Personalized Geo-Social Group Queries in Location-Based Social Networks

Yuliang Ma[1(✉)], Ye Yuan[1(✉)], Guoren Wang[2], Xin Bi[3], and Yishu Wang[1]

[1] School of Computer Science and Engineering, Northeastern University, Shenyang, China
ylma.neuer@gmail.com, yuanye@mail.neu.edu.cn

[2] School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China

[3] Sino-Dutch Biomedical and Information Engineering School, Northeastern University, Shenyang, China

Abstract. Geo-social group query, one of the most important issues in LBSNs, combines both location and social factors to generate useful computational results, which is attracting increasing interests from both industrial and academic communities. In this paper, we propose a new type of queries, *personalized geo-social group* (PGSG) queries, which aim to retrieve both a user group and a venue. Specifically, a PGSG query intends to find a group-venue pattern (consisting of a venue and a group of users with size h), where each user in the group is socially connected with at least c other users in the group and the maximum distance of all the users in the group to the venue is minimized. To tackle the problem of the PGSG query, we propose GVPS, a novel search algorithm to find the optimal user group and venue simultaneously. Moreover, we extend the PGSG query to top-k personalized geo-social group (TkPGSG) query. Instead of finding the optimal solution in the PGSG query, the TkPGSG query is to return multiple feasibility solutions to guarantee the diversity. We propose an advanced search algorithm TkPH to address the TkPGSG query. Comprehensive experimental results demonstrate the efficiency and effectiveness of our proposed approaches in processing the PGSG query and the TkPGSG query on large real-world datasets.

1 Introduction

With the progress of location acquisition and wireless communication technology, people now are able to add location dimension into traditional social networks, which fosters a bunch of location-based social networks (LBSNs), such as, *Foursquare*, *Gowalla*, and *Yelp*. People can easily record and share their life experiences via their mobile devices in these service platforms. Therefore, individuals' location data and social data have been readily available from mobile devices. One of the most important applications in LBSNs, *geo-social group query*, combines both location and social factors to generate useful computational results, which is attracting increasing interests from both industrial and academic communities.

J. Pei et al. (Eds.): DASFAA 2018, LNCS 10827, pp. 388–405, 2018.
https://doi.org/10.1007/978-3-319-91452-7_26

In the literature of geo-social group queries, the authors in [11,23] aim to find a group of users close to a given rally point and to ensure that the selected users have a good social relationship. The authors in [22] aim to find the activity time and attendees with the minimum total social distance to the initiator. The authors in [6,7] explore a group of experts whose skills can cover all the requirements and the communication cost among group members is low. The authors in [9] retrieve a user group of size k where each user is interested in the query keywords and they are close to each other in the Euclidean space. Besides, some other types of geo-social group queries have been proposed, such as, geo-social k-cover group query [8], and geo-social group query with minimum acquaintance constraint [28]. While being useful in some applications, these queries mentioned above do not completely utilize new search potential brought by geo-social data.

The following scenario is very common in real life. Assume that user Alice wants to establish an activity (such as, a group of users P including Alice will have dinner together at a venue s). There are some constraints by satisfied: (1) the size of the group $|P|$ is 10. Each user in the group should know at least 3 other users, which can create good atmosphere in the activity; and (2) the maximum distance of all the users in the group to the venue is minimized. However, none of the existing works on geo-social group queries can be used to answer such a scenario. For example, the authors in [23] propose a novel query SSGQ to find a set of users close to a given venue. While, Alice (the query user) does not know such a input venue and wants to find a user group and a venue simultaneously. If we use the SSGQ query to model the above scenario, we need to address the SSGQ queries repeatedly, which is extremely expensive. Thus, the SSGQ query is unsuitable for modeling the above scenario.

Consequently, we propose a novel type of geo-social group queries, called Personalized Geo-Social Group (PGSG) queries. Specifically, a PGSG query intends to find a venue and a user group (including the query user) with size h, where each user in the group is socially connected with at least c other users in the group and the maximum distance of all the users in the group to the venue is minimized. We call such a pair of user group and venue as a *group-venue pattern*. Our proposed PGSG query can model the above mentioned scenario. The size constraint h is 10 and the social topology constraint c is 3. By modeling the scenario as a PGSG query, Alice can obtain a appropriate group-venue pattern (consisting of a user group with size 10 and a venue), which each user in the group is connected with at least 3 other users and the maximum distance of all the users in the group to the venue is minimized. Besides, the PGSG query can be used to model some other real applications, such as spatial task outsourcing [18–20], event planning [3,16,17].

Moreover, we extend the PGSG query to the top-k personalized geo-social group (TkPGSG) query. Instead of finding the optimal group-venue pattern defined in the PGSG query, the TkPGSG query is to return k group-venue patterns $\mathbb{X} = \{X_1, \cdots, X_k\}$ such that: (a) all the k group-venue patterns satisfy the social constraints (group size constraint and the social topology constraint); (b) any group-venue pattern $X_m \notin \mathbb{X}$ satisfying the social constraints has a cost

(the maximum distance of all the users in the group to the venue) that exceeds that of any group-venue pattern $X_i \in \mathbb{X}$.

Challenges. The PGSG query is a hard problem to be tackled. The challenge is threefold.

Firstly, the PGSG query aims to find a user group (including the query user) with size h, where each user is socially connected with at least c other users in the group. If we directly extract all such groups by simply enumerating all possible combinations, the search space is large and redundant. Therefore, the first challenge is how to extract possible user groups efficiently.

Secondly, there are infinitely many user group combinations and venues, which make infeasibility to examine all group-venue patterns to the PGSG query. Therefore, the second challenge is how to efficiently find the optimal group-venue pattern for the PGSG query.

Thirdly, in order to guarantee the diversity of query processing, one common and effective way is to return multiple query results. We extend the PGSG query to the TkPGSG query. It is inefficient to tackle the TkPGSG query by invoking multiple PGSG queries. Thus, the third challenge is how to efficiently return top-k group-venue patterns for the query user.

Our Proposed Methods. In order to tackle the PGSG query efficiently, we propose a novel search algorithm, called *group-venue pattern search* (GVPS). The intuition of GVPS is that we expand a user group from the query user by a breadth search strategy. With the group expanding processing, GVPS reduces the venue search space by a derived lower bound and upper bound of spatial distance. Moreover, based on the GVPS algorithm, we propose an advanced top-k personalized geo-social group query algorithm, namely *top-k group-venue patterns hunter* (TkPH), to tackle the TkPGSG query efficiently.

Contributions. To summarize, we make following contributions in this paper.

- We propose a new type of geo-social group queries called Personalized Geo-Social Group (PGSG). Specifically, a PGSG query aims to find a venue and a group with size h, where each user in the group is socially connected with at least c other users in the group and the sum of the distance from every user in the group to the venue is minimized.
- We extend the PGSG query to the top-k personalized geo-social group (TkPGSG) query. Instead of finding the optimal group-venue pattern defined in PGSG, the TkPGSG query is to return multiple query results to guarantee the diversity of query processing.
- To tackle the problem of PGSG query, we propose a search algorithm, called *group-venue pattern search* (GVPS). Besides, we propose an advanced top-k group-venue patterns search algorithm, namely TkPH, for processing the TkPGSG query.
- Extensive experiments are conducted to demonstrate the efficiency and effectiveness of the proposed approaches on real-world datasets.

The rest of this paper is organized as follows. We formally define the PGSG query and the TkPGSG query in Sect. 2. We present the details of PGSG query algorithm in Sect. 3. In Sect. 4, we present the proposed algorithm to process the TkPGSG query. We show an extensive experimental evaluation in Sect. 5, and overview the related works in Sect. 6. In Sect. 7, we conclude this paper.

2 Problem Formulation

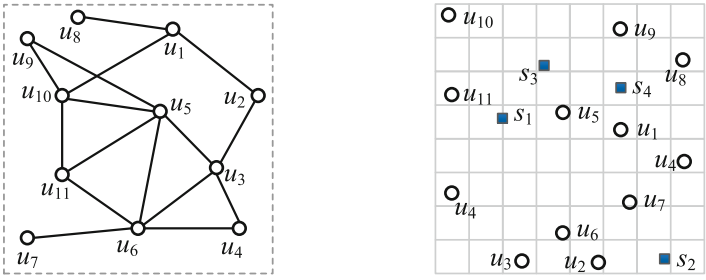
In this section, we describe the terms and notations that we use throughout the paper, and formally define the Personalized Geo-social Group (PGSG) queries.

LBSNs allow users to search location-tagged contents within their social graphs, and consist of the new social structures made up of individuals. We call the integration of location data and social data, generated by individuals in LBSNs, as *geo-social data*.

We model a LBSN as two main components. As shown in Fig. 1(a), the first component is the underlying social network $G = (V, E)$, where each vertex $u \in V$ is a user. Each edge $e \in E$ denotes an acquainted relation (e.g., friendship) between the two users it connects. Moreover, each user u maintains a pair of coordinates indicating the user's location, which can be extracted from the user's profiles (such as home address) or discovered by the existing work [27]. We take the user's location as one part of the input, since user location extraction is not the focus of our work. In this paper, we take the venues in LBSNs as spatial objects. Thus, the second component of a LBSN is a set of spatial venues S. Each venue $s \in S$ is a spatial object associated with a pair of coordinates indicating its geographical position. Figure 1(b) shows an example of the spatial locations of users and venues in a LBSN.

A PGSG query aims to find a user group (including the query user) with size h, where each user in the group is socially connected with at least c other users in the group.

Definition 1 ***(c-core).*** *For a graph $G = (V, E)$, a maximal connected subgraph $G' = (V', E')$ of G is a* c-core, *if each vertex $u \in V'$ has degree at least c.*



(a) Social network (b) Spatial location

Fig. 1. An example of a location-based social network.

The concept of c-core was first proposed by Seidman in [15], which can be widely applied to describe the complex topologies of social networks and reveal the hierarchical structures of networks.

Definition 2 ***(Induced Subgraph).*** *Given a graph $G = (V, E)$, for any subset V' of V, and edge set E':*

$$E' = \{(u, v) | u, v \in V' \text{ and } (u, v) \in E\},$$

we call $G' = (V', E')$ is an induced subgraph of V' in G, denoted as $G[V']$.

Definition 3 ***(Connected c-core Component).*** *Given a graph $G = (V, E)$, a subset V' of V, $G[V']$ is a connected c-core component, if $G[V']$ is a connected component and $\min_{u \in V'} deg_{G[V']}(u) \geq c$.*

Definition 4 ***(Group-Venue Pattern).*** *Given a user group $P = \{u_1, u_2, \cdots, u_h\}$, a venue s, we call such a pair of user group and venue a group-venue pattern, denoted as $X = (P, s)$. The distance of a group-venue pattern X is the maximum distance of all the users in $X.P$ to $X.s$. That is, we have:*

$$Dist(X) = Dist(X.P, X.s) = Max_{u_i \in P} dist(u_i, s) \tag{1}$$

where $dist(u_i, s)$ is the Euclidean distance.

Remember that a *personalized geo-social group* (PGSG) query aims to find a venue and a user group (including the query user) with size h, where the group is a connected c-core component and the maximum distance of all the users in the group to the venue is minimized. Formally, we define the PGSG query as follows:

Definition 5 ***(Personalized Geo-Social Group (PGSG) Query).*** *Given an underlying social network $G = (V, E)$ and a spatial venue set $S = \{s_1, s_2, \cdots, s_m\}$, the personalized geo-social group query $q = \langle u_q, h, c \rangle$, where u_q is the query user who initiate such a query, h is the group size constraint, and c indicates a social constraint c-core, aims to retrieve a user group $P \subseteq V$ and a venue $s \in S$ such that:*

(1) P includes u_q and $|P| = h$;
(2) $G[P]$ is a connected c-core component;
(3) $Dist(P, s)$ is minimized.

Example 1. Take Fig. 1 as an example, assume a PGSG query $q = \langle u_5, 4, 2 \rangle$, which means the query user is u_5, the group size constraint is 4. The user group $P = \{u_5, u_9, u_{10}, u_{11}\}$ contains 4 users including u_5, and the induced subgraph P in G is a 2-core. In this condition, the group-venue pattern $X = (P, s_3)$ is the result of q, since $Dist(P, s_3)$ is minimized.

In order to guarantee the diversity of query processing, we extend the PGSG query to the top-k personalized geo-social group (TkPGSG) query. Instead of finding the optimal group-venue pattern defined in the PGSG query, the TkPGSG query aims to find k group-venue patterns. Formally, we define the TkPGSG query as follows:

Definition 6 ***(Top-k Personalized Geo-Social Group (TkPGSG) Query).*** *Given an underlying social network $G = (V, E)$ and a spatial venue set $S = \{s_1, s_2, \cdots, s_m\}$, the top-k personalized geo-social group query $q = \langle u_q, h, c, k\rangle$, where u_q is the query user who initiate such a query, h is the group size constraint, c indicates a social constraint c-core, and k the number of group-venue patterns needed to return, aims to retrieve k group-venue patterns $\mathbb{X} = \{X_1, X_2, \cdots, X_k\}$ such that:*

(1) for any $X_i \in \mathbb{X}$, X_i includes u_q and $|X_i.P| = h$;
(2) for any $X_i \in \mathbb{X}$, $G[X_i.P]$ is a connected c-core component;
(3) for any $X_i \in \mathbb{X}$, and any $X_j \notin \mathbb{X}$ such that $X_j.P$ satisfies (1) and (2), $Dist(X_i) \leq Dist(X_j)$.

If the number of group-venue patterns satisfying the conditions (1) and (2) in Definition 6, the TkPGSG query returns all these group-venue patterns. It's worth noting that the TkPGSG query can also be extended to other metrics, such as average distance and minimal distance.

Example 2. Continue to use Fig. 1 as an example. Assume a TkPGSG query $q = \langle u_5, 4, 2, 3\rangle$, which means the query user is u_5, the group size constraint is 4, the core constraint is 2, q aims to find 3 group-venue patterns. There are four user groups: $P_1 = \{u_5, u_3, u_4, u_6\}$, $P_2 = \{u_5, u_3, u_6, u_{11}\}$, $P_3 = \{u_5, u_6, u_{10}, u_{11}\}$, and $P_4 = \{u_5, u_9, u_{10}, u_{11}\}$, where each group satisfies the condition (1) and (2) presented in Definition 6. The results of q are: $X_1 = (P_1, s_1)$, $X_2 = (P_2, s_1)$, and $X_3 = (P_4, s_3)$. The user group P_3 can not be included in the results, since the distance of any group-venue pattern X_j containing P_3 exceeds the third-largest distance in the results.

3 PGSG Query Processing

In this section, we discuss how to process the PGSG query. We firstly introduce a baseline algorithm, and then propose a novel search algorithm.

3.1 Baseline Algorithm

According to our problem statement, a PGSG query aims to find a venue s and a user group P (including the query user u_q) with size h, where the induced subgraph $G[P]$ is a connected c-core component and the maximum distance of all the users in P to s is minimized. To process the PGSG query, a baseline solution can be readily described as follows. Firstly, we enumerate all the c-core groups including the query user with size h. We regard these c-core groups as group candidates (called GCS). To find a feasible venue s_i for each group $P_i \in GCS$, we utilize the method proposed in [12], which aims to find the aggregate nearest neighbor for a given query point set in spatial databases. We take the group-venue pattern $X_i = (P_i, s_i)$ as a candidate solution. Finally, the group-venue pattern with the minimum distance is returned as the result.

The methods proposed in [12] to address the aggregate nearest neighbor query are based on a spatial access method, R-tree [5]. In order to index social relations in LBSNs, a new concept, core bounding rectangles (CBRs), has been proposed in [8,28]. Based on this concept, SaR-tree structure [28] and enhanced SaR-tree [8] are proposed to facilitate social relations processing. But they both relies on inputs of corresponding geo-social group queries, which are unsuitable for our PGSG queries. In this paper, we utilize the R-tree in all our proposed algorithms to speed up spatial distance computation.

3.2 Group-Venue Pattern Search

Apparently, the baseline algorithm is time-consuming and impractical. It is inefficient to enumerate all c-core groups including the query user with size h, since most of these groups are not the final result. Besides, it is expensive and time consuming to find a feasible venue for each above enumerated group. Thus, we propose a novel search algorithm, called *group-venue pattern search* (GVPS).

The intuition of the GVPS algorithm can be present as follows. GVPS expands a user group from the query user by a breadth search strategy. With the group expansion processing, GVPS reduces the venue search space by a derived lower bound and upper bound of spatial distance. And then, a user group and a venue can be retrieved simultaneously by a novel group-venue pattern search method.

Group Expansion. When a PGSG query $q = \langle u_q, h, c\rangle$ is initiated, we firstly adopt a core decomposition algorithm to obtain the c-core G_c. And then, we take the query user u_q as a center to do breadth search in G_c. The size of a user group increases with the breadth expansion. In order to accelerate processing, we select multiple vertices in one breadth expansion. In each expansion, we select the vertex with maximum degree in the current group as the new center for next expansion. When the size of a group P_i reaches h, we check whether the induced subgraph $G[P_i]$ is a connected c-core component. If yes, P_i must be a connected c-core component and contain the query user, since we start to expand a user group from the query user and select the neighbors of the center in each expansion.

Take the social graph G in Fig. 1 as an example. Assume a PGSG query $q = \langle u_{11}, 4, 2\rangle$. Figure 2 shows an example of group expansion. Figure 2(a) shows the 2-core of G. We start to do breadth search from the query user u_{11}. We select at least 2 neighbors of u_{11}. The expansions from v_{11} are shown in Fig. 2(b). As the size constraint h is 4 in q, b_1 has already finished expansion. In a user group (such as b_2) needed to be extended, we selected the node with maximum degree as new center to do next group expansion. The size of b_2 is 3, we can add only *one* user into the current user group. The user groups extended from b_2 are shown in Fig. 2(c).

Group-Venue Pattern Search. Before illustrating the produce of group-venue pattern search method, we derive the upper bound and the lower bound of the distance of a group-venue pattern X. Note that X consists of a user group

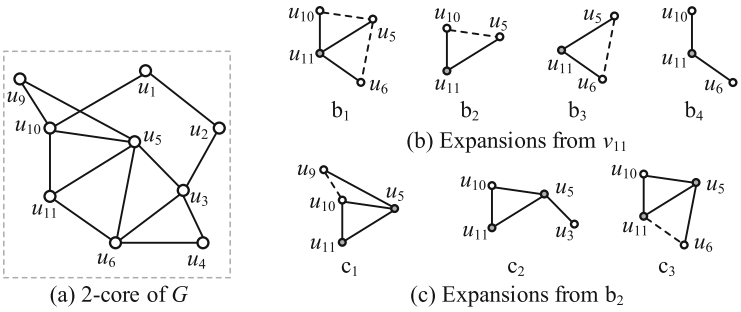


Fig. 2. Example of group expansion.

and a venue. According to the problem definition of the PGSG query, one of the query constraint is that the returned user group in a pattern should include the query user. Thus, in the rest of this section, discussions are restricted to a user group including the query user, unless otherwise stated.

Theorem 1. *Given a user group P including the query user u_q, and a venue s, the* ***upper bound*** *of the maximum distance of a user in P to s is $dist(u_q, s) + \max_{u_i \in P} dist(u_i, u_q)$.*

Proof. Suppose the maximum distance of any user in P to s is the distance of a user u_0 to s, denoted as $dist(u_0, s)$. As the user group P includes the query user u_q, we have $dist(u_0, s) < dist(u_q, s) + dist(u_0, u_q)$ by the triangle inequality. This theorem is proved, since $dist(u_0, u_q) \leq \max_{u_i \in P} dist(u_i, u_q)$.

Theorem 2. *Given a user group P including the query user u_q, and a venue s, the* ***lower bound*** *of the maximum distance of a user in P to s, $Dist(P, s)$, is $\max\{dist(u_q, s), \max_{u_i \in P} dist(u_i, u_q) - dist(u_q, s)\}$.*

Proof. As the user group P includes the query user u_q, the maximum distance of a user in P to a venue s is at least the distance of u_q to s, i.e., $dist(u_q, s) \leq Dist(P, s)$. Suppose the maximum distance of any user in P to s is the distance of a user u_0 to s, by the triangle inequality, we have $|dist(u_q, s) - dist(u_q, u_0)| < Dist(P, s)$. If $dist(u_q, s) \geq dist(u_q, u_0)$, we have $|dist(u_q, s) - dist(u_q, u_0)| < dist(u_q, s) < Dist(P, s)$. If $dist(u_q, s) < dist(u_q, u_0)$, we have $dist(u_q, u_0) - dist(u_q, s) < Dist(P, s)$. And than, $\max_{u_i \in P} dist(u_i, u_q) - dist(u_q, s)\} < Dist(P, s)$ can be derived by the triangle inequality. Thus, we take $\max\{dist(u_q, s), \max_{u_i \in P} dist(u_i, u_q) - dist(u_q, s)\}$ as the lower bound of $Dist(P, s)$.

Now, the GVPS algorithm is ready to be presented. GVPS maintains the current minimum distance of the current optimal group-venue pattern X^*. By leveraging the lower bound (denoted as $LDist(P, s)$) and the upper bound (denoted as $UDist(P, s)$) of the distance of a group-venue pattern $X = (P, s)$, the search space can be reduced. In order to retrieve the user group and the

Algorithm 1. Group-Venue Pattern Search (GVPS)

Input: A graph $G = (V, E)$, $S = \{s_1, s_2, \cdots, s_m\}$, a PGSG query $q = \langle u_q, h, c \rangle$
Output: A group-venue pattern $X^* = (P^*, s^*)$

1 Initialize $X^* \leftarrow \emptyset$;
2 Initialize $Dist(X^*) \leftarrow \infty$;
3 Initialize priority queue $UQ \leftarrow u_q$, $DQ \leftarrow \emptyset$, $DV \leftarrow \emptyset$;
4 Find the c-core G_c of G;
5 **while** $UQ \neq \emptyset$ **do**
6 $P_i \leftarrow UQ.dequeue()$;
7 **if** $|P_i| < h$ **then**
8 Select the node $v \in P_i$ with highest degree and $v \notin DV$;
9 $DV \leftarrow DV \cup v$;
10 **for** *each set* $Vp \subseteq N(v)/P_i$, $c \leq |Vp| + d_{G[P_i]}(v)$ ***and*** $|Vp| + |P_i| \leq h$ **do**
11 induce a subgraph G_i of $P_i \cup Vp$ in G_c;
12 **if** $G_i \not\subset UQ \cup DQ$ **then**
13 $UQ \leftarrow UQ \cup G_i$;
14 **if** $|P_i| = h$ *and* $P_i \not\subset DQ$ **then**
15 $DQ \leftarrow DQ \cup P_i$;
16 **if** $\{v \in P_i : c > |deg_{G[P_i]}(v)|\} \neq \emptyset$ **then**
17 Continue;
18 Generate a venue candidate set S_i for P_i;
19 $Dist(X^*) \leftarrow Dist(P_i, s_0)$;
20 **for** *each venue* $s_j \in S_i$ **do**
21 **if** $LDist(P_i, s_j) \geq Dist(X^*)$ **then**
22 Continue;
23 **if** $UDist(P_i, s_j) < Dist(X^*)$ ***or*** $Dist(P_i, s_j) < Dist(X^*)$ **then**
24 Update $P^* \leftarrow P_i$, $s^* \leftarrow s_j$;
25 Return X^*;

venue of X^* simultaneously, GVPS retrieves the venues along with the user group expansion processing. That is, when the size of a user group P_i reaches h, GVPS utilizes the lower bound to prune venues. GVPS prunes the pattern $X_i = (P_i, s_j)$, if $LDist(X_i) \geq Dist(X^*)$. On the other hand, GVPS leverages the upper bound to check whether the group can emerge in the final query result. If $UDist(X_i) < Dist(X^*)$, GVPS updates the current optimal solution without more distance computation. Let s_0 be the nearest venue of u_q. Instead of taking the whole venues in a LBSN as the search space for a user group P_i, we generate a venue candidate set S_i. That is, we adopt $Dist(P_i, s_0)$ to do a range query at each user in P_i. The venues in these query ranges are regarded as venue candidates. By this way, we can largely reduce the venue search space.

Algorithm 1 details the procedure of the GVPS algorithm. At the beginning, GVPS initially sets the final optimal group-venue pattern X^* to $\emptyset$ and its distance $Dist(X^*)$ to ∞ (lines 1–2). GVPS maintains two priority queues UQ

and DQ. UQ stores user groups that have not been extended. DQ manages the user groups that have already been extended. Besides, the vertices in G are stored in DV if they are impossible to be included in any other user groups except these already lie in UQ and DQ. GVPS exploits c-core to prune users in the original social graph (line 4). And then, GVPS leverages the above-mentioned group expansion to search user groups (lines 6–13).

When the size of a user group P_i reaches h, GVPS checks whether the induced subgraph of P_i is a connected c-core component (lines 14–17). If yes, GVPS generates a venue candidate set S_i for P_i and update $Dist(X^*)$ (lines 18–19). Here, we only update the value of $Dist(X^*)$ rather than X^*, since an earlier given $Dist(X^*)$ can bring a better pruning effectiveness. And then, GVPS utilizes the lower bound to reduce the search space (lines 21–22). That is, a group-venue pattern (P_i, s_j) can be pruned directly, if $LDist(P_i, s_j) \geq Dist(X^*)$. Next, the upper bound can be used to speed up the update of current optimal solution (lines 23–24). Final, the optimal group-venue pattern X^* is returned as the result of the PGSG query.

The following example illustrates how Algorithm 1 works.

Example 3. Continue to use Fig. 1 as an example. Assume a PGSG query $q = \langle u_{11}, 4, 2\rangle$. GVPS starts to do group expansions from u_{11} (as shown in Fig. 2). When the size of a user group P reaches 4 (suppose $P = \{u_5, u_9, u_{10}, u_{11}\}$), GVPS executes multiple range queries to generate a venue candidate set. Referring to Fig. 1 again, s_1 is the nearest venue of u_{11}. GVPS utilizes $Dist(P, s_1)$ (i.e., $dist(u_9, s_1)$) to do range query at each user in P. We can see that s_2 does not lie in any query range. Thus s_2 can pruned directly about P. Finally, the group-venue pattern (P, s_3) is returned as the optimal result of q.

Complexity Analysis. According to Algorithm 1, we can analyze the time complexity of GVPS algorithm from the following three aspects. Let a given graph $G = (V, E)$ and a PGSG query $q = \langle u_q, h, c\rangle$. Firstly, the time complexity of *core decomposition* $O(|V| + |E|)$ by following the existing work [2]. Secondly, the time complexity of *group expansion* is $O(\bar{d}^h \cdot h^2)$, where $\bar{d}$ is the average degree of the vetices in G, $O(\bar{d}^h)$ is the time complexity of packing a user group with size h from u_q, and $O(h^2)$ is the time of estimating whether the induced subgraph of a user group with size h constitutes a c-core. Thirdly, the time complexity of once group-venue pattern search is $O(h^2 \cdot |S|)$, where $|S|$ is the number of venues in a LBSN. Overall the time complexity of the GVPS algorithm is $O(|V| + |E| + \bar{d}^h \cdot (h^2 + h^2 \cdot |S|))$.

4 TkPGSG Query Processing

As mentioned above, we extend the PGSG query to the top-k personalized geo-social group (TkPGSG) query. Instead of finding the optimal group-venue pattern defined in the PGSG query, the TkPGSG query is to return k group-venue patterns $\mathbb{X} = \{X_1, \cdots, X_k\}$. Although the parameter k is a small positive integer, it is time-consuming and inefficient to invoke multiple Algorithm 1 for TkPGSG

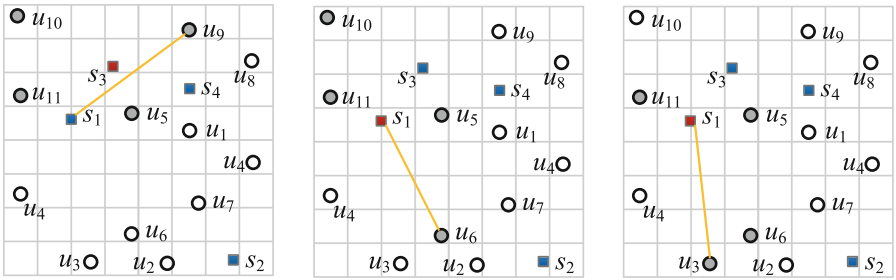
query processing. Thus, we propose an advanced search algorithm (top-k group-venue patterns hunter, TkPH) to tackle the TkPGSG query.

The TkPH algorithm is designed based on the GVPS algorithm presented in Subsect. 3.2. Thus, we only elaborate the group-venue pattern search phase, and the group expansion is the same with the GVPS algorithm.

TkPH maintains $\mathbb{X}$ to store the current best k group-venue patterns. In $\mathbb{X}$, the patterns are sorted in ascending order of their group-venue distance. Given a user group P including the query user u_q, and a venue s, let $Dist(X_k^*)$ be the distance of the k-th group-venue pattern, the group-venue pattern (P, s) can be pruned directly if $LDist(P, s) \geq Dist(X_k^*)$. On the other hand, the group-venue pattern (P, s) can be updated into the $\mathbb{X}$, if $UDist(P, s) < Dist(X_k^*)$. That is, we update $\mathbb{X}$ by replacing the k-th group-venue pattern with (P, s).

Once a user group P_i includes u_q with size h, and $G[P_i]$ is a connected c-core component, TkPH retrieves k nearest venues for P_i to constitute k group-venue patterns. By this way, an initial solution can be obtained quickly. An earlier viable solution can bring greater pruning performance. Consequently, the distance of the k-th group-venue pattern can be leveraged to reduce the search space.

The following example illustrates how this algorithm works. As the group expansion phase of TkPH algorithm is similar to that in the GVPS algorithm, thus we simply show how TkPH keeps tracking of the current best k results.



(a) Group-venue pattern X_1 (b) Group-venue pattern X_2 (c) Group-venue pattern X_3

Fig. 3. An example of the TkPH algorithm. (Color figure online)

Example 4. Take the social network in Fig. 1 as an example. Assume a TkPGSG query $q = \langle u_{11}, 4, 2, 2\rangle$. The group expansion procedure is shown in Fig. 2. As shown in Fig. 3(a), suppose $P_1 = \{u_5, u_9, u_{10}, u_{11}\}$ is the first user group that satisfies the size constraint and social core constraint, we take $\mathbb{X} = \{(P_1, s_3), (P_1, s_1)\}$ as an initial solution. And then, we utilize $Dist(P_1, s_1)$ to reduce the search space. As shown in Fig. 3(a), (b), and (c), we can find three group-venue patterns (consisting of the grey user nodes and the red venue in each figure). Finally, $\mathbb{X} = \{X_1, X_2\}$ is returned as the result of q, since $Dist(X_3) > Dist(X_2)$.

5 Experiments

In this section, we experimentally study the performance of the proposed approaches. We perform a series of sensitivity tests to study the impact of query parameters with real-world datasets. In the following, we first present the experimental settings, and then analyze the experimental results.

5.1 Experimental Settings

Algorithms. We implement four various algorithms: BL, BL+GE, GVPS, and TkPH. Specifically, BL denotes the baseline algorithm presented in Subsect. 3.1. Note that, to find all the connected c-core components with size h, BL utilizes brute-force strategy to do user group expansion. Then, BL adopts the methods proposed in [12] to find a optimal venue for each extracted user group. BL+GE denotes the baseline algorithm with the group expansion strategy presented in Subsect. 3.2. GVPS denotes the group-venue search algorithm presented in Subsect. 3.2. TkPH is the top-k group-venue patterns hunter algorithm presented in Sect. 4. All algorithms are implemented in C++. All experiments are conducted on a computer with 3.20 GHz Intel i5-6500 CPU and 16 GB memory.

Dataset. We use three real world datasets, i.e., Gowalla, Brightkite, and Foursquare. Table 1 gives the details. The first two datasets are downloaded from (http://snap.stanford.edu/data/index.html). We crawled the Foursquare datasets via Foursquare API[1] from November 2014 to January 2016. This dataset has 76,503 users and 1,531,357 social edges. We obtained 299,995 venues located in Singapore.

Table 1. Some statistics of datasets

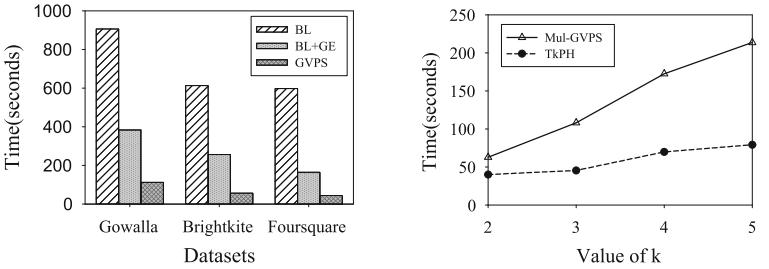
Datasets	Gowalla	Brightkite	Foursquare
# of venues	1,280,969	772,789	299,995
# of users	196,591	58,228	76,503
# of social edges	950,327	214,078	1,531,357

5.2 Experimental Results

Efficiency. The objective of this set of experiments is to study the efficiency of the proposed algorithms with different real datasets.

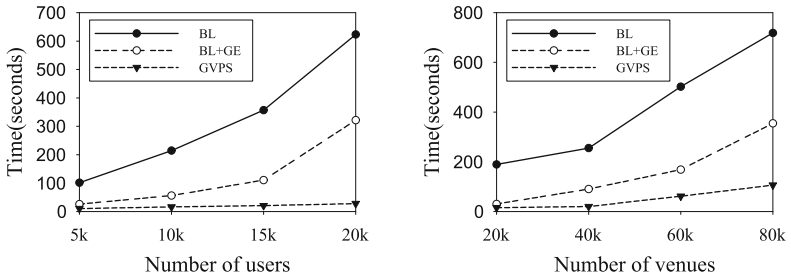
We first test the efficiency of BL, BL+GE, and GVPS algorithms to address the PGSG query. Note that the PGSG query can be model as $\langle u_q, h, c\rangle$, where u_q is the query user, h is the user group size constraint, and c is the core number. We set the value of size constraint h to be 8, c to be 3, and the degree of query user to be 9. Figure 4(a) shows the runtime (average time of 50 queries) of the three

[1] https://developer.foursquare.com/.



(a) Efficiency on PGSG query (b) Efficiency on TkPGSG query

Fig. 4. Efficiency



(a) Scalability vs. graph scale (b) Scalability vs venue scale

Fig. 5. Scalability

algorithms on the three real datasets. It can be seen that the BL+GE algorithm runs faster that the BL algorithm, which indicates that the group expansion strategy presented in Subsect. 3.2 can efficiently reduce the search space. The runtime is related to the number of venues and the size of social networks. The reason is that our aim is to find a group-venue pattern. The group extraction processing relies on the size of social network and the venue search depends on the number of venues.

To study the efficiency of TkPH algorithm to tackle the TkPGSG query, we construct a comparative algorithm by invoking multiple the GVPS algorithm, denoted as Mul-GVPS. With the same parameter setting of h, c, and degree of u_q in Fig. 4(a), the experimental results of the TkPH algorithm and the Mul-GVPS algorithm on Gowalla datasets are shown in Fig. 4(b). We observe that the TkPH algorithm runs faster than the Mul-GVPS algorithm. The reason is that TkPH algorithm keeps track of the current best k results rather than tracking only the current best result. The TkPH algorithm is proposed based on the GVPS algorithm. In the following experiments, we just show the performance of the algorithms on tackling the PGSG query.

Scalability. We study the effect of the scale of graph (vertex number) and the number of venues on the performance of our proposed algorithms, which can evaluate the scalability of our proposed algorithms.

By extracting the subgraph of the social graph in Gowalla datasets, we obtain 4 datasets containing 5,000, 10,000, 15,000, and 20,000 users, respectively. The corresponding induced graphs can be obtained. Figure 5(a) shows the runtime of the proposed algorithms (the size constraint h is 6, $c = 2$, and the degree of the query user is 6) with the number of users increasing. Analogously, by extracting the subset of the venues in Gowalla datasets, we obtain 4 datasets containing 20K, 40K, 60K, and 80K venues, respectively. Figure 5(b) shows the runtime of the proposed algorithms (the size constraint h is 6, $c = 2$, and the degree of the query user is 6) with the number of venues increasing. It can be seen that the relative ratio of GVPS algorithm changes only slightly, and the other two are opposite. Thus, the scalability of GVPS algorithm outperforms the other two algorithms.

Varying the Query Parameters. In this set of experiments, we study the effect of the three parameters on the performance of our proposed algorithms. Particularly, we regulate u_q for fixed coreness and size constraint pairs to estimate the effect of the degree of query user. Similarly, we evaluate the impact of core number c (or size constraint h) by fixing the other two parameters.

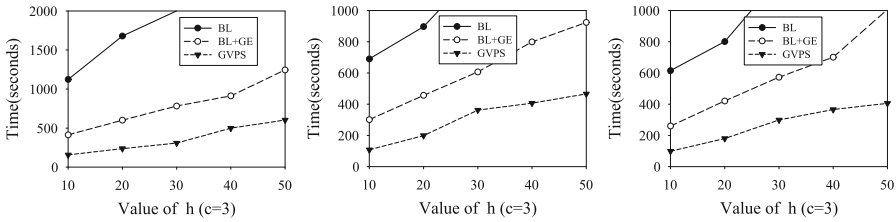
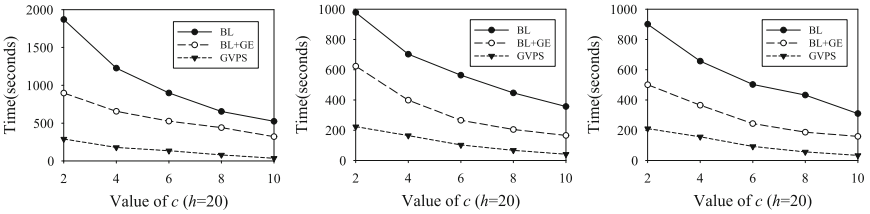


Fig. 6. Varying size constraint h

Varying h. In this set of experiments, we study the effect of h in the PGSG query. Let $c = 3$, we alter the size constraint h from 10 to 50. We regard the average runtime of 50 queries as the performance measure of our proposed algorithms, where the degree of query users is 9. Figure 6 shows the results on the three datasets.

For a fixed coreness and degree of query user, the three algorithms run slower with the size constraint h increasing. The reason is that a query with a larger size constraint h is more likely to have a greater search space, that is, all the algorithms are required to settle more group expansions.

Varying c. In this set of experiments, we study the effect of c in the PGSG query. Let the size constraint $h = 20$, we vary c from 2 to 10. We regard the average

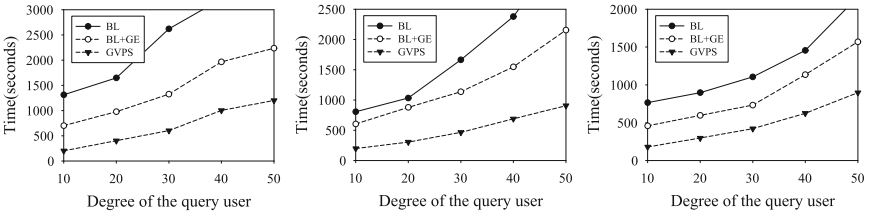


(a) Runtime (Gowalla) (b) Runtime (Brightkite) (c) Runtime (Foursquare)

Fig. 7. Varying c

runtime of 50 queries as the performance measure of our proposed algorithms, where the degree of query users is 9. The results on the three real datasets is shown in Fig. 7. The three algorithms run faster as the coreness c increases. The reason is that a query with a larger coreness c is more likely to have a stronger social constraint, which indicates that the query has fewer group expansions to process.

Varying the Query User (Degree). In this set of experiments, we study the effect of the degree with respect to the query user. Let the size constraint $h = 20$ and $c = 4$, we vary the query users with different degree from 10 to 50. Figure 8 shows the running time of the three algorithms over the three real datasets. For a fixed size constraint h and core constraint c pair, the three algorithms run slower with the degree of query user increasing. The reason is that a larger degree of query user leads to a larger search space.



(a) Runtime (Gowalla) (b) Runtime (Brightkite) (c) Runtime (Foursquare)

Fig. 8. Varying query user (degree)

6 Related Works

Our work is related to three main research areas: (1) group recommendation, (2) activity-partner recommendations, and (3) geo-social group query.

Group Recommendations. A group recommendation system suggests items to a group of users engaged in a group activity. By taking the different preferences of different group members into consideration, most of the studies on group

recommendations aims to recommend items for a given user group [1,10,13,14, 24,26]. Whereas, in this paper, we focus on retrieving a user group rather than research on a given user group.

Activity-Partner Recommendations. Tu *et al.* [21] propose and study the problem of recommending activity partners to Web uses for activity items suggested to them. Besides, they study the use of partner factor for improving activity recommendation. She *et al.* [16,17] propose a novel problem, utility-aware social event-participant planning (USEP), that provides personalized event planning for each participant. By taking the minimum-participant requirement constraint for each evnet, Cheng *et al.* [3] formalize the global event planning with constraints (GEPC) problem, and its incremental variant. Our work is different from this research area. Our work is not only to find a group of users (partners), but also to find a venue (activity item), which yields specified social constraint and spatial constraint in LBSNs.

Geo-Social Query Processing. With the increasing prevalence of location-based services, geo-social queries considering both spatial and social relations are attracting increasing attention [4,6–9,11,22,23,25,28]. The most related work to ours is [23], which study the Socio-Spatial Group Query (SSGQ): given a query venue s, the SSGQ returns a group users, such that (a) each user in the group is socially connected with at least a number of other members, and (b) the sum of distances of all members in the group to s is minimized. Our work is different from [23] in that, (a) the PGSG query aims to find both a venue and a user group simultaneously, (b) the user group is a connected c-core component with a specified size, and (c) the maximum distance of all the users in the group to the venue is minimized.

7 Conclusions

In this paper, we propose a new type of geo-social group query to find both a venue and a user group. We formally define the problem as *Personalized Geo-Social Group* (PGSG) query, which aims to find a group-venue pattern (consisting of a venue and a group of users with size h), where each user in the group is socially connected with at least c other users in the group and the maximum distance of all the users in the group to the venue is minimized. We study the problem of PGSG query and propose a novel search algorithm to find the optimal user group and venue simultaneously. Moreover, we extend the PGSG query to top-k personalized geo-social group (TkPGSG) query. Instead of finding the optimal solution in PGSG query, the TkPGSG query is to return multiple feasibility solutions to guarantee the diversity. We propose an advanced search algorithm TkPH to address the TkPGSG query. Extensive experimental results demonstrate the effectiveness and efficiency of the proposed approaches. A direction for future work is to investigate the issue of social trust and how to integrate social trust into geo-social group query.

Acknowledgments. This research is partially funded by the National Natural Science Foundation of China (No. 61572119, 61622202, U1401256, 61732003, 61729201, 61702086) and the Fundamental Research Funds for the Central Universities (No. N150402005).

References

1. Amer-Yahia, S., Roy, S.B., Chawlat, A., Das, G., Yu, C.: Group recommendation: semantics and efficiency. Proc. VLDB Endow. **2**(1), 754–765 (2009)
2. Batagelj, V., Zaversnik, M.: An O(m) algorithm for cores decomposition of networks. Comput. Sci. **1**(6), 34–37 (2003)
3. Cheng, Y., Yuan, Y., Chen, L., Giraud-Carrier, C., Wang, G.: Complex event-participant planning and its incremental variant. In: 2017 IEEE 33rd International Conference on Data Engineering, ICDE, pp. 859–870. IEEE (2017)
4. Fang, Y., Cheng, R., Li, X., Luo, S., Hu, J.: Effective community search over large spatial graphs. Proc. VLDB Endow. **10**(6), 709–720 (2017)
5. Guttman, A.: R-Trees: A Dynamic Index Structure for Spatial Searching, vol. 14. ACM, New York (1984)
6. Lappas, T., Liu, K., Terzi, E.: Finding a team of experts in social networks. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 467–476. ACM (2009)
7. Li, C.T., Shan, M.K.: Team formation for generalized tasks in expertise social networks. In: IEEE Second International Conference on Social Computing, pp. 9–16 (2010)
8. Li, Y., Chen, R., Xu, J., Huang, Q., Hu, H., Choi, B.: Geo-social k-cover group queries for collaborative spatial computing. IEEE Trans. Knowl. Data Eng. **27**(10), 2729–2742 (2015)
9. Li, Y., Wu, D., Xu, J., Choi, B., Su, W.: Spatial-aware interest group queries in location-based social networks. Data Knowl. Eng. **92**, 20–38 (2014)
10. Li, Y.M., Chou, C.L., Lin, L.F.: A social recommender mechanism for location-based group commerce. Inf. Sci. **274**, 125–142 (2014)
11. Liu, W., Sun, W., Chen, C., Huang, Y., Jing, Y., Chen, K.: Circle of friend query in geo-social networks. In: Lee, S., Peng, Z., Zhou, X., Moon, Y.-S., Unland, R., Yoo, J. (eds.) DASFAA 2012. LNCS, vol. 7239, pp. 126–137. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29035-0_9
12. Papadias, D., Tao, Y., Mouratidis, K., Hui, C.K.: Aggregate nearest neighbor queries in spatial databases. ACM Trans. Database Syst. (TODS) **30**(2), 529–576 (2005)
13. Quijano-Sanchez, L., Recio-Garcia, J.A., Diaz-Agudo, B., Jimenez-Diaz, G.: Social factors in group recommender systems. ACM Trans. Intell. Syst. Technol. (TIST) **4**(1), 8 (2013)
14. Quijano-Sanchez, L., Sauer, C., Recio-Garcia, J.A., Diaz-Agudo, B.: Make it personal: a social explanation system applied to group recommendations. Expert Syst. Appl. **76**, 36–48 (2017)
15. Seidman, S.B.: Network structure and minimum degree. Soc. Netw. **5**(3), 269–287 (1983)
16. She, J., Tong, Y., Chen, L.: Utility-aware social event-participant planning. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, pp. 1629–1643. ACM (2015)

17. She, J., Tong, Y., Chen, L., Cao, C.C.: Conflict-aware event-participant arrangement and its variant for online setting. IEEE Trans. Knowl. Data Eng. **28**(9), 2281–2295 (2016)
18. Tong, Y., Chen, L., Zhou, Z., Jagadish, H.V., Shou, L., Lv, W.: SLADE: a smart large-scale task decomposer in crowdsourcing. IEEE Trans. Knowl. Data Eng. (2018). https://doi.org/10.1109/TKDE.2018.2797962
19. Tong, Y., She, J., Ding, B., Wang, L., Chen, L.: Online mobile micro-task allocation in spatial crowdsourcing. In: 2016 IEEE 32nd International Conference on Data Engineering, ICDE, pp. 49–60. IEEE (2016)
20. Tong, Y., Wang, L., Zhou, Z., Ding, B., Chen, L., Ye, J., Xu, K.: Flexible online task assignment in real-time spatial data. Proc. VLDB Endow. **10**(11), 1334–1345 (2017)
21. Tu, W., Cheung, D.W., Mamoulis, N., Yang, M., Lu, Z.: Activity recommendation with partners. ACM Trans. Web (TWEB) **12**(1), 4 (2017)
22. Yang, D.N., Chen, Y.L., Lee, W.C., Chen, M.S.: On social-temporal group query with acquaintance constraint. Proc. VLDB Endow. **4**(6), 397–408 (2011)
23. Yang, D.N., Shen, C.Y., Lee, W.C., Chen, M.S.: On socio-spatial group query for location-based social networks. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 949–957 (2012)
24. Yuan, Q., Cong, G., Lin, C.Y.: COM: a generative model for group recommendation. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 163–172. ACM (2014)
25. Yuan, Y., Lian, X., Chen, L., Sun, Y., Wang, G.: RSkNN: kNN search on road networks by incorporating social influence. IEEE Trans. Knowl. Data Eng. **28**(6), 1575–1588 (2016)
26. Zhang, C., Gartrell, M., Minka, T., Zaykov, Y., Guiver, J., et al.: GroupBox: a generative model for group recommendation (2015)
27. Zheng, Y., Zhang, L., Ma, Z., Xie, X., Ma, W.Y.: Recommending friends and locations based on individual location history. Acm Trans. Web **5**(1), 1–44 (2011)
28. Zhu, Q., Hu, H., Xu, C., Xu, J., Lee, W.C.: Geo-social group queries with minimum acquaintance constraints. VLDB J. **26**(5), 709–727 (2017)