max}} \rfloor$ from N.

Thus, the worst-case time complexity of the approximate algorithm becomes $O(2^{N/\lfloor 1.4\sqrt{n_{max}} \rfloor})$, which is a significant improvement over the EFGS algorithm that has a worst-case time-complexity of $O(2^N)$.

6.2 Worst Case Approximate Ratio of EFGS-FA

We will derive a theoretical lower bound on the score of a suboptimal group (G_{sub}) retrieved by the approximation algorithm where the algorithm misses a group that has a higher possible score (G_{opt}).

Let c_{min} and d_{max} be the minimum connectivity and maximum allowable distance, respectively. The set of unexplored members (V_R) is sorted according to Equation (19).

In the worst possible scenario, each member in G_{sub} has a distance d_{max} from the meeting point, and there are no inter-member or member-poi keyword matches. Now, the approximate algorithm will only retrieve such members if and only if the social connectivity score of G_{sub} is greater than G_{opt} (assuming secondary ordering using $|N_v^{V_I}|$, discussed in Section 5.3.2); otherwise, members from G_{opt} would be retrieved first. Furthermore, each member in G_{sub} must have at least one more connection in the solution group than each member in G_{opt}. The number of connections for each member in G_{sub} that satisfies the above condition and keeps the ratio of the social-connectivity scores of the two groups at a minimum is $c_{min} + 1$. Hence, the number of connections for each member in G_{opt} is c_{min}. Moreover, in the worst possible case, there will be no inter-member or member-POI keyword matches in G_{sub} and hence, the keyword-scores for that group will be zero, and so will be the spatial score (since each member is d_{max} distance away from the POI). However, G_{opt} will have the spatial score and the keyword score equal to 1.

Thus, the score of the suboptimal group can be expressed as

$$S_{sub} = \mathrm{w}_1 \frac{2(c_{min}+1)}{|G_{sub}|-1} + \mathrm{w}_2 \cdot 0 + \mathrm{w}_3 \cdot 0 + \mathrm{w}_4 \cdot 0 + \mathrm{w}_5 \frac{|G_{sub}|}{n_{max}} = \mathrm{w}_1 \frac{2(c_{min}+1)}{|G_{sub}|-1} + \mathrm{w}_5 \frac{|G_{sub}|}{n_{max}}.$$

And the score of the missed group (optimal) can be expressed as

$$S_{opt} = \mathrm{w}_1 \frac{2c_{min}}{|G_{opt}|-1} + \mathrm{w}_2 + \mathrm{w}_3 + \mathrm{w}_4 + \mathrm{w}_5 \frac{|G_{opt}|}{n_{max}}.$$

The ratio of these two scores is the approximate ratio

$$\frac{S_{sub}}{S_{opt}} = \frac{w_1 \frac{2(c_{min}+1)}{|G_{sub}|-1} + w_5 \frac{|G_{sub}|}{n_{max}}}{w_1 \frac{2c_{min}}{|G_{opt}|-1} + \mathrm{w}_2 + \mathrm{w}_3 + \mathrm{w}_4 + \mathrm{w}_5 \frac{|G_{opt}|}{n_{max}}}.$$

To find the worst-case approximation ratio, we compute the lowest possible value of the numerator and the highest possible value of the denominator.

We observe that the scores here are convex functions (hyperbolas) of the group size with groups of size > 1. Hence, we set the derivative of the numerator w.r.t. the group size equal to zero (i.e., find the minima) to determine the group size that results in the lowest possible score for G_{sub} denoted by $|G^*_{sub}|$

$$|G^*_{sub}| = \mathtt{min}\left(n_{max}, \mathtt{max}\left(n_{min}, 1 + \sqrt{\frac{2\mathrm{w}_1(c_{min}+1)n_{max}}{\mathrm{w}_5}}\right)\right).$$

For $w_1 << w_5$, the denominator is maximized when $|G_{opt}| = n_{max}$. For $w_1 >> w_5$, the denominator is maximized when $|G_{opt}| = n_{min}$.We consider both extremes.

Thus, the approximation ratio is given by

$$\frac{S_{sub}}{S_{opt}} = \frac{w_1 \frac{2(c_{min}+1)}{|G^*_{sub}|-1} + w_5 \frac{|G^*_{sub}|}{n_{max}}}{\max\left(w_1 \frac{2c_{min}}{n_{max}-1} + w_5, w_1 \frac{2c_{min}}{n_{min}-1} + w_5 \frac{n_{min}}{n_{max}}\right) + w_2 + w_3 + w_4}.$$

Theoretically, this approximation ratio can be as high as 1 and as low as 0, depending on which factors are emphasized by the query issuer. For example, if the query issuer cares only about the social score ($w_1 = 1$) and wants to find closely knit fixed size groups only, then from the above equation we can see that the approximation ration will be high (~1) and it will be equal to 1 in cliques. (Figure 9(e)). However, if all the emphasis is on keyword matching/spatial score (i.e., $w_j = 0, j \in [1, 5]$) and the worst-case scenario described above occurs (every user in a group found by EFGS-FA is at a distance d_{max} from the POI and there is no keyword match), then the approximation ratio can be theoretically 0. However, in real-world scenarios, the chances of a group's keyword scores and spatial the score being 0 are almost non-existent, and it is highly unlikely that a query issuer will emphasize only spatial or keyword scores. Hence, the cases where the approximation ratio is 0 never occur in the real world, and EFGS-FA works well in general, as seen in our experimental results in Section 7.5.

7 EXPERIMENTAL RESULTS

We have conducted an extensive experimental evaluation to show the efficiency and effectiveness of our proposed exact (EFGS) and approximate (EFGS-FA) algorithms and compare these approaches with our baseline approach (Section 4.2). We compare the execution time and the number of explored nodes (i.e., intermediate groups) that measure the efficiency and scalability, respectively, while processing Top-k SSKGQ by each of these algorithms.

7.1 Platform configuration

We use the hardware and software configurations as shown in Table 4, while running our experiments.

Table 4. Machine Configuration

CPU	Intel Core i7-4702MQ CPU @ 2.20 GHz × 8
Memory	8 GB DDR3 @ 1,600 MHz
Disk	Transcend SSD230S Solid State Drive @ 6 Gb/s
Graphics	Intel Haswell Mobile
Programming Language	Python 3.7.0
Operating System	Arch Linux; Kernel: 4.18.14-arch1-1-ARCH
Operating System Type	Linux 64-bit

7.2 Datasets

We used the Brightkite and the Gowalla [8] datasets to build our socio-spatial graph, and then we used Foursquare textual Dataset [29, 30] for augmenting the Brightkite and Gowalla socio-spatial graph with keywords.

7.2.1 Brightkite and Gowalla Dataset. The location-based social networking service provider Brightkite enabled its users to share their locations by checking in. The dataset was built by

Table 5. Brightkite and Gowalla Dataset Statistics

Attribute	Brightkite	Gowalla
Nodes	58,228	19,6591
Edges	214,078	950,327
Check-ins	4,491,143	6,442,890
Nodes in largest WCC	56,739 (0.974)	196,591 (1.000)
Edges in largest WCC	212,945 (0.995)	950,327 (1.000)
Nodes in largest SCC	56,739 (0.974)	196,591 (1.000)
Edges in largest SCC	212,945 (0.995)	950,327 (1.000)
Average clustering coefficient	0.1723	0.2367
Number of triangles	494,728	2,273,138
Fraction of closed triangles	0.03979	0.007952
Diameter (longest shortest path)	16	14
90-percentile effective diameter	6	5.7

collecting the friendship network by using their public API. The dataset consists of 58,228 nodes and 214,078 edges. Nodes and edges represent users and social connection between two users, respectively. The dataset contains a total of 4,491,143 check-ins of users over the period of April 2008 to October 2010.

Similar to Brightkite, Gowalla is a location-based social networking website. The friendship network and check-in data was collected using the Gowalla public API, and consists of 196,591 nodes and 950,327 edges, and a total of 6,442,890 check-ins. The data was collected over the period of February 2009–October 2010.

Since members may have multiple check-ins, we consider the most frequent check-in location as the location of a particular member in both Brightkite and Gowalla. Members with no check-in data are discarded. A summary of these datasets is shown in Table 5.

7.2.2 Augmenting Socio-Spatial Graph with Keywords. Since the Brightkite and Gowalla datasets do not have any textual attributes associated with the users, we augment each user with a set of keywords obtained from a real-world spatio-textual datasets of Foursquare [29, 30].

This dataset, which was collected from Foursquare from 24 October 2011 to 20 February 2012, includes check-in and tags (keywords) of restaurant venues in New York City. We collected 2,603 sets of keywords.

To augment a user v with a keyword-set $v.\psi$, we first determine the size of $v.\psi$ (i.e., $|v.\psi|$), which is a random number between the minimum and the maximum size of the original 2603 keyword-sets $S_i, \forall i = 1, 2, \ldots, 2, 603$. In other words, $min(|S_i|) \leq |v.\psi| \leq max(|S_i|); \forall i = 1, 2, \ldots, 2, 603; \forall v \in V$. After that, we construct $v.\psi$ by weighted random sampling from $\bigcup S_i$ (the union of the original keyword-sets) where the weight wt_m of a keyword m is given by the following equation:

$$wt_m = \frac{\#\text{ of occurrences of } m \text{ among all sets } S_i}{\sum_{i=1}^{2,603} |S_i|}. \tag{23}$$

7.2.3 POI Sets Generation. For both datasets, we generate the meeting point (POI) co-ordinates from the co-ordinates of the user locations using the Alias Method [27]. The Alias Method ensures that the POI coordinates are sampled from the same underlying probability distribution of the user coordinates. For augmenting POIs with keywords, we use a similar strategy as we took for augmenting users with keywords.

Table 6. Parameter Values

Parameter	Range	Default Value
Number of meeting points, mp	{50, 100, 150, 200}	100
Minimum group size, n_{min}	{5, 6, 7, 8}	7
Maximum allowable distance, d_{max}	{2.0, 2.5, 3.0,3.5, 4.0, 4.5}	4.0
Minimum acquaintance constrain, c	{2, 3, 4, 5}	3
Solution list size, k	{4, 8, 12, 16}	8
Maximum group size, n_{max}	—	$1.5 \times n_{min}$
Weights, w_1, w_2, w_3, w_4, w_5	—	0.20

7.3 Parameters

We have varied a wide range of parameters. The parameters that we varied in our experiments are: (i) minimum acquaintance constraint c, (ii) number of meeting points mp, (iii) maximum allowable distance d_{max}, (iv) minimum group size n_{min}, and (v) solution list size, k.

For each parameter, we conducted experiments for different values of the parameters, keeping the other parameters at their default values, which were chosen empirically. The values of the parameters for which the experiments were conducted are summarized in Table 6.

7.4 Performance Evaluation

We conduct experiments by varying each parameter described in Table 6 and compare the performance of the three algorithms on the following two metrics:

- speed of execution: run-time,
- scalability: number of intermediate groups (nodes) explored during the search.

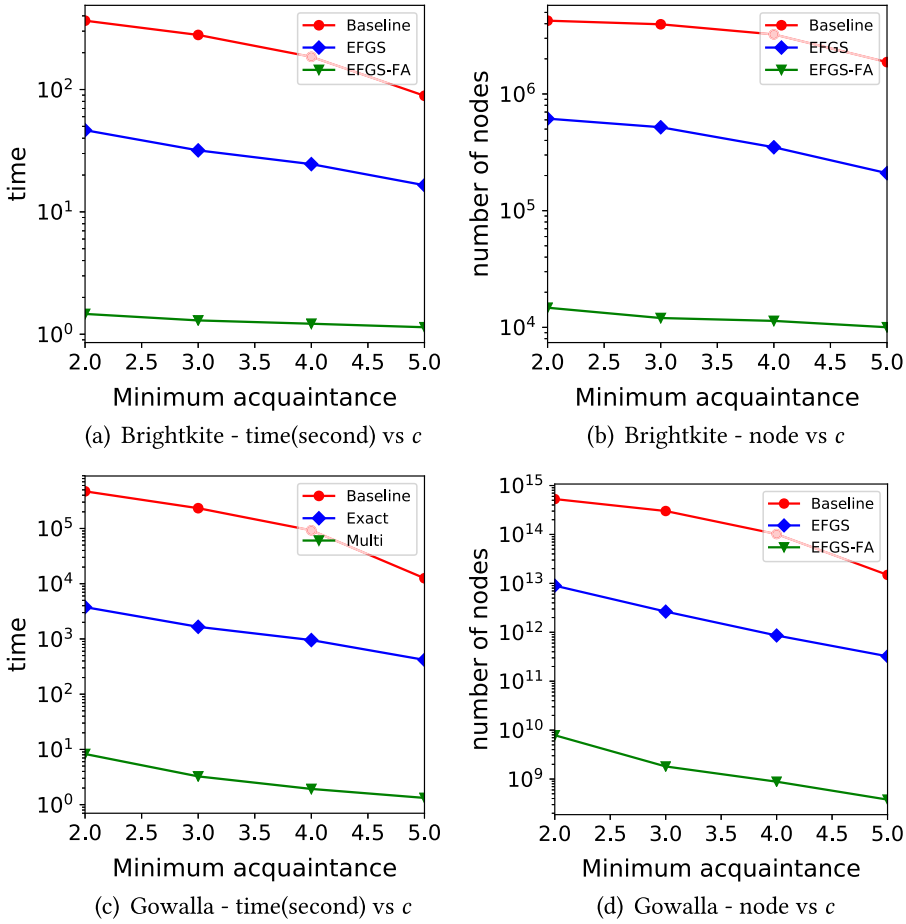
7.4.1 Varying the Minimum Social Connectivity c. c denotes the minimum number of members in a group with whom each member in that group needs to have social connectivity. As the value of c increases, the number of groups satisfying the social constraint decreases, because each member has only a limited number of *friends*. As a result, both the runtime and the number of nodes evaluated in the search tree decrease as c increases, as shown in Figure 3.

The EFGS algorithm is about one order of magnitude faster on the Brightkite dataset and two orders of magnitude faster on the Gowalla dataset than the baseline algorithm. Also, the number of nodes explored by our algorithm are 2 to 3 orders of magnitude less than that of the baseline for different values of c. As expected, EFGS-FA runs 1 to 2 orders of magnitude faster than our exact algorithm EFGA by sacrificing the quality of the group slightly.

7.4.2 Varying the Number of Meeting Points mp. As the intuition suggests, both the runtime and the number of groups evaluated increase as the number of meeting points increases. Nevertheless, EFGS achieves a speedup of one order of magnitude compared to the baseline method and explores 2 orders of magnitude fewer nodes in the search tree on the Brightkite dataset. We observe a similar performance gain with Gowalla dataset (Figure 4).

We also observe that he EFGS-FA algorithm is 20 times and 1,000 times faster in Brightkite and Gowalla datasets, respectively.

7.4.3 Varying the Maximum Allowable Distance d_{max}. We observe that the runtime for executing SSKGQ increases rapidly with the increase in d_{max} as more members satisfy the spatial distance constraint to become eligible to be included in a candidate solution and the size of the search tree grows. However, the curves for runtime and the number of nodes visited for both EFGS and EFGS-FA are much less steep because of the efficient pruning techniques.



(a) Brightkite - time(second) vs c

(b) Brightkite - node vs c

(c) Gowalla - time(second) vs c

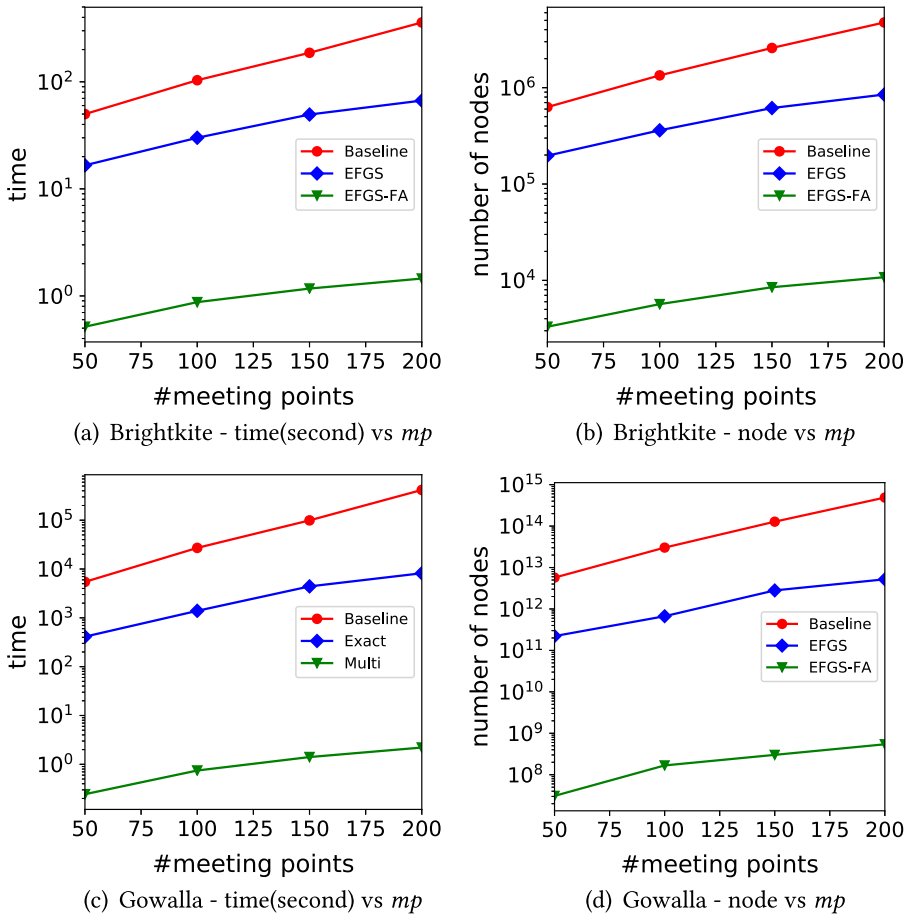
(d) Gowalla - node vs c

Fig. 3. Varying c.

For higher values of d_{max}, the superiority of these algorithms over the baseline method is evident, as shown in Figure 5. Compared to the baseline, EFGS is 1 to 2 orders of magnitude faster and EFGS-FA achieves a speedup of 2 to 4 orders of magnitude.

7.4.4 Varying the Minimum Group Size n_{min}. Similar to varying d_{max}, the cost of operation rapidly increases with n_{min}. For lower values of n_{min}, EFGS clearly outperforms the baseline (around 10× faster on Brightkite and 100× faster on Gowalla). However, their performances become somewhat similar for higher values of n_{min}. EFGS-FA, however, consistently outperforms both these algorithms (2 orders of magnitude faster than the baseline on Brightkite and 2 orders of magnitude faster on Gowalla). Figure 6 shows the effect of varying n_{min}.

7.4.5 Varying the Solution List Size k. The size of the solution list seems to have little impact on the runtime or number of nodes evaluated for a particular algorithm. Similar to varying all the other parameters, the EFGS achieves considerable speedup (3 times on Brightkite, 100 times on Gowalla) over the baseline. As expected, EFGS-FA is even faster with a speedup of orders of magnitude of 1 and 4 on Brightkite and Gowalla datasets, respectively. The curves for the number of nodes evaluated follow similar trend (Figure 7).



(a) Brightkite - time(second) vs *mp*

(b) Brightkite - node vs *mp*

(c) Gowalla - time(second) vs *mp*

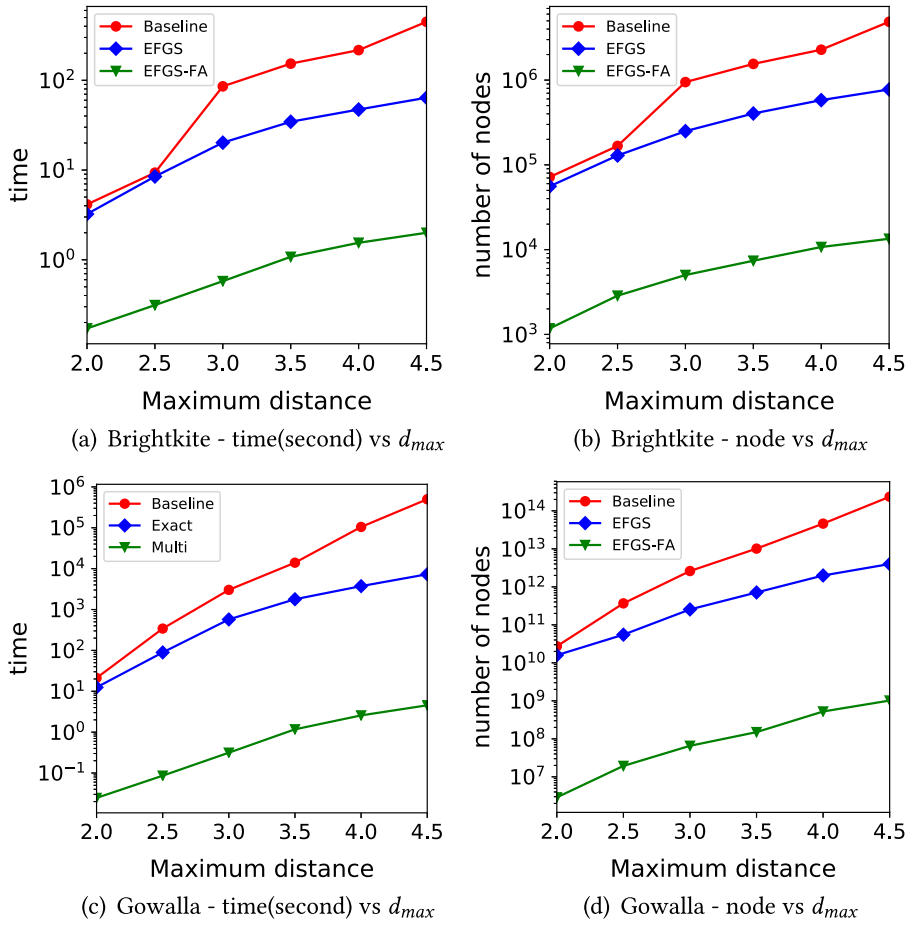
(d) Gowalla - node vs *mp*

Fig. 4. Varying *mp*.

7.4.6 Varying Weights w_1, w_2, w_3, w_4, w_5. How the weights of the components in the score function are set depends on the querists and on what factors they prioritize. We conducted experiments on various combinations of these weights and observed similar trends in the graphs. Hence, we report the results on only the default weight combination for the sake of conciseness of the article and making the graphs more presentable. However, the effect of varying the weights on the quality of the retrieved groups is discussed in Section 7.5.

7.4.7 Findings from the Experiments. It is apparent from the experimental results that the EFGS algorithm beats the baseline by 1–3 orders of magnitude both in terms of execution time and the number of nodes explored. The EFGS-FA algorithm is even more efficient, achieving 2–3 orders of magnitude performance boost over EFGS, but at the cost of slightly compromising the quality of the solution list, with occasional retrieval of suboptimal groups.

In our experiments, ordering based on $|N_v^{V_I}|$ resulted in slightly better performance over ordering based on Γ_v (Section 5.3.2). Therefore, we present the performance graphs of EFGS and EFGS-FA algorithms based on social connectivity ordering regarding V_I.



(a) Brightkite - time(second) vs d_{max}

(b) Brightkite - node vs d_{max}

(c) Gowalla - time(second) vs d_{max}

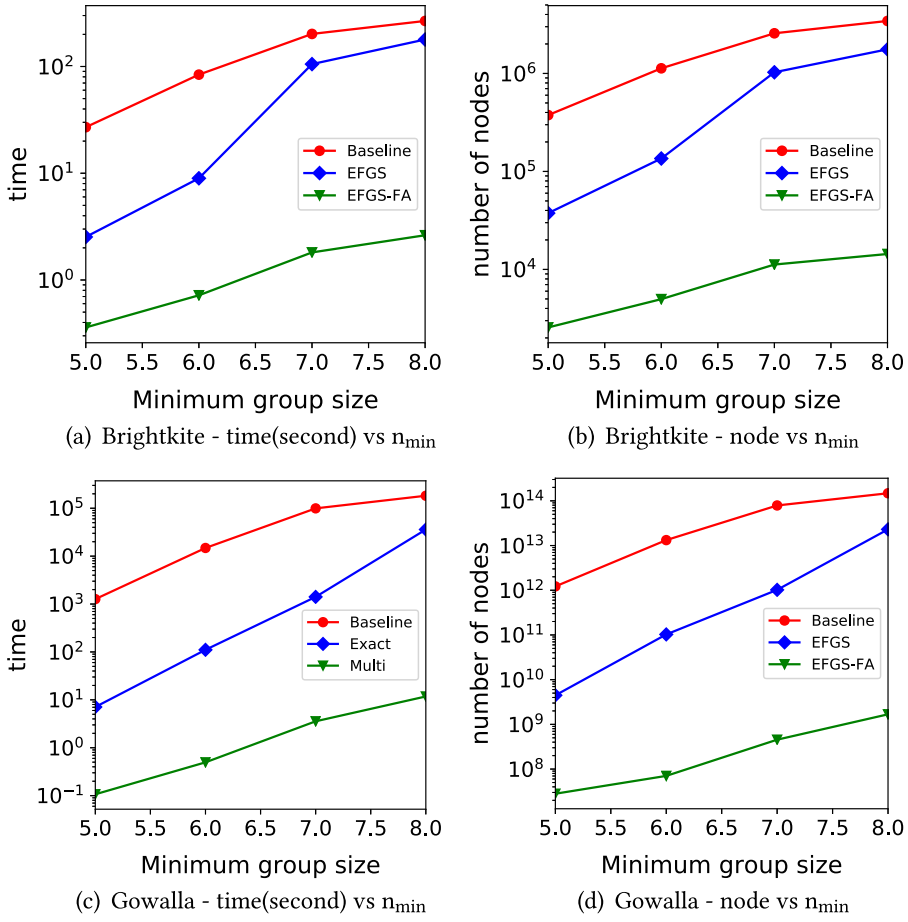
(d) Gowalla - node vs d_{max}

Fig. 5. Varying d_{max}.

7.5 Quality of the Groups Retrieved by EFGS-FA

From the results of the experiments on real datasets, it is evident that the EFGS-FA method is far superior to the baseline and the exact EFGS method both in terms of time complexity and scalability. However, the quality (score) of the groups in the solution list returned by EFGS-FA may be inferior. The exhaustive search method (i.e., the baseline) and the EFGS always return the best k groups in the social network. However, for the EFGS-FA method, we have stricter bounds compared to EFGS, which may result in early termination of search in a potential optimum branch of the search tree.

As a measure of the quality of an algorithm, we have selected the ratio of the average score of the groups in its solution list and the average score of the actual best k groups in the network. Since both the exhaustive search method and the EFGS methods are optimal, the score ratios will always be 1, i.e., 100%. This might not be always the case for EFGS-FA. A score ratio close to 100% indicates that an algorithm performs well in terms of quality, and the scores of the groups returned by the algorithm are almost identical to the actual best groups. A lower ratio is, however, an indicator of the poor quality of the resultant group.



(a) Brightkite - time(second) vs n_{min} (b) Brightkite - node vs n_{min}

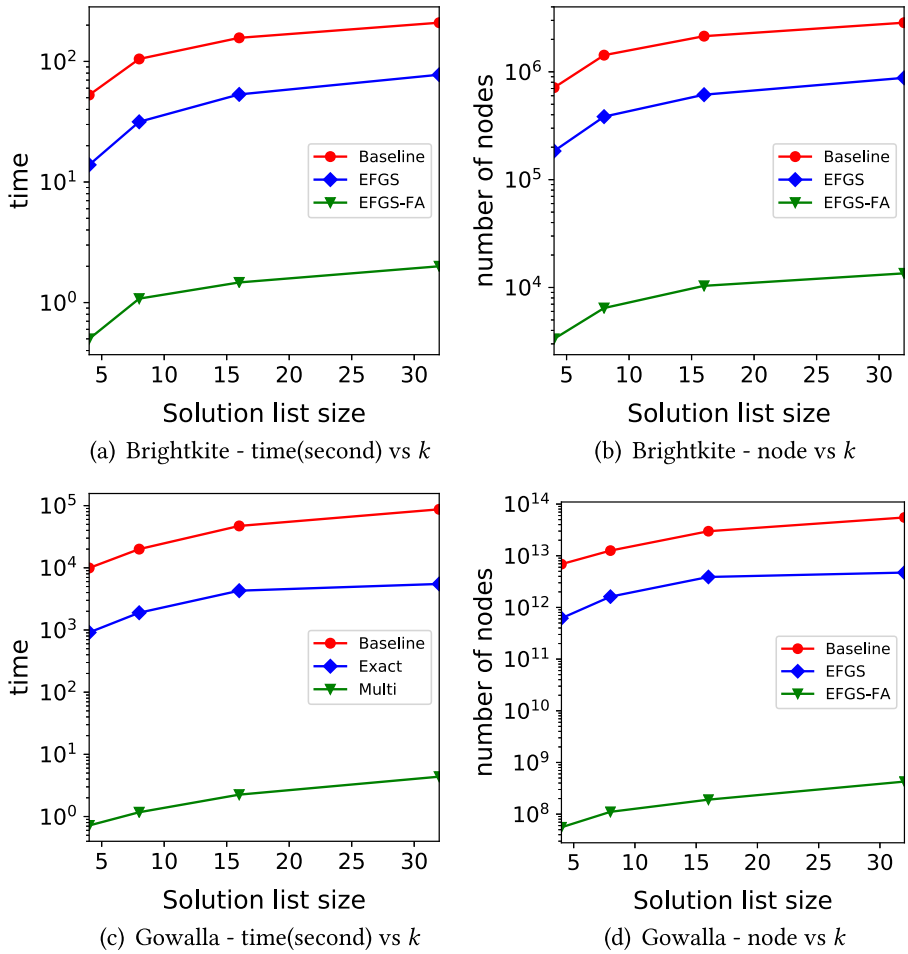
(c) Gowalla - time(second) vs n_{min} (d) Gowalla - node vs n_{min}

Fig. 6. Varying n_{min}.

We conducted experiments for various combinations of weights of S_{sc}, S_{sp}, $S_{kw(u,v)}$, $S_{kw(v,q)}$, and S_{sz} (i.e., various combinations of w_i; $i \in [1, 5]$) and the score ratio are presented in Figures 8 and 9. Figure 8 shows the score ratio for the default weight combination ($w_i = 0.2$; $i \in [1, 5]$) for both the Brightkite and the Gowalla dataset. Figure 9 shows the score ratio for various other combinations of the weights for the Brightkite dataset. For the sake of conciseness, we have omitted the results for the Gowalla dataset in Figure 9 as they show similar trends.

In our experiments, the EFGS-FA algorithm performed excellent for the Brightkite dataset, and reasonably well for the Gowalla dataset.

In the Brightkite dataset, the EFGS-FA algorithm performs almost to perfection for most practical weight-combinatios, significantly better than its performance in Gowalla. This might be a consequence of Brightkite being a relatively smaller dataset compared to Gowalla, with a lower number of valid members eligible for forming a group around each POI. This makes the search space relatively smaller, and the branches of the search tree relatively shorter. Now, if we revisit our strategy for the EFGS-FA algorithm, then we are reminded that the further down we go along a branch in the search tree, the more number of members we add (or discard) at a time to (or from) the intermediate solution group. In other words, the longer a branch gets, the higher the number



(a) Brightkite - time(second) vs k

(b) Brightkite - node vs k

(c) Gowalla - time(second) vs k

(d) Gowalla - node vs k

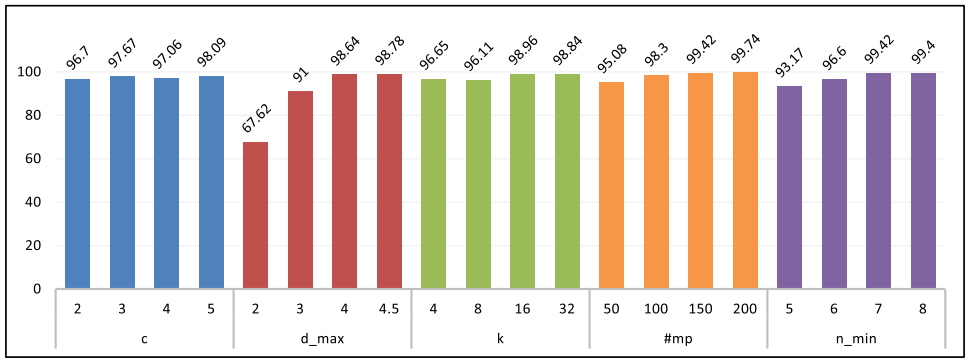
Fig. 7. Varying k.

of discarded candidate solution groups becomes. Thus, we evaluate a relatively lower percentage of total valid groups in longer branches, which explains the relatively inferior performance of the EFGS-FA algorithm in the Gowalla dataset compared to its performance in the Brightkite dataset.

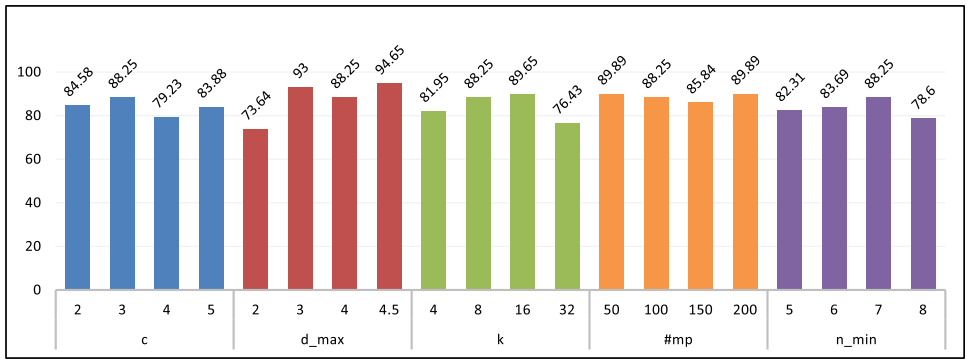
The performance of EFGS-FA is comparatively worse when d_{max} is very low. This is because setting the maximum allowable distance to a low value for the members rules out many members from forming a group. EFGS-FA even returns less than k groups on some occasions if d_{max} is set extremely low (for instance. when $d_{max} = 2$).

We see that the FEGS-FA algorithm is quite robust and, in general, performs well across different weight combinations. The performance worsens slightly for extreme conditions such as when $w_2 = 1$, i.e., all the emphasis is on distance score (Figure 9(c)), which is highly unlikely in real-world scenarios. However, EFGS-FA scores a 100% in fixed-sized cliques and where the social connectivity is of utmost importance (Figure 9(e)), as we had theoretically concluded in Section 6.2. Altogether, the quality of the groups retrieved by EFGS-FA is well enough for practical purposes.

We have omitted the use of traditional ranked list comparison methods such as the Rank-Biased Overlap [28] and the Kendall Rank Correlation Coefficient [16], because these methods compare



(a) Brightkite



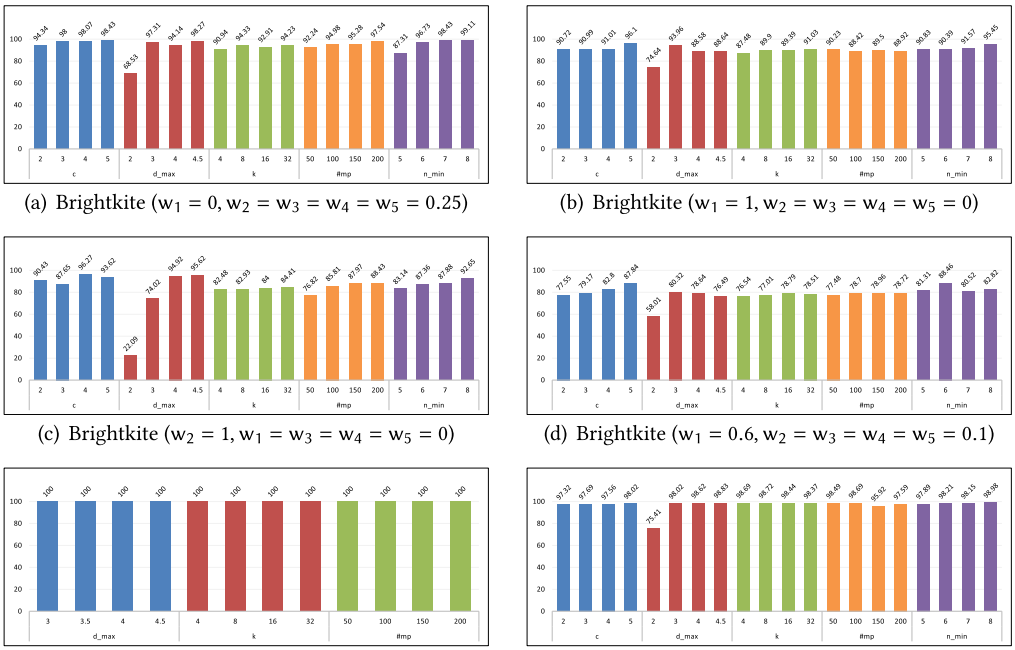
(b) Gowalla

Fig. 8. Approximation ratio for EFGS-FA ($w_1 = w_2 = w_3 = w_4 = w_5 = 0.2$).

only the actual elements themselves in the rank-list and give no emphasis on the scores of the objects. However, in our rank-list, it is possible that none of the best k groups in the social network is present in the rank-list returned by EFGS-FA, but their scores may be very close. Moreover, in large datasets such as Brightkite and Gowalla, it is possible that multiple groups with the same score exist. Hence, the exact algorithm and the approximate algorithm may even report different groups with the same score. Both the Rank-based Overlap and the Kendall Rank Correlation Coefficient methods would label such a solution list as completely unreasonable by assigning a score of 0 (zero) to it. But in practice, these groups are good enough for practical applications.

8 CONCLUSION AND FUTURE WORK

In this article, we are the first to propose a novel query, *Top-k Flexible Social-Spatial-Keyword Aware Group Query (Top-k SSKGQ)*, that considers social, spatial, and keywords while forming flexible size groups in social networks. We devise efficient branch-and-bound algorithms: EFGS and EFGS-FA for processing the *Top-k SSKGQ*. In these algorithms, we incorporate various strategies such as Distance Ordering, Spatial-Keyword Ordering, Distance Pruning, and Social-Spatial-Keyword Ordering to prune the unnecessary search space and obtain the solution efficiently. Our proposed algorithms use the IR-Tree as an indexing structure and perform significantly better than the baseline search method. While the EFGS algorithm provides optimal results in a fraction of time required by state-of-the-art techniques, the EFGS-FA algorithm has even faster performance, the



(a) Brightkite ($w_1 = 0, w_2 = w_3 = w_4 = w_5 = 0.25$)

(b) Brightkite ($w_1 = 1, w_2 = w_3 = w_4 = w_5 = 0$)

(c) Brightkite ($w_2 = 1, w_1 = w_3 = w_4 = w_5 = 0$)

(d) Brightkite ($w_1 = 0.6, w_2 = w_3 = w_4 = w_5 = 0.1$)

(e) Brightkite ($c_{min} = 4, n_{min} = n_{max} = 5, w_1 = 1, w_2 = w_3 = w_4 = w_5 = 0$)

(f) Brightkite ($w_3 = w_4 = 0.5, w_1 = w_2 = w_5 = 0$)

Fig. 9. Approximation ratio for EFGS-FA at different weight combinations for Brightkite.

trade-off being the possibility of providing suboptimal results. We confirm our claims by conducting experiments on real datasets to prove the superiority of our proposed algorithms.

ACKNOWLEDGMENTS

This work is done at DataLab (datalab.buet.io), Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET).

REFERENCES

[1] Mohammed Eunus Ali, Egemen Tanin, Peter Scheuermann, Sarana Nutanong, and Lars Kulik. 2016. Spatial consensus queries in a collaborative environment. *ACM Trans. Spatial Algor. Syst.* 2, 1 (2016), 3:1–3:37. https://doi.org/10.1145/2829943

[2] Marc Barthélemy. 2011. Spatial networks. *Phys. Rep.* 499, 1 (2011), 1–101. https://doi.org/10.1016/j.physrep.2010.11.002

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, and Alec Radford. 2020. Language Models are Few-Shot Learners. Retrieved from https://arXiv:cs.CL/2005.14165.

[4] Xin Cao, Gao Cong, Christian S. Jensen, and Beng Chin Ooi. 2011. Collective spatial keyword querying. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'11)*. ACM, New York, NY, 373–384. https://doi.org/10.1145/1989323.1989363

[5] Lisi Chen, Gao Cong, Christian S. Jensen, and Dingming Wu. 2013. Spatial keyword query processing: An experimental evaluation. *Proc. VLDB Endow.* 6, 3 (Jan. 2013), 217–228. https://doi.org/10.14778/2535569.2448955

[6] Lu Chen, Chengfei Liu, Rui Zhou, Jianxin Li, Xiaochun Yang, and Bin Wang. 2018. Maximum co-located community search in large scale social networks. *Proc. VLDB Endow.* 11, 10 (June 2018), 1233–1246. https://doi.org/10.14778/3231751.3231755

[7] X. Chen, C. Zhang, Y. Hu, B. Ge, and W. Xiao. 2016. Temporal social network: Group query processing. In *Proceedings of the 27th International Workshop on Database and Expert Systems Applications (DEXA'16)*. 181–185. https://doi.org/10.1109/DEXA.2016.047

[8] Eunjoon Cho, Seth A. Myers, and Jure Leskovec. 2011. Friendship and mobility: User movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'11)*. ACM, New York, NY, 1082–1090. https://doi.org/10.1145/2020408.2020579

[9] Blerim Cici, Athina Markopoulou, Enrique Frias-Martinez, and Nikolaos Laoutaris. 2014. Assessing the potential of ride-sharing using mobile and social data: A tale of four cities. In *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp'14)*. ACM, New York, NY, 201–211. https://doi.org/10.1145/2632048.2632055

[10] I. De Felipe, V. Hristidis, and N. Rishe. 2008. Keyword search on spatial databases. In *Proceedings of the IEEE 24th International Conference on Data Engineering*. 656–665. https://doi.org/10.1109/ICDE.2008.4497474

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Retrieved from https://arXiv:cs.CL/1810.04805.

[12] Yixiang Fang, Reynold Cheng, Xiaodong Li, Siqiang Luo, and Jiafeng Hu. 2017. Effective community search over large spatial graphs. *Proc. VLDB Endow.* 10, 6 (Feb. 2017), 709–720. https://doi.org/10.14778/3055330.3055337

[13] Yixiang Fang, Reynold Cheng, Siqiang Luo, and Jiafeng Hu. 2016. Effective community search for large attributed graphs. *Proc. VLDB Endow.* 9, 12 (Aug. 2016), 1233–1244. https://doi.org/10.14778/2994509.2994538

[14] Bishwamittra Ghosh, Mohammed Eunus Ali, Farhana Murtaza Choudhury, Sajid Hasan Apon, Timos Sellis, and Jianxin Li. 2018. The flexible socio spatial group queries. *Proc. VLDB* 12, 2 (2018), 99–111. Retrieved from http://www.vldb.org/pvldb/vol12/p99-ghosh.pdf.

[15] Norio Katayama and Shin'ichi Satoh. 1997. The SR-tree: An index structure for high-dimensional nearest neighbor queries. *SIGMOD Rec.* 26, 2 (June 1997), 369–380. https://doi.org/10.1145/253262.253347

[16] M. G. Kendall. 1938. A new measure of rank correlation. *Biometrika* 30, 1/2 (1938), 81–93. Retrieved from http://www.jstor.org/stable/2332226.

[17] Junghoon Kim, Tao Guo, Kaiyu Feng, Gao Cong, Arijit Khan, and Farhana M. Choudhury. 2020. Densely connected user community and location cluster search in location-based social networks. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'20)*. Association for Computing Machinery, New York, NY, 2199–2209. https://doi.org/10.1145/3318464.3380603

[18] Theodoros Lappas, Kun Liu, and Evimaria Terzi. 2009. Finding a team of experts in social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'09)*. ACM, New York, NY, 467–476. https://doi.org/10.1145/1557019.1557074

[19] Cheng-Te Li and Man-Kwan Shan. 2010. Team formation for generalized tasks in expertise social networks. In *Proceedings of the IEEE 2nd International Conference on Social Computing (SOCIALCOM'10)*. IEEE Computer Society, Washington, DC, 9–16. https://doi.org/10.1109/SocialCom.2010.12

[20] Y. Li, R. Chen, L. Chen, and J. Xu. 2017. Towards social-aware ridesharing group query services. *IEEE Trans. Services Comput.* 10, 4 (July 2017), 646–659. https://doi.org/10.1109/TSC.2015.2508440

[21] Yang Li, Feifei Li, Ke Yi, Bin Yao, and Min Wang. 2011. Flexible aggregate similarity search. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'11)*. Association for Computing Machinery, New York, NY, 1009–1020. https://doi.org/10.1145/1989323.1989429

[22] Zhisheng Li, Ken C. K. Lee, Baihua Zheng, Wang-Chien Lee, Dik Lee, and Xufa Wang. 2011. IR-Tree: An efficient index for geographic document search. *IEEE Trans. Knowl. Data Eng.* 23, 4 (Apr. 2011), 585–599. https://doi.org/10.1109/TKDE.2010.149

[23] Dimitris Papadias, Qiongmao Shen, Yufei Tao, and Kyriakos Mouratidis. 2004. Group nearest neighbor queries. In *Proceedings of the 20th International Conference on Data Engineering (ICDE'04)*. IEEE Computer Society, Washington, DC, 301. Retrieved from http://dl.acm.org/citation.cfm?id=977401.978090.

[24] Nick Roussopoulos, Stephen Kelley, and Frédéric Vincent. 1995. Nearest neighbor queries. *SIGMOD Rec.* 24, 2 (May 1995), 71–79. https://doi.org/10.1145/568271.223794

[25] Jieming Shi, Nikos Mamoulis, Dingming Wu, and David W. Cheung. 2014. Density-based place clustering in geo-social networks. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'14)*. ACM, New York, NY, 99–110. https://doi.org/10.1145/2588555.2610497

[26] Shi-Jie Chen and Li Lin. 2004. Modeling team member characteristics for the formation of a multifunctional team in concurrent engineering. *IEEE Trans. Eng. Manage.* 51, 2 (May 2004), 111–124. https://doi.org/10.1109/TEM.2004.826011

[27] Alastair J. Walker. 1977. An efficient method for generating discrete random variables with general distributions. *ACM Trans. Math. Softw.* 3, 3 (Sept. 1977), 253–256. https://doi.org/10.1145/355744.355749

[28] William Webber, Alistair Moffat, and Justin Zobel. 2010. A similarity measure for indefinite rankings. *ACM Trans. Info. Syst.* 28, 4, Article 20 (Nov. 2010), 38 pages. https://doi.org/10.1145/1852102.1852106

[29] Dingqi Yang, Daqing Zhang, Zhiyong Yu, and Zhu Wang. 2013. A sentiment-enhanced personalized location recommendation system. In *Proceedings of the 24th ACM Conference on Hypertext and Social Media (HT'13)*. ACM, New York, NY, 119–128. https://doi.org/10.1145/2481492.2481505

[30] Dingqi Yang, Daqing Zhang, Zhiyong Yu, and Zhiwen Yu. 2013. Fine-grained preference-aware location search leveraging crowdsourced digital footprints from LBSNs. In *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp'13)*. ACM, New York, NY, 479–488. https://doi.org/10.1145/2493432.2493464

[31] De-Nian Yang, Yi-Ling Chen, Wang-Chien Lee, and Ming-Syan Chen. 2011. On social-temporal group query with acquaintance constraint. *Proc. VLDB Endow.* 4, 6 (Mar. 2011), 397–408. https://doi.org/10.14778/1978665.1978671

[32] De-Nian Yang, Chih-Ya Shen, Wang-Chien Lee, and Ming-Syan Chen. 2012. On socio-spatial group query for location-based social networks. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'12)*. ACM, New York, NY, 949–957. https://doi.org/10.1145/2339530.2339679

[33] Qijun Zhu, Haibo Hu, Cheng Xu, Jianliang Xu, and Wang-Chien Lee. 2017. Geo-social group queries with minimum acquaintance constraints. *VLDB J.* 26, 5 (Oct. 2017), 709–727. https://doi.org/10.1007/s00778-017-0473-6

[34] Armen Zzkarian and Andrew Kusiak. 1999. Forming teams: An analytical approach. *IIE Trans.* 31, 1 (Jan. 1999), 85–97. https://doi.org/10.1023/A:1007580823003

Received July 2020; revised April 2021; accepted July 2021