Spatial-aware interest group queries in location-based social networks

Li, Yafei; Wu, Dingming; XU, Jianliang; CHOI, Koon Kau; Su, Weifeng

Published in:
Data and Knowledge Engineering

DOI:
10.1016/j.datak.2014.06.001

Published: 01/07/2014

Document Version:
Peer reviewed version

Available online at www.sciencedirect.com

ScienceDirect

Data & Knowledge Engineering 00 (2014) 1–21

Journal Logo

Spatial-aware Interest Group Queries in Location-based Social Networks

Abstract

With the recent advances in positioning and smartphone technologies, a number of social networks such as Twitter, Foursquare and Facebook are acquiring the dimension of location, thus bridging the gap between the physical world and online social networking services. Most of the location-based social networks released check-in services that allow users to share their visiting locations with their friends. In this paper, users' interests are modeled by check-in actions. We propose a new type of *Spatial-aware Interest Group* (SIG) query that retrieves 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. We prove that the SIG query problem is NP-complete. A family of efficient algorithms based on the IR-tree are thus proposed for the processing of SIG queries. Experiments on two real datasets show that our proposed algorithms achieve orders of magnitude improvement over the baseline algorithm.

Keywords: Spatial database, Query processing, Group queries, Location-based service

1. Introduction

Smartphone and social networking have been identified as the two most important innovations that have emerged in the past ten years [1]. The convergence of these two technologies has given rise to a new line of applications, namely *location-based social networks*, that bridge the gap between the physical world and online social networking services. Existing works in the literature have considered group queries in location-based social networks. [2, 3] aim at finding a group of attendees close to a rally point and ensure that the selected attendees have a good social relationship to create a good atmosphere in the activity. [4] aims to find the activity time and attendees with the minimum total social distance to the initiator. [5, 6] explore a group of experts whose skills can cover all the requirements and the communication cost among group members is low.

Most of the location-based social networks released check-in services that allow users to share their visiting locations with their friends. These locations, considered as spatial objects, are usually associated with a few tags that describe the features of those locations, e.g., spatial object 'starbucks' with tags 'food', 'beverage', and 'coffee'. If a user checks in the spatial object 'starbucks', the user may be interested in 'food', 'beverage', or 'coffee'. These voluntary check-in actions reflecting the users' interests can benefit many applications. Utilizing such information, this paper proposes a new type of *Spatial-aware Interest Group* (SIG) query that retrieves a user group of size k where each user is interested in the query keywords and the users are close to each other in the Euclidean space. Different from existing work, the SIG query retrieves a user group of size k that maximizes a ranking function combining the diameter of the group (i.e., the distance between the farthest pair of users) and the group's interest in the query keywords. Such queries are useful in many scenarios. For example, consider that a company wants to hold promotion campaigns in some regions. The company is interested in identifying the regions containing potential customers who are interested in the features (query keywords) of the product promoted. Another example is for interest-based group gathering. Query keyword 'movie' may find a group of nearby people who are movie lovers, while query keyword

'NBA' could retrieve a group of nearby people who like playing basketballs. Note that the group size in these queries is usually constrained due to limited venue capacity and/or financial budget.

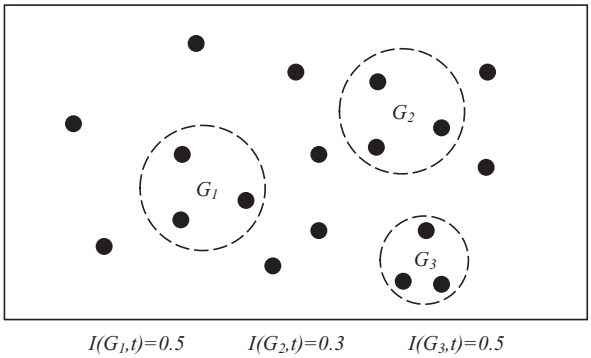


Figure 1. A General Case of 3-Size SIG Query

Figure 1 shows an example 3-size SIG query over several users shown as black dots. Consider three user groups G_1, G_2 and G_3, where $I(G_i, t)$ means group G_i's interest in query keyword t. Bigger value indicates higher interest. Comparing group G_1 with group G_2, group G_1 is more preferable, since the interest of G_1 on query keyword t is bigger than that of G_2, although G_1 and G_2 have the same layout in geo-space. Considering group G_1 with group G_3, group G_3 is more preferable, since the inter-user distances in G_3 is smaller than that of G_1, although G_1 and G_3 have the same interest value on query keyword t. Overall, group G_3 is the most preferable among these three groups.

To process SIG queries, a naive method is to enumerate all possible groups of size k, and then find the group with the highest ranking score. Obviously, this method is inefficient given a large number of possible groups. To avoid enumerating all possible groups, in this paper, we propose two efficient algorithms, namely Interest Oriented Algorithm (IOAIR) and Diameter Oriented Algorithm (DOAIR), based on the IR-tree [7] for the processing of SIG queries. The basic idea is to derive an upper bound on the ranking score for each possible user group and construct user groups in an iterative fashion; if the ranking score of the current found group is higher than the upper bounds of the unseen groups, the query processing can be stopped by returning the current found group as the result. The two proposed algorithms adopt different orders in the construction of user groups: Algorithm IOAIR explores the search space by group interest while algorithm DOAIR proceeds by group diameter. We conduct extensive experiments on two real datasets to evaluate the performance of the proposed algorithms. The experiment results show that our algorithms achieve orders of magnitude improvement over the baseline algorithm.

Our contributions made in this paper can be summarized as follows:

1. Proposing a new type of query SIG that finds a k-size maximum interest group in location-based social networks and proving that the SIG query problem is NP-complete.
2. Developing two efficient algorithms IOAIR and DOAIR based on the IR-tree for the processing of SIG queries.
3. Validating the performance efficiency of the proposed query processing algorithms by empirical evaluation.

The rest of the paper is organized as follows. Section 2 presents the problem definition. Section 3 presents two efficient algorithms based on IR-tree for the processing of SIG queries. Section 4 shows the empirical study of our proposed algorithms on two real datasets. Section 5 reviews recent work on related queries in location-based social networks. Section 6 concludes the paper and outlines possible future work.

2. Problem Definition

Let $\mathcal{D}$ be a set of spatial objects. Each spatial object p is associated with a set of tags $p.\Gamma$. Let $\mathcal{U}$ be a set of users. Each user $u \in \mathcal{U}$ is a triple (id, λ, ν), where id is the user's identifier, λ is the user's location, and ν is a vector of the user's interests for the tags that are associated with the spatial objects checked in by the user. The interest value for a set of tags is defined in Definition 1. We may use interest and interest value interchangeably.

Definition 1. Let D_u be the set of spatial objects checked in by user u and let D_t be the set of spatial objects that are associated with tag t. Function $Count(u, p)$ counts the times of spatial object p checked in by user u. User u's interest value for tag t is computed as:[1]

$$I(u,t) = \frac{\sum_{p \in D_u \wedge p \in D_t} Count(u,p)}{\sum_{p \in D_u} Count(u,p)}. \tag{1}$$

Given a set of tags T, if a user's interest value for every tag $t \in T$ is positive, we say the user *fully covers* T. Specifically, the interest value of a user u for a tag set T is defined as:

$$I(u,T) = \sum_{t \in T} I(u,t). \tag{2}$$

As an example, Table 1 shows an interest vector. Higher values indicate higher interest. For example, the user is more interested in 'hotel' than 'sport'. The user's interest value for tags 'hotel' and 'sport' is 0.36+0.2=0.56.

Table 1. Example Interest Vector

Tag	movie	airport	hotel	music	sport
Interest Value	0.10	0.20	0.36	0.14	0.20

We follow existing works [8][9][10] to define a ranking function as a weighted sum of the normalized group interest and group diameter.[2] It ranks a user group G_k of size k with regard to a query q, denoted by $rank_q(G_k)$:

$$rank_q(G_k) = \alpha \frac{I(G_k, q.T)}{I_{max}(q.T)} + (1-\alpha)(1 - \frac{D(G_k)}{D_{max}}), \tag{3}$$

Here $q.T$ is a set of query keywords that belong to the tag space of the dataset, $I(G_k, q.T)$ is the group interest on the query keywords $q.T$, which is defined as the minimum interest of the users in the group if G_k fully covers $q.T$, i.e.,

$$I(G_k, q.T) = \begin{cases} \min\{I(u, q.T) \mid u \in G_k\} & \text{if the users in } G_k \text{ jointly fully cover } q.T \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

and $D(G_k)$ is the diameter of group G_k, i.e., the Euclidean distance between the farthest pair of users in the group,

$$D(G_k) = \max\{\|u_i.\lambda \;\; u_j.\lambda\| \mid u_i, u_j \in G_k\}, \tag{5}$$

where $\|u_i.\lambda \;\; u_j.\lambda\|$ is the Euclidean distance between two users. Parameter $\alpha \in [0,1]$ is used to balance the group interest and the group diameter.

Definition 2. A *Spatial-aware Interest Group* (SIG) query $q = (T, k)$ consists of two parameters: a set of keywords $q.T$ and the size k of the requested user group. It retrieves a user group of size k where each user is interested in the query keywords and the users are close to each other in the Euclidean space, meaning that the ranking function (Eq. 3) is maximized.

Example 1. Figure 2 illustrates an example SIG query with three different values of α in the ranking function (Eq. 3). The circles, squares, and triangles in the figure depict the locations of a set of users. Given a SIG query q, the sizes of those shapes indicate the user interests for a set of query keywords $q.T$. The bigger the size, the higher the user interest. Query q requests a user group of size 4 that maximizes the ranking function. The gray circles are the result group when $\alpha = 0$, i.e., only the group diameter is considered. The gray squares are the result group when $\alpha = 0.5$. The gray triangles represent the query result when $\alpha = 1$, i.e., only the group interest is considered.

[1] The definition of user interest based on check-in counts is adopted due to its simplicity. We can also use other models, such as user ratings or likes/dislikes, to quantify the user interest; and no any modification on the query processing algorithm is needed.

[2] Note that the group interest and diameter factors are not directly comparable. In order to treat these two factors fairly, we use the global maximum group diameter D_{max} and maximum group interest $I_{max}(q.T)$ to normalize them so that they will be kept in the same value domain [0, 1].

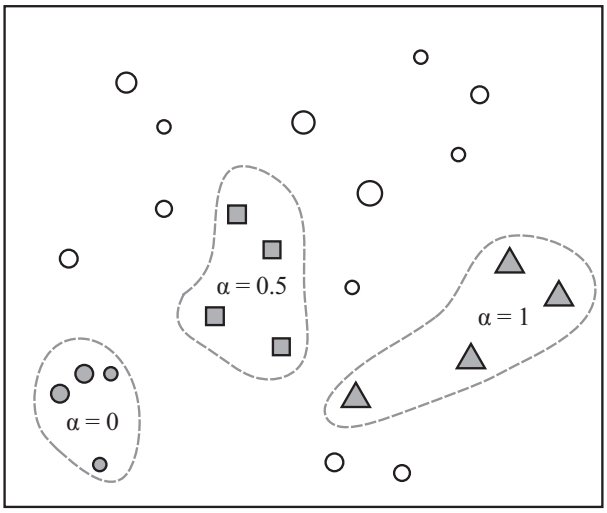


Figure 2. Example SIG Query

Theorem 1. *The SIG query problem is NP-complete.*

Proof. We establish the hardness by a reduction from a classical NP-complete problem, namely the minimum set cover problem (MSC). An instance of the MSC problem consists of a universe set $U = \{e_1, e_2, \ldots, e_n\}$, a collection of sets $S = \{S_1, S_2, \ldots, S_m\}$, where S_i is a subset of U and an integer k. The decision problem of MSC is to find whether there is a subset $S' \subseteq S$, such that $|S'| \leq k$ and the union of S' fully covers U.

Given an instance of MSC, we construct an instance of SIG query $q = (T, k)$ on a set of users. Each element e_i in U is a keyword t_i in $q.T$, each set S_i is a user u_i, and the elements in S_i are u_i's interests (keywords). We set the value α of the SIG query to 1. We remark that $I_{max}(q.T)$ is a constant under this setting. Thus, maximizing $rank_q(G_k)$ is equivalent to maximizing $I(G_k, q.T)$.

Suppose that we have a PTIME algorithm A that returns the query answer $G_k = \{u'_1, u'_2, \ldots, u'_k\}$ of a SIG query. There are two cases. Case 1: If $q.T$ is fully covered by the interests of G_k, then $\{S'_1, S'_2, \ldots, S'_k\}$ fully covers U and its size is k. Therefore, a solution of the MSC is found. Case 2: If G_k does not fully cover $q.T$, then there does not exist another group G'_k, such that the interests of G'_k fully cover $q.T$. Otherwise, G'_k would be returned as an answer as in Case 1. Therefore, with such a G_k, one can conclude that the MSC instance does not have a solution. By using A, the MSC problem is solved in PTIME, a contradiction. Therefore, there does not exist a PTIME algorithm A that can solve the SIG query problem.

3. Proposed Approaches

In this section, we present two efficient algorithms, namely Interest Oriented Algorithm (IOAIR) and Diameter Oriented Algorithm (DOAIR), for the processing of SIG queries based on the IR-tree [7]. Section 3.1 introduces the index structure IR-tree. Section 3.2 presents the basic ideas of the two algorithms. Sections 3.3 and 3.4 elaborate the two algorithms.

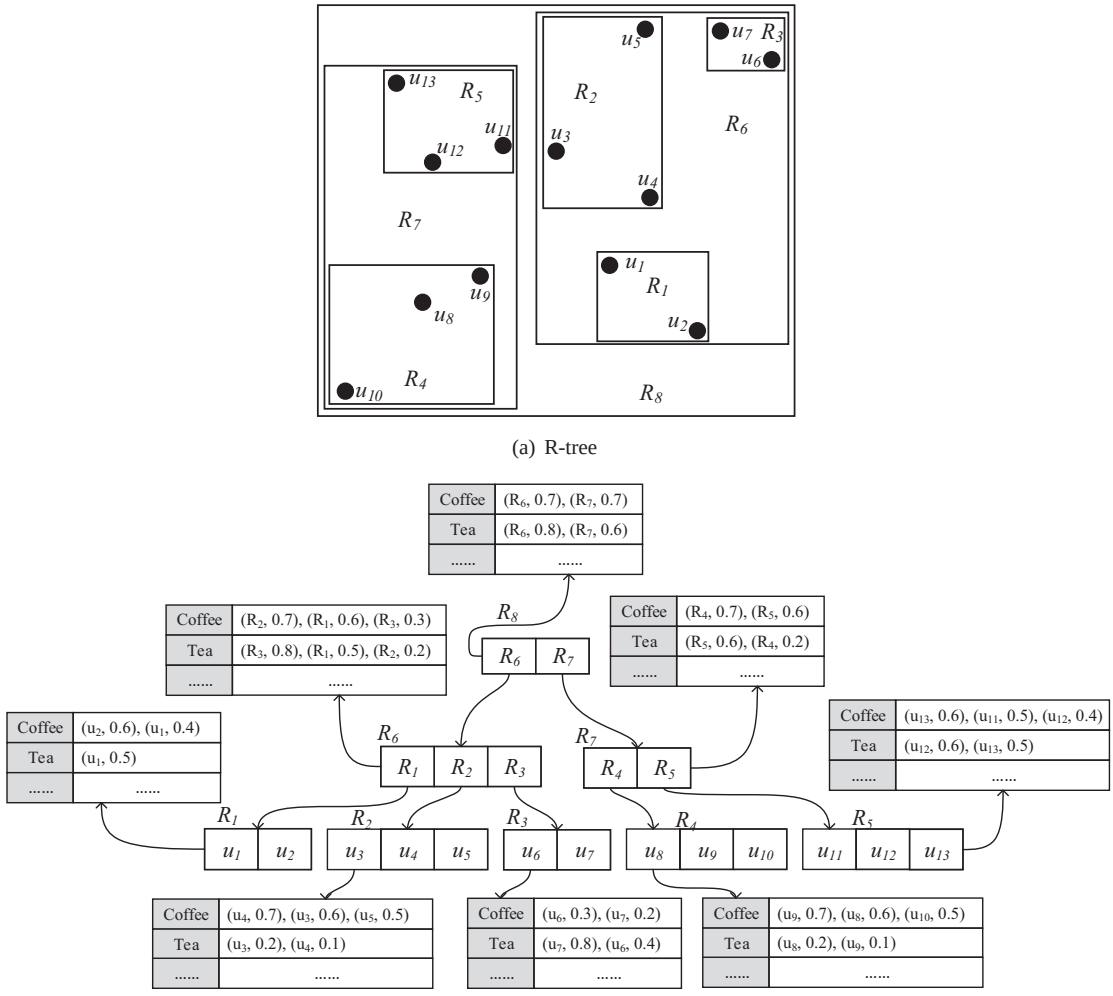
3.1. Preliminary: IR-Tree

We adopt the IR-tree index structure [7], where users are considered as spatial objects, users' locations and interest vectors are considered as the locations and documents of objects, respectively. Figure 3(a) and Figure 3(b) show the users' locations and its corresponding IR-tree. IR-tree is essentially an R-tree attached with inverted files. The leaf nodes in the IR-tree contain a number of entries of the form $(u, u.\lambda)$, where u refers to a user and $u.\lambda$ is the MBR

(minimum bounding rectangle) of the user's location. Each leaf node also includes a pointer to an inverted file that indexes the interest vectors of the users stored in the node.

Each non-leaf node in the IR-tree includes several entries in the form of (ch, mbr), where ch is the identifier of a child node and mbr is the MBR of all rectangles in the child nodes. Each non-leaf node also includes a pointer to an inverted file that indexes the pseudo interest vectors of the entries stored in the node. The pseudo interest vector of an entry contains all the tags that appear in its child nodes. The interest value for each tag is the maximum value in the child nodes.

We remark that the user locations (e.g., when they are referred to home/office addresses) may not be frequently updated. When the user location changes, the IR-tree should be updated accordingly to support efficient query processing. Fortunately, this can be well handled by the embedded updating mechanism of IR-tree, whose efficiency has been demonstrated in [7].



(a) R-tree

(b) IR-tree

Figure 3. Tree Index Structure

3.2. Overview

To avoid enumerating all possible groups, algorithms IOAIR and DOAIR construct groups in special orders. If the ranking score of the current found group is higher than the upper bounds on the ranking scores of the unseen groups,

the current found group is returned as the result. The derivation of the upper bound on the ranking score of a group is shown in Theorem 2.

Theorem 2. *Let D_{min} be the distance between the closest pair of users in the dataset. An upper bound on the ranking score of a group $G_k(u_i)$ of size k containing user u_i is*

$$rank_q^u(G_k(u_i)) = \alpha \frac{I(u_i, q.T)}{I_{max}(q.T)} + (1-\alpha)(1 - \frac{D_{min}}{D_{max}}). \quad (6)$$

Proof. According to Eq. 4, we have $I(G_k(u_i), q.T) \leq I(u_i, q.T)$. And since $D_{min} \leq D(G_k(u_i))$, we derive

$$\begin{aligned} rank_q^u(G_k(u_i)) &= \alpha \frac{I(u_i, q.T)}{I_{max}(q.T)} + (1-\alpha)(1 - \frac{D_{min}}{D_{max}}) \\ &\geq \alpha \frac{I(G_k(u_i), q.T)}{I_{max}(q.T)} + (1-\alpha)(1 - \frac{D(G_k(u_i))}{D_{max}}) \\ &= rank_q(G_k(u_i)). \end{aligned}$$

3.3. Interest Oriented Algorithm

Interest Oriented Algorithm (IOAIR) classifies groups in terms of users' interests. Let set $\mathcal{G}_k$ contain all possible user groups of size k. Let set $\mathcal{G}_k(u_i)$ cover all the groups of size k that contains user u_i and have the same level of interest as u_i, i.e., $\forall G \in \mathcal{G}_k(u_i)(I(G, q.T) = I(u_i, q.T))$. Obviously, $\cup_{u_i \in \mathcal{U}} \mathcal{G}_k(u_i) = \mathcal{G}_k$. Algorithm IOAIR follows the descending order of the user interest and iteratively constructs the group $G_k(u_i)$ with the maximum ranking score in $\mathcal{G}_k(u_i)$. If the ranking score of the current constructed group $G_k(u_i)$ is higher than the upper bound on the ranking score of the next group $G_k(u_{i+1})$ (termination condition), the current found group is returned as the result. The correctness of the termination condition is guaranteed by Lemma 1. The correctness of algorithm IOAIR is guaranteed by Theorem 3.

Lemma 1. *Let $S = \{u_1, u_2, \cdots, u_n\}$ be a sorted list of users in descending order of their interests. If $rank_q(G_k(u_i)) > rank_q^u(G_k(u_j))$, we have $rank_q(G_k(u_i)) > rank_q(G_k(u_m))$, where $k \leq i < j \leq m \leq n$.*

Proof. For $j \leq m \leq n$, we have $I(u_j, q.T) \geq I(u_m, q.T)$. According to Eq. 6, we derive $rank_q^u(G_k(u_j)) \geq rank_q^u(G_k(u_m))$. Hence, we get $rank_q(G_k(u_i)) > rank_q^u(G_k(u_j)) \geq rank_q^u(G_k(u_m)) \geq rank_q(G_k(u_m))$ based on Theorem 2.

Theorem 3. *Algorithm* IOAIR *finds the correct answer to a SIG query.*

Proof. We prove it by contradiction. Assume that given a SIG query q, algorithm IOAIR returns G as the result. Now suppose there exists G' with the maximum ranking score such that $rank_q(G) < rank_q(G')$. Let u_i be the user with the minimum interest in G and u_i' be the user with the minimum interest in G'. Hence, we have G and G' are the groups with maximum ranking score in $\mathcal{G}_k(u_i)$ and $\mathcal{G}_k(u_i')$, respectively. There are three possible cases. (1) If $I(u_i, q.T) < I(u_i', q.T)$, algorithm IOAIR first considers $\mathcal{G}_k(u_i')$, and then $\mathcal{G}_k(u_i)$. According to Lemma 1, we have $rank_q(G') \geq rank_q^u(G) \geq rank_q(G)$. Thus, algorithm IOAIR must return the group G' in $\mathcal{G}_k(u_i')$, not G in $\mathcal{G}_k(u_i)$. (2) If $I(u_i, q.T) = I(u_i', q.T)$, we have $\mathcal{G}_k(u_i') = \mathcal{G}_k(u_i)$. Algorithm IOAIR must return the group G' since $rank_q(G) < rank_q(G')$. (3) If $I(u_i, q.T) > I(u_i', q.T)$, algorithm IOAIR first considers $\mathcal{G}_k(u_i)$, and then $\mathcal{G}_k(u_i')$. According to Lemma 1, we have $rank_q(G) \geq rank_q^u(G') \geq rank_q(G')$, which contradicts the assumption that $rank_q^u(G) < rank_q(G')$. Hence, the correctness of algorithm IOAIR is proved.

Algorithm 1 shows the pseudo code of the IOAIR algorithm. The candidate group G_k is initialized as the k-sized user group with the minimum diameter in (line 1). It processes users in descending order of their interests for the query keywords (line 3) by calling function GetNextUser that adopts the Threshold Algorithm [11]. For the current obtained user u_i, function IOAIRGetNextGroup constructs a group of size k containing u_i with the maximum ranking score (i.e., minimum diameter), denoted as $G_k(u_i)$, where $I(G_k(u_i), q.T) = I(u_i, q.T)$ (line 5). The constructed group $G_k(u_i)$ is assigned as the candidate group if its ranking score is higher than that of the candidate group G_k (lines 6 and 7). If the ranking score of the candidate group is higher than the upper bound on the ranking score of $G_k(u_{i+1})$ that is the group of size k containing the next user u_{i+1} with the maximum ranking score, the algorithm returns the candidate group as the result and terminates (lines 8 and 9).

Algorithm 1 IOAIR(Integer k, Keywords T, InvertedFile *invf*, IRTree *irtree*)

1: Result $G_k \leftarrow$ the k-sized user group with the minimum diameter;
2: $D_c \leftarrow \infty$;
3: **while** $u_i, u_{i+1} \leftarrow$ GetNextUser(T, *invf*) **do**;
4: Update D_c according to Equation 7;
5: $G_k(u_i) \leftarrow$ IOAIRGetNextGroup(*irtree*, u_i, k, D_c, T);
6: **if** $rank_q(G_k(u_i)) > rank_q(G_k)$ **then**
7: $G_k \leftarrow G_k(u_i)$;
8: **if** $rank_q(G_k) > rank_q^u(G_k(u_{i+1}))$ **then**
9: Return G_k;
10: Return G_k;

In order to find group $G_k(u_i)$ such that $I(G_k(u_i), q.T) = I(u_i, q.T)$ with the maximum ranking score in $\mathcal{G}_k(u_i)$, function IOAIRGetNextGroup (Algorithm 2) uses the IR-tree to retrieve the users who have higher interest values than does u_i and puts them in G'_k. Taking the advantage of the IR-tree where each entry in each node has an upper bound on the users' interests contained in the subtree pointed to by the entry, it is able to prune the nodes whose interest is smaller than the interest of user u_i, since no user in the subtree can have a larger interest than does user u_i (line 17). Since the interest of group $G_k(u_i)$ is determined, constructing $G_k(u_i)$ with the maximum ranking score is equivalent to find a group of size k with the minimum diameter from G'_k (line 14). We apply backtracking method to enumerate all possible k size groups, each group also needs to be checked if it fully covers T.

Algorithm 2 IOAIRGetNextGroup(IRTree *irtree*, User u_i, Integer k, Double D_c, Keywords T)

1: *Queue* $\leftarrow$ NewPriorityQueue();
2: *Queue*.Enqueue(*irtree.root*, 0);
3: Add u_i to G'_k;
4: **while** *Queue* is not empty **do**
5: Entry $e \leftarrow$ *Queue*.Dequeue();
6: **if** e refers to a user **then**
7: **if** $D(G_k) \leq \|u_i\ e\|$ **then**
8: **if** $D(G_k) < D_c$ **then**
9: Return G_k;
10: **else**
11: Return NULL;
12: Add e to G'_k;
13: **if** G'_k contains more than k users **then**
14: $G_k \leftarrow$ select the group of size k with the minimum diameter from G'_k;
15: **else**
16: **for** each entry e' in the node pointed to by e **do**
17: **if** the interest of $e' >$ the interest of u_i **then**
18: **if** $\|u_i\ e'\| < D_c$ **then**
19: *Queue*.Enqueue($e', \|u_i\ e'\|$);
20: Return G_k;

Early Stop. Let G'_k contain all the users with higher interests than u_i. It is possible to prune some users in G'_k so that $G_k(u_i)$ can be quickly found from G'_k. Function IOAIRGetNextGroup considers the users with higher interests than u_i in ascending order of their distances to u_i. If $k-1$ users have been obtained, a candidate group of size k including u_i is formed. If the diameter of the candidate group is not greater than the distance between u_i and the newly added user (line 7), the candidate group is the one with the maximum ranking score. Otherwise, the candidate group is updated by considering the newly added user. The correctness is guaranteed by Theorem 4 (illustrated by Example 2).

Theorem 4. *Let $S = \{u_1, u_2, \cdots, u_m, u_{m+1}, \cdots, u_n\}$ be a sorted list of users with higher interests than u_i and in ascending order of their distances to user u_i. Let $G_k(u_i)$ be the user group of size k containing u_i with the maximum ranking score calculated from $S' = \{u_1, u_2, \cdots, u_m\}$. If $\|u_i\ u_{m+1}\| \geq D(G_k(u_i))$, $G_k(u_i)$ is the user group of size k containing u_i with the maximum ranking score calculated from S.*

Proof. Suppose we can find a group $G'_k(u_i)$ of size k containing u_i from $S'' = \{u_1, u_2, \cdots, u_m, \cdots, u_{m+j}\}$ where $1 \leq j \leq n - m$, such that $rank_q(G'_k(u_i)) > rank_q(G_k(u_i))$. Then we have $\exists j(u_{m+j} \in G'_k(u_i))$. Since $\forall u \in S''(I(u_i, q.T) \leq I(u, q.T))$, we have $I(G'_k(u_i), q.T) = I(G_k(u_i), q.T)$ and derive $D(G'_k(u_i)) < D(G_k(u_i))$. Since $u_{m+j} \in G'_k(u_i)$, we have $D(G'_k(u_i)) \geq \|u_i\ u_{m+j}\|$. Since $\|u_i\ u_{m+j}\| \geq D(G_k(u_i))$, we have $D(G'_k(u_i)) \geq \|u_i\ u_{m+j}\| \geq D(G_k(u_i))$ that contradicts $D(G'_k(u_i)) < D(G_k(u_i))$ derived before and thus complete the proof.

Example 2. We illustrate Theorem 4 in Figure 4. Let $S = \{u_1, u_2, u_3, u_4, u_5\}$ be a sorted list of users with higher interests than u_i and in ascending order of their distances to user u_i. Let $G_4(u_i) = \{u_i, u_1, u_2, u_3\}$ be the user group of size 4 containing u_i with the maximum ranking score calculated from $S' = \{u_1, u_2, u_3\}$. The diameter is $D(G_4(u_i)) = \|u_1\ u_2\|$. Next, we consider u_4 and have $\|u_i\ u_4\| < D(G_4(u_i))$. Hence, we obtain a new group $G_4(u_i) = \{u_i, u_2, u_3, u_4\}$ from $S'' = \{u_1, u_2, u_3, u_4\}$ with the maximum ranking score and diameter $D(G_4(u_i)) = \|u_i\ u_4\|$. We then consider u_5 and have $D(G_4(u_i)) < \|u_i\ u_5\|$. Theorem 4 guarantees that $G_4(u_i) = \{u_i, u_2, u_3, u_4\}$ is the user group of size 4 containing u_i with the maximum ranking score calculated from S.

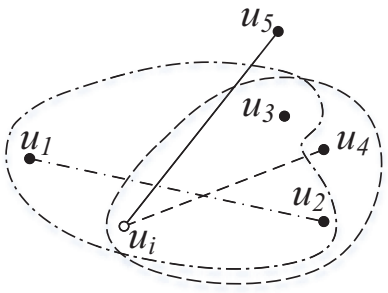


Figure 4. Example of Theorem 4

Diameter Constraint. When retrieving user group $G_k(u_i)$, it is not necessary to consider all the users whose interests are higher than u_i. We propose a diameter constraint D_c that can be used to prune the search space (Theorem 5) so that less users are considered when selecting a group $G_k(u_i)$ with the maximum ranking score.

Lemma 2. *$rank_q(G_k(u_i)) > rank_q(G_k(u_j)) \iff D(G_k(u_i)) < D_c$, where*

$$D_c = D_{max}(1 - \frac{rank_q(G_k(u_j)) - \alpha\frac{I(u_i,q.t)}{I_{max}(q.t)}}{(1-\alpha)}). \tag{7}$$

The proof can be easily derived based on Eq. 3 and thus omitted.

Theorem 5. *If $rank_q(G_k(u_i)) > rank_q(G_k(u_j))$ and $\|u_i\ u_m\| \geq D_c$, we have $u_m \notin G_k(u_i)$.*

Proof. We prove it by contradiction. Suppose $u_m \in G_k(u_i)$. Since $\|u_i\ u_m\| \leq D(G_k(u_i))$, we have

$$\begin{aligned} rank_q(G_k(u_i)) &= \alpha\frac{I(u_i,q.T)}{I_{max}(q.T)} + (1-\alpha)(1 - \frac{D(G_k(u_i))}{D_{max}}) \\ &\leq \alpha\frac{I(u_i,q.T)}{I_{max}(q.T)} + (1-\alpha)(1 - \frac{\|u_i\ u_m\|}{D_{max}}) \\ &\leq \alpha\frac{I(u_i,q.T)}{I_{max}(q.T)} + (1-\alpha)(1 - \frac{D_c}{D_{max}}) \\ &= rank_q(G_k(u_j)). \end{aligned}$$

It contradicts the condition $rank_q(G_k(u_i)) > rank_q(G_k(u_j))$.

Lemma 2 indicates that the condition of a group with smaller interest having higher ranking score is that its diameter must be small enough. Based on Lemma 2, Theorem 5 guarantees that it prunes the users whose distances to u_i is larger than D_c, since it is impossible to construct a group containing those users and having higher ranking score than does the candidate group (illustrated by Example 3). Hence, the search space is pruned (line 18 in Algorithm 2). Only the group with higher ranking score than the candidate group is returned (line 8 in Algorithm 2). The value of D_c is updated when a user group with higher ranking score is found (line 7 in Algorithm 1).

In order to facilitate our example description, we set $q.T$ = 'coffee', and the value of user's interest in $q.T$ is shown in Figure 3(b).

Example 3. Figure 3(a) shows the location layout of the users appearing in the IR-tree of Figure 3(b). Let D_{max} = 100, α = 0.5, I_{max} = 1.0, $I(u_1,\ q.T)$ = 0.4, and the current maximum ranking score be 0.6. Suppose the current processing group is $G_k(u_1)$. Based on Lemma 2, we can obtain D_c = 20. Thus the diameter of $G_k(u_1)$ should be less than 20. Figure 5 shows the distances between u_1 and IR-tree nodes or its neighbor users. With the diameter constraint D_c, we do not need to consider tree nodes $\{R_3\}$ and users $\{u_6, u_7, u_{10}, u_{13}\}$ during the query processing.

u_1	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8
$\|u_1\ R_i\|$	0	6	20	9	10	0	10	0

u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	u_{10}	u_{11}	u_{12}	u_{13}
$\|u_1\ u_i\|$	7	8	6	18	21	20	17	9	25	11	15	24

Figure 5. Distance between u_1 and its neighbors

3.4. Diameter Oriented Algorithm

Diameter Oriented Algorithm (DOAIR) classifies groups in terms of group diameters. Let set $\mathcal{G}_k$ contain all possible user groups of size k. Let set $\mathcal{G}_k(u_i, \cdot)$ cover all the groups of size k, taking user u_i as one end of the group diameter. Obviously, $\cup_{u_i \in \mathcal{U}} \mathcal{G}_k(u_i, \cdot) = \mathcal{G}_k$. Note that $\mathcal{G}_k(u_i, \cdot)$ may be an empty set. Algorithm DOAIR follows the descending order of the user interest and constructs the group $G_k(u_i, \cdot)$ with the maximum ranking score in $\mathcal{G}_k(u_i, \cdot)$. If the ranking score of the current found group $G_k(u_i, \cdot)$ is higher than the upper bound on the ranking score of the next group $G_k(u_{i+1}, \cdot)$ (termination condition), the current found group is returned as the result. The correctness of the termination condition is guaranteed by Lemma 3. The correctness of algorithm DOAIR is guaranteed by Theorem 6.

Lemma 3. *Let $S = \{u_1, u_2, \cdots, u_n\}$ be a sorted list of users in descending order of their interests. If $rank_q(G_k(u_i, \cdot)) > rank^u_q(G_k(u_j, \cdot))$ where $rank^u_q(G_k(u_j, \cdot)) = rank^u_q(G_k(u_j))$ (Eq. 6), we have $rank_q(G_k(u_i, \cdot)) > rank_q(G_k(u_m, \cdot))$, where $k \le i < j \le m \le n$.*

Proof. For $j \le m \le n$, we have $I(u_j, q.T) \ge I(u_m, q.T)$. According to Eq. 6, we derive $rank^u_q(G_k(u_j, \cdot)) \ge rank^u_q(G_k(u_m, \cdot))$. Hence, we get $rank_q(G_k(u_i, \cdot)) > rank^u_q(G_k(u_j, \cdot)) \ge rank^u_q(G_k(u_m, \cdot)) \ge rank_q(G_k(u_m, \cdot))$ based on Theorem 2.

Theorem 6. *Algorithm* DOAIR *find the correct answer to a SIG query.*

Proof. We prove it by contradiction. Assume that given a SIG query q, algorithm DOAIR returns G as the result. Now suppose there exists G' with the maximum ranking score such that $rank_q(G) < rank_q(G')$. Suppose G and G' are the groups with maximum ranking score in $\mathcal{G}_k(u_i, \cdot)$ and $\mathcal{G}_k(u'_i, \cdot)$, respectively. There are three possible cases. (1) If $I(u_i, q.T) < I(u'_i, q.T)$, algorithm DOAIR first considers $\mathcal{G}_k(u'_i, \cdot)$, and then $\mathcal{G}_k(u_i, \cdot)$. According to Lemma 3, we have $rank_q(G') \ge rank^u_q(G) \ge rank_q(G)$. Thus, algorithm DOAIR must return the group G' in $\mathcal{G}_k(u'_i, \cdot)$, not G in $\mathcal{G}_k(u_i, \cdot)$. (2) If $I(u_i, q.T) = I(u'_i, q.T)$, we have $\mathcal{G}_k(u'_i, \cdot) = \mathcal{G}_k(u_i, \cdot)$. Algorithm DOAIR must return the group G' since $rank_q(G) < rank_q(G')$. (3) If $I(u_i, q.T) > I(u'_i, q.T)$, algorithm DOAIR first considers $\mathcal{G}_k(u_i, \cdot)$, and then $\mathcal{G}_k(u'_i, \cdot)$. According to Lemma 3, we have $rank_q(G) \ge rank^u_q(G') \ge rank_q(G')$, which contradicts the assumption that $rank^u_q(G) < rank_q(G')$. Hence, the correctness of algorithm DOAIR is proved.

Algorithm 3 shows the pseudo code of the DOAIR algorithm. The candidate group G_k is initialized as the k-sized user group with the minimum diameter (line 1). It processes users in descending order of their interests for the query keywords (line 3) by calling function GetNextUser that adopts the Threshold Algorithm [11]. For the current obtained user u_i, function DOAIRGetNextGroup constructs a group of size k with the maximum ranking score, taking user u_i as one end of the group diameter, denoted as $G_k(u_i, \cdot)$ (line 7). If its ranking score is higher than that of the candidate group G_k (lines 8 and 9). If the ranking score of the candidate group is higher than the upper bound of $G_k(u_{i+1}, \cdot)$, the algorithm then returns the candidate group as the result and terminates (lines 10 and 11). Algorithm DOAIR is able to skip the construction of group $G_k(u_i, \cdot)$ if the distance D_{u_i} between u_i and its nearest neighbor is larger than D_c and the candidate group interest is also higher than u_i's interest (lines 5 and 6), meaning that it is impossible to find a group taking u_i as one end of the group diameter with higher ranking score than the candidate group. Theorem 7 guarantees the correctness of this pruning step.

Algorithm 3 DOAIR(Integer k, Keywords T, InvertedFile *invf*, IRTree *irtree*)

```
1:  Result G_k ← the k-sized user group with the minimum diameter;
2:  D_c ← ∞;
3:  while u_i, u_{i+1} ← GetNextUser(T, invf) do;
4:      Update D_c according to Equation 7;
5:      if D_{u_i} > D_c then
6:          Continue;
7:      G_k(u_i, ·) ← DOAIRGetNextGroup(irtree, u_i, k, D_c, T);
8:      if rank_q(G_k(u_i, ·)) > rank_q(G_k) then
9:          G_k ← G_k(u_i, ·);
10:     if rank_q(G_k) > rank^u_q(G_k(u_{i+1}, ·)) then
11:         Return G_k;
12: Return G_k;
```

Theorem 7. *Let G_k be the candidate group and D_{u_i} be the distance between u_i and its nearest neighbor. If $D_{u_i} > D_c$, we have $rank_q(G_k) > rank_q(G_k(u_i, \cdot))$ where*

$$D_c = D_{max}(1 - \frac{rank_q(G_k) - \alpha\frac{I(u_i,q.t)}{I_{max}(q.t)}}{(1-\alpha)}).$$

Proof. We derive $rank_q(G_k) > rank_q(G_k(u_i, \cdot))$ as follows:

$$\begin{aligned} rank_q(G_k(u_i,\cdot)) &= \alpha\frac{I(G(u_i,\cdot),q.T)}{I_{max}(q.T)} + (1-\alpha)(1-\frac{D(G_k(u_i,\cdot))}{D_{max}}) \\ &\leq \alpha\frac{I(u_i,q.T)}{I_{max}(q.T)} + (1-\alpha)(1-\frac{D_{u_i}}{D_{max}}) \\ &< \alpha\frac{I(u_i,q.T)}{I_{max}(q.T)} + (1-\alpha)(1-\frac{D_c}{D_{max}}) \\ &= rank_q(G_k). \end{aligned}$$

In order to find group $G_k(u_i, \cdot)$ with the maximum ranking score in $\mathcal{G}_k(u_i, \cdot)$, function DOAIRGetNextGroup (Algorithm 4) uses the IR-tree to retrieve the users in ascending order of their distances to u_i (line 19). For an encountered user e, it tries to construct a group G_k of size k with diameter $\overline{u_i\, e}$ (lines 9 and 10). To avoid enumerating all possible diameter $\overline{u_i\, e}$ and find out group $G_k(u_i, \cdot)$ efficiently, an early stop condition (line 12) and two interest constraints (lines 14 and 18) are designed. The diameter contraint (Theorem 5) is also applied here (line 18).

Early Stop. Function DOAIRGetNextGroup considers the users in ascending order of their distances to u_i. Let G_k be the current found group with diameter $\overline{u_i\, e}$. If the interest of G_k equals the interest of u_i, group G_k is the group with the maximum score in $\mathcal{G}_k(u_i, \cdot)$. All the rest users farther than e from u_i do not need to be considered. The correctness is guaranteed by Theorem 8.

Algorithm 4 DOAIRGetNextGroup(IRTree *irtree*, User u_i, Integer k, Double D_c, Keywords T)

```
1: Queue ← NewPriorityQueue();
2: Queue.Enqueue(irtree.root, 0);
3: I_c ←0,
4: Add u_i to G'_k;
5: while Queue is not empty do
6:     Entry e ← Queue.Dequeue();
7:     if e refers to a user then
8:         Add e to G'_k;
9:         if G'_k contains more than k users then
10:            G_k ← GetCurrentResult(G'_k, u_i, e, T);
11:            if G_k is not empty then
12:                if the interest of G_k = the interest of u_i then
13:                    Return G_k;
14:                Update Queue and G'_k, delete the users whose interest ≤ the interest of G_k (Theorem 10).
15:    else
16:        for each entry e' in the node pointed to by e do
17:            Update I_c according to Theorem 9;
18:            if the interest of e' > I_c ∧ ||e' u_i|| < D_c then
19:                Queue.Enqueue(e', ||u_i e'||);
20: Return G_k;
```

Theorem 8. *Let $S = \{u_1, u_2, \cdots, u_m, u_{m+1}, \cdots, u_n\}$ be a sorted list of users in ascending order of their distances to user u_i. Let $G_k(u_i, u_m)$ be the user group with diameter $\overline{u_i\ u_m}$ and $rank_q(G_k(u_i, u_m)) > rank_q(G_k(u_i, u_j))$ where $1 \le j < m$. If $I(G_k(u_i, u_m), q.T) = I(u_i, q.T)$, $rank_q(G_k(u_i, u_m)) > rank_q(G_k(u_i, u_j))$ where $1 \le j \le n$, $j \ne m$.*

Proof. Suppose we can find a group $G_k(u_i, u_j)$ with diameter $\overline{u_i\ u_j}$ where $m < j \le n$, such that $rank_q(G_k(u_i, u_m)) < rank_q(G_k(u_i, u_j))$. Since $D(G_k(u_i, u_m)) = \|u_i\ u_m\| < \|u_i\ u_j\| = D(G_k(u_i, u_j))$ and $I(G_k(u_i, u_m), q.T) = I(u_i, q.T) \ge I(G_k(u_i, u_j), q.T)$, we can derive $rank_q(G_k(u_i, u_m)) > rank_q(G_k(u_i, u_j))$ that contradicts the assumption and thus complete the proof.

Interest Constraint I_c. In function IOAIRGetNextGroup, when retrieving user group $G_k(u_i)$, it uses a distance constraint D_c to prune the search space. Besides D_c, function DOAIRGetNextGroup contains an interest constraint I_c to further prune the search space such that selecting a group $G_k(u_i, \cdot)$ with the maximum ranking score is more efficient. Specifically, if the interest of a user e is lower than I_c, the ranking score of the group with diameter $\overline{u_i\ e}$ is lower than the current found candidate group.

Lemma 4. *Suppse u_m and u_n are the m^{th} and n^{th} nearest neighbors of u_i, where $m < n$. We have, $rank_q(G_k(u_i, u_m)) < rank_q(G_k(u_i, u_n)) \iff I(G_k(u_i, u_n), q.T) > I_c$, where*

$$I_c = \frac{I_{max}(q.T)}{\alpha}(rank_q(G_k(u_i, u_m)) - (1-\alpha)(1 - \frac{\|u_i\ u_n\|}{D_{max}})) \tag{8}$$

The proof is trivial and thus omitted (easily derived based on Eq. 3).

Theorem 9. *Let u_m and u_n be the m^{th} and n^{th} nearest neighbors of u_i, where $m < n$. Let $G_k(u_i, u_m)$ be the current found group with maximum ranking score. If $I(u_n, q.T) < I_c$, we have $rank_q(G_k(u_i, u_m)) > rank_q(G_k(u_i, u_n))$.*

Proof. We prove it by contradiction. Assume $rank_q(G_k(u_i, u_m)) \le rank_q(G_k(u_i, u_n))$. Then we have $I_c > I(u_n, q.T) \ge I(G_k(u_i, u_n), q.T)$ that contradicts Lemma 4.

Interest Constraint I_G. Let $G_k(u_i, u_m)$ be the current found candidate group and $I_G = I(G_k(u_i, u_m), q.T)$. When constructing group $G_k(u_i, u_n)$ where $\|u_i\ u_m\| < \|u_i\ u_n\|$, the users whose interest is lower than I_G can be pruned. In other words, if group $G_k(u_i, u_n)$ is successfully constructed such that $rank_q(G_k(u_i, u_m)) < rank_q(G_k(u_i, u_n))$, the users whose interest is lower than I_G must not belong to group $G_k(u_i, u_n)$. It is used to prune the search space when constructing a group with specific diameter. The correctness is guaranteed by Theorem 10.

Theorem 10. *Let u_m and u_n be the m^{th} and n^{th} nearest neighbors of u_i, where $m < n$. If $rank_q(G_k(u_i, u_m)) < rank_q(G_k(u_i, u_n))$, $\forall u_j(I(u_j, q.T) \le I_G) \Longrightarrow u_j \notin G_k(u_i, u_n)$.*

Proof. Assume $\exists u_j(I(u_j, q.T) \le I_G, u_j \in G_k(u_i, u_n)$. Since $D(G_k(u_i, u_m)) = \|u_i\ u_m\| < \|u_i\ u_n\| = D(G_k(u_i, u_n))$ and $I(G_k(u_i, u_n), q.T) \le I(u_j, q.T) \le I_G = I(G_k(u_i, u_m), q.T)$, we have $rank_q(G_k(u_i, u_m)) \ge rank_q(G_k(u_i, u_n))$ that results in a contradiction.

Given two users u_i and e, function GetCurrentResult (Algorithm 5) is invoked by function DOAIRGetNextGroup (Algorithm 4) to construct group $G_k(u_i, e)$ with the maximum ranking score. Based on Lemma 5, function GetCurrentResult first constructs $C(u_ie)$ to minimize the search space of $G_k(u_i, e)$ (line 1, illustrated by Example 4). If the size of $C(u_ie)$ is less than k-2 or $C(u_ie)$ cannot fully cover T, it returns NULL (lines 2–3), because it is impossible to formulate a group $G_k(u_i, e)$ with less than k-2 users. Otherwise, we use the line segment $\overline{u_ie}$ to partition the search space $C(u_ie)$ into two sets G_L and G_R (lines 4–5). The users from G_L and G_R whose interests are no less than that of u_i and e are put into G_{LU} and G_{RU}, respectively (lines 6–7). If there are no less than k-2 users in G_{LU} or G_{RU}, we randomly select k-2 users from G_{LU} and G_{RU} such that their union with $\{u_i, e\}$ fully covers T, and return them together with $\{u_i,\ e\}$ as the result (lines 8–10, illustrated by Example 5). The correctness is guaranteed by Lemma 6. We then split the users into two sets G_{up} and G_{down} according to $I_{min}\{u_i, e\}$ (here, $I_{min}\{u_i, e\}$ denotes the minimum interest of u_i and e) (lines 11-12). G_{up} represents the user set whose interest is no less than $I_{min}\{u_i, e\}$, and G_{down} represents the user set whose interest is less than $I_{min}\{u_i, e\}$. Afterwards, we iteratively construct $G_k(u_i, e)$ with the users in G_{down} in a decreasing order of group interest (lines 13–20). In other words, we prefer to search the group with high interest, because a higher interest means a higher ranking score for group $G_k(u_i, e)$ when the diameter $\overline{u_ie}$ is known. In each iteration, if there exists k-2 users such that the distance between any pair of users is no more than $\|u_i\ e\|$ and their union with $\{u_i, e\}$ fully covers T (by enumerating all possible groups of size k-2), $G_k \cup \{u_i, e\}$ is returned as the result (lines 15–17). Otherwise, the user p with the highest interest in G_{down} will be moved into G_{up} (lines 18–19). This process is repeated until all users in G_{down} have been checked.

Lemma 5. *Let $C(u_i, \overline{u_iu_j})$ and $C(u_j, \overline{u_iu_j})$ be two circles centered at u_i and u_j with the same raduis $\|u_i\ u_j\|$. The intersection of $C(u_i, \overline{u_iu_j})$ and $C(u_j, \overline{u_iu_j})$ is denoted by $C(u_iu_j)$. We have $\forall u \in G_k(u_i, u_j)(u \in C(u_iu_j))$.*

The proof is obvious and thus omitted.

Example 4. Figure 6 shows the two circles $C(u_1, \overline{u_1u_{11}})$ and $C(u_{11}, \overline{u_1u_{11}})$ centered at u_1 and u_{11} with radius $\|u_1\ u_{11}\|$. The intersection $C(u_1u_{11})$ covers 3 users (e.g., u_3, u_4 and u_9). Group $G_k(u_1, u_{11})$ can be constructed from the users inside $C(u_1u_{11})$. In other words, the search space of $G_k(u_1, u_{11})$ is $C(u_1u_{11})$.

Lemma 6. *If the number of users on one side s of diameter $\overline{u_iu_j}$ inside $C(u_iu_j)$ is no less than $(k-2)$ and their interest is no smaller than the minimum interest of u_i and u_j, group $G_k(u_i, u_j)$ can be constructed by randomly selecting $k-2$ users from s, including u_i and u_j.*

Proof. Since $k-2$ users are selected from s, the distance between all pair users in $G_k(u_i, u_j)$ is no larger than diameter $\|u_i\ u_j\|$. Hence, $D(G_k(u_i, u_j)) = \|u_i\ u_j\|$ and $I(G_k(u_i, u_j), q.T) = \min\{I(u_i, q.T), I(u_j, q.T)\}$. The ranking score of $G_k(u_i, u_j)$ is maximized.

Example 5. Consider the 13 users of IR-tree in Figure 6. Based on Lemma 5, the search space of $G_4(u_1, u_{11})$ is $C(u_1u_{11})$ which contains 2 users $\{u_3, u_4\}$ whose interest is no less than 0.4 (the minimum interest of u_1 and u_{11}). Based on Lemma 6, due to u_3 and u_4 are in one side of the diameter $\overline{u_1u_{11}}$ and 2≥4-2, thus $\{u_3, u_4\} \bigcup \{u_1, u_{11}\}$ is returned as $G_4(u_1, u_{11})$ with maximum ranking score.

Algorithm 5 GetCurrentResult(Group $G_k^{'}$, User u_i, User e, Keywords T)

$C(u_ie) \leftarrow$ Users from $\{u|u \in G_k^{'} \wedge |||u\ \ e|| \leq ||u_i\ \ e||\}$;
if $|C(u_ie)| < k-2$ or $C(u_ie)$ cannot fully cover T **then**
    Return NULL;
$G_L \leftarrow$ Users from $C(u_ie)$ that are above the line segment $\overline{u_ie}$;
$G_R \leftarrow$ Users from $C(u_ie)$ that are below the line segment $\overline{u_ie}$;
$G_{LU} \leftarrow$ Users from G_L whose interest are no less than $I_{min}\{u_i, e\}$;
$G_{RU} \leftarrow$ Users from G_R whose interest are no less than $I_{min}\{u_i, e\}$;
if $|G_{LU}| \geq k-2$ *or* $|G_{RU}| \geq k-2$ **then**
    $G_k \leftarrow$ Select any $k-2$ users from G_{LU} or G_{RU} such that their union with $\{u_i, e\}$ fully covers T ;
    Return $G_k \cup \{u_i, e\}$;
$G_{up} \leftarrow G_{LU} \cup G_{RU}$;
$G_{down} \leftarrow C(u_ie) - G_{up}$;
Queue $\leftarrow$ Sort the users in G_{down} according to their interest in descending order;
repeat
    $G_k \leftarrow k-2$ users from G_{up} such that the distance between any pair of users is no more than $||u_i\ \ e||$ and their union with $\{u_i, e\}$ fully covers T ;
    if G_k is not empty **then**
        Return $G_k \cup \{u_i, e\}$;
    User $p \leftarrow$ Queue.Dequeue();
    $G_{up} \leftarrow G_{up} \cup \{p\}$;
until Queue is empty
Return NULL;

4. Performance Evaluation

This section describes the experiments used to evaluate the algorithms proposed for the processing of SIG queries (i.e., IOAIR and DOAIR). We also consider a baseline algorithm that is similar to algorithm IOAIR, but using the traditional R-tree index without diameter constraint. We introduce the datasets and queries used in Section 4.1 and the experiment setup in Section 4.2. The experimental results are presented in Section 4.3.

4.1. Datasets and Queries

We collect data from two popular location-based social networks in China, i.e., Jiepang[3] and Dianping[4]. Jiepang provides the check-in service for the visitors who may check-in the tourist places they like. Dianping provides the check-in service for the users to share review comments on the POIs such as restaurants they prefer. The properties of these two real datasets are shown in Table 2 below.

Table 2. Dataset Properties

	Jiepang	Dianping
Total # of users	353,493	2,053,214
Total # of spatial objects	244,331	1,466,188
Total # of check-in actions	5,250,466	17,527,599
Total # of unique tags	2,101	153,211
Average # of tags per spatial object	2	23
Average # of tags per user interest	1.3	37

[3]http://www.jiepang.com
[4]http://www.dianping.com

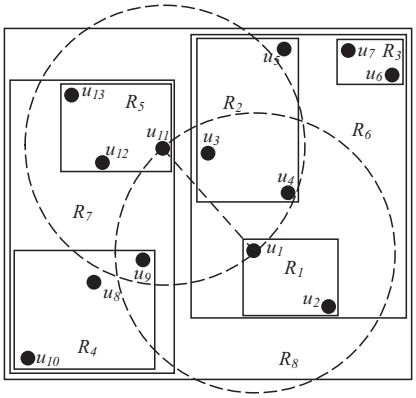


Figure 6. Constructing $G_4(u_1, u_{11})$

We randomly generate two query sets on the two datasets. The query set on Jiepang contains 200 queries, while the query set on Dianping contains 500 queries. Each query contains several keywords and the specified group size. The keywords are randomly generated from the tag set of the dataset. The number of the keywords varies from 1 to 5. The group size k is assigned to values $\{20, 40, 60, 80, 100\}$.

4.2. Setup

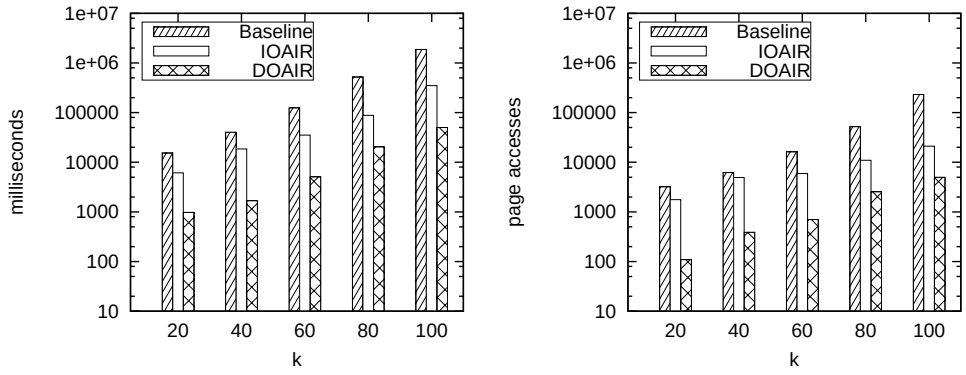
The indexes, including the R-tree, the inverted file, and the IR-tree, used in this paper are disk resident. The page size is set to 4KB. The fanouts of the R-tree and the IR-tree are both set at 100. All the algorithms are implemented in Java programming language. The models of the CPU and RAM are Intel Core 2 Quad Processor 2.4G Hz and 4GB DDR3 memory, respectively. The default values of k, α, and the number of query tags are 50, 0.5, and 1, respectively.

4.3. Experimental Results

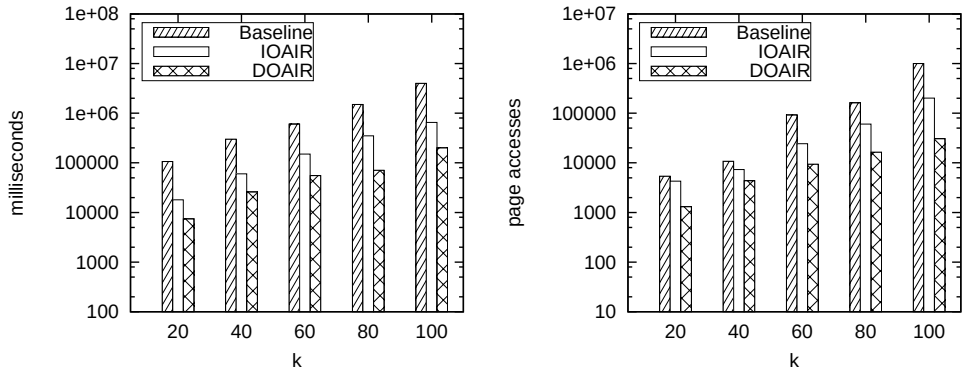
We evaluate the performance of the three algorithms when varying the value of parameter k, α, the number of keywords, and the buffer size. We also test the scalability of the proposed algorithms on the two different datasets. As in many other performance evaluation on query processing, we report the overall performance using the average elapsed time and the average I/O cost.

Varying group size k: In this experiment, we evaluate the performance of our proposed algorithms varying the group size k. Figure 7 shows the average elapsed time and the simulated I/O cost on Jiepang and Dianping datasets. The IOAIR and DOAIR algorithms outperform the baseline approach for all values of k in terms of both metrics, since the IR-tree is able to prune irrelevant leaf nodes whose interest is less than the current interest constraint as early as possible. Notably, the algorithm DOAIR achieves much better performance than IOAIR. This is because Theorems 9 and 10 effectively prune a significant amount of search space and Lemmas 5 and 6 assist to reduce distance computation time for DOAIR.

Varying α**:** Parameter α is used to balance the group interest and the group diameter. Users can adjust α to determine query results bias to interest or diameter. Figure 8 shows the performance of the three algorithms with different values of α. As discussed in Section 3, IOAIR is a slight bias towards finding the k-size maximum interest group with higher group interest, while DOAIR is in favour of searching the maximum interest group with a smaller group diameter. When α is varied from 0.1 to 0.9, the average elapsed time and the average simulated I/O cost of DOAIR on both datasets are increasing, but that of IOAIR is decreasing. Owing to the advantages of DOAIR presented in the section above, DOAIR still has overall better performance than IOAIR algorithm. However, when α is high and the group size k is small, IOAIR achieves better performance than DOAIR (see Figure 9). The reason is two-folded. First, IOAIR has the priority to deal with the group with a high group interest. Thus, with a large α value, IOAIR can find the final results quickly and terminate the query processing early. Second, as discussed above, the performance efficiency of



(a) Varying k on Jiepang



(b) Varying k on Dianping

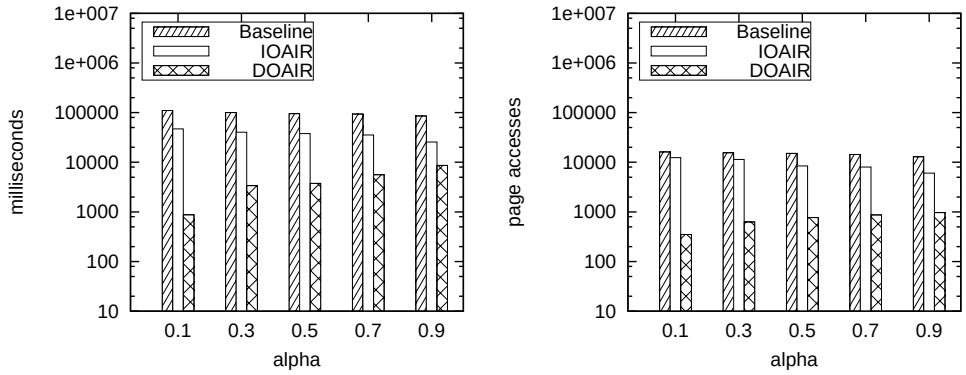
Figure 7. Varying k

DOAIR is much attributed to its strong pruning ability to reduce the distance computation cost (based on Lemmas 5 and 6). When k is small, the distance computation cost is not very high, thereby weakening the pruning effect of DOAIR. Combining these two effects, IOAIR outperforms DOAIR when $\alpha = 0.9$ and $k = 10$.

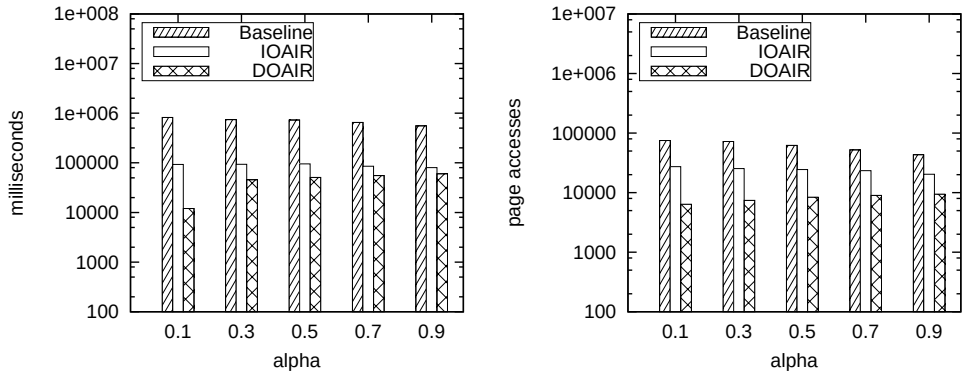
Varying the number of query tags: In this experiment, we evaluate the performance of our proposed algorithms by varying the number of the query tags. In most cases, the number of the query tags is small, thus we only consider the cases where varying the number from 1 to 5. In Figure 10, we can see that again DOAIR demonstrates the best performance in all cases tested. As no user's interest information is integrated into the R-tree, the baseline algorithm cannot prune the irrelevant tree nodes whose interest does not satisfy the interest constraint; thus the baseline algorithm performs the worst in running time and I/O cost. As discussed earlier, DOAIR shows better performance than IOAIR due to its stronger pruning power and reduced distance computation cost.

Varying buffer size: To some extent, the buffer size affects the algorithm performance. The bigger size of buffer setting in the memory, the more disk pages are buffered, and thus the less I/O cost is incurred. In this experiment, we adopt the LRU (Least Recently Used) buffering strategy to cache the disk pages. Figure 11 shows that DOAIR outperforms the other two algorithms in all settings. With increasing the buffer size, as expected the average I/O cost decreases notably. The average elapsed time also keeps the decreasing pattern, but the degree is not that significant. This is because the most time-consuming part is to compute the SIG groups after the irrelevant tree nodes are pruned.

Varying the number of users: With the purpose of testing the scalability of our proposed algorithms, in this set of experiments we vary the number of users in the two testing datasets. As shown in Figure 12, our proposed algorithms DOAIR and IOAIR exhibit good scalability performance. As the number of users grows, the average elapsed time and



(a) Varying α on Jiepang



(b) Varying α on Dianping

Figure 8. Varying α

the average simulated I/O cost of these algorithms on both datasets increase more slowly than the baseline algorithm, resulting in better performance improvement for larger datasets.

5. Related Work

Spatial Query Processing: The research on spatial query processing using R-tree and R*-tree has been extensively studied over the past three decades. The existing research has focused on various types of queries, including *k*-nearest-neighbor queries [12, 13, 14, 15, 16], range queries [17, 18], and closest-pair queries [19, 20, 21].

Roussopoulos *et al*. [12] 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 the *k*-nearest-neighbor search. Katayama *et al*. [13] proposed a new index structure named SR-tree, which integrates bounding spheres and bounding rectangles for high-dimensional nearest neighbor queries. Hjaltason *et al*. [14] proposed an incremental algorithm to efficiently query the nearest neighbor based on the R*-tree. Kolahdouzan and Shahabi [15] presented a novel approach using first-order voronoi diagrams to efficiently evaluate *k*-nearest-neighbor queries in spatial network databases. Papadias *et al*. [16] extended the concept of the nearest neighbor query by considering a group of points which aims to find a set of data points with the smallest sum of distances to all the query points, and proposed various pruning heuristics to efficiently process such group nearest-neighbor queries. Tao *et al*. [18] studied the range search on multidimensional uncertain data. They presented a novel concept of "probabilistically constrained rectangle", which supports effective pruning/validation of nonqualifying/qualifying data. They also developed a new index structure called the U-tree for

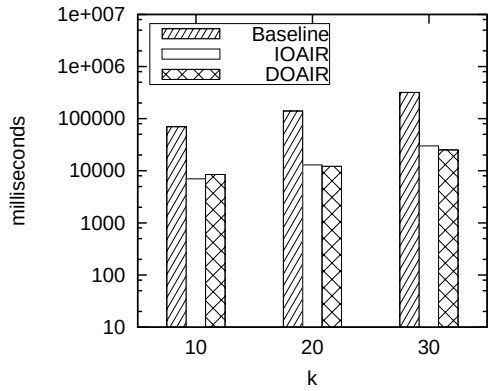


Figure 9. Varying k on Dianping ($\alpha = 0.9$)

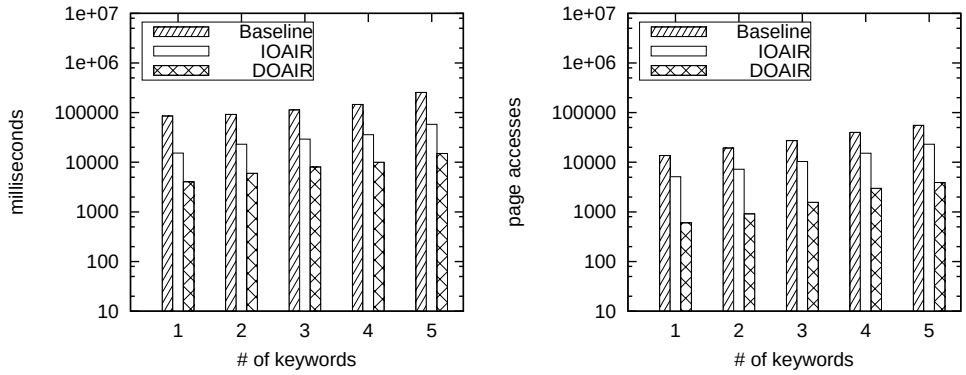
minimizing the query overhead. Pagel *et al*. [17] proposed a probabilistic model for user-defined window queries, and characterized the efficiency of spatial data structures in terms of the expected number of data bucket accesses needed to perform a window query. Corral *et al*. [19] presented nonincremental recursive and iterative branch-and-bound algorithms for k-closest pairs queries. Hjaltason and Samet *et al*. [20] proposed an incremental algorithm based on priority queues for distance join queries. Shin *et al*. [21] suggested adaptive multistage and plane-sweep techniques for *K*-distance join queries and incremental distance join queries.

Our work can be seen as extending the R-tree to handle queries with mixed spatial and keyword information, which retrieves a set of users whose locations are close enough and who have high interest in the query keywords.

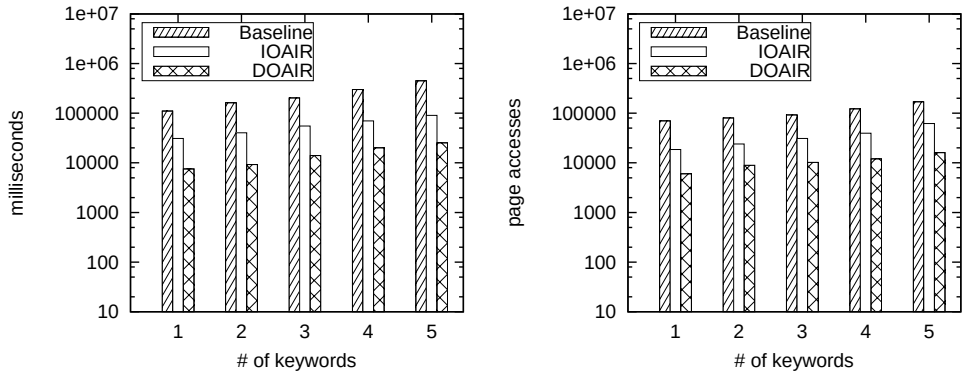
Spatial Keyword Query Processing: Recently, spatial queries have been extended to incorporate text keywords, known as spatial keyword queries in the literature. Zhou *et al*. [22] proposed a hybrid index structure to handle both textual and spatial queries. Cong *et al*. [7] presented a new indexing scheme called IR-tree, which integrates the R-tree and inverted files for location-aware top-*k* object retrieval. Rocha *et al* [23] proposed the top-k spatial keyword queries on road networks where the distance between the query location and the spatial object is the shortest path. An efficient method based on a new hybrid index, cell-keyword conscious B^+-tree, was proposed by Cong *et al*. [24] to process top-k queries on trajectories database. Wu *et al*. [25] studied the efficient processing of continuously moving top-k spatial keyword (MkSK) queries. They proposed two efficient methods for computing safe zones that guarantee correct results at any time with the minimum communication cost. Fellpe *et al*. [26] considered how to find the k-nearest-neighbor of the query location, with each object in the result containing the set of keywords issued in the query. The difference between the top-k query and k-nearest-neighbor query lie in whether the keywords in the query are used as a soft or hard constraint [27]. Fan *et al*. [28] studied the problem of spatio-textual similarity queries on a new kind 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. Lu *et al*. [29] proposed a hybrid index tree called IUR-tree to efficiently process reverse spatial textual k-nearest-neighbor queries which finds the objects that take the query object as one of their *k* most spatial-textual similar objects. Zhang *et al*. [30] studied an *m*-closest-keyword (*m*CK) query that finds a set of partially closest objects covering *m* specified keywords. Cao *et al*. [31] proposed a collective spatial keyword query that retrieves a group of nearby spatial objects to collectively cover the specified keywords. Long *et al*. [27] presented a distance owner-driven approach to solve collective spatial keywords queries.

It is noteworthy that, unlike these previous studies, the proposed SIG query explores the relationship between users' locations and interests in the query keywords and searches the *k*-size maximum interest group in location-based applications.

Group and Team Query Processing: Group and team queries have been studied in the context of social networks [32, 33, 34, 35], including socio-spatial group queries [2], social-temporal queries [4], and expert collaboration queries [5, 6]. In detail, Yang *et al*. [2] proposed a socio-spatial group query (SSGQ) to select a group of nearby attendees with tight social relationships. They designed a new index structure called Social R-tree to integrate the users' social



(a) Varying the number of query tags on Jiepang



(b) Varying the number of query tags on Dianping

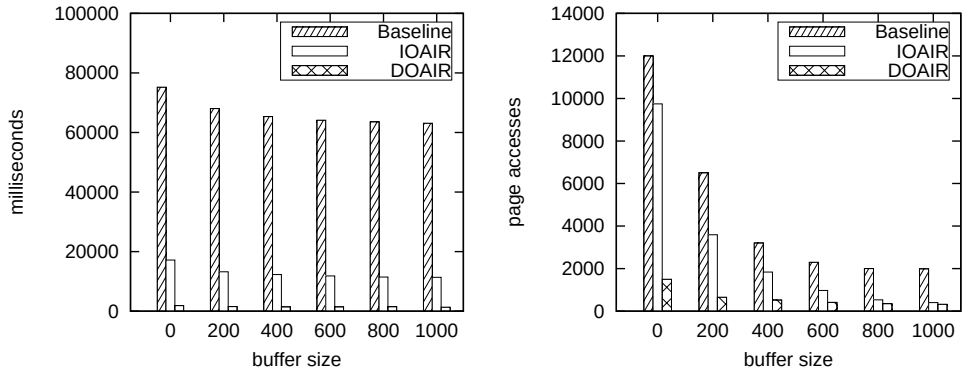
Figure 10. Varying the number of query tags

relationships into an R-tree. In [4], Yang *et al.* proposed a social-temporal group query to find a group of activity attendees with the minimum total social distance to the query issuer. They proposed two efficient algorithms, SGSelect and STGSelect, which include effective pruning techniques and employ the idea of pivot time slots to substantially reduce the query processing time. Lappas *et al.* [5] and Li *et al.* [6] 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.

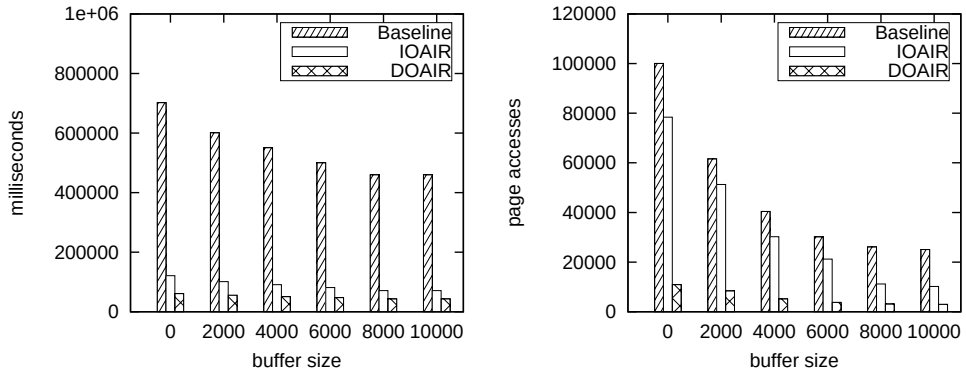
In contrast, our group query concentrates on the common interest of group members rather than their social relationships.

6. Conclusion and Future Work

In this paper, we have presented a new SIG query that considers both the users' spatial locations and their common interest in query keywords. We have proposed a family of efficient algorithms based on the IR-tree, namely IOAIR and DOAIR, for the efficient processing of SIG queries. IOAIR processes SIG queries based on the descending order of interest to search the result group with the minimum diameter. IOAIR integrates the distance constraint into the query optimization to prune search space. In contrast, DOAIR adapts a diameter-oriented strategy to process SIG queries, which takes into account the interest and diameter order simultaneously. Effective pruning techniques have been developed to prune irrelevant search space and accelerate the search speed. The experiments based on two real datasets demonstrate that the DOAIR algorithm achieves the best performance and outperforms the baseline algorithm by orders of magnitude.



(a) Varying buffer size on Jiepang



(b) Varying buffer size on Dianping

Figure 11. Varying Buffer Size

As for future work, we plan to work on the following two extensions. First, we will extend the SIG query to a top-k SIG query that finds the best k user groups in a single query. Second, so far we have not considered the social relationships among users. We will incorporate social relationships as an important criterion in group formation and develop novel query processing techniques.

Acknowledgement

This work is partly supported by GRF211512, GRF210510, FRG1/13-14/042, FRG2/12-13/081, Guangdong NSF S2013010016852 and UIC research grants.

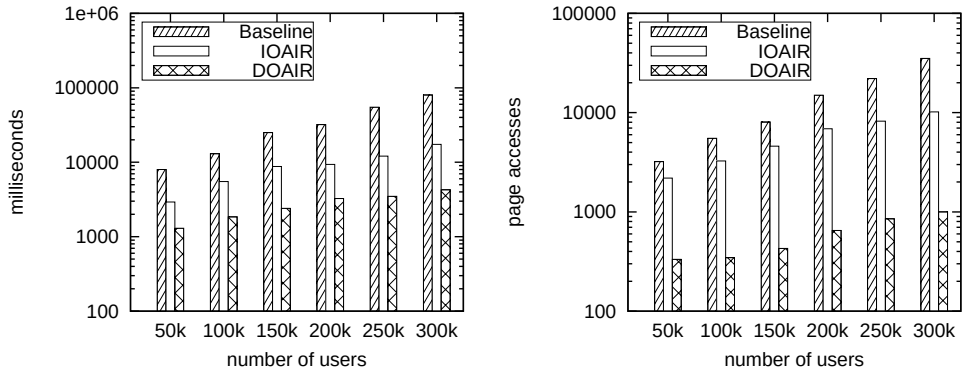
References

[1] P. Ross, Top 11 technologies of the decade, IEEE Spectrum 48 (1) (2011) 27–63. doi:10.1109/MSPEC.2011.5676379.

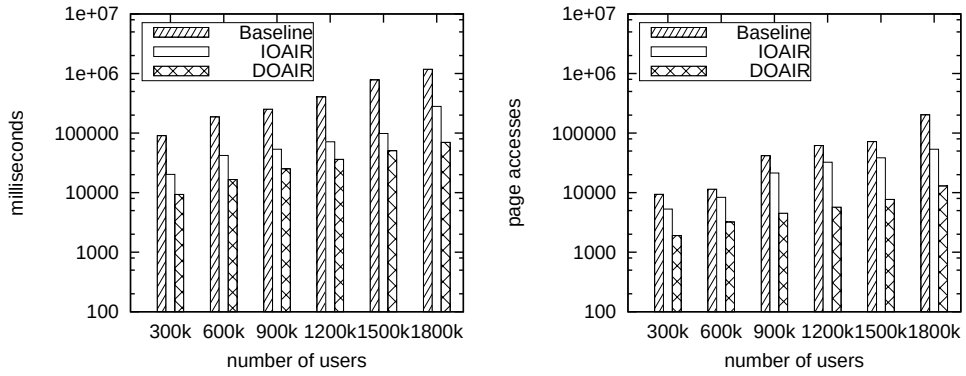
[2] D.-N. Yang, C.-Y. Shen, W.-C. Lee, M.-S. Chen, On socio-spatial group query for location-based social networks, in: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, ACM, New York, NY, USA, 2012, pp. 949–957. doi:10.1145/2339530.2339679.

[3] W. Liu, W. Sun, C. Chen, Y. Huang, Y. Jing, K. Chen, Circle of friend query in geo-social networks, in: Proceedings of the 17th International Conference on Database Systems for Advanced Applications, DASFAA '12, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 126–137.

[4] D.-N. Yang, Y.-L. Chen, W.-C. Lee, M.-S. Chen, On social-temporal group query with acquaintance constraint, in: Proceedings of the VLDB Endowment, PVLDB '11, VLDB Endowment, 2011, pp. 397–408.



(a) Varying the Number of Users on Jiepang



(b) Varying the Number of Users on Dianping

Figure 12. Varying the Number of Users

[5] T. Lappas, K. Liu, 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 '09, ACM, New York, NY, USA, 2009, pp. 467–476. doi:10.1145/1557019.1557074.

[6] C.-T. Li, M.-K. Shan, Team formation for generalized tasks in expertise social networks, in: IEEE Second International Conference on Social Computing, 2010, pp. 9–16. doi:10.1109/SocialCom.2010.12.

[7] G. Cong, C. S. Jensen, D. Wu, Efficient retrieval of the top-k most relevant spatial web objects, in: Proceedings of the VLDB Endowment, PVLDB '09, VLDB Endowment, 2009, pp. 337–348.

[8] B. Martins, M. J. Silva, L. Andrade, Indexing and ranking in geo-ir systems, in: Proceedings of the 2005 Workshop on Geographic Information Retrieval, GIR '05, ACM, New York, NY, USA, 2005, pp. 31–34. doi:10.1145/1096985.1096993.

[9] L. Chen, G. Cong, C. S. Jensen, D. Wu, Spatial keyword query processing: An experimental evaluation, Proc. VLDB Endow. 6 (3) (2013) 217–228.

[10] D. Wu, G. Cong, C. Jensen, A framework for efficient spatial web object retrieval, The VLDB Journal 21 (6) (2012) 797–822. doi:10.1007/s00778-012-0271-0.

[11] J. Zobel, A. Moffat, Inverted files for text search engines, ACM Computing Surveys 38 (2). doi:10.1145/1132956.1132959.

[12] N. Roussopoulos, S. Kelley, F. Vincent, Nearest neighbor queries, in: Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, SIGMOD '95, ACM, New York, NY, USA, 1995, pp. 71–79. doi:10.1145/223784.223794.

[13] N. Katayama, S. Satoh, The sr-tree: An index structure for high-dimensional nearest neighbor queries, in: Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, SIGMOD '97, ACM, New York, NY, USA, 1997, pp. 369–380. doi:10.1145/253260.253347.

[14] G. R. Hjaltason, H. Samet, Distance browsing in spatial databases, ACM Transactions on Database Systems 24 (2) (1999) 265–318. doi:10.1145/320248.320255.

[15] M. Kolahdouzan, C. Shahabi, Voronoi-based k nearest neighbor search for spatial network databases, in: Proceedings of the Thirtieth International Conference on Very Large Data Bases, VLDB '04, VLDB Endowment, 2004, pp. 840–851.

[16] D. Papadias, Q. Shen, Y. Tao, K. Mouratidis, Group nearest neighbor queries, in: Proceedings of the 20th International Conference on Data Engineering, ICDE '04, IEEE Computer Society, Washington, DC, USA, 2004, pp. 301–312.

[17] B.-U. Pagel, H.-W. Six, H. Toben, P. Widmayer, Towards an analysis of range query performance in spatial data structures, in: Proceedings of the Twelfth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS '93, ACM, New York, NY, USA, 1993, pp. 214–221. doi:10.1145/153850.153878.
[18] Y. Tao, X. Xiao, R. Cheng, Range search on multidimensional uncertain data, ACM Transactions on Database Systems 32 (3) (2007) 15–54. doi:10.1145/1272743.1272745.
[19] G. R. Hjaltason, H. Samet, Incremental distance join algorithms for spatial databases, in: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, SIGMOD '98, ACM, New York, NY, USA, 1998, pp. 237–248. doi:10.1145/276304.276326.
[20] A. Corral, Y. Manolopoulos, Y. Theodoridis, M. Vassilakopoulos, Closest pair queries in spatial databases, SIGMOD Record 29 (2) (2000) 189–200. doi:10.1145/335191.335414.
[21] H. Shin, B. Moon, S. Lee, Adaptive multi-stage distance join processing, SIGMOD Record 29 (2) (2000) 343–354. doi:10.1145/335191.335428.
[22] Y. Zhou, X. Xie, C. Wang, Y. Gong, 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 '05, ACM, New York, NY, USA, 2005, pp. 155–162. doi:10.1145/1099554.1099584.
[23] J. a. B. Rocha-Junior, O. Gkorgkas, S. Jonassen, K. Nørvåg, Efficient processing of top-k spatial keyword queries, in: Proceedings of the 12th International Conference on Advances in Spatial and Temporal Databases, SSTD '11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 205–222.
[24] G. Cong, H. Lu, B. C. Ooi, D. Zhang, M. Zhang, Efficient spatial keyword search in trajectory databases, CoRR abs/1205.2880.
[25] D. Wu, M. L. Yiu, C. S. Jensen, G. Cong, Efficient continuously moving top-k spatial keyword query processing, in: Proceedings of the 2011 IEEE 27th International Conference on Data Engineering, ICDE '11, IEEE Computer Society, Washington, DC, USA, 2011, pp. 541–552. doi:10.1109/ICDE.2011.5767861.
[26] I. De Felipe, V. Hristidis, N. Rishe, Keyword search on spatial databases, in: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering, ICDE '08, IEEE Computer Society, Washington, DC, USA, 2008, pp. 656–665. doi:10.1109/ICDE.2008.4497474.
[27] C. Long, R. C.-W. Wong, K. Wang, A. W.-C. Fu, Collective spatial keyword queries: A distance owner-driven approach, in: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, SIGMOD '13, ACM, New York, NY, USA, 2013, pp. 689–700. doi:10.1145/2463676.2465275.
[28] J. Fan, G. Li, L. Zhou, S. Chen, J. Hu, Seal: Spatio-textual similarity search, CoRR abs/1205.6694.
[29] J. Lu, Y. Lu, G. Cong, Reverse spatial and textual k nearest neighbor search, in: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, SIGMOD '11, ACM, New York, NY, USA, 2011, pp. 349–360. doi:10.1145/1989323.1989361.
[30] D. Zhang, Y. M. Chee, A. Mondal, A. Tung, M. Kitsuregawa, Keyword search in spatial databases: Towards searching by document, in: IEEE 25th International Conference on Data Engineering, ICDE '09, 2009, pp. 688–699. doi:10.1109/ICDE.2009.77.
[31] X. Cao, G. Cong, C. S. Jensen, B. C. Ooi, Collective spatial keyword querying, in: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, SIGMOD '11, ACM, New York, NY, USA, 2011, pp. 373–384. doi:10.1145/1989323.1989363.
[32] L. Backstrom, D. Huttenlocher, J. Kleinberg, X. Lan, Group formation in large social networks: Membership, growth, and evolution, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '06, ACM, New York, NY, USA, 2006, pp. 44–54. doi:10.1145/1150402.1150412.
[33] N. Garg, G. Konjevod, R. Ravi, A polylogarithmic approximation algorithm for the group steiner tree problem, in: Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '98, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1998, pp. 253–259.
[34] S.-J. Chen, L. Lin, Modeling team member characteristics for the formation of a multifunctional team in concurrent engineering, IEEE Transactions on Engineering Management 51 (2) (2004) 111–124. doi:10.1109/TEM.2004.826011.
[35] A. Zzkarian, A. Kusiak, Forming teams: an analytical approach, IIE Transactions 31 (1) (1999) 85–97. doi:10.1023/A:1007580823003.