计 算 机 研 究 与 发 展
Journal of Computer Research and Development
DOI: 10.7544/issn1000-1239.202330609
61(7): 1771−1790, 2024

基于空间位置关系的轨迹数据高效降维和查询算法

巢 成[1] 蒲非凡[1] 许建秋[1] 高云君[2]
[1](南京航空航天大学计算机科学与技术学院/人工智能学院/软件学院 南京 211100)
[2](浙江大学计算机科学与技术学院 杭州 310058)
(chaocheng@nuaa.edu.cn)

Efficient Dimensionality Reduction and Query Algorithm of Trajectory Data Based on Spatial Position Relation

Chao Cheng[1], Pu Feifan[1], Xu Jianqiu[1], and Gao Yunjun[2]
[1](*College of Computer Science and Technology/College of Artificial Intelligence/College of Software*, *Nanjing University of Aeronautics and Astronautics*, *Nanjing* 211100)
[2](*College of Computer Science and Technology*, *Zhejiang University*, *Hangzhou* 310058)

Abstract Due to the rapid development of information technology, society is in a critical period of digitalization and information transformation, and the demand for information systems based on database technology in various industries is becoming increasingly prominent. Location-based services rely on massive real-time generated trajectory data. In the processing of hundreds of millions of continuously changing trajectory data, dimensionality reduction algorithm and query technology have been the key to research. By reducing the scale of trajectory data and reducing the time of data processing during query operations, the performance of query can be effectively improved, and whether high-quality and efficient query can be achieved is very important for the database. In this paper, a UGC(uniform grid code) and a NGDR(non-uniform grid dimensionality reduction algorithm) for trajectory data are proposed, which convert the coordinates of trajectory data into one-dimensional string storage, merge the grids that do not meet the requirements, fully retain the complex interrelationship between trajectory data through spatial position mapping, and use range query and nearest neighbor query to test the performance of the reduced data. The real trajectory and virtual generated trajectory data in different cities are used as datasets, and the uniform grid code algorithm, non-uniform grid algorithm proposed in this paper are compared with three benchmark methods. Experiments show that the spatial position relationship similarity of the data after NGDR can be up to 82.5%. The range query time of NGDR is improved at least by 73.86% compared with the other queries, and the nearest neighbour query time is improved at least by 52.26%, which achieves better results than other benchmark methods.

Key words trajectory data; dimensionality reduction algorithm; non-uniform grid; spatial position relationship; query technology

摘 要 由于新型信息技术的快速发展,社会处于数字化、信息化转型的关键时期,各行业对于以数据库技术为基础的信息系统的需求也日益凸显.基于位置的服务依赖于海量实时生成的轨迹数据,在处理亿万级随时间连续变化的轨迹数据时,降维算法和查询技术一直是研究的关键,通过降低轨迹数据的规模,减少查询操作时处理数据的时间,能有效提升查询的性能,而能否实现高质量、高效率查询对于数据库而

收稿日期: 2023-07-26; **修回日期**: 2023-12-01
基金项目: 国家自然科学基金项目(U23A20296)
This work was supported by the National Natural Science Foundation of China (U23A20296).
通信作者: 许建秋(jianqiu@nuaa.edu.cn)

言至关重要.提出了面向轨迹数据的均匀网格编码,并在进一步优化后提出非均匀网格降维算法,将轨迹数据的坐标转化为1维字符串存储,对不符合要求的网格进行合并处理;通过空间位置映射充分保留轨迹数据间复杂的相互关系,并采用范围查询与最近邻查询对降维后的数据进行性能测试.实验使用不同城市真实轨迹数据与模拟生成轨迹数据作为数据集,将提出的均匀网格算法、非均匀网格算法与3种基准方法进行对比.实验证明,优化后的非均匀网格算法降维后数据的空间位置关系相似度可高达82.50%,范围查询时间较其他查询时间提升了至少73.86%,最近邻查询时间提升了至少52.26%,与其他基准方法相比取得了更好的效果.

关键词 轨迹数据;降维算法;非均匀网格;空间位置关系;查询技术

中图法分类号 TP391

随着互联网技术的不断发展,新一轮科技革命与产业革命强势来袭,数字化服务成为加速传统产业向智能化、绿色化、融合化方向转型升级的动力能源.大数据、人工智能、智慧城市与智慧交通等领域的研究促进了一系列新型信息技术的广泛应用,信息化技术也被赋予了新的内涵.而移动定位设备和无线技术的普及,产生了海量的轨迹数据,各行业也开始为用户提供基于位置的服务,例如智慧城市交通、驾驶路线导航、用户兴趣点推荐等[1].移动设备及传感器的采样频率不断提高,轨迹数据规模日益庞大,用户对于信息系统精准服务的需求也日趋多元,轨迹数据的价值逐渐凸显.为了更好地提供以数据为驱动的信息系统服务,研究者使用移动对象数据库[2]存储带有时间戳的空间数据序列,将轨迹数据的经纬度位置信息按照时间的先后顺序进行排列.移动对象库中亿万级的轨迹数据往往蕴含着丰富的信息与规律,因此对于轨迹数据的分析、存储和查询工作一直以来都是数据库领域研究的热点.

轨迹数据是在物体运动过程中每隔一段时间记录的物体位置信息,数据库往往需要保存一个对象在多个时刻的位置信息,形成一组连续的随时间变化的数据,包含对象标识、时间、经纬度、高度等多个属性.轨迹数据来源于空间传感器、卫星遥感影像、移动终端设备等.由于物体的位置信息更新频繁,数据生成频率极高,轨迹数据呈现出爆发式的增长态势.同时,对于同一空间对象,在不同研究条件下,数据会呈现出不同的几何表达,例如坐标点、等高线、不规则格网等.轨迹数据的多样表示为数据查询检索工作增加了极大的难度,在特定的数据库应用中,通常需要将不同形式的空间数据转化为一种统一的存储方式.因此,轨迹数据具有多维、海量、多样的特点.

在对轨迹数据进行数据分布的分析时,易知其还具有分布不均匀的特点,图1为50个移动对象的时空分布示意图,其中移动对象区域的经度范围是[116°34′31″, 116°69′19″],纬度范围是[39°85′07″, 40°04′08″],时间范围是某日0时至16时,因为需要表示时间、经度和纬度3个属性,所以将空间均匀地划分为3×3的网格,并对网格依次进行标识,使用网格*ID*表示该对象在固定时间区间中所处的位置,网格中数量越多,表示移动对象在该网格中聚集性越高.从图1中易知,在时间*ID*=1时,所有对象均在同一网格中,随着时间变化,移动对象向不同的网格区域移动,但始终呈现出明显的聚集特点,轨迹覆盖区域较大,但是极不均匀,大部分网格为空即该时段没有对象经过该区域.可见,轨迹数据还具有分布广、聚集多、非均匀的特点.

基于上述特点,传统的数据处理技术不适用于海量规模的轨迹数据,而且索引、日志等文件需要占用更多的存储资源.因此,有效提取轨迹数据间的位置关系、降低存储数据[3]及索引所需占用的空间,对于提高轨迹数据的时空操作效率至关重要,已经成为研究轨迹数据的核心问题.

为了解决针对多维数据在相关性计算方面难以保证数据的相关性和存储空间的问题,研究者们提出了数据降维技术,其旨在通过相关映射机制,将多维数据对应至低维数据空间中,并在低维表示中尽可能地保留多维数据的原始结构.

目前,已有学者针对轨迹数据降维开展研究工作并取得相关成果[4].经典的数据降维技术主要分为2类:线性降维和非线性降维.前者相对来说易于实现且快速简单,它假定数据在多维的局部线性子空间附近,通过线性映射将多维数据转换为低维数据.虽然线性降维技术方便易用,但是它并不能有效处理多维复杂的非线性数据[5],会在降维过程中损失非线性数据的重要特征.非线性降维技术可以进一步

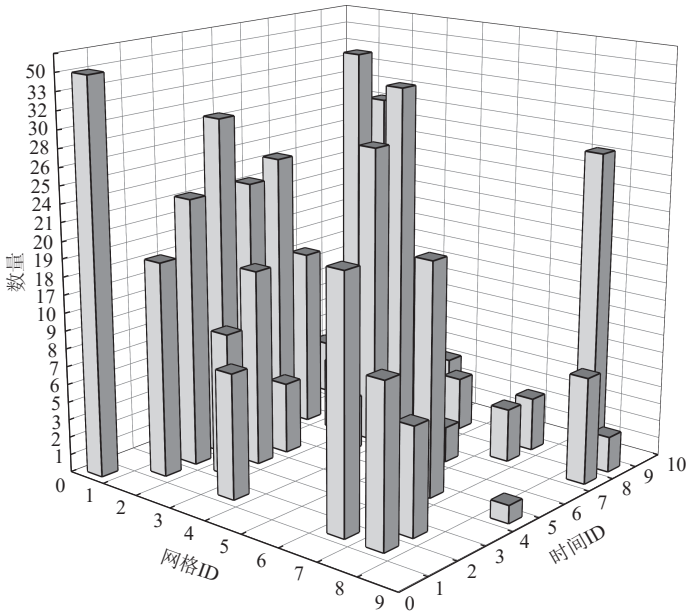


Fig. 1 Schematic diagram of spatial-temporal distribution for 50 objects

图 1 50 个对象时空分布图

被细分为全局降维和局部降维，经典的非线性降维算法都是通过找出每一个数据点周围的局部信息，然后通过非线性的映射将多维数据映射到一个低维的特征空间中，从而保留原始数据集的局部结构特征[6]，但是针对轨迹数据空间位置特征的保留仍是研究的难点.

同时，根据轨迹数据查询的不同需求，学术界开展了关于索引和查询算法的研究，如点范围查询、时空范围查询、最近邻查询，还有针对历史的轨迹数据的查询、针对实时动态轨迹数据的查询等. 基于 B+树、R 树的不同索引技术则是为不同情形下的查询处理要求提供了有效的支持. 面向空间的索引通常是预先对空间进行划分，基于索引空间的划分块得到轨迹数据的索引值，如 GeoMesa, Quadtree, TrajMesa 等技术被相继提出，但传统的查询处理在处理移动对象轨迹数据的查询问题时，在轨迹数据处理上仍有受限空间，缺乏复杂应用场景下移动对象点的查询研究.

因此，本文提出了一种针对轨迹数据特点与移动对象数据库查询需求的均匀网格降维算法，通过将轨迹数据坐标转化为 1 维网格编码，压缩轨迹数据在移动对象数据库中的存储规模. 根据轨迹数据的非均匀分布特点，提出了网格密度概念，因为均匀网格划分将造成空间的浪费，较大地影响数据处理效率，因此本文在均匀网格划分的基础上，进一步优化设计了基于网格密度的非均匀网格降维算法，使得多维数据存储有效减少空间占用的同时，充分保留其空间位置关系的特点，并与其他经典降维算法进行对比分析. 此外，本文所研究的降维算法，在将空间数据转化为 1 维值后，会根据网格编码包含其所在空间区域范围信息，考虑了轨迹数据的分布特点，并非只对数值进行降维，是针对未来查询研究、构造查询索引所作的准备工作. 为了使得不同轨迹数据在经过不同降维算法转化为 1 维值后，仍然能在同一标准下进行性能评估，本文还提出了一种基于相对位置关系的评价指标，即空间位置相似度，用于考察降维算法对于原始数据集中空间相似性特点的保留程度. 网格降维算法对查询操作起到过滤作用，本文设计引入了 2 种在轨迹数据查询领域应用最为广泛的查询算法：范围查询和最近邻查询. 范围查询在实际场景中可以用于搜索某轨迹所在地理位置的一定范围内所含其他轨迹的数量、某指定区域的轨迹数量，也可用于智慧交通区域流量分析等. 最近邻查询在实际应用中查询与指定轨迹最近的轨迹点，可用于地理信息系统中空间对象的查询和分析、路网环境下距离最近场所设施的推荐等. 查询过程先通过降维的网格编码进行过滤，获得可能符合查询条件的网格，然后对所选网格内的轨迹数据进行遍历，以确保查询结果正确. 实验通过比较空间位置关系、查询性能等，从不同角度对本文提出的均匀网格降维算法与非均匀网格降维算法的性能进行评价分析.

综上所述,本文的主要贡献有4个方面:

1)提出了一种基于网格划分的均匀网格编码(uniform grid code, UGC),对GeoHash的编码方式进行变形,将移动对象轨迹数据的空间坐标转化为1维字符串.

2)提出了非均匀网格降维算法(non-uniform grid dimensionality reduction algorithm, NGDR),定义网格密度概念,对密度低于阈值的网格进行合并,并提出了一种降维评价指标——空间位置相似度.

3)基于降维后得到的数据,采取先过滤筛选后遍历查询的方法,设计实现了实际应用场景中常用的范围查询与最近邻查询算法.

4)在移动对象数据库SECONDO中使用4种包含真实数据与模拟生成数据的数据集,将本文提出的均匀网格算法、非均匀网格算法与其他3种基准方法进行对比.最终通过非均匀网格算法降维后数据的空间位置关系相似度可高达82.5%,优化后的非均匀网格降维算法的范围查询时间较其他查询减少了至少73.86%,最近邻查询时间减少了至少52.26%.

1 相关工作

目前大量的学者针对轨迹数据开展了广泛的研究,例如数据建模、索引技术、查询操作等方向.随着维度的增加,计算量呈现爆发式增长,进而演变为维度灾难[7].维度灾难是指因为数据维度的增大,导致目标函数优化、参数估计等变得越来越困难,严重影响到机器学习、模式识别、文本分析等领域.

1.1 数据降维技术

维度灾难所带来的影响,对多维数据处理中的数据分析带来了重大挑战,近年以来,越来越多的学者为解决此问题开展降维技术相关的研究工作,并取得了一系列的成果.目前学界通常会将降维技术分为线性降维与非线性降维.线性降维技术相对来说操作易于实现、过程快速简单.主成分分析[8]、线性判别分析[9]等是经典的线性降维技术,被广泛应用于线性可解的多维数据.轨迹数据因存在明显的时空特征,且属性存在非实数类型值,所以通常采用非线性映射的降维技术.

非线性降维技术的一个重要分支是流形学习,流形可以看作很多不同维数曲面的叠加,流形学习则是一种非线性的维数简化方法[10],是一种从流形到欧氏空间的映射.

1)局部降维算法

最经典的流形学习算法之一是局部线性嵌入[11](locally linear embedding, LLE),它是后续许多非线性降维算法的基础.局部线性嵌入的基本假设是,原本的数据在距离其最近或相似度最高的几个数据点形成的邻域上可以近似为欧氏空间,在此假设的基础上,每个数据点都可以用其邻近数据点的最优线性表示,保留了局部的线性特征,这就保证了数据局部几何性质的一致性.

与局部线性嵌入思想类似的还有拉普拉斯特征映射[12](Laplacian eigenmaps, LE),它也是基于局部结构在低维空间中重新构建数据点邻接关系的降维算法.拉普拉斯特征映射是一种基于图的降维算法,旨在使得原始图数据中相连接的点在降维后的空间中能够尽可能地靠近,从而达成在降维后仍然能够保持原有数据结构的目的.

基于局部的切空间排列[13](local tangent space alignment, LTSA)算法是用数据点邻域切坐标来近似表示流形的局部特征.数据点的局部几何特征是在样本点的切空间中体现,但是此算法在数据点非均匀分布或是样本邻域的平均值和自身偏差较大的情况下会造成误差加大,影响降维效果.

2)全局降维算法

基于全局的降维算法是指选择对于整个分类影响最大的若干个特征来构成新的特征空间,从而实现对于原特征空间的降维.

多维尺度[14](multi-dimensional scaling, MDS)分析是典型的非线性全局降维技术,主要的算法思想是保持欧氏距离,使得多维空间中的欧氏距离在低维空间中能够近似表示.此技术按照对于距离相似需求的不同,可以分为2种:一种确保降维后低维数据点之间的距离和原数据点之间的相对距离尽量保持一致,以尽可能保留数据对象间相对距离的相似性;另一种则是按照数据库中数据的排列顺序进行降维后的排序,与对象点的相对距离无关.

等距映射[15]与多维尺度分析的不同在于其使用了测地线距离来计算多维空间距离,因此它不适用于曲率较大的流形降维.同时,由于需要计算任意2点之间的距离,所以只能用于连通图,若图中存在不连通部分,则需要分别开展降维.

基于元学习的自适应线性数据降维算法[16]是针对数据的类型和特点而提出的降维方式,该算法引入元学习和选择统计的方法作为数据集的构造和分类规则,从而达到降维过程中的分类错误率.自适应的降维算法适用于多种数据类型,但是在处理含有

较多异常值的数据集时性能较差.

GeoHash 编码[17]是一种基于哈希映射的降维算法, 通过将地理空间不断交替进行 2 分, 形成一个具有唯一性的二进制字符串. 由于每一个下级单元网格均由上级单元网格递归划分而成, 所以 GeoHash 编码具有明显的递归性, 在同一区域中的不同网格具有相同的前缀编码, 且编码的长度越短, 表示的区域范围就会越大. 其编码递归性的特点, 使得在此编码基础上建立的索引具有较好的性能, 有效提升了查询速度.

金字塔技术[18]是将 d 维空间分成 $2d$ 个金字塔, 每个金字塔将数据空间的中心点作为它的顶点, 数据空间的 1 个 d–1 维平面表面积作为它的积; 将 $2d$ 个金字塔再分成若干个块, 每个块对应 B+树中的 1 个数据页. 金字塔技术中的降维方式是将 1 个多维数据转化为 1 个金字塔值, 其编码方式为: 首先确定 1 个给定点 v 的金字塔 p_t, 接着确定维度到中心点的最大偏移度, 然后确定在该金字塔上的高度值 $h_v = |0.5 - v_{i \bmod d}|$. 由此就得到了点 v 的唯一金字塔值 $pv_v = i + h_v$.

3)空间填充曲线

空间填充曲线[19]是一种被广泛用于轨迹数据降维的方法, 其主要思想是用一条曲线填充 2 维甚至更高维度空间中的所有点, 空间填充曲线随参数的增加连续映射到单位空间中. 在常见的 2 维空间中, 当空间填充曲线的阶数接近无限大时, 充满空间的曲线就成为一种分形. Z-order, Hilbert 等曲线是目前最为常见的空间填充曲线.

Z-order 曲线[20]是当前在降维领域应用较为广泛的空间填充曲线之一, 通常也被称为 Morton 曲线或 N 形曲线, 以 Z 形曲线为基本分形. 将数据空间划分为 4 个子空间, 这 4 个子空间分别由基本分形填充, 再将各个子空间按照顺序首尾连接后得到. 编码方式是将 int 型的 x, y 值分别转化为 m 位二进制, 然后依次将 x, y 按位交叉排列得到一个长度为 $2m$ 的 *value* 值. 其实现简单, 但是查询过程中, 会由于曲线分布而遍历大量的无效点, 影响效率.

Hilbert 空间填充曲线[21]是以 U 形曲线为基本分形, 将数据空间划分为 4 个子空间, 这 4 个子空间以基本分形顺序填充, 再将子空间按照顺序首尾连接后得到. Hilbert 曲线的编码方式是将空间坐标转换为二进制码; 2 个二进制码两两交叉, 形成 1 个新的二进制串; 将新得到的二进制串从左到右分为 2b 长的 n 个字符串; 规定每个 2b 长的串的十进制 d, 例如“00”等于 0, “01”等于 1, “10”等于 3, “11”等于 2; 对于数组中的每个数字 j, 如果 j=0, 则把后面数组中出现的所有 1 变成 3, 并把所有出现的 3 变成 1. 如果 j=3, 则把后面数组中出现的所有 0 变成 2, 并把所有出现的 2 变成 0, 最后自左至右连接所有的串, 并计算其十进制值. Hilbert 曲线具有更好的空间聚集特性, 能够很好地保留空间对象的邻近关系, Hilbert 值邻近的空间对象在空间位置上也能保持邻近的空间位置关系. Hilbert 曲线因其良好特性被广泛应用于空间指数和地理空间数据管理[22]中, 并基于多尺度地理空间数据[23]进行了多种研究改进.

空间填充曲线的映射机制影响着映射数据的空间聚集能力, 在将多维对象点映射为 1 维值的过程中, 需要较好地保留空间对象间的位置关系才能保证数据的查询性能.

此外, 为了解决原始轨迹数据在存储过程中消耗大量存储资源的问题, 学者们提出了轨迹压缩技术[24]. 在对轨迹进行压缩时, 由于无损压缩会导致较差的压缩比和较大的数据重建开销, 因此有损压缩技术逐渐成为轨迹压缩的主流[25], 例如分段线简化[26]、基于参考的时空轨迹压缩的 REST 压缩框架等.

1.2 轨迹数据查询

在轨迹数据研究领域, 查询算法一直是被广泛研究的主题, 对于含有时空多重属性的空间数据, 面向点和区域的查询算法均已被众多研究者提出. 现有相关工作中对于轨迹数据的常见查询主要分为以下 4 个类型.

1.2.1 范围查询

范围查询是面向移动对象轨迹数据最基本、应用最广泛的查询类型之一, 其步骤是由用户提交查询的给定区域范围, 数据库在指定的有限时间内查询所有满足范围条件的轨迹数据, 并按照要求输出所有移动对象轨迹数据的查询结果.

Kulkarni[27]提出本地化差分隐私机制, 采取多层级方法在保护隐私的前提下, 查询某一范围内所有对象的出现频次之和, 其优势在于把区域进行分层处理, 进一步减少了误差累积. Ghane 等人[28]提出的轨迹数据范围查询算法是根据轨迹利用差分隐私机制统计出空间直方图, 进而快速查询某一空间内的查询对象数量. 范围查询可以通过建立索引来提高查询效率, 学者最初用 R 树索引结构来对移动对象轨迹数据进行范围查询.

1.2.2 最近邻查询

最近邻查询, 也常被称为相似性查询, 是移动对

象轨迹数据查询领域的重要问题，随着机器学习的广泛应用，其也成为了数据挖掘、模式识别、图像处理等诸多领域的基础性问题. 最近邻查询的步骤是由用户提交给定的 1 个或多个需查询的轨迹数据，并指定某个时刻或时间段，数据库查询与该时刻或时间段的指定轨迹数据距离满足查询条件的所有轨迹数据. 最常见的最近邻查询是 k 近邻(k nearest neighbor, KNN)查询，也就是查询距离指定的轨迹数据最近的 k 个轨迹数据，其中 k 为随机指定的数字.

Li 等人[29] 使用不确定 Voronoi 图作为索引结构提出了一种基于概率的聚集型最近邻查询算法，该算法包括处理阶段、修剪阶段和细化阶段，将候选集中概率不小于用户指定阈值的 k 个数据点组成的集合存储到结果集中并返回给用户. 最终，通过实验验证了所提算法在概率阈值聚合最近邻查询中的有效性与优越性. Zhang 等人[30] 针对维数特征向量匹配精度低的问题提出了一种基于快速库的近似最近邻搜索的图像匹配算法，并引入随机抽样一致性算法点误匹配，通过实验仿真消除错误匹配点. 实验结果表明，在匹配算法提高匹配准确率的同时，算法的实用性也得到了提高.

1.2.3 相似轨迹查询

相似轨迹查询是通过比较 2 条轨迹数据在相同时间段内通过相邻节点的数目是否达到阈值要求，通常研究认为通过相邻节点的数目越多，这 2 条轨迹的相似性就越高. 其步骤是根据用户所提交的查询轨迹的若干条件，用基于特征提取的索引降维找到相似轨迹，查询满足相似度条件的轨迹数据. 相似轨迹查询可以分为基于阈值的查询和 Top-k 相似性查询，前者是当用户给定一个相似性阈值时，数据库通过查询返回轨迹数据集中所有与查询轨迹的相似值小于阈值要求的轨迹，而后者是返回与查询轨迹相似度最高的 k 条轨迹，实质上就是对基于阈值的相似性查询的返回结果进行了排序，并按照排序结果进行筛选，所以二者的核心算法并没有区别. 因为相似轨迹查询需要基于大规模的移动对象轨迹数据集，所以目前对于相似轨迹的查询算法大部分都是基于分布式集群进行研究与计算的.

Zhao 等人[31] 为了克服轨迹受空间相似度和时间相似度混合的影响造成的总体相似度接近但实际轨迹特征相差甚远的问题，提出了通过同步匹配空间距离和时间距离来测量相似性，同时通过轨迹数据库网格索引和查询分区 2 种技术来有效地解决 Top-k 个相似度检索问题. Zheng 等人[32] 提出了一个分布式内存管理框架，来处理 Top-k 轨迹相似查询问题，用参考点索引来组织轨迹数据以进行局部搜索，此外他们还设计了一种新的异构全局分区策略来消除分布式设置中的负载不平衡.

1.2.4 其他查询算法

针对移动对象轨迹数据的查询除了上述最常用的范围查询、最近邻查询和相似轨迹查询外，学者还对其他查询方式开展了大量的研究.

聚集查询是针对用户提交给定的区域范围，查询得到该范围内所有移动对象的轨迹数据，返回查询结果为非空间特征的显示属性的描述. 通常情况下，学者们会使用聚集函数对一个元组集进行处理，然后返回一个值来归纳包含在元组集中的信息特点. 连续查询则是需要用户提交时间范围区间、数据库在给定的时间段内进行连续查询[33]，因为移动对象是连续不断运动的，所以它的位置数据是不断更新改变的，对应的查询结果也需要不断地更新，以确保查询结果的准确性.

根据查询的要求不同，移动对象轨迹数据的查询处理可以从不同维度进行划分，除了上述提及的查询方法，还可以分为针对历史的移动对象轨迹数据、针对当前的数据以及未来的预测数据等的查询，而不同的索引技术也为不同情形下的查询处理要求提供了有效的支持.

2 问题定义

2.1 数据降维基础定义

定义 1. 轨迹数据点 P.

$$P=\left\{(t,x,y)|t\in T,(x,y)\in\mathbb{R}^2\right\},$$

其中 t 为时间，x 为经度，y 为纬度，每个轨迹数据点都由 (t,x,y) 三元组表示.

定义 2. 轨迹数据 MP. 同一对象在某时间段内的轨迹数据由多个轨迹数据点(t,x,y)组成，在移动对象数据库中用 MP 来表示这样的轨迹数据：

$$MP=\langle P_1,P_2,\cdots,P_n\rangle,$$

其中轨迹数据点按照时间顺序排列，不重复地连接成为一段完整轨迹.

定义 3. 轨迹数据降维函数 f.

$$f:P\rightarrow P',$$
$$P'=\{(t,c)|t\in T,c\in\mathbb{R}\},$$

将轨迹数据点 P 映射为以时间与 1 维编码构成的 2 维对象 P'，降维后的数据将减少空间占用率，且降维

后数据可通过运算还原至多维.

定义 4. 降维轨迹数据 *RMP*.经降维函数 f 降维后的轨迹数据为

$$RMP = \{P'_1, P'_2, \cdots, P'_n\}.$$

由降维后的数据点按照时间顺序排列而成，即对原始轨迹数据 *MP* 中的各元素进行降维后的结果表示. 本文降维算法针对轨迹数据，将 2 维坐标降至 1 维.

2.2 数据查询相关定义

本文为了检验使用非均匀网格算法进行降维后移动对象轨迹数据的查询性能，引入了 2 种查询算法：范围查询和最近邻查询，并将查询效果与 4 种基准降维算法所得数据后的查询结果进行对比，来检验降维对查询效率的影响.

定义 5. 降维轨迹数据-范围查询 *RMP-RQ*(dimensionally reduced MP data-range query). 时空范围查询是经典移动对象数据库中最常用的查询方式，查询在特定时间内出现在某区域范围内的移动对象的轨迹数据. 时空范围查询可以分为时间点查询、时间段查询、空间位置点查询和空间区域查询等若干种类，因本文仅对空间属性进行降维处理，故选择采用制定空间区域的范围查询，并将此查询操作定义为 *RMP-RQ*.

$$RMP\text{-}RQ(MP, f) = \{(P_1, P_2, \cdots, P_n) \mid (P \in MP), (f(P) \in f(Bbox))\}.$$

定义空间查询区域为 2 维数据空间中的矩形边界框 *Bbox*(bounding box)，则指定查询区域 $Bbox = (X_{\min}, Y_{\min}, X_{\max}, Y_{\max})$. f 为降维函数，将 2 维区域映射至 1 维区间 $f(Bbox) = [r_1, r_2]$，其中 $r_1 < r_2$，r_1 和 r_2 分别为矩形框左下角点和右上角点的坐标降维后的 1 维值.

定义 6. 降维轨迹数据-最近邻查询 *RMP-NQ*(dimensionally reduced MP data-nearest neighbor query). 最近邻查询是在数据库中查询最靠近查询点的轨迹数据，基于本文降维数据的最近邻查询操作定义为 *RMP-NQ*.

$$RMP\text{-}NQ(P) = \{P_j \mid (Pre(C_Q, C_j) \geqslant Pre(C_Q, C_i)), (dis(P_j, P_Q) \leqslant dis(P_i, P_Q))\}, P_i \in P.$$

其中 $C_Q = N(P_Q)$，$C_j = N(P_j)$，$C_i = N(P_i)$. N 为非均匀网格降维函数，C_i 为点 P_i 降维后所得 1 维编码值，P_Q 为指定查询点，*Pre* 为返回字符串公共前缀个数的函数，*dis* 为返回 2 个数据点之间欧氏距离的函数.

本文提出的非均匀网格降维后的数据点，只需访问查询点所在网格及其相邻网格中的数据点，若所在网格中已无其他数据点，则连续访问其周边网格中的点，从而找到所查询数据点的最近邻.

3 均匀网格编码算法

近年来，尽管有大量的学者在研究如何针对多维数据进行降维处理，但是适用于存储和查询移动对象轨迹数据的方案仍然较少. 本文提出了一种基于均匀网格划分的编码算法，对网格中的轨迹数据点赋网格编码值，从而使得空间坐标变为 1 维值.

3.1 均匀网格编码

在构建非均匀网格之前，需要先将轨迹数据区域划分为若干个均匀网格. 本文所采用的均匀网格编码，是基于 GeoHash 编码的变形，是一种将轨迹数据点的经度与纬度先按照高斯-克吕格投影[34]方式转换坐标，然后通过编码高斯-克吕格坐标生成 1 维二进制字符串，以实现空间数据降维目的的算法.

图 2 展示了均匀网格编码算法的基本思想，首先通过遍历轨迹数据点找出可以移动对象轨迹区域的最小外接矩形，此最小外接矩形需与坐标轴相平行，将最小外接矩形平均分成 2 等分.

假设此矩形的对角点坐标为 $(X_{\min}, Y_{\min}, X_{\max}, Y_{\max})$，则第 1 次划分得到的 2 个矩形的对角点坐标分别为

$$\left(X_{\min}, Y_{\min}, \frac{X_{\min}+X_{\max}}{2}, Y_{\max}\right),$$

$$\left(\frac{X_{\min}+X_{\max}}{2}, Y_{\min}, X_{\max}, Y_{\max}\right),$$

并对横坐标范围

$$\left[X_{\min}, \frac{X_{\min}+X_{\max}}{2}\right]$$

的矩形编码为二进制 0，对横坐标范围

$$\left(\frac{X_{\min}+X_{\max}}{2}, X_{\max}\right]$$

的矩形编码为二进制 1.

第 2 次划分得到的 4 个矩形的对角点坐标分别为

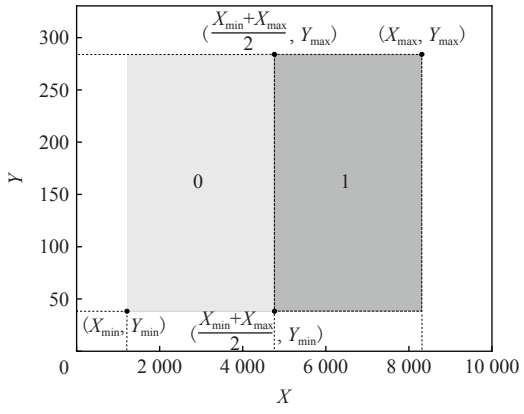
$$\left(X_{\min}, Y_{\min}, \frac{X_{\min}+X_{\max}}{2}, \frac{Y_{\min}+Y_{\max}}{2}\right),$$

$$\left(X_{\min}, \frac{Y_{\min}+Y_{\max}}{2}, \frac{X_{\min}+X_{\max}}{2}, Y_{\max}\right),$$

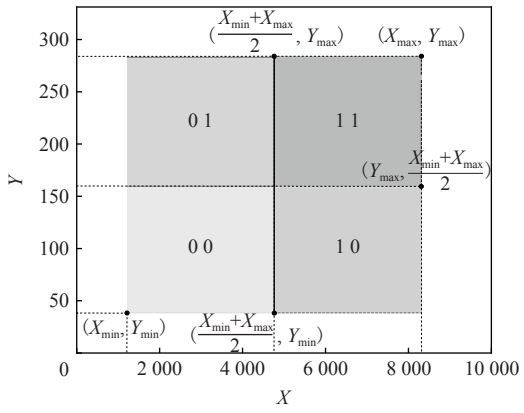
$$\left(\frac{X_{\min}+X_{\max}}{2}, Y_{\min}, X_{\max}, \frac{Y_{\min}+Y_{\max}}{2}\right),$$

$$\left(\frac{X_{\min}+X_{\max}}{2}, \frac{Y_{\min}+Y_{\max}}{2}, X_{\max}, Y_{\max}\right),$$

并对纵坐标范围



(a) 第1次划分网格



(b) 第2次划分网格

Fig. 2 Schematic diagram of uniform grid encoding

图 2 均匀网格编码示意图

$$\left[Y_{\min}, \frac{Y_{\min}+Y_{\max}}{2}\right]$$

的矩形编码为二进制 0, 对横坐标范围

$$\left(\frac{Y_{\min}+Y_{\max}}{2}, Y_{\max}\right]$$

的矩形编码为二进制 1, 至此在经历 2 轮网格划分后, 生成 2 位二进制编码, 作为 1 轮划分操作.

此后不断重复上述操作, 递归地对网格进行 2 等分得到更多的矩形区域, 生成更长的二进制编码, 区域精确度也在二进制编码不断增长的过程中得到了提高.

3.2 均匀网格编码算法

均匀网格在编码的过程是将 2 维坐标递归 2 分后交叉组成 1 维编码, 根据定理 1 易知, 前缀相同的网格必定被包含在同一个上级网格中, 例如图 2(b)中的 00 网格与 01 网格, 含有公共前缀“0”, 则这 2 个网格必定被包含在 0 网格中.

定理 1. 给定轨迹数据点 P_1 和 P_2, 根据均匀网格编码可得对应网格编码值 $Code(P_1)$ 和 $Code(P_2)$, 如果 $Prefix(Code(P_1)) = Prefix(Code(P_2)) = Pre$, $|Code(P_1)| \geqslant Pre$, $|Code(P_2)| \geqslant Pre$, 那么 P_1 和 P_2 一定都在编码为 Pre 的网格中.

证明. 对于具有相同编码前缀 Pre 的 2 个数据点 P_1 和 P_2, 假设它们不在同一个网格中. 因为 $Prefix(Code(P_1))=Pre$, 所以 P_1 在 2 分编码过程中, 被划分到编码为 Pre 的网格中, 又因为 $Prefix(Code(P_2))=Pre$, 所以 P_2 在 2 分编码过程中, 也被划分到编码为 Pre 的网格中, 所以 P_1 和 P_2 都在编码为 Pre 的网格中, 与它们不在同一网格的假设矛盾, 假设不成立. 综上, 如果 2 个数据点具有相同的前缀, 那么它们一定都在公共网格中. 证毕.

对于均匀网格编码算法 UGC 的具体说明, 描述见算法 1.

算法 1. 均匀网格编码算法 UGC.

输入: 轨迹数据点数组 MP, 轨迹数据点个数 M, 高斯-克吕格投影后的数据点 $GKpoints$, 最小点 P_{min}, 最大点 P_{max}, 1 维编码长度 $length$;

输出: 1 维编码 C.

① $MP \leftarrow$ 移动对象点的坐标;
② for $iter \leftarrow 1$ to $|MP|$
③ $GKpoints[i] \leftarrow GKprojection(MP[iter])$;
④ end for
⑤ P_{min}, $P_{max} \leftarrow GoThrough(Gkpoints)$;
⑥ for $iter \leftarrow 1$ to M
⑦ $X_{median} \leftarrow GetAverage(GetX(P_{min}), GetX(P_{max}))$;
⑧ $Y_{median} \leftarrow GetAverage(GetY(P_{min}), GetY(P_{max}))$;
⑨ for $iter \leftarrow 1$ to $length$
⑩ if $len(C)$ % 2 = 1 then
⑪ if $x\text{-}coordinate \leqslant X_{median}$ then
⑫ $C[i] <<1$, 更新 X_{median};
⑬ else $C[i] <<1$, $C[i] |= 1$, 更新 X_{median};
⑭ end if
⑮ else if $len(C)$ % 2 = 0 then
⑯ if $y\text{-}coordinate \leqslant Y_{median}$ then
⑰ $C[i] <<1$, 更新 Y_{median};
⑱ else $C[i] <<1$, $C[i] |= 1$, 更新 Y_{median};
⑲ end if
⑳ end if
㉑ end for
㉒ end for
㉓ return C.

在算法 1 中, 第 1 步将移动对象轨迹数据点的坐标放入数组 MP 中; 第 2 步通过 1 次循环将数组 MP

中的所有数据点进行高斯-克吕格投影，找出投影区域的对角点，即最小点和最大点；第 3 步是通过一个 2 重循环对每个数据点进行编码，奇数位为经度编码，若当前数据点经度小于经度平均值，则编码左移 1 位，末位赋 0，否则赋 1，然后将经度范围更新为当前数据点所在网格的经度范围. 偶数位为纬度编码，步骤同上，依次类推至编码长度达到给定精度要求. 最后退出循环，得到 1 维编码数组.

基于算法 1 描述与定理 1 的编码特点，易知 2 个轨迹数据点在空间中越临近，通过网格降维后的公共前缀就会越长. 同时可知，如果编码 C_1 以编码 C_2 为前缀，那么 C_1 对应的轨迹数据点一定落在前缀编码 C_2 所对应的区域中. 均匀网格降维需对网格进行编码，进而将轨迹点的坐标值转化为网格编码进行存储，以达到降维的目的. 为进一步压缩降维后所需的存储空间，使用 *bitset* 容器存储 UGC 算法降维后的编码，有效节省了时间. 同时，此容器支持基本的位运算，n 位的 *bitset* 在执行 1 次位运算时的复杂度为 $n/32$，因为对于 1 个仅存 0 或 1 的 int 型变量而言，*bitset* 所需的空间与时间仅是它的 1/32.

4 非均匀网格降维算法

考虑到轨迹数据存在不均匀分布的特点，均匀网格划分将造成空间的浪费，较大影响数据处理效率. 本文在均匀网格划分的基础上，进一步提出了非均匀网格降维算法，在节省空间的同时充分保留轨迹数据点间的空间位置关系，并设计了一种新的评价指标.

4.1 网格密度

为便于分析轨迹数据分布，本文提出了网格密度值(grid density, Dens)的概念，引入符号如表 1 所示.

Table 1 Definition of Uniform Grid Symbols

表 1 均匀网格符号定义

符号	说明
M	轨迹数据总数
Ul	单位网格长度
Uw	单位网格宽度
Num	单位网格总数
PN_i	网格 i 中轨迹数据点的个数

给定移动对象的数据集中共有 M 个轨迹数据采样点，对移动对象的轨迹区域进行均匀划分后共产生 Num 个均匀网格，则第 i 个网格的密度值为

$$Dens_i = \frac{Num \times PN_i}{M}, \tag{1}$$

网格密度值用于衡量网格中轨迹数据点的密集程度. 如图 3 所示，网格的颜色越深表示该网格的密度值越大.

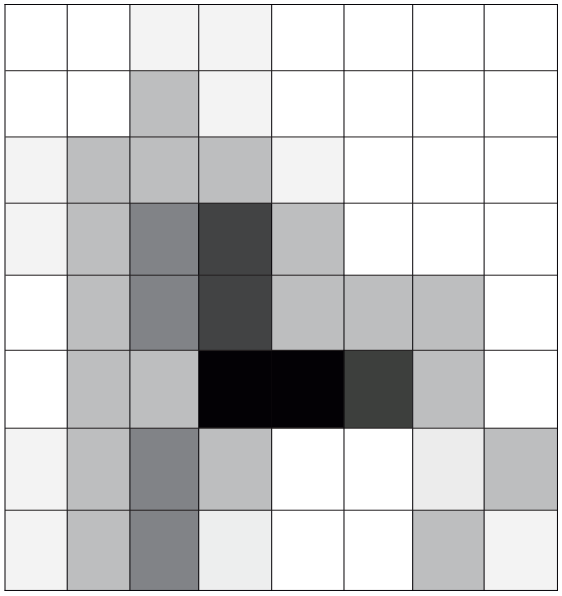


Fig. 3 Schematic diagram of the uniform grid distribution of trajectory data

图 3 轨迹数据均匀网格分布示意图

4.2 非均匀网格构建

通过观察轨迹数据在均匀网格中的分布，发现均匀网格划分会导致大量网格只有极少的轨迹数据点，甚至有部分网格不存在轨迹数据点. 易知如果轨迹数据在区域中完全均匀分布，则每个网格对应的 *Dens* 值都为 1，但是如图 3 所示真实数据集映射后不同网格的 *Dens* 值偏差较大.

由于网格的 *Dens* 值偏差较大会导致：1)大量空网格、低密度网格编码后需存储，浪费存储空间；2)查询阶段，无法将符合条件的空网格、低密度网格过滤，造成冗余的读取操作，耗费查询时间，影响查询性能，因此本文对网格密度进行平衡操作.

为了使得每个网格中包含的轨迹数据点的数据量趋于平衡，本文提出了非均匀网格降维算法 NGDR，见算法 2. 设定网格密度阈值，将密度未达到阈值的网格进行合并.

算法 2. 非均匀网格降维算法 (NGDR).

输入：网格密度集合数组 *Dens*，轨迹数据点个数 M，均匀网格总数 Num，1 维编码长度 *length*；

输出：1 维非均匀编码 C.

① $Dens \leftarrow \varnothing$;

② C ← 根据 UGC 算法得到编码;
③ *average*(*Dens*) ← *get*(*Num*, *M*) ;
④ for *iter* ← 1 to *Num*
⑤ C_i ← 遍历网格得到每个网格编码;
⑥ *Dens* ← 计算每个网格密度值;
⑦ while $Dens_i$ < *average*(*Dens*) do
⑧ C_i >> 1;
⑨ end while
⑩ end for
⑪ return C.

算法 2 中, 第 1 步是通过轨迹数据点总数 M 与均匀网格总数 *Num* 求出网格密度平均值, 第 2 步将网格密度平均值作为判定是否需要进行网格合并的阈值. 这样就能将空网格与所含数据点数极少的网格进行合并, 有效地节省存储空间. 同时, 由于采取了交叉编码的方式, 所以当网格满足合并要求时, 无需在合并后更改每个网格的标识或新建列表记录网格合并的操作, 仅需将二进制编码右移 1 位, 即可实现与其邻近网格的合并操作, 其原理基于定理 1, 因为该网格合并即表示其成为上一级公共网格, 仅需提取公共前缀作为新的编码. 最后遍历完成后返回得到 1 维非均匀编码数组. 执行完非均匀网格降维算法后的数据划分如图 4 所示.

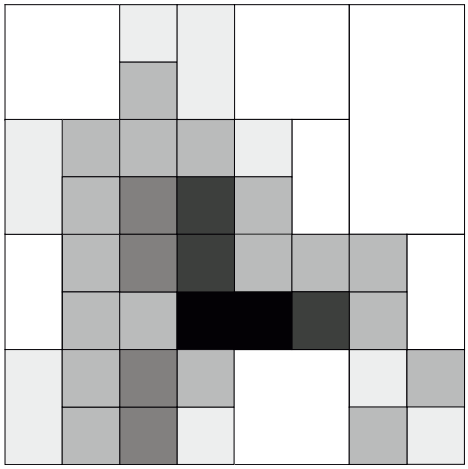


Fig. 4 Schematic diagram of the non-uniform grid distribution of trajectory data

图 4 轨迹数据非均匀网格分布示意图

因所有网格编码均为二进制字符, 编码唯一且前缀递归, 所以对降维后数据建立 B 树索引, 以左子树为 0, 右子树为 1, 每个叶节点对应唯一 1 个网格, 网格与数据点存在 1 对多映射关系, B 树索引叶节点的深度遍历顺序即为 1 维网格值排列顺序.

在轨迹数据动态更新时, 需要将新产生的轨迹数据点引入目前已构建的非均匀网格中, 实现降维处理. 更新场景下的降维过程分为 2 个步骤: 首先需要确定更新的轨迹数据所在的空间网格; 然后将其所在网格更新后的密度值与设置好的最高值进行比较, 若未超过最高值, 则无需其他操作, 以当前网格编码作为其降维后的编码值, 若超过最高值, 就对所在网格进行再次划分, 通过 1 个 8b 长的字段记录 2 次划分的局部编码, 但是当划分次数超过 8, 局部编码溢出时, 需要全局更新, 重新构建非均匀网格. 因此, 非均匀网格算法能够支持历史轨迹数据的更新操作, 适用于更新频率不高、密度分布与历史轨迹数据相似的场景.

4.3 非均匀网格算法优化

4.3.1 前缀分组

非均匀网格算法中需要对每个网格计算密度值, 但是在使用出租车数据集进行实验的过程中, 发现移动对象轨迹数据通常具有集中性, 即某部分网格密度较高, 但轨迹数据覆盖的矩形区域中绝大多数网格中并未含有轨迹数据点, 经过实验验证, 空网格比例达 92% 以上. 因此, 为节省遍历空白网格的时间与构建空白网格的空间, 本文对非均匀网格算法进行优化设计.

从轨迹数据点的唯一网格编码角度出发, 对轨迹数据点的网格编码进行遍历和比较, 若相同编码的轨迹数据个数已经超过设定阈值, 则当前网格密度值已超过阈值, 不再选择合并, 否则对网格进行合并处理. 但是, 在对轨迹数据点进行遍历的过程中, 若对完整的编码串进行比较则会浪费不必要的时间, 因此本文在比较过程中引入最长公共前缀比较法, 并对轨迹数据的非均匀网格编码打标签, 以降低比较所需循环数, 进一步减少降维算法的时间.

4.3.2 子轨迹 *Bbox* 的提取

对于轨迹数据而言, 因位置更新较为频繁, 产生大量数据点, 但若按照轨迹数据 *ID* 或时间段划分, 则轨迹段数量较少, 且数据点之间的间隔往往极其接近, 降维结果也极其相似, 对每个数据点进行降维编码会耗费大量时间. 因此, 为进一步优化降维时间, 本文采用粗粒度的非均匀网格降维算法, 按照指定时间段对移动对象轨迹数据进行划分, 并对每一个子轨迹构造最小矩形框 *Bbox*, 然后取该 *Bbox* 的中心点作为该段子轨迹中所有包含点的代表点, 仅对代表点进行降维处理, 取代表点的网格编码值作为该段子轨迹中所有包含点的网格编码值.

通过移动对象数据库实现 *MPDT*(mpoint divided

by time) 操作：*mpoint*→*stream*(*mpoint*)，*MPDT*(*mpoint*，*time*)．根据用户输出 *dt* 的时间要求，将时间区域间隔为 *dt* 的点划分为同一个 *mpoint*. 进而将划分后的 *mpoint* 作为降维单元，有效节省了降维过程中访问数据的时间，降低了需降维的数据规模，进一步提高降维效率.

4.4 高维数据非均匀网格降维

4.3 节所述的非均匀网格算法在高维数据处理时同样适用，其主要思想仍然为对每个维度所在区域递归地进行 2 分，在指定精度要求下，确定当前数据所在区域并编码. 针对维度为 *n* 的数据，首先计算得到 *n* 个维度的区域值

$$[min_1, max_1], [min_2, max_2], \cdots, [min_n, max_n].$$

在此过程中，经度、纬度等用数值表示的属性易映射为 1 维区域，其他非数值属性应在编码前根据其分布特点构造自定义的转化方法，例如某数据的时间属性为 2023 年 7 月 1 日 9: 00 时，通过分析发现该历史轨迹数据均为同一年份产生，则可以暂时忽略年份，仅对日期与小时进行转化，取每一分钟为实数 1，则该时间通过转化得到 181×24×60 + 9×60 = 261 180，由此可以推出时间维度的实数范围值为轨迹数据开始时间与结束时间对应转化生成的实数.

易知，在对 *n* 维区域进行第 1 次 2 分后，将得到 1 个 *n* 位二进制字符串，其每一位表示数据在该维度下落在了 2 分后的哪一区域中. 此后，每一轮划分，都将得到 1 个 *n* 位二进制字符串，将其按照划分顺序依次拼接，第 *m* 次划分后即得到 *mn* 位二进制字符串，字符串具有唯一性且前缀递归，而每一个字符串即代表了 1 个网格. 通过遍历数据得到每个网格中包含数据对象的个数，并算出网格密度，将密度低于阈值的网格进行合并，需要注意的是每次合并都需充分考虑 *n* 个维度的共同合并，每合并 1 次，字符串右移 *n* 位，最终得到在 *n* 维数据上构造的非均匀网格编码.

4.5 算法复杂度分析

在初始的均匀网格编码降维算法中，需要 2 重循环，第 1 重循环对获得的移动对象点投影值进行逐一遍历，获取该点 2 个维度的空间值，第 2 重循环是对编码从最低位到最高位依次计算赋值和位移，因此均匀网格降维算法的时间复杂度为 $O(M\times len)$，其中 M 为移动对象点个数，len 为编码长度.

在非均匀网格编码降维算法中，原本需要 2 轮循环，第 1 轮循环对轨迹数据进行遍历扫描，以统计每个均匀网格的密度，第 2 轮循环逐一分析每个均匀网格的网格密度是否到达阈值，若未到达阈值需要对网格进行合并，并将合并后网格的更新值做记录，直到网格合并满足条件为止. 但是这种算法的时间复杂度将为 $O(M)+O(Num)$，其中网格数 $Num=2^{len}$.

为提高降维效率，本文优化算法直接对需要合并的网格中轨迹点的 1 维编码进行右移 1 位的操作，同时对原始轨迹点增加标签属性，记录该点当前所在网格内含有轨迹数据点的总数，这样仅需通过 1 轮遍历轨迹数据点的操作，即可执行网格合并操作，且合并过程为位运算，时间复杂度为 $O(M)$.

但轨迹数据点的规模庞大，因此 $O(M)$ 级别的时间复杂度仍然需要耗费大量的时间，于是本文引入前缀编码分组与子轨迹 *Bbox* 提取 2 种优化方法，在对前缀编码进行分组后，所有的轨迹数据点就按照前 5 次 2 分后所在网格的编码前缀被分配到不同的数组中，通过数组大小就可以明显区分聚集程度大的网格和聚集程度小的网格，对于密度明显大于密度阈值的分组避免遍历，从而节省时间. 对于子轨迹 *Bbox* 提取，按照不同时间区间进行划分，将指定区间内的所有数据点用 *Bbox* 的中心点代替，进一步压缩了需要处理的轨迹数据点的规模，使得时间复杂度从 $O(M)$ 转化为小于 $O(M/m)$，其中 m 为子轨迹 *Bbox* 中含轨迹点数量的平均值，后续实验结果表明降维时间因引入优化方法而大幅度减少.

5 基于网格降维的轨迹查询算法

轨迹数据在经过网格降维后，有效压缩了数据规模，但对于数据库的上层应用来说，轨迹数据的查询服务才是关键内容. 为考察经过均匀网格降维与非均匀网格降维后数据在数据库查询操作中的效率，本文采用了 2 类经典的查询算法，即范围查询和最近邻查询.

由于轨迹数据本身蕴含着移动对象的自主行为模式特征，并具有一定的整体环境规律，这就导致轨迹数据呈现范围广、聚集多、分布不均的特点，使查询时容易产生数量较多的子任务，且查询子任务之间的工作分配不均衡，所以需要根据现有的降维算法与数据存储模型对查询算法进行改进设计.

5.1 索引建立

在对降维后的轨迹数据进行查询操作前，需要在数据上建立索引，B 树索引仅支持 1 维数据的存储和查找，所以仅在没有时间维度的查询要求下有效. 为了使查询可以支持不同时间粒度，本文在降维数据上结合时间建立 R 树索引，这是一种面向移动对

象数据库的动态索引算法，也是目前移动对象数据库中应用最为广泛的索引之一.

R 树中的每个非叶子节点存放的是数据集空间中的一个矩形框和指向下一层节点的指针，该矩形框是包含其所有子节点的最小矩形框(minimum bounding rectangle，MBR)，在本文所建立的 R 树中，这个 2 维 MBR 表示的是由降维后 1 维值与时间 2 个维度构成的区域，其中时间存储为 Instant 格式，该格式定义为

year-month-day[-*hour*:*minute*[:*second*[.*millisecond*]]]，

其中方括号内参数为可选值，因此若数据集中时间参数精度足够，查询时间粒度可精确至毫秒级. R 树的叶子节点存放指向轨迹数据点的指针，R 树具有很好的灵活性，能够保持较高的空间利用率. 基于非均匀网格降维数据的查询工作需要经历 2 个阶段，网格降维为查询范围进行 1 轮过滤，筛选出可能满足查询条件的网格，缩小查询范围以降低查询时间，之后再对网格内的轨迹数据进行遍历查询，将符合查询条件的结果返回，确保查询的正确性.

5.2 范围查询

范围查询主要是指查询某对象在指定时间段内的轨迹数据，或是查询在指定时间段内在某个指定空间区域间的轨迹等. 在移动对象研究领域，时空范围查询相对来说应用得更为广泛，通过给定时间段范围与空间区域范围，查询在此范围内的轨迹，算法 3 对非均匀网格降维数据开展范围查询并进行描述，图 5 展示了本文提出的这一查询过程.

算法 3. 范围查询算法 *RMP-RQ*.

输入：查询空间区域对角点 *ss*，*se*，查询时间区域 *TR*，R 树索引 *R-tree*；

输出：查询结果 *Res*.

① *Res* ← ∅；

② *ss′*，*se′* ← *ReduceDimension*(*ss*)，*ReduceDimension*(*se*)；

③ *prefix* ← *GetCommonPrefix*(*ss′*，*se′*)；

④ *SearchMBR* ← *get*(*prefix*，*TR*)；

⑤ *N* ← *GetRoot*(*R-tree*)；

⑥ if *N* 是叶子节点

⑦ *Res* ←获取 *SearchMBR* 中的所有点；

⑧ return *Res*；

⑨ else

⑩ while *P* ≠ null

⑪ 根据 *SearchMBR* 转移节点并更新 *P*；

⑫ *P* ← *R-tree.Pointer*(*N*)；

⑬ if *Intersect*(*P.MBR*，*SearchMBR*)

⑭ *i* ← *SmallestNumberNode*(*P*)；

⑮ while *i* ≠ null

⑯ if *i.gird* 在查询范围内

⑰ *Res* ← *i.grid* 内的所有点；

⑱ else if *i.grid* 与查询范围相交

⑲ *Res* ←查询范围中的点；

⑳ end if

㉑ 移动至下个叶子节点并更新 *i*；

㉒ end while

㉓ end if

㉔ end while

㉕ end if

㉖ return *Res*.

算法 3 具有查询一致性，其第 1 步是通过给定 2 个对角点限定查询范围，以形成对于数据的约束条件，计算出降维后 2 个对角点的 1 维编码；第 2 步通过比较得到最长公共前缀，在基于非均匀网格编码和时间生成的 R 树中，相同前缀的叶节点处于同一子树中，因此通过最长公共前缀筛选出指定子树，并遍历该子树的所有叶节点，每个叶节点表示 1 个划

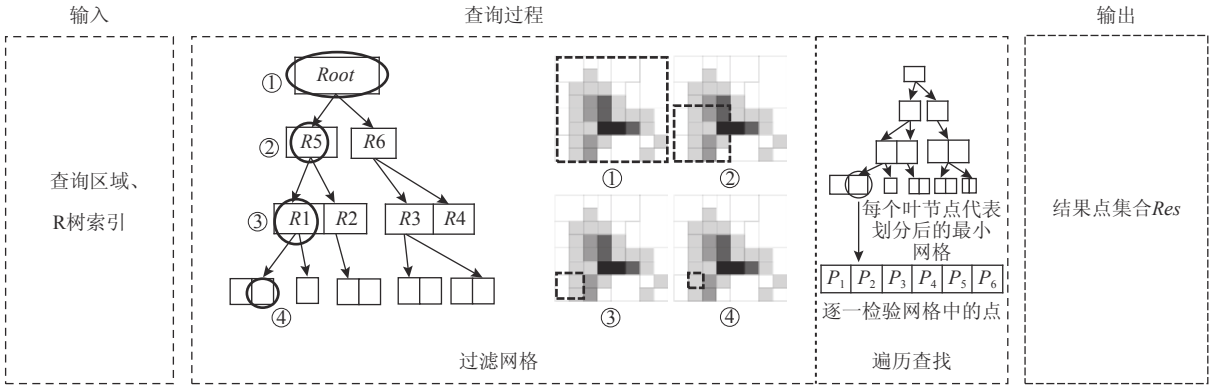


Fig. 5 Workflow of the range query operation

图 5 范围查询操作工作流

分后的最小网格；第 3 步通过比较网格边界与查询范围，得到网格与查询范围的包含关系. 对于包含在查询范围内的网格，它里面所有的数据点均是结果点；对于与查询范围交叉的网格，需要遍历它里面的数据点并选取符合查询条件的点作为结果点，最终将查询结果返回.

5.3 最近邻查询

最近邻查询是指，在数据库中查询与指定轨迹相邻近的轨迹，最近邻查询面向的查询对象有 2 种，可以面向某个数据点进行最近邻查询，也可以面向某段指定的轨迹进行最近邻查询. 基于数据点的查询通过指定坐标与时间，查询与它相邻近的轨迹. 基于轨迹的查询给出的则是一段轨迹与时间.

算法 4. 最近邻查询 *RMP-NQ*.

输入：查询点 Q，R 树索引 *R-tree*；

输出：查询结果 *Res*.

①$Res \leftarrow \varnothing$；

② $Q' \leftarrow ReduceDimension(Q)$；

/*对查询点进行降维，得到降维后 1 维网格编码*/

③ $N \leftarrow GetRoot(R\text{-}tree)$；

④ if N 是叶子节点

⑤ $Q_{\min} \leftarrow MinDistance(Q, N)$；

⑥ $Res \leftarrow Q_{\min}$；

⑦ return *Res*；

⑧ else

⑨ Set P = *R-tree.Pointer*(N)；

⑩ while *NonIntersect*(*P.MBR* , Q')

⑪ 根据 Q' 移动至下一节点并更新 P；

⑫ end while

⑬ $Q_{\min} \leftarrow MinDistance(Q, P)$；

⑭ 根据指针 P 找到相邻的网格；

⑮ for *iter* ← 1 to 相邻网格数量

⑯ *Distance*(Q, 相邻网格)；

⑰ if 存在更近邻

⑱ 更新 $Q_{\min}$；

⑲ end if

⑱ end for

⑲ $Res \leftarrow Q_{\min}$；

⑳ end if

㉑ return *Res*.

如算法 4 所示，本文算法是在已执行降维操作的基础上进行最近邻查询，第 1 步对查询点 Q 进行非均匀网格降维操作，因为在划分非均匀网格的过程中采用了 GeoHash 的编码方式，所以充分保留了数据点的空间位置关系，因此第 2 步可以通过比较数据点的非均匀网格编码值得出最近邻结果，若该区域只有 1 个网格，则在网格中寻找最接近该数据点的邻居点，否则对其所在网格即外围一周网格内的数据点进行遍历计算，以查找最近邻居. 最后，结束循环，返回最近邻居点.

算法 4 具有查询一致性，如图 6 所示，对查询点 Q 所在网格及其外围相邻的网格进行遍历，采用欧氏距离得出最近的数据点，结果仅有 1 个数据点. 对查询点 Q 的非均匀网格编码进行坐标的位序列分解，并使用简单的二进制加法和减法定位相邻网格的坐标，再进行交叉编码后即可得到其外围相邻网格的编码值，进而开展遍历查询.

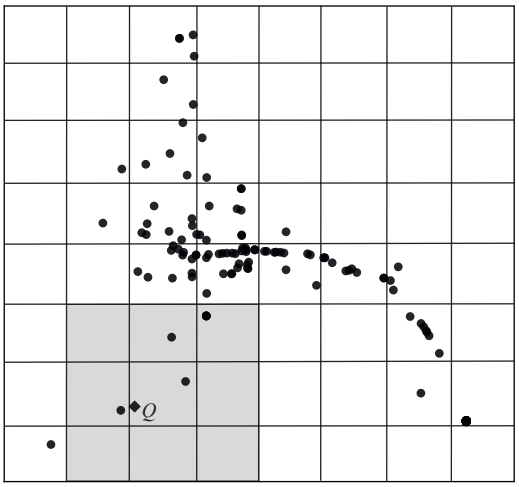


Fig. 6 Illustration of the nearest neighbor query

图 6 最近邻查询示意图

6 实 验

为验证本文所提出的均匀网格降维算法与非均匀网格降维算法的效率，以下为使用 C++编译语言在 Ubuntu20.04 环境下实验的过程与结果说明. 本文在可扩充数据库 SECONDO 中添加操作代码，实现均匀网格降维、非均匀网格降维与其他 3 种基准降维方式，并进行降维效果对比. 实验环境为：Intel® Core™ i7-5600U CPU @ 2.60 GHz 2.59 GHz，RAM 16.0 GB，Linux 5.15.0-76-generic.

6.1 数据集

为保证实验的完整性与可靠性，本文在研究过程中选取了 4 个具有不同特点的数据集，使得数据集既包含真实的城市出租车数据、人工合成的车辆

数据, 也包含存在线性分布特点的随机生成点集.

轨迹数据集基本属性如表 2 所示, 分别为: 2008 年 2 月的北京市出租车轨迹点, 规模为 3 200 余条; 2023 年 10 月的深圳市单日出租车轨迹点, 规模为 300 余条; SECONDO 移动对象数据库中的柏林市根据时间调度模拟生成火车轨迹点, 规模为 500 余条.

Table 2 Table of Attributes of Trajectory Data Set

表 2 轨迹数据集属性表

城市	时间范围	轨迹条数	轨迹点数	采样频率/s
北京	2008 年 2 月 2 日 15 时至 8 日 15 时	3 203	5 222 752	300
深圳	2013 年 10 月 22 日 0 时至 24 时	302	917 006	30
柏林	2003 年 11 月 20 日 6 时至 9 时	562	51 544	120

在实验预处理阶段, 将上述数据集均处理为统一数据格式: *ID* 为 Int 类型, 日期和时间为 String 类型, 经度和纬度为 Float 类型.

为进一步测试本文提出的算法在分布均匀的轨迹数据上的性能, 通过随机函数生成了以北京市某地点坐标(116.231 7, 39.542 7)为期望数组, 维数设置为 2, 2 个维度的协方差矩阵参数分别为 [1 , 1.5], [1.5 , 4] 的数据点, 数据点两两之间不重复, 数据规模为 1 000 000.

6.2 实验设置

考虑到非均匀网格降维的编码长度对于降维性能的影响, 本实验使用深圳市数据集分别对 16b, 32b, 48b, 64b 这 4 种长度的降维编码进行了空间位置关系评估, 并计算得到 4 种编码方式下的空间位置相似度, 如图 7 所示.

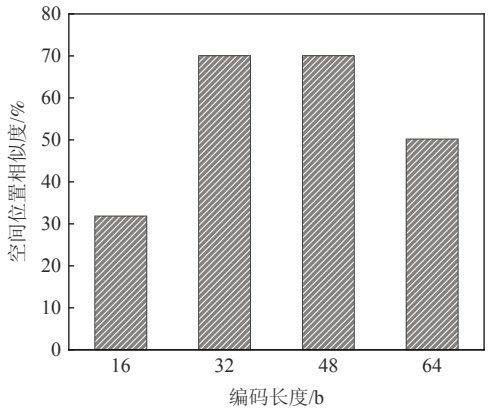


Fig. 7 Spatial position similarity under different coding lengths

图 7 不同编码长度的空间位置相似度

经过实验统计, 在 32b 编码与 48b 编码情况下的空间位置相似度完全一致, 也就是说针对当前的轨迹数据, 32b 编码划分的网格足以将轨迹数据点按照空间特点进行划分, 剩余距离十分靠近的对象点即使再度划分也不再具备优势, 甚至可能由于过度划分而破坏点与点之间的相关性. 因此, 本实验所选取的非均匀网格编码长度为 32b.

为进一步优化比较时间, 取编码前 5b 进行 Base32 的编码转换, 选取前 5b 进行编码的依据为 Base32 以 ASCII 码进行数据编码, 实现简单且被广泛通用; 前 5b 编码最多划分为 32 组, 较为合适, 且针对轨迹数据点 32b 编码, 取前 5b 可以有效避免因选择位数过少而使得数据划分过于密集、因选择位数过多而使得数据过于分散的极端情况. 本文实验部分所用数据集按照前 5b 编码可将全部数据划分为 8~10 组. 本文通过在移动对象数据库中定义一个 const char *base32_codes*[] 直接实现, 在为对象点进行 GeoHash 编码时, 将编码按照 Base32 进行分类, 存入一个 2 维 vector〈vector〈struct〉〉中, 这样再次进行比较时, 仅需对前 5b 中 Base32 值相同的对象点编码进行遍历比较即可, 时间开销进一步优化.

在对测试数据集进行网格编码分组时, 取前 5b 编码最大阈值为 32 组的情况下, 出现 8 个不同的编码分组, 其数量分布如图 8 所示.

为对比本文所提出非均匀网格降维算法的综合

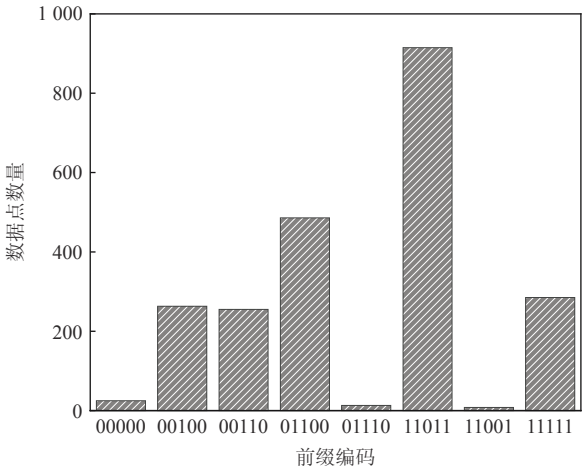


Fig. 8 Prefix grouping diagram of test data set

图 8 测试数据集前缀分组图

性能，实验选取了 3 种基准降维方式：Z-order 曲线、Hilbert 曲线和金字塔 (Pyramid) 技术，与本文提出降维算法所得实验结果进行对比分析.

6.3 降维实验

本文对于降维算法的评价分为 2 个部分：降维效果和降维性能. 其中降维效果的评估，本文提出了一种基于空间位置关系的评估方法，其旨在判定降维后的数据是否仍然保留原本的空间位置关系，同时也为了使非均匀网格降维方式得到的数据能在同一维度下与其他 4 种基准降维后得到的数据进行对比，以评估非均匀网格降维的优势；降维性能的评估则以降维所需时间为指标，最后通过实验结果综合分析 2 个评价指标下不同降维算法的表现.

本文提出的空间位置关系相似度评价的具体指标及计算过程详见数据降维评价指标(附录 A)，以下对其进行简要说明，通过聚类算法生成指定数量的标记点，然后分析降维前后与每个轨迹数据点距离最近的标记点是否更改.

对轨迹数据点 P 定义整型标签值 *Label*，标签值 *Label* 是距离其最近的标记点 *ID*，在降维后由于轨迹数据点和标记点均降为 1 维值 P'，*Label*′，因而在 1 维空间下重新计算得到距离每个点最近的标记点 *ID* 作为降维后的标签值. 通过计算将降维前后标签值相同的数据点数量占整个数据集总数的比重记为该算法的空间位置关系相似度 *Sim*：

$$Sim=\frac{Count(Label_P=Label_{P'})}{Count(P)}\times 100\%, \qquad (2)$$

其中 *Count* 为计数函数. 从式(2)易知，*Sim* 值越大，降维前后轨迹数据的空间位置关系保留程度就越大，若所有点的空间位置关系均得以保留，则空间位置关系相似度 *Sim*=100%.

为了验证本文提出 UGC 与 NGDR 算法在降维过程中对轨迹数据点空间位置关系的影响，对 4 个数据集进行降维，得到的空间位置相似度结果如图 9 所示.

通过图 9 的结果，可以看出在针对真实轨迹数据、人工合成轨迹数据以及随机生成点数据进行降维时，本文提出的 UGC 算法在部分数据集上表现出优势，而进一步优化得到的 NGDR 算法对于轨迹数据点之间空间位置关系的保留均优于其他基准降维算法，其中对于深圳市出租车数据进行降维后所得的空间位置相似度优势最大，能达到 82.5%，比基准方式中效果最好的 Pyramid 算法高 14.48 个百分点.

为了进一步验证本文所提出降维算法的执行效率，实验还对 UGC 算法、NGDR 算法与 3 种基准降维算法在处理数据量为 910 000 余条的深圳市数据集时所耗费的 CPU 时间进行了统计分析，所得结果如图 10 所示. 在 NGDR 中，计算每个网格的密度值需要时间 $O(Num)$，其中 *Num* 为网格总数，*Num* 根据编码长度以指数级增长，合并网格后更新编码需要时间 $O(M)$，M 为轨迹数据点总数. 因此初始状态下的 NGDR 复杂度为 $O(Num)+O(M)$，而其他降维算法仅需对数据点进行计算编码，复杂度仅为 $O(M)$，所以未优化的 NGDR 降维时间代价较大.

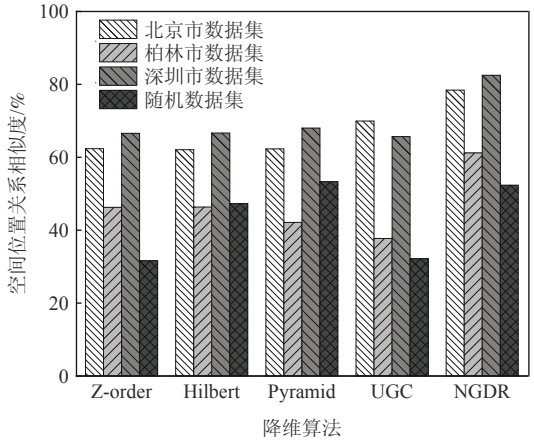


Fig. 9 Comparison of spatial position similarity of different dimensionality reduction algorithms

图 9 不同降维算法的空间位置相似度对比

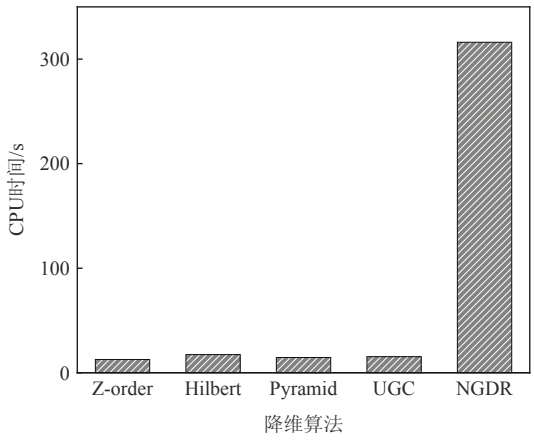


Fig. 10 Comparison of dimensionality reduction time of Shenzhen dataset

图 10 深圳市数据集降维时间对比图

通过对于深圳市数据集的降维性能对比，综合分析空间位置关系相似度和 CPU 所用的时间，具体对比结果如图 11 所示. 由图 11 可以发现虽然在空间位置关系相似度上，NGDR 体现出明显优于其他降维算法的趋势，但是在降维时间上，NGDR 具有明显

劣势, 虽然数据降维在数据库操作中仅需执行 1 次, 但降维时间仍然是衡量一个降维算法是否高效的重要因素, 因此本文对 NDGR 研究做出进一步优化.

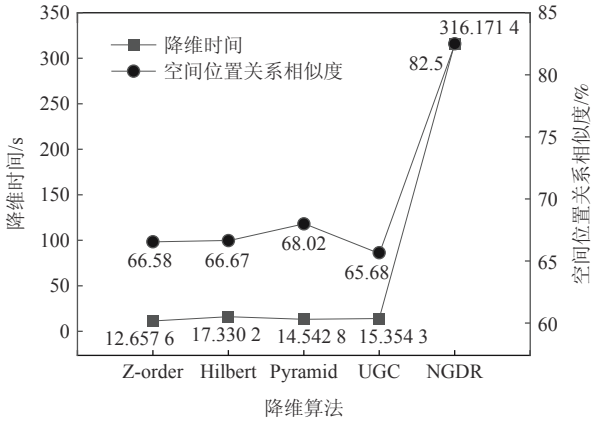


Fig. 11 Comparison of the initial performance of different dimensionality reduction algorithms

图 11 不同降维算法的初始性能对比

通过前缀分组预处理, 将前 5b 前缀相同的编码集中存储, 并记录存储位置, 此后合并网格更新编码仅需遍历前缀相同的编码值即可. 同时, 不再通过遍历网格方式计算密度值, 而是根据轨迹点编码直接计算落在同一网格中的轨迹点数量是否超过阈值, 节省了遍历空网格的时间. 修改后的非均匀网格算法可避免一轮 2 重循环, 降维时间较分组前减少 68.3%, 效果显著.

在子轨迹划分优化时间方面, 本文分别对深圳市数据集以 1 min, 5 min, 10 min, 15 min, 30 min 这 5 种时间区间进行轨迹划分, 并分别记录随着划分区间越来越长, 轨迹在降维过程中的空间位置关系相似度和 CPU 所用时间的变化趋势.

通过实验发现, 降维操作时间随着划分矩形框个数的减少急剧下降, 空间位置关系相似度也在不断降低, 特别是在以 10min 为划分区间变为以 15min 为划分区间时下降得尤为明显, 如图 12 所示.

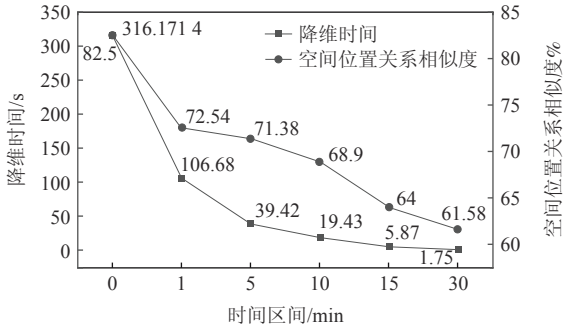


Fig. 12 Impact of different time intervals on performance

图 12 不同时间区间对性能的影响

由图 11 和图 12 可以看出以 10min 划分时, 降维时间比其他降维算法多, 空间位置相似效果比其他降维算法高; 但以 15min 划分时, 降维时间明显低于其他降维算法, 空间位置相似效果也相应地有所损失. 再次经过多轮实验后发现, 在最优划分区间的综合性能达到最佳, 与其他基准实验结果对比如图 13 所示.

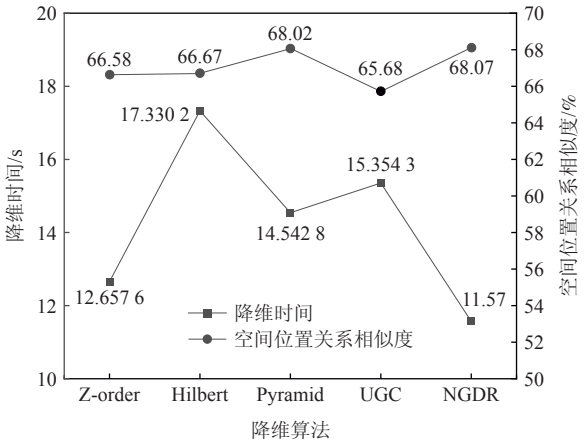


Fig. 13 Performance comparison of different dimensionality reduction algorithms after optimization

图 13 优化后不同降维算法的性能对比图

6.4 查询实验

为检验数据被不同降维算法转化为 1 维值后在数据库查询性能上的变化, 本文引入了 2 种查询技术进行测试.

在进行范围查询时, 对于深圳市数据集会生成 3 种不同大小的区域窗口进行查询, 区域窗口大小分别为 100% 轨迹区域、50% 轨迹区域以及 25% 轨迹区域. 所有降维数据均构建索引以支持查询, 查询结果如图 14 所示, 非均匀网格降维后的数据查询时间

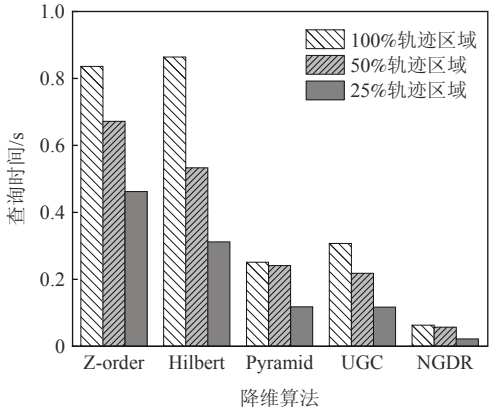


Fig. 14 Query time comparison of different regional windows range

图 14 不同区域窗口范围的查询时间对比

比其他降维方式更短，可至少减少 73.86%.

通过非均匀网格降维后的数据同样支持不同时间窗口的时间范围查询，对降维数据及其对应时间构建索引. 然后对数据集生成 5 种不同大小的时间窗口进行查询，时间查询窗口分别为 24 h, 12 h, 2.4 h, 1.2 h, 0.12 h, 查询结果如图 15 所示.

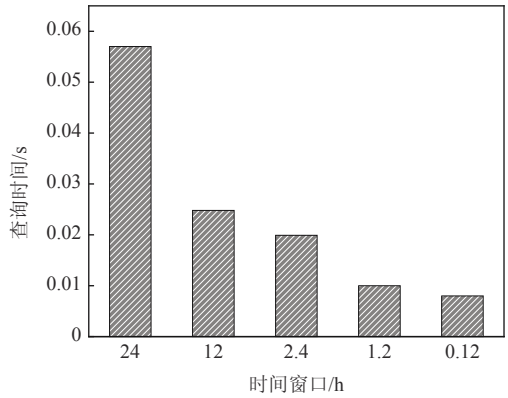


Fig. 15 Query time comparison of different time windows range

图 15 不同时间窗口范围的查询时间对比

在进行最近邻查询时，在深圳市数据集上随机生成 10 个查询点，进行 10 次最近邻测试，将测试的平均时间作为该种降维方式下最近邻查询所需的时间. 所有降维数据均构建 R 树索引以支持查询，查询结果如图 16 所示，非均匀网格降维后的数据查询时间同样表现出较优的性能，与其他基准降维后的数据相比，查询时间可减少 52.26%~71.02%.

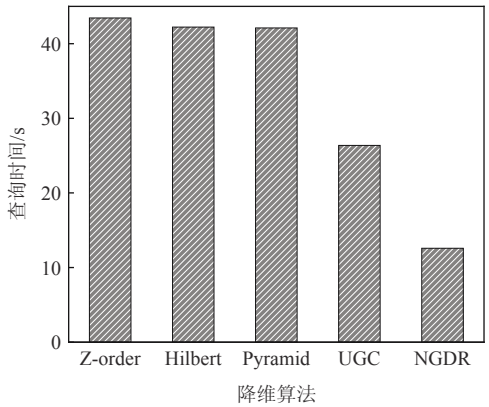


Fig. 16 Comparison of nearest neighbor query time

图 16 最近邻查询时间对比

本文通过在降维基础上引入不同应用场景下使用频率较高的空间查询算法，对不同降维方式下的数据性能进行对比，发现非均匀网格降维算法比其他基准降维算法性能更优，其查询代价因引入优化算法而有效减少，在范围查询上查询时间至少可减少 73.86%，在最近邻查询上查询时间至少可减少 52.26%，展现出较为良好的查询性能.

7 总结与展望

本文提出了基于网格划分的均匀网格降维算法和非均匀网格降维算法，采用 GeoHash 编码方式，将移动对象轨迹数据的空间坐标转化为 1 维字符串；并在移动对象数据库 SECONDO 中实现了均匀网格降维算法和非均匀网格降维算法，还提出了降维算法对于原始数据集空间相似性特点保留的评价指标——空间位置关系相似度. 使用北京市出租车、深圳市出租车、柏林模拟轨迹和随机生成数据等多个数据集，实现经典的 Z-order, Hilbert, Pyramid 降维算法，与本文提出的均匀网格降维和非均匀网格降维算法进行对比，比较不同算法之间的性能，对降维后数据进行范围查询和最近邻查询，以检验降维效果. 最终，实验结果显示，优化后的非均匀网格降维算法的范围查询时间较其他降维算法减少至少 73.86%，非均匀网格降维算法的最近邻查询时间较其他降维算法减少至少 52.26%. 同时，非均匀网格降维后数据的空间位置关系准确率可高达 82.5%，对于轨迹数据具有更好的降维效果.

未来的工作在现实空间中不可避免会有一些地理条件的限制（如建筑、河流等），因此 2 个对象之间的最短距离必须考虑障碍物的因素. 所以与传统无障碍物等相对理想情况下的最近邻查询研究相比，在障碍环境下的研究将更具现实意义.

作者贡献声明：巢成提出了算法思路和实验方案，并完成实验与论文撰写；蒲非凡提出实验方案并协同完成实验；许建秋提出指导意见并修改论文；高云君提出指导意见.

参 考 文 献

[1] Koldewey C, Rabe M, Dumitrescu R, et al. Introduction to the minitrack on data-driven services in manufacturing: Innovation, engineering, transformation, and management[C/OL]// Proc of the 56th Hawaii Int Conf on System Science. Hawaii: HICSS, 2023[2023-01-03].https://scholarspace.manoa.hawaii.edu/items/02775c54-ee1a-44b4-aafe-78bac2b11fc5

[2] Duentgen C , Behr T , Gueting R H . BerlinMOD: A benchmark for moving object databases[J]. Very Large Data Bases Journal, 2009,

18(6): 1335−1368

[3] Wu Wanqing, Zhao Yongxin, Wang Qiao, et al. A safe storage and release method of trajectory data satisfying differential privacy[J]. Journal of Computer Research and Development, 2021, 58(11): 2430−2443 (in Chinese)
(吴万青, 赵永新, 王巧, 等. 一种满足差分隐私的轨迹数据安全存储和发布方法[J]. 计算机研究与发展, 2021, 58(11): 2430−2443)

[4] Li Jiangneng, Wang Zheng, Cong Gao, et al. Towards designing and learning piecewise space-filling curves [C]// Proc of the 49th Int Conf on Very Large Data Bases. New York: ACM, 2023: 2158−2171

[5] Fujiwara T, Kuo Y H, Ynnerman A, et al. Feature learning for nonlinear dimensionality reduction toward maximal extraction of hidden patterns[C]// Proc of the 16th Pacific Visualization Symp (PacificVis). Piscataway, NJ: IEEE, 2023: 122−131

[6] Roweis S, Saul L. Nonlinear dimensionality reduction by locally linear embedding[J]. Science, 2000, 290(5500): 2323−2326

[7] Berchtold S, Christian B, Kriegel H P. The pyramid-technique: Towards breaking the curse of dimensionality[C]// Proc of the 1998 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 1998: 142−153

[8] Sundararajan R R. Principal component analysis using frequency components of multivariate time series[J]. Computational Statistics & Data Analysis, 2021, 157: 107164

[9] Zhu Xiaofeng, Zhang Shichao, Jin Zhi, et al. Missing value estimation for mixed-attribute data sets[J]. IEEE Transactions on Knowledge and Data Engineering, 2010, 23(1): 110−121

[10] Wang Jing. Research on manifold learning: Theories and approaches [D]. Hangzhou: Zhejiang University, 2006 (in Chinese)
(王靖. 流形学习的理论与方法研究[D]. 杭州: 浙江大学, 2006)

[11] Wang Jiaming, Shao Zhenfeng, Huang Xiao, et al. Deep locally linear embedding network[J]. Information Sciences, 2022, 614: 416−431

[12] Arena P, Patanè L, Spinosa A G. Robust modelling of binary decisions in laplacian eigenmaps-based echo state networks[J]. Engineering Applications of Artificial Intelligence, 2020, 95: 103828

[13] Zhang Peng, Qiao Hong, Zhang Bo. An improved local tangent space alignment method for manifold learning[J]. Pattern Recognition Letters, 2011, 32(2): 181−189

[14] Kruskal, Joseph B. Nonmetric multidimensional scaling: A numerical method[J]. Psychometrika, 1964, 29(2): 115−129

[15] Hamm K, Henscheid N, Kang Shujie. Wassmap: Wasserstein isometric mapping for image manifold learning[J]. SIAM Journal on Mathematics of Data Science, 2023, 5(2): 475−501

[16] Jeng Yihnen, Lin Sanyih, Lee Zongshiaw. Smoothing of the multiple one-dimensional adaptive grid procedure[J]. AIAA Journal, 2000, 38(12): 2353−2355

[17] Verma J P V, Mankad S H, Garg S. GeoHash tag based mobility detection and prediction for traffic management[J]. SN Applied Sciences, 2020, 2: 1−13

[18] Schler M, Grebhahn A, Schrter R, et al. QuEval: Beyond high-dimensional indexing à la carte[J]. VLDB Endowment, 2013, 6(14): 1654−1665

[19] Asano T, Ranjan D, Roos T, et al. Space-filling curves and their use in the design of geometric data structures[J]. Theoretical Computer Science, 1997, 181(1): 3−15

[20] Morton G M. A computer oriented geodetic data base and a new technique in file sequencing [J/OL]. Physics of Plasmas, 1966[2023-03-09].https://www.researchgate.net/publication/242678487_A_Computer_Oriented_Geodetic_Data_Base_and_a_New_Technique_in_File_Sequencing

[21] Wu Yuhao, Cao Xuefeng, An Zipeng. A spatiotemporal trajectory data index based on the Hilbert curve code[J]. IOP Conf Series Earth and Environmental Science, 2020, 45(9): 1403−1411

[22] Huang Jian, Zhao Jinling, Guo Yixiao, et al. The application on distributed geospatial data management based on Hadoop and the application in WebGIS[C/OL]// Proc of the 9th Int Conf on Agro-Geoinformatics (Agro-Geoinformatics). Piscataway, NJ: IEEE, 2021 [2023-03-12].https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9530350

[23] Lei Yi, Tong Xiaochong, Wang Dali, et al. W-Hilbert: A W-shaped Hilbert curve and coding method for multiscale geospatial data index[J]. International Journal of Applied Earth Observation and Geoinformation, 2023, 118: 103298

[24] Lin Xuelian, Ma Shuai, Jiang Jiahao, et al. Error bounded line simplification algorithms for trajectory compression: An experimental evaluation[J]. ACM Transactions on Database Systems, 2021, 46(3): 1−44

[25] Zheng Kai, Zhao Yan, Lian Defu, et al. Reference-based framework for spatio-temporal trajectory compression and query processing[J]. IEEE Transactions on Knowledge and Data Engineering, 2019, 32(11): 2227−2240

[26] Fikioris G, Patroumpas K, Artikis A, et al. Optimizing vessel trajectory compression for maritime situational awareness[J]. GeoInformatica, 2023, 27(3): 565−591

[27] Kulkarni T. Answering range queries under local differential privacy[C]//Proc of the 2019 Int Conf on Management of Data. New York: ACM, 2019: 1832−1834

[28] Ghane S, Kulik L, Ramamoharao K. A differentially private algorithm for range queries on trajectories[J]. Knowledge and Information Systems, 2021, 63(2): 277−303

[29] Li Song, Li Bohan, Yu Jiaxi, et al. Probabilistic threshold K-ann query method based on uncertain Voronoi diagram in Internet of vehicles[J]. IEEE Transactions on Intelligent Transportation Systems, 2020, 22(6): 3592−3602

[30] Wang Beiyi, Zhang Xiaohong, Wang Weibing. Feature matching method based on SURF and fast library for approximate nearest neighbor search[J]. Integrated Ferroelectrics, 2021, 218(1): 147−154

[31] Zhao Peng, Rao Weixiong, Zhang Chengxi, et al. SST: Synchronized spatial-temporal trajectory similarity search[J]. GeoInformatica, 2020, 24: 777−800

[32] Zheng Bolong, Weng Lianggui, Zhao Xi, et al. Repose: Distributed top-*k* trajectory similarity search with local reference point tries[C]// Proc of the 37th IEEE Int Conf on Data Engineering (ICDE).

Piscataway, NJ: IEEE, 2021: 708−719

[33] Mokbel M F, Aref W G, Kamel I. Analysis of multi-dimensional space-filling curves[J]. GeoInformatica, 2003, 7(3): 179−209

[34] Liu Jian, Liu Gaofeng. Algorithm of coordinates conversion in Gauss-Kruger projection[J/OL]. Computer Simulation, 2005, 10[2022-12-20]. http://en.cnki.com.cn/Article_en/CJFDTOTAL-JSJZ200510031.htm



Chao Cheng, born in 1999. Master. Her main research interest includes moving objects databases.

巢　成, 1999 年生. 硕士. 主要研究方向为移动对象数据库.



Pu Feifan, born in 1999. Master candidate. His main research interest includes time-series data management. (pff1999@nuaa.edu.cn)

蒲非凡, 1999 年生. 硕士研究生. 主要研究方向为时序数据管理.



Xu Jianqiu, born in 1982. PhD, professor, PhD supervisor. Senior member of CCF. His main research interests include data software, spatial databases, moving object databases, and extensible databases.

许建秋, 1982 年生. 博士, 教授, 博士生导师. CCF 高级会员. 主要研究方向为数据软件、空间数据库、移动对象数据库、可扩充数据库.



Gao Yunjun, born in 1977. PhD, professor, PhD supervisor. Senior member of CCF. His main research interests include database, big data management and analytics, and DB4AI and AI4DB. (gaoyj@zju.edu.cn)

高云君, 1977 年生. 博士, 教授, 博士生导师. CCF 高级会员. 主要研究方向为数据库、大数据管理与分析、DB 与 AI 融合.

附录 A.

1) 数据降维评价指标. 本文对于降维算法的评价分为 2 个部分——降维效果和降维性能. 其中降维效果的评估, 本文提出了一种基于空间位置关系的评估方法, 即空间位置关系相似度, 旨在判定降维后的数据保留原本空间位置关系的程度, 同时也为了使非均匀网格降维方式能在同一维度下与其他 4 种基准降维后得到的数据进行对比, 以评估非均匀网格降维的优势.

空间位置关系相似度计算的具体步骤是通过聚类算法生成指定数量的标记点, 然后对每个数据轨迹点按照欧氏距离进行划分, 记录与其最近的标记点 *ID* 作为标签值, 然后分析在降维之后的 1 维空间中与每个轨迹数据点距离最近的标记点是否一致, 将前后距离一致的数据点所占数据集的比例作为空间位置关系相似度结果值. 过程中引入变量与符号定义如表 A1 所示.

Table A1 Definition of the Symbol of the Spatial Location Relationship

表 A1 空间位置关系符号定义表

符号	说明
B	生成的标记点集
Label	整型标签值，记录与轨迹数据点距离最近的标记点 *ID*
U_i	i 点集，标签为 i 的所有数据点的集合
Center	中心点集，点集 U_i 的中心点的集合

2) 标记点集 B. 根据数据集生成的 3 个聚类中心点的集合 $B=\{b_1, b_2, b_3\}$.

初始状态下在轨迹数据点集中随机选择 3 个点作为标记初始点 $B^0=\{b_1^0, b_2^0, b_3^0\}$, 每个对象点选取距离最近的标记点 *ID* 作为 *Label*, 生成 U_1, U_2, U_3 这 3 个标记点集, 计算每个标记点集的中心点, 生成中心点集合 $Center^0$, 计算公式为:

$$Center_i=\left(\frac{\sum_{j=1}^{card[i]}x_j}{card[i]},\frac{\sum_{j=1}^{card[i]}y_j}{card[i]}\right), \tag{A1}$$

其中 x_j 表示第 j 个点的横坐标, y_j 表示第 j 个点的纵坐标, $Center_i$ 为中心点集 $Center^0$ 中的第 i 个点, $card[i]$ 表示 U_i 点集中元素的个数.

将得到的 $Center^0$ 作为新的标记点集 $B^1=Center^0$, 则 $b_1^1=Center_1$, $b_2^1=Center_2$, $b_3^1=Center_3$, 每个对象点重新选取距离最近的标记点 *ID* 作为 *Label*, 生成新的 U_1, U_2, U_3 这 3 个标记点集, 再次计算每个标记点集的中心点, 生成中心点集合 $Center^1$, 方式同上. 迭代计算, 直到生成的新的中心点集合 $Center^N=Center^{N-1}$, 此时的标记点集 B^N 即最终需要的 B.

3) 空间位置关系相似度 *Sim*. 对于任意对象点 P, 定义标签 $Label_P=l$, 当且仅当 $dis(P, l)$ 满足:

$$dis(P,l)=\min_{b_i\in B}\left|\sqrt{(x_{b_i}-x_P)^2+(y_{b_i}-y_P)^2}\right|, \tag{A2}$$

其中, b_i 为标记点集 B 中的点, l 为标记点集 B 中点的 *ID*, 对于任意 1 个数据点, 将这种不以绝对数值为衡量对象, 取标记点作为参照物的距离关系定义为空间位置关系. *Label* 记录与其距离最近的标记点 *ID*, 从而保留其与标记点的相对远近特征.

对象点 P 经过降维操作变为 1 维值 P' 后, 标记点集中所有点 b_i 也能得到降维后的 1 维值 b_i', 定义降维后的标签 $Label_{P'}=l'$, 当且仅当 $dis(P', l')$ 满足:

$$dis(P',l')=\min_{b_i'\in B'}|P'-b_i'|. \tag{A3}$$

若 1 维值类型为实数则计算欧氏距离, 若 1 维值类型为字符串则计算编辑距离. 再通过比较对象点 P 得到的 $Label_P$ 和 $Label_{P'}$, 得到降维前后的空间位置关系相似度 *Sim*:

$$Sim=\frac{Count(Label_P=Label_{P'})}{Count(P)}\times 100\%, \tag{A4}$$

其中, *Count* 为计数函数, 从式(A4)易知 *Sim* 值越大, 降维前后轨迹数据的空间位置关系保留程度就越大, 若所有点空间位置关系均得以保留, 则空间位置关系相似度 *Sim* 的值为 100%.