地理社交网络中基于K近邻的兴趣组查询

王佳楠 陈 默 巩树凤 于 戈
(东北大学计算机科学与工程学院 沈阳 110819)

摘 要 为满足地理社交网络平台中用户对附近区域内具有相同兴趣的其他用户的查找需求,提出一种新型空间查询——基于K近邻的兴趣组查询(K-Nearest Neighbor Based Interest Group Query,KNNIG)。与基于距离约束的传统空间K近邻查询不同,KNNIG查询还加入了基于查询关键字的兴趣值约束,并在此基础上提出了D-I评价函数。查询结果为分值最高的用户集合。此外,提出了3种查询处理算法:基本KNNIG查询处理算法(KNNIG-G)、KNNIG查询的优化算法(KNNIG-G*)以及基于网格的距离松弛算法(KNNIG-DR)。在KNNIG-G基础上,KNNIG-G*和KNNIG-DR分别通过空间剪枝和距离松弛策略,在可容忍误差范围内有效地减少了计算开销,提高了查询效率。在真实数据集上进行的实验验证了所提算法的可行性与有效性。

关键词 地理社交网络,K近邻,兴趣组,剪枝,网格
中图法分类号 TP301 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.09.038

K-nearest Neighbor Based Interest Group Query in Geo-social Networks

WANG Jia-nan CHEN Mo GONG Shu-feng YU Ge
(School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China)

Abstract Users on geo-social networks may want to find the other users with the same interests, motivated by which, we proposed a new type of spatial query named K-nearest neighbor based interest group query(KNNIG). Different from traditional spatial K-nearest neighbor queries which only consider the constraint of distance, KNNIG query also considers the users' interest values of the query keyword, based on which, we derived a D-I ranking function. KNNIG query retrieves a user group of size k that maximizes the ranking function. In addition, we proposed three kinds of query processing algorithms, including a basic processing algorithm KNNIG-G, an optimization algorithm KNNIG-G* and a distance relaxation algorithm KNNIG-DR based on grid index. Based on KNNIG-G, KNNIG-G* and KNNIG-DR are developed by a spatial pruning strategy and a distance relaxation strategy respectively, which effectively reduce computational cost and improve the query efficiency. Experimental results on a real dataset demonstrate the feasibility and effectiveness of the proposed algorithms.

Keywords Geo-social networks, K-nearest neighbor, Interest group, Pruning, Grid

1 引言

地理社交网络如Foursquare和Facebook Places等提供的签到服务使用户可以随时获取、上传并与朋友共享自己的位置信息,从而加强了好友在现实生活中的联系[1-3]。这些签到地点可以看作是带有描述用户兴趣的属性标签的位置点。每个位置点l带有个数不等的属性标签,如$l.t_1$,$l.t_2$等。例如,位置点"星巴克"带有标签"食物"、"饮料"、"咖啡"。如果某一用户在"星巴克"签到,则该用户可能对其标签项感兴趣,即通过用户历史签到数据可以提取出用户对不同属性标签的兴趣值[4]。

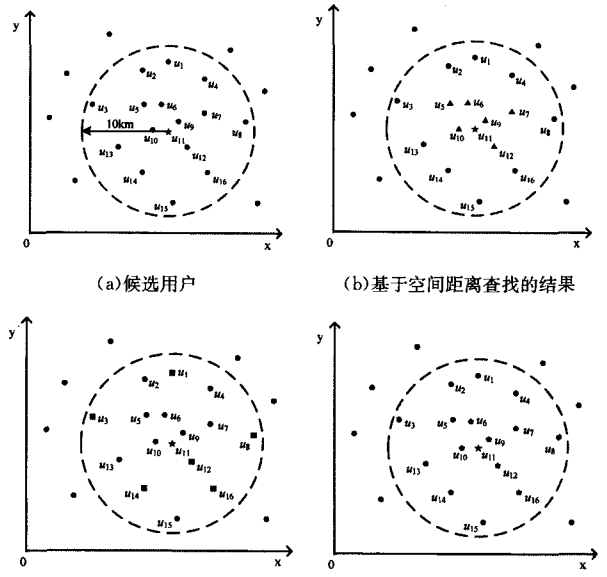
在地理社交网络平台中,用户会希望找到附近区域内具有相同兴趣的其他用户,为满足这类需求,本文提出一种新型查询——基于K近邻的兴趣组查询(K-Nearest Neighbor Based Interest Group Query,KNNIG)。KNNIG查询通过用户历史签到数据提取用户的兴趣值,返回距离其较近并且与其有相同兴趣爱好的k个其他用户组成的集合。

图1给出KNNIG查询的一个示例。若一个喜爱健身的用户u_{11}想要找到其所在健身房附近10km范围内同样爱好健身的6个人来组织健身活动,则该KNNIG查询的查询点为该用户所在健身房(用"★"表示),查询范围半径为10km

到稿日期:2016-08-13 返修日期:2016-11-09 本文受国家自然科学基金资助项目(61402093,61402213),教育部中央高校基本科研业务费(N141604001),辽宁省自然科学基金(2015020018)资助。

王佳楠(1992—),女,硕士,主要研究方向为地理社交网络数据管理,E-mail:wjn920227@163.com;陈 默(1983—),女,博士,讲师,主要研究方向为空间数据库,E-mail:chenmo@mail.neu.edu.cn;巩树凤(1991—),男,硕士,主要研究方向为大数据、云计算;于 戈(1962—),男,教授,博士生导师,主要研究方向为分布式并行数据库、OLAP、图数据管理、数据融合。

（查询范围为图 1 中的虚线圆内），查询兴趣关键字为“健身”，查询用户数 k 为 6。图 1(a)给出了候选用户的分布情况。采用传统空间查询[5]的查询结果如图 1(b)所示（用“▲”表示），由于只考虑空间距离约束，该查询结果为距离查询点最近的 k 个用户。若只考虑用户对兴趣关键字的兴趣值大小，则查询结果如图 1(c)所示（用“■”表示），即返回查询范围内兴趣值最高的 k 个用户。KNNIG 查询综合距离和兴趣值两方面约束给出的最佳查询结果如图 1(d)所示（用“●”表示），与图 1(b)的结果集相比，KNNIG 查询的结果集移除了距离 u_{11} 较近但兴趣值较小的 u_5 和 u_7，加入了距离 u_{11} 稍远但兴趣值比较大的 u_{14} 和 u_{16}。



(a)候选用户　　(b)基于空间距离查找的结果

(c)基于兴趣值查找的结果　　(d)基于空间距离和兴趣值双因素查找的结果

用户	坐标	兴趣值	用户	坐标	兴趣值
u_1	(7.0,9.0)	0.5	u_9	(7.5,5.5)	0.1
u_2	(5.5,8.5)	0.2	u_{10}	(6.0,5.0)	0.2
u_3	(2.5,6.5)	0.6	u_{11}	(7.0,5.0)	0.8
u_4	(9.0,8.0)	0.3	u_{12}	(8.0,4.0)	0.7
u_5	(5.5,6.5)	0.1	u_{13}	(4.0,4.0)	0.3
u_6	(6.5,6.5)	0.4	u_{14}	(5.5,2.5)	0.7
u_7	(9.0,6.0)	0.1	u_{15}	(7.5,1.5)	0.4
u_8	(11.5,5.5)	0.7	u_{16}	(9.5,2.5)	0.9

图 1　KNNIG 查询示例

为实现 KNNIG 查询，本文提出 3 种有效的查询处理算法：KNNIG-G、KNNIG-G* 以及 KNNIG-DR。其中 KNNIG-G* 对 KNNIG-G 进行空间剪枝，有效减少了 I/O 代价；KNNIG-DR 则在 KNNIG-G* 的距离约束方面进行松弛计算，在可容忍误差范围内有效地提高了查询效率。在地理社交网站 Gowalla 的真实数据上进行的实验验证了本文所提算法的有效性。

2　相关工作

传统的空间查询一般只考虑空间距离的约束，如文献[5]中查找距查询点距离最近的 k 个空间点；文献[6]给定两个空间点集合 P 和 Q，查找 P 中的 k 个空间点，使这 k 个点距 Q 中所有点的距离和最小；文献[7-9]致力于找到距离和最短的一组空间点并将查询应用于不同场景。Roussopoulos 等人提出了有效的分支定界 R 树遍历算法来查找查询点的最近邻并将其扩展到 K 近邻查询中[5]。Hjaltason 等人提出基于 R* 树的增量算法来有效查找最近邻[10]。这一类空间查询只关注空间点之间的距离信息，并未涉及其他方面的约束。

在近期的研究工作中，传统的空间查询被扩展为包含熟悉度以及文本关键字等多方面的约束而不仅仅考虑距离方面的约束。如 Zhou 等人[11]提出一种混合索引结构来处理空间文本查询；Cong 等人[12]提出一种新型索引 IR-tree，它结合了 R-tree 和倒排文件来解决基于位置的 top-k 对象检索问题；Chen 等人[13]提出了一种混合索引 IQ-tree，并基于 IQ-tree 提出匹配 BRC(Boolean Range Continuous)查询和地理文本对象的算法。文献[14]致力于在地理文本对象流上为用户维持最佳的 top-k 个对象。以上研究工作都是基于空间文本关键字进行的在线查询处理，而本文涉及到的关键字为兴趣关键字，仅用于在预处理过程中提取用户兴趣值。

与本文较为相关的多约束空间查询有 Yang 等人[15]提出的 SSGQ(Socio-Spatial Group Query)以及 Liu 等人[16]提出的 CoFQ(Circle of Friend Query)。SSGQ 返回到查询点距离和较近并且互相之间满足社交关系要求的 p 个用户。该查询考虑了空间距离和熟悉度两方面因素进行查询，用距离和来进行空间距离约束，并用 R 树对位置信息进行索引以及剪枝，用自定义的熟悉度函数进行社交关系约束。而本文提出的 KNNIG 查询则考虑了空间距离和用户兴趣值两方面的因素，在位置索引方面采用了简单高效的网格索引结构，查找的目标用户与发起者有相同的兴趣。CoFQ 也考虑了距离和熟悉度两方面来查找附近的朋友组。由此上述两种查询与本文提出的 KNNIG 查询的目标用户有所不同。

此外，Jiang 等人[17]提出的 TkLUS(Top-k Local User Search)查找某位置点周围一定范围内发布的微博内容与查询关键词相关的 top-k 个用户；Li 等人[4]提出的 SIG(Spatial-aware Interest Group)查询从某一空间点集合中选取对查询关键字感兴趣且彼此距离较近的 k 个用户。本文应用了 SIG 查询中的兴趣值模型，但与 SIG 查询有较大区别。SIG 查询是在某一空间范围内查找最符合查询约束的对象组，无查询发起点，并采用用户集合的直径距离进行距离约束。该查询主要应用于市场营销中，例如为一家公司选取推销某种产品的合适地点。而 KNNIG 查询的查询范围以用户指定查询点为中心，其使用用户集合中所有用户到查询点的距离和进行距离约束，KNNIG 查询主要用于满足地理社交网络应用中对附近有相同兴趣的用户的查询需求。

综上所述，已有研究工作均与本文研究目标及应用背景有所不同，因此 KNNIG 查询有较高的研究必要性及应用价值。

3　问题定义

设 U 为用户集合，其中每个用户 u ($u\in U$)为一个三元

组，用(id,l,v)表示。其中，id为用户的唯一身份标识，如该用户在某社交网络中的用户名；l为用户在近一段时间（如6个月）内在社交网络平台签到次数最多的地理位置点；v为用户的兴趣向量，记录了该用户对不同兴趣标签的兴趣值[4]。

定义1（兴趣值） D_u表示用户u在近一段时间内所有签到过的位置点集合，D_t表示所有关联兴趣标签t的位置点集合。函数$N(u,l)$统计用户u在位置点l处签到的次数。则u关于t的兴趣值为：

$$I(u,t)=\frac{\sum_{l\in D_u\cap l\in D_t}N(u,l)}{\sum_{l\in D_u}N(u,l)} \tag{1}$$

表1列出用户u_1的兴趣向量示例，兴趣值越大表示用户对该兴趣标签越感兴趣，如用户u_1相比“电影”对“游泳”更感兴趣。

表1 兴趣向量示例

标签	电影	游泳	音乐	健身
兴趣值	0.20	0.30	0.36	0.14

定义2（D-I评价函数） 该函数用来评价候选用户集合$G(|G|=k)$对KNNIG查询q的符合程度，该函数考虑了兴趣值与距离和两方面影响因素，表示如下：

$$Rank(G,q)=\lambda\frac{D_{\min}}{\sum_{u\in G}dist(u,q.l)}+(1-\lambda)\frac{\sum_{u\in G}I(u,q.t)}{I_{\max}} \tag{2}$$

其中，D_{min}为距查询发起点$q.l$最近的k个用户到$q.l$的距离和，$\sum_{u\in G}dist(u,q.l)$为当前检验的用户集合G中用户到$q.l$的距离和，$dist(u,q.l)$为用户u的位置点$u.l$与查询发起点$q.l$之间的欧氏距离，$\sum_{u\in G}I(u,q.t)$为G中所有用户对查询关键字$q.t$的兴趣值的总和，I_{max}为所有候选用户对查询兴趣标签的前k个兴趣值之和，参数$\lambda\in[0,1]$用来调整距离和与兴趣值所占比重。

由于KNNIG查询期望找到距$q.l$距离和较小且对$q.t$的兴趣值较大的用户集合，因此距离和与$Rank$值负相关，兴趣值与$Rank$值正相关。式(2)中$Rank$值随距离和$\sum_{u\in G}dist(u,q.l)$的增加单调递减，随兴趣值$\sum_{u\in G}I(u,q.t)$的增加单调递增，符合查询要求。并且当距$q.l$最近的k个用户恰好是兴趣值最大的k个用户时，$Rank$值为1，即获得最理想的结果集。

定义3（基于K近邻的兴趣组查询，KNNIG） 给定查询q的查询关键字$q.t$（即前文所述的兴趣标签）、查询点$q.l$、查询范围半径$q.r$、查询用户数k，KNNIG查询返回$Rank$值（式(2)）最高的用户集$G_k(|G_k|=k)$。

4 KNNIG查询的查询处理方法

为实现KNNIG查询，在此给出3种查询算法：基于网格的查询处理算法KNNIG-G、基于网格的优化算法KNNIG-G*以及基于网格的距离松弛算法KNNIG-DR。其中KNNIG-G*在KNNIG-G的基础上进行空间剪枝从而提高了查询效率，KNNIG-DR则在KNNIG-G*的距离约束方面进行松弛计算，在可容忍误差范围内有效地提高了查询效率。

方便起见，首先给出本节算法中涉及的符号所代表的含义，如表2所列。

表2 符号释义

符号	含义
$q.r$	查询范围半径
r	查询初始半径
δ	网格单元格边的长度
$dist(l_i,l_j)$	位置点l_i与l_j之间的欧氏距离
$influence\_queue$	存放接下来要检验的用户的队列
G_k	查询中维持的当前最佳结果集
C_q	查询点所在的网格单元
$min\_dist(C,q.l)$	查询点$q.l$到网格C范围内最近的距离
$max\_dist(C,q.l)$	查询点$q.l$到网格C范围内最远的距离
S_C	当前候选用户集
$Rank_{max}$	查询中维持的当前最大$Rank$值

4.1 Baseline算法

为实现KNNIG查询，首先提出较易实现的Baseline算法：1）遍历数据集中的全部用户，找到在查询范围内的用户；2）在这些用户中随机选取k个用户作为初始G_k集合并计算$Rank$值；3）将其余用户依次放入G_k中分别替换G_k中的k个用户并计算$Rank$值，若得到更高的$Rank$值则保留替换结果，直到查询范围内的用户全部替换完毕，返回G_k即为最佳结果集。

4.2 KNNIG-G算法

Baseline算法虽易于实现，但计算代价过大。在此提出基于网格的查询处理算法KNNIG-G。为了对地理社交网络中大量频繁移动的用户位置进行高效索引，满足查询的实时响应需求，本文选用索引速度较快的网格索引。此外，根据网络索引结构的特点，在可容忍误差范围内，给出距离松弛计算等查询优化策略，从而进一步提高查询效率。

设每个网格单元C为$\delta\times\delta$大小的正方形，用$C[i,j]$表示网格单元位于第i列和第j行（$C[0,0]$位于数据空间的左下角）。由地理坐标系变换到网格坐标系后，位置坐标为(x,y)的用户落在网格$C[i,j]$中，其中$i=\lfloor x/\delta\rfloor$，$j=\lfloor y/\delta\rfloor$。

KNNIG-G算法从查询点所在网格C_q开始逐层向外访问网格单元，查询过程中应用CircularTrip算法[18]查找每层网格单元。CircularTrip算法是一种有效的网格单元访问算法，其伪代码如算法1所示。如图2所示，CircularTrip算法可以快速找到所有与以$q.l$为圆心、r为半径的圆相交的网格单元的集合，即图中阴影部分网格。该算法顺时针访问阴影部分所有网格。最先被访问的为与圆弧相交且位置在最左边的网格，即C_{start}。当访问到网格C时，由于第四象限顺时针方向圆弧的走向为右上方向，因此下一个被访问的网格是C的上侧或右侧的相邻网格，即C_u或C_r。由于$min_dist(C_u,q.l)>r$，即C_u与查找圆不相交；$min\_dist(C_r,q.l)<r$，即C_r与查找圆相交，则下一个访问的网格为C_r。每进入一个新的象限需重新判断圆弧的走向（第9—10行）。这样就可以依次访问到该层全部的网格单元。

算法1 CircularTrip($q.l,r$)

输入：查询点q.l，半径r

输出：C={c|r−δ≤min_dist(c,q.l)<r}

1. C←Ø，c←C_{start}=C[i,j](i=$\left\lfloor\frac{q.l.x-r}{\delta}\right\rfloor$，j=$\left\lfloor\frac{q.l.y}{\delta}\right\rfloor$)；

2. D_{cur}←Up；//顺时针改变：Up→Right→Down→Left→Up

3. repeat

4. 将 c 放入 C 中，c′←c 在 D_{cur} 方向上的邻近网格单元；
5. if min_dist(c′, q. l)≥r
6. c′←c 在 D_{cur} 下一个方向上的邻近网格单元；
7. c←c′；
8. if (c. i=C_q. i and c. j=$\left\lfloor\frac{(q.l.y\pm r)}{\delta}\right\rfloor$) or (c. i=$\left\lfloor\frac{(q.l.x\pm r)}{\delta}\right\rfloor$ and c. j=C_q. j) then
9. D_{cur}←D_{cur}顺时针的下一个方向；
10. until c=C_{start}
11. return C

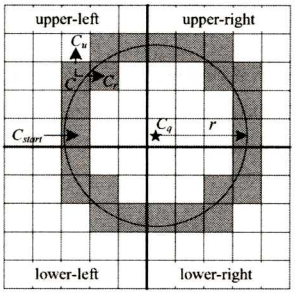


图 2　CircularTrip 算法访问网格的过程

算法 2 给出了 KNNIG-G 算法的伪代码。KNNIG-G 算法从 $q.l$ 开始逐层向外访问查询范围内的网格单元，每次访问与查找圆相交的网格。查找圆的圆心为 $q.l$，初始半径 r 为 $max\_dist(C_q, q.l)$，此后每次 r 值增大 δ，访问过的网格不再重复查找。对每层网格内的候选用户进行如下操作：1)将这些用户加入集合 S_c，将 S_c 中 $dist(u_i, q.l)<r$ 的用户 u_i 从 S_c 中移出并根据 $dist(u_i, q.l)$ 升序入队 $influence\_queue$(第 4－5 行)。2)依次出队，若 $|G_k|<k$，出队用户加入 G_k(第 8－9 行)。若 $|G_k|=k$($|G_k|$ 首次达到 k 时初始化 $Rank_{max}$ 为 $Rank(G_k, q)$)，检验出队用户的兴趣值，若其兴趣值小于或等于 G_k 中所有用户的兴趣值，则直接舍弃(第 11－15 行)；否则将其放入 G_k 中依次替换兴趣值比其小的用户，每次替换后计算 G_k 的评分值 $Rank(G_k, q)$，选出替换操作后得到的最大的 $Rank(G_k, q)$，若 $Rank(G_k, q)>Rank_{max}$，则更新 $Rank_{max}$，保留该次替换结果为 G_k，否则 G_k 不变(第 17－20 行)。3)不断进行出队操作直到 $influence\_queue$ 为空，即本次查找结束。当 $r>q.r+\delta$ 时结束向外查找，返回 G_k 即为最佳结果集(第 21－24 行)。

算法 2　KNNIG-G($U, q.t, q.l, q.r, k$)

输入：候选用户集合 U，查询关键字 q. t，查询点 q. l，查询范围半径 q. r，查询用户数 k

输出：结果集 G_k

1. G_k←Ø，S_c←Ø，r←max_dist(C_q, q. l)，计算 I_{max}；
2. repeat
3. C←CircularTrip(q. l, r)；
4. 将集合 C 中所有网格单元中的用户对象 u_i 放入 S_c；
5. 将 S_c 中 dist(u_i, q. l)<r 的用户 u_i 从 S_c 中移出并根据 dist(u_i, q. l)升序入队 influence_queue；
6. repeat
7. 得到队首元素 u_i；
8. if $|G_k|$<k
9. 将 u_i 放入 G_k 中；
10. else
11. if $|G_k|$首次达到 k
12. 计算 D_{min}，$Rank_{max}$←Rank(G_k, q)；
13. else
14. if I(u_i, q. t)≤G_k 中所有用户的兴趣值
15. 跳转到步骤 7；
16. else
17. 将 u_i 放入 G_k 中依次替换兴趣值比其小的用户并分别计算 Rank 值，选出几次替换中最大的 Rank(G_k, q)；
18. if Rank(G_k, q)>$Rank_{max}$
19. $Rank_{max}$← Rank(G_k, q)并更新 G_k；
20. else G_k 恢复到替换之前的状态；
21. until influence_queue 为空
22. r←r+δ；
23. until r>q. r+δ
24. return G_k；

下面以图 3 中的示例更加直观地阐述 KNNIG-G 算法的处理过程，用户 u_6(“★”表示)发起一个 KNNIG 查询，$q.t$ 为“游泳”，$q.l$ 为(41. 15, 40. 85)，$q.r$=5km，k=3。每个用户 u_i 的数据($u_i.l.x, u_i.l.y; I(u_i, q.t)$)分别为：$u_1$(40. 8, 41. 25; 0. 7)，$u_2$(41. 75, 40. 3; 0. 3)，$u_3$(41. 2, 41. 15; 0. 4)，$u_4$(41. 4, 40. 9; 0. 1)，$u_5$(40. 85, 40. 7; 0. 6)，$u_6$(41. 15, 40. 85; 0. 5)，$u_7$(41. 9, 40. 1; 0. 7)，$u_8$(40. 35, 40. 15; 0. 2)。$\delta$ 设为0. 5km，λ 设为 0. 5。

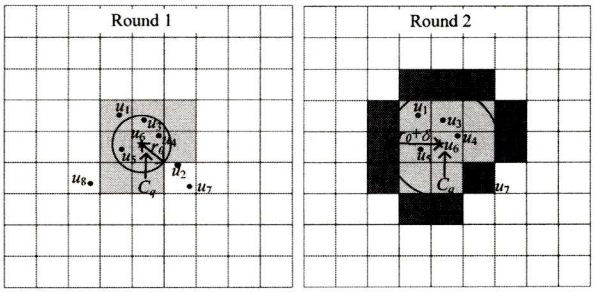


图 3　KNNIG 查询示例

KNNIG-G 算法首先找到第一层网格单元即图中浅色阴影部分，将其中的用户放入 S_c 中，即 $S_c=\{u_1, u_3, u_4, u_5\}$，然后将其中距 $q.l$ 的距离小于查找圆半径的用户 u_3, u_4, u_5 根据距离升序入队 $influence\_queue$，入队顺序为 u_4, u_3, u_5。由于 G_k 此时为空，因此 u_4, u_3, u_5 依次出队放入 G_k 中，此后 $S_c=\{u_1\}$，$G_k=\{u_3, u_4, u_5\}$，初始化 $Rank_{max}=0.7619$，$D_{min}=0.8944$km(初始计算 $I_{max}=2.1$)，第一层查找结束。查找圆的半径增大后找到的第二层网格单元为图中深色阴影部分，将其中用户 u_2, u_7, u_8 放入 S_c 中，即 $S_c=\{u_1, u_2, u_7, u_8\}$。随后将 u_1, u_2 按 u_1, u_2 顺序入队。u_1 出队，此时 $|G_k|=3$，即 $|G_k|=k$，由于 u_1 的兴趣值为 0. 7，大于 G_k 中所有用户的兴趣值，因此用 u_1 依次替换 G_k 中的用户 u_3, u_4, u_5 并分别计算 $Rank$ 值。u_1 替换 u_3 后 $Rank$ 值为 0. 7407，u_1 替换 u_4 后 $Rank$ 值为0. 7947，u_1 替换 u_5 后 $Rank$ 值为 0. 7051，其中最大值为0. 7947>$Rank_{max}$，更新 $Rank_{max}$ 为 0. 7947，更新 G_k 为 $\{u_1, u_3, u_5\}$。u_2 出队，由于 u_2 的兴趣值小于 G_k 中所有用户

的兴趣值，因此直接舍弃 u_2，第二层查找结束。按上述过程不断向外查找直到 $r > q.r+\delta$ 时查询结束，返回 G_k。

4.3 KNNIG-G* 算法

KNNIG-G* 算法对 KNNIG-G 算法进行空间剪枝从而提高查询效率。剪枝策略的核心思想为：若新出队的用户 u_i 不能被替换到 G_k 中或替换进入 G_k 后 *Rank* 值没有增大，则根据定理 1 对 *influence_queue* 中的用户进行剪枝。

定理 1 若出队用户 u_i 不能被替换到 G_k 中或替换进入 G_k 后 *Rank* 值没有增大，则 *influence_queue* 中所有兴趣值小于或等于 $I(u_i,q.t)$ 的用户均可被剪枝。

证明：对于出队用户 u_i，若其不能被替换到 G_k 中或替换进入 G_k 后 *Rank* 值没有增大，则对 $\forall u_j \in G_k$，$Rank(G_k-\{u_j\}\cap\{u_i\},q) \leqslant Rank_{max}$。若对于 *influence_queue* 中用户 u_m，$I(u_m,q.t) \leqslant I(u_i,q.t)$，则对 $\forall u_j \in G_k$，有：

$$Rank(G_k-\{u_j\}\cap\{u_m\},q) = Rank(G_k-\{u_j\}\cap\{u_i\},q) - \lambda \cdot D_{min} \cdot \left(\frac{1}{\sum_{u\in G_k-\{u_j\}\cap\{u_i\}} dist(u,q.l)} - \frac{1}{\sum_{u\in G_k-\{u_j\}\cap\{u_m\}} dist(u,q.l)}\right) - (1-\lambda)\cdot\frac{I(u_i,q.t)-I(u_m,q.t)}{I_{max}} \quad (3)$$

由于 $dist(u_i,q.l) \leqslant dist(u_m,q.l)$，因此 $\sum_{u\in G_k-\{u_j\}\cap\{u_i\}} dist(u,q.l) \leqslant \sum_{u\in G_k-\{u_j\}\cap\{u_m\}} dist(u,q.l)$，$\lambda \cdot D_{min} \cdot \left(\frac{1}{\sum_{u\in G_k-\{u_j\}\cap\{u_i\}} dist(u,q.l)} - \frac{1}{\sum_{u\in G_k-\{u_j\}\cap\{u_m\}} dist(u,q.l)}\right) \geqslant 0$。由于 $I(u_m,q.t) \leqslant I(u_i,q.t)$，因此 $(1-\lambda)\frac{I(u_i,q.t)-I(u_m,q.t)}{I_{max}} \geqslant 0$。因此式(3) $\leqslant Rank(G_k-\{u_j\}\cap\{u_i\},q) \leqslant Rank_{max}$，即 u_m 若被替换进入 G_k 中不会得到更大的 *Rank* 值，即可被剪枝掉。证毕。

KNNIG-G* 算法在查询处理过程中，每当出队用户不能被替换到 G_k 中或替换进入 G_k 后 *Rank* 值没有增大时就进行一次剪枝操作，这样有效地缩小了候选用户空间(算法 3 第 14－22 行)，从而提高了算法的运行效率。算法 3 给出了 KNNIG-G* 算法的伪代码。

算法 3 KNNIG-G* ($U,q.t,q.l,q.r,k$)

输入：候选用户集合 U，查询关键字 q.t，查询点 q.l，查询范围半径 q.r，查询用户数 k

输出：结果集 G_k

```
1. Gk←Ø,Sc←Ø,r← max_dist(Cq,q.l),计算 Imax;
2. repeat
3. C←CircularTrip(q.l,r);
4. 将集合 C 中所有网格单元中的用户对象 ui 放入 Sc;
5. 将 Sc 中 dist(ui,q.l)<r 的用户 ui 从 Sc 中移出并根据 dist(ui,q.l)
   升序入队 influence_queue;
6.   repeat
7.   得到队首元素 ui;
8.   if |Gk|<k
9.     将 ui 放入 Gk 中;
10.  else
11.    if |Gk|首次达到 k
12.      计算 Dmin,Rankmax←Rank(Gk,q);
13.  else
14.    if I(ui,q.t)≤Gk 中所有用户的兴趣值
15.      将 influence_queue 中兴趣值小于或等于 I(ui,q.t)的
         用户移除;
16.      跳转至步骤 7;
17.    else
18.      将 ui 放入 Gk 中依次替换兴趣值比其小的用户并分别
         计算 Rank 值，选出几次替换中最大的 Rank(Gk,q);
19.      if Rank(Gk,q)>Rankmax
20.        Rankmax←Rank(Gk,q)并更新 Gk;
21.      else
22.        Gk 恢复到替换之前的状态并将 influence_queue 中
           兴趣值小于或等于 I(ui,q.t)的用户移除;
23. until influence_queue 为空
24.  r←r+δ;
25. until r>q.r+δ
26. return Gk;
```

4.4 KNNIG-DR 算法

在地理社交网络的实际应用中，用户与“附近对象”之间的距离差值在某一范围内是可接受的。例如，候选用户 u_1，u_2 分别距离查询点 1.5km 和 1.0km，若其距离差值 0.5km 在查询用户可接受范围内，则 u_1 和 u_2 可视为距查询点同等相近。本文将由距离差值导致的结果集的 *Rank* 值差值占精确 *Rank* 值的百分比定义为误差，由此提出基于网格的距离松弛算法——KNNIG-DR 算法。

算法 4 给出了 KNNIG-DR 算法的伪代码。KNNIG-DR 算法在距离方面采用松弛计算策略，在查询过程中将每层找到的网格单元中的用户 u_i 的 $dist(u_i,q.l)$ 均看作是该层查找圆的半径 r(第 4 行)，因此对每层用户只需比较其对 $q.t$ 的兴趣值大小即可。对每层用户 u_i，将 $I(u_i,q.t)$ 最大的 k 个用户按兴趣值降序入队 *influence_queue*(该层其余用户在距离和兴趣值两方面均无优势时可直接剪枝)(第 5 行)。依次出队，对出队用户的检验规则同 KNNIG-G* 算法。一旦出队用户不能被替换到 G_k 中或替换进入 G_k 后 *Rank* 值未增大，由定理 1 可知 *influence_queue* 中用户均可被剪枝，因此清空 *influence_queue*(第 14－15，22 行)，继续进行下一层查找。

算法 4 KNNIG-DR($U,q.t,q.l,q.r,k$)

输入：候选用户集合 U，查询关键字 q.t，查询点 q.l，查询范围半径 q.r，查询用户数 k

输出：结果集 G_k

```
1. Gk←Ø,Sc←Ø,r← max_dist(Cq,q.l),计算 Imax;
2. repeat
3. C←CircularTrip(q.l,r);
4. 将集合 C 中所有网格单元中的用户对象 ui 放入 Sc,dist(ui,q.l)←r;
5. 将 Sc 中兴趣值最高的 k 个用户 ui 从 Sc 中移出并根据 I(ui,q.t)降
   序入队 influence_queue,清空 Sc;
6.   repeat
7.   得到队首元素 ui;
8.   if |Gk|<k
9.     将 ui 放入 Gk 中;
```

10. else
11.　if |G_k|首次达到 k
12.　　计算 D_{min}，$Rank_{max}$←Rank(G_k，q)；
13.　else
14.　　if I(u_i，q. t)≤G_k 中所有用户的兴趣值
15.　　清空 influence_queue；
16.　　跳转至步骤 7；
17.　　else
18.　　将 u_i 放入 G_k 中依次替换兴趣值比其小的用户并分别计算 Rank 值，选出几次替换中最大的 Rank(G_k，q)；
19.　　if Rank(G_k，q)>$Rank_{max}$
20.　　$Rank_{max}$←Rank(G_k，q)并更新 G_k；
21.　　else
22.　　G_k 恢复到替换之前的状态并清空 influence_queue；
23. until influence_queue 为空
24. r←r+δ；
25. until r>q. r+δ
26. return G_k；

以图 3 为例给出 KNNIG-DR 算法的具体处理过程。首先将首次找到的候选用户距 $q.l$ 的距离均设为 r，即 $max\_dist(C_q, q.l)$，并将这些用户放入 S_c 中，即 $S_c=\{u_1, u_3, u_4, u_5\}$。然后将 S_c 中对 $q.t$ 的兴趣值最大的 3 个用户 u_1, u_3, u_5 从 S_c 中取出并按兴趣值降序入队 *influence_queue*，入队后 *influence_queue* 为$\{u_1, u_5, u_3\}$，清空 S_c。u_1, u_5, u_3 依次出队放入 G_k 中，即 $G_k=\{u_1, u_3, u_5\}$，初始化 $Rank_{max}=0.9048$，$D_{min}=1.4849$km(初始计算 $I_{max}=2.1$)，首次查找结束。将查找圆的半径增大 δ 后进行第二层查找，将找到的网格单元中的用户 u_2, u_7, u_8 距 $q.l$ 的距离均设为 r，即 $max\_dist(C_q, q.l)+\delta$，并将它们放入 S_c 中，即 $S_c=\{u_2, u_7, u_8\}$。然后将 u_2, u_7, u_8 从 S_c 中取出并按兴趣值降序入队 *influence_queue*，入队后 *influence_queue* 为$\{u_7, u_2, u_8\}$。u_7 出队，用 u_7 依次替换 G_k 中的用户 u_3, u_5 并分别计算 *Rank* 值，u_7 替换 u_3 后 *Rank* 值为 0.8502，u_7 替换 u_5 后 *Rank* 值为 0.8026，均小于 $Rank_{max}$，因此将 *influence_queue* 中的用户舍弃，清空 *influence_queue*，第二层查找结束。按上述过程不断向外查找直到 $r>q.r+\delta$ 时查询结束，返回 G_k。

4.5　算法复杂度分析

本文提出的 4 种算法的时间复杂度分为距离计算开销和选择计算开销两部分。其中距离计算开销是由计算用户与用户之间以及用户与网格之间的距离引起的，选择计算开销是由计算 *Rank* 值引起的。

Baseline 算法的距离计算开销为 $O(N)$，选择计算开销为 $O(k\cdot(\alpha\cdot N-k))$。其中 α 为查询范围内的用户占数据集中全部候选用户的比例，$\alpha\in(0,1)$。

KNNIG-G 算法的距离计算开销为 $O(\alpha\cdot N+b)$，选择计算开销为 $O(k_1\cdot\beta\cdot N)$。其中 b 为在查询过程中 Circular-Trip 算法所涉及的网格个数；k_1 为 KNNIG-G 中需进行替换测试的用户在 G_k 中的平均替换次数，且 $k_1<k$；β 为 KNNIG-G 中需放入 G_k 进行替换测试的用户的平均比例，且由 KNNIG-G 的替换策略可知 $\beta<\alpha$。由此可见，KNNIG-G 与 Baseline 相比在距离计算和选择计算两方面均节省了较大的开销。

KNNIG-G* 算法的距离计算开销为 $O(\alpha\cdot N+b)$，选择计算开销为 $O(k_2\cdot\gamma\cdot N)$。其中 k_2 为 KNNIG-G* 中需进行替换测试的用户在 G_k 中的平均替换次数，且 $k_2<k$；γ 为 KNNIG-G* 中需放入 G_k 进行替换测试的用户的平均比例，且由定理 1 可知 $\gamma<\beta<\alpha$。由此可见，KNNIG-G* 与 KNNIG-G 相比距离计算开销相同，而由于定理 1 所述的剪枝策略，KNNIG-G* 比 KNNIG-G 节省了一部分选择计算开销。

KNNIG-DR 算法的距离计算开销为 $O(b)$，选择计算开销为 $O(k_3\cdot\theta\cdot N)$。其中 k_3 为 KNNIG-DR 中需进行替换测试的用户在 G_k 中的平均替换次数，且 $k_3<k$；θ 为 KNNIG-DR 中需放入 G_k 进行替换测试的用户的平均比例，θ 与 γ 无比较意义。由此可见，KNNIG-DR 采用距离松弛策略，因此 KNNIG-DR 与其他 3 种算法相比节省了大量的距离计算开销。

5　实验结果与分析

本节将对以上提出的算法进行实验评估。所有实验代码采用 Java 语言编写，实验运行环境采用 Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz 和 8GB 内存。

5.1　数据集及设置

本文采用地理社交网站 Gowalla[1] 在 2009 年 2 月到 2010 年 10 月用户签到的真实数据集，其中标签信息为模拟产生。该数据集含有 107092 个用户的 6442892 条签到数据，属性标签共计 300 个，平均每个位置点带有 2 个属性标签。实验从真实数据集中提取出用户个数从 20000 到 100000 不等的 5 个数据集。每组实验查询 200 次，每次随机选择一个不同的查询点，最终求出平均值作为实验结果。实验参数如表 3 所列。

表 3　实验参数

参数	设置	默认值
k	5,10,15,20,25	10
λ	0.3,0.5	0.5
$q.r$/km	2.5,5.0,7.5,10.0,12.5	10.0
δ/km	0.5,1.0,1.5,2.0,2.5	1.0
$\|U\|$	20000,40000,60000,80000,100000	60000

在实验中评估以下 4 种方法：Baseline 算法、KNNIG-G 算法、KNNIG-G* 算法、KNNIG-DR 算法。

5.2　实验结果与分析

本节分别评估查询用户数 k、查询半径 $q.r$、网格参数 δ、用户数量 $|U|$ 对 4 种算法的查询结果的影响，并对 KNNIG-DR 算法进行误差评估。

5.2.1　查询用户数 k 的影响

图 4 示出了 k 值变化对 4 种算法的运行时间的影响。从实验结果可以看出，随着 k 值的增大，Baseline 的运行时间急剧增大，而应用网格对用户位置进行索引的 KNNIG-G，KNNIG-G* 以及 KNNIG-DR 的运行时间较小且增长缓慢(例如

[1] Gowalla 数据集可从 SNAP 网站下载(http://snap.stanford.edu/data/loc-gowalla.html)

当 $k=10$ 时，4 种算法的运行时间分别为 5.15ms，1.41ms，1.26ms，1.175ms），这是由于这 3 种算法采用网格索引结构减小了大量的距离计算开销，并且有效的距离索引结构使查询性能比较平稳、可扩展性强。其中 KNNIG-G* 在 KNNIG-G 的基础上进行空间剪枝，从而提高了运行效率，在 KNNIG-G* 的基础上使用距离松弛策略的 KNNIG-DR 算法也有效地提高了查询效率。

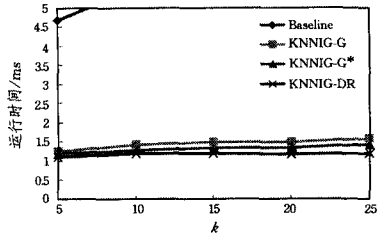


图 4 查询用户数 k 对运行时间的影响

本实验中的查询运行时间较其他工作中算法的运行时间较短的原因为：1）本文实验所用数据集均基于内存读取，不涉及磁盘访问；2）本文算法采用预计算方式对用户的相关属性进行提取，例如兴趣值。

5.2.2 查询半径 $q.r$ 的影响

图 5 给出了查询范围半径 $q.r$ 从 2.5km 增大到 12.5km 时，KNNIG-G，KNNIG-G* 以及 KNNIG-DR 的运行时间的变化。从实验结果可以看出，3 种算法的运行时间均随着 $q.r$ 的增大而增大（例如随着 $q.r$ 从 2.5km 增大到 12.5km，KNNIG-DR 的运行时间分别为 0.315ms，0.545ms，0.790ms，1.175ms，1.560ms），这是由查询范围增大、候选用户增多而引起的计算量增大导致的。其中，KNNIG-G* 的性能始终优于 KNNIG-G 验证了剪枝策略的有效性，KNNIG-DR 的性能持续优于 KNNIG-G 和 KNNIG-G* 证明了距离松弛策略的可行性。

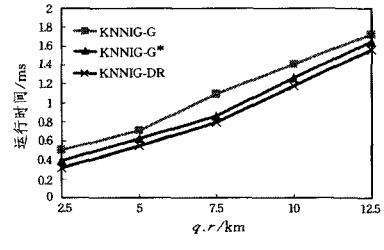


图 5 查询半径 $q.r$ 对运行时间的影响

5.2.3 网格参数 δ 的影响

图 6 给出了网格格边距离 δ 从 0.5km 增大到 2.5km 时 KNNIG-G，KNNIG-G* 以及 KNNIG-DR 的运行时间的变化。本实验中的网格格边距离的步长调整间距为数据空间边长的 5%，即 0.5km。从实验结果可以看出，3 种算法的运行时间整体均随着 δ 的增大而小幅减小，这是由于网格粒度增大导致查询过程中向外查找的网格层数减少引起的。图中折线走势平缓证明了 3 种查询处理算法的稳定性。同时 KNNIG-G* 的剪枝策略的有效性以及 KNNIG-DR 的距离松弛策略的可行性在此也得到了很好的证明。

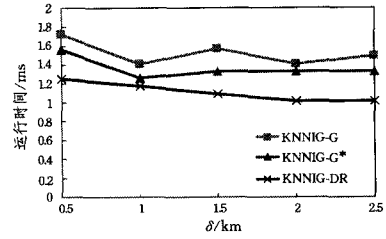


图 6 网格参数 δ 对运行时间的影响

5.2.4 用户数量 $|U|$ 的影响

图 7 示出了用户数量 $|U|$ 从 20000 增大到 100000 对 4 种算法的运行时间的影响。从实验结果可以看出，随着 $|U|$ 值的增大，Baseline 方法的运行时间急剧增加，而应用网格索引结构的 KNNIG-G，KNNIG-G* 以及 KNNIG-DR 的运行时间较短且增长缓慢，再一次证明了网格索引的有效性以及这 3 种查询算法较高的稳定性和可扩展性。

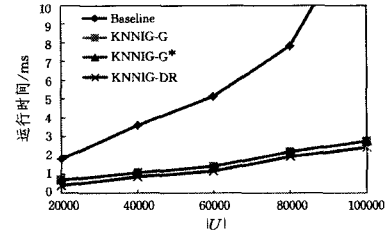


图 7 用户数量 $|U|$ 对运行时间的影响

图 8 和图 9 在用户数量不同的数据集下从距离计算次数和评分值计算次数两方面来对比 KNNIG-G，KNNIG-G* 以及 KNNIG-DR 的计算代价。由图 8 可知，KNNIG-G 和 KNNIG-G* 的距离计算次数相等且随数据集增大而增大，这是由于这两种算法在查询过程中对每层查找到的用户均需计算其到查询点的距离，因此在距离计算方面二者代价相同。而 KNNIG-DR 的距离松弛策略省去了对每层用户进行距离计算，因此距离计算次数为 0。由图 9 可知，KNNIG-G 的评分值计算次数随数据集增大而显著增大，而 KNNIG-G* 和 KNNIG-DR 的评分值计算次数随数据集增大基本保持不变且远小于 KNNIG-G。这是由于 KNNIG-G* 的空间剪枝策略有效地减少了候选用户，从而减少了大量将用户替换进入 G_k 并计算以及比较评分值的操作，KNNIG-DR 对每层用户只获取兴趣值较大的前 k 个进行检验，有效地减少了候选用户从而大幅降低了评分值计算次数。

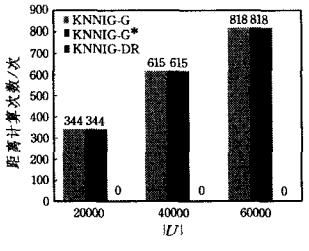


图 8 用户数量 $|U|$ 对距离计算次数的影响

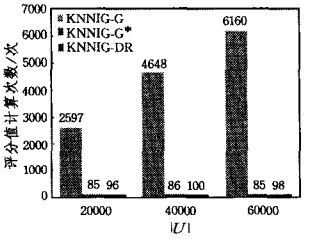


图 9 用户数量 $|U|$ 对评分值计算次数的影响

综上所述，KNNIG-G* 的空间剪枝策略在 KNNIG-G 的

基础上大幅减少了评分值计算次数，从而提高了查询效率。KNNIG-DR 的距离松弛策略使其在距离计算和评分值计算两方面均有明显的性能优势，从而提高了查询效率。

5.2.5　KNNIG-DR 的误差评估

图 10 给出了网格参数 δ 从 0.5km 增大到 2.0km 时 KNNIG-DR 的运行时间以及误差的变化。误差由 KNNIG-G 与 KNNIG-DR 得到的结果集的评分值对比得出。图中曲线为非线性拟合后的实验结果，为了更有效地对 KNNIG-DR 算法进行误差评估，本实验中 λ 设置为 0.3。从实验结果可以看出，随着 δ 的增大，KNNIG-DR 的运行时间整体呈下降趋势，而误差整体呈上升趋势。运行时间减少是由于网格粒度增大导致查询过程中向外查找的网格层数减少，而误差逐渐增大则是由于随着网格粒度增大对距离的松弛程度增大导致的。由图 10 可知，当 δ 在 0.8km 到 1.6km 范围内时，KNNIG-DR 的运行时间较短且误差值较小(在可接受范围内)。由此可见 KNNIG-DR 的距离松弛策略的可行性以及有效性，其在可容忍误差范围内大幅地减小了计算代价，提高了查询效率。

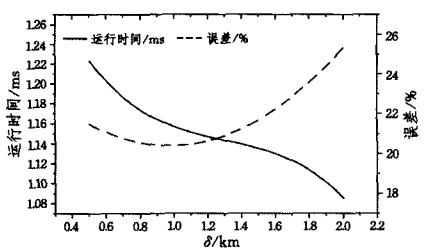


图 10　KNNIG-DR 的误差评估

结束语　本文研究了一种新型空间查询——基于 K 近邻的兴趣组查询(KNNIG)，KNNIG 查询能够解决地理社交网络平台中用户对附近区域内具有相同兴趣的其他用户的查找问题。本文首先提出基于网格的基本 KNNIG 查询处理算法 KNNIG-G，然后在此基础上提出空间剪枝算法 KNNIG-G* 以及距离松弛算法 KNNIG-DR 对查询处理过程进行优化。其中，与 KNNIG-G 相比，KNNIG-G* 通过剪枝策略有效减少了计算代价；与 KNNIG-G* 相比，KNNIG-DR 在距离方面进行松弛计算，从而在可容忍误差范围内大幅地减小了计算代价，提高了查询效率。在真实数据集上进行的实验验证了本文所提算法的可行性与有效性。

未来研究工作将把 KNNIG 查询进一步扩展到动态连续查询的路网环境下，提出相应的高效查询处理方法。

参 考 文 献

[1] HRISTOVA D, WILLIAMS M J, MUSOLESI M, et al. Measuring Urban Social Diversity Using Interconnected Geo-Social Networks[C]// Proceedings of the 25th International Conference on World Wide Web. 2016: 21-30.

[2] GAO H, TANG J, LIU H. Addressing the cold-start problem in location recommendation using geo-social correlations[J]. Data Mining and Knowledge Discovery, 2015, 29(2): 299-323.

[3] ANEJA N, GAMBHIR S. Geo-Social Semantic Profile Matching Algorithm for Dynamic Interests in Ad-hoc Social Network[C]// 2015 IEEE International Conference on Computational Intelligence & Communication Technology (CICT). IEEE, 2015: 354-358.

[4] LI Y F, WU D M, XU J L, et al. Spatial-aware interest group queries in location-based social networks[C]// Proceedings of the 21st ACM International Conference on Information and Knowledge Management. 2012: 2643-2646.

[5] ROUSSOPOULOS N, KELLEY S, Vincent F. Nearest neighbor queries[J]. ACM Sigmod Record. ACM, 1995, 24(2): 71-79.

[6] PAPADIAS D, SHEN Q, TAO Y, et al. Group nearest neighbor queries[C]// 20th International Conference on Data Engineering, 2004. IEEE, 2004: 301-312.

[7] TAO Y, PAPADIAS D, SHEN Q. Continuous nearest neighbor search[C]// Proceedings of the 28th International Conference on Very Large Data Bases. VLDB Endowment, 2002: 287-298.

[8] GAO Y, ZHENG B, LEE W C, et al. Continuous visible nearest neighbor queries[C]// Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology. ACM, 2009: 144-155.

[9] GAO Y, ZHENG B. Continuous obstructed nearest neighbor queries in spatial databases[C]// Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data. ACM, 2009: 577-590.

[10] HJALTASON G R, SAMET H. Distance browsing in spatial databases[J]. ACM Transactions on Database Systems (TODS), 1999, 24(2): 265-318.

[11] ZHOU Y, XIE X, WANG C, et al. Hybrid index structures for location-based Web search[C]// Proceedings of the 14th ACM International Conference on Information and Knowledge Management. ACM, 2005: 155-162.

[12] CONG G, JENSEN C S, WU D. Efficient retrieval of the top-k most relevant spatial web objects[J]. Proceedings of the VLDB Endowment, 2009, 2(1): 337-348.

[13] CHEN L, CONG G, CAO X. An efficient query indexing mechanism for filtering geo-textual data[C]// ACM SIGMOD International Conference on Management of Data. 2013: 749-760.

[14] CHEN L, CONG G, CAO X, et al. Temporal Spatial-Keyword Top-k publish/subscribe[C]// International Conference on Data Engineering. IEEE, 2015: 255-266.

[15] YANG D N, SHEN C Y, LEE W C, et al. On socio-spatial group query for location-based social networks[C]// Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2012: 949-957.

[16] LIU W, SUN W, CHEN C, et al. Circle of friend query in geo-social networks[M]// Database Systems for Advanced Applications. Springer Berlin Heidelberg, 2012: 126-137.

[17] JIANG J, LU H, YANG B, et al. Finding top-k local users in geo-tagged social media data[C]// 2015 IEEE 31st International Conference on Data Engineering (ICDE). IEEE, 2015: 267-278.

[18] CHEEMA M A, YUAN Y, LIN X. Circulartrip: an effective algorithm for continuous kNN queries[M]// Advances in Databases: Concepts, Systems and Applications. Springer Berlin Heidelberg, 2007: 863-869.