方向感知的路网移动对象范围查询算法

董天阳 尚跃辉 程 强
(浙江工业大学计算机科学与技术学院 杭州 310023)

摘 要 路网移动对象的范围查询作为空间查询处理中经典的查询类型之一,已经在很多领域中得到了广泛应用。但现有的路网移动对象范围查询方法仍然存在一些不足:一方面,大多数的路网移动对象范围查询方法仅考虑了路网距离,而很少关注范围内移动对象在路网中的运动方向;另一方面,为数不多的考虑了移动对象运动方向的查询方法,几乎都基于欧氏空间进行查询处理,不能应用到大规模的路网来判断范围内的移动对象是否朝向查询点运动。针对在大规模复杂路网下如何高效地查找附近范围内所有朝向查询点的移动对象的问题,提出了一种方向感知的路网移动对象范围查询算法。该算法使用 R-tree 和简单网格作为底层索引支撑,同时利用一种高效的朝向查询点的路网移动对象判定方法,来高效地查找范围内朝向查询点的移动对象。分别从查询范围、移动对象数量以及网格划分数量 3 个方面进行实验分析,结果表明方向感知的路网移动对象范围查询算法在合理的参数范围内具有较高的实用性和有效性。

关键词 方向感知,路网,移动对象,范围查询
中图法分类号 TP392 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.11.033

Direction-aware Moving Object Range Query Algorithm in Road Network

DONG Tian-yang SHANG Yue-hui CHENG Qiang
(School of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China)

Abstract The range query on moving objects in road network is one of the classical query methods in spatial query processing, which has been widely used in many fields. However, there are still some shortcomings in the existing moving object range query methods in road networks. On the one hand, most of them only consider the network distance, but pay less attention to the movement direction of moving objects in road networks. On the other hand, a few query methods considering the movement direction are based on Euclidean space and can't not be applied to large-scale road network to determine whether moving objects move toward query point. Aiming at the problem of accurately and efficiently finding moving objects toward the query point in large and complex road network, this paper proposed a direction-aware moving object range query algorithm in road network. In this method, R-tree and simple grid are taking as underlying index support, and an efficient determination method of moving object toward query point in road network is exploited, so as to efficiently query the moving object toward query point. At last, this paper designed experiments in terms of the query range, the number of moving object and the number of grid partition to evaluate the performance of the proposed algorithm. The experimental results show that the algorithm can achieve good query performance even in the case of large-scale complex road network under the condition that the index configuration is reasonable.

Keywords Direction-aware, Road network, Moving object, Range query

1 引言

随着智能移动设备的发展,许多查询技术应运而生,例如范围查询技术,用于查找给定范围内的对象。目前,针对移动对象的空间范围查询技术更多地关注于它们的位置信息,根据不同的距离度量方式,范围查询又可分为基于欧氏距离的范围查询和基于路网距离的范围查询。基于欧氏距离的方位查询相对简单,例如:在图 1 中,以查询点 q 为中心,查找欧氏范围在 5 km 以内的移动对象。以 5 km 为半径构造查询窗口,所有被该圆形覆盖的移动对象都符合查询结果,其结果是 p_1,p_2,p_3,p_4,p_5,p_6。但是,如果查询点 q 的查询请求是:"返回所有路网距离小于 5 km 的移动对象",图 1 中竖线为分割

到稿日期:2017-11-22 返修日期:2018-02-14 本文受国家自然科学基金项目(61672464,61572437),浙江省重点研发计划项目(2017C01013)资助。

董天阳(1977—),男,博士,副教授,CCF 会员,主要研究方向为虚拟现实、大数据与人工智能,E-mail:dty@zjut.edu.cn(通信作者);**尚跃辉**(1992—),女,硕士生,CCF 会员,主要研究方向为智能交通、空间数据库,E-mail:18758594584@163.com;**程 强**(1990—),男,硕士生,主要研究方向为智能交通、数据挖掘。

线，则查询结果为 p_1，p_2，p_3，p_4，p_5，这是由于移动对象 p_6 到查询点的路网距离是大于 5 km 的，不符合条件。

这种范围查询方法忽略了移动对象的方向信息，使得现有的空间范围查询技术无法满足人们实时感知移动对象方向的需求。例如，某个路段发生了交通事故，导致了严重的交通堵塞，如果此时交通监控系统及时发出通知，建议附近范围内所有朝向交通事故发生地点行驶的车辆避开该路段，那么就可以避免交通拥塞的加剧。在上述场景中，车辆所处的环境并不是简单的欧氏空间，而是更为错综复杂的路网约束空间，这为车辆的方向判断带来了很大的挑战。

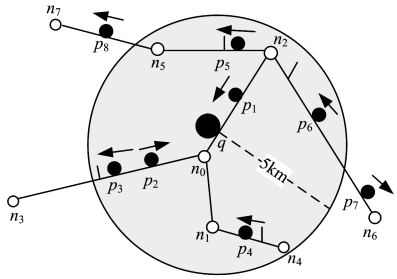

图 1　范围查询示例

Fig. 1　Example of range query

鉴于此，本文提出了方向感知的路网移动对象范围查询算法。该算法采用了 R-tree 结合简单网格的双索引方案，其利用 R-tree 索引静态路网数据，利用简单网格索引频繁更新的移动对象，从而充分发挥了两个索引的优势，采用最小垂直距离方式可有效地将移动对象映射到对应的路网边上。在双索引的支撑下，利用朝向查询点的路网移动对象判定方法，并通过扩展查询范围内的局部路网，来高效地筛选出查询范围内所有朝向查询点的路网移动对象。最后，设计实验进行查询算法的性能分析。

2　相关工作

目前，随着智能移动设备的快速发展，获取移动对象的位置和方向信息越来越容易，为此人们对移动对象的空间查询有更多的需求。已经有许多学者针对范围查询进行了研究，并提出查询处理算法，这些查询处理大致分为欧氏空间的查询处理、路网空间的查询处理以及方向感知的查询处理。

2.1　空间范围查询处理

早期的范围查询集中在欧氏空间下的对象。从海量的空间数据中获取移动对象信息的关键是索引技术，因此学者们将研究的重点放在了空间数据的索引结构上。Guttman[1] 提出了空间数据库发展史上最为经典的一种索引结构 R-tree，在此之后，基于 R-tree 的各种衍生索引也相继被提出，包括 R^+-tree[2]，R^*-tree[3]，Hilbert R-tree[4]，PR-tree[5] 等，这些索引方法是对欧氏空间对象进行索引的，都支持欧氏空间下的范围查询处理。在 R-tree 索引的基础上，Korn 等[6] 首先利用范围查询获得 KNN 候选集，然后基于候选集合精选出 KNN 目标对象。除了以 R-tree 为索引支撑的查询算法外，还有一些基于 Voronoi 图的查询算法。比如，Berchtold 等[7] 和 Zheng 等[8] 通过构建 Voronoi 图的索引，解决了高维空间下的快速查询问题。但这些方法只高效地解决了近邻查询问题，不能处理范围查询。Sharifzadeh 等[9] 提出了 VoR-Tree 索引，结合 Voronoi 图和 R-tree 各自的优点，首先在 R-tree 上进行粗粒度的查询范围搜索，然后在 Voronoi 图上进行细粒度的快速查询。这一类查询算法在近邻个数确定的情况下非常有效，若个数不确定，则会导致频繁的 Voronoi 图索引的重构，使得查询性能急剧下降。

欧氏空间下查询处理从静态对象查询转变为移动对象查询。虽然 R-tree 索引在静态对象的索引上表现得非常稳定和高效，但面对移动对象频繁的位置更新时，未表现出高效的查询性能，一些学者针对欧氏空间下的移动对象提出了索引结构，如 Jensen 等[10] 通过 Hilbert 空间填充曲线，将高维空间数据线性化，并基于 B^+-tree 索引提出了适用于索引移动对象的 Bx-tree，同时提出了相应的范围查询。

由于路网空间的查询处理中必须考虑空间对象受路网的约束，因此查询处理过程与欧氏空间不同。基于路网空间的查询技术一直以来都得到了国内外学者的广泛关注和研究。Papadias 等[11] 提出了空间网络查询处理框架，采用渐进式路网扩展(INE)技术对路网的边按照到查询点的距离进行访问排序，即以查询点为中心，逐步进行扩展搜索，在扩展过程中比较检索到的移动对象到查询点的距离，当扩展半径超过第 K 个近邻对象到查询点的距离时，扩展结束。Kolahdouzan 等[12] 提出了一种新颖的基于路网 Voronoi 图的查询算法 VN^3，首先通过构建路网的 Voronoi 图，将大规模的全局路网空间划分成很多小的 Voronoi 子区域，然后预计算每个 Voronoi 子区域内的距离，以及穿过该子区域的距离。该方法无须计算任意两个路网节点之间的距离，节省了大量额外存储空间和计算开销。这种基于预计算的查询算法在处理数据密度较大的情况下会出现严重的性能下降问题。针对不同的数据密度，Huang 等[13] 提出了 Islands 算法，通过引入“island”的概念，为每个数据点构建一个半径为 r 的“island”范围，然后预计算数据点到“island”范围内每个路网节点的距离，并将数据点的引用及其距离保存在 island 范围内对应的路网节点中，从而节省了路网距离的计算开销。该算法对于不同的数据密度可以动态调整“island”半径，具有较好的灵活性。

针对路网中的移动对象，Huang 等[14] 提出了 S-GRID 算法，对整个路网空间进行等大的网格划分，并引入虚拟的“边界点”对跨网格的边进行分割，使得不同网格之间的路网必须通过边界点才能连通，然后抽取这些边界点来抽象地表示路网的连通性，从而减少路网预计算的成本。路网和移动对象更新操作只会在网格内部进行，通过改变网格划分数量和大小就可以控制更新操作花费的额外成本。在 S-GRID 的基础上，学者提出的范围查询算法对移动对象更新速度的处理较慢。Hong 等[15] 在 S-GRID 的基础上采用分布式方式来解决大规模移动对象的更新问题，能够避免过多重复访问近邻网格而造成的额外开销。Wang 等[16] 提出了 R-tree 结合简单网格的双索引结构，利用 R-tree 索引静态的路网数据，利用简单网格索引频繁移动的数据点；并在双索引结构的基础上提出了相应的范围查询算法 MNDR，首先根据欧氏下界属性执行欧氏范围查询，获得查询范围内的局部路网，然后采用 Dijkstra 算法预计算路网距离，最后进行局部路网扩展，筛选符

合条件的目标对象。方向属性是空间对象的主要特征，但上述查询处理算法没有对其进行考虑。

2.2 基于方向查询处理

方向是空间对象的重要属性，将传统的空间查询与方向特征相结合，可以满足人们特定的方向感知需求，为人们提供更加人性化的查询服务。因此，基于方向的查询处理引起了学者的关注。

Patroumpas 等[17]解决了欧氏空间下移动对象朝向查询点运动的范围查询问题。为了快速判定移动对象与查询点之间的朝向问题，其通过以查询点为极点的极化映射，以及为每个查询点生成一棵 PolarTree，来高效地完成朝向查询点的范围查询。但是该方法只适用于查询点固定且已知的场合，无法应对随机发起的查询请求和路网空间下的方向查询需求。

Li 等[18]提出了方向感知空间关键字搜索(Direction-Aware Spatial Keyword Search，DESKS)方法来查找满足关键字和方向约束条件的 k 个最近邻居。该方法为所有兴趣点构造一个最小边界矩形 MBR，根据兴趣点到 MBR 左下角的距离进行分组，并根据它们对左下角的方向对各组中的对象进行分类。在执行查询处理时，可以推断出方向范围。对于任何对象，如果其到 MBR 左下角的方向不在查询的方向范围内，则不满足查询条件。Li 等[18]提出的 View field k Nearest Neighbor Query(VFkNN)，解决了与 DESKS 方法类似的以关键字和方向约束为条件的查询问题，但是 VFkNN 支持移动对象的查询处理。该算法采用网格索引兴趣点，仅访问视角范围内的网格，直接排除视角范围外的网格，大大加快了查询效率。在以上查询处理的空间对象中，只有查询点具有方向的约束，而我们的问题是处理具有空间方向的对象，查找朝向查询点方向运动的移动对象。Lee 等[20]提出了 DCkNN(Direction-Constrained k Nearest Neighbor Query)方法。与 DESKS 方法和 VFKNN 方法相比，DCkNN 方法中的所有对象都有自己的方向，解决了与查询点具有相同朝向的近邻查询问题。给定查询点的位置及查询点的方向，DCkNN 可以从兴趣点集合中查找出与查询点方向相同或者与查询点方向的误差不超过某个阈值 θ 的兴趣点。

Lu 等[21]和 Lee 等[22]研究了方向感知的双色反向 k 近邻查询算法 DART。DART 从移动对象的角度出发，通过设置视角 θ 和半径 r，为每个移动对象计算一个有效区域，这个区域代表了移动对象前进方向上的可视区域。如果有效区域包含了查询点，那么这个移动对象就是朝向查询点运动的。此方法是以移动对象为方向的出发点，不适用于查询点任意时刻发出方向的查询请求。

Li 等[23]研究了方向感知的空间关键字搜索(SKQ)，该查询算法请求的查询条件包括位置和关键词，此外必须指定方向区间，查询结果返回距离查询者最近且覆盖了所有关键词以及在指定方向区间的 k 个对象。此算法只返回指定的方向区间的 K 个对象，对于查询范围内移动对象是否朝向查询点运动方向的处理是无效的。

相比于传统的欧氏空间查询和基于路网的查询，基于方向的查询处理更为复杂，特别是基于路网移动对象的方向感知问题。目前大多数基于方向的查询处理都是在欧氏空间进行的，因此这类查询算法无法有效解决路网约束下的方向感知范围查询问题。为此，本文对方向感知的路网移动对象范围查询进行研究，提出了解决算法。本文所提出的范围查询算法采用双索引结构作为支撑，其中 R-tree 索引静态的路网数据，简单网格索引频繁更新的移动对象，为了高效完成路网移动对象的方向判定问题，本文在 INE[11]算法的基础上提出了朝向查询点的判定方法。方向感知的路网移动对象范围查询利用朝向查询点的路网移动对象判定方法，通过扩展范围内局部路网，高效地筛选出查询范围内所有朝向查询点的路网移动对象。

3 预备工作

本节将对方向感知的路网移动对象范围查询所涉及的一些概念以及数据结构进行描述；并在此基础上对本文要解决的问题进行定义；然后采用了一种高效的移动对象方向判定方法；为了提高查询性能，本文采用 R-tree 和简单网格相结合的双索引机制。

3.1 数据模型与问题定义

为了高效解决方向感知的路网移动对象范围查询，本节对相关数据模型进行概要描述，包括路网、路网距离、移动对象等；然后给出了具体的问题定义。

路网：道路网络由节点和边组成，可以用无向带权图 $G(N,E,W)$ 来表示。其中，N 表示节点集合，存储路网上所有的节点信息；E 表示边的集合，存储路网上的路段，每一个网段 n_s 表示起点，n_e 表示终点；W 表示路段的权值，系统所设权值为路网的长度。

路网距离：给定路网中任意两个位置 v_i 和 v_j，如果它们之间存在一条最短路径，则最短路径的长度就是它们之间的路网距离，表示为 $DN(v_i,v_j)$。如果最短路径不存在，则路网距离 $DN(v_i,v_j)=\infty$。

路网移动对象：路网移动对象用集合 $S(t)$ 表示，$p(t)\in S(t)$ 表示 t 时刻在路网中的一个移动对象。每个路网移动对象 $p(t)$ 都包含了自身的空间信息，如某个时刻移动对象所在的位置、速度、运动方向等信息，这些空间信息都是关于时间 t 的函数。

朝向查询点的路网移动对象：如果 t 时刻路网移动对象 p 与查询点 q 之间的最短路径长度为 d_t，$t+1$ 时刻它们之间的最短路径长度为 d_{t+1}，且 $d_{t+1}<d_t$，则称 p 是朝向查询点的路网移动对象。

问题定义：给定路网 G、查询点 q、查询范围半径 r 以及 t 时刻路网移动对象的快照 S，从 S 中快速找出路网距离小于 r 的朝向查询点的路网移动对象。

为了进一步阐述方向感知的路网移动对象范围查询，下面给出一个具体例子进行对比说明。图 1 中，假设查询点 q 的查询请求为："返回所有路网距离小于 5 km，且朝向查询点运动的移动对象"，则满足查询条件的结果为 p_1，p_2，p_4 这 3 个移动对象，因为移动对象 p_3，p_5 的运动方向是朝着远离查询点的方向，所以不符合查询条件。若判断一个对象是否朝向查询点运动，根据定义，基本做法是计算 t 时刻移动对象到查询的最短距离，同时根据 t 时刻移动对象的速度和方向预

测 $t+1$ 时刻的最短距离，最后比较距离大小，显然这一计算过程很耗时。为了加快查询处理的速度，需要解决如何在一个复杂的路网中快速判断移动对象是否朝向查询点运动。

3.2　朝向查询点的路网移动对象判定方法

路网中，移动对象 p 与查询点 q 的位置关系可以分两种情况来讨论：1）移动对象与查询点在同一条边上；2）移动对象与查询点不在同一条边上。本文假设移动对象的运动方向在一段持续的时间内是稳定不变的。

情况 1　如果路网移动对象与查询点在同一条边上，此时只需要判断移动对象的运动方向是否指向查询点即可。若指向查询点，则沿着该方向前进，它与查询点的路网距离会更短，也就是朝向查询点运动的。如图 2 所示，q 表示查询点，p_i 表示移动对象，箭头表示移动对象运动的方向，其中移动对象 p_1 和 p_2 与查询点 q 在同一条边上，p_1 是朝向查询点 q 的路网移动对象，而 p_2 则不是。

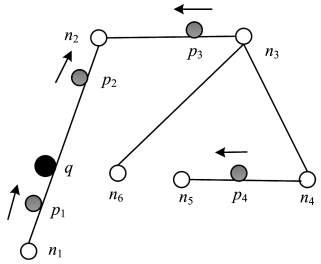

图 2　路网移动对象与查询点的方向关系

Fig. 2　Direction relationship between moving object and query point in road network

情况 2　移动对象与查询点不在同一条边上。如图 2 所示，移动对象 p_3，p_4 与 q 不在同一条边上，p_3 虽没有指向 q，但它实际上是朝着靠近查询点 q 的方向运动的，而 p_4 即使指向了 q，但沿着其方向行驶离查询点 q 的距离会越来越远。因此，移动对象与查询点不在同一条边上时，是无法通过指向查询点的方式判断移动对象是否朝向查询点。

本文在 INE 算法的启发下，采用路网扩展的方式，将移动对象和查询点在不同边的情况转化为在同一条边的情况，以实现高效的方向判定方法，从而快速判断出移动对象是否朝向查询点方向运动。

在 INE 算法中，路网的每一次动态扩展是以查询点为中心，逐渐远离查询点的过程。如图 3 所示，以查询点 q 为中心依次扩展的边是：$e_2(n_2,n_3)$，$e_3(n_3,n_6)$，$e_4(n_3,n_4)$，$e_5(n_4,n_5)$，图中空心箭头表示了路网的扩展方向。由此可见，路网的扩展方向正好与朝向查询点的运动方向相反。

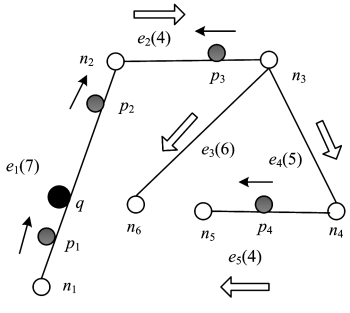

图 3　INE 查询的处理过程

Fig. 3　Process of INE query

为了简化路网扩展的方向以及移动对象的运动方向，本文利用了方位角，如图 4 所示，水平夹角 β 为参考点 n_1 到目标方向线 n_1n_2 之间的方位角。

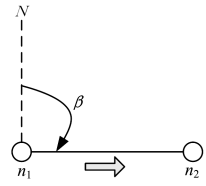

图 4　方位角

Fig. 4　Azimuth angle

对于一个正在被访问的路网节点，其扩展方向就是该节点指向其邻接节点的方向，用方位角 β 表示。如图 5 所示，若当前访问的节点为 n_2，则其扩展的邻接边为 $e_2(n_2,n_3)$，扩展方向是 n_2 指向 n_3，此时 $\beta=\beta_1$。当访问节点为 n_3 时，其扩展邻接边为 e_3 和 e_4，可扩展方向为两个，当扩展边为 $e_3(n_3,n_6)$ 时，扩展方向是 n_3 指向 n_6，此时 $\beta=\beta_3$；当扩展边为 $e_4(n_3,n_4)$ 时，扩展方向是 n_3 指向 n_4，此时 $\beta=\beta_2$。在路网中，节点 n_i 要么是边的起点，要么是边的终点，因此有 $\beta=\alpha$ 或者 $\beta=(\alpha+180)\bmod 360$。

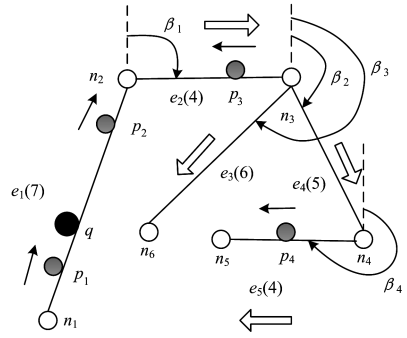

图 5　路网扩展方向的方位角表示

Fig. 5　Azimuth angle representation of expansion direction in road network

利用路网扩展的方向，朝向查询点的路网移动对象的判定规则为：路网扩展到邻接边 $e_i(n_i,n_j)$，边的方向表示为 α，移动对象的运动方向表示为 θ，路网扩展方向表示为 β。

情况 1　当 $\beta=\alpha$ 时，若满足 $180-\varepsilon\leqslant\theta-\alpha\leqslant180-\varepsilon$（$\varepsilon$ 为方位角最小偏差值），则移动对象是朝向查询点运动的；

情况 2　当 $\beta=(\alpha+180)\bmod 360$ 时，若满足 $\theta-\alpha\leqslant\varepsilon$（$\varepsilon$ 为方位角最小偏差值），则移动对象是朝向查询点运动的。

朝向查询点的路网移动对象的判定简单高效，逆着路网节点扩展方向运动的移动对象就是朝向查询点的路网移动对象。如图 5 所示，当前访问的节点为 n_2 时，扩展邻接边为 e_2，扩展方向为 n_2 指向 n_3，而移动对象 p_3 的运动方向是 n_3 指向 n_2，满足逆着路网节点扩展方向运动的条件，因此 p_3 是朝向查询点运动的移动对象；访问到节点 n_4 时，可扩展方向为 n_4 指向 n_5，对象 p_4 的移动方向与扩展方向一致，因此 p_4 不是朝向查询点运动的对象。

3.3　双索引机制

考虑到路网和移动对象各自的特点，本文采用了 R-tree 结合简单网格的双索引方案。其中 R-tree 索引路网数据，其

查询性能非常稳定。简单网格具有结构简单、更新速度快的特点，用于索引频繁更新的移动对象数据。

由于路网与移动对象分开索引导致从 R-tree 中只能检索路网边的信息，从简单网格中只能获取移动对象信息，因此利用路网移动对象映射方法检索边中的对象。例如，对于获取边 e 上的移动对象，首先可以利用简单网格索引获取 e 所覆盖的有效网格；其次获取有效网格内的所有移动对象；再次需要对这些移动对象做进一步的过滤，因为它们并不一定都在 e 上，本文利用最小垂直距离判定法过滤移动对象，一个移动对象在某个路段上的必要条件是该移动对象在路段的最小边界矩形中，因此借助 R-tree 索引递归地自顶向下搜索，获取到的最小边界矩形包含了该移动对象的所有候选路段；然后，计算移动点到候选路段之间的垂直距离；最后，取垂直距离最小的路段作为移动对象所在的路段。

4 方向感知的路网移动对象范围查询算法

方向感知的路网移动对象范围查询处理的流程如图 6 所示，其主要分为两个阶段：局部路网构建阶段和方向感知路网扩展阶段。局部路网构建阶段需要预计算路网节点到查询点的距离；方向感知路网扩展阶段利用 3.2 节的朝向查询点的路网移动对象判定方法扩展整个局部路网。

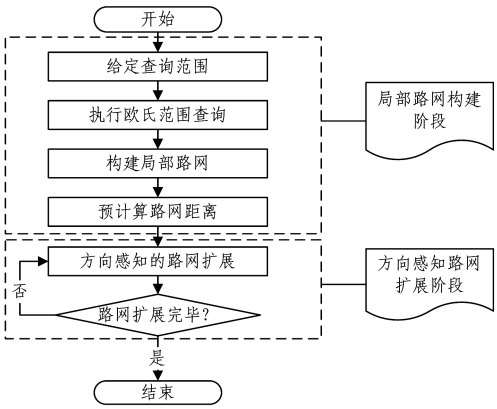

图 6 方向感知的路网移动对象范围查询

Fig. 6 Direction-aware moving object range query in road network

4.1 局部路网构建阶段

本阶段通过查询范围构建局部路网，首先执行基于欧氏距离的范围查询，该步骤实际上包含两层含义，分别是在简单网格和 R-tree 上执行欧氏距离范围查询。

通过在简单网格上执行欧氏距离的范围查询可以排除所有路网距离大于查询范围 r 的移动对象，从而获得方向感知的路网移动对象范围查询的候选对象。一个移动对象 p 到查询点 q 的路网距离为 $DN(p,q)$，假设它们之间的欧氏距离为 $DE(p,q)$，由于移动对象 p 到查询点 q 的路网距离永远大于或者等于它们之间的欧氏距离，因此不等式 $DN(p,q)\geqslant DE(p,q)$ 永远成立。如果移动对象 p 的欧氏距离大于 ω，即 $DE(p,q)>\omega$，则它的路网距离 $DN(p,q)$ 肯定也大于 ω。如图 7 所示，灰色阴影部分表示半径为 r 的查询范围，该查询范围内包含了 p_1，p_2，p_3，p_4，p_5 和 p_6，因此它们是方向感知的路网移动对象范围查询的候选对象，而 p_7 和 p_8 不在查询范围内，即它们到查询点 q 的欧氏距离大于 r，故它们到查询点的路网距离肯定大于 r，不可能成为方向感知的路网移动对象范围查询的候选对象，直接将它们过滤掉。

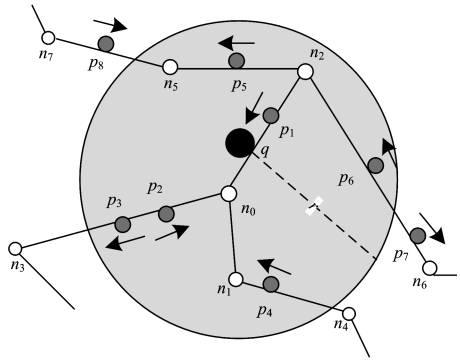

图 7 欧氏距离的范围查询

Fig. 7 Range query of Euclidean distance

接下来拷贝全局路网中被查询范围覆盖的局部路网数据，在内存中重新构造该局部路网，同时将查询点作为新的路网节点加入到局部路网中。局部路网的规模相对整个全局路网来说要小得多，因此可以采用 Dijkstra 算法来高效地预计算局部路网中各个节点到查询点的路网距离。图 8 给出了局部路网的构建过程，图 8(a)为最初拷贝自全局路网的局部路网信息，图 8(b)的路网加入了查询点 q，并预计算了各路网节点到查询点 q 的路网距离。

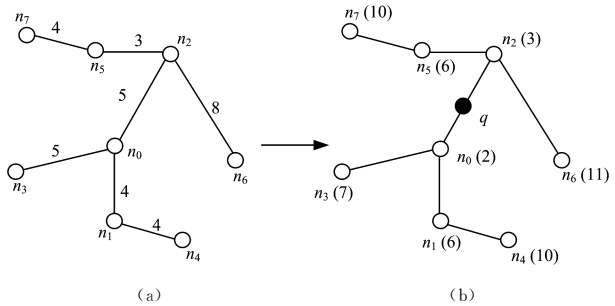

图 8 局部路网的构建

Fig. 8 Construction of local road network

4.2 方向感知的路网扩展阶段

方向感知的路网扩展阶段的目的是在构建的局部路网中筛选出路网距离小于给定查询范围 r 且朝向查询点的路网移动对象。该阶段以查询点为中心，逐步向外扩展路网，每扩展一条边，就获得边上的所有移动对象，然后结合朝向查询点的路网移动对象判定方法获取朝向查询点的路网移动对象；接着计算其到查询点的路网距离，若小于给定的查询范围则将其加入结果集中，否则舍弃。具体处理过程为：首先按照路网节点到查询点的距离将其以从小到大的顺序排序，构造路网节点优先级队列 Q，如果该优先级队列为空，则方向感知的路网扩展阶段结束，否则，从优先队列中取出一个节点；接着获得当前节点的所有未被扩展的邻接边，如果邻接边为空，则继续访问下一个路网节点，否则扩展这些邻接边；每扩展一条边就将其状态标记为“已扩展”，然后获得这条边上的所有移动对象，如果移动对象为空，则继续扩展下一条邻接边，否则利用朝向查询点的路网移动对象的判定方法，对这些移动对象进行精确过滤；如果这个移动对象的运动方向指向当前节点，

而且它到查询点的路网距离小于查询半径 r，则将该移动对象加入结果集中，否则直接将其丢弃。

算法 1 给出了整个方向感知的路网移动对象范围查询的伪代码。

算法 1　方向感知的路网移动对象范围查询算法

```
输入：查询点 q，查询范围半径 r
输出：查询范围内所有朝向查询点的路网移动对象 resultSet
1. resultSet＝Ø
2. candidateSet＝Ø
3. /＊执行欧氏范围查询＊/
4. candidateSet＝gridEuclideanRange(q,r)
5. edges＝rtreeEuclideanRange(q,r)
6. /＊根据欧氏范围内的边，构建局部路网＊/
7. localNetwork＝createRoadNetwork(copyOf(edges))
8. /＊预计算局部路网距离，根据距离节点到查询点的距离从小到大
   排序，构造节点优先级队列＊/
9. belongedEdge＝findEdge(q,edges)
10. localNetwork. addVertex(q,belongedEdge)
11. vertexPriorityQueue＝computeDistance(localNetwork,q)
12. visitedEdge＝Ø
13. /＊方向感知的路网扩展＊/
14. While !vertexPriorityQueue. isEmpty() do
15.    minVertex＝vertexPriorityQueue. poll()
16.    adjEdges＝getAdjEdges(minVertex)
17.    for each edge e in adjEdges do
18.       if !visitedEdge. contain(e) then
19.          visitedEdge. add(e)
20.          points＝mobileObjectMapping(e,0)
21.          for each point p in points do
22.             if isOrientated(p,e,minVertex) then//判断对象方向是
                否指向当前扩展节点
23.                dist＝computeDistance(p,e)
24.                if dist＜r then//若路网距离小于范围 r，则将移动点加
                   入结果集
25.                   resultSet. add(p)
26.                end if
27.             end if
28.          end for
29.       end if
30.    end for
31. end while
32. return resultSet
```

5　实验与分析

本节通过设计详细的实验方案，对方向感知的路网移动对象范围查询算法进行评估，算法的性能指标主要由 CPU 执行时间来衡量。本次实验通过在不同的参数环境下测试算法的查询时间，从整体上分析算法的查询效率。实验结果表明，在参数配置合理的条件下，算法均能取得不错的查询效率。

5.1　实验设计

本文实验数据包括路网数据和移动对象数据，如表 1 所列，其中 Los Angeles 路网数据来源于 TIGER/Line，共有 193948 个道路顶点，265489 条道路；移动对象数据由 Brinkoff 路网数据模拟器[24] 生成，其在 Los Angeles 路网中的分布相对均匀，数量为 10000～100000。

表 1　实验数据

Table 1　Experimental data

实验数据	数量
Los Angeles 路网边/条	265489
Los Angeles 路网节点/个	193948
移动对象/个	10000～100000

实验所用的原型系统由 Java 语言实现，运行环境为四核 Intel(R) Core(TM) i5-2400 CPU @3.10 GHz 处理器、8 GB 内存的 64 位 Windows 7 系统。

算法的性能指标主要通过 CPU 执行时间来衡量，本次实验通过控制变量法分别考查了网格划分数量、移动对象数量以及查询范围半径对查询时间的影响。考虑到查询点在路网中所处的位置对查询时间有较大影响（比如繁华地区的路网比较密集，移动对象数量相对集中，范围查询时间较长，而偏远地区路网稀释，移动对象较少，范围查询时间较短），实验非并发执行 1000 次不同位置上的范围查询，然后统计平均一次范围查询所需的时间，并将该平均查询时间作为最终的实验结果进行分析。实验过程中各个参数的默认配置如表 2 所列。

表 2　实验默认参数设置

Table 2　Setting of experimental default parameters

参数	默认值
R-tree 节点最大容量/entries	50
R-tree 节点最小容量/entries	20
简单网格划分数量/个	1000＊1000
移动对象数量/个	50000
查询范围半径/km	5

5.2　本文实验结果与分析

实验分别针对查询范围半径、网格划分数量以及移动对象数量这 3 个参数进行。在分析其中某一个参数对方向感知的路网移动对象范围查询性能的影响时，其他两个参数若没有特别说明，均采用默认值。

(1)查询范围半径对查询性能的影响

图 9 描述了在拥有 50000 个移动对象的 Los Angeles 路网中执行一次范围查询所需的查询时间随查询范围半径的变化情况。其中折线图部分表示范围查询整体的时间消耗情况，柱状图表示范围查询各个阶段的时间消耗情况。从折线图部分可以看出，范围查询时间随着查询范围半径的增加而增加，而且查询时间的增加速度逐渐变快，但是即使查询半径增加到 10km，范围查询的时间也没有超过 400ms。从柱状图中可以看出，在范围查询的两个阶段中，方向感知的路网扩展阶段消耗了绝大部分时间，而局部路网构建阶段的耗时较短，几乎可以忽略不计。这两个阶段的耗时也都随着查询范围半径的增加而增加。

范围查询时间随着查询范围半径的增加而增加，一方面是因为局部路网的规模增加导致了额外的开销。图 10 给出了局部路网规模随查询范围半径的变化情况，可以看出，当查询范围半径增大时，路网中节点和边的数量都快速增加，查询半径为 10km 的范围内局部路网的节点个数近 15000，边的数量高达 20000 条。可以预见，局部路网的规模越大，路网节点

到查询点最短路径的计算成本就越大，局部路网的扩展过程也就越漫长，这些因素都会导致查询时间的增加。

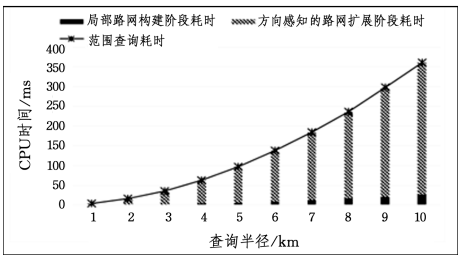

图 9 方向感知的路网移动对象范围查询时间随范围半径的变化情况

Fig. 9 Change of query time in direction-aware network moving object range query with range radius

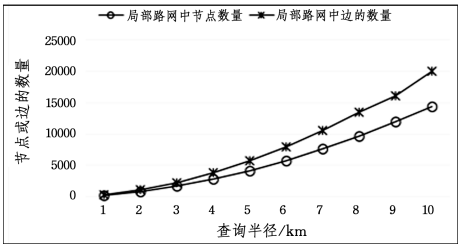

图 10 局部路网规模随查询范围半径的变化情况

Fig. 10 Change of scale of local road network with query range radius

另一方面，查询范围半径的增加导致了查询范围内移动对象数量的增加，从而造成查询时间的增加。图 11 给出了局部路网中移动对象随查询范围半径的变化情况。当查询范围半径增加时，局部路网中的候选移动对象数量快速增加，与此同时，朝向查询点的移动对象数量也相应增加。在查询范围半径为 10 km 的局部路网中共有将近 3000 个候选移动对象，其中朝向查询点的移动对象近 1200 个。在路网扩展过程中，这些移动对象都需要被计算一遍，因此造成了查询时间成本的上升。

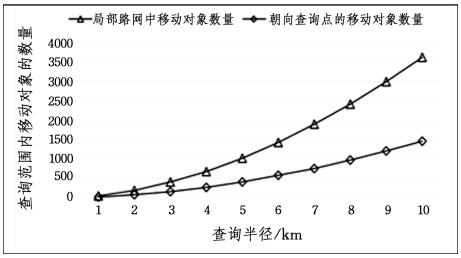

图 11 局部路网中移动对象数量随查询范围半径的变化情况

Fig. 11 Change of moving objects in local network with query range radius

(2)网格划分数量对查询性能的影响

图 12 给出了在拥有 50000 个移动对象的 Los Angeles 路网中执行一次 5 km 范围查询所需的查询时间随网格划分数量的变化情况。其中折线图表示范围查询整体的时间消耗情况，柱状图部分表示范围查询各个阶段的时间消耗情况。从折线图中可以看出，当网格划分数量仅为 200 * 200 时，范围查询的耗时相对较高，为 650 ms 左右，随着网格划分数量的增加，范围查询的时间消耗下降得很快，但是当网格划分数量增加到 1000 * 1000 后，范围查询的时间消耗下降非常缓慢。从柱状图中可以看出，网格划分数量对方向感知的路网扩展阶段的影响较大，对局部路网构建阶段的影响并不大，这是因为局部路网构建阶段几乎不需要与网格索引交互，因此该阶段对不同的网格划分并不敏感。而方向感知的路网扩展阶段在路网扩展过程中需要利用路网移动对象映射算法获取路网边上的移动对象，网格划分数量越少，网格就越大，每个网格容纳的移动对象就越多，这将导致获取路网边上移动对象的时间成本大大增加。

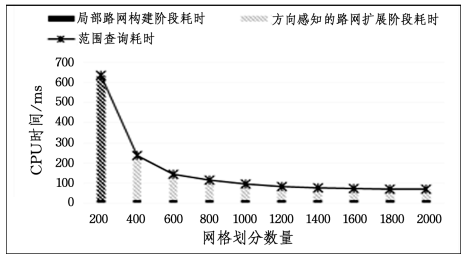

图 12 范围查询性能随网格划分数量的变化情况

Fig. 12 Change of performance of range query with the number of grid partition

但是网格划分数量也不是越多越好，当网格划分数量增加时，一方面会导致创建网格索引的时间成本上升，另一方面会导致内存的使用量急剧增加，实验结果如图 13 所示。因此在内存资源相对有限的情况下，可以适当减少网格划分数量，比如 800 * 800 个～1200 * 1200 个网格数量是一个比较合适的范围。

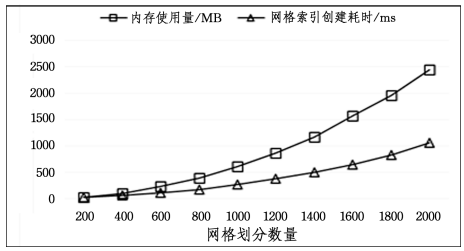

图 13 内存使用量及索引创建时间随网格划分数量的变化情况

Fig. 13 Change of memory usage and index creation with the number of grid partition

图 14 给出了局部路网构建阶段查询时间随网格划分数量的变化情况。其中柱状图部分为局部路网构建阶段中欧氏范围查询和单源最短路径计算的耗时情况，无论网格划分数量如何变化，欧氏范围查询耗时均为 3 ms 左右，单源最短路径计算耗时均为 2 ms 左右，从而导致局部路网构建阶段耗时总是稳定在 5 ms 左右，如折线图所示。

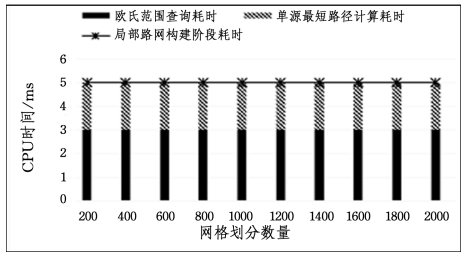

图 14 在局部路网构建阶段查询时间随网格划分数量的变化情况

Fig. 14 Change of query time in construction phase of local road network with the number of grid partition

图 15 给出了方向感知的路网扩展阶段查询时间随网格划分数量的变化情况。从柱状图部分可以看出，随着网格划分数量的增加，路网移动对象映射耗时下降，而方向判定及路网距离计算耗时几乎没有变化。当网格划分数量在 200 * 200 到 800 * 800 区间时，路网移动对象映射耗时下降最快。这是因为随着网格划分粒度不断变小，平均每条路网边所覆盖的有效网格范围不断减小，在移动对象数量不变的情况下，有效网格内移动对象的数量不断下降，路网移动对象映射的耗时也相应地不断下降。当网格划分数量在 1000 * 1000 以上时，路网移动对象映射耗时下降得非常缓慢。这是因为在移动对象数量不变的情况下，随着网格划分粒度继续变小，出现了很多空的网格，此时即使平均每条路网边所覆盖的有效网格范围减小了，但是有效网格内的移动对象数量并没有明显的下降，因此路网移动对象映射所需的时间下降得很缓慢。

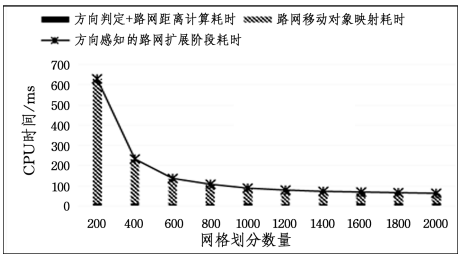

图 15　在路网扩展阶段查询时间随网格划分数量的变化情况

Fig. 15　Change of query time in network expansion phase with number of grid partition

(3)移动对象的数量 n 对查询性能的影响

图 16 描述了在 Los Angeles 路网中执行一次 5 km 范围查询所需的查询时间随全局路网中移动对象数量的变化情况。从图中可以看出，随着全局路网中移动对象数量的增加，范围查询所需的时间成本上升。图 16 中柱状图部分表明局部路网构建阶段对路网中移动对象数量的变化并不敏感，而方向感知的路网扩展阶段的查询时间则随着移动对象数量的增加而稳定上升。

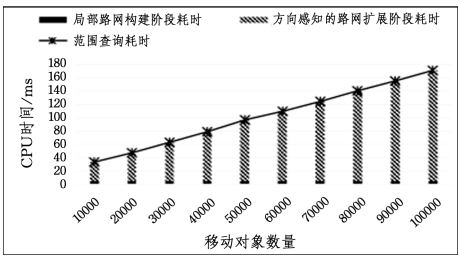

图 16　路网扩展阶段查询时间随移动对象数量的变化情况

Fig. 16　Change of query time in the expansion phase of road network with number of moving objects

在局部路网构建阶段，欧氏范围查询仅仅从网格索引中获取查询范围内的所有移动对象，这个操作非常快速，而单源最短路径计算则与移动对象没有任何关系，因此这个阶段在移动对象数量增加时查询性能不会出现退化。在路网扩展阶段，随着全局路网中移动对象数量的不断增加，局部路网中移动对象的数量也相应增加，这就导致在路网扩展过程中需要处理更多的移动对象。

图 17 给出了 5 km 查询范围内移动对象数量随全局路网移动对象数量的变化情况。可以看出，当全局路网移动对象数量增加时，局部路网中的移动对象数量相应增加，当然，朝向查询点的移动对象数量也随之增加。

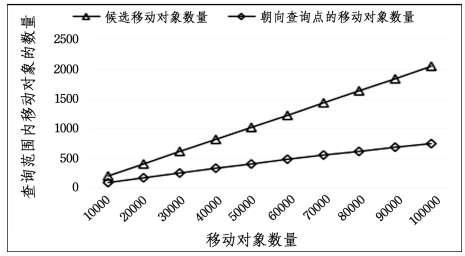

图 17　5 km 查询范围内移动对象数量随路网移动对象数量的变化情况

Fig. 17　Change of number of moving objects within 5 km range with number of moving objects in road network

图 18 给出了在方向感知的路网扩展阶段，查询时间随路网移动对象数量的变化情况。可以看出，随着移动对象数量的增加，路网移动对象映射耗时逐渐增加，移动对象的方向判定和路网距离计算的耗时几乎不变。其中路网移动对象映射耗时几乎占用了方向感知的路网扩展阶段的全部查询时间，而移动对象的方向判定和路网距离计算耗时较短。在网格划分数量不变的情况下，移动对象数量增加会导致平均每个网格所包含的移动对象数量增加，从而使得路网边覆盖的有效网格范围内的移动对象数量增加，因此，路网移动对象映射所需的时间随之增加。移动对象的方向判定和路网距离计算耗时几乎不变的原因是，它们都是非常高效的计算过程，即使移动对象数量有所增加，也几乎不会带来额外的时间开销。

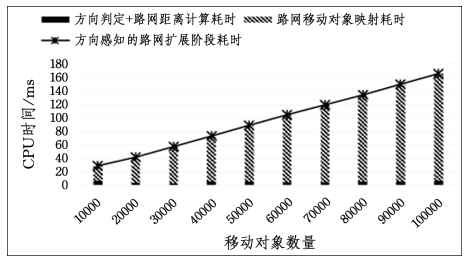

图 18　在路网扩展阶段查询时间随路网移动对象数量的变化情况

Fig. 18　Change of query time with number of moving objects in road network during the expansion phase

5.3　基于 INE 的范围查询算法与本文算法的实验对比与分析

基于 INE 的范围查询算法和本文提出的方向感知的路网移动对象范围查询算法都可用于处理范围查询的问题。为了对比两个算法的异同之处，本文对两种查询算法在不同查询半径情况下的查询性能和查询结果集进行了对比与分析。

图 19 给出了随着查询范围半径的增加，基于 INE 的范围查询和本文提出的方向感知的路网移动对象范围查询的耗时情况。两种范围查询算法的耗时随着查询范围半径的变大而呈上升趋势。这是由于随着查询半径的不断增加，范围内路网的节点、边以及移动数量都会增多，导致构建局部路网的规模变大，过滤局部路网中符合条件的移动对象的计算成本

增多，因此随着查询半径的不断增大，两种算法的耗时都呈上升趋势。但这两种算法在处理查询范围时，所消耗的时间几乎相同，在查询半径为 10km 的情况下，基于 INE 算法范围查询的耗时为 360ms，方向感知的路网移动对象范围查询的耗时为 369ms，两种算法所花费的查询时间差别很小，由此可见，利用朝向查询点的路网移动对象判定方法非常高效，基本不影响整体的范围查询效率。

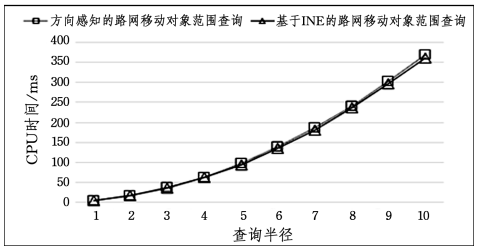

图 19 基于 INE 的范围查询和方向感知的范围查询随不同查询半径的耗时对比

Fig. 19 Comparison of query time based on INE and direction-aware with different range radius

为了分析两种算法查询结果集的差异，本文对不同查询半径的查询结果集进行了对比分析。图 20 描述了基于 INE 的范围查询和方向感知的范围查询在不同查询半径时查询到的移动对象数量。从图 20 中可以看出，随着查询半径的增大，这两种范围查询算法的移动对象结果集数量快速增多。这主要是因为查询半径越大，查询范围内的移动对象数量就越多。此外，基于 INE 的范围查询检索出的移动对象数量比方向感知的范围查询检索出的移动对象数量多。这主要是因为基于 INE 的范围查询算法检索出的移动对象未考虑方向约束，没有过滤掉远离查询点运动的移动对象。以查询范围半径为 10km 为例，基于 INE 的范围查询检索出的路网距离小于 10km 的移动对象数量为 1825 个，而方向感知的范围查询算法获得范围内的移动对象数量为 815 个。基于 INE 的范围查询仅检索出路网距离小于查询半径的移动对象结果集，结果中有 1010 个移动对象都不满足方向约束条件，因此基于 INE 的范围查询的对象结果集不能满足人们对方向感知的需求。从 1～10km 的范围查询结果可以看出，本文提出的方向感知的路网移动对象范围查询算法能够过滤掉不符合方向约束条件的移动对象，找出范围内朝向查询点的移动对象结果集。

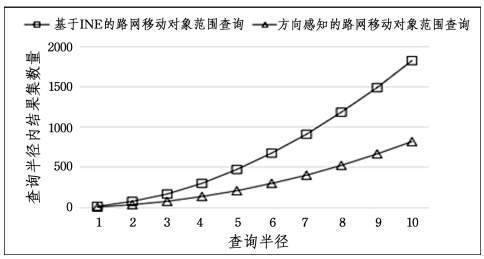

图 20 基于 INE 的范围查询和方向感知的范围查询的结果集对比

Fig. 20 Comparison of range query result sets based on INE and direction-aware range query

结束语 本文提出了方向感知的路网移动对象范围查询算法，可高效地查找出路网范围内朝向查询点运动的移动对象。方向感知的路网移动对象范围查询算法分为两个阶段，分别是局部路网构建和方向感知的路网扩展。局部路网构建阶段在大规模路网中执行欧氏范围查询，构建范围查询范围内的局部路网，并采用 Dijkstra 算法预计算路网节点到查询的路网距离；方向感知的路网扩展利用朝向查询点的路网移动对象判定方法，在 INE 算法的基础上进行增量式的方向判定，高效地实现了复杂路网下移动对象的方向感知。为了加快查询处理过程，特别是在处理大规模路网和移动对象的情况下，本文采用了 R-tree 结合简单网格索引的双索引方案，其中采用 R-tree 索引静态的路网数据，利用简单网格索引动态的移动对象。由于移动对象和路网对象是分开索引的，因此对于获取边上的移动对象，首先获取被路网边覆盖的有效网格，然后采用最小垂直距离判定法反向验证每个移动对象是否在给定的边上，在保证准确性的条件下高效实现路网移动对象的获取。

为了评估查询算法的性能，本文从查询范围、移动对象数量以及网格划分数量等方面进行实验，结果表明，方向感知的路网移动对象范围查询算法在合理的参数范围内能取得不错的查询性能。

参 考 文 献

[1] GUTTMAN A. R-trees: a dynamic index structure for spatial searching[C]// International Conference on Management of Data. 1984: 47-57.

[2] SELLIS T, ROUSSOPOULOS N, FALOUTSOS C, et al. The R^+-Tree: A Dynamic Index for Multi-Dimensional Objects[C]// International Conference on Very Large Data Bases. 1987: 507-518.

[3] BECKMANN N, KRIEGEL H, SCHNEIDER R, et al. The R^*-tree: an efficient and robust access method for points and rectangles[J]. ACM Sigimod Record, 1990, 19(2): 322-331.

[4] KAMEL I, FALOUTSOS C. Hilbert R-tree: An Improved R-tree using Fractals[C]// International Conference on Very Large Data Bases. 1994: 500-509.

[5] ARGE L, DE BERG M, HAVERKORT H J, et al. The Priority R-Tree: A Practically Efficient and Worst-Case-Optimal R-Tree [C]// Proceedings of SIGMOD International Conference on Management of Data. 2004: 347-358.

[6] KORN F, SIDIROPOULOS N, FALOUTSOS C, et al. Fast Nearest Neighbor Search in Medical Image Databases[C]// International Conference on Very Large Data Bases. 1996: 215-226.

[7] BERCHTOLD S, ERTL B, KEIM D A, et al. Fast nearest neighbor search in high-dimensional space[C]// International Conference on Data Engineering. 1998: 209-218.

[8] ZHENG B, LEE D L. Semantic Caching in Location-Dependent Query Processing[C]// Symposium on Large Spatial Databases. 2001: 97-113.

[9] SHARIFZADEH M, SHAHABI C. VoR-Tree: R-trees with Voronoi Diagrams for Efficient Processing of Spatial Nearest

Neighbor Queries[C]// Proceedings of the Vldb Endowment. 2010:1231-1242.

[10] JENSEN C S,LIN D,OOI B C. Query and update efficient B+-tree based indexing of moving objects[C]// Proceedings of the Thirtieth International Conference on Very Large Data. 2004:768-779.

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

[12] KOLAHDOUZAN M,SHAHABI C. Voronoi-based k nearest neighbor search for spatial network databases[C]// Proceedings of the Thirtieth International Conference on Very Large Data Bases. 2004:840-851.

[13] HUANG X,JENSEN C S,ŠALTENIS S. The Islands Approach to Nearest Neighbor Querying in Spatial Networks[C]// International Conference on Lecture Notes in Computer Science. 2006,3633:73-90.

[14] HUANG X,JENSEN C S,LU H,et al. S-GRID:A versatile approach to efficient query processing in spatial networks[M]// Advances in Spatial and Temporal Databases. Springer Berlin Heidelberg,2007:93-111.

[15] HONG S,CHANG J. A new k-NN query processing algorithm based on multicasting-based cell expansion in location-based services[J]. Journal of Convergence,2013,4(4):1-6.

[16] WANG H,ZIMMERMANN R. Location-based query processing on moving objects in road networks[C]// Proceedings of International Conference on Very Large Data Bases (VLDB 2007). 2007:321-332.

[17] PATROUMPAS K,SELLIS T. Monitoring Orientation of Moving Objects around Focal Points[C]// Symposium on Large Spatial Databases. 2009:228-246.

[18] LI G,FENG J,XU J. Desks:Direction-aware spatial keyword search[C]// IEEE 28th International Conference on Data Engineering (ICDE). 2012:474-485.

[19] YI S,RYU H,SON J,et al. View field nearest neighbor:A novel type of spatial queries[J]. Information Sciences,2014,275(3):68-82.

[20] LEE M J,CHOI D W,KIM S Y,et al. The direction-constrained k nearest neighbor query[J]. GeoInformatica,2016,20(3):471-502.

[21] LU B L,CUI X Y,LIU N. Bichromatic Reverse Nearest k Neighbor Query Processing on Road Network[J]. Journal of Chinese Mini-Micro Computer Systems,2015,36(2):266-270. (in Chinese)
卢秉亮，崔晓玉，刘娜. 路网中双色反向 k 近邻查询处理[J]. 小型微型计算机系统，2015,36(2):266-270.

[22] LEE K W,CHOI D W,CHUNG C W. Dart:An efficient method for direction-aware bichromatic reverse k nearest neighbor queries[M]// Advances in Spatial and Temporal Databases. Springer Berlin Heidelberg,2013:295-311.

[23] LI G L,FENG J H,XU J. DESKS:Direction-aware spatial keyword search[C]// 2012 IEEE 28th International Conference on Data Engineering. Washington:IEEE,2012:474-485.

[24] BRINKHOFF T. A framework for generating network-based moving objects[J]. GeoInformatica,2002,6(2):153-180.

(上接第 198 页)

[18] HU H,ZHANG H,XUAN J,et al. Effective Bug Triage Based on Historical Bug-Fix Information[C]// Proceedings of the 25th IEEE International Symposium on Software Reliability Engineering. New York:IEEE Press,2014:122-132.

[19] YAN M,ZHANG X H,YANG D,et al. A Component Recommender for Bug Reports Using Discriminative Probability Latent Semantic Analysis[M]. Butterworth-Heinemann,2016,73:37-51.

[20] ZHANG W,WANG S,WANG Q. KSAP:An Approach to Bug Report Assignment Using KNN Search and Heterogeneous Proximity [J]. Information and Software Technology,2016,70:68-84.

[21] XIA X,LO D,WANG X,et al. Dual Analysis for Recommending Developers to Resolve Bugs [J]. Journal of Software:Evolution and Process,2015,27(3):195-220.

[22] BHATTACHARYA P,NEAMTIU I,SHELTON C R. Automated, Highly-Accurate, Bug Assignment Using Machine Learning and Tossing Graphs [J]. Journal of Systems and Software,2012,85(10):2275-2292.

[23] MIKOLOV T,SUTSKEVERI,CHEN K,et al. Distributed Representations of Words and Phrases and their Compositionality [C]// Proceedings ofthe 27th Annual Conference on Neural Information Processing Systems. La Jolla:Neural Information Processing Systems Foundation,2013:3111-3119.

[24] GAN J,CHEN L C. Research of improved IF-IDF Weighting algorithm[C]// Proceedings of the 2nd International Conference on Information Science and Engineering. New York:IEEE Press,2011:2304-2307.

[25] LILLEBERG J,ZHU Y,ZHANG Y. Support vector machines and word2vec for text classification with semantic features[C]// Proceedings of the 14th IEEE International Conference on Cognitive Informatics & Cognitive Computing. New York:IEEE Press,2015:136-140.

[26] CHANG C C,LIN C J. LIBSVM:a library for support vector machines[J]. ACM Transactions on Intelligent Systems and Technology,2011,2(3):1-27.

[27] RONG X. word2vec parameter learning explained [EB/OL]. https://arXiv.org/abs/1411.2738.

[28] GOLDBERG Y,LEVY O. word2vec explained:deriving mikolov et al. negative-sampling word-embedding method[EB/OL]. https://arXiv.org/abs/1402.3722.