# 用户偏好约束的空间关键词范围查询处理方法

郭 帅 刘 亮 秦小麟
(南京航空航天大学计算机科学与技术学院 南京 210016)

**摘 要** 随着基于地理位置的个性化服务的广泛应用,用户偏好约束的空间关键词范围查询成为了研究热点。现有面向空间关键词范围查询的索引没有考虑用户偏好属性,导致剪枝性能和查询效率较低。为了解决该问题,提出了一种支持用户偏好属性、空间位置、关键词协同剪枝的混合索引 BRPQ;并在此基础上,提出了高效的用户偏好约束的空间关键词范围查询处理算法。实验结果表明,相比现有索引,BRPQ 索引的构建时间平均减少了 13%,查询效率平均提升了 20%。

**关键词** 空间文本对象,空间关键词范围查询,用户偏好,混合索引
**中图法分类号** TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.04.031

## Spatial Keyword Range Query with User Preferences Constraint

GUO Shuai LIU Liang QIN Xiao-lin
(College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

**Abstract** With the wide application of location-based personalized service, spatial keyword range query with user preferences constraint becomes a research hotspot. The existing indexes for spatial keyword range query do not take user preferences into account, resulting in poor pruning performance and low query efficiency. In order to solve these problems, a hybrid index called BRPQ(Boolean Range with Preferences Query index) was proposed to support user preferences, spatial location and keywords collaborative pruning. This paper also proposed an efficient query processing algorithm for spatial keywords range query with user preferences constraint. Experimental results show that BRPQ outperforms the existing indexes in terms of building time and query processing efficiency.

**Keywords** Spatio-textual object, Spatial keyword range query, User preferences, Hybrid index

## 1 引言

随着定位技术的广泛应用,大量的文本数据均具有地理标签。例如,微博内容和位置通常密不可分,社交照片共享网站(如 Flickr)每天会产生大量具有描述性标签和地理信息的照片;微博用户会搜索其住所周围的一些话题(如房屋失窃),Flickr 用户会搜索在某一地点拍摄且包含“竹林”关键词的照片。每时每刻都有大量的微博和 Flickr 数据产生,面对如此大规模的地理文本数据,如何高效地处理空间关键词查询成为目前亟需解决的问题[1-2]。

传统的空间关键词查询方法返回离查询位置近、文本与查询关键词相关度高的对象。随着个性化服务的发展,用户更期望查询到符合自己偏好的对象。例如,图 1 展示了大众点评中的一个商户,从商户名称和简介中可以提取出关键词,从商户地址中可以获取位置信息,从口味、环境、服务和营业时间可以获取用户的偏好属性信息。针对这样的对象信息,相对于空间关键词查询,用户更倾向于偏好约束的空间关键词查询。例如,一个用户将在晚上 8 点与合作伙伴吃饭,了解到合作伙伴喜欢川菜,喜欢有音乐的氛围,对口味、环境、服务都比较看重,而且需在 2 个小时内吃完,于是此用户就会查询位于合作伙伴公司 5 公里之内,提供川菜,有音乐氛围,口味、环境、服务评分均大于 8.5,并且在晚上 8 点－10 点营业的商户。本文将诸如口味这类具体数值的属性称为数值点属性,将诸如营业时间这类区间值属性称为数值段属性,并将数值点属性和数值段属性统称为用户偏好属性。


图 1 大众点评网的商户信息

Fig. 1 Public comment network's business information

目前绝大多数地理文本数据查询处理研究[3-5]都没有考

到稿日期:2017-03-30 返修日期:2017-05-15 本文受国家自然科学基金项目(61373015,61402225),江苏省自然科学基金项目(BK20140832),中国博士后基金项目(2013M540447),江苏省博士后基金项目(1301020C)资助。

郭 帅(1993－),男,硕士生,CCF 会员,主要研究方向为空间文本数据查询处理技术,E-mail:marvel_agent@nuaa.edu.cn;刘 亮(1985－),男,博士后,讲师,主要研究方向为传感器网络数据库、时空数据管理等;秦小麟(1953－),男,教授,博士生导师,主要研究方向为空间与时空数据库、分布式数据管理与安全等,E-mail:qinxcs@nuaa.edu.cn(通信作者)。

虑用户偏好的约束。文献[6]率先研究了该问题,通过构建概要树(Synopses Tree)来索引数值点用户的偏好属性,并与 IR-tree 相结合,统计每个 R 树节点下所有对象的用户偏好属性值范围,在查询时判断其与偏好查询区间相交与否来达到剪枝效果;并在此基础上,提出了用户偏好约束的空间关键词 top-k 查询处理框架 LINQ。本文率先研究用户偏好约束的空间关键词范围查询,用 LINQ 框架算法来处理用户偏好约束的空间关键词 top-k 查询,将其适当改造后也可用于处理用户偏好约束的空间关键词范围查询,但在查询处理时存在以下不足。

1)利用用户偏好属性剪枝的效率较低。基于索引查询时,不获取对象的全部信息,仅根据索引来确定对象满足或者不满足用户偏好查询,从而达到过滤对象的目的,这一过程被称为利用用户偏好属性剪枝。LINQ 框架的改造算法使用用户偏好属性进行剪枝时,易受到近邻商户差值的影响。由于现实中相邻(最终会被划归到一个 R 树节点下)的商户相差迥异且评分相差悬殊,使得 R 树底层节点根据概要树(synopses tree)统计出的用户偏好属性值范围很大,在查询时与偏好查询区间相交的数量较少而难以达到高效的剪枝效率,进而影响查询效率。

下面从大众点评商户信息中选取两组具有代表性的数据来说明该现象,其中一组数据是位于南京市新街口中山路 18 号的高档商户区德基广场数据,另一组是位于南京市江宁区将军大道 9 号的中低档商户区乐尚天地数据。从表 1 可以看出,位置邻近的商户的评分迥异。

表 1　近邻商户的评分差异

Table 1　Adjacent business's rating deference display

| 商户名称 | 地理位置 | 口味 | 环境 | 服务 |
|---|---|---|---|---|
| 石打食 | 德基广场 | 7.7 | 7.7 | 7.6 |
| 鱼四季料理 | 德基广场 | 9.2 | 9.3 | 9.1 |
| 川婵亭料理 | 乐尚天地 | 7.1 | 7.2 | 6.9 |
| 阿田大虾火锅 | 乐尚天地 | 9.1 | 9.2 | 9.2 |

2)不能高效的处理数值段用户偏好属性。文献[6]只考虑了数值点用户偏好属性,未考虑数值段用户偏好属性(如营业时间),而用户对营业时间的偏好在此类查询中通常不可或缺,在实际应用中使用频繁。例如,如果不考虑营业时间,查询到的商户对象的营业时间很可能不包含用户需要消费的时间段,查询结果毫无实际价值。

另外,传统的空间关键词范围查询处理技术通常在空间关键词信息剪枝后查询候选结果集的全表信息以进行偏好查询过滤,因此在处理用户偏好约束的空间关键词范围查询时效率不高。

为了解决上述问题,提出了一种支持用户偏好属性、空间位置、关键词协同剪枝的混合索引。主要贡献如下:

1)构建哈希表结构,以提高利用数值点用户偏好属性进行剪枝的效率,从而提高查询效率。

2)提出基于倒排文件的数值段用户偏好属性索引结构,以有效地对数值段属性进行剪枝。

3)提出一种支持数值点用户偏好属性、数值段用户偏好属性、空间位置以及文本协同剪枝的混合索引 BRPQ(Boolean Range with Preferences Query index),以及相应的优化查询算法,高效地解决了用户偏好约束的空间关键词范围查询。

为了评估所提索引及相应算法的性能,从剪枝率、查询时间、索引构建时间等角度进行了大量的实验。结果表明,本文提出的索引及相应算法与 LINQ 框架算法的改造算法相比,在剪枝效率上平均提升了 25%,在查询时间上平均缩短了 20%,并且索引的构建时间更短。

本文第 2 节介绍相关工作;第 3 节阐述相关问题并介绍 BRPQ 索引,给出查询算法,并介绍索引的构建及维护过程;第 4 节进行实验评估;最后总结全文。

## 2　相关工作

近年来,随着基于位置服务的普及,使用给定查询位置和查询关键词搜索相关地理文本对象受到了极大的关注[2,7-9]。现有研究大都对地理文本数据建立混合索引[10-13]。$R^*$-IF 索引[10]利用 R 树索引位置信息,每个 R 树的叶子节点关联一个倒排文件来索引文本信息,查询时根据 R 树找到近邻的叶子节点,再在每个叶子节点下根据文本相似度对对象进行排序。IR-tree[11]则给 R 树的每个节点关联一个倒排文件,利用优先队列查询出文本相似度和距离接近度最大的 $k$ 个对象。bR-tree[12]将 R 树的每个节点关联位图来索引文本信息,查询时根据位图中是否包含所有查询关键词来对 R 树节点进行剪枝,最后根据距离接近度对对象进行排序。文献[13]通过建立 PMR-quad 树和 3 个内存表的数据结构来处理路网中空间关键词的连续 k 近邻查询。但这些研究都只关注空间文本数据的查询处理,在处理用户偏好约束的空间关键词查询时,对于得到的候选结果集,只能获取对象的全部信息,并通过判断其是否满足偏好查询来确定最后结果集,查询效率较低。本文关注用户偏好约束的空间关键词的查询处理。

在用户偏好约束的空间关键词查询处理系统中,对象包括位置信息、关键词信息、数值点属性、数值段属性。相应地,查询语句包括位置、关键词、数值点偏好、数值段偏好。目前,针对该问题的研究仍然处于起步阶段,只有文献[6]进行了相关的研究。文献[6]在文献[11]的空间关键词 top-k 查询处理框架和算法基础上进行了改进,保留了原有的处理空间关键词的 IR-tree 索引,设计了概要树来对数值点用户偏好属性建立索引,并与 IR-tree 相结合来处理用户偏好约束的空间关键词 top-k 查询。在概要树中,统计每个 R 树节点下所有对象的用户偏好属性值的区间范围,在查询时若查询语句中的偏好区间与 R 树节点的统计区间不相交,则对此 R 树节点及其子节点剪枝。由于近邻对象的用户偏好属性值迥异,因此利用用户偏好属性的剪枝率不高,但在 top-k 查询时,由于其只需要得到 $k$ 个结果,利用空间文本相关度的剪枝率高,因此用户偏好属性剪枝率不高的劣势并不会对查询效率产生很大的影响。但对于用户偏好约束的空间关键词范围查询,此劣势极大地影响了查询效率,而且概要树不能处理营业时间这个数值段的用户偏好属性。本文研究了用户偏好约束的空间关键词范围查询处理技术,提出了混合索引 BRPQ 以及相应的查询算法来解决上述问题。

## 3 用户偏好约束的空间关键词范围查询

### 3.1 问题描述

1)地理文本对象。$o=(\psi,\rho,P,S)$,其中,$o.\psi$代表对象关键词集合;$o.\rho$代表对象位置的经纬度;$o.P=\{o.P_1,o.P_2,\cdots,o.P_n\}$,代表对象数值点用户偏好属性值集合,其中$o.P_i$代表数值点属性$P_i$的值;$o.S=(S_1,S_2)\cup(S_3,S_4)\cup\cdots\cup(S_m,S_{m+1})$且$S_i<S_{i+1}$,$1\leqslant i\leqslant m$,代表对象数值段用户的偏好属性值,其中$(S_i,S_{i+1})$代表数值$S_i$到$S_{i+1}$的时间区间。

2)用户偏好约束的空间关键词范围查询。$q=(\psi,r,L,T)$,其中,$q.\psi$代表查询关键词集合;$q.r$代表查询空间范围;$q.L=\{q.L_1,q.L_2,\cdots,q.L_n\}$,代表数值点偏好的最小值集合,其中$q.L_i$代表针对数值点属性$P_i$的偏好的最小值;$q.T=(T_1,T_2)$,代表查询时间段。

3)满足查询的条件。若一个对象满足一个查询,当且仅当下列条件均满足:①$o.\psi\supseteq q.\psi$;②$o.\rho\in q.r$;③$P.P_1>L.L_1$,$P.P_2>L.L_2$,$P.P_n>L.L_n$;④$S\supseteq T$。

具体地,给定一个用户偏好约束的空间关键词范围查询,对于查询处理框架查询出的结果对象,它们的位置在查询区域之内,它们的关键词包含查询关键词,它们的数值点用户偏好属性值大于查询偏好数值,它们的数值段用户偏好属性值包含查询偏好区间。

### 3.2 BRPQ 索引

首先提出了处理数值点和数值段用户偏好属性信息的PSPQ(Point and Segment Preferences Query)索引,然后结合空间关键词索引,提出了一种支持数值点用户偏好属性、数值段用户偏好属性、空间位置以及文本的混合索引 BRPQ(Boolean Range with Preferences Query index)。

#### 3.2.1 PSPQ 索引结构

PSPQ 索引结构由哈希表和倒排文件两部分组成,如图 2 所示。

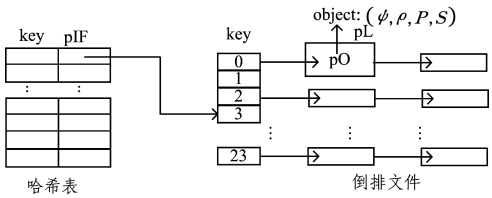



图 2 PSPQ 索引结构

Fig. 2 Index structure of PSPQ

1)哈希表。哈希表用于索引数值点用户偏好属性信息,包含(key,pIF)的键值对。其中,key 为数值点用户偏好属性信息的标识,pIF 为指向倒排文件的指针。

哈希表 key 值的设计:数值点用户偏好属性有多个,且针对各个属性的查询都是范围查询,在保证剪枝率的同时对多个属性进行联合范围查询是设计 key 值的一大挑战。针对这一挑战,将各个属性值区间切分为多个小区间,每个小区间的 key 值被赋为区间最小值,然后将各个属性的小区间 key 进行笛卡尔积,从而得到哈希表的 key 值。该方法有两大优势:①根据 key 值可以快速地对数据进行存取,并能对多个属性进行联合范围查询;②剪枝时可以根据偏好查询快速过滤掉不符合查询范围的哈希表 key 值,各个属性值区间切分得越多,过滤掉的哈希表 key 值就越多,剪枝能力越强。

通过切分属性值区间提取哈希表的 key 值:针对每一个数值点用户偏好属性,对全部对象的属性值进行等份切分,从而得到各个区间。等份的目的在于使每份对应的区间中的对象个数大致相等,数据均匀关联在各个哈希表的 key 值下,从而使剪枝更趋于平稳,不会出现剪枝掉的哈希表 key 值关联的对象数量太多或者太少的情况。

当然,由于用户的数值点偏好属性是数值属性,并且不同偏好属性之间是独立的,因此可以使用 B 树索引数值点偏好属性。但基于 B 树后的代价比 BRPQ 索引的代价更高,因此本文并不采用 B 树索引。下面对此进行详细说明。

基于 B 树后的查询代价会增加:基于 B 树索引多个数值点偏好属性时,需要针对每一个属性建立一个 B 树索引,多个 B 树索引不能与文中处理空间关键词的索引以及处理数值段用户偏好属性的索引相结合。如图 3 所示,采用 B 树后,需要利用空间关键词索引得到满足空间和关键词查询的候选结果集,利用数值段用户偏好属性索引得到满足数值段偏好的候选结果集,在每一个 B 树上求得满足此属性偏好的候选结果集,然后对所有候选结果集求交集。如图 4 所示,基于 BRPQ 索引进行查询时,先对数据集进行空间关键词查询,然后再对所得结果集进行偏好查询。假设数值点用户偏好属性的个数为 $p$,基于 B 树后需要对全部数据集查询 $p+2$ 次,而 BRPQ 索引只需要查询 1 次。在查询过程中,两者的空间关键词查询过程的代价相同,而对于偏好查询,BRPQ 是在空间关键词查询所得结果的基础上再进行偏好查询,而基于 B 树后是针对数据集进行 $p$ 次偏好查询以查找满足各个属性偏好的对象;而且 BRPQ 针对数值点偏好查询时采用哈希技术,时间复杂度为 O(1),而 B 树的时间复杂度为 $O(\log_2 N)$,$N$ 为 B 树的高度,$p$ 个 B 树的时间复杂度为 $O(p\log_2 N)$。综上,基于 B 树后的查询代价明显高于 BRPQ。

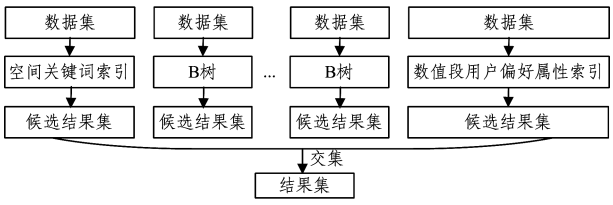



图 3 基于 B 树的查询

Fig. 3 Query based on B tree

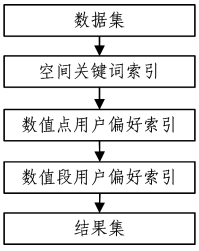



图 4 BRPQ 索引查询

Fig. 4 Query based on BRPQ index

基于 B 树后索引空间增大:基于 B 树索引多个数值点偏好属性时,需要针对每一个属性建立一个 B 树索引。假设数值点用户偏好属性的个数为 $p$,数据集对象的个数为 $n$,不仅需要保存$(p+2)*n$个指向数据的指针,而且 $p$ 个 B 树索引的

占用空间较大。而 BRPQ 只需要增加哈希表的 key 值即可。

基于 B 树后的插入代价增大:基于 B 树索引数值点用户偏好属性后,需要对全部数据集插入 $p+2$ 次;而 BRPQ 只需要插入 1 次。在插入过程中,两者插入空间关键词索引的代价相同;但在插入偏好属性时,BRPQ 采用哈希技术,插入代价为 O(1),基于 B 树索引的最坏插入代价为 $O(\log n)$,插入 $p$ 个 B 树的时间复杂度为 $O(p\log n)$。因此,基于 B 树后的插入代价明显高于 BRPQ。

文献[6]也说明了在用户偏好约束的空间关键词 top-k 查询中不用 B 树索引用户偏好属性的原因:假定只有一个偏好属性,且其偏好查询的对象满足率为 $x$,则先经过 B 树进行偏好查询后再逐一进行空间关键词查询的代价为 $HT+LB*x+n*x$[16],其中,$HT$ 是 $B$ 树的高度,$LB$ 是叶子块的数量,$n$ 是对象数量。当对象数量较大时,代价将较大。

2)倒排文件。倒排文件用于索引数值段用户偏好属性信息,包含(key,pL)的键值对。其中,key 为数值段用户偏好属性信息的标识,pL 为 key 对应的倒排链表的指针,倒排链表中存储对象的标识或者指针 pO,能唯一标识一个对象。

倒排文件 key 值的设计:处理场景中只存在营业时间(如商户的营业时间、旅馆的营业时间等)这一数值段属性,因此 BRPQ 只处理营业时间这一数值段属性。由于营业时间区间内的时间点存在离散特征(例如对于营业时间在 13 点－15 点的商户,只需要标识 13,14,15 这 3 个点即可),将 key 值设置为 0－23。基于倒排文件索引查询时,DAAT 算法[15]在搜索同时出现在多个倒排列表中的某一数据时效率较高。在进行数值段偏好查询时,利用 key 值进行快速、大量剪枝,利用 DAAT 算法[15]进行高效查询。

### 3.2.2　PSPQ 与空间关键词索引结合

PSPQ 索引支持数值点和数值段用户偏好属性的协同剪枝。为了实现用户偏好约束的空间关键词范围查询,需要将 PSPQ 与空间关键词索引相结合。

目前对空间关键词的查询已有大量的研究成果,对空间位置的索引一般利用四叉树或 R 树,BRPQ 采用四叉树,但也可以采用 R 树。考虑到有少量的中间结果,以及用有序关键词树(Ordered Keyword Trie)检索有序关键词时提供了较好的查询性能[14],采用有序关键词树索引文本信息。有序关键词树剪枝出来的路径明确满足关键词查询条件[14]。组合后的索引 BRPQ 结构如图 5 所示。四叉树叶子节点包含一个指向有序关键词树的指针,并且为四叉树叶子节点所关联的对象设置阈值 $\beta$,达到阈值时四叉树叶子结点分裂。为了保证生成的中间结果较少,继续对叶子节点(后文只将关联有序关键词树的节点称为叶子节点)进行划分,直到每个四叉树节点中包含的对象个数少于或等于 $k$ 个。有序关键词树的每条路径包含一个指向 PSPQ 的指针。

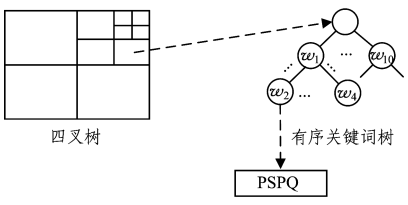



图 5　BRPQ 索引结构

Fig. 5　Index structure of BRPQ

### 3.2.3　索引的构建

本小节将阐述上述 BRPQ 索引的构建过程,如算法 1 所示。

**算法 1**　索引建立 IndexBuilding

输入:所有对象 o=($\psi$,$\rho$,P,S)

```
1. pretreatment(allObject o)
2. for each object o
3.    find current leaf node n containing o and insert//将对象插入包含对象位置的对应四叉树叶子节点中
4.    if n.objectNum>β
5.       split()
6.    end if
7. end for
8. for each leaf node n in quad-tree
9.    split node till each node has less than k object
10.   for each object o in n
11.      find path p in ordered keyword trie associated with n
12.      find pIFin hash table associated with p
13.      find lists ls in inverted file associated with pIF
14.      insert object o's id to ls//在对应时间点的倒排列表中插入对象 id
15.   end for
16. end for
```

算法 1 给出了构建混合索引的伪代码。首先进行预处理,构建哈希表(第 1 行)。然后,将所有对象插入到四叉树节点中,一个四叉树节点中的对象数大于阈值时,分裂成 4 个相等的四叉树节点(第 2－7 行)。由于四叉树节点的分裂将给其他部分带来更新消耗,考虑到建立索引的速度要求,将所有对象存入四叉树中,对象的完整信息也将在每个节点中预存,后续部分建立完成后再删除四叉树节点的预存信息。接着,每一个四叉树叶子节点向下分裂,直到每个节点包含的对象数少于或等于 $k$。最后,对于四叉树叶子节点中的每个对象,根据有序关键词在四叉树叶子节点关联的有序关键词树(ordered keyword trie)中找到或者新建路径,在此路径下根据计算出的 key 值找到指向倒排文件的指针,并根据营业时间将对象 id 存入各个倒排列表中(第 8－15 行)。

算法 1 中预处理构建哈希表 pretreatment 的算法如算法 2 所示。

**算法 2**　预处理构建哈希表 Pretreatment

输入:所有对象 o=($\varphi$,$\rho$,P,S)

```
1. for each numerical point user preference attribute p
2.    for each object o
3.       pDataSortedList←o's p value//将属性值插入排序列表中
4.    end for
5.    divide pDataSortedList to γ equal parts//将排序列表均分为 γ 等份
6.    pKeys←parts's low boundary values//取每等份的小边界值作为等份 key 值
7. end for
```

算法 2 给出了生成哈希表的伪代码。对于每个属性,首先读入所有对象并按属性值排序;然后将属性值等分为 $\gamma$ 段,得到每段对应的低边界值;最后将各个属性得到的值进行笛卡尔积,以形成哈希表的 key 值。

3.2.4 索引维护

在实际运用中，对象的新增和删除或信息的调整（例如大众点评商户的倒闭、注册或者商户信息的调整），涉及到索引的动态维护。对象的新增可以通过创建索引算法来解决，同样地，也可以根据新增时的步骤找到倒排列表中的对象 id 并删除对象。由于四叉树节点中的对象数达到阈值时会花费很大的代价，而大众点评商户的新增不太频繁，因此在新增到 $m$ 倍阈值时再执行分裂操作。对于删除和合并操作，当删除商户所在节点的所有子节点之和小于阈值时，执行合并操作。在现实生活中，商户的倒闭都伴随着商户的注册，合并和分裂操作次数很少，一般只会在大批量一次性删除和新增时出现。对于商户信息的更新，位置和关键词一般不会变动，当数值点属性变动时，将原哈希表 key 对应的倒排列表中的对象 id 转移到更新后的哈希表 key 的倒排列表中；当数值段属性变动时，将倒排列表更新即可。以上论述表明在一般情况下 BRPQ 可以简单快速地进行低代价维护，只有在少数情况下执行分裂和合并操作时需要大范围的调整。

### 3.3 查询算法

本小节在上述混合索引 BRPQ 的基础上，提出了高效的用户偏好约束的空间关键词范围查询处理算法。

给定一个查询，首先找到与查询区域相交的四叉树叶子节点，并继续向下遍历，得到在查询范围内的对象的 id 集合 $S$。其次，对于每一个四叉树叶子节点，查询与它相关的有序关键词树，找到包含所有查询关键词的路径。然后，对于每一条路径，找到满足数值点偏好条件的哈希表 key 值，在 key 值对应的指针指向的倒排列表中找到满足数值段偏好条件的对象，通过判断对象 id 是否在 $S$ 中来确定对象是否为候选结果集对象。最后，针对在不完全满足数值点偏好条件的哈希表 key 值下所得到的对象，判断其数值点属性是否满足查询条件。

算法 3 给出了用户偏好约束的空间关键词范围查询过程的伪代码。首先，用队列来遍历四叉树，找到与查询区域相交的四叉树叶子节点（第 3－8 行），并向下遍历得到满足空间查询范围的对象集合 $S$（第 9 行）。在节点关联的有序关键词树中找到包含所有查询关键词的路径（第 10 行），然后通过计算出的 key 在哈希表中找到对应的指针，$[L_n]$表示包含且大于 $L_n$ 值的等分段的 key 值集合，$[L_1]$，$[L_2]$，…，$[L_n]$表示各个属性的 key 值进行笛卡尔积后得到的 key 值集合（第 14 行）。对于每一个指针，找到其指向的倒排文件中出现在查询段 $T$ 的各个列表中的对象（第 15－18 行）。根据满足位置查询范围的对象集合 $S$ 对候选结果集做筛选处理（第 19 行）。对于在不完全满足数值点偏好条件的哈希表 key 值下所得到的对象，从数据库中提取具体信息并判断所得对象是否是所求结果（第 20－22 行）。经过以上处理，最后得到完全满足查询条件的对象。

**算法 3** 用户偏好约束的空间关键词范围查询算法 SKPQ

输入：用户偏好约束的空间关键词范围查询 q=(ψ,r,L,T)

输出：满足查询条件的对象集合 Result

```
1. Result←Ø
2. n←root
3. queue.push(n)
4. while queue is not empty
5.     n←queue.pop()
6.     if n is not leaf node
7.         queue.push(n.children)
8.         else
9.         idSet←find id in query range traversing n //遍历节点 n 找到满足范围查询的 id 集合
10.        pathsResult←findPaths(n) //搜索包含所有查询关键词的路径
11.        while pathsResult is not empty
12.            path←get next path from pathsResult
13.            hash table←path's associate hash table
14.            pIFs←hash([L1],[L2],…,[Ln]) in hash table//找到满足偏好查询的指向倒排文件的指针
15.            while pIFs is not empty
16.                PIF←get next pIF from pIFs
17.                f←pIF's associate inverted file
18.                tempResult←object o simultaneously existing in T's list
19.                tempResult←idSet contains tempResult's id //根据 idSet 过滤得到的候选结果集
20.                if pIF is not full qualified for all preferences //判断不完全满足偏好的哈希表 key 值下的对象
21.                tempResult←o.P1＞q.L1,…,and o.Pn＞q.Ln and o.Pn＞q.Ln in tempResult
22.                end if
23.                Result.add(o in tempResult)
24.            end while
25.        end while
26.    end if
27. end while
```

### 3.4 复杂度分析

本小节对 BRPQ 索引的建立以及查询算法的时间复杂度进行分析。

首先，分析索引建立前的预处理过程，即构建哈希表的时间复杂度。假设有 $n$ 个对象，每个对象有 $m$ 个数值点用户偏好属性。算法首先根据属性值，通过二分查找法将对象插入到排序队列中，时间复杂度为 $O(n\log n)$。将属性值划分为 $\gamma$ 等份的时间复杂度为 $O(n)$。因为有 $m$ 个属性，所以时间复杂度为 $O(mn\log n)+O(mn)=O(mn\log n)$，又因为 $m$ 为常量，所以构建哈希表的时间复杂度为 $O(n\log n)$。

然后，分析索引建立的时间复杂度。假设有 $n$ 个对象，每个对象平均有 $d$ 个关键词，数据集中不同关键词的个数为 $s$，四叉树的高度为 $h$，哈希表的键值个数为 $p$。每个对象需要首先插入到四叉树的叶子节点中，然后插入到叶子节点关联的有序关键词树中，在最坏情况下其时间复杂度为 $O(nh*sd)$，由于 $d$ 和 $h$ 为常量，并且插入哈希表的复杂度为 $O(1)$，因此建立索引的时间复杂度为 $O(spn)$。由于处理场景中的关键词为标签式关键词，因此 $s$ 不会很大，并且哈希表的键值个数 $p$ 也不会很大。

最后，分析查询算法的时间复杂度。假设四叉树的高度为 $h$，四叉树叶子节点的个数为 $M$，每个对象平均有 $d$ 个关键词，数据集中不同关键词的个数为 $s$，哈希表的键值个数为 $p$。最坏情况下须查找整个区域，即在查找关键词时每一层都查询到了最后一个节点，其时间复杂度为 $O(hMsd)$。由于 $d$ 和 $h$ 为常量，并且查询哈希表的复杂度为 $O(1)$，因此查询算法的总时间复杂度为 $O(spM)$。

## 4　实验结果与分析

### 4.1　实验环境的设置

实验在 ThinkPad T450 上进行，配置如下：CPU 为 Inter(R) Core(TM) i5-5200U CPU @ 2.20GHz，内存为 6GB，硬盘为 500GB，操作系统为 Windows 10。实验中的所有算法均采用 Java 实现，集成开发环境为 IntelliJ IDEA Community Edition 14.0.2。

实验数据使用 yelp 网站提供的 yelp_academic_dataset_busines 数据集。该数据集收集了 yelp 网站上 4 个国家共 11 个城市的 85901 个商户信息。数据集中的每一行记录了一个商户信息，商户信息包括商户标识、地址、营业时间、分类、经纬度等 31 项。实验使用经纬度作为位置信息，将分类作为关键词信息，提取营业时间来作为数值段用户偏好属性，并补充了数值点用户偏好属性信息，其数值范围为[7.0，10.0]，数值随机产生。实验还通过随机采样生成的方法对原有数据集进行了扩充，将其扩充为 500000 个商户。

本实验选取数据集中的 2 个关键词为固定查询关键词，采用一致的查询点和查询范围，查询营业时间的范围固定。若无特殊说明，数值点属性个数为 3，属性值被划分为 10 等份，查询每个数值点用户偏好属性值大于 8.5。数据集采用扩展后的 500000 个商户。

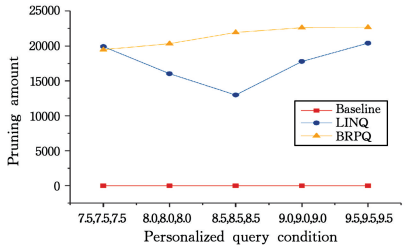
### 4.2　实验结果

处理用户偏好约束的空间关键词查询时，最直观的解决方法是：首先利用空间关键词范围查询处理技术处理数据，然后针对候选结果集，用扫描算法查询满足用户偏好的结果。本实验将此算法作为 Baseline 算法。文献[6]研究了用户偏好约束的空间关键词 top-k 查询，通过构建概要树来索引数值点用户偏好属性，并与传统 IR-tree 相结合，提出了 LINQ 查询处理框架。在统计每个 R 树节点下所有对象的用户偏好属性值范围后，在查询时通过判断与查询区间相交与否来达到利用用户偏好属性剪枝的效果。本文研究用户偏好约束的空间关键词范围查询，LINQ 框架算法在被适当改造后也可用于处理用户偏好约束的空间关键词范围查询。

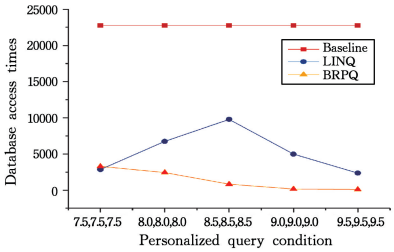
在现有研究中，只有文献[6]进行了用户偏好约束的空间关键词查询的相关研究，因此本实验将所提方法与文献[6]中的改进方法进行对比。本实验从用户偏好属性剪枝量、数据库访问次数、查询时间、索引建立时间等方面对 LINQ 框架算法的改进算法和本文 BRPQ 框架算法进行比较。在本文框架的实现过程中，将对象全局信息存入 MySQL 数据库中，当经过索引后不能明确对象是否属于结果集时，从数据库中读取此对象的全部信息以进行判断，从而造成一次数据库访问。

1）偏好变化时不同索引框架的查询比较

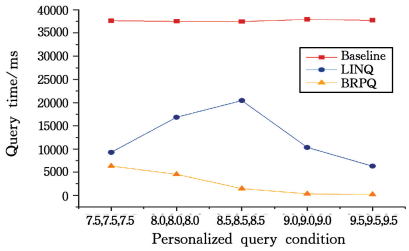
图 6 显示了在不同偏好下，数值点用户的偏好属性个数为 3、划分等份为 10 时，LINQ，BRPQ 以及 Baseline 的剪枝量、数据库访问次数、查询时间的对比。图 6 中的横坐标7.5，7.5，7.5 表示查询的 3 个用户偏好属性值均大于 7.5。在 LINQ 框架中，根据查询时的偏好查询区间与概要树统计出的用户偏好属性值范围是否相交来达到剪枝的效果，越靠近用户偏好属性值范围的边界，同时满足或不满足多个属性的偏好查询的概率越高，剪枝效率越好，因此图 6 中 LINQ 呈现非单调性。由于数值点用户偏好属性值是[7.0，10.0]区间内随机产生的数据，越靠近两边，剪枝效率越高，因此 LINQ 在中间数值 8.5，8.5，8.5 处出现拐点。从图 6(a)中可以看出，BRPQ 利用用户偏好属性的剪枝量远优于 Baseline 和 LINQ。相应地，剪枝量的多少会直接影响数据库的访问次数和查询时间，图 6(b)和图 6(c)分别展示了 BRPQ 相对于 Baseline 和 LINQ 在数据库访问次数和查询时间上的优越性。BRPQ 查询时间的优越性体现在两个方面：①数值点用户偏好属性的剪枝量多；②针对营业时间偏好查询的处理较为高效。LINQ 由于不能处理数值段用户偏好属性，查询时需要从数据库中读取全局信息来判断对象是否属于结果集，造成了额外的查询时间；而 Baseline 算法经过空间关键词查询处理后的大量候选结果集需要从数据库中读取全局信息来判断对象是否满足偏好，因此会消耗更长的时间。




(a)不同偏好时的剪枝量比较




(b)不同偏好时的数据库访问次数比较



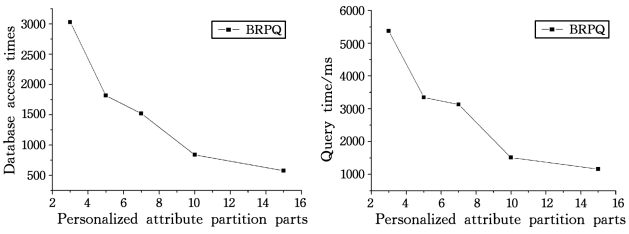

(c)不同偏好时的查询时间比较

图 6　不同偏好时的比较

Fig. 6　Comparison of different preferences

2）用户偏好属性划分的等份数造成的影响

图 7 给出了在其他条件不变且查询每个数值点用户偏好属性值大于 8.5、属性个数为 3 的情况下，只改变用户偏好属性划分的等份数时对数据库访问次数和查询时间的影响。从图 7 中可以看出，数据库访问次数和查询时间都随着用户偏好属性划分等份数的增加而减少，其原因是当划分等份数增加时，会形成更多的哈希表 key 值，相应地，可以过滤掉更多的 key，从而过滤掉更多的对象。但是随着划分等份数的增加，哈希表 key 的数量会大量增长，从而带来较大的空间开销。



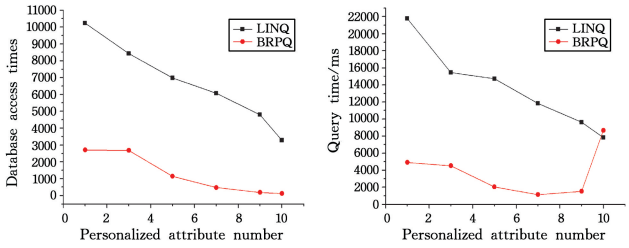

(a)不同用户偏好属性划分等份数下数据库访问次数的变化 (b)不同用户偏好属性划分等份数下查询时间的变化

图 7 不同用户偏好属性划分等份数的影响

Fig. 7 Effect of different attribute division number

3)用户偏好属性个数造成的影响

图 8(a)显示了在其他条件不变且查询每个数值点用户偏好属性值大于 8.0、属性划分等份为 10 的情况下,只改变用户偏好属性个数时对数据库访问次数的影响。从图 8 中可以看出,数据库访问次数随着用户偏好属性个数的增加而减少,其原因是当用户偏好属性个数增加时,同时满足各个用户偏好属性的查询对象减少,从而可以过滤掉更多的对象。图 8(b)显示了在其他条件不变的情况下,只改变用户偏好属性个数时对查询时间的影响。从图 8(b)中可以看出,BRPQ 的查询时间随着用户偏好属性个数的增加先减少后增加,而且减少速率比图 8(a)缓慢得多,其原因是当用户偏好属性增加时,哈希表 key 的数量会大量增加,读取各个哈希表的时间增加。LINQ 查询时间随着用户偏好属性的增加而减少,其原因是 LINQ 随着用户偏好属性的增加,不会产生大量额外的时间消耗。




(a)不同用户偏好属性个数下数据库访问次数的变化 (b)不同用户偏好属性个数下查询时间的变化

图 8 用户偏好属性个数的影响

Fig. 8 Effect of different attribute number

4)数据集大小对索引构建时间的影响

图 9 显示了数据集大小对索引构建时间的影响结果。实验中用对象的个数来表示数据集的大小。

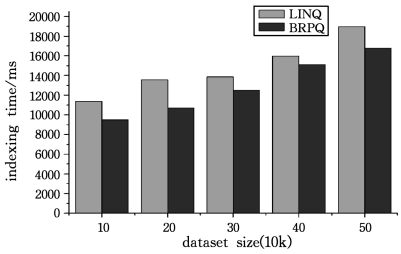


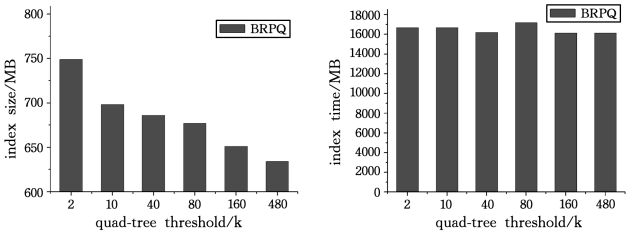

图 9 数据集大小对索引构建时间的影响

Fig. 9 Effect of dataset size on constructing index time

从图 9 中可以看出,BRPQ 的索引构建时间小于 LINQ 的索引构建时间,其原因是概要树的构建复杂,其时间消耗大于 BRPQ 中哈希表和倒排文件结构的构建时间。

5)四叉树阈值对索引大小和索引构建时间的影响

图 10(a)显示了四叉树阈值变化对索引大小的影响。从图 10 中可以看出,BRPQ 的索引大小随着四叉树阈值的增大而减小,其原因是四叉树阈值的增大会导致四叉树叶子节点减少,而每个四叉树叶子节点会关联后续部分,导致索引减小。图 10(b)显示了四叉树阈值变化对索引构建时间的影响。由于四叉树阈值变化不会消耗额外的索引构建时间,因此 BRPQ 索引的构建时间基本不会随着四叉树阈值的变化而变化。




(a)不同四叉树阈值下的索引大小 (b)不同四叉树阈值下的索引构建时间

图 10 四叉树阈值对索引大小和索引构建时间的影响

Fig. 10 Effect of quad-tree threshold on index size and index constructing time

**结束语** 空间关键词范围查询成为了近年来的研究热点,但用户偏好约束的空间关键词范围查询的研究刚开始。针对该问题,提出了一种支持用户偏好属性、空间位置、关键词协同剪枝的混合索引;基于该索引,提出了一种高效的用户偏好约束的空间关键词范围查询处理算法,并将其与现有算法在剪枝率、查询时间、索引建立时间等方面进行了对比。实验结果表明,本文提出的索引及相应算法具有索引建立时间短、利用用户偏好属性剪枝的效率高、查询效率高等优势。下一步计划将本文提出的索引拓展应用于用户偏好约束的空间关键词连续空间范围和 KNN 等查询中。

## 参 考 文 献

[1] ZHOU A Y, YANG B, JIN C Q, et al. Location-Based Services: Architecture and Progress[J]. Chinese Journal of Computers, 2011, 34(7): 1155-1171. (in Chinese)
周傲英,杨彬,金澈清,等. 基于位置的服务:架构与进展[J]. 计算机学报,2011,34(7):1155-1171.

[2] CAO X, CHEN L, CONG G, et al. Spatial Keyword Querying [C]// International Conference on Conceptual Modeling. Springer-Verlag, 2012: 16-29.

[3] 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.

[4] WANG X, ZHANG Y, ZHANG W, et al. AP-Tree: Efficiently support continuous spatial-keyword queries over stream[C]// 2015 IEEE 31st International Conference on Data Engineering. 2015: 1107-1118.

[5] 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.

[6] LIU X, CHEN L, WAN C. LINQ: A Framework for Location-Aware Indexing and Query Processing[J]. IEEE Transactions on Knowledge & Data Engineering, 2015, 27(5): 1288-1300.

[7] CHEN L，CONG G，JENSEN C S，et al. Spatial keyword query processing：an experimental evaluation[J]. Proceedings of the VLDB Endowment，2013，6(3)：217-228.

[8] DE FELIPE I，HRISTIDIS V，RISHE N. Keyword Search on Spatial Databases[C]// International Conference on Data Engineering. IEEE Computer Society，2008：656-665.

[9] ZHANG D，TAN K L，TUNG A K H. Scalable top-k spatial keyword search[C]// International Conference on Extending Database Technology. ACM，2013：359-370.

[10] ZHOU Y H，XIE X，WANG C，et al. Hybrid index structures for location-based Web search[C]// DBLP. 2005：155-162.

[11] 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.

[12] ZHANG D X，CHEE Y M，MONDAL A，et al. Keyword search in spatial databases：Towards searching by document[C]// International Conference on Data Engineering. IEEE，2009：688-699.

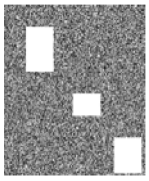
[13] LI Y H，HUANG Q，JIANG H，et al. Research on Processing Continuous Spatial Keyword Range Queries in Road Networks [J]. Computer Science，2014，41(7)：232-235. (in Chinese)
李艳红，黄群，蒋宏，等. 路网中空间关键字连续范围查询算法研究[J]. 计算机科学，2014，41(7)：232-235.

[14] HMEDEH Z，KOURDOUNAKIS H，CHRISTOPHIDES V，et al. Subscription indexes for web syndication systems[C]// International Conference on Extending Database Technology. ACM，2012：312-323.

[15] MANNING C D，RAGHAVAN P，SCHÜTZE H. An Introduction to Information Retrieval[J]. Journal of the American Society for Information Science & Technology，2008，43(3)：824-825.

[16] SILBERSCHATZ A，KORTH H F，SUDARSHAN S. Database System Concepts[M]. New York，USA：McGraw-Hill，2006.

(上接第 181 页)



(a)加密后的图像经部分裁剪后的图像

(b)解密图像

(c)对(b)进行中值滤波后的图像

图 5　针对改进算法的密文抗裁剪测试攻击的测试

Fig. 5　Anti-cutoff test attack test in improved algorithm

**结束语**　本文对基于感知器模型的混沌图像加密算法进行了安全性分析，结果发现该算法不能抵抗选择明文的攻击。通过选择明文攻击，得出了该密码算法的等效密钥，从而破解出了目标明文。同时，指出了该算法存在的另外两个安全缺陷：1)密文对明文的变化不敏感；2)算法不能抵抗剪切攻击。基于前面的分析，对原算法进行了改进，以提高其安全性能。改进措施主要包括：1)增加置乱操作，使算法进一步抵抗剪切、噪声污染及压缩的攻击；2)将原算法步骤 3)中式(5)的 $m$ 值用前一步加密得到的密文像素值代替，使得密钥流与密文图像有关，从而达到“一次一密”的加密效果；3)为进一步提高系统的安全性并增加算法的复杂度，进行了必要的扩散、混淆操作。

## 参 考 文 献

[1] SHANNON C E. Communication theory of secrecy system[J]. Bell System Technical Journal，1949，28(11)：656-715.

[2] SCHIENER B. Applied cryptography：protocols，algorithms and source code in C [M]. New York：John Wiley and Sons，1996：30-40.

[3] BAPTISTA M S. Cryptography with chaos[J]. Physics Letters A，1998，240(1/2)：50-54.

[4] LIU H J，WANG X Y. Color image encryption based on one-time keys and robust chaotic maps[J]. Computers and Mathematics with Applications，2010，59(10)：3320-3327.

[5] WANG X Y，TENG L，QIN X. A novel colour image encryption algorithm based on chaos[J]. Signal Processing，2012，92(4)：1101-1108.

[6] HAN F Y，ZHU C X. New permutation-substitution image encryption scheme based on chaos[J]. Journal of Wuhan University(Natural Science Edition)，2014，60(5)：447-452. (in Chinese)
韩凤英，朱从旭. 新型置换和替代结构的图像混沌加密算法[J]. 武汉大学学报(理学版)，2014，60(5)：447-452.

[7] LIU Q，LI P Y，ZHANG M C，et al. Image encryption algorithm based on chaos system having markov portion[J]. Journal of Electronics & Information Technology，2014，36(6)：1271-1277. (in Chinese)
刘泉，李佩玥，章明朝，等. 基于可 Markov 分割混沌系 统的图像加密算法[J]. 电子与信息学报，2014，36(6)：1271-1277.

[8] LIU H J，WANG X Y. Color image encryption using spatial bit-level permutation and high-dimension chaotic system[J]. Optics Communications，2011，284(16/17)：3895-3903.

[9] LIU H J，WANG X Y. Image encryption using DNA complementary rule and chaotic maps[J]. Applied Soft Computing，2012，12(5)：1457-1466.

[10] ANCHAL J，NAVIN R. A robust image encryption algorithm resistant to attacks using DNA and chaotic logistic maps[J]. Multimedia Tools Applications，2015，29(1)：1-18.

[11] HUANG X L，YE G D. An image encryption algorithm based on hyper-chaos and DNA sequence[J]. Multimedia Tools Applications，2014，72(1)：57-70.

[12] KUMAR M，IQBAL A，KUMAR P. A new RGB image encryption algorithm based on DNA encoding and elliptic curve Diffie-Hellman cryptography[J]. Signal Processing，2016，125(C)：187-202.

[13] AHMAD J，KHAN M A，HWANG S O，et al. A compression sensing and noise-tolerant image encryption scheme based on chaotic maps and orthogonal matrices[J]. Neural Computing and Applications，2017，28(1)：953-967.

[14] WANG X Y，YANG L，LIU R，et al. A chaotic image encryption algorithm based on perceptron Model[J]. Nonlinear Dynamics，2010，62(3)：615-621.

[15] WILLIAM S，等. 密码编码学与网络安全：原理与实践[M]. 杨明，等译. 北京：电子工业出版社，2001：50-60.

[16] RHOUMA R，BELGHITH S. Cryptanalysis of a new image encryption algorithm based on hyper-chaos [J]. Physics Letters A，2008，372(38)：5973-5978.