Geo-Social K-Cover Group Queries for Collaborative Spatial Computing

Yafei Li, Rui Chen, Jianliang Xu, Qiao Huang, Haibo Hu, Byron Choi

Abstract—With the rapid development of location-aware mobile devices, ubiquitous Internet access and social computing technologies, lots of users' personal information, such as location data and social data, has been readily accessible from various mobile platforms and online social networks. The convergence of these two types of data, known as *geo-social data*, has enabled *collaborative spatial computing* that explicitly combines both location and social factors to answer useful *geo-social queries* for either business or social good. In this paper, we study a new type of *Geo-Social K-Cover Group* (GSKCG) queries that, given a set of query points and a social network, retrieves a minimum user group in which each user is socially related to at least k other users and the users' associated regions (e.g., familiar regions or service regions) can jointly cover all the query points. Albeit its practical usefulness, the GSKCG query problem is NP-complete. We consequently explore a set of effective pruning strategies to derive an efficient algorithm for finding the optimal solution. Moreover, we design a novel index structure tailored to our problem to further accelerate query processing. Extensive experiments demonstrate that our algorithm achieves desirable performance on real-life datasets.

Index Terms—Location-based services, query processing, group queries, social constraints

1 Introduction

With the rapid development of location-aware mobile devices, ubiquitous Internet access and social computing technologies, individuals' location data and social data have been readily available from smart phones and mobile platforms. The convergence of location data and social data, known as *geo-social data*, has enabled a new computing paradigm that explicitly combines both location and social factors to generate useful computational results for either business or social good. In this paper, we use the term *collaborative spatial computing* to represent this emerging paradigm. The idea of collaborative spatial computing has been widely used in various domains, including location-based social networks [14], geo-crowdsourcing [16], activity planning [35], group decision making [24], and disaster rescue [5]. One of the most important applications of collaborative spatial computing in the database field is *geo-social queries*, which are attracting increasing interest from both industrial and academic communities.

The study of geo-social queries is in its incipiency. The pioneering studies [10], [22], [35], [38] typically consider geo-social queries that take as inputs a set of mobile users, a query location point and certain social acquaintance constraint and that return a set of users with the minimum location distance while satisfying the social constraint. For example, a user can create a party invitation by issuing a query that returns a set of nearby users with relatively tight social relations. While being useful in some applications (e.g., activity planning), these queries do not fully exploit new search possibilities brought by geo-social data.

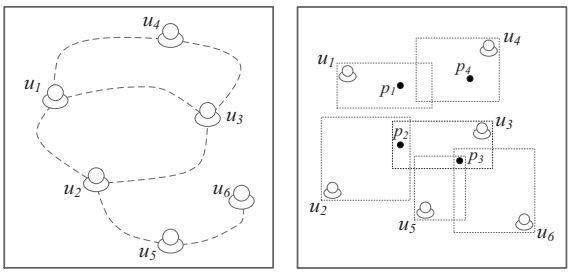
In this paper, we propose a novel type of geo-social queries, called *Geo-Social K-Cover Group* (GSKCG) queries, which is based on *spatial containment* and a new modeling of social relationships. Intuitively, given a set of spatial query points and an underlying social network, a GSKCG query finds a minimum user group in which the members satisfy certain social relationship and their associated regions can jointly cover all the query points. Figure 1 shows an example of the GSKCG query where u_1, u_3, u_4 form a minimum group with tight social relations and their associated regions jointly cover all the query points p_1, p_2, p_3, p_4. Such GSKCG queries are useful for a wide range of applications. We provide several motivating examples below.

- *Travel recommendation*: To recommend a self-drive tour of a few points of interest (POIs) (e.g., [1], [2]), a GSKCG query helps to find a minimal group of tourists who are collectively familiar with these POIs (e.g., in terms of weather, accommodation safety, road conditions, and traffic laws) so as to reduce accident risks and who have relatively tight social relations in order to make the tour more trustful and more harmonious. The

- *Y. Li, R. Chen, J. Xu, H. Hu and B. Choi are with the Department of Computer Science, Hong Kong Baptist University, Kowloon Tong, Hong Kong SAR, China. Email: {yafeili, ruichen, xujl, haibo, bchoi}@comp.hkbu.edu.hk.*

- *Q. Huang is with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China and the Department of Computer Science, Hong Kong Baptist University, Kowloon Tong, Hong Kong SAR, China. Email: tkdsheep@zju.edu.cn.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2419663, IEEE Transactions on Knowledge and Data Engineering



(a) Social networks (b) Associated regions

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

minimum group size makes it easier for all group members to reach a consensus in subsequent planning.

- *Spatial task outsourcing*: Given a set of spatial tasks, each associated with a spatial location, one needs to distribute them to a set of workers, each having a service region. To successfully accomplish the tasks, the service regions of the selected workers should cover all spatial tasks' locations, and the workers are expected to have good collaborative relationships so that the tasks can be efficiently performed. A GSKCG query directly addresses this worker selection problem in spatial task outsourcing. In practice, the size of the group of selected workers should be minimum to minimize employment cost.
- *Collaborative team organization*: GSKCG queries are useful for marketing and promotion agencies. For example, in an agency, each agent has several familiar market areas and several good collaborators. If a company wants to hire a marketing team to promote its products in some market areas, a GSKCG query finds a good team that covers all promotion locations and that is cohesive while causing the minimum cost for the company. As another example, a community organization can resort to a GSKCG query to find a minimal group of investigators to conduct a questionnaire survey in several sites. The returned group will be jointly familiar with all the sites and have a good collaborative atmosphere in order to efficiently deliver, collect and analyze the questionnaires.

We formally define a GSKCG query to capture the natural search requirements driven by the real-life applications. GSKCG queries differ from the existing geo-social queries [10], [22], [35] in both the spatial and social factors. For the spatial factor, instead of finding a group of users close to the query points (e.g., spatial task sites or a rally point), a GSKCG query finds a user group whose associated regions (e.g., service regions or familiar regions) jointly cover a set of query points; for the social factor, we employ the more reasonable k-*core* notion to measure the intensity of the relationships of users in the selected group, for example, each user should be familiar with at least k other users. For this reason, the techniques developed for previous geo-social queries cannot be directly applied to our problem. Albeit its practical usefulness, the GSKCG query problem is an extremely challenging problem to tackle. Indeed, we prove that this problem is NP-complete. Therefore, designing an efficient algorithm to find the optimal solution requires non-trivial efforts.

Contributions. Our major contributions are summarized as follows:

- We formulate a new type of GSKCG queries, which is of practical usefulness in many real-life applications. Given a set of query points and a social network, a GSKCG query retrieves a minimum user group in which each user is connected to at least k other users and the users' associated regions can jointly cover all the query points. We formally prove that the problem is NP-complete.
- We propose the `KCGFinder` algorithm to answer GSKCG queries. We improve the performance of `KCGFinder` by exploring a set of effective pruning techniques from different perspectives.
- We design a novel index structure, the Enhanced Social-aware R-tree (SaR-tree), which encodes not only users' familiar spatial regions but also their social relations. This structure provides extra pruning capabilities on top of the pruning techniques developed for `KCGFinder`. We then propose the `SaRBasedKCGFinder` algorithm that integrates `KCGFinder` and the Enhanced SaR-tree structure.
- We conduct extensive experiments on real-life datasets and demonstrate that our proposed algorithm performs well under a wide range of parameter settings.

The rest of this paper is organized as follows. Section 2 formally formulates the problem and analyzes its complexity. Section 3 presents the `KCGFinder` algorithm along with a set of effective pruning techniques. Section 4 presents the Enhanced SaR-tree structure and introduces the integrated `SaRBasedKCGFinder` algorithm. Experimental results are provided in Section 5. We review the related works in Section 6. Finally, we conclude the paper and present possible future work in Section 7.

2 Problem Formulation

In this section, we give some preliminaries and provide the problem statement, followed by an example to elaborate the problem defined. Table 1 summarizes the notations used throughout this paper.

A GSKCG query is defined over a location-based social network (LBSN) $G = (V, E)$, where each vertex $u \in V$ is a user and each edge $e \in E$ denotes an acquainted relation between the two users it connects.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2419663, IEEE Transactions on Knowledge and Data Engineering

TABLE 1
Summary of notations

Notation	Definition
$G = (V, E)$	Location based Social Network (LBSN)
u, v	a user of G
$u.R$	an associated region of user u
$Q = (k, P)$	a GSCKG query Q, P is a set of query points, and k indicates a social constraint k-core
$G[V']$	a subgraph of G contains vertices V'
SI	an intermediate solution where $\|SI\| \leq s$
SU	the set of remaining users
P_S	the set of query points covered by users in S
$NB_S(v)$	the number of v's neighbors in S
V_p	a set of users whose associated region covers query point p
U^k	a set of users that may appear in a k-core
C_s	a group of size s
C^k	a connected k-core component
C_s^k	a s-size group where $G[C_s^k]$ is a k-core and C_s^k fully covers P
M	the maximum group size of query GSKCG
$List_P$	a sorted user list according to the increasing size of V_p where $p \in P$
$\mathcal{A}(p)$	the index of the last user u in $List_P$ where $p \in u.R$
$k(u)$	a k-core with u inside
$CBR_{u,k}$	a rectangle that does not contain a $k(u)$
$iCBR_{u,k}$	an internal CBR of u that does not contain a $k(u)$
$eCBR_{u,k}$	an external CBR of u that does not contain a $k(u)$
$MBP(P)$	a minimum bound rectangle which contains the query point set P

For any two users $u, v \in V$, there exists an edge $(u, v) \in E$ if and only if u and v are familiar with each other. Moreover, each user $u \in V$ has an associated region denoted by $u.R$.[1] Such an LBSN can be easily derived by combining the location and social data collected from real-life applications.

A GSKCG query aims to find a group of users with a desired social relationship. In this paper, we quantify the desire of the social relationship within a user group in terms of *k-core* [30], a widely used model for detecting community structures in a graph.

Definition 1: (k-core) For a graph $G = (V, E)$, a connected subgraph $G' = (V', E')$ of G is a *k-core* if every vertex $v \in V'$ has at least degree k. ■

We argue that k-core is a reasonable model to measure a user group's social acquaintance level for two main reasons. First, the minimum degree constraint of k-core is an important measure of group cohesiveness in social science research [30] and has been widely adopted in the research of graph problems [8], [28], [36]. In our problem, k-core is effective and flexible to capture a user group's acquaintance level in real-life LBSNs. Second, k-core decomposition has a linear time complexity, which makes it appealing in real-life applications. Indeed, it has been used as an important social constraint in practical applications [32]. Based on the k-core model, we formally define a GSKCG query below.

Definition 2: (GSKCG query) Given an LBSN $G = (V, E)$, a *Geo-Social k-Cover Group* (GSKCG) query is defined as a 2-tuple $Q = (k, P)$, where k is a positive integer, indicating the social acquaintance constraint, and $P = \{p_1, p_2, \cdots, p_m\}$ is a set of query points, indicating the spatial coverage constraint, and returns a set of users $V' \subseteq V$ such that:

1) $P \subset \bigcup_{u \in V'} u.R$,
2) the subgraph $G[V']$ of G is a k-core, and
3) the cardinality of $G[V']$ is minimum. ■

Note that we require the returned user group to have the minimum cardinality. This requirement is naturally derived from the real-world demands. For example, in the motivating examples in Section 1, retrieving a minimum set of users normally leads to the minimum employment cost or ease of reaching a consensus. We choose to make k as an input parameter in order to provide a generic geo-social query service for different service requesters. For a service requester that aims to find a single user who covers all the tasks, he/she can set $k = 0$, which allows the GSKCG query to consider only the spatial containment constraint, but not the social constraint. In many other cases, setting k to a non-zero value will provide much more flexibility for a requester. For example, a service requester can issue multiple GSKCG queries with different k values in parallel and then select a proper group that fits his/her business needs.

Example 1: Consider a simple LBSN $G = (V, E)$ where the users' acquaintance relations and associated regions are shown in Figure 1(a) and Figure 1(b), respectively. The GSKCG query $Q = (k, P)$ with k = 2 and $P = \{p_1, p_2, p_3, p_4\}$ returns the user group $V' = \{u_1, u_3, u_4\}$ because: 1) the joint regions of users in V' can cover all the query points in P; 2) the subgraph $G[V']$ of G is a 2-core; and 3) the cardinality of $G[V']$ is minimum among all user groups that satisfy the first two conditions. ■

As formally defined in Definition 2, a GSKCG query finds a set of users that satisfy the given spatial and social constraints. For ease of presentation, we call a user group *valid* if it satisfies both Conditions 1 and 2 in Definition 2. Next we analyze the complexity of the GSKCG query problem.

Theorem 1: GSKCG query is NP-complete. ■

Proof: We establish the hardness by a reduction from a classical NP-complete problem, namely the minimum set cover (MSC) problem. An instance of the MSC problem consists of a universe $U = \{e_1, e_2, \ldots, e_n\}$ and a set of sets $S = \{S_1, S_2, \ldots, S_m\}$, where $S_i \subset U$. The decision problem is to decide if we can find a subset S' of S such that all the elements in U are fully covered by S' and the size of S' is minimum.

Given an instance of MSC, we construct an instance of a GSKCG query $Q = (k, P)$ on a set of users. Each element e_i in U corresponds to a spatial query point in P, each set S_i corresponds to a user u_i's associated region $u_i.R$, and the elements in S_i corresponds to the spatial points in u_i's associated region $u_i.R$. We consider the restricted case of GSKCG query when $k = 0$. It can be seen that there exists a solution to the MSC problem if and only if there exists a solution to

1. For ease of exposition, we consider each user to have one associated region. Our solution can be easily extended to the case where a user has multiple associated regions, as discussed later in Section 4.3.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2419663, IEEE Transactions on Knowledge and Data Engineering

Algorithm 1 KCGFinder (Query points P, Integer k, LBSN G)

1: $S \leftarrow$ The set of users in G that each covers at least one point in P;
2: $U^k \leftarrow$ The set of users belonging to S that may appear in a k-core;
3: $H \leftarrow$ All connected components of $G[U^k]$ that each fully covers P;
4: $M \leftarrow \max_{C^k \in H} |C^k|$;
5: **for** s from k+1 to M **do**
6: **for** each C^k in H **do**
7: **if** $|C^k| \geq s$ **then**
8: $C_s^k \leftarrow$ `GetOptimalGroup`(C^k, k, s, P);
9: **if** $C_s^k \neq \emptyset$ **then**
10: Return C_s^k;
11: Return $\emptyset$;

Q (i.e., find a minimum set of users such that all given query points are fully covered by their associated regions).

Suppose we have a polynomial-time algorithm A that returns the query answer $G' = \{u'_1, u'_2, \ldots, u'_m\}$ to a GSKCG query Q. If P is fully covered by the associated regions of G', then $\{S'_1, S'_2, \ldots, S'_m\}$ fully covers U and its size m is minimum. This implies that a polynomial-time solution to the MSC problem is found, leading to a contradiction. Therefore, there does not exist a polynomial-time algorithm A for the GSKCG query problem. □

In this paper, we study how to efficiently process GSKCG queries. We aim for an optimal solution that has short response time. This is mainly achieved by a set of effective pruning strategies (see Section 3) and a novel index structure (see Section 4).

3 Algorithm Design

In this section, we present our `KCGFinder` algorithm and a set of pruning strategies for answering GSKCG queries.

3.1 Basic Algorithm

To satisfy the minimum cardinality requirement of a GSKCG query, the general idea of `KCGFinder` is to process the user groups in increasing order of group size and return the current group as soon as it is valid.

Algorithm 1 gives the pseudo code of the `KCGFinder` algorithm. Before performing a search on the input LBSN $G = (V, E)$, we first conduct two filtering operations: spatial filtering and social filtering. In spatial filtering, we use an R-tree to get the users whose associated regions cover at least one query point $p \in P$ (Line 1, Algorithm 1). In social filtering, we adopt the core decomposition algorithm [4] to identify the user set U^k in which the users belonging to S may appear in a k-core, and invoke a depth-first search (DFS) to find the set of connected components H of $G[U^k]$ that each fully covers P (Lines 2–3, Algorithm 1).

In Line 4 of Algorithm 1, we compute the maximum cardinality M of the components in H, which gives the upper bound of the size of the returned user group. By definition, the cardinality of a k-core is $>= k + 1$. Thus, we enumerate user groups in increasing order of size from $k+1$ to M. Given a size s, for each component C^k with size $\geq s$, we invoke the `GetOptimalGroup` function (see Algorithm 2) to find a size-s user group C_s^k whose joint regions fully cover P (for short, we say "C_s^k covers P") and which is a k-core. If C_s^k is not empty, it is returned as the final optimal answer to the GSKCG query.

Algorithm 2 GetOptimalGroup (Component G, Integer k, Integer s, Query points P)

1: **for** each size-s user group C_s of G **do**
2: **if** the number of edges of $G[C_s] \geq k(k+1)/2$ **then**
3: **if** $G[C_s]$ is k-core and $P \subseteq \bigcup_{u \in C_s} u.R$ **then**
4: Return C_s;
5: Return $\emptyset$;

It can be observed that the main complexity of `KCGFinder` comes from the `GetOptimalGroup` function. Therefore, in the rest of this section, we focus on how to optimize `GetOptimalGroup` via a set of pruning techniques. We give the general idea of `GetOptimalGroup` in Algorithm 2. `GetOptimalGroup` enumerates all size-s user groups and checks whether they are valid. By the definition of k-core, we can prune out a user group C_s if the number of edges in $G[C_s]$ is $< k(k+1)/2$ (Line 2, Algorithm 2).

For a systematic enumeration of all candidate user groups, we employ the *branch and bound* algorithm [18]. In the branch and bound search process, we keep track of two user sets *SI* and *SU*, which represent the intermediate solution set and the set of remaining users, respectively. Initially, *SI* is empty, and *SU* is the set of all users in component G. We iteratively add users from *SU* to *SI* to check whether the resultant group is valid. This process can be organized into a tree structure, as illustrated in Figure 2, in which an internal node represents an *SI* and a leaf node represents a size-s candidate group. In the rest of this section, we explore a set of effective pruning strategies to speed up the branch and bound search.

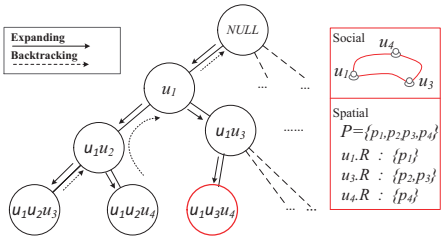


Fig. 2. Branch and bound search tree

3.2 Basic Pruning

We start with two basic pruning strategies, k-core (KC) based pruning and spatial query-point coverage (SQPC) based pruning, based on the degree constraint in a k-core and the spatial query point coverage constraint, respectively.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2419663, IEEE Transactions on Knowledge and Data Engineering

3.2.1 KC *based pruning*

By the definition of k-core, we know that the minimum degree of each vertex in a k-core should be no less than k. Therefore, in the branch and bound search, if the minimum degree constraint cannot be satisfied after adding any new users from *SU* to *SI*, the search process should backtrack to the previous state of *SI* (that is, the parent node of the node representing *SI* in the search tree). We give the critical condition under which the current *SI* may form a valid group below.

Theorem 2: Let $u_{min} \in$ SI be the user with the minimum number of neighbors in SI. If SI is in any valid group with size s, then $|NB_{SI}(u_{min})| + s - |SI| \geq k$ where $|NB_{SI}(u_{min})|$ is the number of u_{min}'s neighbors in SI. ■

Proof: Since we can add only $s - |SI|$ users from *SU* to *SI*, the degree of u_{min} in any valid group with size s is at most $|NB_{SI}(u_{min})| + s - |SI|$ (when all users in *SU* are neighbors of u_{min}). By Definition 1, to form a valid group, the degree of u_{min} in the group should be $\geq k$. This establishes the theorem. □

Theorem 2 implies that if the current *SI* cannot satisfy this condition, the entire subtree rooted at the node representing *SI* can be skipped.

Example 2: Consider the LBSN in Figure 1. Let *SI*=$\{u_2, u_4\}$, *SU*=$\{u_1, u_3\}$, $s = 3$ and $k = 2$. Since u_2 has the minimum number of neighbors in *SI* and $|NB_{SI}(u_2)| = 0$, we can verify that the condition in Theorem 2 does not hold, and therefore we can stop searching the users in *SU*. ■

3.2.2 SQPC *based pruning*

Any valid user group should cover all query points P. If *SU* cannot fully cover the rest query points in $P - P_{SI}$, where P_{SI} is the set of points covered by *SI*, adding any user from *SU* to *SI* cannot form a valid group. In this case, the search process can safely prune the subtree rooted at *SI* without missing the optimal solution.

In some cases, even though *SU* can cover all query points in $P - P_{SI}$, the users of *SU* are still not a member of any valid group. Theorem 3 is given to capture such cases.

Theorem 3: Let $u_{max} \in SU$ be the user whose region covers the most query points in $P - P_{SI}$. To form a valid group with size s, SU should satisfy:

$$\frac{|P - P_{SI}|}{s - |SI|} \leq |P_{u_{max}}| \tag{1}$$

where $|P_{u_{max}}|$ is the number of query points covered by u_{max}. ■

Proof: $|P - P_{SI}|$ is the number of query points not covered by SI, and $s - |SI|$ is the number of users to be added from SU to SI. On average, each user to be added should cover at least $\frac{|P - P_{SI}|}{s - |S_I|}$ query points. Therefore, the number of points in $P - P_{SI}$ covered by u_{max} must be greater than or equal to the average. □

Intuitively, given the number of query points, the size of the user group and SI, Equation 1 gives the lower bound of the maximum number of query points in $P - P_{SI}$ that a user in SU should cover.

Example 3: Consider the LBSN in Figure 1. Suppose $SI = \{u_2\}$, $SU = \{u_1, u_3, u_4, u_5, u_6\}$, $k = 2$ and $s = 3$. We can compute $\frac{|P - P_{SI}|}{s - |S_I|} = \frac{|\{p_1, p_3, p_4\}|}{3-1} = \frac{3}{2}$ and $|P_{u_{max}}| = 1$. Since Equation 1 is not true, there is no need to search users in SU. ■

3.3 Diameter Based Pruning

In this section, we propose pruning techniques based on the concept of *social diameter*. We first give the definition of the diameter of a group C_s^k with size s.

Definition 3: (Diameter) The *diameter* of a user group C_s^k in an LBSN G is defined as the longest shortest path length between any two users in $G[C_s^k]$, denoted by $\text{DIA}(C_s^k)$. ■

Let $\text{DIA}_{ub}(C_s^k)$ denote the upper bound of $\text{DIA}(C_s^k)$. It is easy to derive that $\text{DIA}_{ub}(C_s^k) = s - k$. However, this bound is too loose when s is big. In this paper, we make use the more strict bound proposed in [30].

Theorem 4: For a user group C_s^k,

$$\text{DIA}_{ub}(C_s^k) = \begin{cases} 1 & \text{if} \quad s = k+1 \\ 2 & \text{if} \quad k+1 < s < 2k+2 \\ 3[\frac{s}{k+1}] + r(s,k) - 3 & \text{if} \quad s \geq 2k+2 \end{cases} \tag{2}$$

$$where\ r(s,k) = \begin{cases} 0 & \text{if} \quad mod(s, k+1) = 0 \\ 1 & \text{if} \quad mod(s, k+1) = 1 \\ 2 & \text{if} \quad mod(s, k+1) = 2 \end{cases} \blacksquare$$

This diameter upper bound of C_s^k introduces a way to measure whether two users can co-exist in C_s^k. Next we present two pruning techniques called social shortest path (SHP) based pruning and spatial-social shortest path (SOSP) based pruning.

3.3.1 SHP *based pruning*

The SHP based pruning is inspired by the observation that, if the shortest path length between two users exceeds $\text{DIA}_{ub}(C_s^k)$, they cannot appear simultaneously in C_s^k. It follows that a user $v \in SU$ can be added into SI only when the shortest path length between v and $u \in SI$ satisfies the condition presented in Theorem 5.

Theorem 5: Let v be the user to be added into SI from SU, and $Dist(SI, v)$ be the maximum shortest path length between the users in SI and v. v can be added into SI only if the following inequation is satisfied:

$$Dist(SI, v) \leq \text{DIA}_{ub}(C_s^k) \blacksquare \tag{3}$$

Proof: By Definition 3, the shortest path length between any two users in a valid group C_s^k should be $\leq \text{DIA}(C_s^k)$. If v can be added into SI, then $Dist(SI, v) \leq \text{DIA}(C_s^k)$ must be true. Therefore, $Dist(SI, v) \leq \text{DIA}_{ub}(C_s^k)$ is also true. □

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2419663, IEEE Transactions on Knowledge and Data Engineering

Example 4: Consider the LBSN in Figure 1. Suppose k=1 and s=3. Let $SI = \{u_4\}$ and $SU = \{u_5, u_6\}$. By Theorem 4, we can compute, for any valid group C_s^k, $\text{DIA}_{ub}(C_s^k) = 2$. The shortest path lengths between u_4 and u_5, u_4 and u_6, are 3 and 4, respectively. Thus, no user in SU can be added into SI to form a valid group. ■

3.3.2 SOSP *based pruning*

SHP based pruning can quickly verify whether there exists a user in SU to form a valid group with the current SI. To further reduce the search space, we present SOSP based pruning, which considers not only the shortest path length between two users but also the users' covered query points.

Intuitively, for any valid user group C_s^k, if a user u and all other users in the circle centered at u with diameter $\text{DIA}(C_s^k)$ cannot fully cover all query points P, u cannot be a member of C_s^k. This implies that, in this case, for the given specific values of k and s, u could be removed from the search space without missing the optimal solution. This provides extra pruning capabilities on top of SHP based pruning. Let NB_u^p be the user that has the minimum shortest path length to a user u in an LBSN and that covers a query point $p \in P$. We formally capture this intuition in Theorem 6.

Theorem 6: Let $Dist(u, NB_u^p)$ be the shortest path length between u and NB_u^p. For any valid group C_s^k, if $Dist(u, NB_u^p) > \text{DIA}_{ub}(C_s^k)$ for some $p \in P$ and $p \notin u.R$, then u cannot be a user of C_s^k. ■

Proof: Suppose v is a user of C_s^k and there exists a query point $p \in P$ satisfying $Dist(v, NB_v^p) > \text{DIA}_{ub}(C_s^k)$. Since C_s^k is a valid group, by Theorem 5 we have $Dist(v, u) \leq \text{DIA}(C_s^k)$ where $u \in C_s^k$. Since $p \in P$ must be covered by C_s^k, we have $NB_v^p \in C_s^k$ and therefore $Dist(v, NB_v^p) \leq \text{DIA}(C_s^k)$, leading to a contradiction. This completes the proof. □

Below we provide an example to illustrate how SOSP based pruning works.

Example 5: Consider constructing a valid group C_s^k with $k = 1$ and $s = 3$ for the LBSN in Figure 1. The user set of this LBSN is $\{u_1, u_2, u_3, u_4, u_5, u_6\}$. From Theorem 4, we get $\text{DIA}_{ub}(C_s^k) = 2$. Since $NB_{u_5}^{p_4}$ and $NB_{u_6}^{p_4}$ are both u_4, we have $Dist(u_5, u_4) = 3$ and $Dist(u_6, u_4) = 4$. From Theorem 6, we learn that both u_5 and u_6 should be removed from the search space of finding C_s^k. Now we get a smaller search space $\{u_1, u_2, u_3, u_4\}$. ■

For diameter based pruning techniques, we need to compute the shortest path length between any pair of users. However, it is impossible to calculate the length on the fly, because it will substantially increase the total running time of our algorithm. A possible method is to pre-compute all the lengths offline and then index them for online query processing. However, this approach needs $O(n^2)$ storage, where n is the number of users in the LBSN. It is not feasible to store such big indexes when n is large. In this paper, we adapt the tree-structured index constructed based on the concept of vertex cover (VC-index) [7], which can efficiently process distance queries between users with a small storage cost. We also employ the caching technique to accelerate querying the shortest path length of users. Given two users, we first retrieve the shortest path length between them in the cache. If the length is cached, we read it directly. Otherwise, the length is calculated from the VC-index. For the strategy of replacing the cache, we adopt the least recently used (LRU) method.

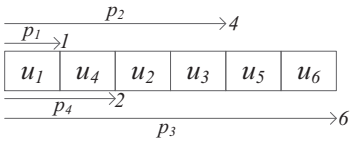


Fig. 3. Sorted user list $List_P$

3.4 Access Order Based Pruning

In the section, we propose the last pruning strategy based on the observation that more search space can be pruned if the users are accessed in a certain order in the branch and bound search. Given a set of users V and a set of query points P, we place the users in V into several sets $VP = \{V_{p_1}, V_{p_2}, \cdots, V_{p_{|P|}}\}$, where V_{p_i} is the set of users whose associated region covers the point $p_i \in P$. Note that a user u may belong to multiple V_{p_i}, because u's associated region may cover one or more query points. We first sort VP in increasing order of V_{p_i}'s size, and then sequentially access V_{p_i} and push all users in V_{p_i} into a user list $List_P$. If a user has been pushed into $List_P$, he/she can be skipped in later operations. Thereafter, the search process adds users from SU to SI according to the their indexes in $List_P$. We give an example of constructing $List_P$.

Example 6: Consider the users and query points in Figure 1(b). We can place the users into four sets, $V_{p_1} = \{u_1\}$, $V_{p_2} = \{u_2, u_3\}$, $V_{p_3} = \{u_3, u_5, u_6\}$, $V_{p_4} = \{u_4\}$. To construct the sorted user list $List_P$, we first add u_1 (in V_{p_1}) to $List_P$, then u_4, u_2, u_3 in order. After that, since u_3 has been added when processing V_{p_2}, he/she will be skipped when accessing V_{p_3}. Finally, u_5 and u_6 are added. The constructed $List_P$ is given in Figure 3 (ignore the arrows for the moment). ■

Next we discuss how to make use of $List_P$ to gain additional pruning capability. For a query point p, we define its *access index* in $List_p$ as follows.

Definition 4: (Access index) The *access index* of a query point $p \in P$ in a sorted user list $List_P$, denoted by $\mathcal{A}(p)$, is the index of the last user whose associated region covers p. ■

The access indexes are illustrated in Figure 3. Suppose p is the query point in P that has not been covered by SI. If the smallest index of users in SU is greater than $\mathcal{A}(p)$, the search process should backtrack

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2419663, IEEE Transactions on Knowledge and Data Engineering

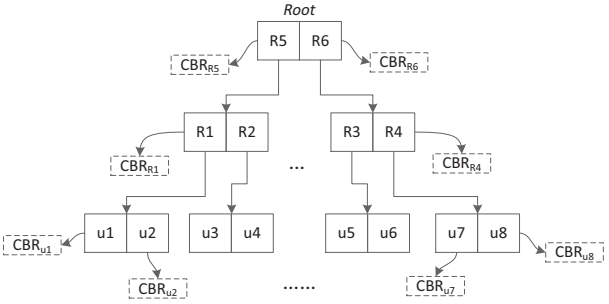


Fig. 4. A sample SaR-tree

to the parent node of SI. For a GSKCG query, we maintain the access index for each query point. Note that, for a GSKCG query, the access indexes and $List_p$ just need to be calculated once and do not need to be updated. Therefore, they can be constructed efficiently.

Example 7: Continue with Example 6. We have $\mathcal{A}(p_1) = 1$, $\mathcal{A}(p_2) = 4$, $\mathcal{A}(p_3) = 6$ and $\mathcal{A}(p_4) = 2$. Suppose $SI = \{u_1\}$, $SU = \{u_2, u_3, u_5, u_6\}$, and $P - P_{SI} = \{p_2, p_3, p_4\}$. According to the access order in $List_p$, u_2 should be the first to be added to SI. We compare u_2's index in $List_P$, 3, with $\mathcal{A}(p_2)$, $\mathcal{A}(p_3)$ and $\mathcal{A}(p_4)$. Since $\mathcal{A}(p_4) = 2 < 3$, no user in SU can be added to SI. The search process backtracks to SI's previous state. ■

4 Hybrid Indexing

In this section, we design a novel index structure, the Enhanced Social-aware R-tree (SaR-tree), to further accelerate query processing.

4.1 SaR-tree

The SaR-tree structure [38] is a variant of R-tree that indexes both spatial locations and social relations. Figure 4 illustrates a simple SaR-tree. Different from a classical R-tree, each entry of an SaR-tree contains two major pieces of information: a set of core bounding rectangles (CBRs) (see Definition 5) that encodes the social information and a minimum bounding rectangle (MBR) that encodes the spatial information as in an R-tree. Intuitively, a CBR bounds the users by the social constraint while an MBR bounds the users by the spatial constraint, and therefore an SaR-tree gains the ability of both social-based and spatial-based pruning for GSKCG query processing.

Definition 5: (Core bounding rectangle) Consider a user $u \in G$. Given a minimum degree constraint k, the *core bounding rectangle* $CBR_{u,k}$ is a rectangle that contains u and inside which any user group with u (excluding the users on the bounding edges) cannot be a k-core. ■

Note that, for given u and k, $CBR_{u,k}$ may not be unique. We illustrate the idea of CBR in the following example.

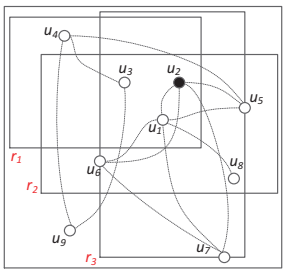


Fig. 5. Example of CBRs in an SaR-tree

Example 8: Consider the LBSN in Figure 5. Given $k = 2$, the rectangle r_1 is a $CBR_{u_2,2}$ because any user group inside r_1 that contains u_2 is not a 2-core. Similarly, r_3 is another $CBR_{u_2,2}$ for u_2. In contrast, r_2 is *not* a $CBR_{u_2,2}$ because $\{u_1, u_2, u_5\}$ in r_2 form a 2-core. ■

In addition to CBRs and an MBR, each entry in an SaR-tree also contains a *core number*. A user u's core number is the maximum k for which u belongs to a k-core, denoted by $cn(u)$. The core number of an entry e is defined as the maximum of the core numbers of the users covered by e, denoted by $cn(e)$.

4.2 Enhanced SaR-tree

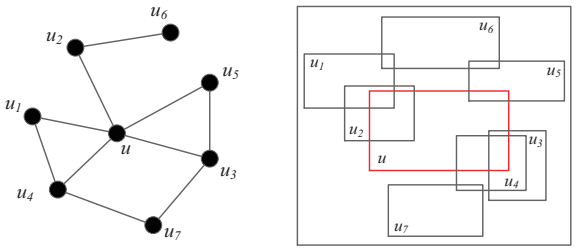
Unfortunately, the SaR-tree structure proposed in [38] cannot support GSKCG queries. The main reason is that the method of computing CBRs in [38] assumes that each user is associated with a spatial point, whereas in our problem each user has an associated region. This fact significantly complicates the problem and demands a new method to construct CBRs.

We propose a novel index structure, known as the enhanced SaR-tree, to address this problem. To construct an Enhanced SaR-tree over an LBSN, we first construct a standard R-tree $rtree$ and then compute the CBR for each entry in $rtree$. To compute the CBR of an entry, we should know how to build a user's CBR. The general idea of constructing a user's CBR includes two steps. First, as the users' associated regions may intersect with each other, we calculate the user's internal CBR (see Definition 6). Second, given the user's internal CBR, we expand it to obtain the corresponding external CBR (see Definition 7), from which the user's CBR will be selected. We give the formal definitions of these two types of CBRs below. For ease of exposition, we denote "a k-core containing a user u" by "$k(u)$".

Definition 6: (Internal CBR) Given a k value, a user u's *internal CBR* $iCBR_{u,k}$ is a rectangle that is inside $u.R$ and that does not contain a $k(u)$. ■

Example 9: Consider the LBSN in Figure 6. Figure 7 shows some $iCBR_{u,2}$ of user u, marked by the shaded areas. Figure 7(a–b) and Figure 7(c–e) show $iCBR_{u,2}$ of user u in *x-direction* and *y-direction*, respectively. In Figure 7(a), the shaded area is an $iCBR_{u,2}$ of user u because: 1) it is inside $u.R$, and; 2) the users in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2419663, IEEE Transactions on Knowledge and Data Engineering



(a) Social networks (b) Associated regions

Fig. 6. A sample LBSN for constructing CBR

this $iCBR_{u,2}$ (i.e., u_1, u_2, and u) cannot form a 2-core containing u. ■

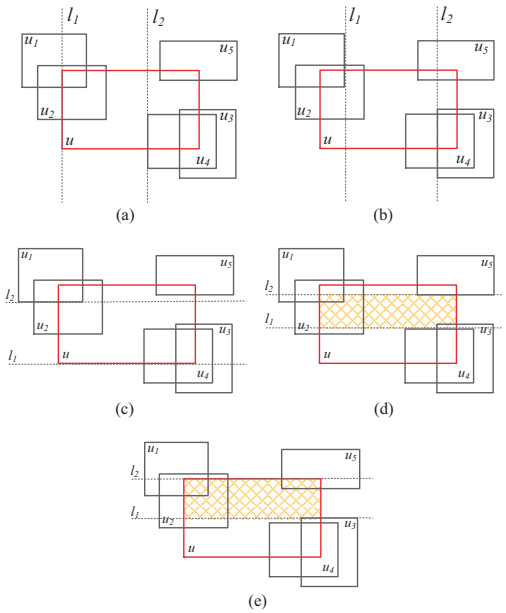


Fig. 7. Constructing user u's internal CBRs

Definition 7: (External CBR) Given a user u's internal CBR $iCBR_{u,k}$, the corresponding *external CBR* $eCBR_{u,k}$ is defined as a rectangle that: 1) contains this $iCBR_{u,k}$, and; 2) is inside the MBR of u's parent in $rtree$, and 3) does not contain a $k(u)$. ■

Example 10: Continue with Example 9. Given a user u's $iCBR_{u,2}$ in Figure 7(a), Figure 8 shows the corresponding $eCBR_{u,2}$. The outermost rectangle marks the MBR of u's parent in the enhanced SaR-tree. ■

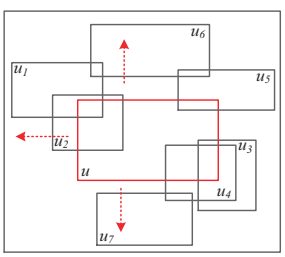


Fig. 8. Constructing a user u's external CBRs

Algorithm 3 describes how to construct a CBR of a user u. We first use an R-tree to find the users

Algorithm 3 GetUserCBR (User u, Integer k, LBSN G, Enhanced SaR-tree $rtree$)

```
1: H ← The users whose familiar regions overlap with u.R;
2: X ← Left and right edges of the familiar region of each user in H;
3: Y ← Top and bottom edges of the familiar region of each user in H;
4: Sort the elements in X and Y in ascending order;
5: iCBR(X) ← GetInternalCBRs(u, k, X, H, G);
6: iCBR(Y) ← GetInternalCBRs(u, k, Y, H, G);
7: iCBRs ← iCBR(X) ∪ iCBR(Y);
8: eCBRs ← GetExternalCBRs(u, k, iCBRs, G, rtree);
9: Return the element of eCBRs with the maximum area;
```

Algorithm 4 GetInternalCBRs (User u, Integer k, Line set X, User set H, LBSN G)

```
1: LB ← A line on the left edge of u.R;
2: ℓ1 ← LB;
3: ℓ2 ← LB;
4: iCBRs ← ∅;
5: while ℓ1 and ℓ2 do not exceed the right edge of u.R do
6:     OperateBoth(ℓ1, ℓ2, X, G);
7:     OperateL2(ℓ2, X, G);
8:     iCBRs.add(Λ[ℓ1, ℓ2, u.R]);
9:     OperateL1(ℓ1, X, G);
10: Return iCBRs;
```

whose associated regions overlap with $u.R$, and add them into a user set H (Line 1, Algorithm 3). We then construct a set of $iCBR_{u,k}$ of u from two directions (i.e., x-direction and y-direction). We put the left and right (or bottom and top) edges of the familiar regions of the users in H into the line set X (or Y), respectively, and sort the lines in X (or Y) in ascending order in order to facilitate the construction of internal CBRs (Line 4, Algorithm 3). Then we use the `GetInternalCBRs` function to generate the internal CBRs on both directions. Based on these internal CBRs, we invoke the `GetExternalCBRs` function to calculate the corresponding external CBRs. Finally, the external CBR with the maximum area is returned as u's CBR. Next we elaborate `GetInternalCBRs` and `GetExternalCBRs`.

The `GetInternalCBRs` function. The general idea of constructing $iCBR_{u,k}$ of a user u is to alternately slide two vertical (or horizontal) lines on $u.R$ in x-direction (or y-direction). In the end, the area inside the intersection of these two lines and $u.R$ will be u's $iCBR_{u,k}$. Since the construction of $iCBR_{u,k}$ in y-direction is similar to that in x-direction, we only discuss the case for x-direction.

Given a user u, a value of k, a sorted line set X and a user set H, we primarily perform three kinds of operations on lines ℓ_1 and ℓ_2 (i.e., move ℓ_1 and ℓ_2 simultaneously, move ℓ_2 alone, and move ℓ_1 alone) to obtain $iCBR_{u,k}$ of u in x-direction. Initially, we place both ℓ_1 and ℓ_2 on the left edge of $u.R$ (Lines 1–3, Algorithm 4), and then move ℓ_1 and ℓ_2 rightward using one of the following operations:

1) `OperateBoth`: When ℓ_1 and ℓ_2 overlap with each other, we move them rightward to the next line in X (but *not* exceeding the right edge of $u.R$) such that the users in H whose familiar regions are touched by ℓ_1 and ℓ_2, denoted by $H(\ell_1)$, do not form a $k(u)$.

Algorithm 5 GetExternalCBRs (User u, Integer k, Internal CBRs $iCBRs$, LBSN G, Enhanced SaR-tree $rtree$)

```
1: eCBRs ← ∅;
2: for each internal CBR iCBR_{u,k} in iCBRs do
3:     eCBR_{k,u} ← Expand each edge of iCBR_{u,k} until a k(u) appears
       or the edge reaches the MBR boundary of u's parent in rtree;
4:     eCBRs.add(eCBR_{k,u});
5: Return eCBRs;
```

2) `OperateL2`: We move ℓ_2 rightward to the next line in X (*not* exceeding the right edge of $u.R$) such that the users in the rectangle bounded by ℓ_1, ℓ_2 and $u.R$, denoted by $\Lambda[\ell_1, \ell_2, u.R]$, form a $k(u)$.[2] At this moment, $\Lambda[\ell_1, \ell_2, u.R]$ is an internal CBR of u.
3) `OperateL1`: We move ℓ_1 rightward to the next line in X (*not* exceeding the right edge of $u.R$) such that the users in $\Lambda[\ell_1, \ell_2, u.R]$ do not form a $k(u)$.[3] Note that ℓ_1 is always on the left hand side of ℓ_2.

We alternate these three types of operations until both ℓ_1 and ℓ_2 stop at the right edge of $u.R$. Finally, `GetInternalCBRs` returns all internal CBRs of u.

Example 11: Consider the LBSN in Figure 6. We illustrate how to compute $iCBR_{u,2}$ of user u in x-$direction$ in Figure 7. Initially, lines ℓ_1 and ℓ_2 are placed on the left edge of $u.R$. Since at this time $H(\ell_1) = \{u_1, u_2, u\}$ and these users do not form a $k(u)$ with $k = 2$, there is no need to move ℓ_1 and ℓ_2. We then move ℓ_2 rightward to the left edge of u_4, and now the users in the rectangle $\Lambda[\ell_1, \ell_2, u.R]$, $\{u_1, u_2, u_4, u\}$, form a $k(u)$. So the current $\Lambda[\ell_1, \ell_2, u.R]$ is an internal CBR of u. Next, we move ℓ_1 rightward until the right edge of u_1 because now the users in $\Lambda[\ell_1, \ell_2, u.R]$, $\{u_2, u_4, u\}$, do not contain a $k(u)$. We continue this process until both ℓ_1 and ℓ_2 reach the right edge of $u.R$ ■

The `GetExternalCBRs` function. Given the set of $iCBR_{u,k}$ returned by Algorithm 4, Algorithm 5 is designed for constructing user u's $eCBR_{u,k}$. The basic idea is to expand each $iCBR_{u,k}$ in $iCBRs$ to obtain the corresponding $eCBR_{u,k}$. We expand each edge of $iCBR_{u,k}$ outward until the users within $iCBR_{u,k}$ form a $k(u)$. Recall that, by Definition 7, u's $eCBR_{u,k}$ is inside the MBR of u's parent in the enhanced SaR-tree $rtree$. So we should stop expanding an edge once it reaches the boundary of the MBR.

Example 12: Continue with Example 11. Given $iCBR_{u,2}$ of user u shown in Figure 7(a), we show an example of constructing the corresponding $eCBR_{u,2}$ in Figure 8. Assume that the outermost rectangle is the MBR of u's parent. We sequentially move each of the four edges of $iCBR_{u,2}$ outward until getting the shadow area. ■

Finally, we discuss how to compute the CBR of each entry in the enhanced SaR-tree by a bottom-up approach. A leaf entry's CBR is the CBR of the user it represents. For an internal entry e, let its child entries be $e_1, e_2, \cdots, e_m$. Given the minimum degree constraint k, e's CBR CBR_e can be computed by recursively applying the following function on its child entries' CBRs CBR_{e_i}:

$$CBR_{e_1^{i+1},k} = \begin{cases} CBR_{e_1^i,k}, \\ \quad \textbf{if } MBR_{e_{i+1}} \cap CBR_{e_1^i,k} = \emptyset \\ CBR_{e_1^i,k} \cap CBR_{e_{i+1},k} \\ \quad otherwise \end{cases} \tag{4}$$

where $CBR_{e_j^i,k}$ denotes the CBR constructed from $CBR_{e_j,k}$, $CBR_{e_{j+1},k}$, $\cdots$, $CBR_{e_i,k}$. Therefore, $CBR_{e,k} = CBR_{e_1^m,k}$. It is easy to verify that, by this construction, any user group within $CBR_{e,k}$ cannot be a k-core, giving extra pruning capabilities.

In practice, k usually does not have to be a large value. Setting k to 1, 2, or 3 normally suffices for all ordinary requirements of social constraint. Thus, when k is small, we can build indexes for each of the possible k values with reasonable space and time. Without loss of generality, we also discuss the case when k is large. Here we can select a set of k values to build the indexes by considering the property of k-core, that is, k-core$\subseteq$(k-1)-core. With this property, we can only build indexes for $k = 2^0, 2^1, \cdots, 2^{\lfloor log_2^{cn(e)} \rfloor}$ where $cn(e)$ is the maximum core number of its child entries rooted at entry e. Given a GSKCG query $Q = (k', P)$, we can make use of the CBRs of $k = 2^{\lfloor log_2 k' \rfloor}$($k$ is left close to k') for GSKCG query processing. This method may incur false positive users (the users whose core number is less than k'), which may in turn enlarge the searching space and increase the computation cost in the later steps. However, it does not compromise the correctness of the query results and normally can be done with reasonable space cost of the indexes and efficiency of query processing.

We also give a space complexity analysis for our proposed Enhanced SaR-tree. For an entry e/user u, we only store the CBRs of e or u for the core number $2^0, 2^1, \cdots, 2^{\lfloor log_2^{cn(e)} \rfloor}$. Let M denote the maximum core number of the users in G, s be the fanout of our index, and n be the number of users in an LBSN G. The upper bound of the total number of CBRs (denoted by N_{cbr}) in an Enhanced SaR-tree can be computed:

$$\begin{aligned} N_{cbr} &\leq \frac{2n(\lfloor log_2^M \rfloor + 1)}{s} + \textstyle\sum_{u \in V}(\lfloor log_2^{cn(u)} \rfloor) \\ &\leq n(\frac{2n(\lfloor log_2^M \rfloor + 1)}{s} + 1 + \lfloor log_2 \frac{\sum_{u \in V} cn(u)}{n} \rfloor) \end{aligned} \tag{5}$$

Since M and $\frac{\sum_{u \in V} cn(u)}{n}$ is usually small in a social network, thus the space cost of CBRs is comparable to that of G. For the datasets used in our experiments, the maximum M and $\frac{\sum_{u \in V} cn(u)}{n}$ of both datasets are 52 and 3.8, respectively. We set the fanout s of our indexes to 100. Hence, we can calculate the maximum number of CBRs of our Enhanced SaR-tree on both datasets, which is around $2.12n$.

2. Once ℓ_2 touches the left edge of a user's familiar region, this user is in the rectangle.

3. Once ℓ_1 touches the right edge of a user's familiar region, this user is *not* in the rectangle any more.

Algorithm 6 SaRBasedKCGFinder (Query points P, Integer k, Enhanced SaR-Tree $rtree$, LBSN G)

1: $MBR(P) \leftarrow$ The minimum rectangle containing all points in P;
2: Initialize H with the root of $rtree$;
3: **while** H has non-leaf entries **do**
4: $e \leftarrow$ The first non-leaf entry in H;
5: **for** each child entry e' of e **do**
6: **if** $MBR(P) \cap e'.MBR \neq \emptyset$ and $cn(e') \geq k$ and $MBR(P) \not\subset CBR_{e',k}$ **then**
7: $H.push(e')$;
8: $V_H \leftarrow$The set of users represented by the entries in H;
9: Return KCGFinder(P, k, $G[V_H]$);

4.3 GSKCG Query Processing

In this section, we present our integrated algorithm SaRBasedKCGFinder. Generally, the algorithm consists of two steps: 1) filter impossible users based on the enhanced SaR-tree; 2) feed the remaining users to KCGFinder. We give the details of SaRBasedKCGFinder in Algorithm 6.

We first calculate the minimum rectangle containing all query points P (i.e., the coverage of P), denoted by $MBR(P)$. We iteratively prune impossible users in the LBSN G by traversing the enhanced SaR-tree $rtree$. Note that, for the same LBSN G, $rtree$ just needs to be constructed once and thereafter can be used for all GSKCG queries. At each entry e of $rtree$, we compare $MBP(P)$ with e's MBR and CBR and check the core number of e in order to prune out the users that cannot appear in the final result (Line 6, Algorithm 6). Finally, we feed the subgraph of G that contains the users represented by the entries in H to KCGFinder and return its output.

It is easy to extend our algorithm to support the case where each user has multiple associated regions. For each associated region of a user u, we index u and this associated region one time in the Enhanced SaR-tree. Thus, the number of associated regions of u corresponds the times of u being indexed. In spatial filtering, if a user u appears more than one time, we simply combine them together. The following branch and bound search process remains the same as the case where each user has exactly one associated region.

5 Performance Evaluation

In this section, we experimentally study the performance of three algorithms. The first one is the basic KCGFinder (referred to as *Baseline*) presented in Section 3.1. The second one is KCGFinder coupled with the set of pruning techniques (*Advanced*). The third one is SaRBasedKCGFinder (*SaRBased*).

5.1 Datasets and Queries

We evaluate the proposed algorithms on two datasets collected from *Brightkite* and *Gowalla*,[4] two real-world LBSNs. The properties of the two datasets are summarized in Table 2. Since these websites do not directly provide users' regions, we use a density-based clustering method to form their regions from check-in locations. A user may have several clustered associated regions. By default, we choose the region with the most check-ins as a user's associated region. The medium areas of the regions are 12.24 km^2 on *Brightkite* and 10.23 km^2 on *Gowalla*, respectively. More specifically, the cumulative distribution of the region sizes on *Brightkite* is: 21.5% of regions $\leq$ 0.1 km^2; 33.2%, $\leq$1 km^2; 45.6%, $\leq$10 km^2; 79.4%, $\leq$100 km^2, and so on. For *Gowalla*, the distribution is: 19.1% of regions $\leq$ 0.1 km^2; 28.2%, $\leq$1 km^2; 50.7%, $\leq$10 km^2; 80.4%, $\leq$100 km^2, and so on. In our experiments, we also show the results of the case where users have multiple associated regions. The testing GSKCG queries are randomly generated on these two real-life datasets. The query set on *Brightkite* includes 100 queries, while the query set on *Gowalla* includes 200 queries. Each query contains several query points that are randomly selected from all users' associated regions.

TABLE 2
Dataset properties

	Brightkite	*Gowalla*
Total # of users	58,228	196,591
Total # of friend relations	214,018	950,327
Medium region area (km^2)	12.24	10.23
Diameter (longest shortest path)	16	14
Total # of check-ins	4,491,143	6,442,890
Maximum # of cores	52	51

5.2 Setup

All the algorithms are implemented in Java programming language. The models for the CPU and RAM are Intel Xeon X5650 Processor 2.67G Hz and 8GB DDR3 memory, respectively. The number of query points and the value of k in a query both vary from 1 to 5. Unless explicitly specified, the default value of k and the default number of query points in a query are both set to 3. The fanout of the Enhanced SaR-tree is 100. The storage cost and building time of the Enhanced SaR-tree on *Brightkite* and *Gowalla* are (96.3 MB, 10.23mins) and (275.6 MB, 29.03mins), respectively.

5.3 Experimental Results

We evaluate the performance of these three algorithms under different parameter settings. As in many other performance evaluation schemes for query processing [22], [34], [35], we report the overall performance in terms of the *average query running time*.

Effect of the value of k. In the first set of experiments, we evaluate the performance of the algorithms under different k values. From Figure 9, we can observe that both *Advanced* and *SaRBased* perform substantially better than *Baseline*. Note that the y-axis is in log-scale. With the increase of k, *SaRBased* exhibits increasingly better performance than *Baseline* and *Advanced*. This is because, the larger k is, the larger CBRs are, and

4. Publicly available at: http://snap.stanford.edu/.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2419663, IEEE Transactions on Knowledge and Data Engineering

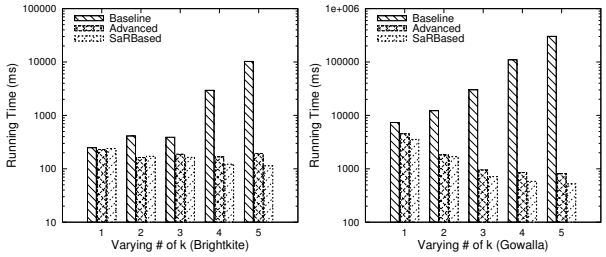


Fig. 9. Running time vs. k value

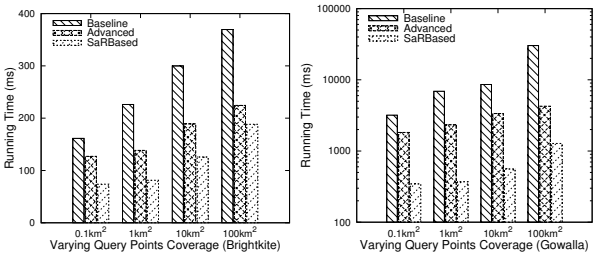


Fig. 11. Running time vs. query point coverage

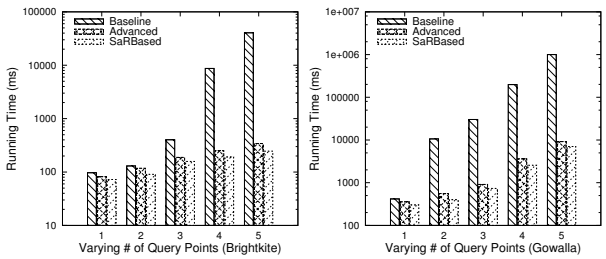


Fig. 10. Running time vs. number of query points

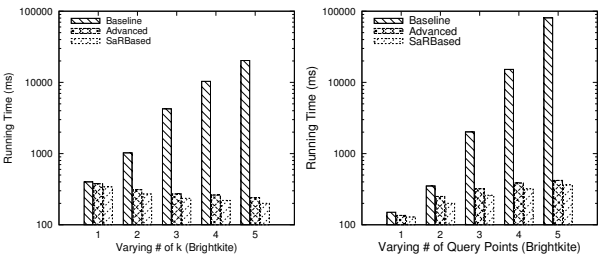


Fig. 12. Running time under multiple familiar regions

therefore the query points are more likely to be covered by larger CBRs, leading more tree branches to be pruned. When $k = 1$, the difference of query time among these three algorithms is small. The reason is that $k = 1$ indicates a loose social constraint and small CBRs, resulting in weak pruning capabilities.

Effect of the number of query points. In Figure 10, we examine the query performance by varying the number of query points. In general, the query time increases when the number of query points increases because more query points require more candidate users to be added into the search space. Compared to the other two algorithms, *SaRBased* is relatively less sensitive to the increase of the number of query points. Even when the number is 5, the performance of *SaRBased* is still reasonably good.

Effect of the coverage of query points. In this set of experiments, we evaluate the performance by varying the coverage of query points. From Figure 11, we find that *SaRBased* still performs best under different coverages. When the coverage is small, the query points are more likely to fully fall into some CBRs of the enhanced SaR-tree, leading to a smaller search space. This explains why the running time is shorter when the coverage is smaller.

Effect of multiple associated regions. As a proof-of-concept, in Figure 12, we present the performance of the algorithms when each user has on average 3 associated regions. Although it takes longer to process the queries, the general trends of Figure 12 are similar to those of a single associated region presented in Figures 9 and 10. When a user is associated with more associated regions, the running time increases because the number of users covering the query points increases, which implies a larger search space. Due to space limitations, we only show the performance on *Brightkite*. We observe similar results on *Gowalla*.

Pruning capabilities of different strategies. In Figure 13, we show the pruning capabilities of different pruning strategies, where *BP*, *DBP*, *AO* and *SAR* stand for basic pruning, diameter based pruning, access order based pruning and enhanced SaR-tree based pruning, respectively. We can see that all strategies can help reduce the running time. In particular, AO is most effective when the value of k or the number of query points is relatively large.

Sizes of returned user groups. Figure 14 shows the average group size of the query results. We can see that when k or the number of query points is increasing, the group size is also increasing. The reason is that when k is increasing, the lower bound of the group size to form a k-core group is increasing. Meanwhile, if the number of the query points is increasing, the group may need more users to cover the query points.

Effect of the size of LSBNs. Next, we show the scalability of the algorithms under various network sizes in Figure 15. We randomly extract several subsets of users with increasing sizes to test the algorithms' scalability. As expected, the result demonstrates that *SaRbased* achieves the best efficiency in all cases. Even

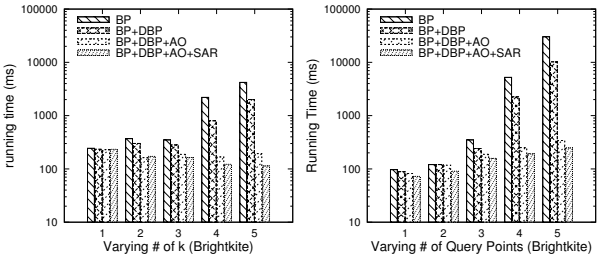


Fig. 13. Pruning capabilities of different schemes

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2419663, IEEE Transactions on Knowledge and Data Engineering

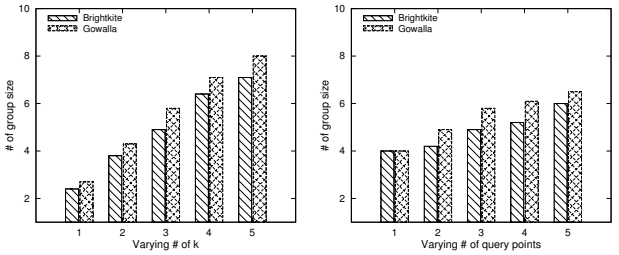


Fig. 14. Size of query results

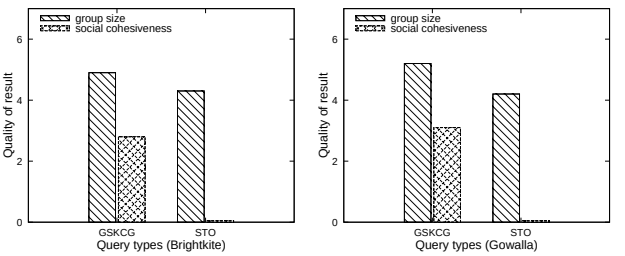


Fig. 16. Quality comparison of the returned groups

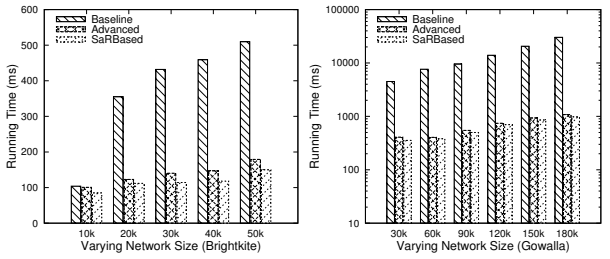


Fig. 15. Running time vs. network size

over relatively large networks (e.g., more than 180k users), it still responds quickly, demonstrating the potential for practical use.

Quality of query results. Finally, we compare the quality of results returned by GSKCG and the existing spatial task outsourcing in terms of the average group size and the average social cohesiveness (e.g., the average number of the familiar persons of each member in a group) of the returned group. Specifically, we consider the typical spatial task outsourcing (STO) problem that finds a minimum group to collaboratively cover a given number of spatial point related task. We set the parameters of the GSKCG query $Q = (k, P)$ to be $k = 2$ and $|P| = 5$ and the parameters of STO query $Q = (P)$ (i.e., not considering the social constraint) to be $|P| = 5$. From the experimental results shown in Figure 16, we have the following interesting observation: the average size of the group returned by GSKCG queries is close to the size of the group returned by STO queries, whereas the average social cohesiveness of the group returned by GSKCG queries is much larger than that of the group returned by STO queries. Thus, we conclude that, compared with STO queries, the GSKCG query can find groups with much better social cohesiveness at the cost of a small increase in the group size. This is very meaningful for the real-life applications of collaborative spatial computing.

6 Related Work

Spatial query processing. Spatial query processing based on R-tree or its extensions has been extensively studied over the past two decades. The existing research has focused on various types of queries, including k-nearest-neighbor queries [13], [15], [17], [26], [27], range queries [25], [33], among others.

Roussopoulos et al. [27] presented an efficient branch-and-bound R-tree traversal algorithm to search the nearest neighbor object to a query point, and then extended it to k-nearest-neighbor search. Hjaltason et al. [13] proposed an incremental algorithm to efficiently query the nearest neighbor based on the R*-tree. Recently, spatial queries have been extended to incorporate text keywords, known as spatial keyword queries in the literature. Zhou et al. [37] proposed a hybrid index structure to handle both textual and spatial queries. Cong et al. [9] presented a new indexing scheme called IR-tree, which integrates the R-tree and inverted files for location-aware top-k object retrieval. Fan et al. [11] studied the problem of spatio-textual similarity queries on a new type of spatio-textual data named regions-of-interest (ROIs). They developed textual-based and grid-based filtering algorithms to efficiently find a set of objects that have large overlap with the query region and high textual similarity. Li et al. [21] proposed a novel spatial-aware interest group (SIG) query and presented two kinds of IR-tree based algorithms, interest-oriented and diameter oriented, to tackle SIG queries efficiently.

However, these works cannot deal with queries considering the social factor, for example, the GSKCG query proposed in our paper. In this paper, we also present a novel index structure, Enhanced Social-aware R-tree, which integrates the user's social relationships into the R-tree, to process GSCKG queries efficiently.

Social query processing. There have been some studies on group and team queries over social networks with the goal of finding a user group with a certain social relationship. Social groups or teams are usually cohesive subgraphs formed by users with acquaintance relations. Their acquaintance levels can be measured by several classical graph models, such as clique [12], k-core [30], and k-plex [23]. The clique model idealizes cohesive properties so that it seldom exists in real-life social networks and is difficult to compute. Both k-core and k-plex focus on a degree based model. However, k-plex is NP-complete since it restricts the subgraph size, while k-core further relaxes to achieve the linear time complexity with respect to the number of edges.

In [34], Yang et al. proposed the social-temporal group query to find a group of activity attendees with

the minimum total social distance to the query issuer. Lappas et al. [19] and Li et al. [20] studied the problem of expert team formulation which aims to find a group of experts covering all required skills and minimize the communication cost among them.

In this paper, we use k-core to model users' social relations, which is different from the previous studies. In addition, a GSKCG query takes into consideration the spatial factor.

Geo-social query processing. Efficiently processing queries considering both spatial and social constraints attracts increasingly more attention recently. A main stream is to mine users' location and social network data to find the relationships between the users and their locations. [6], [29] have shown that users with short social distances usually live geographically close. Research in this direction is still in its infancy. Liu et al. [22] proposed the circle-of-friend query to find minimal-diameter social groups. Shi et al. [31] presented a model by considering both spatial information and the social relationships between users who visit the clustered places. They extended the density-based clustering paradigm and applied it on places which are visited by users of a geo-social network. Armenatzoglou et al. [3] proposed a general framework that offers flexible data management and algorithmic design for Geo-Social Networks (GeoSNs) queries. Their architecture segregates the social, geographical and query processing modules. Each GeoSN query is processed via a transparent combination of primitive queries issued to the social and geographical modules. Yang et al. [35] proposed a socio-spatial group query to select a group of nearby attendees with a tight social relationship. They designed a new index structure called Social R-tree to integrate the users' social relationships into an R-tree for efficient query processing. This index is different from our Enhanced SaR-tree in that it is used to reduce the checking states during the enumeration. Zhu et al. [38] presented a new family of geo-social group queries with minimum acquaintance constraint (GSGQs), and also designed a new index structure named SaR-tree to accelerate the GSGQs queries. However, the SaR-tree cannot be directly adopted by our GSKCG queries due to our regional spatial factor which differs from the point spatial factor in [38].

Unlike the studies [22], [35] that aim to minimize the spatial distance among selected users, our GSKCG query aims to find a group of users whose associated regions jointly cover all query points, a brand new spatial constraint with important real-life applications. Moreover, we use a different model k-core to measure the level of social acquaintance, a more reasonable measure for practical use.

7 Conclusion

In this paper, we have introduced a new practical type of GSKCG queries that considers both users' associated spatial regions and their social acquaintance levels. A GSKCG query aims to find a minimum user group that covers all query points and that is a k-core. We have proposed an efficient algorithm `SaRBasedKCGFinder` to find the optimal solution, whose success lies in a set of effective pruning strategies and a novel index structure. Extensive experiments on two real-life datasets demonstrate the efficiency and effectiveness of our solution.

As for future work, we plan to work on the following two extensions. First, the social graph used in this paper is unweighed, we intend to extend our algorithm to support a weighted social graph. Second, in some cases, we need not an exact solution. How to design an efficient approximation algorithm with a tight approximation bound is also our future work.

Acknowledgments

This work was supported by the Research Grants Council of Hong Kong under Project Nos. HKBU211512 and HKBU12202414.

References

[1] Auto Tours USA: Self-Drive Road Trips. `http://www.autotoursusa.com/`.

[2] New Zealand Self Drive Tours. `http://www.newzealandselfdrivetours.co.nz/`.

[3] N. Armenatzoglou, S. Papadopoulos, and D. Papadias. A general framework for geo-social query processing. *Proceedings of the VLDB Endowment*, 6(10):913–924, Aug. 2013.

[4] V. Batagelj and M. Zaversnik. An o(m) algorithm for cores decomposition of networks. *CoRR*, cs.DS/0310049, 2003.

[5] J. Beal. Spatial computing for networked collaboration. In *Proceedings of the 12th International Conference on Advances in Spatial and Temporal Databases (CTS)*, pages 1–10, 2010.

[6] T. Chen, M. A. Kâafar, and R. Boreli. The where and when of finding new friends: Analysis of a location-based social discovery network. In *Proceedings of the Seventh International Conference on Weblogs and Social Media (ICWSM)*, 2013.

[7] J. Cheng, Y. Ke, S. Chu, and C. Cheng. Efficient processing of distance queries in large graphs: A vertex cover approach. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 457–468, 2012.

[8] J. Cheng, Y. Ke, S. Chu, and M. T. Ozsu. Efficient core decomposition in massive networks. In *Proceedings of the IEEE 27th International Conference on Data Engineering (ICDE)*, pages 51–62, 2011.

[9] G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. *Proceedings of the VLDB Endowment*, 2(1):337–348, Aug. 2009.

[10] Y. Doytsher, B. Galon, and Y. Kanza. Querying geo-social data by bridging spatial networks and social networks. In *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Location Based Social Networks (LBSN)*, pages 39–46, 2010.

[11] J. Fan, G. Li, L. Zhou, S. Chen, and J. Hu. Seal: Spatio-textual similarity search. *Proceedings of the VLDB Endowment*, 5(9):824–835, May 2012.

[12] F. Harary and I. C. Ross. A procedure for clique detection using the group matrix. *Sociometry*, 20(3):205–215, 1957.

[13] G. R. Hjaltason and H. Samet. Distance browsing in spatial databases. *ACM Transactions on Database System*, 24(2):265–318, June 1999.

[14] C. Jensen, W. Lee, Y. Zheng, and M. Mokbel. International workshop on location-based social networks. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Location-Based Social Networks (LBSN)*, 2011.

[15] N. Katayama and S. Satoh. The sr-tree: An index structure for high-dimensional nearest neighbor queries. In *Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 369–380, 1997.

[16] L. Kazemi, C. Shahabi, and L. Chen. Geotrucrowd: trustworthy query answering with spatial crowdsourcing. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*, pages 304–313, 2013.

[17] M. Kolahdouzan and C. Shahabi. Voronoi-based k nearest neighbor search for spatial network databases. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases (PVLDB)*, pages 840–851, 2004.

[18] A. Land and A. Doig. An automatic method for solving discrete programming problems. *50 Years of Integer Programming 1958-2008*, pages 105–132, 2010.

[19] T. Lappas, K. Liu, and E. Terzi. Finding a team of experts in social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 467–476, 2009.

[20] C.-T. Li and M.-K. Shan. Team formation for generalized tasks in expertise social networks. In *Proceedings of the IEEE 2nd International Conference on Social Computing (SOCIALCOM)*, pages 9–16, 2010.

[21] Y. Li, D. Wu, J. Xu, B. Choi, and W. Su. Spatial-aware interest group queries in location-based social networks. *Data and Knowledge Engineering*, 92:20–38, 2014.

[22] W. Liu, W. Sun, C. Chen, Y. Huang, Y. Jing, and K. Chen. Circle of friend query in geo-social networks. In *Proceedings of the 17th International Conference on Database Systems for Advanced Applications (DASFAA)*, pages 126–137, 2012.

[23] B. Mcclosky and I. V. Hicks. Combinatorial algorithms for the maximum k-plex problem. *Journal of Combinatorial Optimization*, 23(1):29–49, Jan. 2012.

[24] S. Moscovici and M. Zavalloni. The group as a polarizer of attitudes. *Journal of Personality and Social Psychology*, 12(2):125–135, Aug. 1969.

[25] B.-U. Pagel, H.-W. Six, H. Toben, and P. Widmayer. Towards an analysis of range query performance in spatial data structures. In *Proceedings of the 20th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 214–221, 1993.

[26] D. Papadias, Q. Shen, Y. Tao, and K. Mouratidis. Group nearest neighbor queries. In *Proceedings of the 20th International Conference on Data Engineering (ICDE)*, pages 301–312, March 2004.

[27] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest neighbor queries. In *Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 71–79, 1995.

[28] A. E. Saríyüce, B. Gedik, G. Jacques-Silva, K.-L. Wu, and U. V. Çatalyürek. Streaming algorithms for k-core decomposition. *Proceedings of the VLDB Endowment*, 6(6):433–444, Apr. 2013.

[29] S. Scellato, A. Noulas, R. Lambiotte, and C. Mascolo. Socio-spatial properties of online location-based social networks. In *Proceedings of the Fifth International Conference on Weblogs and Social Media (ICWSM)*, 2011.

[30] S. B. Seidman. Network structure and minimum degree. *Social Networks*, 5:269–287, 1983.

[31] J. Shi, N. Mamoulis, D. Wu, and D. W. Cheung. Density-based place clustering in geo-social networks. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 99–110, 2014.

[32] M. Sozio and A. Gionis. The community-search problem and how to plan a successful cocktail party. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 939–948, 2010.

[33] Y. Tao, X. Xiao, and R. Cheng. Range search on multidimensional uncertain data. *ACM Transactions on Database Systtem (TODS)*, 32(3), Aug. 2007.

[34] D.-N. Yang, Y.-L. Chen, W.-C. Lee, and M.-S. Chen. On social-temporal group query with acquaintance constraint. *Proceedings of the VLDB Endowment*, 4(6):397–408, Mar. 2011.

[35] D.-N. Yang, C.-Y. Shen, W.-C. Lee, and M.-S. Chen. 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)*, pages 949–957, 2012.

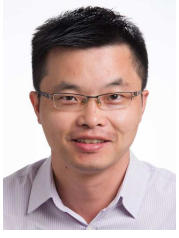
[36] Y. Zhang and S. Parthasarathy. Extracting analyzing and visualizing triangle k-core motifs within networks. In *Proceedings of the IEEE 28th International Conference on Data Engineering (ICDE)*, pages 1049–1060, 2012.

[37] Y. Zhou, X. Xie, C. Wang, Y. Gong, and W.-Y. Ma. Hybrid index structures for location-based web search. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 155–162, 2005.

[38] Q. Zhu, H. Hu, J. Xu, and W.-C. Lee. Geo-social group queries with minimum acquaintance constraint. *CoRR*, abs/1406.7367, 2014.

Yafei Li is a PhD student in the Department of Computer Science, Hong Kong Baptist University and a member of the Database Group (http://www.comp.hkbu.edu.hk/∼db). His research interests include mobile data management, location-based services, bayesian network learning, data extraction and integration.

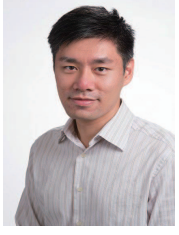
Rui Chen is a research assistant professor in the Department of Computer Science, Hong Kong Baptist University. Before that, he was a postdoctoral fellow at the University of British Columbia. He received his PhD degree in Computer Science from Concordia University. His research interests include data privacy, health informatics, data mining, and recommender systems.



Jianliang Xu is a professor in the Department of Computer Science, Hong Kong Baptist University. He received his BEng degree in computer science and engineering from Zhejiang University, Hangzhou, China, in 1998 and his PhD degree in computer science from Hong Kong University of Science and Technology in 2002. He held visiting positions at Pennsylvania State University and Fudan University. His research interests include data management, mobile/pervasive computing, and networked and distributed systems. He has published more than 120 technical papers in these areas. He is currently serving on the editorial boards of IEEE Transactions on Knowledge and Data Engineering (TKDE) and International Journal of Distributed Sensor Networks (IJDSN). He is a senior member of IEEE.

Qiao Huang is a master student in College of Computer Science and Technology, Zhejiang University, Hangzhou, China. He is also an exchange student in the Department of Computer Science, Hong Kong Baptist University. He received his BEng degree in College of Computer Science and Technology from Zhejiang University in 2012. His research interests include spatial databases and query processing.

Haibo Hu is a research assistant professor in the Department of Computer Science, Hong Kong Baptist University. Prior to this, he held several research and teaching posts at HKUST and HKBU. He received his PhD degree in Computer Science from the HKUST in 2005. His research interests include mobile and wireless data management, location-based services, and privacy-aware computing. He has published over 50 research papers in international conferences, journals and book chapters. He has received over 4 million HK dollars research grants in the capacity of PI. He is also the recipient of many awards, including ACM-HK Best PhD Paper Award and Microsoft Imagine Cup.



Byron Choi received the bachelor of engineering degree in computer engineering from the Hong Kong University of Science and Technology (HKUST) in 1999 and the MSE and PhD degrees in computer and information science from the University of Pennsylvania in 2002 and 2006, respectively. He is now an associate professor in the Department of Computer Science at the Hong Kong Baptist University. His research interests include XML, graph-structured databases, database security, and user interfaces for database systems.