软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2023,34(6):2789−2803 [doi: 10.13328/j.cnki.jos.006492]

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

面向移动对象连续 *k* 近邻查询的双层索引结构*

韩士元[1], 何 清[1], 于自强[2], 童向荣[2], 郑渤龙[3]

[1](济南大学 信息科学与工程学院, 山东 济南 250022)

[2](烟台大学 计算机与控制工程学院, 山东 烟台 264005)

[3](华中科技大学 计算机科学与技术学院, 湖北 武汉 430074)

通信作者: 于自强, E-mail: zqyu@ytu.edu.cn

摘 要: 移动对象连续 k 近邻 (CKNN) 查询是指给定一个连续移动的对象集合, 对于任意一个 k 近邻查询 q, 实时计算查询 q 的 k 近邻并在查询有效时间内对查询结果进行实时更新. 现实生活中, 交通出行、社交网络、电子商务等领域许多基于位置的应用服务都涉及移动对象连续 k 近邻查询这一基础问题. 已有研究工作解决连续 k 近邻查询问题时, 大多需要通过多次迭代确定一个包含 k 近邻的查询范围, 而每次迭代需要根据移动对象的位置计算当前查询范围内移动对象的数量, 整个迭代过程的计算代价占查询代价的很大部分. 为此, 提出了一种基于网络索引和混合高斯函数移动对象分布密度的双重索引结构 (grid GMM index, GGI), 并设计了移动对象连续 k 近邻增量查询算法 (incremental search for continuous k nearest neighbors, IS-CKNN). GGI 索引结构的底层采用网格索引对海量移动对象进行维护, 上层构建混合高斯模型模拟移动对象在二维空间中的分布. 对于给定的 k 近邻查询 q, IS-CKNN 算法能够基于混合高斯模型直接确定一个包含 q 的 k 近邻的查询区域, 减少了已有算法求解该区域的多次迭代过程; 当移动对象和查询 q 位置发生变化时, 进一步提出一种高效的增量查询策略, 能够最大限度地利用已有查询结果减少当前查询的计算量. 最后, 在滴滴成都网约车数据集以及两个模拟数据集上进行大量实验, 充分验证了算法的性能.

关键词: 移动对象; 连续 k 近邻查询 (CKNN); 增量查询算法

中图法分类号: TP311

中文引用格式: 韩士元, 何清, 于自强, 童向荣, 郑渤龙. 面向移动对象连续*k*近邻查询的双层索引结构. 软件学报, 2023, 34(6): 2789–2803. http://www.jos.org.cn/1000-9825/6492.htm

英文引用格式: Han SY, He Q, Yu ZQ, Tong XR, Zheng BL. Double Layer Index for Continuous *k*-nearest Neighbor Queries on Moving Objects. Ruan Jian Xue Bao/Journal of Software, 2023, 34(6): 2789–2803 (in Chinese). http://www.jos.org.cn/1000-9825/6492.htm

Double Layer Index for Continuous *k*-nearest Neighbor Queries on Moving Objects

HAN Shi-Yuan[1], HE Qing[1], YU Zi-Qiang[2], TONG Xiang-Rong[2], ZHENG Bo-Long[3]

[1](School of Information Science and Engineering, Jinan University, Jinan 250022, China)

[2](School of Computer and Control Engineering, Yantai University, Yantai 264005, China)

[3](School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract: For a given set of moving objects, continuous k-nearest neighbor (CKNN) query q over moving objects is to quickly identify and monitor the k-nearest objects as objects and the query point evolve. In real life, many location-based applications in transportation, social network, e-commerce, and other fields involve the basic problem of processing CKNN queries over moving objects. Most of existing work processing CKNN queries usually needs to determine a query range containing k-nearest neighbors through multiple

* 基金项目: 国家自然科学基金 (62172351, 62072392, 61903156, 61873324); 山东省自然科学基金重点项目 (ZR2020KF006)
收稿时间: 2021-04-20; 修改时间: 2021-07-17; 采用时间: 2021-09-14; jos 在线出版时间: 2022-01-28
CNKI 网络首发时间: 2022-11-15

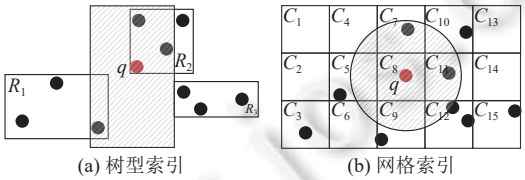
iterations, while each iteration has to identify the number of objects in the current query range, and which dominates the query cost. In order to address this issue, this study proposes a dual index called GGI that consists of a grid index and a Gaussian mixture function to simulate the varying distribution of objects. The bottom layer of GGI employs a grid index to maintain moving objects, and the upper layer constructs Gaussian mixture model to simulate the distribution of moving objects in two-dimensional space. Based on GGI, an incremental search algorithm called IS-CKNN to process CKNN queries. This algorithm directly determines a query region that at least contains k neighbors of q based on Gaussian mixture model, which greatly reduces the number of iterations. When the objects and query point evolve, an efficient incremental query strategy is further proposed, which can maximize the use of existing query results and reduce the calculation of the current query. Finally, extensive experiments are carried out on one real dataset and two synthetic datasets to confirm the superiority of the proposed proposal.

Key words: moving object; continuous k-nearest neighbor query (CKNN); incremental search algorithm

移动对象连续 k 近邻查询是指在二维空间给定一个移动对象集合和查询点 q, 随着移动对象和查询点位置不断变化, 实时计算并更新查询 q 的 k 近邻. 近年来, 随着无线网络和移动终端迅猛发展, 生活中人、车等对象随着位置频繁变化不断产生大量位置数据, 催生出许多基于位置的应用服务, 如基于用户位置的兴趣点推荐、地图搜索、路线导航等. 在这些应用服务中, 面向二维空间大规模移动对象的连续 k 近邻查询是一个基本问题. 例如, 社交软件 (如“微信”) 中每个用户的位置经常变化, 那么为移动用户查找“附近的人”则可以抽象为移动对象的连续 k 近邻查询问题. 在出行服务中, 打车软件为行驶的出租车寻找附近乘客, 也可以抽象成移动对象的连续 k 近邻查询.

本文的 k 近邻查询问题中, 查询点和移动对象连续移动, 且移动方向和速度不做任何限制. 为此, 本文所采取的语义是基于 t_i 时刻的数据快照计算 q 的 k 近邻查询结果, 且该结果在 t_{i+1} (t_{i+1}=t_i+Δt) 时刻是有效的, Δt 表示算法查询时间. 该语义中, 为保证查询结果时效性, 应尽可能减少Δt, 故本文采用基于内存的索引结构和查询算法.

近年来, 移动对象连续 k 近邻查询问题被国内外学者广泛研究, 已有的研究工作在解决连续 k 近邻查询问题时, 大多采用迭代思想确定查询区域, 每次迭代要计算查询区域内移动对象个数. 该过程中, 判断与查询区域有部分重叠区域的索引单元内移动对象是否被查询区域覆盖是一个较为耗时的操作, 该时间复杂度为 O(n), n 为需要判断的移动对象数量. 在树型索引结构中, 移动对象被最小边界矩形 (minimum bounding rectangle, MBR) 包裹, 计算查询区域内的移动对象数量包括两个部分: (1) MBR 被查询区域完全覆盖, MBR 内所有移动对象都属于查询区域. (2) MBR 部分区域被查询区域覆盖, 需要检查 MBR 中每个移动对象是否属于查询区域. 该检查过程需要比较每个移动对象位置和查询区域范围, 移动对象数量越多, 计算时间越大. 例如在图 1(a) 中为了确定查询 q 查询区域 (图中阴影部分) 中移动对象数量, 需要检查 R_1, R_2 中所有移动对象来确定 R_1, R_2 内的每个移动对象是否位于查询区域内. 此外, 基于网格索引的算法也存在该问题. 对于被查询区域部分覆盖的网格, 需要检查网格内每个移动对象, 来确定该对象是否属于查询区域. 如图 1(b) 中需要检查网格 C_5, C_7, C_9, C_{10}, C_{11}, C_{12} 来确定网格内的移动对象是否位于查询区域内.



(a) 树型索引 (b) 网格索引

图 1 查询范围示意图

针对上述问题, 我们通过混合高斯模型模拟移动对象分布, 在模型上对查询区域积分获得移动对象落入该区域的概率, 该概率乘以移动对象总量就可以较为准确的获得查询区域内的移动对象数量, 进而确定查询范围. 此过程中, 时间代价主要为计算积分的时间, 该时间与移动对象个数无关, 时间复杂度为 O(1), 因此我们的方法计算时间更短, 比现有方法更优, 在后续实验中也验证了这一点. 具体而言, 本文采用网格索引和混合高斯函数对海量移动对象建立双层索引 (grid GMM index, GGI), 利用上层索引混合高斯函数快速计算一个较为准确的查询区域, 然后基于底层网格索引在该区域内精确查找查询点的 k 近邻. 在双层索引基础上, 本文提出一种面向大规模移动对

象连续 k 近邻查询的增量查询算法 (incremental search for continuous k nearest neighbors, IS-CKNN). 在 IS-CKNN 算法中, 根据上层混合高斯模型快速估算某查询区域落入移动对象的概率, 以此概率乘以移动对象总量, 近似获得该区域移动对象数量, 从而为查询 q 快速确定候选查询区域, 该区域在很大程度上包含了查询 q 的 k 近邻. 然后在网格索引中精确查找 k 近邻.

本文的主要贡献可归结为以下几点.

● 提出一种模拟二维空间大规模移动对象动态分布的混合高斯模型. 基于该模型, 本文通过对某个区域进行积分操作就能够快速估算移动对象落入该区域的概率, 进而获得区域内的移动对象数量, 对于候选查询区域的快速计算极为重要.

● 提出一种基于网格索引和混合高斯函数的双层索引结构, 提出了连续 k 近邻增量查询算法, 能够充分利用最近一次查询的中间结果, 实现对查询结果的实时增量更新.

● 在多个真实和仿真数据集上对算法性能进行充分验证, 实验结果表明, 本文所设计的移动对象连续 k 近邻增量查询算法的性能较已有算法有显著提高.

1 相关工作

移动对象查询主要问题是提高查询效率, 核心问题在于移动对象索引技术和移动对象查询处理算法. 移动对象的索引一直是研究热点, 早年有学者提出 R-Tree 索引结构, 将空间内的对象根据所在区域进行分组, 任意一组对象用覆盖这组对象的最小矩形表示, 一个最小矩形 (MBR) 代表树中的一个节点, 该索引能对多维空间数据进行有效查询, 但是 R-Tree 节点之间存在重叠, 导致查询效率有所降低. 所以, 众多学者对 R-Tree 索引结构进行改进, 提出多种 R-Tree 的变体索引结构, 如 R*-Tree、TPR-Tree 等. Zheng 等人[1]在 TPR-Tree 基础上引入多时间版本概念, 解决移动对象全时态查询问题.

虽然上述方法都改进了 R-Tree 结构, 但在面临判断被查询范围覆盖的索引单元内移动对象是否属于查询范围的问题, 并没有给出解决方案.

Zhong 等人[2]设计了 G-Tree 索引, 将路网递归划分为子网络并建立树结构, 加入距离矩阵快速计算节点之间的最短路径. Shen 等人[3]提出一种 V-Tree 方法, 与 G-Tree 类似, 通过 V-Tree 与节点维护的距离矩阵加速最短路径距离的计算. Zheng 等人[4]提出一种 PM 树索引结构, 开发出一个可调节置信区间保证结果正确.

除了树型索引外, 泰森多边形索引结构也是解决多维空间查询问题的常见索引结构. 泰森多边形索引结构在对象集合中选取对象作为基准点, 每个基准点对应一个区域, 区域中任意一个对象与该基准点的距离小于该对象到其他基准点的距离, 由此形成的区域就是泰森多边形. Nutanong 等人[5]基于泰森多边形索引结构划分空间, 定义泰森多边形中的一组对象为安全区域, 对安全区域进行增量拓展, 利用已有结果实现连续 k 近邻查询. Hu 等人[6]利用泰森多边形解决 k 近邻问题时, 先找到查询所在区域的控制点 p 作为最近邻, 然后依此寻找 k 近邻. Zhu 等人[7]提出一种基于网格的泰森多边形索引 VO-Grid, 并设计一种剪枝策略过滤和减少冗余对象的访问. 倪巍伟等人[8]基于泰森多边形与 R*提出一种隐私保护的分段式近邻查询处理策略.

基于泰森多边形的索引结构虽然不存在此类问题, 但是这种索引结构维护代价巨大, 并不能很好的应对位置持续发生变化的移动对 k 近邻查询.

除了上述索引类型外, 基于空间网格的索引结构近年来也成为研究热点. Šidlauskas 等人[9]提出了均匀网格索引 P-Grid, 将空间区域划分成完全相等的网格, 移动对象根据自身坐标归于不同单元网格, 每个单元网格指向存储移动对象信息的数据列表. Kumar 等人[10]在 P-Grid 索引基础上, 集成 Hibert 曲线解决多维数据查询问题. Xu 等人[11]考虑速度信息对查询的影响, 提出双空间网格索引结构. Yang 等人[12]提出水桶 PR 四叉树索引, 能快速确定搜索范围. Yi 等人[13]提出一种以网格中心建立候选 k 近邻查询目标的算法. Li 等人[14]设计了一种允许控制精度和速度的算法, 自适应数据密度变化, 支持动态更新数据集.

除了上述空间网格索引, 众多研究人员基于网格索引又提出若干种连续 k 近邻查询算法[15,16], 这些方法在应对欧式距离下的查询非常有效. Yu 等人[15]提出 YPK-CNN 算法, 由查询点位置向外扩展式地查询 k 近邻, 以查询

点为中心, 查询点到第 k 个近邻的距离为半径画圆, 利用圆形区域确定查询的 k 近邻. Mouratidis 等人[16]提出的 CPM 算法与 YPK-CNN 算法类似, 不同的是 CPM 引入一种特殊的网格划分方式, 改进算法的查询效率.

Hua 等人[17]讨论了基于欧式距离与基于路网距离最近邻查询的不同以及基于路网距离是否比基于欧式距离的结果更可靠的问题. Nutanong 等人[18]对近几年基于欧式空间的研究进行总结.

Miao 等人[19]提出一种具有方向约束的新型空间数据查询. Lee 等人[20]提出一种新的算法, 其核心思想是预先找到每个静态对象的 k 近邻, 利用距离查询点 q 最近的对象的 k 近邻查找 q 的 k 近邻. 刘佳[21]针对路网环境提出一种多源最近邻的查询方法. Cheema 等人[22]提出安全区域的概念, 当查询与移动对象不离开自身安全区域, 即可快速计算. Li 等人[23]针对安全区域计算代价高的问题提出安全组概念, 如果 k 近邻比安全组的对象距离查询更近, 则它们是有效的. Zheng 等人[24]在研究 k 近邻查询时, 考虑用户偏好, 关键字等信息. Wang 等人[25]通过隐式地学习用户偏好, 减小结果集的规模. Li 等人[26]基于 GLAD 开发一个基于网格的面向被占用移动对象的索引结构进行可接近的 k 近邻查询, 并提出面向冲突的查询问题. Lee 等人[27]在面对大数据下的查询时, 基于 GeoHash 提出一种轻量级分层索引, 可以很好用于现有系统, 并根据编码前缀过滤并查询相近空间对象. 鲍金玲等人[28]与冯钧等人[29]全面分析并比较了路网环境下的最近邻查询技术, 对相关技术进行总结.

现有的一些研究工作中, 在判断与查询区域有部分重叠区域的索引单元内移动对象是否被查询区域覆盖时, 仍然需要遍历索引单元内的每个移动对象. 一些学者通过预计算的方式建立候选查询集, 避免了查询时的计算消耗, 但在预计算时, 遍历过程仍然存在, 因此计算成本减少的并不多.

2 双层索引结构 (GGI)

双层索引结构是由两层索引构成, 下层索引是网格索引, 每个网格记录该网格内所有移动对象. 上层索引是由所有移动对象聚类形成的混合高斯模型, 模型由多个单高斯模型组成, 能够近似模拟移动对象的分布情况.

2.1 构建网格索引结构

对于给定总数为 N_p 的移动对象集合 P, IS-CKNN 算法采用网格索引结构 (grid-index) 对其建立索引. Grid-index 将整个平面区域划分成若干相同大小的网格, 任意网格 g_l 维护一个移动对象列表 OL_l, OL_l 记录 g_l 中所有移动对象并实时更新. 由于网格索引中移动对象更新过程较为简单, 且相关工作已有介绍, 本文不再详细讨论.

2.2 混合高斯模型数学表示

混合高斯模型 (Gaussian mixture model, GMM) 是由多个单高斯模型 (single Gaussian model, SGM) 合成, 每个单高斯模型对应移动对象的一个聚类, 所有单高斯概率密度函数的加权和就是混合高斯概率密度函数. 本文使用符号及其含义如表 1 所示.

表 1 符号及含义

符号	符号含义	符号	符号含义
P	移动对象集合	RR_q	q的查询范围
N_p	集合P中的移动对象个数	p	RR_q范围内落入移动对象的概率
o_i	第i个移动对象	r	RR_q范围半径
q	查询q	Δr	RR_q半径增量

利用单高斯模型:

$$N(o_i;\mu;\sigma)=\frac{\exp\left[-\frac{1}{2}(o_i-\mu)^{\mathrm{T}}\sigma^{-1}(o_i-\mu)\right]}{\sqrt{(2\pi)^d|\sigma|}} \tag{1}$$

对移动对象进行聚类时, 基本思想是由每个单高斯模型计算任意一个对象属于该聚类的概率, 然后选取概率最大的聚类, 并将该对象添加到该类. 其中 μ 为聚类中心, σ 为聚类协方差. 由此构造单高斯模型, 计算各个聚类的

权值 φ. 设某一移动对象属于聚类 C_j, 那么, 单高斯模型可以改为:

$$N(o_i;C_j)=\frac{\exp\left[-\frac{1}{2}(o_i-\mu)^{\mathrm{T}}\sigma^{-1}(o_i-\mu)\right]}{\sqrt{(2\pi)^d|\sigma|}} \tag{2}$$

所以混合高斯模型 (包含 m 个单高斯模型) 为:

$$G(o_i)=\sum_{j=1}^{m}\varphi_j\times N(o_i;C_j) \tag{3}$$

2.3 构建混合高斯模型

对移动对象集合构建混合高斯模型, 即该集合的概率密度函数, 并且集合中所有移动对象均为独立同分布. 集合中任意一个移动对象落入任意区域的概率就是混合高斯模型在该区域上的积分. 将移动对象落入查询区域的概率乘以移动对象总数, 可以近似获得该区域内移动对象的数量, 如果该区域内的移动对象数量满足查询要求, 则该区域作为候选查询区域, 遍历区域内的移动对象, 从中找到最终的 k 近邻. 因此, 对移动对象集合进行聚类, 构建混合高斯模型, 是为了通过积分快速获得某一区域内的移动对象数量.

IS-CKNN 算法首先采用 K-means 算法对所有移动对象进行初始聚类, 聚类时选择的是移动对象在某一时刻的快照, 并且已知移动对象的位置信息. 得到每个聚类的均值 μ 和协方差 σ 作为单高斯模型初始参数, 构建混合高斯模型, 使用 EM 算法修正参数更新混合高斯模型. 然而, 移动对象位置持续变化, 将影响每个单高斯模型的均值 μ 和协方差 σ, 同时也会影响每个单高斯模型在混合高斯模型中的权值 φ, 到达新时刻时, 移动对象分布可能已经不符合已有的混合高斯模型. 因此, IS-CKNN 算法采用 EM 极大似然估计算法对均值 μ、协方差 σ 和权值 φ 进行实时更新.

采用 EM 算法更新参数时, 首先计算各个对象在各个聚类内的概率密度, 假设每个聚类中所有对象集合表示为 S_j, $|S_j|$为聚类中对象的数目, 那么每个对象 o_i 在各个聚类 C_j 内的概率密度的似然函数为:

$$\beta_{ij}=\frac{\varphi_j N(o_i;\mu_j;\sigma_j)}{\sum_{j=1}^{m}\varphi_j N(o_i;\mu_j;\sigma_j)} \tag{4}$$

更新权值φ'_j、均值μ'_j、协方差σ'_j:

$$\varphi'_j=\frac{\sum_{i=1}^{N_p}\beta_{ij}}{N_p} \tag{5}$$

$$\mu'_j=\frac{\sum_{i=1}^{N_p}\beta_{ij}o_i}{N_p} \tag{6}$$

$$\sigma'_j=\frac{\sum_{i=1}^{N_p}\beta_{ij}(o_i-\mu'_j)(o_i-\mu'_j)^{\mathrm{T}}}{\sum_{i=1}^{N_p}\beta_{ij}} \tag{7}$$

不断地迭代以上步骤, 直到满足 $N'(o_i;\mu'_j;\sigma'_j)-N(o_i;\mu_j;\sigma_j)=\varepsilon$, 即两次迭代的结果差值小于给定阈值 ε 则终止迭代, 通常取 ε=10^{-5}, 由此获得的单高斯模型 $N'(o_i;C_j)$ 与实际分布较为近似. 利用更新后的权值φ'_j和 $N'(o_i;C_j)$ 和 $G(o_i)=\sum_{j=1}^{m}\varphi'_j\times N'(o_i;C_j)$, 可得与整个区域移动对象分布非常近似的混合高斯模型概率密度函数.

3 增量 k 近邻查询算法 (IS-CKNN)

基于 GGI 双层索引结构, 我们提出一种增量查询算法 (IS-CKNN). 给定一个 k 近邻查询, 首先划定一个查询区域, 在混合高斯模型上对该区域进行积分, 获得区域内落入移动对象的概率, 将此概率乘以移动对象总数, 进而获得区域内移动对象个数, 迭代此过程直到满足终止条件 $k+\lambda\leqslant p\times N_p\leqslant k+2\lambda$, 其中 p 为区域内落入移动对象的概率, N_p 为移动对象总个数, $p\times N_p$ 为区域内移动对象个数, λ 为允许的误差范围. 定义终止条件为 $k+\lambda\leqslant p\times N_p\leqslant k+2\lambda$ 是为了尽可能保证候选查询区域内的移动对象数量大于 k, 同时避免候选查询区域内的移动对象数量过多,

导致计算 k 近邻的时间代价增大. 此时, 查询区域为最终查询区域, 保存该区域并在底层网格索引中准确查询 q 的 k 近邻. 当查询对象发生移动后, 采用增量查询方式, 以上次初始查询的最终查询区域作为此次增量查询的候选查询区域, 重新计算查询区域内移动对象数量, 进而确定查询对象的 k 近邻.

3.1 确定查询范围

给定一个查询 q, 首先需要确定查询 q 的查询范围. 传统的方法中, 在迭代扩大查询范围时, 对被查询范围完全覆盖的网格, 可以直接得到网格内移动对象的数量, 而对于被查询范围部分覆盖的网格, 则需要对网格内所有移动对象进行遍历, 判断每一个移动对象是否位于查询范围内. 当查询范围扩大时, 每次新加入的被查询范围部分覆盖的网格都需要进行遍历操作. 这一过程的时间代价是相当大的并且受网格内移动对象数量的影响. 如果网格内存在大量移动对象, 则会浪费大量计算资源.

在 IS-CKNN 算法中, 对移动对象集合建立混合高斯模型, 获得概率密度函数, 一个移动对象落入任意区域的概率就是概率密度函数在该区域的积分. 因此, 基于混合高斯模型对任意区域积分, 可以获得一个移动对象落入该区域的概率, 以此概率乘以移动对象总数就可以获得该区域内移动对象个数, 进而依据移动对象个数是否满足要求确定查询区域. 当改变查询区域时, 只需要在新的查询区域下进行积分, 就可以获得移动对象落入新查询区域的概率, 进而获得新查询区域内移动对象个数. 采用这种方法确定查询区域的时间是常数级的, 并且不受网格内移动对象数量的影响.

如图 2 所示, 对于查询 q, 当查询范围内的移动对象数量大于 10 个时查询范围满足查询要求. 传统方式在计算初始查询区域 RR_1 内的移动对象个数时, RR_1 完全覆盖 C_{10} 网格, 因此 C_{10} 网格不需要遍历, 而 C_5, C_6, C_7, C_9, C_{13}, C_{14}, C_{15} 网格均需要遍历包含的移动对象来判断是否位于查询范围 RR_1 内. 因为 RR_1 查询范围内的移动对象不满足 10 个, 扩大查询范围到 RR_2, 此时 RR_2 完全覆盖 C_6, C_{10}, C_{11} 网格, 因此这 3 个网格不需要遍历, 而 C_2, C_3, C_5, C_7, C_8, C_9, C_{12}, C_{13}, C_{14}, C_{15}, C_{16} 网格仍然需要对网格包含的移动对象进行遍历, 确定移动对象是否位于查询范围 RR_2 内. 我们采用的方式只需要在 RR_1 范围内对混合高斯模型求积分, 得到移动对象落入 RR_1 范围内的概率, 再乘以移动对象总数获得 RR_1 范围内的移动对象个数. 当查询范围改变为 RR_2 时, 也只需要计算 RR_2 范围内混合高斯模型的积分, 然后乘以移动对象总数, 就能获得 RR_2 范围内的移动对象个数.

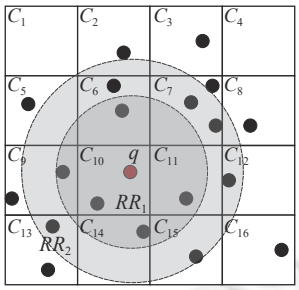


图 2 计算查询范围示意图

3.2 IS-CKNN 算法初始查询

基于上述思路构造出反映数据变化并且能够实时更新的混合高斯模型. 在混合高斯模型中给定一个区域, 该区间内混合高斯模型的函数曲面与 XOY 面形成的曲顶圆柱体积 (如图 3) 近似等于混合高斯模型在该区域的积分, 即移动对象落入该区域的概率, 将此概率乘以移动对象总数进而获得区域内移动对象数量, 确定查询区域. IS-CKNN 算法首先对查询 q 随机选择一个查询区域 RR_q, 在该查询区域 RR_q 内求积分, 结果记为 p, 那么可得区域 RR_q 内移动对象数目约为 $p\times N_p$. 如果 $k+\lambda \leqslant p\times N_p \leqslant k+2\lambda$ 成立, 则 RR_q 即为查询 q 的最终查询区域, 否则调整区域 RR_q, 直到 $k+\lambda \leqslant p\times N_p \leqslant k+2\lambda$ 成立, 获得查询 q 的最终查询区域.

算法 1 给出了 IS-CKNN 算法伪代码描述, 其中 o_i 表示一个移动对象的位置信息.

算法 1. IS-CKNN 算法.

输入: N_p, q, k, RR_q 的半径为 r, $N(o_i; C_j)$, $G(o_i)=\sum_{j=1}^{m}\varphi_j\times N(o_i;C_j)$;

输出: 查询 q 的 k 近邻.

1. 根据 $N(o_i;C_j)$ 计算 q 在各个单高斯模型的概率, 其中最大的概率即为 q 所在的聚类 C_j;

/*确定查询点为中心, 半径 r 范围内的移动对象个数*/

2. 根据 Get-Integration 算法计算区域 RR_q 内落入移动对象的概率为 p;

3. 计算区域 RR_q 内移动对象的数目 $p\times N_p$, RR_q 的半径为 r;

/*RR_q 区域内的移动对象个数是否满足阈值要求, 若不满足则调整半径范围重新计算查询范围内移动对象个数*/

4. **WHILE**$((k+\lambda\leqslant p\times N_p\leqslant k+2\lambda)$== false)

5. $r=r\pm\Delta r$;

6. 根据 Get-Integration 算法计算区域 RR_q 内落入移动对象的概率为 p;

7. 计算区域 RR_q 内移动对象数目 $p\times N_p$;

8. $RR_q=\pi\times r^2$;

9. **END WHILE**

10. 基于 RR_q 计算 q 的最终查询区域 FS_q;

11. 在最终查询区域 FS_q 找出查询 q 的 k 近邻.

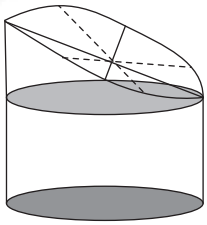


图 3 近似求积分示意图

在 IS-CKNN 算法中, 首先基于所有移动对象建立混合高斯模型, 然后使用 Get-Integration 算法计算以查询点为中心, 给定半径 r 范围内的积分也就是移动对象落入该区域的概率, 以此概率乘以移动对象总数近似获得该范围内的移动对象数量, 若移动对象数量满足一定阈值, 则该区域作为查询的最终查询区域, 查找 k 近邻, 否则, 修改半径 r, 使用 Get-Integration 算法直到获得合适的查询区域.

在积分算法中, 将区域划分为 n 个小区域, 分别对 n 个区域进行近似体积计算并求和, 获得区域内落入移动对象的概率 p. n 的取值会影响积分算法的性能, 我们在实验部分将对 n 的取值对算法性能的影响进行测试.

具体而言, 首先选取范围 RR_q 边上的 n 个点, 分别将查询点与其中两个相邻点组成一个小区域, 将 RR_q 分成 n 个区域, 在每一个区域内选择查询点与两个顶点, 顶点为区域切分点, 利用混合高斯模型计算两个切分点与查询点的函数值并选出最大值与最小值, 在计算几何体体积时, 分为上下两个部分, 下部为圆柱体体积, 选择区域所有顶点中最小的函数值计算圆柱体的体积. 对于上半部分体积, 因为划分为 n 个小块区域, 分别计算各个区域的体积并求和得到上半部分体积. 在 IS-CKNN 算法中, λ 决定了 while 循环次数和 Get-Integration 算法的积分区间, 从而影响查询时间和查询准确率, Δr 决定积分渐变幅度, 也将影响积分的计算结果. 因此, λ 和Δr 是 IS-CKNN 算法中与查询结果相关的重要参数, 本文将在实验部分对其进行充分测试.

当 q 的最终查询区域 FS_q 得到后, 在网格索引上 IS-CKNN 算法将计算并比较 FS_q 区域内移动对象与 q 的距离, 从而得到 q 的 k 近邻.

定理 1. IS-CKNN 算法初次查询的时间代价为 $u\times T_g+v\times T_c+v\times\log k\times T_f$.

证明: 假设初始查询区域在迭代 u 次后, 区域内的移动数量到达 v 个时满足阈值要求, 令一次迭代求积分的时间代价为 T_g, 则该步骤的时间代价为 $u\times T_g$. 下一步, 需从查询区域内的 v 个移动对象找到距离查询点最近的 k 个移动对象. 首先, 令计算某个移动对象到查询点距离的时间代价为 T_c, 则计算 v 个移动对象找到查询点的距离所需时间代价为 $v\times T_c$. 然后, 采用堆排序从 v 个移动对象中找出 k 个最近的移动对象, 令堆的大小为 k, 单个移动对象在堆中调整一次的时间为 T_f, 则该步骤的时间代价为 $v\times\log k\times T_f$. 因此, 初次查询的时间代价为 $u\times T_g+v\times T_c+v\times\log k\times T_f$.

算法 2. Get-Integration 算法.

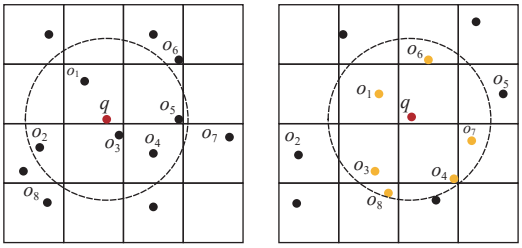
输入: q, RR_q 的半径为 r;
输出: p.

/*将区域 RR_q 等分为扇形小区域, 通过近似计算小区域的体积获得 RR_q 区域的体积*/
1. 选取区域 RR_q 边上 n 个点;
2. 将区域 RR_q 分为 n 个区域;
/*分别计算区域等分点的函数值*/
3. **FOR**($t=1$; $t\leqslant n$; t++)
4. $f_t=N(o_i;C_j)\times\varphi_j$;
5. **END FOR**
/*将 RR_q 区域的体积 (以该区域为底的几何体为曲顶圆柱体) 划分为上下两部分分别计算*/
/*计算底部圆柱体的体积 V_1*/
6. $V_1=\pi\times r^2\times\min(f_t)$;
/*近似计算上部曲顶几何体体积 V_2*/
7. **FOR** ($t=1$; $t\leqslant n$; t++)
/*上部曲顶几何体体积将被划分 n 个区域, 分别计算各区域体积, $f_{t_{RR_{qt}}}$ 为区域顶点函数值*/
8. $VS_t=\frac{1}{n}\times\frac{1}{3}\times\pi\times r^2\times\left[\max(f_{t_{RR_{qt}}})-\min(f_{t_{RR_{qt}}})\right]$;
9. $V_2=V_2+VS_t$;
10. **END FOR**
11. $p=V_1+V_2$.

3.3 IS-CKNN 算法增量查询

当查询对象移动后, 由于我们已经保存上一时刻初始查询的最终查询区域, 所以此次增量查询的候选查询区域 FS'就是初始查询的查询区域 FS, 因为在增量查询时, 虽然查询点和移动对象都发生了移动, 但它们的位置变化后, 仍然保持混合高斯模型的分布情况, 因此查询区域内的移动对象数量几乎是不变的, 所以可以将前一时刻初始查询的最终查询区域作为增量查询的候选查询区域. 以查询 q 移动后位置 q'为圆心, 在模型上计算 FS'区域内落入移动对象的概率, 乘以移动对象总量近似获得区域内移动对象数量, 若满足 $k+\lambda\leqslant p\times N_p\leqslant k+2\lambda$, 则在该区域内查询 k 近邻, 否则修改查询范围直到区域内移动对象数量满足要求, 然后查询 k 近邻.

以图 4 为例, 图 4(a) 为 T_1 时刻查询 q 查询范围内的移动对象位置分布情况, 在图 4 中查询范围内包含 o_1, o_2, o_3, o_4, o_5 这 5 个移动对象, 这里 k=3, λ=2, 当到达 T_2 时刻时, 此时查询区域内的移动对象数量仍满足查询要求, 因此不需要修改查询范围. 图中所有移动对象包括查询点都发生移动, 原 T_1 时刻的 o_2, o_5 在 T_2 时刻已经离开查询范围, 因此它们被移除候选队列, 而 o_6, o_7, o_8 此时进入查询范围, 将它们加入候选队列, o_1 仍然位于查询范围内, 只需要更新它到查询点的距离即可. 然后在范围内的所有移动对象中找出查询 q 的 k 近邻.



(a) T_1 时刻位置 (b) T_2 时刻位置

图 4 增量查询示意图

定理 2. IS-CKNN 算法增量查询的时间代价为 $e\times T_g+v\times T_c+v\times\log k\times T_f$.

证明: 与初次查询的时间代价类似, 增量查询的时间代价可表示为计算候选查询区域的时间代价 $e\times T_g$, 以及从候选查询区域中计算 k 近邻的时间代价 $v\times T_c+v\times\log k\times T_f$. 由于增量查询算法利用初始查询的最终查询区域作为增量查询的候选查询区域, 因为移动对象在两次查询过程中仍然符合混合高斯模型的分布情况, 查询区域包含的移动对象数量几乎不变, 所以在增量查询中迭代次数 e 远小于初次查询中的迭代次数 u, 这使得增量查询的时间代价小于初始查询的时间代价.

4 实 验

本文实验采用配置为 Inter(R) Core(TM) i5-4210U 1.7 GHz CPU, 8 GB 内存和 512 GB 硬盘的 Dell 笔记本一台. 实验数据集采用以下 3 个数据集.

● 成都某日滴滴出行订单数据集 (Didi dataset), 该数据集由订单中汽车行驶路线的经纬度数据组成, 数据集中包含 20000 个移动对象.

● 基于高斯分布生成的模拟数据集 (Gauss dataset, GS), 数据集由 3 个不同单高斯模型加权组成. 3 个高斯模型中心点分别为: (4.77, 7.20), (5.31, 2.73), (1.81, 3.95), 方差 (4.951, 0.01; 0.01, 1.93), (5.277, −0.024; −0.024, 2.372), (0.782, 0.198; 0.198, 4.21).

● 基于泊松分布生成的数据集 (Poisson dataset, PS), 期望与方差均设置为 5.

我们将算法与 Yu 等人[15]提出的 YPK-CNN 算法和 Yi 等人[13]提出的 BP-CNN 算法比较, YPK-CNN 算法查询时, 首先确定一个矩形范围, 该范围包含 k 个移动对象, 然后以查询点到第 k 个近邻的距离作为半径画圆, 以该圆保证 k 近邻的真实性, 最后在区域内查询 k 近邻. BP-CNN 算法为每个网格保存网格中心点的 K ($K>k$) 个最近邻, 网格中心的 k 近邻作为候选移动对象, 然后在候选移动对象中查询 k 近邻, 位于边界附近的查询将比较相邻网格的候选移动对象来确定它的 k 近邻.

下文中, N_p 表示移动对象的数量, $|Q|$为查询个数, o_v 表示移动对象的移动速度, q_v 表示查询点的移动速度, r 表示初始查询后的候选半径, λ 为候选查询 k 值误差值, k 为查询近邻个数. 查询点和移动对象的移动方向都是随机的. 整个查询范围被表示为 10×10 的区域. 实验图中参数 N_p=20000 (Didi dataset), $|Q|$=10, q_v=0.0001, o_v=0.0001, $\Delta r=r\times 0.5$, k=10, λ=2.

本文实验中, 每组实验进行 10 次测试, 取 10 次测试的平均值作为最终实验结果.

4.1 IS-CKNN 算法时间性能测试

该部分对 IS-CKNN 算法的时间性能进行测试, 图 5 给出了算法在不同数据集构建高斯模型的时间. 在 3 种不同数据集中, 高斯模型构建时间均在 2 s 以内, 其中两种数据集构建时间在 1 s 以内, 因为在高斯模拟数据集和滴滴数据集中的数据是符合高斯分布, 因此在这两种数据集中构建高斯模型的时间都较少, 而在泊松分布的数据集中, 因分布情况略有不同导致构建模型较慢. 图 6 比较了 IS-CKNN 算法初次查询时间与增量查询时间, 由图中可以看出, 采用增量查询方式, 时间消耗远小于不使用增量查询的方式. 在实际应用中, 初次查询往往只执行一次, 增

量查询的使用频率更高, 因此下面实验均是采用增量查询方法. 图 7 展示了在 3 种数据集下, 当发生位置变化的移动对象数量由 10% 逐渐增至 100% 的过程中, 高斯模型维护时间的变化. 由图中可以看到在滴滴数据集中, 维护时间从 0.2 s 增长至 0.59 s, 维护时间最短. 其次为高斯模拟数据集, 最次为泊松分布数据集, 由 1.21 s 增至 1.25 s, 对于 3 种分布的数据集, 索引结构的维护代价均在可接受范围内. 图 8 是移动对象数量对查询时间的影响, 在泊松数据集下, 移动对象数量增多会导致查询时间明显上升, 而在其他数据集中变化不大. 图 9 比较了移动对象数量不断增多的情况下, 3 种方法的查询时间, IS-CKNN 算法的查询时间更短. 图 10 是在滴滴数据集下测试 k 对查询时间的影响. 图 11 展示了我们在模拟高斯数据集中比较了 IS-CKNN 算法, YPK-CNN 算法, BP-KNN 算法在 k 值个数逐渐增大的过程中查询时间的变化, 在 k 值变化过程中 IS-CKNN 算法较优且保持一个较低的查询时间, 因为在计算查询范围时我们采用增量计算方法, 能够较快获得一个较为准确的查询区域, 减少了每次查询都要迭代计算查询区域的次数. 并在计算查询区域内移动对象数量时, 我们通过积分计算移动对象落入查询区域的概率, 而不是遍历检查每个移动对象.

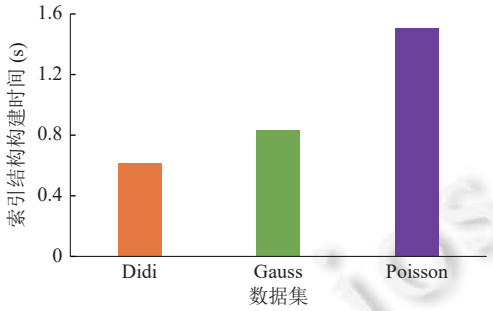


图 5 高斯模型构建时间

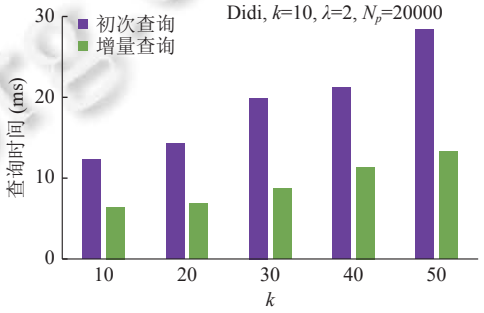


图 6 初次查询与增量查询时间对比

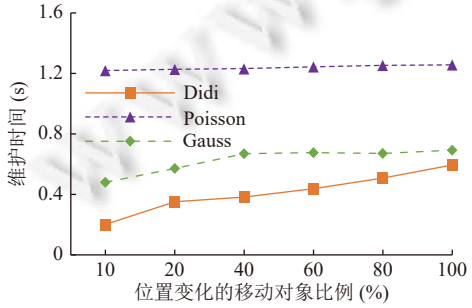


图 7 索引结构维护时间

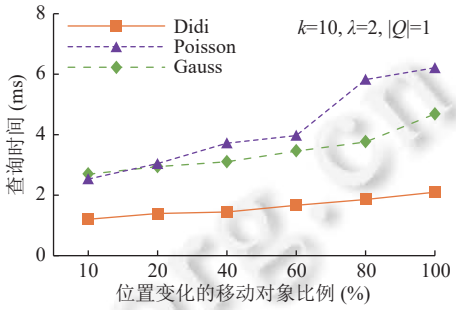


图 8 单个查询平均响应时间

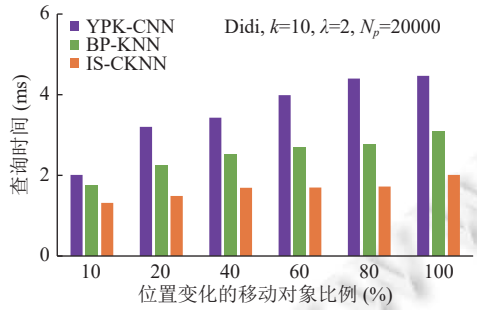


图 9 不同算法查询时间比较

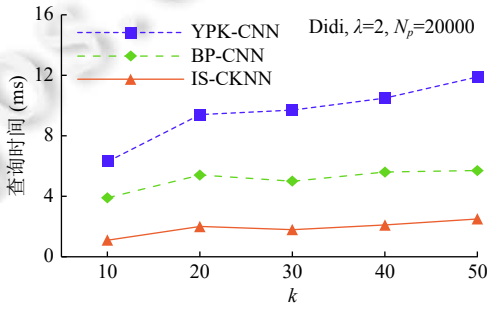


图 10 Didi 上 k 对查询时间的影响

图 12 展示泊松分布数据集下 k 值对查询时间的影响. 图 12 中, IS-CKNN 算法查询时间略微高于其他数据集, 原因是在泊松分布数据集中移动对象数量较少, 需要多次迭代获得合适的查询区域. 图 13 展示了 λ 对 IS-CKNN

算法查询时间的影响, λ 增大查询区域也随之增大, 计算查询区域的迭代次数增加. 表 2 比较了 IS-CKNN 算法与 YPK-CNN 算法在滴滴数据集, 高斯分布模拟数据集与泊松分布数据集下计算查询区域内移动对象个数的时间, 其中 IS-CKNN 算法时间为使用积分计算移动对象落入查询区域的概率乘以移动对象总量,

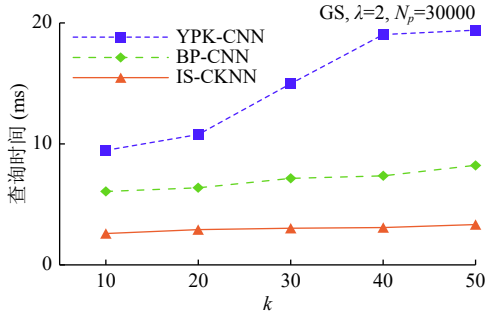


图 11 模拟高斯数据集上 k 值对查询时间的影响

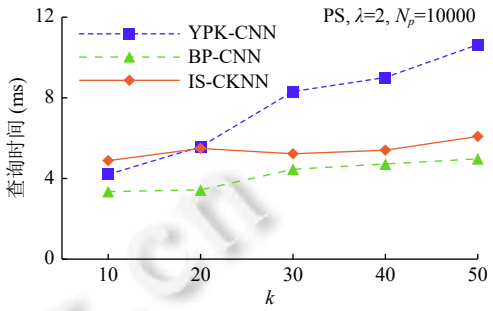


图 12 泊松分布数据集上 k 值对查询时间的影响

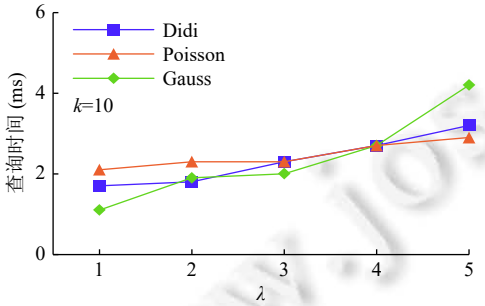


图 13 λ 值对查询时间的影响

表 2 计算区域内移动对象数量的时间 (ms)

算法	Didi	Gauss	Poisson
IS-CKNN	1.1	1.44	2.13
YPK-CNN	6.41	8.25	6

注: 数据空间范围为10×10范围, 查询范围为以查询点为中心, 0.025半径范围的圆形区域

进而获得查询区域内移动对象数量的时间, 数据空间范围为 10×10 范围, 查询范围是以查询点为中心, 0.025 半径范围内的圆形区域. 在 3 组数据集中, IS-CKNN 算法的计算时间明显优于 YPK-CNN 算法的计算时间. 在图 14 中展示了不同规模的高斯模拟数据集下, 算法的查询时间. 5 个数据集分别包含 10 万、20 万、30 万、40 万和 50 万个移动对象并且将查询请求设置为 100 个, 当移动对象数量增多时, 算法时间也在增多, 但是增长缓慢, 这是因为虽然移动对象数量增加对查询时间有一定影响, 但是本文所提出的方法能够快速确定候选查询区域, 受移动对象数量增长的影响较小. 图 15 展示了不同查询数量下查询时间的变化, 随着查询数量的增大, 查询时间也明显增大, 这是因为本文提出的方法为集中式算法, 对多个查询进行串行处理, 所以受查询数量的影响非常明显, 但是当查询数量大于 1 000 时, IS-CKNN 算法查询时间明显小于其他方法.

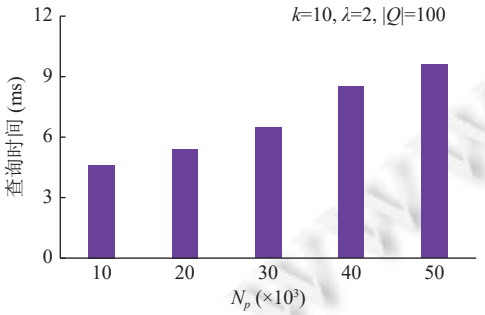


图 14 移动对象数量对查询时间的影响

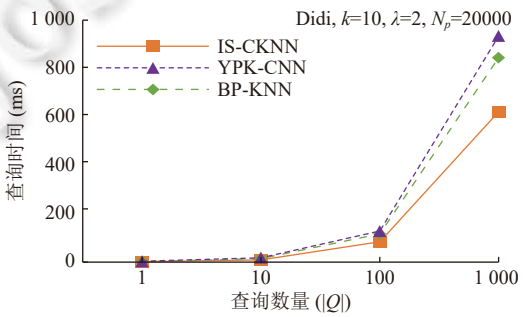


图 15 查询数量对查询时间的影响

4.2 IS-CKNN 算法计算指定查询区域内移动对象数量准确率的测试

该节我们对 IS-CKNN 算法的计算指定查询区域内移动对象数量的准确率 (A)(以下简称“准确率 (A)”) 进行测试, 其中公式 (8) 给出准确率 (A) 的定义.

$$A(v,w)=\begin{cases}1-\dfrac{|v-w|}{w}\times 100\%, & |v-w|\leqslant w\\ 1-\dfrac{w}{|v-w|}\times 100\%, & |v-w|>w\end{cases} \tag{8}$$

其中, v 为 IS-CKNN 算法计算所得查询范围内移动对象的数量, w 为查询范围内实际存在的移动对象数量.

由于准确率 (A) 与 Get-Integration 算法具有紧密联系, 而 Get-Integration 算法的参数 n 又对指定区域的积分计算时间和准确率 (A) 有重要影响, 因此, 我们对这一影响进行了充分测试, 实验结果如图 16 与图 17 所示. 图 16 中, 当 n 值不断增大时, Get-Integration 算法求积分的时间也不断增大, 当 n>256 时, 求积分时间显著上升; 图 17 中, 随着 n 值增大, 准确率 (A) 也逐渐上升. 当 n>256 时, 准确率 (A) 并未随 n 值增大而有明显变化. 如图 18 所示, 我们在滴滴数据集上测试了 k 值对准确率 (A) 的影响, 实验结果显示准确率 (A) 随 k 值变化发生无规律波动且集中在 86% 附近, 表明 k 值对准确率 (A) 的影响并不十分明显. 图 19 所示, 准确率 (A) 随着 λ 增大而下降, 这是因为在计算指定查询区域内移动对象数量时, 需满足 $k+\lambda\leqslant p\times N_p\leqslant k+2\lambda$, λ 增大导致误差增大, 使得准确率 (A) 下降. 同时, Get-Integration 算法的迭代次数也随着 λ 值得增大而逐渐减少, 这是因为 λ 增大导致误差增大, 计算结果落入误差范围内的可能性更大, 导致迭代次数减少. 图 20 展示了 k 值对积分迭代次数的影响, k 值增大会导致积分区域增大, 从而导致积分计算迭代次数增加. 图 21 中, 我们测试了查询数量准确率 (A) 的影响. 实验结果表明查询数量对准确率 (A) 的影响并不明显, 验证了该算法的稳定性. 图 22 中展示了Δr 对算法查询时间与准确率 (A) 的影响, 当Δr 增大时, 积分算法的迭代渐变范围也会增大, 减少迭代次数使得查询时间降低, 但同时Δr 增大会导致每次渐变幅度增大, 积分结果误差增大从而使准确率 (A) 出现下滑. 该参数在算法中影响极大, Δr 参数调整会影响每次积分的范围, 积分的准确性则会影响估算查询区域及区域内移动对象数量的准确性, 进而影响算法准确性.

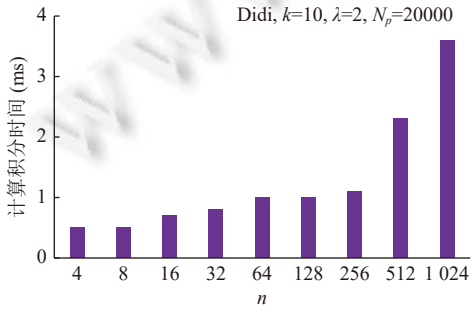


图 16 n 对积分计算时间的影响

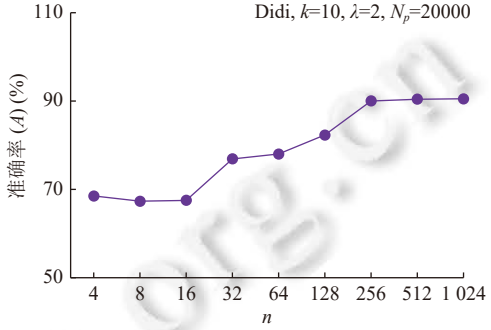


图 17 准确率 (A)

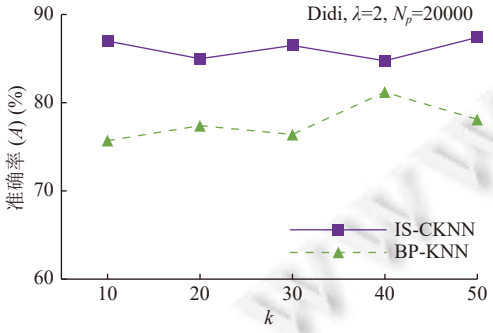


图 18 k 对准确率的影响

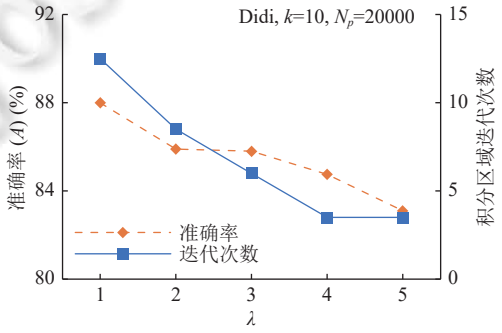


图 19 λ 对准确率 (A) 及迭代次数的影响

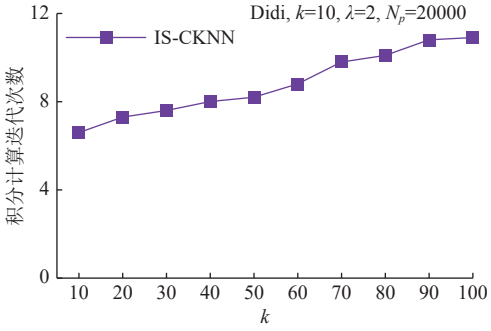


图 20 k 值对积分计算迭代次数的影响

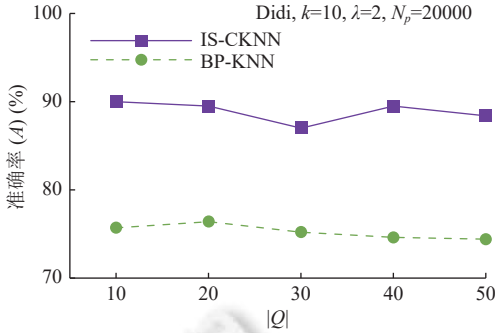


图 21 查询数量对准确率

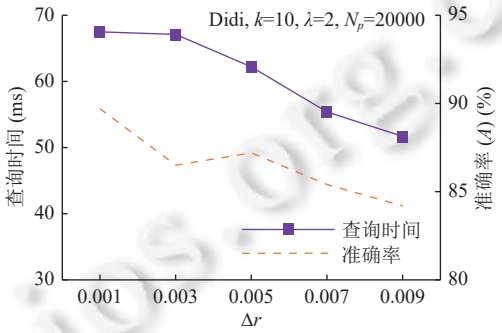


图 22 Δr 变化对查询时间与准确 (A) 的影响

4.3 实验结果分析

在实验中, IS-CKNN 算法查询时间要小于 YPK-CNN 算法, 因为我们在进行查询时, 通过前一次查询的最终查询范围作为下一次查询的积分范围, 并对积分范围增量变化, 避免了每次查询都需要重新计算查询范围的问题, 并且在计算查询范围内移动对象个数时, 通过积分进行计算, 避免了对部分覆盖网格中每个移动对象的检查. 在计算准确率 (A) 上本文提出的方法要优于 BP-KNN 算法, 原因是 BP-KNN 算法是以网格中心点的 k 近邻作为候选 k 近邻对象, 查询时, 从候选对象中查询 k 近邻对象. 当查询目标靠近网格中心时, 候选 k 近邻对象与准确 k 近邻对象重合率较高, 查询准确率较高, 相反越远离网格中心, 查询准确率则会下降, 当移动对象位于网格边缘时, 通常会将相邻网格的移动对象也加入候选近邻中, 准确率又会提升. 此外网格划分越小, 算法查询越准确. 我们的算法通过积分来获得区域内移动对象数量, 因此积分是否准确就决定了准确率 (A) 的高低, 算法中Δr 和 λ 参数会影响到求积区域的大小进而影响积分的准确性. 积分时划分区域的个数也是影响积分准确性的重要因素.

5 总 结

多年来, 移动对象 k 近邻查询问题作为许多基于位置服务的一个基本问题被国内外学者广泛研究. 本文研究了面向大规模移动对象连续 k 近邻查询问题, 提出了基于混合高斯模型的移动对象 k 近邻增量查询算法 (IS-CKNN). 该算法首先提出了基于网格索引和混合高斯模型的双层索引结构, 通过混合高斯模型模拟全局区域的移动对象分布的概率密度函数, 基于该函数对指定查询区域求积分就可以快速得到该区域内的移动对象数量, 能够有效减少确定候选查询区域的迭代扩展次数, 从而降低了查询的响应时间. 最后通过大量实验, 对本文中算法的性能进行了验证.

后续研究中, 将继续探讨如何提高查询区域内移动对象数目估计算法的准确率, 并进一步提高查询速度. 探讨在其他数据分布情况下的相关方法或其他方法的使用, 并考虑将该类方法应用于基于路网的移动对象连续 k 近邻查询问题.

References:

[1] Zheng YL. A fast index method for moving objects on full temporal query. In: Proc. of the 3rd Int'l Conf. on Computer Research and Development. Shanghai: IEEE, 2011. 205–208. [doi: 10.1109/ICCRD.2011.5764281]

[2] Zhong RC, Li GL, Tan KL, Zhou LZ, Gong ZG. G-Tree: An efficient and scalable index for spatial search on road networks. IEEE Trans. on Knowledge and Data Engineering, 2015, 27(8): 2175–2189. [doi: 10.1109/TKDE.2015.2399306]

[3] Shen BL, Zhao Y, Li GL, Zheng WM, Qin Y, Yuan B, Rao YM. V-Tree: Efficient kNN search on moving objects with road-network constraints. In: Proc. of the 33rd Int'l Conf. on Data Engineering. San Diego: IEEE, 2017. 609–620. [doi: 10.1109/ICDE.2017.115]

[4] Zheng BL, Zhao X, Weng LG, Hung NQV, Liu H, Jensen CS. PM-LSH: A fast and accurate LSH framework for high-dimensional approximate NN search. Proc. of the VLDB Endowment, 2020, 13(5): 643–655. [doi: 10.14778/3377369.3377374]

[5] Nutanong S, Zhang R, Tanin E, Kulik L. The V*-diagram: A query-dependent approach to moving kNN queries. Proc. of the VLDB Endowment, 2008, 1(1): 1095–1106. [doi: 10.14778/1453856.1453973]

[6] Hu L, Ku WS, Bakiras S, Shahabi C. Spatial query integrity with voronoi neighbors. IEEE Trans. on Knowledge and Data Engineering, 2013, 25(4): 863–876. [doi: 10.1109/TKDE.2011.267]

[7] Zhu HJ, Yang XC, Wang B, Lee WC, Yin J, Xu JL. Processing continuous *k* nearest neighbor queries in obstructed space with voronoi diagrams. ACM Trans. on Spatial Algorithms and Systems, 2021, 7(2): 8. [doi: 10.1145/3425955]

[8] Ni WW, Li LQ, Liu JQ. Voronoi-R*-based privacy-preserving *k* nearest neighbor query over road networks. Ruan Jian Xue Bao/Journal of Software, 2019, 30(12): 3782–3797 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5583.htm [doi: 10.13328/j.cnki.jos.005583]

[9] Šidlauskas D, Šaltenis S, Jensen CS. Parallel main-memory indexing for moving-object query and update workloads. In: Proc. of the 2012 ACM SIGMOD Int'l Conf. on Management of Data. Arizona: ACM, 2012. 37–48. [doi: 10.1145/2213836.2213842]

[10] Kumar S, Madria S, Linderman M. M-Grid: A distributed framework for multidimensional indexing and querying of location based data. Distributed and Parallel Databases, 2017, 35(1): 55–81. [doi: 10.1007/s10619-017-7194-0]

[11] Xu XF, Xiong L, Sunderam V. D-Grid: An in-memory dual space grid index for moving object databases. In: Proc. of the 17th IEEE Int'l Conf. on Mobile Data Management. Porto: IEEE, 2016. 252–261. [doi: 10.1109/MDM.2016.46]

[12] Yang KT, Chiu G M. Monitoring continuous all *k*-nearest neighbor query in mobile network environments. Pervasive and Mobile Computing, 2017, 39: 231–248. [doi: 10.1016/j.pmcj.2016.07.002]

[13] Yi X, Paule R, Bertino E, Varadharajan V. Practical approximate *k* nearest neighbor queries with location and query privacy. IEEE Trans. on Knowledge and Data Engineering, 2016, 28(6): 1546–1559. [doi: 10.1109/TKDE.2016.2520473]

[14] Li K, Malik J. Fast *k*-nearest neighbour search via dynamic continuous indexing. In: Proc. of the 33rd Int'l Conf. on Machine Learning. New York: JMLR, 2016. 671–679.

[15] Yu XH, Pu KQ, Koudas N. Monitoring *k*-nearest neighbor queries over moving objects. In: Proc. of the 21st Int'l Conf. on Data Engineering. Tokyo: IEEE, 2005. 631–642. [doi: 10.1109/ICDE.2005.92]

[16] Mouratidis K, Papadias D, Hadjieleftheriou M. Conceptual Partitioning: An efficient method for continuous nearest neighbor monitoring. In: Proc. of the 2005 ACM SIGMOD Int'l Conf. on Management of Data. Maryland: ACM, 2005. 634–645. [doi: 10.1145/1066157.1066230]

[17] Hua H, Xie HR, Tanin E. Is euclidean distance really that bad with road networks? In: Proc. of the 11th ACM SIGSPATIAL Int'l Workshop on Computational Transportation Science. Seattle: ACM, 2018. 11–20. [doi: 10.1145/3283207.3283215]

[18] Nutanong S, Ali ME, Tanin E, Mouratidis K. Dynamic nearest neighbor queries in euclidean space. In: Shekhar S, Xiong H, Zhou X, eds. Encyclopedia of GIS. Cham: Springer, 2017. 496–501. [doi: 10.1007/978-3-319-17885-1_1558]

[19] Miao X, Guo X, Wang H, Wang ZS, Ye XD. Continuous nearest neighbor query with the direction constraint. In: Proc. of the 17th Int'l Symp. on Web and Wireless Geographical Information Systems. Kyoto: Springer, 2019. 85–101. [doi: 10.1007/978-3-030-17246-6_8]

[20] Lee JM. Fast *k*-nearest neighbor searching in static objects. Wireless Personal Communications, 2017, 93(1): 147–160. [doi: 10.1007/s11277-016-3524-1]

[21] Liu J. Research on multi-source nearest neighbor query method in road network environment [Ph.D. Thesis]. Qinhuangdao: Yanshan University, 2019. 3 (in Chinese with English abstract). [doi: 10.27440/d.cnki.gysdu.2019.000013]

[22] Cheema MA, Zhang WJ, Lin XM, Zhang Y, Li XF. Continuous reverse *k* nearest neighbors queries in euclidean space and in spatial networks. The VLDB Journal, 2012, 21(1): 69–95. [doi: 10.1007/s00778-011-0235-9]

[23] Li CW, Gu Y, Qi JZ, Yu G, Zhang R, Yi W. Processing moving *k*NN queries using influential neighbor sets. Proc. of the VLDB Endowment, 2014, 8(2): 113–124. [doi: 10.14778/2735471.2735473]

[24] Zheng BL, Zheng K, Jensen CS, Hung NQV, Su H, Li GH, Zhou XF. Answering why-not group spatial keyword queries. IEEE Trans. on Knowledge and Data Engineering, 2020, 32(1): 26–39. [doi: 10.1109/TKDE.2018.2879819]

[25] Wang WC, Wong RCW, Xie M. Interactive search for one of the top-*k*. In: Proc. of the 2021 Int'l Conf. on Management of Data. Virtual Event: ACM, 2021. 1920–1932. [doi: 10.1145/3448016.3457322]

[26] Li MQ, He D, Zhou XF. Efficient *k*NN search with occupation in large-scale on-demand ride-hailing. In: Proc. of the 31st Australasian Database Conf. Melbourne: Springer, 2020. 29–41. [doi: 10.1007/978-3-030-39469-1_3]

[27] Lee K, Ganti RK, Srivatsa M, Liu L. Efficient spatial query processing for big data. In: Proc. of the 22nd ACM SIGSPATIAL Int'l Conf. on Advances in Geographic Information Systems. Texas: ACM, 2014. 469–472. [doi: 10.1145/2666310.2666481]

[28] Bao JL, Wang B, Yang XC, Zhu HJ. Nearest neighbor query in road networks. Ruan Jian Xue Bao/Journal of Software, 2018, 29(3): 642–662 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5442.htm [doi: 10.13328/j.cnki.jos.005442]

[29] Feng J, Zhang LX, Lu JM, Wang C. Review on moving objects query techniques in road network environment. Ruan Jian Xue Bao/Journal of Software, 2017, 28(6): 1606–1628 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5254.htm [doi: 10.13328/j.cnki.jos.005254]

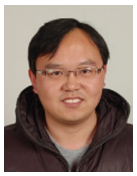
附中文参考文献:

[8] 倪巍伟, 李灵奇, 刘家强. 基于Voronoi-R*的隐私保护路网*k*近邻查询方法. 软件学报, 2019, 30(12): 3782–3797. http://www.jos.org.cn/1000-9825/5583.htm [doi: 10.13328/j.cnki.jos.005583]

[21] 刘佳. 路网环境下的多源最近邻查询方法研究 [博士学位论文]. 秦皇岛: 燕山大学, 2019. 3. [doi: 10.27440/d.cnki.gysdu.2019.000013]

[28] 鲍金玲, 王斌, 杨晓春, 朱怀杰. 路网环境下的最近邻查询技术. 软件学报, 2018, 29(3): 642–662. http://www.jos.org.cn/1000-9825/5442.htm [doi: 10.13328/j.cnki.jos.005442]

[29] 冯钧, 张立霞, 陆佳民, 王冲. 路网环境下的移动对象查询技术研究综述. 软件学报, 2017, 28(6): 1606–1628. http://www.jos.org.cn/1000-9825/5254.htm [doi: 10.13328/j.cnki.jos.005254]



韩士元(1985－), 男, 博士, 副教授, CCF 专业会员, 主要研究领域为智能交通系统复杂数据分析及应用.

童向荣(1975－), 男, 博士, 教授, CCF 高级会员, 主要研究领域为多 Agent 系统, 分布式人工智能, 数据挖掘技术.

何清(1996－), 男, 硕士, 主要研究领域为时空大数据.

郑渤龙(1989－), 男, 博士, 副教授, 博士生导师, CCF 专业会员, 主要研究领域为时空大数据管理与分析, 高维数据管理, 城市计算.

于自强(1984－), 男, 博士, 副教授, CCF 专业会员, 主要研究领域为海量数据管理与分析, 流数据分布式计算, 视频数据结构化查询.