软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2017,28(2):310−325 [doi: 10.13328/j.cnki.jos.005050]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

一种针对反向空间偏好 top-*k* 查询的高效处理方法*

李 淼[1], 谷 峪[1], 陈 默[2], 于 戈[1,2]

[1](东北大学 计算机软件与理论研究所,辽宁 沈阳 110819)

[2](东北大学 计算中心,辽宁 沈阳 110819)

通讯作者: 于戈, E-mail: yuge@mail.neu.edu.cn

摘 要: 随着地理位置定位技术的蓬勃发展,基于在线位置服务技术的应用也越来越多.提出一种查询类型——反向空间偏好 top-*k* 查询.类似于传统的反向空间 top-*k* 查询,对于给定的空间查询对象,该查询返回使该对象满足 top-*k* 属性得分的那些用户.但不同的是,该对象的属性不是自身具有的特性,而是通过计算该对象与其他偏好对象之间的空间关系(如距离)而确定.这种查询在市场分析等许多重要领域具有需求,例如,根据查询结果,分析出某个地区中某个设施受欢迎的程度.但是,由于大量空间对象的存在导致对象之间空间关系的计算代价非常高,如何实时地计算出对象的空间属性得分,给查询处理带来很大的挑战.针对该问题提出优化的查询处理算法包括:数据集剪枝、数据集批量处理、基于权重的用户分组等策略.通过理论分析和充分的实验验证,证明了所提出方法的有效性.与普通方法相比,这些方法能够大幅度提高查询处理的执行时间和 I/O 效率.

关键词: top-*k* 查询;反向 top-*k* 查询;四叉树;分组;查询优化

中图法分类号: TP311

中文引用格式: 李淼,谷峪,陈默,于戈.一种针对反向空间偏好 top-*k* 查询的高效处理方法.软件学报,2017,28(2):310−325.http://www.jos.org.cn/1000-9825/5050.htm

英文引用格式: Li M, Gu Y, Chen M, Yu G. Efficient processing method for reverse top-*k* spatial preference queries. Ruan Jian Xue Bao/Journal of Software, 2017,28(2):310−325 (in Chinese). http://www.jos.org.cn/1000-9825/5050.htm

Efficient Processing Method for Reverse top-*k* Spatial Preference Queries

LI Miao[1], GU Yu[1], CHEN Mo[2], YU Ge[1,2]

[1](Institute of Computer Software and Theory, Northeastern University, Shenyang 110819, China)

[2](Computing Center, Northeastern University, Shenyang 110819, China)

Abstract: With the proliferation of geo-positioning techniques, there has been increasing popularity of online location-based services. Specifically, reverse top-k spatial preference queries provide such services to retrieve the users that deem a given database object as one of their top-*k* results. The attributes of the query object are given by the spatial distance from users' preference. However in real world, users not only consider the non-spatial attributes about the objects, but also hope to find the spatial objects based on the qualities of features in their spatial neighborhood. While reverse top-*k* spatial preference queries have significant amount of real-life applications such as market analysis, for example, to predict the popularity of a facility in a region, they face a great challenge to compute the score of the spatial attributes online. This paper presents a processing framework and some optimal techniques including pruning and user preference grouping methods. Theoretical analysis and experimental evaluation demonstrate the efficiency of the proposed algorithms and the improvement on running time and I/O.

Key words: top-*k* query; reverse top-*k* query; quad-tree; grouping; query optimization

* 基金项目: 国家自然科学基金(61272179, 61472071, 61402093); 中央高校基本科研业务费专项资金(N141604001)

Foundation item: National Natural Science Foundation of China (61272179, 61472071, 61402093); Fundamental Research Funds for the Central Universities (N141604001)

收稿时间: 2015-09-10; 修改时间: 2016-01-25, 2016-02-19; 采用时间: 2016-02-23

随着空间数据库管理的不断发展,基于用户角度的 top-k 查询在许多领域中得到了广泛应用[1,2].最近,学术界提出了反向 top-k 查询(reverse top-k query)[3].反向 top-k 查询查找所有的用户,这些用户的 top-k 查询结果包含该查询对象.反向 top-k 查询主要用于评估某些潜在的产品在市场上的影响力,可以了解某个产品能够吸引哪些用户或者该产品是否在不同用户偏好之下有很好的排名.由于反向 top-k 查询能够帮助拥有某个产品的生产者评估该产品对用户的影响,在商业分析中具有重要价值.

反向 top-k 问题首次被定义在文献[3]中.反向 top-k 查询通过查询对象 q 和用户的偏好集合 W 来定义,查询结果是所有满足如下条件的用户:设用户的偏好集合为 W',$W'\subseteq W$, 满足该用户偏好集合 W' 的前 k 个查询结果中包含 q.用户的偏好 $w\in W'$ 都用偏好权重 $w[i]$ 表示,该权重值表示用户对于查询对象的某个属性的偏好程度.与之前的研究类似,本文仍然假设用户偏好权重被标准化在[0,1]区间内,并且同时满足 $\sum_i w[i]=1$.

具体以宾馆在线预订为例.如图 1 所示,该空间区域中有 7 个宾馆和宾馆 4 种不同设施,如:餐馆、商场、剧院、咖啡馆(记为 F_1,F_2,F_3,F_4).对于每个宾馆,给出它与每种设施的最近距离的函数值,作为该宾馆在每个属性上的得分.假设在该区域中存在 3 个用户,分别为 May,Jerry 和 Tom.他们希望宾馆能够与其周围设施的距离越近越好,每个宾馆及周围设施的分布如图 1(a)所示.若每个用户给出他们对于每种设施的偏好权重值(如图 1(b)所示),则根据线性模型能够得到每个宾馆对于每个用户的得分情况(该例中,用户对于宾馆的得分值越高越好,并且假设空间最大距离为 10).从该例中可以得出以下结果:May 的 top-2 宾馆为 o_2 和 o_3 (得分函数为 $\sum w[F_i]\cdot(10-dist(o_i,F_i))$, 其中, $dist(o_i,F_i)$ 是宾馆到不同设施的最短距离),同理,Jerry 的 top-2 宾馆为 o_6 和 o_7 ,Tom 的 top-2 宾馆为 o_2 和 o_5 .因此,宾馆 o_2 的反向 top-2 的结果集是 Marry 和 Tom,而宾馆 o_3 的反向 top-2 结果集为 May.

到目前为止,已知最好的解决反向 top-k 查询的算法是 RTA 算法[3](阈值分析法)和 BBR 法算法(分支定界法)[4].但是,RTA 算法在处理反向 top-k 查询时存在着如下缺点:(1) RTA 需要访问所有的用户.对于每一个用户,在给定的一个缓存区中,如果查询对象的排名低于在缓冲区中任意对象的排名,则这个对象被排除在用户的 top-k 列表中;否则,继续计算查询对象的排名并且更新缓存区.(2) RTA 要求对属于结果集的每个用户的偏好权重值去执行一个 top-k 操作.当用户数量很多时,RTA 算法的计算代价很大.不同于 RTA 算法,BBR 算法能够有效地解决当用户数量很多时的反向 top-k 查询,因为不需要访问每个用户的偏好.

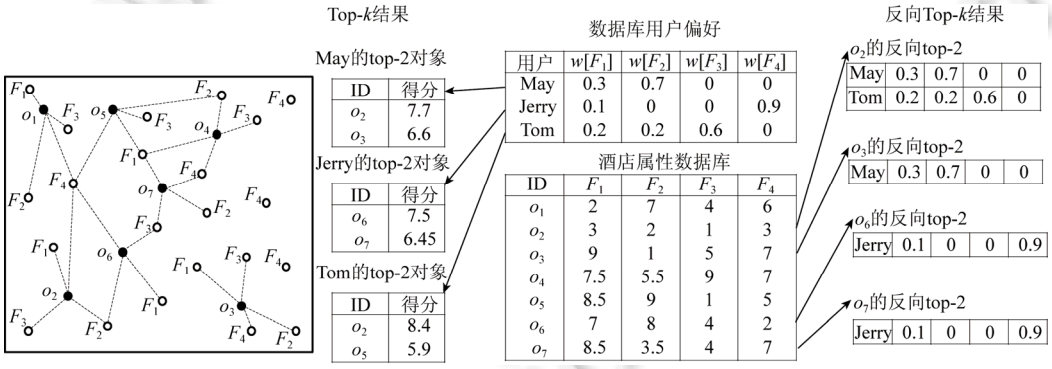


Fig.1 An example of reverse top-k

图 1 反向 top-k 的一个例子

在实际生活中,无论是用户对空间对象的偏好,还是对空间对象的属性,都不仅需要考虑非空间性质,而且也需要考虑空间性质.空间性质是指与对象相邻的周边设施相关的属性(如,欧式距离).之前有一些文献提出,在执行查询之前,把空间对象与周边设施相关的空间属性转化成其非空间属性,但是,这样做只适合当空间对象周边设施种类较少时.实际上,空间中设施的种类有很多,并且用户给出的偏好的数量远小于设施种类的数量,所以预先计算出所有空间对象到不同设施的空间属性浪费了大量的存储空间,给查询处理的性能带来相当大的困难.

同样以图 1 中的宾馆在线预订为例.如果图 1(b)中宾馆 o 周围可能存在的不同设施有 100 种,简单记为

$F_1,F_2,...,F_{100}$, 并且每一种设施对象的数量有很多,然而用户提出的偏好可能只有其中的几种,如 May 的偏好为 $(F_{49}:0.3,F_{100}:0.7)$, Jerry 的偏好为 $(F_1:0.4,F_{80}:0.6)$, Tom 的偏好为 $(F_{12}:0.1,F_{49}:0.4,F_{100}:0.5)$.

为了解决上述提到的浪费大量存储空间的问题,本文提出一种新型的查询,称为反向空间偏好 top-k 查询.该查询的查询结果与反向 top-k 查询结果相同.不同之处在于,对象的属性不是直接给出,而是由对象与周边设施的空间位置关系计算得到.假设一个宾馆经理根据不同用户给出的不同偏好,想知道哪些宾馆有更高的得分,反向空间偏好 top-k 查询的结果不仅可以把宾馆推荐给未来一些对宾馆有偏好属性的用户,而且也可以为不太受欢迎的宾馆在其周围进行招商.然而,随着宾馆周围设施种类的不断增加,将给查询处理性能带来很大的挑战.本文针对这样的问题提出一些优化算法加以解决.

本文的贡献点如下:

1) 本文定义了一种新型的查询,称为反向空间偏好 top-k 查询(RSP-k 查询).虽然传统的反向 top-k 查询也能处理从空间对象角度出发的查询,但是它们并没有考虑到对象周围的设施及对象与这些设施间的位置信息.

2) 本文提出了一些新颖的解决方案来有效地处理反向空间偏好 top-k 查询(RSP-k 查询).现有的方法直接处理这类问题会产生巨大的存储代价,存在较高的时间复杂度 $O(m\cdot n)$ 和较高的空间复杂度 $O(n\cdot F)$, 其中, m 是用户的数量, n 是查询空间对象的数量, F 是对象周围的设施种类数量.因此,本文提出 4 种方法来有效地减少代价:(1) 基于 R-tree 的基本算法;(2) 批量计算对象的剪枝算法;(3) 改进的批量剪枝算法;(4) 用户的偏好权重分组算法.前 3 种方法采用不同的索引结构来减少对查询对象得分的计算量.第 4 种方法针对用户偏好权重,把所有用户按照一定的规则进行分组以减少查询计算量.

3) 在真实数据集和模拟数据集上进行充分的实验,验证了所提出方法的有效性.

本文第 1 节回顾与本文相关的研究工作.第 2 节提出新型的 RSP-k 查询并给出形式化的定义.第 3 节介绍 4 种优化策略并给出详细的算法.第 4 节针对本文提出的每一种优化算法,分析其时间复杂度和空间复杂度.第 5 节给出充分的实验结果,展示所提出算法的有效性.第 6 节对全文进行总结.

1 相关工作

近年来,支持有效的 top-k 查询过程已经在数据库研究中得到了广泛的关注.而反向 top-k 查询是和 top-k 查询有着十分紧密的关联.下面,总结一些具有代表性的研究的大致思路.

Top-k 查询[5−8]提出了一种方便的方法,用户可以通过其偏好来找到很重要的对象.文献[6]提出了一种阈值的方法.该方法是在一个聚合函数 f 下,从不同的存储列表中合并对象的排名.阈值算法以循环方式按顺序扫描列表,并且计算每一个对象的聚合得分,从而维护 top-k 结果集.一旦剩余对象的聚合得分没有超过到目前为止被找到的 top-k 结果,则阈值算法终止.由于阈值算法的普遍性,有许多阈值算法的变形被提出来(如文献[5]).文献[9]首先生成实体化视图,一个 top-k 查询通过扫描与查询最相似的视图被回答.当 top-k 查询不断地被提出来时,一个算法可以决定最好的视图被实体化.然而,在文献[9]中,需要在确保满足性能之前实体化很多视图.

文献[7]中提到了 BRS 方法.该方法是通过 R*-tree 来处理数据对象的 top-k 查询的一个分支定界算法.BRS 使用一个堆来维护候选记录,以从上到下的方式遍历 R*-tree.在每一次迭代中,BRS 从堆中取最好的记录.如果记录是 R*-tree 的叶子节点,那么该记录输出作为下一个结果;如果已经有了足够多的结果(k 个),则算法停止.如果该记录是中间节点,那么对应的节点将被存储并且对于该节点的每一条记录 e 的最大得分被计算出来,同时将 e 插入到这个堆中.正像文献[9]这样,BRS 是一种优化 I/O 的算法,这意味着它访问仅仅包含 top-k 结果的树中的节点.由于最大得分是一个通用的概念,所以该算法能够同时应用在单色和非单色的偏好函数上.

Yiu 等人[10,11]首次研究空间偏好 top-k 查询问题.他们把对象分为两类:数据对象和辅助对象,并且建立索引分别索引两种不同对象.其中,数据对象被存在 R-tree 中,每一种辅助对象集被存在一个单独的聚合 R-tree 中(aR-tree)[12].同时,他们提出 3 种算法来解决基于 R-tree 和 aR-tree 的空间偏好查询.为了提高算法在文献[10]中的有效性,Rocha-Junior 等人提出一种方法来解决基于实体化方法的空间偏好 top-k 查询问题,从而在很大程度上节省了计算代价和 I/O 代价.近年来,处理空间偏好 top-k 查询的问题已经在路网上展开了研究[13,14].然而,无论是

传统的 top-k 查询还是空间偏好 top-k 查询都是从用户角度考虑,而没有从空间对象的角度考虑.

反向最近邻查询(RNN)和反向 skyline 最近邻查询返回对一个给定查询点作为最近邻查询点的一个集合[15].现在已有两种方法进行 RNN 查询:基于预计算的方法和动态方法[16].基于预计算的方法是预计算每一个点的 k 近邻结果集之后存储在系统中一些必要信息,为了更好地、有效地执行未来的某个查询[17].第 2 种动态方法经常使用一个索引结构来加速查询过程,比如 R-tree[18,19].

多样的 top-k 查询已在文献[20]中进行研究.文中给出一组数据点集合和一系列排名函数,并提出一些方法来计算所有关于 top-k 的函数,称为全 top-k 查询.文中提到的方法利用相似查询分享共同的结果来避免一个个地估计 top-k 查询.第 1 种算法(BINL)假设数据点被索引在一个多维索引结构上,并且函数基于相似性被分成组.BINL 分别处理每一组函数,并且为了避免计算 MBRs 的精确得分,使用界值来代替.虽然 BINL 减少了需要被检索的所有 top-k 查询的计算代价,但是 BINL 却不支持提前结束,这意味着数据点在索引结构中不得不被每一个函数分组遍历一次.于是文中提出了第 2 种方法,再物化视图的 BLPTA.BLPTA 通过遍历一次视图应答多样的 top-k 查询,并且可以提前终止算法.

与本文最相关的是反向 top-k 查询[3,21].反向 top-k 通过市场中潜在的商品对用户的影响被提出来,这些商品被认为是一些用户偏好的 top-k 结果.近年来,许多变化的反向 top-k 查询被提出来,包括识别最有影响的空间对象[22]、基于移动用户的位置流行程度的监测[23].代表性的工作是基于阈值的算法(RTA)[3]、基于网格算法(GRTA)[3]和基于分支定界算法的反向 top-k 查询(BBR)[4].通过一个小的缓存区,RTA 访问所有的用户.对于每一个用户,如果查询的空间对象的排名小于缓存区中任意一个空间对象的排名,那么该空间对象明显不属于用户偏好的 top-k 空间对象;否则,算法将继续查询空间对象的精确排名并且同时更新缓存区.GRTA 划分整个空间对象空间为一系列的网格,然后在内存中实现一些反向 top-k 的视图以有效地处理查询过程.BBR 使用分支定界的思想,从而达到不需要访问每个用户的偏好来处理查询过程的目的.

Zhang 等人[17]研究反向 k 排名问题.他们假设有一系列查询向量 Q 和一系列数据记录 D.给定一个正整数 m 和一个兴趣点 $p\in D$, 查询在 Q 中对于 p 的前 m 个候选的查询向量.他们考虑了查询向量是带有约束的,所提出的剪枝算法基于查询向量的特定位置来实现.

然而,这些反向 top-k 查询及变形与本文提出的 RSP-k 查询有着本质的不同.对于对象的属性,无论是空间性质还是非空间性质,在进行查询之前,这些属性得分都是预先给定的,而本文考虑的对象属性与其周围设施的空间位置相关,并且在得到用户偏好后再计算对象的属性得分.

2 问题定义

令 D 表示一个包含 n 个空间对象的集合,即 $D=\{O_1,O_2,...,O_n\}$.并且每一个对象 O 被表示为一个元组 $O=\langle l, F\rangle$,其中,l 表示对象 O 的空间位置,F 表示对象 O 的一个标签.假设标签种类共有 c 种,即 F 有 c 个不同种类,把 c 个不同种类的标签表示为一个集合,即 $\{F_i\mid i\in[1,c]\}$. 在 c 个标签种类的空间对象中,对应于查询对象的标签称为主标签,所有具有主标签的对象称为主体对象.其余 $c-1$ 种对象作为用户的偏好被提出来,称为辅助对象.W 是 m 个用户提出的偏好权重集合,每一个用户提出的偏好权重被描述成一个可用向量表示的 w_i, 即 $w_i=\{F_j\}$. w_i 中的一个向量 F_j 表示用户 i 对标签为 F_j 的对象感兴趣,并给出对于该标签的偏好权重值,表示为 $w_i[F_j]$, 简写为 $w_i[j]$. 假设集合 W 已知,那么对于一个用户 w_i 来说,有 $\sum_{j=1}^{c-1}w_i[j]=1$. 这种带有偏好的权重模型已经在许多相关文献中广泛应用[20−28].

为方便表示,以下用 F_j 代替 $O.F_j$ 表示标签为 F_j 的空间对象.假设用符号 o 表示主体对象,对于任意一个用户 i,由得分函数 $f(w_i,o)$ 来计算主体对象 o 对于用户 i 的得分.该函数被定义为 $f(w_i,o)=\sum_{j=1}^{c-1}\left(w_i[j]\left(1-\frac{\min d(o,F_j)}{d_{\max}}\right)\right)$, 其中, $\min d(o,F_j)$ 表示主体对象 o 与标签为 F_j 的所有辅助对象中最近的一个对象的距离, $d_{\max}$ 表示给定空间中容忍的最大距离.不失一般性,假设一个用户对空间中一个对象的偏好是得分越大越

好,也就是说,对于用户提出的偏好,具有该偏好标签的辅助对象离主体对象的距离越近越好.

因为 top-k 查询和反向 top-k 查询与本文的工作有很强的相关性,所以首先回顾一下相关两个查询的定义,然后提出反向空间偏好 top-k 查询(RSP-k 查询).

定义 1(top-k 查询($TP(k,w)$)). 给定一个空间对象数据库 D,一个用户的偏好 w 和一个正整数 k,top-k 查询返回一个对象集合 $S, S\subseteq D, |S|=k$. 那么,对于 $\forall o_i\in S, \forall o_j\in(D-S)$, 有 $f(w,o_i)\geqslant f(w,o_j)$.

定义 2(反向 top-k 查询)[3]. 给定一个对象数据库 D,一个权重集 W,一个正整数 k 和一个指定的查询点 q,反向 top-k 查询返回一个权重集 W 中的一个集合 $S, S\subseteq W$. 对于每一个 $w\in S, q\in TP(k,w)$, 相反,对于任意一个 $w\notin S, q\notin TP(k,w)$.

下面介绍本文提出的反向空间偏好 top-k 查询(RSP-k 查询).RSP-k 是反向 top-k 的变形.

首先了解一下 RSP-k 查询与反向 top-k 查询不同的相关定义.

定义 3(空间地理对象).一个空间地理对象表示为一个元组 $O=\langle l,F\rangle$, 其中, l 表示空间中对象的经度和纬度位置(例,52.16N,21.76E), F 表示该对象的一个标签(例,宾馆).

进一步,给出 RSP-k 查询的定义.

定义 4(反向偏好 top-k(RSP-k)查询). 给定一个含有 n 个空间地理对象的数据库 D,D 中包含主体对象和辅助对象,用户的权重集合 W 和一个正整数 k,若 o 表示主体对象,则 RSP-k 查询返回一个集合 $S, S\subseteq W$. 对于 $\forall w_i\in S, \forall w_j\in(W-S)$, 有 $f(w_i,o)\geqslant f(w_j,o)$.

下一节,本文将提出几种新颖的解决方案,以有效地处理 RSP-k 查询.

3 查询处理方法

本节介绍几种优化方法,以有效地处理所提出的 RSP-k 查询问题.首先给出系统结构,如图 2 所示.通过对空间对象和用户的同时处理,最终得到结果集.

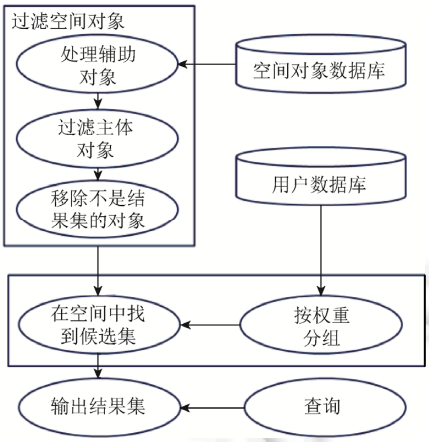


Fig.2 Flow of processing

图 2 处理流程图

3.1 基于R-tree的剪枝算法

最直接的方法是计算每一个主体对象对每一个用户提出的偏好的得分,但是计算代价特别大.通常,可以利用 R-tree 避免查询时产生的巨大计算代价.本节提出一种基于 R-tree 的剪枝算法来处理 RSP-k 查询.基于 R-tree 的方法是基于文献[4]中分支定界的方法而提出的.

首先,回顾一下两个点的支配关系.这种关系被很多文献提到过,包括 skyline[29]和反向 top-k 查询[3].

定义 5(支配). 一个对象 o_1 能够支配另外一个对象 o_2, 当且仅当满足以下两个条件:(i) $\forall j, o_1[j]\leqslant o_2[j]$;

(ii) $\exists h, o_1[h] < o_2[h]$. 其中, $o[j], o[h]$ 是对象 o 对于属性 j, h 的得分.

从定义 5 可以推出,若 $f(w_i,o)$ 表示返回对象 o 对偏好 w_i 的得分函数,则对于 $\forall i, w_i \geqslant 0$, 当对象 o_1 支配另外一个对象 o_2 时,有 $f(w,o_1) \leqslant f(w,o_2)$. 当 RSP-$k$ 查询发出时,为所有的主体对象构建一个 R-tree 索引. r 表示在 R-tree 上的一个最小边界矩形(MBR), $r_{\min}$ 和 $r_{\max}$ 表示 r 与 w_i 中辅助对象距离的最小值和最大值.

根据以上分析可以得到两个推论.

推论 1. 给出一个查询对象 q,一个偏好权重 w_i 和 R-tree 上的一个 MBR r.如果 $f(w_i,q) < f(w_i,r_{\max})$, 那么对于 $\forall o \in r$, 有 $f(w_i,q) < f(w_i,o)$, 可得 r 中所有对象 o 的排名都比 q 靠前.

推论 2. 给出一个查询对象 q,一个偏好权重 w_i 和 R-tree 上的一个 MBR r.如果 $f(w_i,r_{\min}) < f(w_i,q)$, 那么对于 $\forall o \in r$, 有 $f(w_i,o) < f(w_i,q)$, 可得 r 中所有对象 o 的排名都比 q 靠后.

算法 1 展示了基于 R-tree 剪枝算法的主要步骤.

算法 1. 基于 R-tree 的剪枝算法(RT).

```
1. Input: Q,Q';   /* 初始化两个队列 Q 和 Q' */
2.        R[w_i] ← ∅, W' ← ∅;   /* R[w_i] 记录 q 在每个用户权重 w_i 中的排名, W' 记录结果的集合 */
3. Output: W';
4. for each w_i in W do   /* 对于每一个用户提出的偏好 */
5.     enqueue( Q,R.root );
6.     while ( r = dequeue(Q) ≠ ∅ ) do
7.         if ( f(w_i,r_min) < f(w_i,q) ) then   /* 满足推论 2 时 */
8.             prune all the objects in r;   /* 剪枝掉 MBR r 中所有对象 */
9.         else if ( f(w_i,q) < f(w_i,r_max) ) then   /* 满足推论 1 时 */
10.            |R[w_i]| ← |R[w_i]| + |r|;   /* 计算在查询对象 q 排名之前的对象数量 */
11.        if ( |R[w_i]| < k ) then
12.            if ( D is a leaf node) then   /* MBR r 是叶子节点 */
13.                enqueue( Q', r );
14.            else if ( ∀r' is r's child) then   /* MBR r 是非叶子节点,对 r 的孩子节点进行操作 */
15.                enqueue( Q , r' );
16.        for each r in Q' do
17.            for each o in r do
18.                if f(w_i,o) > f(w_i,q) then
19.                    |R[w_i]| ← |R[w_i]| +1;   /* 对所有排名在查询对象 q 之前的对象都放到 W' 中 */
20.                    W' ← W' ⋃ {w_i};
21.    return W';
```

假设所有的主体对象已经在 R-tree 上被索引.首先,初始化两个队列 Q 和 Q' (第 1 行).根据查询过程,通过遍历 R-tree 来计算每个用户提出的偏好 $w_i \in W$ 的得分并进行排名.对于每一个在 R-tree 上的 MBR r (第 4 行、第 5 行), 算法计算关于 w_i 的 $r_{\min}$ 和 $r_{\max}$ 的得分.通过推论 1 和推论 2,如果节点满足 $f(w_i,q) < f(w_i,r_{\max})$ 或者 $f(w_i,r_{\min}) < f(w_i,q)$, 则节点被安全地剪枝掉(第 6 行~第 10 行).否则,所有 r 中的孩子节点(当 o 是非叶子节点时)被添加到 Q 中做进一步查询.队列 Q' 记录所有未被剪枝的叶子节点(第 12 行~第 16 行).

基于 R-tree 的剪枝算法的优势在于可以将所有与辅助对象距离相近的主体对象通过 MBRs 同时进行估计.然而,这种方法必须考虑计算出所有用户偏好的辅助对象与主体对象之间的距离.这样不仅浪费存储代价而且查找起来也会消耗大量时间.下面提出一种优化计算方法,当用户给出偏好权重时,可以快速地查找到 RSP-k 查询的结果集.

3.2 批量计算主体对象的剪枝算法

当主体对象的数量很多时,实际就会有很大一部分主体对象不属于任何用户的 top-k 个结果.所以不需要把所有主体对象与其最近的所有辅助对象的距离都计算出来,只需要通过用户的偏好来计算.那么,为了更快地计算,本节采用一种 Quad-tree 索引结构来聚集空间中的主体对象.

在本文中使用 Quad-tree 索引的原因是 Quad-tree 方便更新.因为 Quad-tree 中的每个网格(cell)都会把 RSP-k 查询中的所有主体对象索引在一个互斥的 cell 中,所以对主体对象进行插入或者删除操作都简单、方便.相反地,在 R-tree 索引的 MBR r 中,可能存在互相重叠的 MBR r,这样,当对空间中主体对象进行插入或删除时,Quad-tree 平均耗时远小于 R-tree 的插入和删除的花费.

另外,为了避免计算所有的主体对象到其辅助对象之间的最短距离,本文也把辅助对象进行了索引,这样不仅能够方便查询,而且也能减少计算代价.具体方法如下:

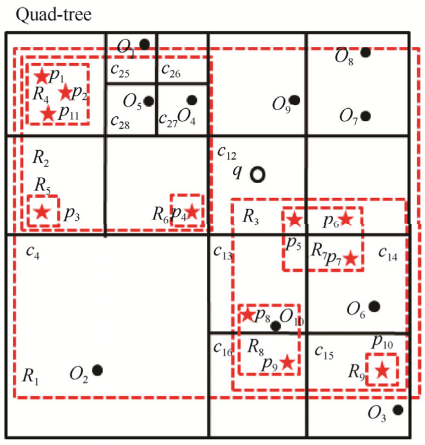


Fig.3 Spatial structure

图 3 空间结构

1) 为了把有距离相近的主体对象进行聚集,对主体对象用 Quad-tree 建立索引.同时,为了最大化有效地剪枝并且减少查询计算量,所有辅助对象建立一个 IR-tree[17].索引结构如图 3~图 5 所示.其中,图 3 表示主体对象和辅助对象在空间中的分布情况,图 5 是对图 3 中的主体对象和辅助对象分别建立的索引结构,图 4 是辅助对象在 IR-tree 的倒排文件中的具体信息.根据用户提出的偏好 W,得到用户提出的偏好的辅助对象 F_i. 由辅助对象的 IR-tree 找出与查询对象 q 最近的包含 F_i 的 MBR r.

2) 对于每一个用户给定的偏好 w_i, 根据 Quad-tree 索引,每个带有主体对象的 cell c 都需要找到最近的包含 F_i 的 MBR r. 比较 q 所在 cell $c.q$ 与 $r.F_i$ 的最近距离 $\min dist(c.q,r.F_i)$ 和其他主体对象所在 cell $c'.o$ 与 $r.F_i$ 的最远距离 $\max dist(c'.o,r.F_i)$, 比较 $c.q$ 与 $r.F_i$ 的最远距离 $\max dist(c.q,r.F_i)$ 和其他 $c'.o$ 与 $r.F_i$ 的最近距离 $\min dist(c.o,r.F_i)$.

辅助对象	InvFile 4	InvFile 5	InvFile 6	InvFile 7	InvFile 8	InvFile 9
F_1	p_1			p_5,p_6		
F_2		p_3	p_4		p_8	p_{10}
F_3	p_2					
F_4	p_{11}			p_7	p_9	

辅助对象	InvFile 2	InvFile 3	InvFile 1
F_1	R_4	R_7	R_2,R_3
F_2	R_5,R_6	R_8,R_9	R_2,R_3
F_3	R_4		R_2
F_4	R_4	R_7,R_8	R_2,R_3

Fig.4 Inverted files of objects in Fig.3 and Fig.5

图 4 图 3 和图 5 中对象的倒排文件

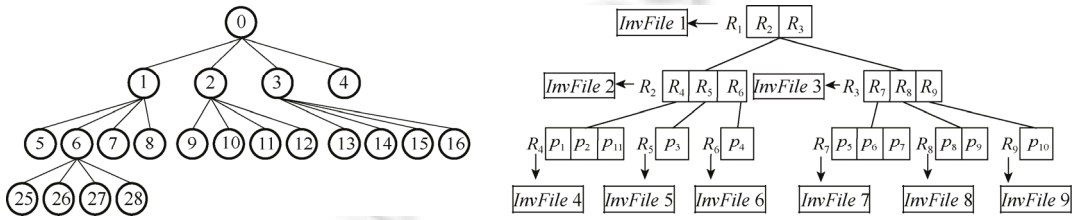


Fig.5 Index structure of objects in spatial

图 5 空间中对象的索引结构

当满足 $\min dist(c.q,r.F_i) > \max dist(c'.o,r.F_i)$ 和 $\max dist(c.q,r.F_i) < \min dist(c'.o,r.F_i)$ 条件时,该主体对象 $c'.o$ 所在 cell 被存在一个缓存区中.其余的对象将进行精确计算.可以得到以下的推论.

推论 3. 给出一个查询对象 q 所在的 cell c,一个辅助对象 F_i 所在 MBR r.如果 $\min dist(c.q,r.F_i) > \max dist(c'.o,r.F_i)$,那么对于所有 c' 中的对象 o,有 $f(F_i,q) < f(F_i,c'.o)$,可得查询对象 q 和主体对象 o 关于辅助对象 F_i 的得分,c' 中所有对象 o 的排名都比查询对象 q 靠前.反之,如果 $\max dist(c.q,r.F_i) < \min dist(c'.o,r.F_i)$,则 c' 中所有对象 o 的排名都比查询对象 q 靠后.

证明:若存在 $\min dist(c.q,r.F_i) > \max dist(c'.o,r.F_i)$,则可以得到,在 cell c 中任意位置的查询对象 q 与另一个 cell c'中的对象 o 满足: $dist(q,r.F_i) > \max dist(c'.o,r.F_i)$,根据得分 $f(w_i,o)=\sum_{j=1}^{c-1}\left(w_i[j]\left(1-\frac{\min d(o,F_j)}{d_{\max}}\right)\right)$,可以推导出 $f(F_i,q) < \min f(F_i,c'.o)$,有 $f(F_i,q) < f(F_i,c'.o)$,所以可得查询对象 q 和主体对象 o 关于辅助对象 F_i 的得分,c' 中所有对象 o 的排名都比 q 靠前.若存在 $\max dist(c.q,r.F_i) < \min dist(c'.o,r.F_i)$,则可以得到对于在 cell c 中的任意位置的查询对象 q 与另一个 cell c'中的对象 o 满足: $\max dist(q,r.F_i) < dist(c'.o,r.F_i)$,可以推导出 $\min f(F_i,q) > f(F_i,c'.o)$,有 $f(F_i,q) > f(F_i,c'.o)$,所以可得查询对象 q 和主体对象 o 关于辅助对象 F_i 的得分,c'中所有对象 o 的排名都比 q 靠后. □

3) 找出所有用户提出的偏好的所有辅助对象 F_i 与查询对象 q 和主体对象 o 之间的关系.

4) 为了快速并且有效地得到推论 3,根据以上步骤进行验证.若满足推论 3 中 c' 中所有对象 o 的排名都比 q 靠前,则说明 Quad-tree cell 中的对象 o 对于用户提出的偏好无论权重的得分,一定大于对象 q,如果个数大于 k,则说明查询对象 q 不是 RSP-k 查询的结果.若个数没有达到 k,则需要精确求解,求解之前,需要观察是否满足推论 3 中 c' 中所有对象 o 的排名都比 q 靠后,若满足,则这些节点中的主体对象 o 不参与精确计算,被安全地剪枝掉.剩下的点进行精确求解,当求到有 k 个主体对象得分时,算法结束.

以图 3 为例,考虑空间数据库中包含主体对象集 $D=\{o_1,o_2,...,o_7\}$ 及其中一个查询点 q 和辅助对象集,已知用户偏好数据库包含 3 个用户, $W=\{w_1,w_2,w_3\}$, 并且每个用户的偏好辅助对象及偏好权重分别为 $w_1=(F_1:0.4, F_3:0.6), w_2=(F_1:0.7,F_2:0.3), w_3=(F_1:0.8,F_3:0.2)$. 假设 $k=2$. 首先,检验 w_1. 可以看到, w_1 中提到的辅助对象是 F_1 和 F_3, 从图 3 和图 4 中看出,含有 F_1 的对象在 MBRs R_2,R_3,R_4 和 R_7 中,其次观察查询点 q 所在的 cell c_{12} 和其他 cell 与 R-tree 上的 R_2 的关系.经计算, q 到 R_2 的最近距离是 0,最远距离是 12.从 Quad-tree 结构中可以看出,没有任何一个 cell 到 R_2 的最远距离小于 q 到 R_2 的最近距离,所以不存在能够被剪枝掉的主体对象.同时,没有任何一个 c 到 R_2 的最近距离大于 q 到 R_2 的最远距离,所以也不存在能够被剪枝掉的主体对象.如此继续查找 R_4, 同样观察 q 到 R_4 的最近距离是 7, q 到 R_4 的最远距离是 12,能够得到 c_{25},c_{28} 和 c_{27} 到 R_4 的最远距离都比 q 到 R_4 的最近距离要近,则记录偏好对象 F_1 到 q 的距离大于所有在 cell c_{25},c_{28} 和 c_{27} 中的对象到 q 的距离.同时记录辅助对象 F_1 到对象的距离一定远于 F_1 到 q 的距离, c_{13},c_{14} 和 c_{15} .类似地,找出辅助对象 F_3 与查询对象 q 和其他主体对象的关系.

3.3 改进的批量剪枝算法

之前的方法虽然有效地减少了计算的工作量.但是在过滤掉不可能成为任何用户的 top-k 的对象后,如果剩余需要计算的每个叶子节点中的对象数量很多,那么计算代价大仍是问题.本节提出一种改进的批量剪枝算法,以进一步降低计算代价.

在批量剪枝算法进行完对主体对象剪枝后,需要对 Quad-tree 中未被剪枝的 cells 中所有主体对象进行精确计算得分.为了更加有效地剪枝和减少计算代价,本文在批量剪枝后进一步利用 Quad-tree 中心点来划分空间.

具体划分方法如下:对于不能直接过滤的同一个 cell 中的主体对象,根据该 cell 的中心点向下划分,并且根据用户给出的偏好权重值来判断该 cell 的子节点中的主体对象是否需要被精确求解.

已知一个 cell 被划分为 4 个子节点,并且每个子节点的边长为 l_c, 如果中心点表示为 c_v, 那么根据正方形定理可以得到:

$$d(c_v,F_i)-\frac{\sqrt{2}}{2}l_c \leqslant d(o,F_i) \leqslant d(c_v,F_i)+\frac{\sqrt{2}}{2}l_c,$$

于是可以得到 Quad-tree 中任意一个对象 o 到辅助对象 F_i 的最近距离和最远距离.

定理 1. Quad-tree 中任意一个对象 o 到辅助对象 F_i 的最近距离为 $d(o,F_i)=d(c_v,F_i)-\frac{\sqrt{2}}{2}l_c$, 最远距离为 $d(o,F_i)=d(c_v,F_i)+\frac{\sqrt{2}}{2}l_c$.

证明:根据 Quad-tree 的性质,每个节点都是由一个正方形组成.那么对于一个正方形中任意的一个点 o 到正方形的边缘的距离都存在 $\pm\frac{\sqrt{2}}{2}l_c$ 的误差,其中 l_c 为该正方形的边长.因此可以得到对于正方形内任意一点到空间中另外一点的最大、最小距离分别为 $d(o,F_i)=d(c_v,F_i)+\frac{\sqrt{2}}{2}l_c$ 和 $d(o,F_i)=d(c_v,F_i)-\frac{\sqrt{2}}{2}l_c$. □

算法 2 给出了改进的批量剪枝算法具体流程.

算法 2. RSP-k(q,W,k).

```
1. Input: Q, Q', R[w_i] ← ∅, W' ← ∅;   /* 初始化队列 Q 和 Q', R[w_i] 记录 q 在每个用户权重 w_i 中的排名,
                                          W' 记录结果的集合 */
2. Output: W';
3. enqueue( Q, Quad-tree.Root);
4. while ( c = dequeue(Q) ≠ ∅ ) do
5.     if ( max dist(c.q,r.F_i) < min dist(c.o,r.F_i) ) then
6.         prune all the objects in c;   /* 剪枝掉 cell c 中所有对象 */
7.     else if ( min dist(c.q,r.F_i) > max dist(c.o,r.F_i) ) then
8.         |R[w_i]| ← |R[w_i]| + |c|;   /* 所有排名比查询对象 q 靠前的对象都放入 R[w_i] 中 */
9.         if ( |R[w_i]| < k ) then   /* W' 中对象数量小于 k 个,继续划分 cell c */
10.            if (c is a leaf node && Num(c) < λ ) then   /* cell c 是叶子节点并且 cell c 中主体对象的数量
                                                              比给定阈值 λ 小时 */
11.                enqueue( Q',c );   /* 对于不能划分的 cell c,进行求解计算 */
12.            if ( Num(c) ≥ λ ) then   /* 当 cell 中主体对象的数量比给定阈值 λ 大时 */
13.                divide c;   /* 继续分割 cell c */
14.    for ( ∀c' is the children of c) do   /* 对于继续划分的 cell c,对其孩子节点进行优化 */
15.        enqueue( Q,c' );
16.        for each c' in Q' do
17.            for each object o in c' do
18.                if f(w_i,o) < f(w_i,q) then
19.                    |R[w_i]| ← |R[w_i]| +1;
20.                    W' ← W' ∪ {w_i};
21.    return W';
```

若一个子节点的中心点与辅助对象所在的 MBR r 的最近距离的得分小于查询点 q 到相同偏好对象的 MBR r 距离得分,则这个子节点中的所有主体对象的得分都比 q 的得分小,不需要考虑这个子节点中的主体对象,该子节点中的主体对象被安全剪枝(第 4 行、第 5 行).否则,计算其与辅助对象所在的 MBR r 的最远距离的得分,若得分大于查询点 q 到相同辅助对象的 MBR r 的距离的得分,则需要计算整个子节点中的对象的个数,使用一个参数 λ 控制 cell 的划分.如果 cell 中主体对象的数量 $Num(c)$ 大于 λ,则该 cell 向下划分;否则不继续划分(第 6 行~第 12 行).若其中所有对象的得分都比 q 的得分大,则该节点中的所有对象点都被保留在之前的缓存区中,如果缓存区中对象的个数大于 k 个,则查询点 q 不是用户 w 的前 k 个结果;若个数小于 k ,则继续进行精确求解(第 13 行~第 19 行).

以上研究的 3 种方法都是针对主体对象和辅助对象如何剪枝而提出的,对于用户偏好的提出并没有做出优化.因此,仍然需要对每一个用户进行计算.下一节提出针对用户偏好的处理方法,能够进一步降低查询时间.

3.4 基于用户权重的分组算法

虽然之前的方法可以通过 IR-tree 和 Quad-tree 将一部分主体对象过滤掉.然而,当用户数量和提出的偏好对象重复数量很大时,上面的算法性能将有所下降.在现实生活中,如果一个度假村的经理想查询该度假村能够吸引哪些游客,通常,游客去度假村可能组团去,这样,同一个旅行团中的游客对周边设施可能有同样的偏好.所以,本节提出一种用户的权重分组法来合并相似的偏好和权重,以减少计算代价.

在给定所有用户的偏好后,找出出现频率最高的前 α 个辅助对象,然后找到偏好包含这些辅助对象的所有用户.把这些用户给出的偏好按照权重大小分组.每种辅助对象被分为 θ 组等间隔的组,表示为 $\left[0,\frac{1}{\theta}\right],\left[\frac{1}{\theta},\frac{2}{\theta}\right],...,\left[\frac{\theta-1}{\theta},1\right]$,那么一共存在 θ^{α} 这样的分组,表示为 g_i.注意,没有任何偏好权重的分组不需要考虑.假设 Lg_i 和 Ug_i 分别表示每一组中每个区间的下界和上界.那么,一个分组 g_i 如果满足以下条件才被认为是有效的:$\sum_{m-1}^{\alpha} Lg_i(m)<1$ 或者 $\sum_{m-1}^{\alpha} Ug_i(m)>1$.进而可以找到每一个组与查询点 q 的关系.比较需要求解的所有主体对象所在 cell 与 q 的关系.由于每个对象分组都有上下界,这样,每个分组与查询点 q 的得分与其他主体对象所在 cell 的得分存在以下几种关系.

1) $Lrel(c,g_i)$

当且仅当 $\forall o\in c,\forall w\in g_i, f(w,o)<f(w,q)$ 满足,可以得到在 c 中所有的对象在偏好权重属于分组 g_i 中的得分都小于查询点 q 的得分,这样,c 中所有的对象被安全剪枝.

2) $Urel(c,g_i)$

当且仅当 $\forall o\in c,\forall w\in g_i, f(w,o)>f(w,q)$ 满足,可以得到在 c 中所有的对象在偏好权重属于分组 g_i 中的得分都大于查询点 q 的得分,这样,c 中所有的对象被放到结果集中,并记录个数,当个数大于 k 时,算法终止;当个数小于 k 时,算法继续.

3) $Arel(c,g_i)$

当且仅当 $\forall o\in c,\exists w\in g_i, f(w,o)\leqslant f(w,q)\wedge\exists w'\in g_i, f(w',q)\leqslant f(w',o)$ 满足,需要精确计算在 c 中所有的对象在偏好权重属于分组 g_i 中的得分.

4) 若以上情况都不满足,则需要划分节点 c,以进一步找出该节点 c 的子节点 c' 到分组 g_i 的得分.重复以上 3 个步骤,直到结束.

具体地,算法 3 给出了基于用户权重分组的流程.

算法 3. 基于用户权重分组算法.

```
1. Input: Q, Q_{g_i}, R[g_i] ← ∅, W' ← ∅;   /* 初始化队列 Q,Q_{g_i},其中 Q_{g_i} 表示在分组 g_i 中的队列,R[g_i] 记
                                               录 q 在每个分组 g_i 中的排名,W' 记录结果的集合 */
2. Output: W';
3. for each valid g_i do   /* 对每一个有效地分组进行处理 */
4.     enqueue(Q,Quad-tree.Root);
5.     while ( c = dequeue(Q) ≠ ∅ ) do
6.         if ( Lrel(c,g_i) ) then          /* c 中所有的对象被安全剪枝 */
7.             prune all objects in c;
8.         if ( Urel(c,g_i) ) then          /* c 中所有的对象的个数加到 R[g_i] 中 */
9.             |R[g_i]| ← |R[g_i]| + |c|;
10.            if ( |R[g_i]| < k ) then
```

```
11.                continue;
12.            else
13.                q is not a result in group g_i;   /* 查询对象 q 不是组 g_i 中的结果 */
14.        if ( Arel(c, g_i) ) then   /* 计算在 c 中所有的对象在偏好权重属于分组 g_i 中的得分 */
15.            enqueue( Q_{g_i}, c );
16.        else
17.            for ( ∀c' is a child of c ) do
18.                enqueue( Q_{g_i}, c' );
19.        W' ← W' ∪ {w_i};                    /* 更新 W' */
20.        if ( |ℝ[w_i]| < k ) then
21.            report q is a result in group g_i;   /* 查询对象 q 属于组 g_i 中的结果 */
22.        else
23.            q is not a result in group g_i;   /* 查询对象 q 不是组 g_i 中的结果 */
24.        W' ← W' ∪ {g_i};
25.    return W';
```

4 讨 论

本节总结以上所提出方法的时间代价和空间代价.在基本算法中,最坏的情况是需要处理$|W|$top-k 个查询,所以基于 R-tree 的算法是一种蛮力搜索算法.然而,基于 R-tree 的算法通过实验验证是少于$|W|$top-k 个查询而返回正确结果.所以,基于 R-tree 的算法的时间复杂度为$O(n_R \cdot m)$,空间复杂度为$O(n_R)$,其中,n 是空间中对象的个数,m 是用户的权重个数,n_R 是R-tree中MBRs的数量.在批量计算主体对象的剪枝算法中,由于把主体对象和辅助对象分别用不同的索引结构进行索引,则该算法的时间复杂度为$O((n_c + n'_R) \cdot m)$,空间复杂度为$O(n_c + n'_R)$.其中,n_c 表示主体对象在Quad-tree中cells的数量,n'_R 表示辅助对象在IR-tree中MBRs的数量.在用户的权重分组中,把对象提出的偏好按权重大小进行分组,则算法的时间复杂度为$O((n_c + n'_R) \cdot m_g)$,空间复杂度为$O((n_c + n'_R) + m_g)$,其中$m_g$ 为分组的数量.

5 实 验

本节将对以上提出的算法进行实验评估.所有实验都是基于Java语言进行,实验环境采用Intel(R) Core(TM) i7-3770 CPU @3.40GHz和8GB内存.在每一组实验中,查询1 000次,每一次随机地选择一个不同的查询点,最终求出平均值作为实验结果.

5.1 数据集及设置

在实验评估中使用两种类型的数据集,包括空间对象数据集 D 和用户偏好权重数据集 W .

空间对象数据集 D.使用两个真实数据集进行评估.第 1 个数据集 EURO.EURO 是一个欧洲地区中包含有 179 506 个兴趣点的数据集.每一个兴趣点表示一个空间地理对象,并且每一个对象都有其地理位置和标签(如,公园、医院、超市等).第 2 个数据集 FSQ.FSQ 是一个从 Foursquare 中提取的数据集,该数据集包含 200 万个带有空间地理位置和标签的兴趣点.

用户偏好权重数据集 W.生成两个类型的数据集,UNI 和 CLU.UNI 数据集是指生成的用户偏好权重是随机的均匀分布;CLU 数据集通过两个参数来创建:ρ和σ^2.首先随机生成ρ个独立的用户偏好权重,把这ρ个偏好权重当作聚簇中心.然后围绕着聚簇中心生成方差为σ^2 的一些偏好权重.并且根据实际情况选取每个权重集合中包含 2 个~4 个不同的偏好权重,λ默认值为 20.由于真实数据集中兴趣点的种类数量很大,为了检验数据更有效果,实验从真实数据集中选取不同大小的$|F|$来验证所提出算法的有效性.

在实验中,估计以下 4 种方法:(1) 基于 R-tree 的剪枝算法,简单记为 RT.RT 作为基础算法在第 3.1 节中被提出;(2) 批量计算主体对象的剪枝算法,表示为 QIR;(3) 改进的批量剪枝算法,表示为 MQIR;(4) 基于分组处理用户权重,G-MQIR,实验中也对算法 G-MQIR-UNI 和 G-MQIR-CLU 分别进行实验评估.

实验主要对为处理 RSP-k 查询而提出的算法的执行时间和 I/O 次数进行评估.由于本文算法是基于索引结构设计的,算法执行时每对索引中的叶子节点进行一次访问,就相当于进行一次 I/O 读取,所以 I/O 次数即记录每次查询索引结构中叶子节点的访问次数,并且叶节点的大小定义为 4KB.实验中给出的参数变化为:结果数 k 为 10~50,用户偏好的数量 $|W|$ 为 5k~100k,辅助对象标签数量 $|F|$ 为 100~500,聚集中心个数 ρ 为 5~50.默认值设置为: $k=10,|W|=10k,|F|=200,\rho=10$. 其中,为了验证辅助对象标签数量 $|F|$ 的不同对实验效果的影响,本文从真实数据集中的数据点的标签总数量中进行抽取并做比较.实验参数具体在表 1 中给出.

Table 1 Experiment parameters

表 1 实验参数

参数	设置	默认值
#results(k)	10,20,30,40,50	10
$\|W\|$	5k,10k,50k,100k	10k
$\|F\|$	100,200,300,400,500	200
$\|\rho\|$	5,10,20,30,50	10
α	2,3,4,5	4
θ	2,3,4,5	4

5.2 实验评估结果

5.2.1 不同数据集对查询时间的影响

图 6 对比了 4 种不同算法在数据集不同时,运行时间的变化.从图 6 中可以看出,对于两个不同的真实数据集,在不同算法的比较下,算法 G-MQIR-UNI 和 G-MQIR-CLU 比其他算法查询时间要快.这说明提出的优化算法有一定的效果.在其他参数相同的情况下,FSQ 比 EURO 的查询时间要长,由于 FSQ 数据集中空间对象数量比 EURO 数据集的对象数量多,那么即使选定了辅助对象的种类数量,FSQ 的辅助对象的数量仍然要多于 EURO 数据集中的辅助对象数量,所以,在真实数据集 FSQ 上做查询,查询时间要长于在数据集 FSQ 上做查询.

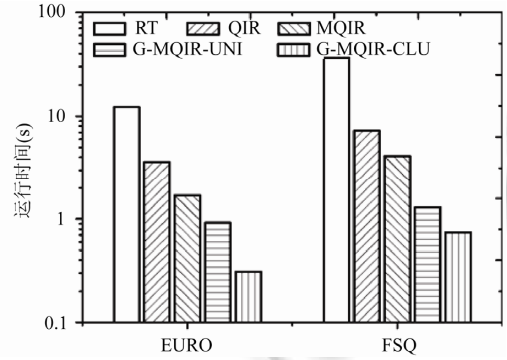


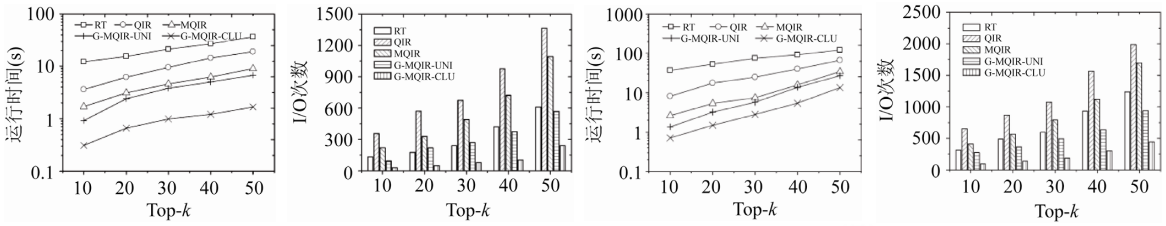
Fig.6 Effect of query time in different datasets

图 6 不同数据集对查询时间的影响

5.2.2 参数 k 对查询结果的影响

图 7 对比了两个真实数据集在不同算法下的 k 值变化对查询结果的影响.查询结果分别从查询时间和 I/O 次数进行比较.从实验结果图中可以看出,随着 k 的不断增大,所有算法的查询时间和 I/O 次数也在增加.算法 G-MQIR-UNI 和 G-MQIR-CLU 比其他算法查询时间要快.其中,算法 G-MQIR-UNI 并没有比算法 MQIR 在查询时间上快很多,因为算法 G-MQIR-UNI 虽然对用户偏好权重进行了分组,但是由于用户偏好权重是均匀分布的,因此对分组后的用户偏好权重不能有效地进行合并计算.但是,如果用户偏好权重按照聚集分布进行分组,查询

性能将有很明显的提升.对于 I/O 次数,由于算法 RT 对辅助对象没有进行预处理,在查询发布时进行在线计算,虽然查询时间比较长,但是 I/O 次数只存在于计算主体对象中.



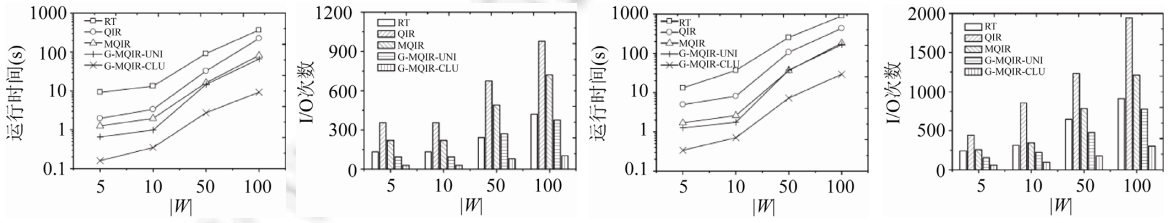
(a) 运行时间-EURO (b) I/O 次数-EURO (c) 运行时间-FSQ (d) I/O 次数-FSQ

Fig.7 Varying of k

图 7 k 的变化

5.2.3 用户偏好权重W对查询结果的影响

图 8 表示的实验结果是当用户偏好的数量$|W|$从 5k 变化到 100k 时,不同算法的查询性能比较.对于所有的算法,随着用户偏好的数量$|W|$的增加,查询时间越长,I/O 次数越大.从实验结果可以看出,算法 G-MQIR-UNI 和算法 G-MQIR-CLU 查询时间仍然是最快的,因为对用户偏好进行分组可以有效地对相似用户偏好进行合并计算,从而减少计算量.同样地,I/O 次数也会随着优化算法剪枝的效果而随之减少.



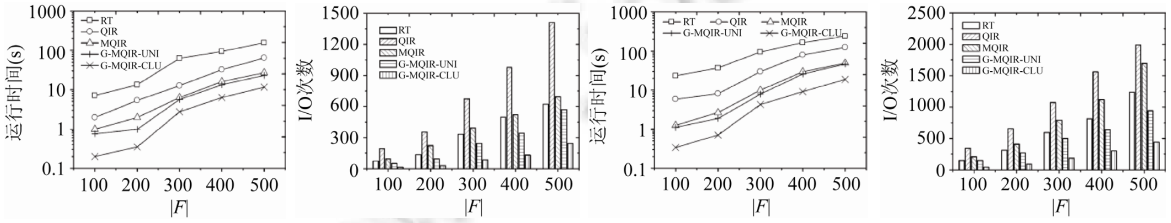
(a) 运行时间-EURO (b) I/O 次数-EURO (c) 运行时间-FSQ (d) I/O 次数-FSQ

Fig.8 Varying of W

图 8 W 的变化

5.2.4 辅助对象标签数量$|F|$对查询结果的影响

图 9 展示了所有方法在辅助对象标签数量$|F|$从 100 变化到 500 时的查询时间和 I/O 次数的变化.从实验结果中可以明显地看出,随着辅助对象标签数量$|F|$的增大,用户偏好的种类也会随之增加,这样,在计算每个主体对象的得分时会产生更多的计算量,查询时间也随之线性地增加.注意到,算法 G-MQIR-UNI 在查询时间上并没有远远优于算法MQIR,因为算法G-MQIR-UNI是把用户的偏好权重均匀地分布,对于分组后的用户偏好权重不能有效地进行合并计算.



(a) 运行时间-EURO (b) I/O 次数-EURO (c) 运行时间-FSQ (d) I/O 次数-FSQ

Fig.9 Varying of F

图 9 F 的变化

5.2.5 聚集中心个数 ρ 对查询性能的影响

图 10 说明了聚集中心的个数选取的不同对查询时间的变化.实验在两个数据集上分别进行比较.从实验结果中可以看出,对于采用聚集分布的用户偏好权重,当聚集中心个数选取的是 10 个和 20 个时查询效果最好.因为当聚集中心个数选取得少时,能够进行合并计算的权重数量也少,所以查询效果不如聚集中心个数多时好;如果聚集中心个数选取过多,虽然可以在合并计算上大幅度减少计算量,但是最后在进行精确求解时仍然需要一个个地计算,从而使得查询效率降低.所以选择合适的聚集中心个数能够得到很好的查询效率.

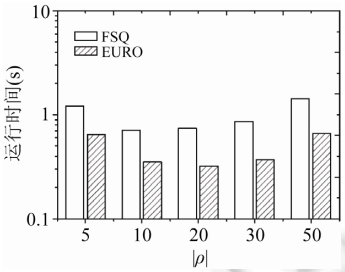


Fig.10 Varying of ρ

图 10 ρ的变化

5.2.6 分组个数 θ 和选取频繁偏好个数 α 对查询性能的影响

图 11 说明了算法 G-MQIR-CLU 当分组个数 θ 变化时,在不同数据集中查询性能的变化.从实验结果中可以看出,当查询时间最小时,θ 取值为 4,最小的分组个数和最大的分组个数都不是最优结果.这说明当分组个数小时,算法不能达到最大程度的剪枝.同样地,当分组个数 θ 增大时,对于每一组都要进行计算,反而增加了查询时间.

下面从图 12 来解释算法 G-MQIR-CLU 在不同数据集中选取频繁偏好个数 α 对查询性能的影响.α 的变化正如查询维度的增加,从实验结果中可以看出,随着 α 的增加,运行时间不断增加.因为 α 变大,用户偏好组的个数 θ^{α} 也随之增大,从而查询计算量也随之增大.另外,从图中可以看出,当 α 变化在 3 和 4 之间时,反而运行时间平稳甚至有略微下降,因为如果频繁偏好个数 α 选取适中,那么保证分组的计算量和剩余用户偏好的计算量就可以达到平衡,同时计算执行时间也随之有所下降.

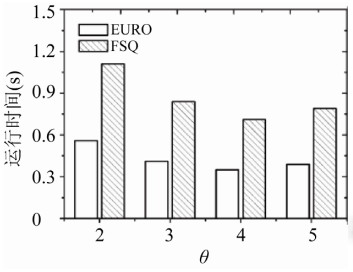


Fig.11 Varying of θ

图 11 θ的变化

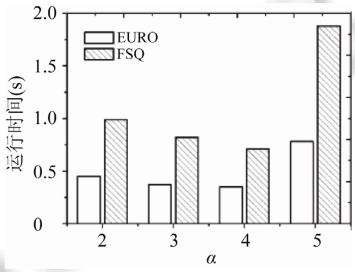


Fig.12 Varying of α

图 12 α的变化

6 结束语

本文研究反向空间偏好 top-k 查询,是从空间对象角度出发并考虑主体对象与周围设施的位置关系的一种新型的反向 top-k 查询.本文提出 3 种剪枝算法来处理反向空间偏好 top-k 查询,即基于 R-tree 的剪枝算法(RT)、批量计算主体对象的剪枝算法(QIR)和改进的批量剪枝算法(MQIR).其中,RT 计算了很多不需要的主体对象与偏好对象的空间距离;QIR 对 RT 进行了优化,减少了不必要的计算;最后,对 QIR 进行改进,提出了 MQIR.另外,

为了更好地减少计算代价,本文对用户提出的偏好权重进行分组,使得拥有相似的用户权重能够同时处理.最后,在真实数据集上给出实验结果,可以证明所提出方法的有效性.

References:

[1] Das G, Gunopulos D, Koudas N, Tsirogiannis D. Answering top-*k* queries using views. In: Proc. of the Int'l Conf. on Very Large Data Bases. 2006. 451−462.

[2] Gunopulos D, Das G, Koudas N, Sarkas N. Ad-Hoc top-*k* query answering for data streams. In: Proc. of the Int'l Conf. on Very Large Data Bases. 2007. 183−194.

[3] Vlachou A, Doulkeridis C, Kotidis Y, Nørvåg K. Reverse top-*k* queries. In: Proc. of the ICDE. 2010. 365−376.

[4] Vlachou A, Doulkeridis C, Nørvåg K, Kotidis Y. Branch-and-Bound algorithm for reverse top-*k* queries. In: Proc. of the ACM SIGMOD. 2013. 481−492. [doi: 10.1145/2463676.2465278]

[5] Chaudhuri S, Gravano L. Evaluating top-*k* selection queries. In: Proc. of the Int'l Conf. on Very Large Data Bases. 1999. 397−410.

[6] Fagin R, Lotem A, Naor M. Optimal aggregation algorithms for middleware. Journal of Computer and System Sciences, 2003,66(4): 614−656. [doi: 10.1016/S0022-0000(03)00026-6]

[7] Tao Y, Hristidis V, Papadias D, Papakonsantinou Y. Branch-and-Bound processing of ranked queries. Information Systems, 2007, 32(3):424−445. [doi: 10.1016/j.is.2005.12.001]

[8] Cong G, Jensen CS, Wu D. Efficient retrieval of the top-*k* most relevant spatial Web objects. Proc. of the VLDB Endowment, 2009, 2(1):337−348. [doi: 10.14778/1687627.1687666]

[9] Hristidis V, Papakonstantinou Y. Algorithms and applications for answering ranked queries using ranked views. VLDB Journal, 2003,13(1):49−70. [doi: 10.1007/s00778-003-0099-8]

[10] Man LY, Dai X, Mamoulis N, Vaitis M. Top-*k* spatial preference queries. In: Proc. of the ICDE. 2007. 1076−1085. [doi: 10.1109/ICDE.2007.368966]

[11] Saranya RS, Saraswathi SME. Ranking spatial data by quality preferences. IEEE Trans. on Knowledge & Data Engineering, 2010, 23(3):433−446. [doi: 10.1109/TKDE.2010.119]

[12] Papadias D, Kalnis P, Zhang J, Tao Y. Efficient OLAP operations in spatial data warehouses. In: Proc. of the SSTD 2001. 2001. 443−459. [doi: 10.1007/3-540-47724-1_23]

[13] Attique M, Qama R, Cho H, Chung T. A new approach to process top-*k* spatial preference queries in a directed road network. In: Proc. of the MobiGIS. 2014. 34−42. [doi: 10.1145/2675316.2675320]

[14] Cho HJ, Kwon SJ, Chung TS. ALPS: An efficient algorithm for top-*k* spatial preference search in road networks. Knowledge & Information Systems, 2013,42(3):599−631. [doi: 10.1007/s10115-013-0696-9]

[15] Korn F, Muthukrishnan S. Influence sets based on reverse nearest neighbor queries. ACM Sigmod Record, 2000,29(2):201−212. [doi: 10.1145/335191.335415]

[16] Lee KCK, Zheng B, Lee WC. Ranked reverse nearest neighbor search. IEEE Trans. on Knowledge & Data Engineering, 2008, 20(7): 894−910. [doi: 10.1109/TKDE.2008.36]

[17] Zhang Z, Jin C, Kang Q. Reverse *k*-ranks query. Proc. of the VLDB Endowment, 2014,7(10):785−796. [doi: 10.14778/2732951.2732952]

[18] Yang C, Lin KI. An index structure for efficient reverse nearest neighbor queries. In: Proc. of the 17th Int'l Conf. on Data Engineering. IEEE, 2001. 485−492. [doi: 10.1109/ICDE.2001.914862]

[19] Stanoi I, Riedewald M, Agrawal D, Abbdadi AE. Discovery of influence sets in frequently updated databases. In: Proc. of the Int'l Conf. on Very Large Data Bases. 2002. 99−108.

[20] Hristidis V, Koudas N, Papakonstantinou Y. PREFER: A system for the efficient execution of multi-parametric ranked queries. ACM Sigmod Record, 2001,30(2):259−270. [doi: 10.1145/376284.375690]

[21] Vlachou A, Doulkeridis C, Kotidis Y, Nørvåg K. Monochromatic and bichromatic reverse top-*k* queries. IEEE Trans. on Knowledge & Data Engineering, 2011,23(8):1215−1229. [doi: 10.1109/TKDE.2011.50]

[22] Vlachou A, Doulkeridis C, Nørvåg K, Kotidis Y. Identifying the most influential data objects with reverse top-*k* queries. Proc. of the VLDB Endowment, 2010,3(1-2):364−372. [doi: 10.14778/1920841.1920890]

[23] Vlachou A, Doulkeridis C, Nørvåg K. Monitoring reverse top-*k* queries over mobile devices. In: Proc. of the 10th ACM Int'l Workshop on Data Engineering for Wireless and Mobile Access. ACM, 2011. 17−24. [doi: 10.1145/1999309.1999313]

[24] Chang YC, Bergman L, Castelli V, Li CS, Lo ML, Smith R. The onion technique: Indexing for linear optimization queries. ACM Sigmod Record, 2000,29(2):391−402. [doi: 10.1145/335191.335433]

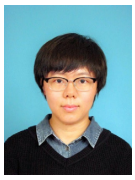
[25] Chester S, Thomo A, Venkatesh S, Whitesides S. Indexing reverse top-*k* queries in two dimensions. Proc. of the DASFAA, 2013(1): 201−208. [doi: 10.1007/978-3-642-37487-6_17]

[26] Ge S, Leong HU, Mamoulis N, Cheung DW. Efficient all top-*k* computation—A unified solution for all top-*k*, reverse top-*k* and top-*m* influential queries. IEEE Trans. on Knowledge & Data Engineering, 2013,25(5):1015−1027. [doi: 10.1109/TKDE.2012.34]

[27] Xin D, Chen C, Han J. Towards robust indexing for ranked queries. In: Proc. of the Int'l Conf. on Very Large Data Bases. 2006. 235−246.

[28] Yu A, Agarwal PK, Yang J. Processing a large number of continuous preference top-*k* queries. In: Proc. of the ACM Sigmod Int'l Conf. on Management of Data. 2012. 397−408. [doi: 10.1145/2213836.2213882]

[29] Borzsony S, Kossmann D, Stocker K. The skyline operator. In: Proc. of the 17th Int'l Conf. on Data Engineering. IEEE, 2001. 421−430. [doi: 10.1109/ICDE.2001.914855]



李淼(1985－),女,辽宁鞍山人,博士,CCF 学生会员,主要研究领域为空间数据库管理.

陈默(1983－),女,博士,讲师,CCF 专业会员,主要研究领域为数据库技术,传感器网络 CPS 数据管理.

谷峪(1981－),男,博士,副教授,博士生导师,CCF 高级会员,主要研究领域为图与空间数据管理.

于戈(1962－),男,博士,教授,博士生导师,CCF 会士,主要研究领域为数据管理理论与技术,分布与并行系统.