doi: 10.3969/j.issn.1005-3026.2015.10.010

面向组近邻的 Top-k 空间偏好查询

陈 默[1], 杨 丹[2], 谷 峪[3], 于 戈[1,3]
(1. 东北大学 计算中心, 辽宁 沈阳 110819; 2. 辽宁科技大学 软件学院, 辽宁 鞍山 114051;
3. 东北大学 信息科学与工程学院, 辽宁 沈阳 110819)

摘 要: 空间偏好查询是当前空间查询研究中的一类热点问题,而现有的空间偏好查询不能有效支持面向组用户的位置服务应用. 为此,提出一类新型空间偏好查询——面向组近邻的 Top-k 空间偏好查询(Top-k spatial preference query for group nearest neighbor). 该查询通过查找特征对象的 λ 子集组近邻最终为用户返回评分值最高的前 k 个 λ 子集. 为了高效执行这一查询,给出了两种查询算法:TSPQ-G 及 TSPQ-G*. 其中 TSPQ-G* 在 TSPQ-G 的基础上,通过空间剪枝及高效的特征对象索引树遍历策略大幅减少 I/O 代价,进而有效提高了该查询的执行效率. 实验采用多个数据集验证了所提算法在不同参数设置下的有效性.

关 键 词: 空间偏好;位置服务;组近邻;剪枝;查询
中图分类号: TP 311.13 **文献标志码**: A **文章编号**: 1005-3026(2015)10-1412-05

Top-k Spatial Preference Query for Group Nearest Neighbor

CHEN Mo[1], *YANG Dan*[2], *GU Yu*[3], *YU Ge*[1,3]
(1. Computing Center, Northeastern University, Shenyang 110819, China; 2. Software College, University of Science and Technology Liaoning, Anshan 114051, China; 3. School of Information Science & Engineering, Northeastern University, Shenyang 110819, China. Corresponding author: YANG Dan, E-mail: asyangdan@163.com)

Abstract: Spatial preference query is a popular focus of the current research on spatial queries. However, the present spatial preference queries cannot be used in the location-based services for group users. To solve this problem, a novel type of spatial preference query, namely, Top-k spatial preference query for group nearest neighbor (TSPG) was proposed, which retrieves the k λ-subsets with the highest score through finding λ-subsets group nearest neighbors of the feature objects. Two algorithms, namely, TSPQ-G and TSPQ-G* were designed for efficient query processing. Based on the TSPQ-G, the TSPQ-G* was developed by performing spatial pruning strategies and efficient traversal strategies of feature objects index, which effectively reduces I/O cost and improves query efficiency. Experimental results on several datasets demonstrated the effectiveness of the proposed algorithms for different setups.

Key words: spatial preference; location-based service; group nearest neighbor; pruning; query

随着地理标签信息的广泛使用,越来越多的 Web 信息系统可通过位置查询为用户提供其感兴趣的搜索结果. 这类位置查询主要局限于简单空间查询,例如搜索给定区域内的对象[1]. 然而,随着用户需求不断扩展,空间偏好查询成为面向位置服务的搜索应用中重要的查询类型. 传统空间查询主要根据对象的空间位置进行查询处理,而空间偏好查询除考虑目标对象的空间位置之外,还要根据目标对象周围的特征对象属性对该对象进行评分,进而搜索满足用户需求的前 k 个目标对象.

已有的空间偏好查询主要基于两类空间约束对目标对象进行评分[2-3],分别是范围约束和最近邻约束. 其中,基于范围约束的空间偏好查询根据给定区域内特征对象的属性对目标对象评分;而基于最近邻约束的空间偏好查询则根据作为最

收稿日期: 2014-09-30
基金项目: 国家自然科学基金资助项目(61402093,61402213); 中央高校基本科研业务费专项资金资助项目(N141604001, N120316001).
作者简介: 陈 默(1983-),女,辽宁沈阳人,东北大学讲师,博士; 于 戈(1962-),男,辽宁大连人,东北大学教授,博士生导师.

近邻的特征对象属性进行评分. 例如,用户在选择旅馆时,同等条件下会将旅馆周围的配套环境也作为参考,有的用户希望旅馆周围有较好的景点和商场(基于范围约束),有的用户则要求景点和商场距离所选旅馆较近(基于最近邻约束). 文献[4－5]将文献[2－3]中问题定义里的单个特征对象集扩展到多个特征对象集,同时定义了三种空间函数:范围关系、最近邻关系及影响区域关系,并以 R 树[6]作为索引基础提出了 SP 等算法. 文献[7]在这些工作的基础上,提出一种基于距离－分值空间映射策略的高效查询处理技术,将目标对象和特征对象组成的对象对从二维笛卡尔坐标空间映射到距离－分值空间. 以上这些查询都基于单查询点的空间查询语义,没有考虑另一类面向多查询点的空间关系——组近邻关系[8-9],不能满足面向组用户偏好的应用需求,需重新设计高效的查询处理方法. 因此,本文提出基于 λ 子集组近邻的 Top－k 空间偏好查询.

1　问题定义

本文提出的查询主要针对组用户的偏好查询应用. 例如,由于旅游旺季剩余客房很少,同去某地旅游的一组用户不能入住同一旅馆,需预定几个旅馆. 如图 1 所示,$O=\{o_1,\cdots,o_6\}$ 表示候选旅馆,商场和景点分别表示为特征对象集 F_1 和特征对象集 F_2,用户从集合 O 中选择 λ 个旅馆($\lambda\leqslant|O|$),且要求距离这组旅馆较近的地方有评价较高的景点和商场. 当$k=1$,$\lambda=3$时,特征对象集

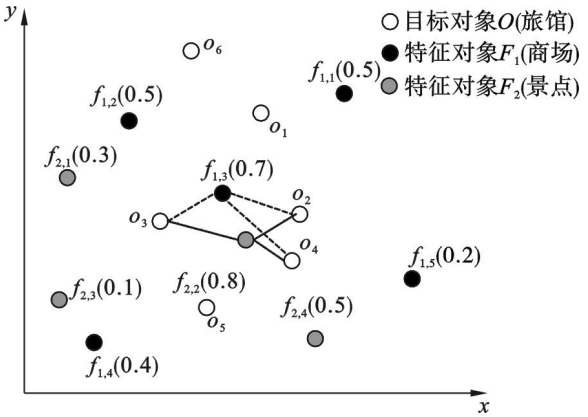

图 1　面向组用户的空间偏好查询示例

Fig. 1　Illustration of top-k spatial preference query for group users

合 F_1 中的 $f_{1,3}$ 以及特征对象集合 F_2 中的 $f_{2,2}$ 到 $\{o_2,o_3,o_4\}$ 的距离和较近且评分较高,则结果集为 $\{o_2,o_3,o_4\}$. 从上述例子可看出,当空间约束从单查询点扩展到多查询点时,查询语义及查询条件与以往的空间偏好查询有很大不同,查询处理过程更为复杂.

定义 1　(Top－k 空间偏好查询). 给定目标对象集 O 以及 m 个特征对象集 $F_1,\cdots,F_m$,Top－k 空间偏好查询返回前 k 个评分最高的目标对象,其中,目标对象的评分取决于与该对象满足给定空间约束的特征对象的评分函数.

定义 2　(λ 子集). 给定目标对象集 O,其 λ 子集 $O^\lambda=\{o_1,\cdots,o_\lambda\}$ ($o_i\in O$),即包含任意 λ 个目标对象的集合.

定义 3　(λ 子集组近邻). 给定 n 个 λ 子集 $O_1^\lambda,\cdots,O_n^\lambda$ 以及特征对象 f ($f\in F$),前 k ($k<n$)个 λ 子集 $O_1^\lambda,\cdots,O_k^\lambda$ 满足如下不等式时,$O_1^\lambda,\cdots,O_k^\lambda$ 定义为 f 的 λ 子集组近邻:

$$\mathrm{Gdist}(O_1^\lambda,f)\geqslant\cdots\geqslant\mathrm{Gdist}(O_k^\lambda,f)\geqslant\cdots\geqslant\mathrm{Gdist}(O_n^\lambda,f)\ . \tag{1}$$

式中,Gdist(,)表示 O^λ 中的每个点到 f 的距离和,称之为组距离,即

$$\mathrm{Gdist}(O^\lambda,f)=\sum_{o_i\in O^\lambda}\mathrm{dist}(o_i,f)\ . \tag{2}$$

其中,dist(,)为两个对象之间的欧式距离.

定义 4　(面向组近邻的 Top－k 空间偏好查询, TSPG). 给定目标对象集 O,m 个特征对象集 $F_1,\cdots,F_m$ 以及参数 λ,基于组近邻的 Top－k 空间偏好查询返回 $\mathrm{SC}(O^\lambda)$ 最高的前 k 个 λ 子集,其中评分函数 $\mathrm{SC}(O^\lambda)$ 的定义为

$$\mathrm{SC}(O^\lambda)=\mathrm{agg}\{\mathrm{SC}_i(O^\lambda)\mid i\in[1,m]\}\ . \tag{3}$$

式中,agg 为单调聚集操作,例如求和,求最小值,求最大值等,本文以下所有示例都以求和为例. $\mathrm{SC}_i(O^\lambda)$ 表示 O^λ 为 F_i 中某特征对象 f 的 λ 子集组近邻时的评分函数,即 $\mathrm{SC}_i(O^\lambda)=r(f)$,$r(f)$ 为对象 f 的评价分值,该分值可由已有的评分数据提供商给出. 例如 f 为某一餐馆,其评价分值可通过大众点评网等第三方消费点评网站的数据获取. 为方便计算,假定 $r(f)$ 的数值在[0,1]内. 对于任意 O^λ,其评分上限 $\mathrm{SC}_{\_u}(O^\lambda)$ 的定义为

$$\mathrm{SC}_{\_u}(O^\lambda)=\mathrm{agg}_{i=1}^{m}\begin{cases}\mathrm{SC}_i(O^\lambda),&\text{当 }\mathrm{SC}_i(O^\lambda)\text{的数值已知;}\\ ,&\text{当 }\mathrm{SC}_i(O^\lambda)\text{的数值未知.}\end{cases} \tag{4}$$

针对上述两类对象集,采用不同的索引结构,对目标对象集采用 R 树索引,同时,为便于剪枝,对特征对象集采用基于最大值的扩展式 R 树索引(AR 树),即在 R 树的每个索引项 e 中添加统计信息 max_er(f),表示 e 为根节点的子树中所

有索引项的 $r(f)$ 最大值.

2 Top – k 空间偏好查询算法

算法 1 给出了 TSPG 处理过程的伪代码. 初始化阶段(第 1 ~2 行), 设置 O 的 R 树根节点的索引项集合 U,存储 top – k 结果的最小堆 R_k 以及 R_k 中最小的评分值 SC^k(即当前第 k 个最高评分). 从 R 树根节点开始,用函数 $CSS(U,\lambda)$ 构建包含 U 的 λ 子集的集合,即 $U^{\#}=\{U_1^\lambda,U_2^\lambda,\cdots,U_n^\lambda\}$(第 3 行)(注:由于 CSS 函数的实现较简单,略去细节). 如果 $U^{\#}$中的子集 U^λ 是非叶节点,对 U^λ 继续计算其 λ 子集,并将 $U^{\#}$更新为 $CSS(U^\lambda,\lambda)$的结果,并重复执行 λ 子集的计算过程(第4 ~ 7 行). 如果 U^λ 是叶子节点(此时 U^λ 等同于 O^λ,为方便理解以下所有算法的描述都采用 U^λ),用 $SNG(U^\lambda,F_i)$计算每个 U^λ 在 F_i 上满足 λ 子集组近邻条件时的评分值 $SC_i(U^\lambda)$(第 8 ~14 行). 其中,首次对目标对象进行评分直接调用 $SNG(U^\lambda,F_i)$,而在以后的评分过程中,需先根据式(4)和上一次评分值计算当前的评分上限,第 13 和 14 行给出增量优化策略,当评分上限$SC_{_u}(U^\lambda)$小于当前第 k 个最高评分 SC^k, 则不必再计算 $SC_i(U^\lambda)$. 如果最终计算得到的 $SC_i(U^\lambda)$大于 SC^k,更新 R_k 及 SC^k(第 15 ~16 行),当 R 树叶子节点被遍历后,返回 R_k(第 17 行).

算法 1 TSPQ – G($O,F_1,\cdots,F_m,\lambda,k$)
输入:目标集 O, 特征集 $F_1,\cdots,F_m$, 参数 λ,k
输出:最优的 k 个查询结果

1. 将集合 U 初始化为 O 的 R 树根节点的索引项集合;
2. $R_k\leftarrow\Phi$; $SC^k\leftarrow 0$;
3. $U^{\#}\leftarrow$CSS(U, λ); //构建包含 U 的 λ 子集的集合
4. **for** $U^{\#}$中的每个子集 U^λ **do**
5. **if** U^λ 是非叶节点 **then**
6. $U^{\#}\leftarrow CSS(U^\lambda,\lambda)$;
7. 跳转到行 4;
8. **else**
9. **if** $i:=1$ **then**
10. $SC_i(U^\lambda)\leftarrow SNG(U^\lambda,F_i)$;
11. **for** $i:=2$ to m **do**
12. 根据式(4)及 $SC_{i-1}(U^\lambda)$计算 $SC_{\_u}(U^\lambda)$;
13. **if** $SC_{\_u}(U^\lambda)>SC^k$ **then** //优化
14. $SC_i(U^\lambda)\leftarrow SNG(U^\lambda,F_i)$;
15. **if** $SC_i(U^\lambda)>SC^k$ **then**
16. 更新 R_k 及 SC^k;
17. 返回 R_k;

$SNG(U^\lambda,F_i)$初始化全局阈值 T_g,该阈值用来表示所有目标对象当前的近邻距离,即 $\sum_{h=1}^{|U^\lambda|}\mathrm{dist}(f_j,u_h)$,设置当前访问过的所有对象点的组近邻距离最小值 Bdist 以及 F_i 中 f 的 λ 子集组近邻 λGNN. 为使连续的 λ 子集组近邻搜索过程尽量在特征集索引树上的遍历路径相似, 首先对 U^λ 中的对象进行 Hilbert 排序,即以各个 u 为顶点做多边形(边数为 λ),找到其质心 c_λ,按照这些质心的 Hilbert 值进行排序. 对于 U^λ 中的每个 u_h 以及 F_i 中的对象 f_j,初始化其欧式距离 dist( ). 当 $T_g<$ Bdist 时,对每个 u_h 计算其下一最近邻,并计算其到最近邻的距离,同时更新 T_g. 当 u_h 的组距离小于 Bdist 时,该点是当前的组近邻点,对全局变量进行更新. 最后将 λ 子集组近邻的评分值返回,由于篇幅所限,该算法伪代码略去.

算法 1 需计算所有目标对象的评分函数值,当目标对象集较大时,该算法计算代价较大,因此, 提出基于剪枝策略的处理算法 TSPQ – G*. 初始化阶段及 U 的 λ 子集计算过程同算法 1. 对于 $U^{\#}$中的叶子节点 λ 子集 U^λ,算法 1 是为每个目标对象依次计算评分,为了减少 I/O 代价, TSPQ – G* 将所有 U^λ 放入集合 W,每次访问特殊目标集时,可通过一次遍历 AR 树,同时将 W 中所有元素的评分值计算出来,即调用 $BNG(W,F_i)$计算每个 U^λ 的 $SC_i(U^\lambda)$值. 当 $SC_i(U^\lambda)$值小于当前第 k 个最高评分 SC^k 时,处理策略同算法 1,最后返回结果集.

算法 2 给出了 TSPQ – G* 中的 BNG 的处理过程. 初始化 Bdist, λGNN 并且设置 N 为 F_i 的索引树根节点(第 1 行). 对于 W 中每个 U^λ 的 u_h,计算 Gmdist (U^λ,N)并初始化列表 L_u(第 2 ~ 3 行). 如果 N 指向的是非叶结点,按 Gmdist(u_h, e)的值由小到大更新 L_u,即依据 Gmdist(u_h,e)的值由小到大将 e 插入 L_u,从 L_u 依次取出 e 并判断是否满足剪枝策略,如果不满足,则继续对列表中的其他元素进行处理,直到满足剪枝策略或者列表为空(第 4 ~12 行). 如果 N 指向的是叶子结点,同样依据 Gmdist(u_h,f)的值由小到大插入

L_u，从 L_u 依次取出 f，如果不满足剪枝条件，则计算 λ 子集组近邻，直到满足剪枝策略或者列表为空（第 13～21 行）. 根据 λ 子集组近邻结果更新对应的分值，并返回结果（第 22～23 行）

算法 2　BNG(W,F_i)

输入：集合 W，特征集 F_i

输出：$SC_i(U^\lambda)$

1. 初始化 Bdist←∞；λGNN←null；N←F_i. root；
2. **for** W 中每个 U^λ 的 u_h **do**
3. 　L_u←{N}；
4. 　**for** N 指向的所有子节点 e **do**
5. 　　**if** e 是非叶节点 **then**
6. 　　按 Gmdist（u_h,e）的值由小到大更新 L_u；
7. 　　**repeat**
8. 　　从 L_u 取下一个 e；
9. 　　**if** $\sum_{h=1}^{|U^\lambda|}$Gmdist(u_h,e) <Bdist **then**
10. 　　N←e；
11. 　　跳转到行 4；
12. 　　**until** $\sum_{h=1}^{|U^\lambda|}$Gmdist(u_h,e) ≥Bdist 或 L_u 为空
13. 　　**else**
14. 　　按 Gmdist（u_h,f）的值由小到大更新 L_u；
15. 　　**repeat**
16. 　　从 L_u 依次取出 f；
17. 　　**if** $\sum_{h=1}^{|U^\lambda|}$Gmdist(u_h,f) < Bdist **then**
18. 　　**if** Gdist(f,U^λ) < Bdist **then**
19. 　　λGNN←f；
20. 　　Bdist←Gdist(f,U^λ)；
21. 　　**until** $\sum_{h=1}^{|U^\lambda|}$Gmdist(u_h,f) ≥Bdist 或 L_u 为空
22. $SC_i(U^\lambda)$←$r(\lambda GNN)$；
23. 返回 $SC_i(U^\lambda)$.

3　实验结果与分析

本文采用多个数据集对所提算法效率进行评测. 实验配置为 Intel Core i3 3.30 GHz 处理器，2 GB内存. 数据集 LA streets（LA）及 Germany utility（GU）来自 R-tree Portal 网站[1]. 其中所有点坐标都归一化到 2D 空间[1，10 000]×[1，10 000]. 由于真实数据集中的特征对象没有已知评分值，采用如下方法生成评分函数 $r(f)$：在地图上随机产生 1 个锚点 a^*，假设距离锚点较近的点其评分值较大，因此，对任意特征对象集 F_i，设置 mind 表示其离 a^* 最近的距离，maxd 表示其离 a^* 最远的距离，则评分函数 $r(f)$ 为

$$r(f)=((\text{mind}-\text{dist}(f,a^*))/(\text{maxd}-\text{mind}))^\sigma.$$

其中，参数 σ 控制分布偏度.

通过在不同的数据集中变化数据集大小及各项参数值考察本文所提算法 TSPQ－G 及TSPQ－G* 的 I/O 性能指标. 为了尽量减少模拟评分函数对实验结果可能带来的影响，以下实验都进行了 10 次并将去掉最大最小值的 8 次结果的平均值作为最终结果. 由图 2 可知，随着 λ 值增大，查询代价逐渐增加，因为 λ 值增大，$U^\#$会明显增加，导致查询代价增加. 图 3 中随着 σ 值增大，两个算法的 I/O 代价都会随之减小，因为 σ 值变大之后，$SC_{\_u}(U^\lambda)$ 分值较低而剪掉的目标对象个数会增加.

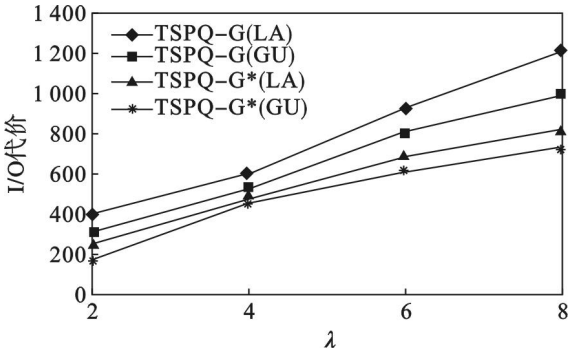

图 2　λ 对算法 I/O 代价的影响

Fig. 2　Effect of λ on I/O cost

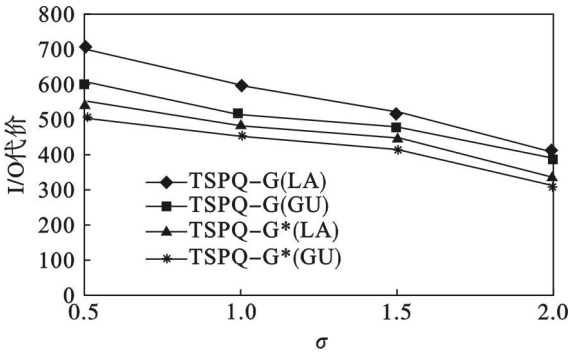

图 3　σ 对 I/O 代价的影响

Fig. 3　Effect of σ on I/O cost

4　结　　论

本文提出了一种新型空间偏好查询——面向组近邻的 Top－k 空间偏好查询，解决了基于组近邻空间约束的偏好查询问题. 本文给出了查询

（下转第 1421 页）