计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.181102231

时间依赖路网上的移动对象 K 近邻查询算法

张 彤 秦小麟
南京航空航天大学计算机科学与技术学院 南京 210016
(15651653679@163.com)

摘 要 随着基于位置服务的广泛应用,时间依赖路网上的对象查询逐渐成为研究热点。以往研究大多只针对时间依赖路网上的静态对象(如加油站、餐厅等),未考虑到移动对象(如出租车)的情况,而移动对象的查询在日常生活中有着非常广泛的应用场景。因此,文中提出了一种针对时间依赖路网上的移动对象 K 近邻查询算法 TD-MOKNN,该算法分为预处理阶段和查询阶段。在预处理阶段,通过建立路网和网格索引,提出了一种新的移动对象到路网的映射方法,解除了以往研究假设移动对象恰好在路网顶点上的限制;在查询阶段,采用启发式搜索,借助倒排网格索引计算了一种新的高效启发值,通过预处理信息和启发值设计了高效 K 近邻查询算法,并给出了算法的正确性证明和时间复杂度分析。实验验证了所提算法的有效性,相比现有算法,TD-MOKNN 算法在遍历顶点数和响应时间上分别减少了 55.91%和 54.57%,查询效率平均提升了 55.2%。

关键词:K 近邻查询;移动对象;时间依赖路网;A* 算法;网格索引
中图法分类号 TP311

K Nearest Neighbors Queries of Moving Objects in Time-dependent Road Networks

ZHANG Tong and 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 services, object-based query on time-dependent road network has gradually become a research hotspot. In the past, most of the researches only focused on static objects on time-dependent road networks (such as gas stations, restaurants, etc.), and did not take into account the situation of mobile objects (such as taxis). The query of mobile objects has a very wide range of applications in daily life. Therefore, the K nearest neighbor query algorithm TD-MOKNN of moving object is proposed for time-dependent road network. The algorithm is divided into pre-processing stage and query stage. In the pre-processing stage, the road network and grid index are established, and a new mapping method of moving objects to the road network is proposed, which removes the limitation of previous researches that moving objects happen to be on the intersection of the road networks. In the query stage, a new efficient heuristic value is calculated by using inverted grid index, and an efficient k-nearest neighbor query algorithm is designed by using pre-processing information and heuristic value. Experiments verify the effectiveness of the algorithm. Compared with existing algorithm, TD_MOKNN algorithm reduces the number of traversing vertices and response time by 55.91% and 54.57% respectively, and improves the query efficiency by 55.2% on average.

Keywords K nearest neighbors query, Moving object, Time-dependent road network, A* algorithm, Grid index

1 引言

随着移动终端的迅猛发展,基于位置的服务(Location Based Services,LBS)成为了路网中的重要应用。在 LBS 中,用户可能会搜索自己感兴趣的地理点(Point of Interest,POI)(如餐厅、加油站等),并计算到这些地理点的路线和所需时间。而实际上,交通时间在很大程度上取决于交通流量,即道路的交通状况会随着时间的变化而发生改变。图 1 展示了南京市洪武路某天中汽油车的通行数量[1]。

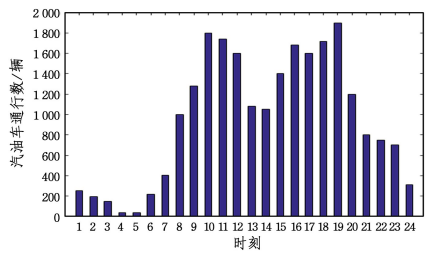


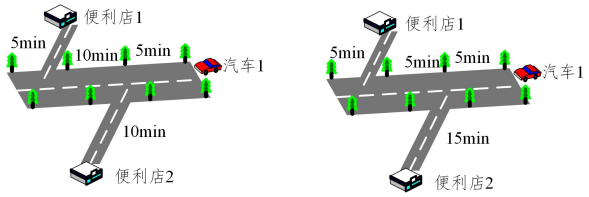
图 1 洪武路一天中不同时段汽油车的通行数量

Fig. 1 Number of petrol vehicles passing through Hongwu Road at different times of the day

到稿日期:2018-11-30 返修日期:2019-04-23 本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家自然科学基金(61373015,61728204)
This work was supported by the National Natural Science Foundation of China (61373015,61728204).
通信作者:秦小麟(qinxl@nuaa.edu.cn)

从图中可以发现:1)白天流量高于夜间流量;2)日流量呈双峰分布,第一个峰值出现在中午 12:00 左右,第二个峰值出现在晚上 20:00 点左右。在车辆通行的高峰时期,路网可能会形成堵塞,道路的通行时间会增加;而在凌晨等时间段车辆稀少,道路的通行时间会相应减少。因此,道路的通行时间不是固定不变的,而是一个关于时间的函数,以往的静态路网研究并没有考虑该问题。

路网通行时间的变化给计算最短路径和估计路线时间带来了巨大的挑战,但是考虑交通状况对提高基于位置的服务的质量至关重要,且具有更高的实际参考价值。在这种情况下,我们将路网建模为时间依赖路网。与静态路网模型不同,在时间依赖路网中,道路的权值(通过这条路的时间)随一天中时间的变化而变化。因此,在寻找 K 近邻等查询问题中,查询结果取决于查询发起的时间。查询发起的时间不同,即使是相同的查询,查询结果也可能不同。图 2 展示了一个在时间依赖路网中查询最近邻的例子,假设汽车 1 想要寻找可以最快到达的便利店。若在上午 8 时发起请求,即时路网如图 2(a)所示,返回结果为便利店 2;若在上午 11 时发起请求,即时路网如图 2(b)所示,返回结果为便利店 1。从这个例子中可以发现,查询发起的时间不同会导致返回的结果集不同。因此,在静态路网中的 K 近邻查询算法[2-3]不适用于时间依赖路网。



(a)上午 8 时的路网图 (b)上午 11 时的路网图

图 2 时间依赖路网上的最近邻查询

Fig. 2 Nearest neighbor query on time-dependent road networks

为了解决 K 近邻算法不适用于静态路网的问题,本文设计了时间依赖路网上的 K 近邻查询算法。时间依赖路网中的 K 近邻查询,是指用户在某一时刻发起查询,查找满足条件且能最快到达的 K 个对象。大多数现有工作只是针对静态兴趣点(如餐厅、加油站等),未考虑移动兴趣点的情况,但查询移动兴趣点 K 近邻在日常生活中有非常大的需求量。例如,在图 3 所示的场景中,一位用户在路口想要寻找一辆能最快到达自己的出租车,图中有两辆出租车可用,出租车公司需要计算哪辆车到该用户的最短路径所需时间更少,遂派单给该车辆。

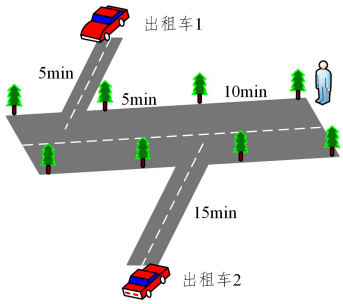


图 3 时间依赖路网下的移动对象最近邻查询

Fig. 3 Nearest neighbor query of moving objects on time-dependent road networks

从这个例子中可以发现,与静态对象 K 近邻查询算法 TD-KNN 不同,本文提出的 TD-MOKNN 算法旨在查询时间依赖路网下的移动对象 K 近邻。

TD-MOKNN 算法试图通过计算从兴趣点(出租车)到查询点(用户)的通行时间,来返回可以最快到达查询点的 K 个移动对象。本文的主要贡献如下:

(1)针对以往研究假设移动对象恰好在路网顶点上的不足,提出了一种新的移动对象映射方法,使得最终结果更加接近真实值;

(2)设计了合适的网格索引结构来管理移动对象,并辅助计算启发值;

(3)采用增量式网络扩展[3]和 A* 算法[4]来确保有效的剪枝和更快的响应速度,并在提出的索引结构基础上设计选取了合理的启发值来指导路网的高效扩展。

本文第 2 节介绍了相关工作;第 3 节介绍了研究过程中需要的基础知识,包括时间依赖路网 K 近邻和相关索引;第 4 节介绍了时间依赖路网上的移动对象 K 近邻查询算法和索引结构;第 5 节对 TD-MOKNN 算法进行了实验评估,并将其与现有算法进行了对比分析;最后总结全文并展望未来。

2 相关工作

静态路网上的最短路径查询和 K 近邻查询已有大量相关研究成果[5-7]。Dijkstra 算法[8]和 A* 算法[4]是两种典型的求解路网中最短路径的方法。Papadias 等[3]介绍了路网中的 KNN 查询问题,并提出了 IER 和 INE 两个框架 。IER 假设网络上两点之间的欧氏距离小于网络距离,先在欧氏空间中计算结果,再使用路网距离将其细化。INE 是对 dijkstra 算法[8]的改编,优于 IER。该方法从查询点 q 开始,按照离 q 从近到远的顺序访问每一个方向上 q 可到达的所有节点,直到找到 K 个最近的数据对象。但当对象在路网上的分布不是很密集时,上述方法的查询效率较低。鉴于此,Kolahdouzan 等提出了一种基于预计算网络 voronoi 多边形[9](NVP)的方法 Voronoi-based approach(VN3)[10],并使用空间访问方法来索引 NVP,使用 NVP 可以使查询点立即找到第一个最近邻,并且降低了 KNN 搜索的在线开销;但该方法不能直接应用到时间依赖路网 K 近邻查询问题上。Huang 等使用“岛”算法[11]来解决路网 KNN 问题,该方法通过预计算的方式,存储到数据点的距离小于 r 的对象。这种方式结合了预计算和网络扩展的优点,但 r 值的选取成为了关键点和难点,不合适的 r 值将大大降低算法的效率。以上研究只适用于静态路网,不能直接应用到边权值不断变化的网络环境中。

除了静态路网,近些年时间依赖路网查询也逐渐成为了热点。时间依赖路网上的最短路径问题与时间依赖路网 KNN 问题有着本质联系,因为最近的兴趣点就是与查询点之间最短路径代价最小的 POI。Cooke 等[12]首次提出了时间依赖路网下的最短路径解决方案,采用了一种改进的 Bellman 迭代方案来寻找路网中任意两个顶点之间的最短路径。文献[13]提出了一个时间聚合图,它将路网中每条边在时间点上的旅行时间聚合成一个时间序列,降低了每个时间窗口计算通过静态路网中权重最小路径的开销。

Demiryurek 等[14]首次提出了时间依赖路网上的 KNN

查询(TD-KNN)，该查询的目标是找到从查询点开始，能在最短时间到达的前 K 个 POI。其设计了两种方法来解决这个问题。第一种方法使用时间扩展图来建模路网，它允许使用静态路网中的解决方案来完成 TD-KNN 查询；但这种方法存在较多缺点，例如存储开销大、响应速度慢且结果可能不正确。第二种方法基于 INE 算法，它从查询点开始扩展搜索，直到找到 K 个最近邻；但该方法由于在每一个方向上都扩展路网，因此属于“盲目”搜索，增加了访问的顶点数和响应时间。作为对该算法的改进，文献[15]提出了一种基于 INE 扩展，并使用 A* 算法来指导扩展的算法。A* 算法有目的地扩展路网，因此它比上述方法访问了更少的顶点，作者在该方法中尽量丢弃那些接近查询点但远离任何 POI 的顶点，并通过旅行时间的上限进行剪枝；但该方法用全天最小值作为启发值，会产生较大的误差。针对这个问题，Komai 等提出了优化算法 TD-FTT[16]，它在每个小时间片段上建立索引，但索引频率的提高也导致了创建索引代价的增加。以上查询均为对静态对象的查询，不能查询移动对象。还有一类查询 TDNS (Time-Dependent Nearest Server)[17] 针对动态对象，是 TD-KNN 查询的一种变种，它查找的是能最快到达查询发起人的服务提供者(如出租车、救护车和消防车等)。文献[17]提出了两种算法，一种是朴素算法，该算法先找到路网上所有的兴趣点，利用迪杰斯特拉算法计算出每一个兴趣点到查询点之间的最短路径，并根据最短路径计算出所需时间，再从中选出所需时间最短的兴趣点作为结果并返回。该方法对所有的兴趣点都进行了计算，没有进行剪枝等相关工作，且时间上的最近邻对象往往离查询点 q 很近，因此其效率不高。另一种算法采用了 A* 方法，其相比朴素算法有所改进，但启发值的计算效率低下，导致在计算阶段耗时较多。针对这个缺点，本研究设计了启发值计算效率更高的算法。

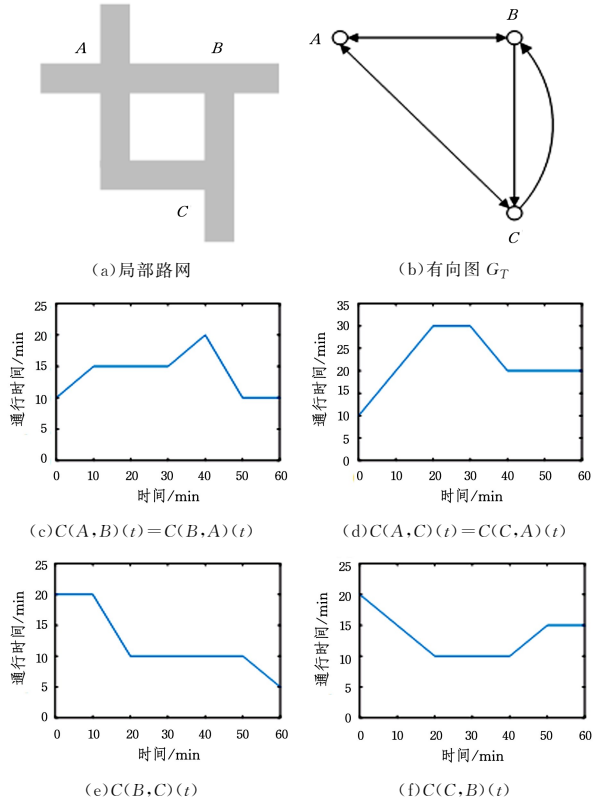
空间网格索引结构简单且高效，因此被广泛地应用于移动对象索引技术中，其基本思想是将研究区域划分为等效网格，并将对象记录在一个或多个网格中。在查询空间对象时，首先查询对象所在的网格，然后在网格中准确、快速地查询所选对象。Nievergelt 等[18] 提出了网格文件索引，该方法将目标空间划分成固定网格，并将每个网格对应一个桶，利用 hash 过程寻找对象所对应的桶。孟妮娜等[19] 提出了一种方便、易实现的网格划分的空间索引技术。以上网格研究可为快速计算启发值提供思路。

3　基础知识

本文将时间依赖路网建模为一个有向带权图，下面给出有关路网的基本概念。

定义 1(时间依赖路网图[14])　时间依赖路网图 G_T 可被定义为 $G_T=(V,E,C)$。其中，$V=\{v_1,v_2,\cdots,v_n\}$ 是顶点的集合，顶点代表道路上的起始点、终点或交叉点；$E=(v_i,v_j\mid v_i,v_j\in V,i\neq j)$ 是边的集合，每条边表示为 $e(v_i,v_j)$，v_i 和 v_j 是两个不同的点，v_i 代表该边的起始点，v_j 代表该边的终点；$C=\{c(v_i,v_j)(t)\mid(v_i,v_j)\in E\}$，$c(v_i,v_j)(t):[0,T]\rightarrow\mathbb{R}^+$，$t\in[0,T]$，$T$ 是 C 中函数的域，表示对于每一条边 $e(v_i,v_j)$，在 t 时刻该边的旅行时间为 $c(v_i,v_j)(t)$。

假设 C 是一组满足 FIFO(First-In First-Out)属性的分段线性函数。对于边，如果边权值函数 $c(v_i,v_j)(t)$ 连续且处处可微，对任意的 t 满足 $c(v_i,v_j)(t)\leqslant t_1+c(v_i,v_j)(t+t_1)$，或者当边的时间函数 $c(v_i,v_j)(t)$ 为离散值时，对任意的 s 和 t 满足 $s+c(v_i,v_j)(s)\leqslant t+c(v_i,v_j)(t)$，则称具有这样性质的边为 FIFO 边。简单来说，如果对象 A 比对象 B 先访问一条边，那么 A 也比 B 先结束对这条边的访问。时间依赖路网中的研究需要满足 FIFO 性质，否则是 NP 难问题[20]，因此该研究的前提是该时间依赖路网满足 FIFO 性质。另外，不要求时间依赖路网中双向的两条边拥有相同的旅行时间，即对于对应的两条边 (a,b) 和 (b,a)，既允许 $c(a,b)(t)=c(b,a)(t)$，也允许 $c(a,b)(t)\neq c(b,a)(t)$，否则所研究的问题将变成 TD-KNN 问题。例如，图 4(a)是一个局部路网，图 4(b)是代表该路网的有向图，图 4(c)－图 4(f)展示了一段时间内路网中每条边的旅行时间。图 4 中，(A,B) 和 (B,A)、(A,C) 和 (C,A) 有相同的旅行时间，而 (B,C) 和 (C,B) 的旅行时间不同。



(a)局部路网　(b)有向图 G_T

(c)$C(A,B)(t)=C(B,A)(t)$　(d)$C(A,C)(t)=C(C,A)(t)$

(e)$C(B,C)(t)$　(f)$C(C,B)(t)$

图 4　时间依赖路网和一小时内各边的旅行时间

Fig. 4　Time-dependent road network and travel time in one hour

定义 2(到达时间[15])　对于时间路网 $G_T=(V,E,C)$，在 t 时刻从 v_i 点出发，通过边 (v_i,v_j) 到达 v_j 点的时间定义为 $AT(v_i,v_j,t)=t+c(v_i,v_j)(t)\ \mathrm{mod}\ T$。

定义 3(路径通行时间[14])　假设 $\{s=v_1,v_2,\cdots,v_k=d\}$ 是一个顶点序列，代表一条从 s 点到 d 的路径，其中 $e(v_i,v_{i+1})\in E$，$i=1,\cdots,k-1$。对于时间依赖路网 $G_T=(V,E,C)$，在 $t(t\in T)$ 时刻，定义从 s 点沿路径到达 d 点所需要的时间为该路径的通行时间(Travel Time，TT)。TT 的计算方法如式(1)所示：

$$TT(s\rightarrow d,t)=\sum_{i=1}^{k-1}c(v_i,v_{i+1})(t_i) \tag{1}$$

其中，$t_1=t$，$t_{i+1}=t_i+c(v_i,v_{i+1})(t_i)$，$i=1,\cdots,k$。

定义 4(时间依赖最快路径(Time-Dependent Fastest

Path, TDFP))[14] 对于时依路网 $G_T=(V,E,C)$，假设在 t 时刻，从 s 点出发到 d 点的所有路径组成集合 $P(s,d)$，如果 $p\in P$，且对其他所有 $p'\in P$ 都有 $TT(p,t)\leqslant TT(p',t)$，则称 p 为在 t 时刻从 s 点到 d 点的时间依赖最快路径，定义为 $p=TDFP(s,d,t)$。

4 时间依赖路网上的移动对象 K 近邻查询

本文算法基于增量式网络扩展 INE 算法和 A* 算法。从查询对象 q 开始，INE 算法按照离 q 从近到远的顺序来访问 q 的所有可达顶点，直到找到 k 个满足条件的顶点。A* 算法使用距离 $d(q,v_i)$ 加上启发函数 $H(v_i)$ 来决定哪些顶点优先访问，当前距离加上顶点 v_i 的启发函数是 q 与经过 v_i 的目标顶点之间的路径代价的估计。启发函数为每个顶点都添加一个潜在的估计值，以便可以快速获取通往最近兴趣点的最短路径。值得注意的是，启发函数不能过高估计顶点与目标之间的距离。对于路网中的每一条路径 $e(v_i,v_j)$，如果 $H(v_i)$ 满足条件 $H(v_i)\leqslant dist(v_i,v_j)+H(v_j)$（$dist$ 为两点之间的时间距离），则 $H(v_i)$ 是单调的，这种情况下，A* 算法不会产生回退现象，因此效率更高。在该场景中，POI 可以随时改变它们的位置，但是在查询执行期间，所有 POI 被关联到相应的路网顶点且在查询处理期间位置不变。该算法分为两个阶段：预处理阶段和查询阶段。在预处理阶段，建立路网和网格索引；在查询阶段，利用预处理信息和查询算法返回满足条件的 k 个对象。

4.1 移动对象映射

以往研究中均假设 POI 恰好在路网的顶点上或将其直接映射到路网的顶点上，但实际情况并非如此。为了解决移动 POI 不恰好在道路顶点上的问题，我们提出了一种新的映射方法。在此之前引入一个定义。

定义 5（活跃点，Active Vertex） 假设 m 是在路径 $e(w,v)$ 上向顶点 v 移动的对象，则称 v 是活跃点，并称 m 是顶点 v 上的活跃对象。从 m 点到任何顶点 u 的最短路径距离为 m 到 v 的距离加上 v 到 u 的最短路径距离，即 $d(m,u)=o+d(v,u)$。有活跃对象的顶点被称为活跃点，反之为非活跃点。

例如，在图 5 中，m_1 在边 $e(A,B)$ 上由 B 向 A 移动，且距离 A 点的偏移量为 o_1，则 m_1 是活跃点 A 上的一个活跃对象，B 点和 C 点均为非活跃点。本文方法保留了偏移量 o 并将其应用到之后的最快路径的计算中，因此结果更加接近真实值。

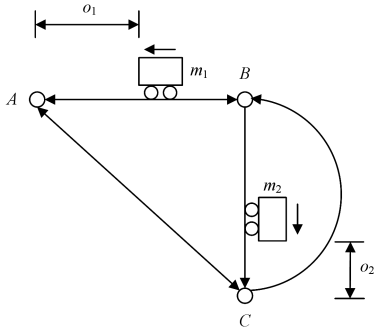


图 5 活跃点和偏移量

Fig. 5 Active vertex and offset

为每个活跃点设计一个结构来保存该点上的活跃对象。假设活跃点 V 上有两个活跃对象 m_1 和 m_2，它们与 V 的距离分别为 10 和 20，则活跃点 V 拥有如表 1 所列的结构。其中，Moving _Objects 代表 V 点上的活跃对象，Offset 代表该活跃对象相对于 V 点的偏移量。

表 1 活跃点的结构

Table 1 Structure of active vertex

Moving_Object	Offset
m_1	10
m_2	20

4.2 网格索引结构的设计

由于网格可以很好地利用空间特性，因此为移动对象设计了网格索引。该索引用均匀的网格将平面区域划分成大小相等的若干块，每一块区域唯一对应一个单元格。对于空间中的点 x，函数 $index(x)$ 可以检索到包含点 x 的网格单元，通过组合倒排索引和空间网格划分来设计倒排网格索引以管理空间对象的位置。本文首先给出倒排网格索引的定义。

定义 6（倒排网格索引） 给定一个由欧氏空间中的一组数据点组成的大规模数据集 P，每个数据点 $p\in P$ 由 $(p.x,p.y)$ 表示。在每个维度上，每个单元格的范围为 δ。网格单元 $c[i,j]$ 表示第 i 列和第 j 行的单元格，最左下角的单元格为 $c[0,0]$。每个单元格维护一个对象列表，其中包含封闭活跃点的标示符。根据规则，点 p 落在单元格 $c[\lfloor p.x/\delta\rfloor,\lfloor p.y/\delta\rfloor]$ 中，δ 是一个参数，我们可以根据不同数据集的密集程度做出相应的调整。

不失一般性，假设服务空间是一个正方形，则可以将空间划分成大小为 $\delta\times\delta$ 的规则网格单元。我们将普通顶点和活跃点均映射到网格中，但每个单元只保存落入该单元的活跃点信息，每个单元中活跃点的信息存储在对象数组中，对象标识是这个数组的索引。图 6 显示了一个 4×4 单元格的网格索引。

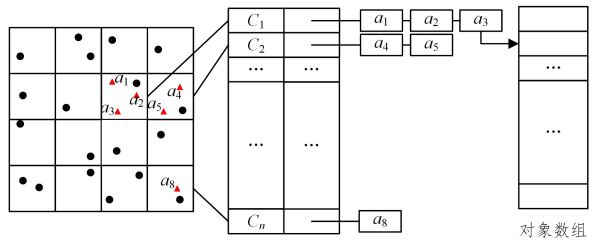


图 6 网格索引

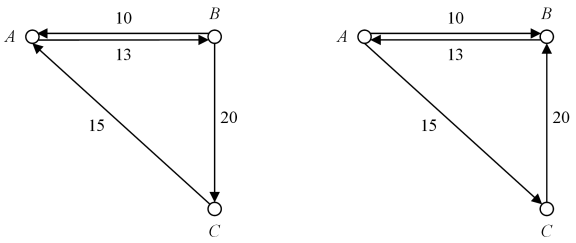
Fig. 6 Grid index

网格单元格的宽度 δ 取决于实际的数据集，如果数据集中的数据点相对密集，则可以将 δ 设置为较小的值，反之可以将 δ 设置为较大的值。总而言之，我们希望绝大多数网格单元中的对象数目是合适的，否则将影响倒排网格索引的查询效率。相比其他复杂数据结构（如 R-tree，Voronoi 等），网格结构更容易更新，当添加新的移动对象时，将相应顶点标识为活跃点，根据坐标和映射规则找到该点落入的网格单元，并在该网格单元的对象列表中加入该活跃点；当删除移动对象时，同样先找到对应活跃点所在的网格单元，从中删除该活跃点；当移动对象位置发生改变时，先从当前网格单元的对象列表中删除对应活跃点，再将其加入到新的网格单元。如果添加或删除移动对象时，其对应的活跃点状态不发生改变，则不对活跃点进行任何操作。

4.3　移动对象 K 近邻查询算法

该算法基于增量式网络扩展(INE)算法，INE 基于迪杰斯特拉算法，它由查询点 q 开始向外辐射式搜索，并按照顶点接近 q 的顺序来访问 q 的所有可达顶点，直到找到最近的兴趣点。由于它是由查询点 q 向兴趣点方向搜索，而我们的目的是要找到从兴趣点到查询点用时最短的对象，因此在计算过程中，对于有向路网图 G，我们需要反转其道路，反转后的时间依赖路网称为反向时间依赖路网。

定义 7(反向时间依赖路网)　反向时间依赖路网与原路网 G 拥有相同的顶点集，但路径相反。如图 7 所示，图 7(a)为原局部路网图 G，图 7(b)为图 7(a)的反向路网图。



(a)原路网图 G　　(b)反向路网图

图 7　路网图与其反向路网图

Fig. 7　Road network graph and its reverse road network graph

(1)预处理阶段

预处理分为两个步骤：1)使用全天道路通行时间的最小值作为权值，在内存中创建路网图 G 的反向时间依赖路网；2)建立倒排网格索引，将路网顶点依据其坐标映射到各自的网格单元中。

(2)查询阶段

在路网的扩展搜索中，我们希望有机会成为 q 最近邻的 POI 以获得最大的优先扩展权。对于顶点来说，我们希望距离其最近 POI 更近的顶点能被优先扩展，我们利用网格索引来计算每个顶点与自己最近 POI 之间的距离。

对于一个顶点 v，算法 1 展示了计算 v 的启发值 $H(v)$的过程。首先，查找 v 所属网格单元 c，通过索引得到 c 中所有的活跃点集合 A。计算 v 和 A 中每个点的欧氏距离，找出距离最小的活跃点 a，$d_{\min}=d_E(v,a)$(第 4－9 行)。然后，计算 v 和 c 相邻的 8 个网格单元 C_i(1≤i≤8)的距离，如果距离小于 $d_{\min}$，则计算 v 和 C_i 中所有活跃点的距离，如果所得值小于 $d_{\min}$，则更新 $d_{\min}$(第 10－17 行)。如果 v 所在单元和相邻单元中都不存在活跃点，则给 $d_{\min}$ 赋予一个较大的默认值，表示 v 离任何一个活跃点都还有较远的距离。$H(v)=d_{\min}/V_{\max}$，$V_{\max}$表示路段上允许的最大速度。

算法 1　Calculate heuristics

输入：顶点 v
输出：启发值 H(v)
1. c←v 所属网格单元
2. A←c 中活跃点集合
3. $d_{\min}$←default
4. for each a∈A
5.　　$d=d_E(v,a)$
6.　　if($d<d_{\min}$) then $d_{\min}=d$
7.　　//得到 v 距 c 中最近活跃点的距离
8.　　end if
9. end for
10. for each c 相邻网格单元
11.　　if($d_E(v,c_i)<d_{\min}$) then
12.　　$d_{\min\_i}$←v 距 c_i 中最近活跃点的距离
13.　　if($d_{\min\_i}<d_{\min}$) then $d_{\min\_i}=d_{\min}$
14.　　//得到 v 距 c 及 c 周围网格中最近活跃点的距离
15.　　end if
16.　　end if
17. end for
18. $H(v)=d_{\min\_i}/V_{\max}$
19. return H(v)

如图 8 所示，$H(v)=d_2/V_{\max}$。由此可见，网格单元宽度 δ 过大或过小都会降低计算效率。

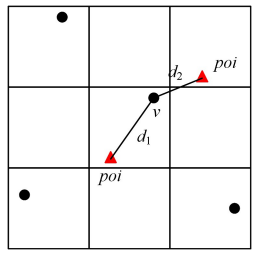


图 8　路网顶点 v 及其最近 POI 距离

Fig. 8　Road network vertex v and its nearest POI distance

定理 1　$H(\cdot)$是一个可接受的启发值，算法不会产生回退现象。

证明：对于查询点 q、当前访问顶点 v 和距离 v 最近的兴趣点 p，假设 $H(v)$过高估计了 v 点离与自己最近 POI 之间的时间距离，即 $dist(v,q)+H(v)=dist(v,q)+\frac{d_E(v,p)}{V_{\max}}>dist(v,q)+dist(p,v)$，其中$\frac{d_E(v,p)}{V_{\max}}$表示 v 到 p 的欧氏时间距离。由欧氏距离不大于路网距离可知，不等式$\frac{d_E(v,p)}{V_{\max}}>dist(p,v)$不成立，因此 $H(v)$没有过高估计到的时间距离，是一个可接受的启发值。同理，对于路网中的任意一条边(v,w)，都有$H(v)\leqslant dist(v,w)+H(w)$，可见 $H(\cdot)$是一个单调函数，因此算法不会产生回退现象。

算法 2 给出了 TD-MOKNN 伪代码。首先，确定查询发起的位置，将查询点 q 作为算法输入值之一。KNN 查询的结果取决于查询发起的时间，因此需要输入查询时间和最大路程时间。算法中使用两个优先队列 Q 和 Q_m，Q 用于寻找近邻活跃点，Q_m 用于在近邻活跃点中找出 q 的近邻移动对象。算法 2 中第 1－4 行初始化相关值，开始先将查询点 q 插入到 Q 中(第 5 行)，Q 存储下一步中用于扩展的候选顶点集，条目是四元组$(v_i,AT_{v_i},TT_{v_i},L_{v_i})$。其中，$AT_{v_i}=AT(q,v_i,t)$，$t$ 是查询发起的时间；$TT_{v_i}=TT(q,v_i,t)$；L_{v_i}是 q 到 v_i 的通行时间加上从 v_i 到最近移动对象的最小通行时间，即 $L_{v_i}=TT_{v_i}+H(v_i)$。L_{v_i}表示通过 v_i 点到达最近移动对象的乐观期望，因此 Q 中元素的优先级由 L_{v_i}值的递增顺序给出，目的是首先检查能够快速找到移动对象的顶点，减少不必要顶点的访问。接下来，顶点依次从 Q 中出队(第 7 行)，出队顶点的访问标记为真，如果出队顶点 u 是活跃点且 L_u 小于 Q_m 队列中第 k 个元素的附加值，则将 u 上的移动对象插入到 Q_m 队列中(第

13 行)。Q_m 条目是二元组(m_i, L'_{m_i}),$L_{m_i}=L_u+m\_offset$。由于 u 是活跃点,因此 L_u 表示从 u 到查询点的通行时间,m_offset 表示 m 相对于 u 点的偏移量,因此 L_u+m_offset 表示从 m 到查询点的通行时间。如果 L_u 已经大于或等于 Q_m 中第 k 个元素的附加值,则搜索过程结束,将 Q_m 中前 k 个移动对象作为结果并返回(第 16 行),证明过程见定理 2。在扩展过程中,对于出队顶点的邻接点 v,首先检查该点是否被访问过,如果被访问过则跳过该点(第 18 行)。接着计算 v 点的通行时间 TT_v 和到达时间 AT_v,如果 TT_v 超过最大通行时间,同样跳过该点。计算完成后,将 v 点插入队列(第 19－23 行)进行下一轮迭代,如果 v 已经在 Q 中,且 L 值小于原 L 值,则更新元素 v 的条目。

算法 2 TD-MOKNN 算法

输入:查询点 q,查询发起时间 t,最大路程时间 m_t

输出:距离查询点所需时间最短的 k 个移动对象

```
1. G'←时间依赖反向路网图
2. AT_q←t+m_t
3. TT_max←0
4. k^th_Q_m=∞
5. Enqueue(q,AT_q,TT_max,L_q)in Q
6. while Q≠∅ do
7.     (u,AT_u,TT_max,L_u)← Dequeue Q
8.     u.visited=true
9.     if(TT_max≤m_t) then
10.      if(u 是活跃点) then
11.        if(L_u<k^th_Q_m) then
12.          for each u 上的移动对象 m
13.            Enqueue (m,L_m') in Q_m
14.            if(Q_m. size ≥k) then
15.                k^th_Q_m=Q_m中第 k 个对象的 L'
16.        else return Q_m 中前 k 个移动对象
17.      for each u 的邻接点 v do
18.        if(v. visited) then continue
19.        TT_current←u_TravelTime+v_dist(u,v)
20.        if(TT_current> m_t) then continue
21.        AT_current←u_ArrivalTime-dist(u,v)
22.        //设置应到达 v 点的时间
23.        Enqueue(v,AT_current,TT_current,L_v) in Q
24. end while
```

定理 2 假设 v 是从 Q 出队的活跃点,若 L_v 大于或等于中第 k 个元素的附加值 L',则当前 Q_m 中前 k 个对象为 KNN。

证明:假设活跃点 v 上有移动对象 $m_1, m_2, \cdots, m_n$($n\geqslant 1$),其中偏移量最小的对象为 m_i,偏移量为 o_i,$L'_{m_i}=L_v+m_i\_offset=L_v+o_i$。假设 Q_m 中第 k 个元素的 $L'=k^{th}\_L'$,由条件知,$L_v\geqslant k^{th}\_L'$,又有 $o_i\geqslant 0$,因此 $L'_{m_i}=L_v+o_i\geqslant k^{th}\_L'$。由此可见,$v$ 的所有移动对象的 L' 值都大于或等于 k^{th}_L'。Q 是以 L 值为序的优先队列,对于 v 之后的所有活跃点 u,有 $L_u\geqslant L_v$,继而 u 上的移动对象的 L' 值也都大于或等于 k^{th}_L',因此当前 Q_m 中前 k 个对象是 q 到查询点最近的 k 个对象,故当前 Q_m 中前 k 个对象为 KNN。

定理 2 证明了 TD-MOKNN 算法的正确性,该算法的时间复杂度为 $O(|E|\cdot\log|V|/m)+O(k\cdot\log m)$,$V$ 和 E 分别是路网的顶点数和边数,m 是移动对象的数目,k 是返回的结果数目。

4.4 实例分析

路网图和各边的旅行时间如图 4 所示,移动对象和偏移量如图 5 和表 1 所示。假设查询点为 A,查询时间 $t=10$,最大通行时间 $m_t=30$,下面展示算法求解最近邻移动对象的过程。将 A 点以条目(A,$AT=10+30=40$,$TT=0$,$L=0$)插入到队列 Q 中。作为 Q 中唯一的元素,A 出队,并将访问标记设为真。由于 A 点是活跃点,因此把 A 上的移动对象 m_1 以条目(m_1,$L'=0+10=10$)插入到队列 Q_m 中。A 的邻接点 B 点和 C 点都未曾访问过,依次计算其 TT 值和 AT 值。对于 B 点,$TT_B=TT_A+c(B,A)(10)=0+15=15$,$AT_B=AT_A-TT_B=40-15=25$,假设启发值 $H(B)=3$,因此 $L_B=TT_B+H(B)=15+3=18$,在 Q 中插入条目(B,25,15,18);利用同样的方法计算 TT_C 和 AT_C,在 Q 中插入条目(C,20,20,20)。B 的 L 值小于 C 的 L 值,因此 B 先出队,访问标记设为真,$TT_B=20\leqslant m_t=30$,但 B 不是活跃点,直接访问 B 还未被访问的邻接点 C,通过 B 的相关值计算出 $TT_C=25$,$AT_C=15$,$L_C=25$,此时由于 C 点已经存在于 Q 中,且原 L 值为 20,$25>20$,因此 C 点条目不做更新。C 点作为 Q 中唯一的元素,出队,并将访问标记设为真,C 是活跃点,但 $L_C=20\geqslant L_m'=10$,根据定理 2,Q_m 中的第一个元素为 NN 结果,因此 m_1 作为最近邻结果返回,查询结束。

5 实验及分析

5.1 实验设置

实验配置:Inter Core i3-2120 CPU @ 3.30GHz 处理器,4GB 内存,Windows 7 操作系统。采用奥尔登堡城市路网图进行仿真实验,路网图每隔 5min 对边赋权值,得到边的 288 个权值。将提出的 TD-MOKNN 算法与 NN-Reverse-TD 算法[17]进行比较,并根据 K 值、POI 密度和路网顶点数评估了本文方法,实验参数如表 2 所列。对于每个实验,改变一项参数的值并将其他两项参数设为默认值。为避免实验结果的随机性,查询时间均选取 30 次测试结果的平均值。

表 2 实验参数

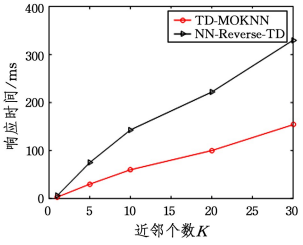
Table 2 Parameters values of experiments

参数	默认	范围
顶点数	10000	1,5,10,15,20000
K 值	20	1,5,10,20,30
POI 密度	0.1	0.05,0.1,0.15,0.2

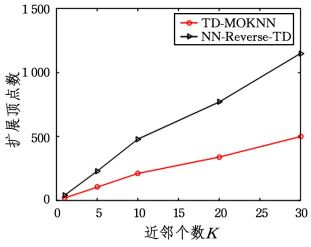
5.2 近邻个数 K 值对查询性能的影响

将 K 值设置为 1,5,10,20,30,POI 密度为 0.1,路网顶点数为 10000。图 9 展示了两种算法的响应时间和扩展顶点数目随 K 值增加的变化情况。图 9(a)显示了两种算法随 K 值的增加,查询响应时间的平均值对比曲线;图 9(b)显示了随 K 值的增加,扩展顶点数的平均值对比曲线;图 9(c)显示了相对于 NN-Reverse-TD 算法,TD-MOKNN 算法剪枝顶点的比例,它是后者比前者多剪枝掉的顶点数与 NN-Reverse-TD 算法扩展顶点数的比值。由图可以看出,响应时间和扩展顶点数随着 K 值的增加而增加,这是因为查询目标顶点的增加,导致需要扩展更多的顶点以得到目标顶点,而扩展更多的

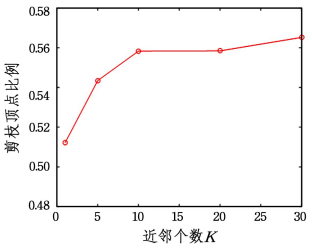
顶点同时也带来了响应时间的增加。



(a)近邻个数 K 值对查询响应时间的影响



(b)近邻个数 K 值对扩展顶点数的影响



(c)近邻个数 K 值对剪枝比例的影响

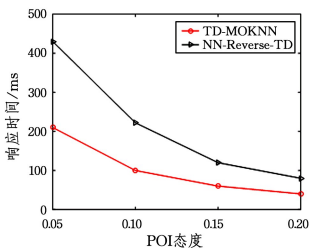
图 9　K 值对查询的影响

Fig. 9　Impact of K on queries

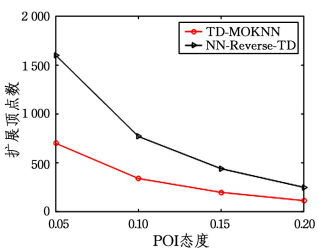
TD-MOKNN 算法比 NN-Reverse-TD 算法的平均遍历顶点个数减少了 54.8%，这是因为两者虽然同为启发式搜索算法，但是所用启发值不同。NN-Reverse-TD 算法在进行路网扩展时，将每个顶点的启发值设置为从该点到最近邻点之间的距离，而 TD-MOKNN 算法则将顶点的启发值设置为该点到最近 POI 之间的距离，因此 TD-MOKNN 算法在扩展时使能更快到达 POI 的顶点具有优先扩展的机会，从而减少了整体扩展的顶点数。TD-MOKNN 算法比 NN-Reverse-TD 算法的平均响应时间减少了 55.3%。响应时间的差异主要是由扩展顶点数不同引起的，因此响应时间和扩展顶点数随 K 值变化的趋势大体相同。当 $K=1$ 时，遍历顶点太少，算法响应时间的差距较小。

5.3　路网 POI 密度对查询性能的影响

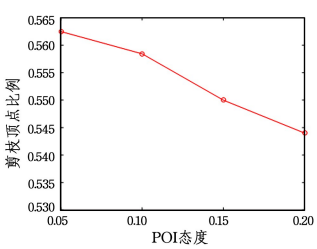
图 10 展示了当固定 K 值为 20，路网顶点数为 10000 时，不同 POI 密度对响应时间和扩展顶点数目的影响。



(a)POI 密度对查询响应时间的影响



(b)POI 密度对扩展顶点数的影响



(c)POI 密度对剪枝比例的影响

图 10　POI 密度对查询的影响

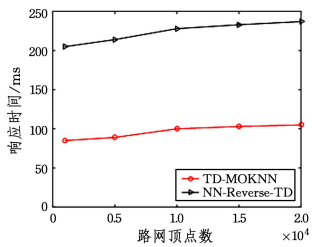
Fig. 10　Impact of POI density on queries

图 10(a)和图 10(b)分别显示了两种算法随 POI 密度的增加，响应时间的平均值对比曲线和遍历顶点数的平均值对比曲线；图 10(c)显示了剪枝顶点比例。由图 10(a)和图 10(b)可知，随着 POI 密度的增加，两种算法的响应时间和扩展顶点数均在减少，这是因为当查询目标顶点数 K 一定时，POI 密度越大，顶点作为目标顶点的概率就越大，因此随着 POI 密度的增加，遍历顶点数减少，响应时间随之缩短。从图 10(c)可以看出，POI 稀疏时，算法之间的剪枝差距较大，随着 POI 密度的增加，TD-MOKNN 算法的修剪能力降低，这是因为 POI 增加导致了启发式函数质量的下降。

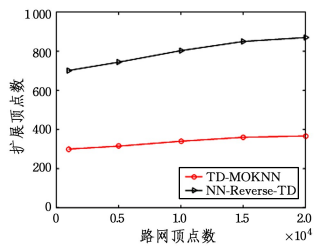
TD-MOKNN 算法比 NN-Reverse-TD 算法的平均扩展顶点数减少了 55.4%，响应时间缩短了 51.5%。这是因为 TD-MOKNN 算法采用的启发值更接近实际值，能优先扩展更快到达 POI 的顶点，从而使得算法更加高效。响应时间由扩展顶点个数决定且具有相同的趋势，因此，由扩展顶点数变化规律可知，响应时间也随 POI 密度的增加而减少。

5.4　路网顶点数对查询性能的影响

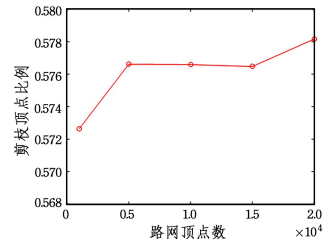
图 11 展示了当固定 K 值为 20，POI 密度为 0.1 时，不同路网规模对响应时间和扩展顶点数目的影响。图 11(a)和图 11(b)显示了两种算法随顶点数的增加，响应时间的平均值对比曲线和遍历顶点数的平均值对比曲线；图 11(c)显示了剪枝顶点的比例。TD-MOKNN 算法比 NN-Reverse-TD 算法的平均扩展顶点数减小了 54.8%，响应时间缩短了 55.3%。由图可见，响应时间和遍历顶点数随着路网顶点数的增加而增加，但没有产生较大影响。这是因为 K 值和 POI 密度固定使得测试图例具有相似的路网结构，小型路网可以看作较大型路网的子图，并且在这些规模不同的路网中，平均扩展的顶点数目是相同的。TD-MOKNN 算法仍因为更接近真实情况的启发值，而比 NN-Reverse-TD 算法在响应时间和扩展范围方面表现得更优。



(a)路网顶点数对查询响应时间的影响



(b)路网顶点数对扩展顶点数的影响



(c)路网顶点数对剪枝比例的影响

图 11　路网顶点数对查询的影响

Fig. 11　Impact of road network size on queries

结束语　相比静态路网，时间依赖路网在服务提供、紧急救援、智能导航方面更具有现实意义。针对以往在时间依赖路网上只能查询静态对象的问题，本文提出了一种高效查询

移动对象的 TD-MOKNN 算法。该算法提出了一种新的移动对象映射方法，解除了以往研究假设移动对象恰好在路网顶点上的限制，并为移动对象和路网顶点建立了空间网格索引以便高效管理。在网格索引的基础上，为启发式算法设计了更加接近真实值的启发值，减少了需要扩展的路网顶点数并提高了算法效率。将 TD-MOKNN 算法与已有算法在多种情况下进行仿真对比分析，结果显示 TD-MOKNN 算法在响应时间和扩展顶点数上均优于对比算法。下一步可将该方法应用到时间依赖路网的移动对象连续 K 近邻查询算法中。

参 考 文 献

[1] YANG H M, WANG H Z, WU Y B. Observation and Characteristics Analysis of Traffic Flow in Nanjing [J]. Environmental Science and Technology, 2011, 24(a02): 98-101.

[2] CHO H J, CHUNG C W. An Efficient and Scalable Approach to CNN Queries in a Road Network [C]// International Conference on Very Large Data Bases. Trondheim, Norway: DBLP, 2005: 865-876.

[3] PAPADIAS D, ZHANG J, MAMOULISM N, et al. Query Processing in Spatial Network Databases [J]. Proc Vldb, 2003, 29: 802-813.

[4] GOLDBERG A V, HARRELSON C. Computing the Shortest Path: A* Search Meets Graph Theory: Technical Report MSR-TR-2004-24[R]. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2005: 156-165.

[5] GEISBERGER R, VETTER C. Efficient Routing in Road Networks with Turn Costs [C]// International Conference on Experimental Algorithms. Berlin: Springer-Verlag, 2011: 100-111.

[6] ABEYWICKRAMA T, CHEEMA M A. Efficient Landmark-Based Candidate Generation for kNN Queries on Road Networks [C]// Proceedings of the 22nd International Conference on Database Systems for Advanced Applications. Berlin: Springer, 2017: 425-440.

[7] XUAN K, ZHAO G, TANIAR D, et al. Constrained Range Search Query Processing on Road Networks [J]. Concurrency and Computation: Practice and Experience, 2011, 23(5): 491-504.

[8] DIJKSTRA E W. A Note on Two Problems in Connection with Graphs [J]. Numerische Mathematics, 1959, 1(1): 269-271.

[9] ERWING M. The Graph Voronoi Diagram with Applications [J]. Networks, 2015, 36(3): 156-163.

[10] KOLAHDOUZAN M, SHAHABI C. The Graph Voronoi Diagram with Applications [C]// Proceedings of the Thirtieth International Conference on Very Large Data Bases. San Francisco, CA: Morgan Kaufmann, 2004: 840-851.

[11] HUANG X, JENSEN C S, ŠALTENIS S. The Islands Approach to Nearest Neighbor Querying in Spatial Networks [C]// Proceedings of the 2005 International Symposium on Spatial and Temporal Databases(SSTD 2005). Berlin: Springer, 2005: 73-90.

[12] COOKE K L, HALSEY E. The Shortest Route Through a Network with Time-Dependent Internodal Transit Times [J]. Journal of Mathematical Analysis & Applications, 1966, 14(3): 493-498.

[13] GEORGE B, KIM S, SHEKHAR S. Spatio-temporal Network Databases and Routing Algorithms: a Summary of Results [C]// International Conference on Advances in Spatial and Temporal Databases. Berlin: Springer-Verlag, 2007: 460-477.

[14] DEMIRYUREK U, BANAEIKASHANI F, SHAHABI C. Towards K-Nearest Neighbor Search in Time-Dependent Spatial Network Databases [C]// International Conference on Databases in Networked Information Systems. Berlin: Springer-Verlag, 2010: 296-310.

[15] CRUZ L A, NASCIMENTO M A, MACÊDO J A F. K-nearest Neighbors Queries in Time-Dependent Road Networks [J]. Journal of Information & Data Management, 2012, 3(3): 211-216.

[16] KOMAI Y, NGUYEN D H, HARA T, et al. kNN Search Utilizing Index of the Minimum Road Travel Time in Time-Dependent Road Networks [J]. Information Sciences, 2014, 255(1): 135-154.

[17] CHUCRE M, NASCIMENTO S, MACEDO J A, et al. Taxi, Please! A Nearest Neighbor Query in Time-Dependent Road Networks [C]// IEEE International Conference on Mobile Data Management. Piscataway, NJ: IEEE, 2016: 180-185.

[18] NIEVERGELT J, HINTERBERGER H, SEVCIKK C. The Grid File: An Adaptable, Symmetric Multi-Key File Structure [J]. ACM TODS, 1981, 9(1): 38-71.

[19] MENG N N, ZHOU X D. Realization on Spatial Index with Grids of Fixed Size [J]. Beijing Surveying and Mapping, 2003(1): 7-11.

[20] ORDA A, ROM R. Shortest-Path and Minimum-Delay Algorithms in Networks with Time-Dependent Edge-Length [J]. Journal of the Acm, 1990, 37(3): 607-625.

ZHANG Tong, born in 1996, postgraduate, is student member of China Computer Federation (CCF). Her main research interests include spatial database query technology.

QIN Xiao-lin, born in 1953, Ph.D, professor, Ph.D supervisor, is member of China Computer Federation (CCF). His main reasearch interests include spatial and spatio-temporal databases, data management and security in distributed environment, etc.