计算机科学
COMPUTER SCIENCE

基于道格拉斯-普克算法的路网轨迹学习索引结构

缪祝青，韩京宇，李彩云，王彦之，毛毅，张怡婷

引用本文

缪祝青，韩京宇，李彩云，王彦之，毛毅，张怡婷. 基于道格拉斯-普克算法的路网轨迹学习索引结构[J]. 计算机科学，2025，52(8)：136-145.

MIAO Zhuqing, HAN Jingyu, LI Caiyun, WANG Yanzhi, MAO Yi, ZHANG Yiting. Douglas-Peuker Algorithm Based Learned Index Structure for Road Network Trajectroy Data [J]. Computer Science, 2025, 52(8): 136-145.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.240600075

基于道格拉斯-普克算法的路网轨迹学习索引结构

缪祝青[1] 韩京宇[1,2] 李彩云[1] 王彦之[1] 毛 毅[1] 张怡婷[1]
1 南京邮电大学计算机学院 南京 210000
2 江苏省大数据安全与智能处理重点实验室(南京邮电大学) 南京 210000
(1022040911@njupt.edu.cn)

摘 要 近年来,基于位置服务的技术迅猛发展,产生了海量的路网轨迹数据。而路径范围查询作为一种路网轨迹查询类型,是支持其他查询类型的基础。为了实现对海量路网轨迹数据的高效索引,同时提供精确的路径范围查询服务,提出了一种基于道格拉斯-普克算法的学习型索引结构(Douglas-Peuker Based Learned Index Structure,DPLI)。首先将轨迹数据分为多个轨迹段,然后取轨迹段中的点作为轨迹数据的表征,利用映射函数将其映射为一维映射值序列,而后根据键值数量将其划分为多个数据分片。在分片内将首尾数据组成一条线段,然后计算其余数据点距离线段的拟合误差,将超过误差阈值的数据点作为新的线段端点,递归分割原有的直线段,直到所有数据点的拟合误差小于阈值,从而拟合分段线性函数。采用多个路网数据和轨迹数据进行了充分的实验,实验结果表明:与传统索引方法相比,DPLI具有更快的构建效率和磁盘访问效率;与学习索引方法相比,DPLI保持了构建效率的优势,并且达到了100%查询召回率。

关键词:位置服务;路网轨迹;学习型索引;范围查询;道格拉斯-普克算法
中图分类号 TP392

Douglas-Peuker Algorithm Based Learned Index Structure for Road Network Trajectroy Data

MIAO Zhuqing[1],HAN Jingyu[1,2],LI Caiyun[1],WANG Yanzhi[1],MAO Yi[1] and ZHANG Yiting[1]
1 College of Computer Science,Nanjing University of Posts and Telecommunications,Nanjing 210000,China
2 Jiangsu Key Laboratory of Big Data Security and Intelligent Processing,Nanjing University of Posts and Telecommunications,Nanjing 210000,China

Abstract In recent years,the technology of location-based services has developed rapidly,which has produced a large amount of road network trajectory data. As a type of road network trajectory query,path range query is the basis to support other query types. This paper proposes a Douglas-Peuker based learned index structure(DPLI) based on Douglas-Purker algorithm,in order to efficiently index massive road network trajectory data and provide accurate path range query service. Firstly,the trajectory data is divided into multiple trajectory segments,then the midpoint of the trajectory segment is taken as the representation of the trajectory data,and the mapping function is used to map the sequence of one-dimensional mapping values,and then the trajectory data is divided into multiple data fragments according to the number of key values. In the fragment,the first and last data are formed into a line segment,and then the fitting error of the distance line segment of the remaining data points is calculated. The data points exceeding the error threshold are taken as the new line segment endpoints,and the original line segment is recursively divided until the fitting error of all data points is less than the threshold,so as to fit the piecewise linear function. The results show that compared with the traditional index methods,DPLI has faster construction efficiency and disk access efficiency. Compared to learning indexing methods,DPLI maintains the advantage of build efficiency and achieves a 100% query recall rate.

Keywords Location services,Road network trajectory data,Learned index,Range query,Douglas-Purker algorithm

1 引言

由于移动设备、蜂窝通信网络和定位设备的广泛使用,路网轨迹数据规模呈现爆炸式增长[1]。通过查询路网轨迹数据并分析其中所蕴含的分布模式,研究人员可以捕捉和理解路网交通状况和人类流动模式[2]。多年来,人们已经研究许多类型的查询,如路径范围查询[2]、距离范围查询、k近邻查询[3-4]以及天际线查询等。本文主要专注于路径范围查询。路径范围查询由空间和时间范围组成,空间范围是路网中一条连续的路径,查询返回数据库中与该路径相交的所有轨迹[2]。

到稿日期:2024-06-11 返修日期:2024-09-04
基金项目:江苏省重点研发项目(BE2022065-5)
This work was supported by the Key Research and Development Project of Jiangsu Province(BE2022065-5).
通信作者:韩京宇(jyhan@njupt.edu.cn)

过去的几十年里，人们设计了许多传统多维索引结构来索引路网轨迹数据[5]，主要分为3类：数据驱动型索引[5-6]、空间驱动型索引[7-8]和映射型索引[9-11]。其通常依赖于树、网格或散列结构来检索数据。但是，随着数据规模的增大，索引本身的复杂程度也会增大，这将影响查询的效率[12]。例如，树型结构的索引，当数据量增大时，树的高度也会增加，导致搜索路径变长，查询性能降低。

近年来，研究人员提出了学习索引。该类方法使用机器学习模型代替传统索引，以减小存储开销并加速查询过程[13-15]。其通过机器学习模型来学习数据的分布模式，以此来建立查询键与存储位置之间的映射函数，从而实现对数据的高效索引[16-18]。针对多维数据，提出了许多学习索引方法，主要分为两类。一类为拟合布局式，如SageDB[19]、Z阶模型(Z-order Model，ZM)[20]、递归空间模型索引(Recursive Spatial Model Index，RSMI)[21]以及多维学习(Multi-dimensional Learned，ML)[22]等。该类方法的原理是利用模型来拟合数据的分布模式，预测数据在排序数组中的位置，并在误差范围内查找数据。另一类为生成布局式，该类方法将数据按照特定分布放置在磁盘上，当新数据到来时按照原有的分布进行数据的放置和删除。如文献[11]提出的空间数据学习索引结构(Learned Index Structure for Spatial Data，LISA)，首先将多维数据映射为一维数据，然后将一维数据分为多个数据分片，对每个数据分片拟合分段线性模型，以确定数据的存储位置。在数据更新方面，其根据已学习的分布进行数据的插入和删除。

然而，目前的学习索引仍存在不足，主要有两点：1)模型拟合运算量大，训练效率低；2)模型预测准确率较低，存在查询召回率的问题。此外，直接应用先前的方法，无法实现路网轨迹的路径范围查询，原因是索引不包含路网信息。此外，若轨迹数据点采样稀疏，还可能会引发查询召回率的问题。

为解决上述挑战，本文提出了一种基于道格拉斯-普克算法的学习索引结构(Douglas-Peuker Based Learned Index Structure，DPLI)，其基本框架分为3个模块：数据预处理、分片预测模型以及本地模型。数据预处理模块对路网数据构建pos-t平面，然后将轨迹划分为多个轨迹段，将其中的点作为最小的数据存储单元，映射至pos-t平面上，然后通过映射函数将中点映射为一维映射值，再对其排序和分组。对每组映射值按照固定分片大小取分割点后输入模型。分片预测模型模块采用分片预测模型DPSP来预测数据所在的分片位置，以实现高效训练和精确查询，从而解决训练效率和查询召回率的问题。最后，本地模型模块为分片构建本地模型，以管理分片内的磁盘页分配。本文与文献[11]的框架相似，其主要区别在于添加了数据预处理模块，以及提出了新的分段线性模型训练方法。

本文的主要贡献有以下4点：

1)提出了一种基于道格拉斯-普克算法的分片预测模型DPSP，解决了之前方法训练效率低、误差不可控的问题。进而提出了针对路网轨迹数据的学习索引结构DPLI，用于路网轨迹数据的高效索引并支持路径范围查询。

2)分析了DPSP模型的参数对预测准确率的影响，同时分析了预测准确率与模型参数之间的量化关系。

3)针对路网轨迹数据设计新的数据预处理方案，即将路网信息嵌入到数据结构中，并将多维数据转化为一维映射值，实现了数据的高效存储。

4)在多个数据集上进行了大量实验，结果表明所提方法能很好地支持路径范围查询，同时具备查询召回率高、训练效率高以及空间占用小的优势。

2 相关工作

根据数据放置方式和模型选择之间的相互关系，可以将学习索引分为两类：拟合布局式和生成布局式。

2.1 拟合布局式学习索引

拟合布局式将数据按照一定的顺序放置到存储器中，然后根据数据在排序数组中的位置来训练存储器地址模型。SageDB[19]是一个包含学习多维索引的学习数据库系统，其中的数据点被依次排序，并沿着维度序列划分为大小相等的单元格，然后按照点所在的单元排序，并学习函数来预测这些点的次序。Z阶模型(Z-order Model，ZM)[20]通过Z阶曲线对空间数据应用多阶段模型索引(Multi-stage Model Index，MMI)。它建立了一个阶段模型，每个阶段包含一个或多个人工神经网络或线性模型。文献[21]提出了递归空间模型索引(Recursive Spatial Model Index，RSMI)来处理点数据。它引入秩序空间来建立点的排序，并将点分组成块来学习存储地址模型。文献[22]提出的多维学习(Multi-dimensional Learned，ML)索引对空间样本进行划分，并将其转换为相对于其最近参考点的一维距离值。文献[23]将空间插值函数作为学习索引来支持对空间数据的范围查询和kNN查询。特别地，它对空间数据进行采样以构建自适应网格，并使用样本数据作为输入来拟合空间插值函数。给定空间搜索键，利用拟合的空间插值函数预测其近似位置，然后进行局部二分查找，找到目标键。

2.2 生成布局式学习索引

生成布局式为近年来提出的学习索引新范式。其原理是将数据按照固定的模式存储在磁盘上，然后通过模型学习其特定的分布模式，从而实现数据索引。文献[11]提出了空间数据的学习索引结构(Learned Index Structure for Spatial Data，LISA)，首次提出从生成存储分布的角度训练学习模型的一般性框架。其方法是，沿一系列轴将空间划分为多个网格单元，并对单元进行编号。然后，LISA根据单元格的边界构造部分单调函数，将数据从多维空间映射到一维空间。接着将映射值分组，每组映射值对应一个分段线性函数，包含多个数据分片(shard)。之后使用分片预测函数SP来学习映射值和分片编号之间的映射关系，最后使用本地模型(Local Model，LM)来管理分片中数据的页面分配。

3 基础概念和基本框架

本章首先介绍问题的形式化定义，预处理过程以及轨迹数据的数据结构，然后介绍方法的基本框架。

3.1 基础概念

定义1(有向路网) 有向路网G是一个有向无环图，由

一个道路交叉口集合 J 和有向路段集合 S 组成，记为 $G=(J,S)$. 其中的每个有向路段 $s\in S$，为一个三元组 (sid,j^s,j^e)，其中 sid 表示路段编号，j^s 表示路段的起始交叉口，j^e 表示路段的终点交叉口。

定义 2（相对位置） 相对位置是有向路网中的某个地理平面坐标点的一维映射值，为浮点数，记为 pos，其整数部分表示所在路段的编号，小数部分表示该坐标点距离路段起点的相对距离。例如，pos=3.8，表示该移动对象位于 3 号路段，到路段起点的距离是该路段总长度的 0.8 倍。

定义 3（轨迹） 一条轨迹是一个按时间顺序排列的 GPS 点序列，记为 $T^p=\{p_1,p_2,\cdots,p_n\}$，其中 GPS 点为一个三元组，记为 $p_i=(lon_i,lat_i,t_i)$，其中 (lon_i,lat_i) 表示（经度，纬度），t_i 表示时间戳；相邻两个点 p_i,p_{i+1} 之间的时间戳之差为采样率。

定义 4（轨迹段） 轨迹段是轨迹的一段子轨迹，记为 $tu=(rid^i,p^i_{st},p^i_{ed})$，其中 rid^i 表示轨迹编号；p^i_{st},p^i_{ed} 表示移动对象的起点时空坐标和终点时空坐标；$p^i_{st}=(lon^i_{st},lat^i_{st},t^i_{st})$，其中 (lon^i_{st},lat^i_{st}) 表示（经度，纬度），t^i_{start} 代表移动对象进入路段起点的时间戳，类似可定义 $p^i_{ed}=(lon^i_{ed},lat^i_{ed},t^i_{ed})$。

定义 5（数据分片） 数据分片是定义在 $[a,b)\subseteq[0,+\infty)$ 上的映射函数 M 的前象，记为 $S=M^{-1}([a,b))$。

3.2 基本框架

图 1 给出了 DPLI 学习索引方法的基本框架，总共包含两个部分：索引构建和路径范围查询。

1）索引构建：如图 1 上半部分所示，首先将原始轨迹数据进行预处理，将其转换为映射值序列，然后将映射值序列按照数据分片大小，利用道格拉斯-普克算法，拟合多个分段线性函数，组成分片预测模型，最后结合磁盘页信息，建立本地模型，完成索引的构建。

2）路径范围查询：如图 1 下半部分所示，输入查询边界后，结合映射函数，将数据映射为一维映射值，得出查询覆盖的映射值范围，然后通过分片预测模型 DPSP 来预测对应的分片，最后调用分片所对应的本地模型来进行磁盘搜索，从而获得查询结果。

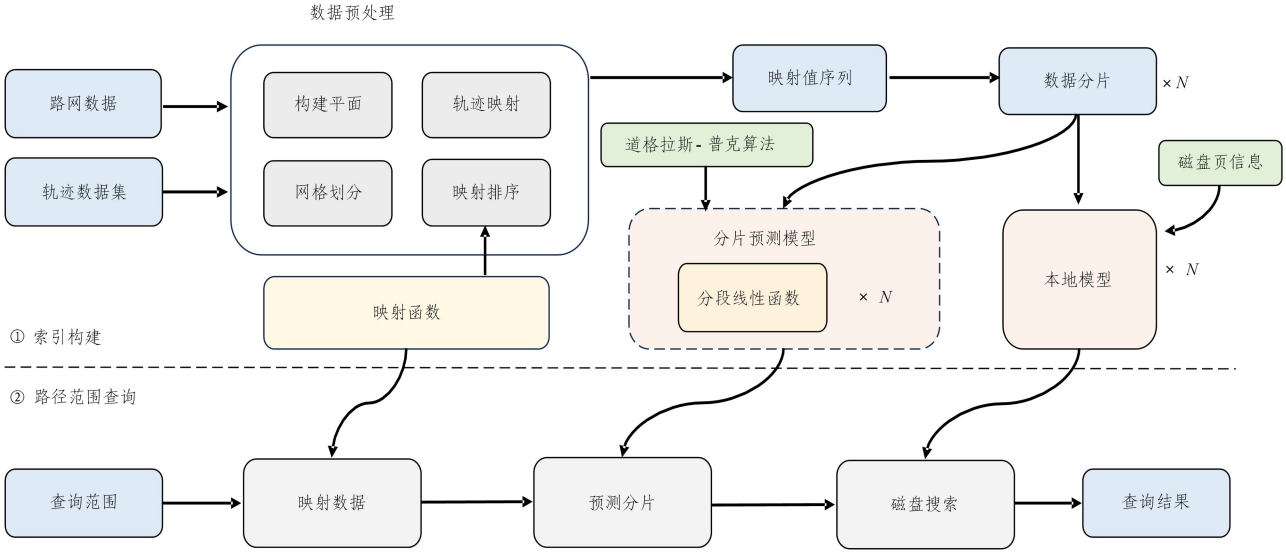

图 1 DPLI 方法的基本框架

Fig. 1 Framework of DPLI

4 数据预处理

数据预处理包括 4 个部分：1）构建 pos-t 平面构建；2）映射轨迹至 pos-t 平面；3）平面网格划分；4）映射数据并排序。

4.1 pos-t 平面构建

首先采用文献[2]中的方法，对路网数据构建 pos-t 平面。

定义 5（pos-t 平面） pos-t 平面是一个二维平面时空坐标系，其横轴为相对位置 pos，纵轴为时间 t。

构建方法为：1）按次序为路段编号；2）根据最大的路段编号确定横轴上界；3）根据数据集的最大时间戳确定纵轴坐标上界。

4.2 轨迹映射

构建 pos-t 平面后，使用文献[24]中的方法进行地图匹配，将原始轨迹映射到路网上。然后将轨迹中的轨迹点坐标映射至 pos-t 平面中，从而获得路网轨迹数据。

在实际场景中，由于设备老化、环境干扰等因素，轨迹点之间的采样间隔可能会很大。如果直接将轨迹点作为数据存储，则很有可能影响到范围查询的精确度。如图 2 所示，虽然查询边界与轨迹相交，但是由于轨迹点 a,b 之间的时间跨度大于查询的时间跨度，查询边界不覆盖任何轨迹点，因此查询失败。

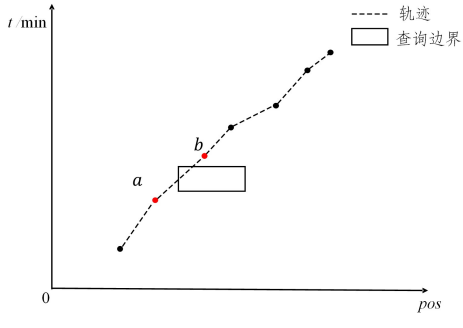

图 2 查询精确度问题示例

Fig. 2 Example of query accuracy problem

为了避免上述情况，不直接将轨迹点作为最小数据单元存储；而是切分轨迹，以获得轨迹段 tu。如果一条轨迹跨越了多个路段，则将其按照道路交叉口进行切分，以确保每个轨迹段 tu 与一条路段 s 相对应。然后将轨迹段的中点作为最小数据单元，排序后存储在磁盘上。例如，给定轨迹段 $tu=(rid,p_1,p_2)$，其中 p_1 和 p_2 为起点、终点的时空坐标，取 tu 中点 $tu_{\text{mid}}=(rid,p_1^{\text{mid}},p_2^{\text{mid}})$，其中 $p_1^{\text{mid}}=\frac{p_1}{2}$，$p_2^{\text{mid}}=\frac{p_2}{2}$。

最终存储数据为三维数据点 $(rid,pos^{\text{mid}},t^{\text{mid}})$，其中 rid 表示轨迹编号或是移动对象编号，$pos^{\text{mid}}$ 表示轨迹单元的长度中点，t^{mid} 表示轨迹单元的时间中点。模型通过二维数据点 $(pos^{\text{mid}},t^{\text{mid}})$ 来学习数据的分布模式。查询时，先扩大查询边界（详见第 5 章），然后再筛选其中的数据，从而解决查询召回率的问题。如图 3 所示，经过上述操作后，只需判断扩大后的查询边界是否包含轨迹段中点，就可以确定查询结果。

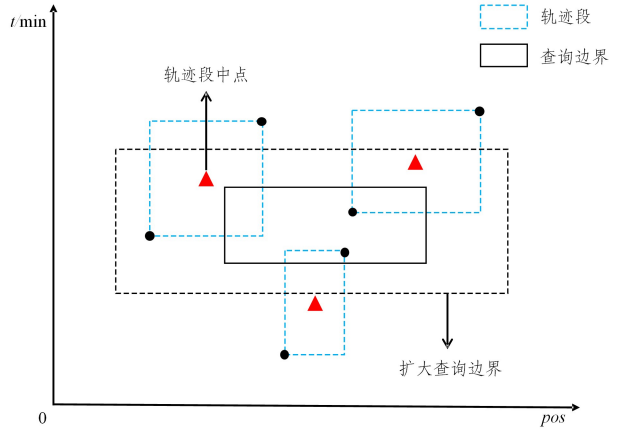

图 3　取中点后的查询示例

Fig. 3　Query example after the midpoint is taken

4.3　平面网格划分

轨迹映射完成后，需对 pos-t 平面划分网格。采用固定长度的划分策略，对于二维数据点的单个维度 $x_i(i=1,2)$，按照长度 L_i 将其划分为 T_i 个部分。使用 $\Theta_i=[\theta_1^{(i)},\cdots,\theta_{T_i}^{(i)}]$ $(|\theta_j^{(i)}-\theta_{j+1}^{(i)}|=L_i,1\leqslant j<T_i)$ 来表示单个维度 x_i 上的分割点序列。将整个平面 V 表示为一系列相邻网格的集合，记为 $V=\bigcup_{n=0}^{T_0\times T_1-1}C_n$，其中 $C_n=[\theta_{j_0}^{(0)},\theta_{j_0+1}^{(0)})\times[\theta_{j_1}^{(1)},\theta_{j_1+1}^{(1)})$ 表示二维平面网格，$n=j_0\times T_1+j_1$ 表示总网格数。

由于本文中数据是按照固定分组长度 L_i 划分的，因此不存储每个维度的分割点序列 Θ_i，而是通过函数计算来获取数据的网格编号。假设待映射数据点为 (pos_i,t_i)，两个维度上的分组长度为 L_0 和 L_1，则网格编号 i 的计算式如下：

$$i=(\lceil pos_i/L_0\rceil-1)\cdot C+\lceil t_i/L_1\rceil-1$$

其中，$C=\left\lceil\frac{T}{L_1}\right\rceil$ 表示 t 维度的网格总数，T 表示总的时间长度。

4.4　映射与排序

将二维数据点 $(pos^{\text{mid}},t^{\text{mid}})$ 通过映射函数 M 转换为一维数据。映射函数 M 需满足以下条件：

$$M(x_i\in V)<M(x_j\in V)$$

$$i<j,x_i\in C_i,x_j\in C_j$$

其中，$C_k=[\theta_{j_0}^{(0)},\theta_{j_0+1}^{(0)})\times[\theta_{j_1}^{(1)},\theta_{j_1+1}^{(1)})$ 表示一个二维网格单元。M 的函数表达式如下：

$$M(x)=i+\frac{\mu(H_i)}{\mu(C_i)}$$

其中，$H_i=[\theta_{j_0}^{(0)},x_0)\times[\theta_{j_1}^{(1)},x_1)$，$\mu(\cdot)$ 表示在 $\mathbb{R}^2$ 上的勒贝格度量（Lebesgue Measure）[25]。如此设计可保证 M 单调递增，从而方便分段线性模型的拟合。数据映射完成后按递增顺序排列，存储在磁盘上。

5　分片预测模型 DPSP

5.1　模型构建算法

道格拉斯-普克算法[26]是一种将曲线近似表示为一系列点，并减少点的数量的算法。受此启发，本文提出基于道格拉斯-普克算法的分片预测模型（Douglas-Peuker Algorithm Based Shard Prediction Model，DPSP），以训练高效且误差可控的分片预测模型。算法的基本流程如下：

1）将数据按一定次序排序；

2）将首尾两点连线，形成直线段，求出其余各点到该直线段的距离；

3）选取距离最大者与距离阈值相比较，若大于阈值，则离该直线距离最大的点保留，否则将直线按该点分为两段；

4）对曲线分成的两部分分别重复第 2 步和第 3 步操作，直到无点可舍去。

图 4 给出了道格拉斯-普克算法的一般流程。如图 4(a)所示，首先计算 1，2，3 点到线段的距离，然后选择距离最大的点作为分割点，将原来的直线分为两段，如图 4(b)所示。重复以上操作，如图 4(c)所示，最终获得分段线性函数，如图 4(d)所示。由此可以看出，相比于最优化方法，使用道格拉斯-普克算法拟合可以避免复杂的矩阵运算，且不需要人为确定函数分段数量。此外，还可以有效地保证分段线性函数中的每个数据点的误差小于特定的阈值，从而使得预测误差可控。

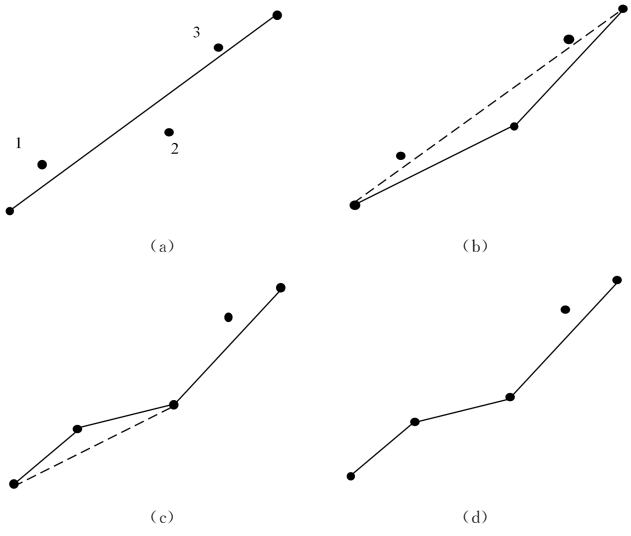

图 4　道格拉斯-普克算法示例

Fig. 4　Illustration of Douglas-Peuker algorithm

5.2　训练方法

接下来介绍具体的模型训练方法。假设第 $i(i=0,1,\cdots)$ 个一维映射值分组 $M_i=[m_1,\cdots,m_n]$ 包含 n 个映射值，分片中的最大数据量为 V，将 M_i 按照分片大小切割，获得 D 个

分片，$D=\lceil\frac{n}{V}\rceil$。选取每个分片的最大映射值作为分割点，组成模型输入序列 $X=[m_{i_1},m_{i_2},\cdots,m_{i_D}]$，模型输出则为对应的分片编号序列 $Y=[i_1,i_2,\cdots,i_D]$。DPSP 模型预测过程的形式化表示为：

$$DPSP(x)=g(x)+D\cdot i$$

其中，$g(\cdot)$ 表示道格拉斯-普克算法所预测出的分片编号。

对每个映射值分组 M_i 训练分段线性模型 F_i。在训练时，记录最大预测误差 E_i。在预测时，对模型预测出的分片号进行向上取整，得出最后的分片编号，因此分片编号为 $S_i=\lceil F_i(m_i)\rceil$，其中 m_i 为待查询数据点的一维映射值。然后根据最大预测误差扩大分片搜索范围至 $[\max(0,S_i-E_i),\min(S_i+E_i,S_{\max})]$，其中 $S_{\max}$ 表示最大分片编号。将上述范围内的数据分片读入内存中进行查找，从而找到相应数据。

5.3 模型参数分析

本节分析道格拉斯-普克算法的阈值参数对于 DPSP 预测准确率的影响。首先介绍相关定义，然后分析阈值与预测准确率之间的关系。预测准确率指所有预测样本中预测准确的样本占比，其计算式为 $P=n/v$，其中 n 表示预测准确的样本数量，v 表示总的样本数量。

定义 6（上部点与下部点） 在一个函数段 f，输入到分段线性模型中的数据点，记为 m_i，其二维坐标为 $(m_i,f(m_i))$。若 $f(m_i)<i$，则称 m_i 为上部点；若 $f(m_i)>i$，则称 m_i 为下部点。

如图 5 所示，在函数段中有 p_1-p_4 4 个数据点。其中，p_1 和 p_4 为上部点，p_2 和 p_3 为下部点。每一个上部点或下部点，都存在查询未命中的风险，原因是采用向上取整计算分片编号，对于一个上部点，其右邻域范围内的映射值的分片预测将出错，而下部点的左邻域范围内的映射值将会预测错误。例如，下部点 p_2 坐标为 $(m_2,2)$，p_3 坐标为 $(m_3,3)$，给定映射值 $m_i\in(m_2,m_3]$，其正确的预测值 $S_i=3$。然而，若 $m_i\in(f(3)^{-1},m_3]$，则预测值 $S_i=\lceil f(m_i)\rceil=4$，造成预测误差。因为当 $m_i>f(3)^{-1}$ 时，$\lceil f(m_i)\rceil>3$。若搜索键落在上述区域中，则会导致预测错误，影响查询准确性。为此引入误差度量的概念，以分析预测准确率的影响因素。

定义 7（误差度量） 指在一个一维实数区间 $[m_i,m_{i+N}]$ 中，预测错误的数据的映射值区间长度。若查询数据落在该区间中，则会预测错误。每个上分界点或下分界点均存在一个误差度量，记为 l。

如图 6 所示，对于函数段 f，有 4 个数据点 p_k，p_i，p_j，p_{k+N}（$k<i<j<k+N$）。其中，上部点 p_i 和下部点 p_j 的坐标分别为 (m_i,i) 和 (m_j,j)，根据定义，p_i 的误差度量 $l_i=f(i)^{-1}-m_i$，p_j 的误差度量 $l_j=m_j-f(j)^{-1}$。

接下来分析参数与预测准确率的关系。首先，根据问题定义可以得出预测准确率 P 的计算式：

$$P=1-\sigma,\sigma\in[0,1] \tag{1}$$

其中，σ 表示预测错误的概率。接下来通过分析参数 σ 来分析 P 的取值范围。为了简化计算，假设搜索键的分布服从均匀分布，则可以定义预测错误率 σ 的计算式为：

$$\sigma=\frac{1}{m_{i+N}-m_i}\sum_{K_1\cup K_2}l_i \tag{2}$$

其中，K_1 表示上部点集合，K_2 表示下部点集合。

由定义 7 可以得出第 i 个数据点 m_i 的误差度量 l_i 的计算式：

$$l_i=\begin{cases}\frac{i-B}{A}-m_i, & i\in K_1\\ m_i-\frac{i-B}{A}, & i\in K_2\end{cases} \tag{3}$$

根据问题的定义可知，映射值 m_i 相对于函数段 f 的预测误差 $L_i=|i-f(m_i)|$，将其展开，可得：

$$L_i=\begin{cases}i-A\cdot m_i-B, & i\in K_1\\ A\cdot m_i+B-i, & i\in K_2\end{cases} \tag{4}$$

将式(2)乘以系数 A，可得：

$$A\cdot l_i=\begin{cases}i-B-A\cdot m_i, & i\in K_1\\ A\cdot m_i-i+B, & i\in K_2\end{cases} \tag{5}$$

观察式(4)和式(5)，可得：

$$l_i=\frac{1}{A}\cdot L_i<\frac{1}{A}\cdot L,i\in K_1\cup K_2 \tag{6}$$

由式(6)可得预测错误率 σ 的取值范围为：

$$0\leqslant\sigma=\frac{1}{A\cdot(m_{i+N}-m_i)}\sum_{K_1\cup K_2}L_i \tag{7}$$

由于 $L_i<L$，因此可得：

$$0\leqslant\sigma<\frac{S}{A\cdot(m_{i+N}-m_i)}L \tag{8}$$

其中，S 表示 $K_1\cup K_2$ 中的元素总数。

进一步，将端点坐标带入 f 的表达式，可将 A 展开为 $N/(m_{i+N}-m_i)$，带入式(8)可得：

$$0\leqslant\sigma<\frac{S}{N}L \tag{9}$$

根据式(9)可得模型预测准确率 P 的取值范围为：

$$1-\frac{S}{N}L<P\leqslant1 \tag{10}$$

从式(10)可以看出，P 与函数段 f 中包含的数据总数 N、上下部点总数 S 以及模型误差阈值 L 均有关系。在此基础上，根据问题背景可知 $S<N$，即 $S/N<1$，故可以进一步推出：

$$1-L<P\leqslant1 \tag{11}$$

从式(11)可以看出，预测准确率 P 有一个明确的下界 $1-L$。其中，阈值参数 L 与查询命中率呈负相关，即 L 越大，命中率 P 越小。这说明误差阈值参数越小，模型的预测准确率越高。

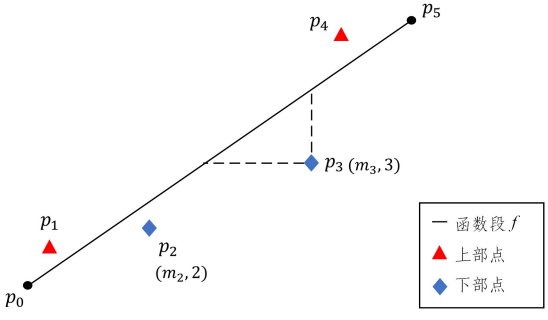

图 5 上部点和下部点示例

Fig. 5 Illustration of upper points and lower points

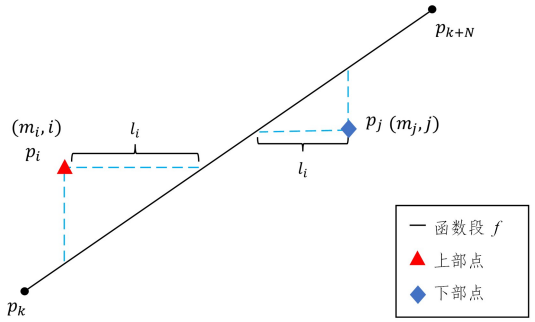

图 6 误差度量示例

Fig. 6 Illustration of error metric

6 本地模型

本地模型(Local Model，LM)的作用是对数据分片进行磁盘页面管理，每个数据分片都对应一个本地模型 LM。

LM 中主要保存两个序列：1) 磁盘页号序列，记为 PA；2)映射值分界点序列，记为 PM。PA 记录了所有在分片 S 中的页面号，PM 记录了每个页面的边界映射值。假设 $I_i=(k_0,\cdots,k_{n-1})$表示落在分片 S_i 中的 n 个数据点，且按照映射值有序排列，Ω 表示磁盘页面大小。如果 $n\leqslant\Omega$，则说明 I_i 可以存储在同一个磁盘页中。只需将页号存储到 $LM_i.PA$ 中，并且将 $LM_i.PM$ 设置为空即可。如果 $n>\Omega$，则需要将 I_i 沿映射值进行分组，使得每组映射值存储在同一个页面上。此种情况需要将页面的页号存储到 $LM_i.PA$ 中，并将每组的边界映射值存储到 $LM_i.PM$ 中。

本地模型 LM 还提供两个搜索函数，即 lbound 函数和 ubound 函数。lbound 函数的功能是搜索映射值的上边界页号，表达式如下：

$$LM.lbound(m)=j$$

其中，$LM.PM[j-1]<m\leqslant LM.PM[j]$。

$ubound$ 函数的功能是搜索映射值的下边界页号，表达式如下：

$$LM.ubound(m)=j$$

其中，$LM.PM[j-1]\leqslant m<LM.PM[j]$。

当 $LM.PM$ 为空时，$LM.lbound(m)$和 $LM.ubound(m)$均返回 0。当 $LM.PM$ 长度为 $j(j>0)$，$m_{\max}=LM.PM[j-1]$，且 $m>m_{\max}$时，$LM.lbound(m)=j-1$；而当 $m\geqslant m_{\max}$时，$LM.ubound(m)=j-1$。

DPSP 模型的参数在训练完成后不再改变，而 $LM_i.PA$ 和 $LM_i.PM$ 的参数会根据分片 S_i 中的数据更新情况进行动态调整。具体来说，若有新数据插入索引，则首先更新边界映射值序列 PM。然后，判断当前本地模型中页面是否满，若满则更新磁盘页号序列 PA；若有数据删除，则首先更新边界映射值序列 PM，然后判断当前本地模型中页面是否小于页面大小，若小于则合并相邻页面。

建立索引的完整算法如算法 1 所示。首先将所有数据映射为一维映射值，得到映射值序列，该过程的时间复杂度为 $O(L_1+L_2)$。然后，按照分片大小对映射值序列进行划分，之后对数据分片构建 DPSP 模型，并生成分片序列 S，该过程的时间复杂度为 $O(|S|\log|S|)$。第 3－20 行遍历分片序列 S，对每个分片构建本地模型，分为两步：获取分界点序列和获取磁盘页号序列，该过程的时间复杂度为 $O(|S|\times V)$，其中 V 表示每个数据分片中超出整数页的数据个数，V 小于磁盘存储数据量上限。最后，返回 DPSP 模型以及本地模型 LM 序列。因此，总的时间复杂度为 $O(L_1+L_2+|S|\log|S|+|S|\times V)$。

算法 1 索引构建算法

输入：轨迹数据集 D，pos 分组数量 L_1，t 分组数量 L_2，页面大小 Ω

输出：分段预测模型 DPSP，本地模型序列 LMs

1. 根据L_1，L_2和映射式，计算所有数据的映射值，将其保存在数组中；
2. 构建 DPSP 模型，生成分片序列 S；
3. for each s ∈ S do
4. 根据页面大小 Ω 划分分片，获取分界点序列 $I_i=(k_0,\cdots,k_{n-1})$；
5. 初始化本地模型 LM_i，将 $LM_i.PA$ 和 $LM_i.PM$ 设置为空；
6. if n≤Ω then
7. 分配新的磁盘页 P，并将I_i存储于 P；
8. 将 P 的页号存储于$LM_i.PA$；
9. end if
10. else
11. $W\leftarrow\left\lceil\frac{n}{\Omega}\right\rceil,\Delta\leftarrow\left\lceil\frac{n}{W}\right\rceil$；
12. for j ← 1 to L_1-1 do
13. 分配新的磁盘页 P；
14. 将$I_i^{(j)}=(k_{(i-1)\times\Delta},\cdots,k_{i\times(\Delta-1)})$存储于 P 中；
15. 将 P 的页号存储于$LM_i.PA$；
16. 将$m_{i\times\Delta}$存储于$LM_i.PM$；
17. end for
18. 将LM_i加入 LMs 中；
19. end else
20. end for
21. return DPSP，LMs.

7 数据更新

本章介绍了 DPLI 的数据更新操作，包括数据的插入操作和删除操作。

7.1 插入操作

给定一个待插入数据 k，插入操作的流程为：首先计算数据 k 的一维映射值m_k，然后将m_k输入到预测模型 DPSP 中，计算对应的分片编号s_k，之后根据编号访问相应的本地模型 LM_i，从而找到目标插入页 P。判断 P 是否已满，若不满，则直接插入；若满，则分配新的页面P'，将 P 中的数据和数据 k 进行排序，然后依次存储到 P 和P'中。具体过程如算法 2 所示。

算法 2 插入算法

输入：待插入数据 k

1. 根据L_1，L_2和映射式，计算数据 k 的映射值m_k；
2. $s_k\leftarrow$DPSP(m_k)；/ * 计算分片编号 * /
3. if $LM_i.PA$ 为空 then
4. 分配新的页面 P，并将 k 存储于 P 中；
5. 将 P 的地址加入到$LM_i.PA$中；
6. else
7. u←LM_i.ubound(m_k)；

```
8.    获取页号LM_i.PA[u]所对应的页面 P;
9.    if P 未满 then
10.     将 k 插入到 P 中;
11.   else
12.     将 P 中所有数据和 k 组成序列 L;
13.     将序列中的数据按照映射值大小排序;
14.     将 P 中的数据清空,并将 L 中前一半的数据存储在 P 中;
15.     分配新的磁盘页 P',将 L 中剩余数据存储在 P'中;
16.     将 P'的页号加入 LM_i.PA 中;
17.     计算 P 中边界数据的映射值 m';
18.     将 m'插入到 LM_i.PM 中。
```

7.2 删除操作

删除操作和插入操作类似，给定一个待删除数据 k，删除操作的流程为：首先计算数据的一维映射值m_k，然后计算对应的分片编号$s_k=DPSP(m_k)$，之后根据编号s_k访问相应的本地模型序列LM_i。遍历序列$LM_i.PA$，查找 k 所在的页面 P，然后删除对应的数据。若删除后页面 P 为空，则释放 P。特别地，若经过多次删除后，两个相邻页面之间的数据总量少于一定阈值，则进行页面合并操作，将两个页面合并为一个页面，并修改本地模型序列。具体过程如算法 3 所示。

算法 3 删除算法

```
输入:待删除数据 k
1. 根据L_1,L_2和映射式,计算数据 k 的映射值m_k;
2. s_k←DPSP(m_k);/* 计算分片编号 */
3. pid←-1,T←LM_i.PA.size();
4. for j←l to u do
5.    获取页号LM_i.PA[j]所对应的页面 P;
6.    if P 包含 k then
7.       删除 k,pid←j;
8.       if P 为空 then
9.          释放 P,清空LM_i.PA[j];
10.         if LM_i.PM.size()>0 then
11.            j'← min(LM_i.PM.size()-1,max(0,j-1));
12.            删除LM_i.PM 中的第 j'个元素;
13.         return;
14. l←-1;
15. if pid≥0 且 T>1 then
16.      if pid≠0 then
17.         if LM_i.PA[pid-1]+LM_i.PA[pid]≤Ω:
18.         l←pid-1;
19.      if pid≠T-1 then
20.         if LM_i.PA[pid]+LM_i.PA[pid+1]≤Ω:
21.            l←pid;
22. if l≥0 then
23.      获取 LM_i.PA[l]和 LM_i.PA[l+1]所对应的页面 P_0和 P_1;
24.      将 P_1中的所有元素加入到 P_0中;
25.      释放 P_1并删除 LM_i.PA[l+1]中的元素。
```

8 路径范围查询

路径范围查询 $Q=(PR,TR)$，其中 PR 表示待查询的连续路段序列，$PR=(s_1,s_2,s_3,\cdots,s_n)$，$s_1$和$s_n$表示查询的首尾路段的一部分，而$s_i(i=2,3,\cdots,n-1)$则表示中间路段。结合 2.2 节，将 PR 表示为 $PR=[pos_1,pos_2]$，其中 pos_1 和 pos_2 表示查询的首尾路段的范围。TR 则表示时间戳范围，形式上，$TR=[t_1,t_2]$，其中t_1为起始时间戳，t_2为终止时间戳。综上，查询 Q 输入一个时空范围 (PR,TR)，返回在该范围内的所有子轨迹段编号。

具体查询流程为，首先将 Q 的空间范围 $PR=[pos_1,pos_2]$ 扩大至 $\left[\max\left(\lfloor pos_1\rfloor,pos_1-\frac{h}{2\cdot l}\right),\min\left(pos_2+\frac{h}{2\cdot l},\lceil pos_2\rceil\right)\right]$，其中 h 表示该位置所在路段中最长的轨迹单元长度，l 表示所在路段的长度。同理，将 TR 扩大范围为 $\left[t_1-\frac{w}{2},t_2+\frac{w}{2}\right]$，其中 w 表示对应的位置路段中轨迹单元的最大时间戳范围。然后找出与该查询框相交的网格，按照网格对其进行分割，分解为多个小型的范围查询 q，每个小型范围查询 q 只与一个网格相交，$Q=\bigcup_{j=0}^{N}q_j$，其中$q_j=[pos_{j_0},t_{j_0})\times[pos_{j_1},t_{j_1})\subseteq C_{i_j}$，$N$ 表示小范围查询的数量。结合 2.2 节，当分组数量 L_0，L_1 以及 Q 很大时，N 可能也会很大。这种情况下，将连续的查询框进行合并。

接下来，计算所有小型查询框$q_j=[l,u](j=0,1,2,\cdots,N)$的上下边界的一维映射值，分别记为 $m_l^{(j)}$，$m_u^{(j)}$。然后将其带入分段线性函数中，获取其分片编号$k_l^{(j)}$，$k_u^{(j)}$，按照最大误差 E 扩大查询范围，最后调用相应的本地模型，查找对应的磁盘块，读取块中的数据。

路径范围查询算法如算法 4 所示。第 1－3 行将查询框 Q 扩大，并按照相交的网格对其进行分解，合并相邻的网格，获得次级查询框序列 qr。由于分解和合并可以通过递归方式实现，因此该过程的时间复杂度为 $O(L_1\log L_2)$。第 4－17 行对每个次级查询框进行查询，并更新本地模型 LM 的磁盘页号序列，该过程的时间复杂度为 $O(N\times V)$，其中 V 表示 $LM.PA$ 的平均长度。第 18－20 行搜索磁盘块，将数据加入查询结果中，该过程的时间复杂度为 $O(P)$，其中 P 表示查询到的所有磁盘页个数。因此，总的时间复杂度为 $O(L_1\log L_2+N\times V+P)$。

算法 4 路径范围查询

```
输入:查询框 Q
输出:位于查询范围内的查询键 R
1. 将 Q 按照两个维度分别扩大,得到Q';
2. 将 Q'按照网格进行分解,获得小型查询框序列 qr=⋃_{j=0}^{N} q_j;
3. 初始化集合 PageAddrs;
4. for j←0 to N do
5.    m_l^(j) ←M(l_j0,l_j1),m_u^(j) ←M(u_j0,u_j1);
6.    k_l^(j) ←DPSP(m_l^(j)),k_u^(j) ←DPSP(m_u^(j));
7.    l←LM_{k_l^(j)}(m_l^(j));
8.    for i←l to LM_{k_l^(j)}.PA.size()-1 do
9.       PageAddrs.add(LM_{k_l^(j)}.PA[i]);/* 更新磁盘页号序列 */
10.   end for
11.   for s ← k_l^(j)+1 to k_u^(j)-1 do
12.      PageAddrs.addall(LM_s.PA);
13.   end for
14.   u ← LM_{k_l^(j)}.ubound(m_u^(j));
```

15. for t ← 0 to u do
16. PageAddrs.add($LM_{k_u^{(j)}}$.PA[t]);
17. end for
18. for each P∈PageAddrs do
19. 将所有落入查询框Q′的数据加入到R;
20. end for
21. return R.

9 实验及分析

采用真实数据集和模拟数据集对DPLI的构建效率、查询效率、查询精确度以及召回率进行了全面的实验。所有实验均在一台内存为16 GB、处理器为3.4 GHz的Intel(R) Core(TM) i7-3770、操作系统为Ubuntu 16.04的机器上进行[1)]。

9.1 实验数据集

本文采用两个轨迹数据集和两个路网数据集来测试本文方法的性能。

1)模拟轨迹数据集sim：由多模态交通仿真软件SUMO模拟得出。相邻轨迹点采样间隔设置为1 s，模拟时间设置为604 800 s。每43 200 s生成一个流量低峰和一个流量高峰，平均高低峰车流量比例为3∶1。最终获得轨迹数据总数100 000条，GPS采样点总数为174 830 241，磁盘占用量为18.3 GB。

2)成都出租车轨迹数据集chd：该数据集包含中国成都市从2014-08-03到2014-08-30总计28天的出租车轨迹数据，总共包含1 099 765条轨迹数据，487 195 895个GPS采样点。相邻轨迹点采样率为3 s，磁盘占用量为27.8 GB。

3)北京市路网数据：从OSM中下载的北京市矩形区域内的路网数据，经度范围为[116.05，116.633]，纬度范围为[39.817，40.083]，共有36303条边，83 435个顶点，磁盘占用量为16.3 MB。

4)成都市路网数据：从OSM中下载的成都市矩形区域内路网数据，经度范围为[103.653，104.512]，纬度范围为[30.324，30.958]，共有68 896条边，178 177个顶点，磁盘占用量为11 MB。

对于每个数据集，根据时间维度将其划分为10个周期，每个周期作为一个实验数据子集。在查询方面，对每个数据子集都生成100个范围查询以测试模型的性能。为了能全面地测试索引在不同选择下的查询表现，将生成的范围查询分为4类，查询面积分别占总数据面积的1/100 000，1/25 000，1/12 500，1/62 500，其中每一类包含25个查询。

9.2 衡量标准

本文主要研究基于磁盘存储的索引结构，因此主要比较不同方法的磁盘I/O效率、索引空间大小、构建时间以及索引的准确率和召回率。

9.3 对比方法

本文比较以下索引结构，包含传统的基于树结构的索引以及学习型索引。

1) R-tree[5]是一种传统的树形索引结构，支持2维以上的多维数据索引。

2)R*-tree[6]不同于R-tree，R*-tree将数据区域递归划分，并将不同数据区域进行整合，相比R-tree有更高的查询表现。

3)T-PARINET[2]是一种检索路网内轨迹的分区索引。其底层是PARINET索引，它基于图划分和一组B+树的组合来索引历史轨迹。

4)LISA[11]为空间数据的学习索引结构，由映射模型、分片预测函数和本地模型组成。

5)RSMI[20]为递归空间模型索引，其建立二维空间点的排序，然后使用递归模型索引来处理空间数据索引问题。

6)SPRIG++[23]利用空间插值函数来支持空间数据的范围和kNN查询。其对空间数据进行采样以构建自适应网格，并使用样本数据作为输入来拟合空间插值函数。给定空间搜索键，利用拟合的空间插值函数预测一个近似位置，然后进行局部二值搜索寻找样本。

其中，R-tree，R*-tree和T-PARINET为传统索引；LISA，RSMI和SPRIG++为最先进的多维学习型索引。在对比方法的参数设置方面，根据现有的工作来配置对比方法的最优参数，其中LISA的参数设置分别为$T=240$，$U=1\,000$，$\delta=100$，分别表示维度网格数、线性模型个数和模型分段数。RMI的参数设置为$N=50\,000$，表示模型预测的最大点数。SPRIG++的参数设置为$m=n=200$，分别表示每个维度的分组数量。DPLI参数设置为分组长度$R_0=R_1=100$，每个分组的分片数量$w=500$，默认误差阈值$L=0.3$。为了保证比较的公平性，所有的轨迹数据都按照文献[24]中的方法匹配到路网上；每条轨迹按照2.2节的方法转换为pos-t数据；所有的轨迹数据以及索引数据都是以文件形式存储在磁盘上的，磁盘页面大小设置为4 kB。索引结构和查询数据各设置了1 MB大小的主存缓冲区。

9.4 实验结果

9.4.1 阈值对预测准确率的影响验证

首先验证模型阈值参数对预测准确率的影响。在4.3节中，通过参数分析，推导出模型的阈值参数对于查询召回率呈现负相关，并进一步计算出具体的召回率下界(详见4.3节中的式(11))。本节主要验证所提的结论的有效性。具体实验方法为，选择不同的模型误差阈值来训练模型，然后测试其在不同参数阈值条件下的预测准确率，并与4.3节中的式(11)进行对比，以验证分析是否准确。

从图7中可以看出，在chd数据集和sim数据集上，随着阈值L的增加，查询的召回率出现了下降趋势，并且其召回率的数值均超过了估计值，数量关系符合式(11)预期，说明4.3节的分析能够准确反映实际查询时的一般情况。

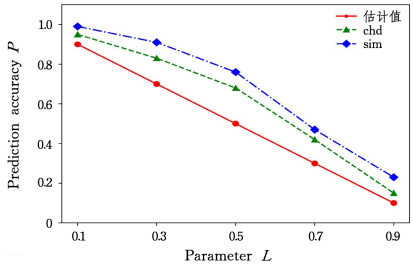

图7 参数对召回率影响的对比

Fig. 7 Comparison of the impact of parameter on recall rate

1) https://github.com/hjyresearch228/mdm

9.4.2 索引大小对比

比较不同方法在 sim 和 chd 两个数据集上的空间占用大小，结果如图 8 所示。可以看出，RSMI 的空间占用最高，其原因是，RSMI 利用神经网络来拟合数据分布，模型参数占用了大部分的索引空间。相比之下，DPLI 的空间占用最小，原因是 DPLI 使用了基于网格的多维索引框架，并且采用函数映射的方式来定位网格，无须保存网格边界参数，因此具有较低的空间占用。

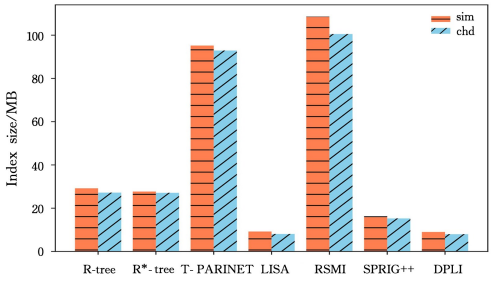

图 8 索引大小对比

Fig. 8 Comparison of index size

9.4.3 索引构建时间比较

图 9 展示了不同索引的构建时间。结果表明，在 sim 和 chd 数据集上，DPLI 的构建时间都远短于传统索引 R 树、R* 树和 T-PARINET 以及学习型索引 LISA，RSMI 和 SPRIG++。这是因为 DPLI 只需计算每个数据点到直线段的距离，通过递归来完成线性分段函数的构建，不需要执行树的插入算法，也不需要进行复杂的矩阵运算，所以构建时间较其他方法短很多。而传统的树型索引结构需要对树进行插入操作，该操作在数据量较大时，消耗时间较长。RSMI 采用神经网络模型，构建耗时最长。SPRIG++需先进行分区，然后进行拟合，其中分区过程耗时较长。LISA 则需要对分段线性函数进行最小二乘优化，涉及多次矩阵运算，较为耗时。

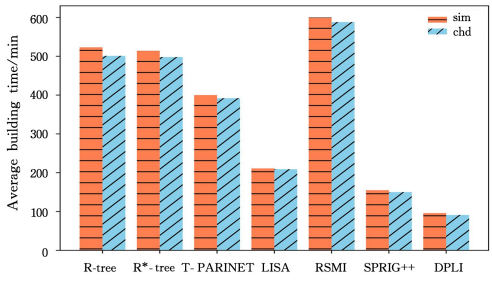

图 9 索引构建时间对比

Fig. 9 Comparison of index building time

9.4.4 查询时间效率对比

在查询时间效率方面，本文索引均是存储在磁盘上的，磁盘的访问时间在总体查询响应时间中占比最大，往往直接决定了查询时间效率的高低，因此通过磁盘的平均 I/O 次数来比较查询的时间效率，结果如图 10 所示。可以看出，RSMI 的磁盘 I/O 次数最大，原因是其使用分区的方式对数据分布进行预测，分区不具有连续性，因此降低了访问效率。T-PARINET 的 I/O 效率也较低，其将路段进行了图分区，导致在面对连续路径查询时，相邻路段有可能处于不同的图分区中，从而导致较低的访问效率。而 DPLI 的查询效率优于 SPRIG++，对比 LISA 稍低，其原因主要在于，对于 SPRIG++来说，DPLI 使用了数据分片的数据结构，使得数据网格划分更加精细；而对比 LISA，DPLI 需要在最后扩大查询的范围，以确保查询正确，因此效率稍低。

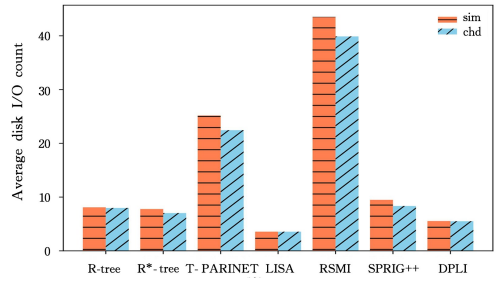

图 10 磁盘平均 I/O 对比

Fig. 10 Comparison of average disk I/O count

9.4.5 查询精确度对比

精确度指，在范围查询时搜索到的数据结果中，符合条件的数据个数占总数据个数的百分比。精确度是用于评价索引查询的准确率的重要指标之一，精确度越高则说明查询的准确率越高。由于传统索引在范围查询时是精确查找，因此不进行比较。这里仅比较 DPLI，LISA，RSMI 和 SPRIG++这几种学习索引方法的查询精确度。由表 1 可知，上述方法均达到了 100%的查询精确度。

表 1 查询精确度对比

Table 1 Comparison of query precision rate

(%)

方法	chd	sim
LISA	100	100
RSMI	100	100
SPRIG++	100	100
DPLI	**100**	**100**

9.4.6 查询召回率对比

召回率指，在范围查询时搜索到的符合条件的数据量占范围实际覆盖的数据量的百分比。召回率是用于评价索引查询的准确率的重要指标之一，召回率越高则说明查询的准确率越高。由于传统索引在范围查询时是精确查找，因此不进行比较。这里仅比较 DPLI，LISA，RSMI 和 SPRIG++这几种学习索引方法的查询召回率。由表 2 可知，DPLI 实现了 100%的查询召回率，说明其在数据查询方面具有很高的精确度；SPRIG++同样实现了 100%的查询召回率，这是因为其在查询后添加了筛选操作，以确保查询的数据正确；而 LISA 和 RSMI 均未达到 100%的召回率，原因是其机器学习模型存在预测误差。

表 2 查询召回率对比

Table 2 Comparison of query recall rate

(%)

方法	chd	sim
LISA	77.2	92.1
RSMI	92.8	97.3
SPRIG++	100	100
DPLI	**100**	**100**

结束语 本文提出了一个基于道格拉斯-普克算法的学习索引结构 DPLI，能够实现海量轨迹数据的高效查询。具体

来说，该方法包含数据预处理、分段预测模型以及本地模型3个模块，能够实现路径范围查询，且模型训练效率高，预测误差可控。此外，分析了阈值参数与预测准确率之间的量化关系。实验结果表明，所提方法数据定位快速，查找效率高；函数拟合无需复杂的运算，预测误差有理论上的保证，具有较高的效率和准确性。在未来的工作中，将进一步研究如何与其他方法整合；同时考虑将本文方法与分布式数据库系统结合，更进一步地提升查询效率。

参 考 文 献

[1] ZHANG S M，CAI P，LI C P，et al. High-dimensional Learned Index Based on Space Division and Dimension Reduction [J]. Journal of Software，2023，34(5)：2413-2426.

[2] SANDU POPA I，ZEITOUNI K，ORIA V，et al. Indexing in-network trajectory flows [J]. The VLDB Journal，2011(20)：643-669.

[3] MORRIS B T，TRIVEDI M M. Trajectory Learning for Activity Understanding：Unsupervised，Multilevel，and Long-Term Adaptive Approach [J]. IEEE Transactions on Software Engineering，2011，33(11)：2287-2301.

[4] TAO Y，PAPADIAS D. MV3R-tree：A spatio-temporal access method for timestamp and interval queries[C]// Proceedings of the 27th International Conference on Very Large Data Bases. San Francisco，CA：Morgan Kaufmann Publishers Inc.，2001：431-440.

[5] GUTTMAN A. R-trees：A dynamic index structure for sparial searching [J]. ACM Sigmod Record，1984，14(2)：47-57.

[6] BECKMANN N，KRIEGEL H P，SCHNEIDER R，et al. The R*-tree：an efficient and robust access method for points and rectangles[C]// International Conference on Management of Data. New York：ACM，1990：322-331.

[7] SINGH M，ZHU Q，JAGADISH H V. SWST：A Disk Based Index for Sliding Window Spatio-Temporal Data[C]// 2012 IEEE 28th International Conference on Data Engineering. IEEE，2012：342-353.

[8] ROBINSON J T. The K-D-B-tree：a search structure for large multidimensional dynamic indexes[C]// Proceedings of the 1981 ACM SIGMOD International Conference on Management of Data. New York：ACM，1981：10-18.

[9] PFOSER D，JENSEN C S. Indexing of network constrained moving objects[C]// CIKM03：12th International Conference on Information and Knowledge Management. New York：ACM，2003：25-32.

[10] ZHANG R，QI J，STRADLING M，et al. Towards a Painless Index for Spatial Objects [J]. ACM Transactions on Database Systems，2014，39(3)：1-42.

[11] LI P，LU H，ZHENG Q，et al. LISA：A Learned Index Structure for Spatial Data[C]// International Conference on Management of Data. New York：ACM，2020：2119-2133.

[12] LI R，RUAN S，BAO J，et al. Querying Massive Trajectories by Path on the Cloud[C]// the 25th ACM SIGSPATIAL International Conference. New York：ACM，2017：1-4.

[13] KRASKA T，BEUTEL A，CHI E H，et al. The Case for Learned Index Structures[C]// Proceedings of the 2018 International Conference on Management of Data. New York：ACM，2018：489-504.

[14] DING J，MINHAS U F，YU J，et al. ALEX：An Updatable Adaptive Learned Index[C]// International Conference on Management of Data. New York：ACM，2020：969-984.

[15] GALAKAATOS A，MARKOVITCH M，BINNIG C，et al. FI-Ting-Tree：A Data-aware Index Structure[C]// Proceedings of the 2019 International Conference on Management of Data. New York：ACM，2019：1189-1106.

[16] WANG Y T，WANG Y S，YUAN Y. Survey of Learned Index [J]. Computer Science，2023，50(1)：1-8.

[17] WU F，HAN J Y，LIU Y，et al. Trajectory Index Based on Piecewise Linear Regression Tree[J]. Journal of Chinese Computer Systems，2022，43(6)：1245-1253.

[18] ZHANG Z，JIN P Q，XIE X K. Learning index：Current situation and research prospects [J]. Journal of Software，2021，32(4)：1129-1150.

[19] KRASKA T，ALIZADEH M. SageDB：An Instance-Optimized Data Analytics System [J]. Proceedings of the VLDB Endowment，2019，15(13)：4062-4078.

[20] WANG H，FU X，XU J，et al. Learned Index for Spatial Queries [C]// 2019 20th IEEE International Conference on Mobile Data Management(MDM). IEEE，2019：569-574.

[21] QI J，LIU G，JENSEN C S，et al. Effectively learning spatial indices [J]. Proceedings of the VLDB Endowment，2020，13(12)：2341-2354.

[22] DAVITKOVA A，MILCHEVSKI E，MICHEL S. The ML-Index：A Multidimensional，Learned Index for Point，Range，and Nearest-Neighbor Queries [C] // International Conference on Extending Database Technology. Copenhagen：Open Proceedings. org，2020：407-410.

[23] ZHANG S，RAY S，LU R，et al. Efficient Learned Spatial Index With Interpolation Function Based Learned Model [J]. IEEE Transactions on Big Data，2023，9(2)：733-745.

[24] NEWSON P，KRUMM J. Hidden Markov map matching through noise and sparseness [C] // Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. New York：ACM，2009：336-343.

[25] CHRISTOPHER H. Introduction to Real Analysis [M]. New York：Wiley，2019：33-86.

[26] WU S T，MARQUEZ M R G. A non-self-intersection Douglas-Peucker algorithm[C]// Computer Graphics and Image Processing. IEEE，2003：60-66.

MIAO Zhuqing，born in 1997，postgraduate，is a student member of CCF(No. O8027G). His main research interests include machine learning and spatio-temporal database.

HAN Jingyu，born in 1976，Ph. D，professor，is a member of CCF(No. E20-0012101M). His main research interests include biomedical data processing，machine learning and spatio-temporal database.

（责任编辑：喻藜）