基于道路网的多移动用户动态Skyline查询

周剑刚[1] 秦小麟[1] 张珂珩[2] 许建秋[1]
(南京航空航天大学计算机科学与技术学院 南京 210016)[1] (南瑞集团有限公司 南京 210003)[2]

摘 要 随着无线通信和定位技术的发展,道路网Skyline查询在基于位置的服务等方面越来越重要。但现有的道路网Skyline研究所涉及的空间属性仅考虑距离,并未考虑多个移动用户位置和速度的变化对用户运动时间的影响,当用户运动状态发生变化时,需要动态地调整Skyline结果,进行重新规划。文中分析了用户运动状态与查询间的关联关系,提出了查询处理算法EI,将查询过程分为两步:1)根据时间,通过协同过滤扩展方法确定初始Skyline结果集,并对数据集进行剪枝;2)监测用户的运动状态,一旦用户速度发生变化,就快速根据出入点信息动态调整Skyline集。最后,在真实路网上对算法进行了实验,并将其与现有算法N3S和EDC进行了比较,结果表明EI算法可以高效解决基于道路网的多移动用户动态Skyline查询问题。

关键词 道路网,Skyline查询,运动状态,关联关系
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.09.009

Dynamic Skyline Query for Multiple Mobile Users Based on Road Network

ZHOU Jian-gang[1] QIN Xiao-lin[1] ZHANG Ke-heng[2] XU Jian-qiu[1]
(College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)[1]
(NARI Group Corporation Limited Company, Nanjing 210003, China)[2]

Abstract With the development of wireless communication and positioning technology, the road network Skyline query has become increasingly important in location-based services. However, the spatial attributes involved in the existing road network Skyline research only consider distance, and do not consider the influence of changes in the positions and speeds of multiple mobile users on the user's movement time. When the user's movement state is changed, the Skyline results need to be dynamically adjusted and re-planned. This paper analyzed the incidence relation between the user's motion state and the query, proposed the query processing algorithm EI, and divided the query process into two steps. Firstly, the initial Skyline result set is determined by the collaborative filtering extension method according to time, and the data set is pruned. The user's movement status, as soon as the user's speed changes, quickly adjusts the Skyline set according to the entry point. Finally, the algorithm is tested on the real road network, and is compared with the existing algorithms N3S and EDC. The results show that EI algorithm can efficiently solve the dynamic Skyline query problem of multiple mobile users based on road network.

Keywords Road network, Skyline query, Motion state, Incidence relation

1 引言

Skyline查询最早由Borzonyi等[1]提出,是一种典型的多目标决策问题,用于返回兴趣集中所有不被其他点支配的点的集合。两个数据点之间的支配关系,是指在一个多维数据集中,若存在数据点A,其在每个维度上的属性值都不比数据点B对应维度上的属性值差,并且至少在某一个维度上A的属性值比B的属性值好,则称A支配B。

目前,Skyline查询取得了大量的研究成果,按其所处理数据集属性的不同,可分为如下3类:静态Skyline查询[1-3]、空间Skyline查询[4-5]和道路网Skyline查询[6]。针对不同的应用场景,需要不同的Skyline查询操作来提供高性能和高质量的数据查询,以支撑用户的不同需求。传统的道路网Skyline查询研究主要集中于处理道路网距离之间的支配关系以及数据集剪枝等问题。而实际应用中,用户不仅关心特定数据点之间路程的远近,还会关注到达目的地所花费的时间。Brinkhoff[7]提出在道路网上车速会受诸多因素(如天气、路况、时间段等)的影响而发生变化,其会对用户的行驶时间造

到稿日期:2018-07-10 返修日期:2018-09-15 本文受国家自然科学基金(61373015,61300052,61728204),国家电网公司总部科技资助项目资助。

周剑刚(1993-),男,硕士,CCF会员,主要研究方向为移动对象数据库、分布式数据管理等,E-mail:jiangangzhou@nuaa.edu.cn;**秦小麟**(1953-),男,教授,博士生导师,主要研究方向为空间与时间数据库、分布式数据管理与安全等,E-mail:qinxcs@nuaa.edu.cn(通信作者);**张珂珩**(1973-),男,硕士,主要研究方向为数据库、大数据分析;**许建秋**(1982-),男,博士,副教授,主要研究方向为移动对象数据库、空间数据库技术等。

成极大的影响。因此,需要针对速度和位置的变化情况对 Skyline 集做出动态调整。

考虑以下查询请求:

(1)位于城市中 3 个不同地点的用户 A,B,C 查询一家餐馆聚餐,都开车出发,要求路程花费时间短,评分高;

(2)位于城市中 3 个不同地点的用户 A,B,C 查询一家餐馆聚餐,他们分别开车、骑自行车、步行出发,要求路程花费时间短,评分高。

上述查询给定了查询用户的出行方式和查询要求,例如查询(2)中的出行方式为开车、骑自行车和步行,不同的出行方式有着不同的速度,给定的查询要求为时间短、评分高,既考虑了动态属性时间,又考虑了静态属性评分。查询返回满足要求的餐馆,同时监控用户运动状态以及路网状态,进行实时调整,以满足用户到达时间短的需求。

本文将这类查询定义为基于道路网的多移动用户动态 Skyline 查询,查询返回依据当前用户运动状态和路网信息的满足要求的 Skyline 集合。若在时刻 t 用户的速度由 v 变为 v',或者在规划的最短路径上发生限速,则根据用户速度和位置触发调整 Skyline 集合,以满足用户路程花费时间短的要求。

本文的主要贡献如下:

(1)提出基于道路网的多移动用户动态 Skyline 查询,并给出其查询定义;

(2)提出 EI 算法(Exit and In Algorithm),考虑行驶时间,利用查询间的关联关系减少道路网距离的计算,提高了查询效率;

(3)采用多种真实道路网数据集对算法性能进行评估,通过比较检验了所提算法的有效性。

2 相关工作

随着 GPS 以及各种移动定位设备的普及,基于位置的服务(Location Based Services,LBS)在日常生活中的使用越来越广泛。我们拥有越来越多的包含位置信息的数据,用户可以从网络中查询更多种类的兴趣点(POI),如宾馆、餐厅等。对兴趣点的查询一般包括多个维度,属于数据库领域中典型的多目标查询[8]。

Skyline 查询最初被作为最大矢量(Maximal Vectors)问题[9]来研究,Borzsonyi 等[1]提出了块嵌套环算法(BNL)和分治法(D&C)等基本的 Skyline 计算方法。Chomicki 等[2]在 BNL 算法的基础上提出了排序过滤算法(SFS),其提高了 Skyline 的计算效率。文献[3]提出了分支定界算法(BBS),该算法采用 R 树对数据进行索引,基于最近邻的搜索策略进一步提升了 Skyline 的计算效率。之后,学者们又提出了很多高效计算 Skyline 查询的方法[3,10],这些方法只考虑了非空间信息,如价格、评分等。但是实际生活中存在一种情况,如一群在不同地方的人可能想找一个特定的兴趣点 p 来满足这个群体的需要,此时不能只考虑非空间信息。对此,Sharifzadeh 等[4]提出了空间 Skyline 查询(SSQ)。

空间 Skyline 查询即在原来只考虑非空间信息的 Skyline 查询的基础上加入空间因素的限制。两个点 a 和 b 之间的距离,可以是欧氏距离(无约束 constraint-free),也可以是路网距离(有约束 constraint-based)[6]。无约束 Skyline 查询已经有很多相关的研究工作[3-5,11-14],当前基于欧氏空间的 Skyline 查询工作多是考虑多个查询用户[15],以及移动的单用户查询[12-13]。

Deng 等[6]最先提出基于道路网的空间 Skyline 查询——道路网上的多源 Skyline 查询(Multi-source Skyline Query,MSQ),并提出了 3 种算法:协调扩张算法(Collaborative Expansion,CE)、欧氏距离约束算法(Euclidean Distance Constraint,EDC)以及下界约束算法(Lower Bound Constraint,LBC)。其中,CE 算法根据道路网距离,利用最近邻返回 Skyline 集。Son 等[15]利用曼哈顿距离替代路网距离,提出了曼哈顿空间 Skyline 查询。

针对上述道路网 Skyline 查询不能避免路网距离计算带来的较大计算开销的问题,Safar 等[16]采用最近邻算法对 Skyline 查询进行优化,提出了基于最近邻算法的 Skyline 查询算法 N3S(Network Nearest Neighbor Skyline)。该算法对每一个查询点进行渐进的最近邻计算并找到第一个 Skyline 点,然后计算每个查询点到其他查询点最近邻列表中点的距离,并与已经找到的 Skyline 点进行支配检验。

Jang 等[17]最先提出基于道路网的连续 Skyline 查询(CSSQ),在这种情况下,决定多久更新一次 Skyline 结果尤其重要。其提出,对于每一个兴趣点 p,预先计算其 Skyline 范围 R,当查询点移动至 R 中时,兴趣点 p 即为 Skyline 点。Zheng 等[18]提出基于用户位置的空间查询(LDSQ),考虑用户位置的持续变化。另外,Huang 等[19]考虑到对于距离很远的点,即使它的其他属性较优,用户也不一定感兴趣;对此,他们提出了带有距离阈值的连续 Skyline 查询(Continuous d_ε-Skyline Query,Cd_ε-SQ)和 KNN 连续 Skyline 查询(Continuous k Nearest Neighbor-Skyline Query,CKnn-SQ)。Jiang 等[13]利用动态分割技术在行驶路线上找出每个兴趣点 p 对应的分割点,当用户经过此点时,兴趣点增加为新的 Skyline 点或者被删除。

3 问题描述及定义

基于道路网的多移动用户动态 Skyline 查询是 Skyline 查询在道路网环境下的扩充,该问题涉及 3 个组成对象:兴趣点集、多查询用户以及道路网。本文假定 POI 是兴趣点的集合,每一个兴趣点除了位置坐标,还具有 m 维静态属性,如宾馆价格、星级等。$p[i]$表示兴趣点 p 的第 i 维属性。$Q=\{q_1,q_2,\cdots,q_n\}$是查询用户的集合,每一个查询用户 q_i 有位置和速度属性,速度可能随时间而发生变化,同时其具有多维动态属性,是查询用户 q_i 到每一个兴趣点的道路网时间,用 $t(p,q_i)$ 表示。

首先给出基于道路网的多移动用户动态 Skyline 查询的定义。

定义 1(静态支配) 给定一个查询点,以及兴趣点 p_1 和 p_2,当且仅当 $p_1[i]\leqslant p_2[i](\forall i,1\leqslant i\leqslant n)$,并且 $p_1[j]<p_2[j](\forall j,1\leqslant j\leqslant n)$时,称 p_1 静态支配 p_2,记为 $p_1<_s p_2$。

定义 2(道路网时间) 对于任意查询用户 q、q 的移动速度 v 和兴趣点 p,$t(p,q)$表示查询用户 q 到兴趣点 p 的时间。$t(p,q)$受以下因素的影响:

(1)查询用户 q 的位置 L 和速度 v；

(2)道路网状态的变化。

其中，道路网的每条路段都具有最大限速信息。

定义 3(道路网支配)　对于任意查询用户 q、q 的移动速度 v 和兴趣点 p_1 和 p_2，当且仅当满足以下两个条件时，称 p_2 道路网支配 p_1(下文简称“支配”)，记为 $p_2 <_q p_1$。

(1) $p_2 <_s p_1$；

(2) $t(p_2,q) < t(p_1,q), \forall q \in Q$。

定义 4(基于道路网的多移动用户动态 Skyline 查询)　给定一个查询用户集 $Q=\{q_1,q_2,\cdots,q_n\}$、Q 的移动速度 $V(Q)$ 和一个兴趣点集 POI，每次用户速度或路网状态发生变化时，都更新 Skyline 查询结果，返回 POI 的一个子集，其中的每一个对象都不能被 POI 中的任何其他兴趣点道路网支配。

表 1 列出了一个宾馆兴趣点分布在道路网上的查询实例，$p_1,p_2,\cdots,p_{10}$ 是 10 个具有 3 维静态属性的兴趣点。表 2 列出了 3 个查询用户的速度变化情况的查询实例，此处假设用户要经过的路线限速未发生变化。图 1 给出了兴趣点和查询用户在道路网上的位置分布。基于道路网的多移动用户通过动态 Skyline 查询查找价格低、星级高、评价好且用户到达花费时间短的宾馆。对于不同的查询用户，速度一旦发生变化，其到达宾馆花费的时间也会发生变化，因此 Skyline 查询需要更新 Skyline 结果集，以更准确地找出满足要求且到达该宾馆花费时间更短的兴趣点。

表 1　兴趣点集的非空间属性

Table 1　Non-spatial attributes of POI

宾馆	价格	星级	评价
p_1	195	3	3.5
p_2	175	4	4
p_3	180	3	4
p_4	220	2	4.5
p_5	175	2	3.5
p_6	190	4	3
p_7	160	3	3
p_8	210	4	2.5
p_9	185	2	3.5
p_{10}	180	3	4.5

表 2　用户速度

Table 2　User's speed

(单位：km/h)

查询用户	初始速度	6 min	10 min	…
q_1	32	52	25	…
q_2	54	52	52	…
q_3	40	32	40	…

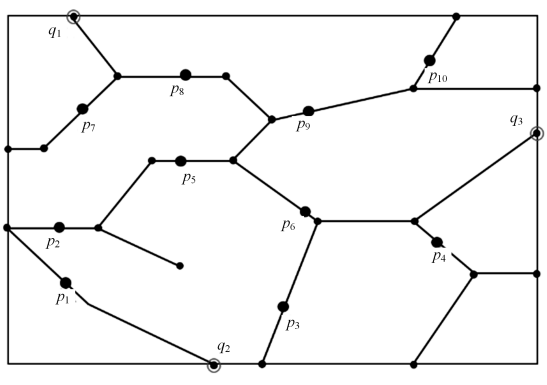


图 1　兴趣点集和查询用户在道路网上的分布

Fig. 1　Distribution of POI and query users on road networks

4　基于出入点的 EI 算法

本节针对基于道路网的多移动用户动态 Skyline 查询，提出 EI 算法，其主要目的是找出 Skyline 点变化与查询用户运动过程的关系，并减少道路网距离计算量。如图 2 所示，ij 是查询用户 q_1 的运动路径，q_1 从 i 向 j 运动。

图 2　运动示例

Fig. 2　Motion example

定义 5(出点)　给定一个查询用户 q、Skyline 兴趣点 p_1、查询用户当前运动路径 ij 上的点 E_{p1}、用户预计运动时间 T，当且仅当：

(1) $t(o,q) > T(\forall o \in E_{p1}j)$

(2) $t(o,q) < T(\forall o \in iE_{p1})$

时，称 E_{p1} 是 p_1 相对于查询用户 q 在当前运动路线上的出点。

定义 6(入点)　给定一个查询用户 q、兴趣点 p_2、查询用户当前运动路径 ij 上的点 I_{p2}、用户预计运动时间 T，当且仅当：

(1) $t(o,q) < T(\forall o \in I_{p2}j)$

(2) $t(o,q) > T(\forall o \in iI_{p2})$

时，称 I_{p2} 是 p_2 相对于于查询用户 q 在当前运动路线上的入点。

对于每一个查询用户，找出候选集中的减小集 CSP_{ij}^{dw} 和增长集 CSP_{ij}^{up}。减小集是指按查询用户当前路线运动到兴趣点，时间减少的兴趣点；增长集是指按查询用户当前路线运动到兴趣点，时间增长的兴趣点。对于增长集中的点，在路线上找出其出点 E_p；对于减小集中的点，在路线上找出其入点 I_p。

定理 1　若某 Skyline 兴趣点对于每一个查询用户都是属于减小集 CSP_{ij}^{dw}，那么随着查询用户的移动，原来是 Skyline 点的兴趣点仍然是 Skyline 兴趣点。若某非 Skyline 兴趣点对于每一个查询用户都是属于增长集 CSP_{ij}^{up}，那么随着查询用户的移动，原来的非 Skyline 点的兴趣点仍然不会是 Skyline 兴趣点。

证明：假设 p_1 属于 Skyline 点，并且其对于每一个查询用户都是属于减小集 CSP_{ij}^{dw}，那么减小集中的每一个兴趣点缩短的时间都是相同的，则在减小集中不会出现其他兴趣点支配 p_1；对于增长集中的兴趣点，它们的时间同时在增长，也不可能支配 p_1。假设 p_2 不属于 Skyline 点，并且其对于每一个查询用户都是属于增长集 CSP_{ij}^{up}，那么增长集中的每一个兴趣点增长的时间都是相同的，因此 p_2 不会在增长集中支配任何兴趣点；同样，对于减小集中的兴趣点，它们的时间都在缩短，p_2 无法支配任何点。因此，p_2 仍然是非 Skyline 点。

定理 2　在路线上的同一点，若车速 v 增大，那么原来的 Skyline 点仍然是 Skyline 点；若车速 v 减小，则须根据 T 将超出运动时间 T 的 Skyline 点剔除。其中，T 为用户到达目的地的预估时间。

证明：假设 p_1 是 Skyline 点，v 增大，那么原来的 Skyline

点的运动时间仍在 T 内，同时所有兴趣点对应的该查询用户的运动时间会相应缩短，因此 p_1 仍不会被任何其他兴趣点支配。若车速 v 减小，对于距离过远的 Skyline 点，其不符合运动时间短的要求，对其进行剪枝。

在利用 EI 算法动态更新 Skyline 结果前，须在用户发出查询的初始位置找到初始 Skyline 结果集，并根据时间，利用协同过滤方法（Collaborative Expansion according Time，CET）快速找出在初始位置的 Skyline 结果反馈给用户。该过程分为以下 3 步。

(1)同时以每一个查询用户为中心，按当前道路网每条路线所能允许的最大运动速度得到的行驶时间沿道路网向外扩展，直至遇到第一个同时到达的点时终止。此时，将所有的已经被访问到的兴趣点加入候选集 *Candidate*，未被访问到的兴趣点加入到候选集 *BpCandidate*，并且第一个被访问到的兴趣点一定为 Skyline 点，将其加入 Skyline 集合。

(2)仅对候选集 *Candidate* 中的点进行继续扩展，扩展到的共同到达点若是候选集 *Candidate* 中的，则与已经找到的 Skyline 点进行比较，若不被支配，则加入 Skyline 集合，否则剔除。对于第二次或之后的共同到达点之外的点，同样从候选集 *Candidate* 中剔除，并将其加入到 *BpCandidate* 中。

(3)从候选集中剔除的和未被访问到的点在动态属性上一定是被当前 Skyline 点集中的点支配的，因此比较 *BpCandidate* 中的点与 Skyline 点集中的点的静态属性，若 *BpCandidate* 中的点的静态属性未被静态支配，则将该点加入 Skyline 集合。此时得到的 Skyline 点集即为初始 Skyline 集合，接下来即可进行动态的 Skyline 查询。

EI 算法如算法 1 所示。

算法 1 EI 算法

输入：原始数据集 POI，查询用户位置 Loc(Q)，速度 V(Q)

输出：Skyline 集合 S_{poi}

1. set S_{poi} be the result of CET
2. T←MaxTime(S_{poi}, V(Q))
   //将兴趣集中质量过差和运动时间大于 T 的点剔除
3. InitialPOI ← Filter(POI, T)
   //对每一个查询点计算增长集、减小集以及出入点
4. for each q in Q
5. for each p in InitialPOI
6. if t(p,q) decrease
7. Add p to CSP_q^{dw}, I_p = CalculateEI(p,q)
8. if t(p,q) increase
9. Add p to CSP_q^{up}, E_p = CalculateEI(p,q)
10. if V(q) change
    //根据变化后的速度调整 E_p 和 I_p 的位置
11. for each p in InitialPOI
12. update E_p, I_p according V(q)
    //根据出入点位置调整兴趣集
13. update InitialPOI accordingE_p, I_p
    //根据出入点位置更新 Skyline 集合
14. if V(q) increase S_{poi} remain unchanged
15. if V(q) decrease
16. for each p in InitialPOI
17. S_{poi} = Update(CSP_Q^{dw}, CSP_Q^{up}, p)
18. return S_{poi}

算法输入原始数据集 *POI*、每一个查询用户的位置 *Loc*(*Q*)、运动速度 *V*(*Q*)、当前预测运动时间 *T*（*T* 根据用户初始目的地所需的运动时间获得）。算法前两行将兴趣集中质量过差以及到达它所需时间过长的兴趣点剔除，并利用 CET 方法获得初始 Skyline 集；第 3－8 行计算每一个兴趣点对每一个查询用户属于增长集还是减小集，并计算出其对应的入点和出点；第 9－11 行根据速度的变化情况动态调整出入点的位置，并根据新的出入点位置更新兴趣集；第 12－16 行根据定理 1 和定理 2 更新 Skyline 结果集；最后将 Skyline 结果反馈给用户进行选择。

对 EI 算法更新 Skyline 集合过程的时间复杂度进行分析。假设发出查询请求时有 n 个查询用户，兴趣点的静态属性有 m 维，总共有 E 个出点和 I 个入点。

首先，若查询用户的速度发生变化，则需要根据最新的速度调整每一个出点和入点的位置，时间复杂度为 $O(E+I)$；然后，将出入点与查询用户的位置进行比较，对候选集中的兴趣点进行更新，时间复杂度为 $O(E+I)$；最后，将候选集中的每一个兴趣点与已有的 Skyline 点进行比较，找出候选集中的 Skyline 点，其时间复杂度为 $O(\max(E,I)*(n+m))$。因此，EI 算法的整体时间复杂度为 $O(E+I)+O(E+I)+O(\max(E,I)*(n+m))=O(2E+2I+\max(E,I)*(n+m))$。

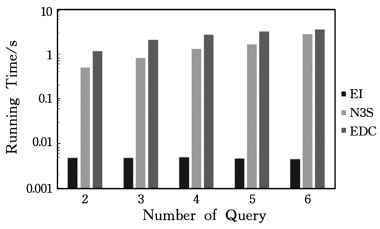
5 实验

本节通过实验来验证本文所提方法的有效性。实验中的道路网数据采用：San Francisco 城市的道路数据[7]，共 221415 条道路信息（175343 个道路交叉点）；Berlin 城市的道路数据，共 11440 条道路信息（8742 个道路交叉点）；Oldenburg 城市的道路数据，共 7035 条道路信息（6105 个道路交叉点）。兴趣点集则是在道路网上随机生成的模拟数据，其非空间属性值依据数据维度间的关系分为 3 类[1]：正相关数据集、独立数据集、负相关数据集。因道路网上的查询兴趣点属性包括路网距离以及查询用户速度，非空间属性值之间的关系的影响并不明显，因此本实验仅利用文献[1]生成的负相关数据集进行验证。查询用户的车速信息是利用 Thomas Brinkhoff 基于路网的移动对象生成器[7,20]生成的。

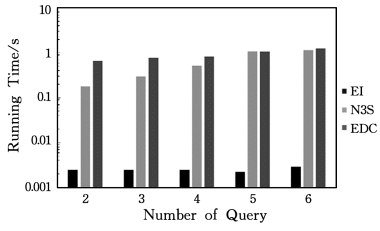
本文的实验环境为：3.40 GHz CPU，4.0 GB 内存，Window 7 操作系统。

5.1 查询用户数量对花费时间的影响

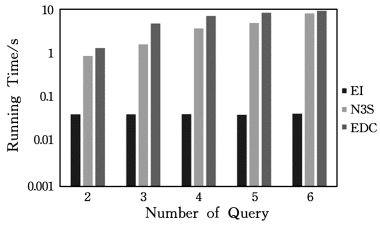
图 3 给出了查询用户数量对各算法查询所花费时间的影响。从图 3 可以看出，在每一个道路网上，EI 算法所花费的时间基本不受查询用户数量的影响，而 N3S 算法和 EDC 算法的查询时间都随着查询用户的增多而变大，且其时间花费远远多于 EI 算法；同时，当路网规模增大时，EI 算法所花费的时间会增加。可见，在查询用户相同时，EI 算法单次查询花费的时间远短于 N3S 和 EDC 算法所花费的时间。



(a)Berlin 路网上查询用户数量对查询时间的影响



(b)Oldenburg 路网上查询用户数量对查询时间的影响



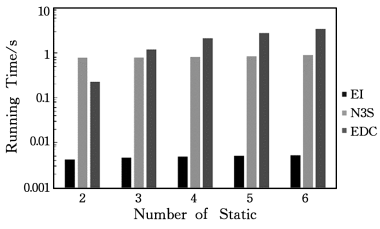
(c)Sanfrancisco 路网上查询用户数量对查询时间的影响

图 3　查询用户数量对查询时间的影响

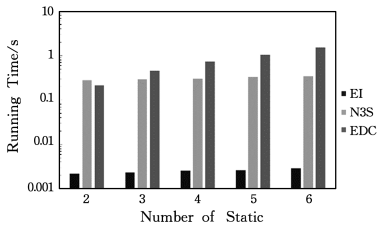
Fig. 3　Influence of number of query users on querying time

5.2　POI 非空间属性维度对花费时间的影响

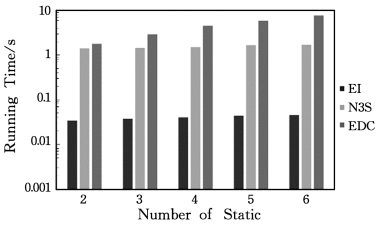
图 4 给出了 POI 非空间属性维度对各算法查询所花费时间的影响。



(a)Berlin 路网上 POI 属性维度对查询时间的影响



(b)Oldenburg 路网上 POI 属性维度对查询时间的影响



(c)Sanfrancisco 路网上 POI 属性维度对查询时间的影响

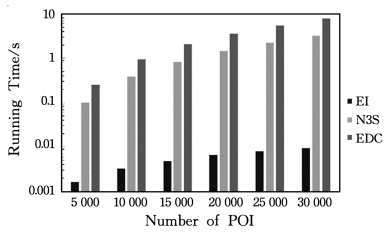
图 4　POI 属性维度对查询时间的影响

Fig. 4　Influence of attribute dimension of POI on querying time

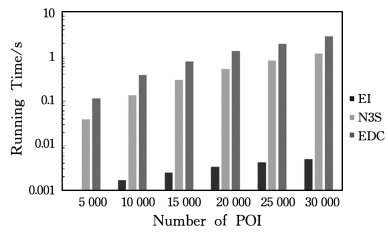
实验测试了 POI 的非空间属性从 2 维增加到 6 维的情况。由图 4 可以看出，随着兴趣点属性维度的增加，3 种算法的查询时间都有一定的增加，并且随着路网规模的变大，算法所花费的时间都会相应增加；同时，在 3 种不同的路网上，EI 算法单次查询所花费的时间远短于 N3S 和 EDC 算法的时间。

5.3　POI 数量对花费时间的影响

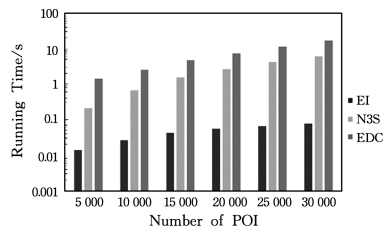
如图 5 所示，兴趣点集 POI 的数量根据路网规模取了不同的值，在 Berlin 路网上兴趣点的数量从 5000 变化到30000，Oldenburg 路网上兴趣点的数量从 3000 变化到 18000，San Francisco 路网上兴趣点的数量从 7000 变化到 40000。由图 5 可以看出，随着兴趣点数量的增加，EI 算法和另外两种算法的查询时间都在增加，并且随着路网规模的变大，算法花费的时间都会相应增加；同时，在 3 种不同的路网上，EI 算法单次查询所花费的时间远短于 N3S 和 EDC 算法的时间。



(a)Berlin 路网上 POI 数量对查询时间的影响



(b)Oldenburg 路网上 POI 数量对查询时间的影响



(c)Sanfrancisco 路网上 POI 数量对查询时间的影响

图 5　POI 数量对查询时间的影响

Fig. 5　Influence of number of POI on querying time

总体而言，EI 算法所花费的时间远短于 N3S 算法和 EDC 算法，这主要是由于当查询用户的位置和速度发生变化时，须动态调整 Skyline 集合，EI 算法可以利用多次查询间的关联关系减少路网距离的计算量，而 EDC 和 N3S 算法都需要再次重新计算。同时，算法在正相关、独立以及负相关数据集上并未体现出明显的相关特性，这主要是因为其还被路网距离这一动态属性所影响。

结束语　本文针对道路网环境下，多用户移动速度和位置的动态变化对 Skyline 点选取结果所产生影响的问题，研究了基于道路网的多移动用户的动态 Skyline 查询，提出了 EI

算法。所提算法有效利用了多次查询间的关联关系,且支持用户偏好的动态交互。实验证明 EI 算法可有效缩短查询时间。

参 考 文 献

[1] BORZSONYI S,STOCKER K,KOSSMANN D. The Skyline Operator[C]// Proceedings 17th International Conference on Data Engineering. 2001:421-430.

[2] CHOMICKI J,GODFREY P,GRYZ J,et al. Skyline with presorting[C]// International Conference on Data Engineering. IEEE,2004.

[3] PAPADIAS D. An Optimal and Progressive Algorithm for Skyline Queries[C]// Acm Sigmod International Conference on Management of Data. ACM,2003.

[4] SHARIFZADEH M,SHAHABI C. The Spatial Skyline Queries[C]// International Conference on Very Large Data Bases. DBLP,2006.

[5] SHARIFZADEH M,SHAHABI C,KAZEMI L. Processing spatial Skyline queries in both vector spaces and spatial network databases[J]. Acm Transactions on Database Systems, 2009, 34(3):1-45.

[6] DENG K,ZHOU X,SHEN H T. Multi-source Skyline Query Processing in Road Networks[C]// IEEE 23rd International Conference on Data Engineering, 2007 (ICDE 2007). IEEE, 2007.

[7] BRINKHOFF T. A Framework for Generating Network-Based Moving Objects[J]. Geoinformatica,2002,6(2):153-180.

[8] ENDRES M. A Survey on Selectivity Estimation for Preference Database Queries[J]. Databases and Information Systems, 2014,270(8):159-172.

[9] KUNG H T,LUCCIO F,PREPARATA F P. On Finding the Maxima of a Set of Vectors[J]. Journal of the Association for Computing Machinery,1975,22(4):469-476.

[10] WANG Y,SHI Z,WANG J,et al. Skyline Preference Query Based on Massive and Incomplete Dataset[J]. IEEE Access, 2017,5(99):3183-3192.

[11] LIN X,YUAN Y,WANG W,et al. Stabbing the sky:efficient skyline computation over sliding windows[C]// International Conference on Data Engineering. IEEE,2005.

[12] CHEEMA M,LIN X,ZHANG W,et al. A safe zone based approach for monitoring moving skyline queries[C]// International Conference on Extending Database Technology. 2013.

[13] JIANG S,ZHENG J,CHEN J,et al. Efficient Computation of Continuous Range Skyline Queries in Road Networks[M]// Intelligent Computing Methodologies. Springer International Publishing,2016:520-532.

[14] GENG M,AREFIN M S,MORIMOTO Y. A Spatial Skyline Query for a Group of Users Having Different Positions[J]. Journal of Software,2014,9(11):137-142.

[15] SON W,HWANG S W,AHN H K. MSSQ:Manhattan spatial skyline queries[C]// International Symposium on Spatial & Temporal Databases. Springer,Berlin,Heidelberg,2011.

[16] SAFAR M,EL-AMIN D,TANIAR D. Optimized Skyline queries on road networks using nearest neighbors[J]. Personal & Ubiquitous Computing,2011,15(8):845-856.

[17] JANG S,YOO J. Processing Continuous Skyline Queries in Road Networks[C]// International Symposium on Computer Science and ITS Applications. IEEE,2008:353-356.

[18] ZHENG B,LEE K C K,LEE W C. Location-Dependent Skyline Query[J]. Mdm,2008:148-155.

[19] HUANG Y K,CHANG C H,LEE C. Continuous distance-based Skyline queries in road networks[J]. Information Systems, 2012,37(7):611-633.

[20] BRINKOFF T. Generating Traffic Data[J]. Bulletin of the Technical Committee on Data Engineering IEEE Computer Society,2003,26:2003.