

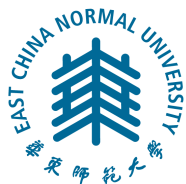
2018 届博士学位论文

分类号: _____

学校代码: 10269

密 级: _____

学 号: 52141500013



華東師範大學

East China Normal University

博 士 学 位 论 文

DOCTORAL DISSERTATION

论文题目:

面向轨迹大数据的管理及查询研究

院 系: 数据科学与工程学院

专 业 名 称: 软件工程

研 究 方 向: 基于位置的服务

指 导 教 师: 周傲英 教授

学位申请人: 章志刚

2018 年 03 月

Dissertation for doctor degree in 2018

University Code: 10269

Student ID: 52141500013

EAST CHINA NORMAL UNIVERSITY

Research on Managing and Querying upon Big Trajectory Data

Department:	School of Data Science and Engineering
	Software Engineering
Major:	Location Based Service
Research direction:	Prof. Aoying Zhou
Supervisor:	Zhigang Zhang
Candidate:	

2017.03

华东师范大学学位论文原创性声明

郑重声明：本人呈交的学位论文《基于矩阵分解的个性化推荐系统》，是在华东师范大学攻读硕士/博士（请勾选）学位期间，在导师的指导下进行的研究工作及取得的研究成果。除文中已经注明引用的内容外，本论文不包含其他个人已经发表或撰写过的研究成果。对本文的研究做出重要贡献的个人和集体，均已在文中作了明确说明并表示谢意。

作者签名：_____

日期： 年 月 日

华东师范大学学位论文著作权使用声明

《基于矩阵分解的个性化推荐系统》系本人在华东师范大学攻读学位期间在导师指导下完成的硕士/博士（请勾选）学位论文，本论文的研究成果归华东师范大学所有。本人同意华东师范大学根据相关规定保留和使用此学位论文，并向主管部门和相关机构如国家图书馆、中信所和“知网”送交学位论文的印刷版和电子版；允许学位论文进入华东师范大学图书馆及数据库被查阅、借阅；同意学校将学位论文加入全国博士、硕士学位论文共建单位数据库进行检索，将学位论文的标题和摘要汇编出版，采用影印、缩印或者其它方式合理复制学位论文。

本学位论文属于（请勾选）

- () 1. 经华东师范大学相关部门审查核定的“内部”或“涉密”学位论文*，于年月日解密，解密后适用上述授权。
- () 2. 不保密，适用上述授权。

导师签名：_____

本人签名：_____

年 月 日

* “涉密”学位论文应是已经华东师范大学学位评定委员会办公室或保密委员会审定过的学位论文（需附获批的《华东师范大学研究生申请学位论文“涉密”审批表》方为有效），未经上述部门审定的学位论文均为公开学位论文。此声明栏不填写的，默认为公开学位论文，均适用上述授权）。

章志刚 博士学位论文答辩委员会成员名单

姓名	职称	单位	备注
***	教授	上海大学	主席
***	教授	复旦大学	
***	研究员	上海交通大学	
***	特聘教授	苏州大学	
***	研究员	华东师范大学	

摘 要

近年来,随着智能手机等移动定位设备的大量使用,人们在生产生活中获得了飞速增长的轨迹大数据。该类大数据包含了车、人、动物甚至商品的移动行为。由于数据采集来源广、数据量大、数据增长快等原因,轨迹大数据往往以分布式形式存储。海量的轨迹数据不仅刻画了移动对象个体和群体的时空动态性,还蕴含着移动对象的行为信息,对交通导航、城市规划、物流调度等应用具有重要的价值。为充分挖掘轨迹数据的价值,学术界和工业界对轨迹数据管理和分析开展了大量研究工作,包括轨迹数据索引、轨迹聚类、轨迹分类、轨迹异常检测和行为预测等问题。

本文首先针对海量轨迹数据,研究了如何实现构建基于计算机集群的轨迹管理系统。现有的轨迹管理系统面临着 I/O 开销过高以致无法提供低延时查询服务的问题。为此,本文设计了基于分布式内存的轨迹管理系统 **TrajSpark**。系统提出了基于列存储的轨迹表达方式,大大提高了内存的利用率。设计了两层索引结构,以提高查找效率。此外, **TrajSpark** 引入了时间衰减模型以监控数据分布的变化并自动更新数据划分策略。该模型能有效降低新数据到达时的划分开销,提高了系统的可扩展性。

进一步地,本文研究了分布式 k 近邻查询。尽管现有方法能提供查询结果,但面临着通信开销大的问题。此外,目前存在着许多轨迹距离度量方法,而现有处理方法往往只适用于单一距离函数,缺乏通用性。为处理以上挑战,本文提出使用概要数据进行距离范围估计并剪枝的思想,以达到降低通信和计算开销的目的。在该思想中,针对不同距离函数,分别设计了基于距离上、下界的剪枝策略和基于下界的剪枝查询策略。前一种策略适用于能根据概要数据同时计算出上、下界的距离函数,而后一种适用于根据概要数据仅能计算出下界的距离函数。最后,验证了具体的欧氏距离和动态时间弯曲距离在两种策略下的实现。本文主要贡献包括如下三方面:

1. 设计了基于分布式内存的轨迹管理系统 **TrajSpark**, 以提供低延时的轨迹查询。现有基于分布式磁盘的管理系统无法提供低延时的查询服务。为此,本文研究并构建了基于分布式内存的轨迹管理系统 **TrajSpark**。**TrajSpark** 提出了新的基于内存的轨迹表达结构 **IndexTRDD**。该结构引入了高效的轨迹表示方法和两层索引策略。系统针对典型轨迹查询,设计了高效的查询算法。此外,系统引入了时间衰减模型以减少数据增加后所造成的重新划分开销。海量轨

迹数据集下的实验结果表明, TrajSpark 相比已有系统具有较好的优越性。

2. **设计了使用概要数据计算距离上、下界的逐步剪枝策略, 实现了欧氏距离在该策略下的查询算法。** 轨迹压缩是降低通信开销的有效方法, 而使用计算复杂度更低的距离上、下界进行剪枝是降低查询时间的有效途径。本文融合这两种方法, 设计了使用轨迹压缩技术以得到多粒度概要数据, 并使用该数据不断计算距离上、下界以逐步剪枝的查询策略 FTB (Framework with Two Bounds)。为验证该策略的有效性, 本文研究了使用该策略进行基于欧氏距离的 k 近邻查询。首先, 本文使用哈尔小波系数作为多粒度概要数据。接着, 设计了基于部分哈尔小波系数的欧氏距离上界和下界, 并证明了随着数据粒度的增加所得到的距离上、下界会越来越紧。基于以上结论, 设计了欧氏距离和 FTB 框架相结合的 ED-FTB 算法。最后, 理论分析和实验结果相结合, 展示了 ED-FTB 算法相比于基准算法的优越性。
3. **设计了使用概要数据计算距离下界的逐步剪枝策略, 实现了动态时间弯曲距离在该策略下的查询算法。** 针对根据概要数据仅能设计出距离下界的情况, 设计了仅使用下界进行逐步剪枝的查询策略 FLB (Framework with Lower Bound)。FLB 通过计算一个不断逼近全局第 k 小距离的阈值来剪枝。为验证该策略的有效性, 本文研究了使用 FLB 策略实现基于动态时间弯曲距离的 k 近邻查询。首先, 设计了多粒度包围信封以作为概要数据。接着, 提出了基于包围信封的动态时间弯曲距离下界, 并证明了随着包围信封粒度的增加所得到的距离下界会越来越紧。基于以上分析, 设计了动态时间弯曲距离和 FLB 框架相结合的 DTW-FLB 算法。该算法引入了在计算下界过程中剪枝的策略以提高执行效率。最后, 我们在真实轨迹数据集上验证了算法的有效性和可扩展性。

关键词: 轨迹, 数据管理, 分布式 k 近邻查询, 距离上/下界, 通信开销, 时间开销.

ABSTRACT

Recently, big trajectory data is generated in an explosive way with the popularity of smartphones and other location-acquisition devices. These devices, monitoring the motion of vehicles, people, animals and goods, are producing massive distributed trajectories rapidly. These data not only reflect the spatio-temporal mobility of individuals and groups, but also contain the behavior information of moving objects. They are invaluable for applications such as route planning, urban planning and logistics scheduling. To make full use of such data, tremendous efforts have been made to support effective trajectory data management analysis, including trajectory indexing, trajectory clustering, trajectory classification, anomaly detection and behavior prediction.

In this paper, we firstly research the problem of how to manage the big trajectory efficiently on a computer cluster. Existing trajectory management systems fail to provide scalable and low latency query services over such data due to the high I/O cost. In view of that, we present TrajSpark, a distributed in-memory system to consistently offer efficient management of trajectory data. TrajSpark introduces a new abstraction called IndexTRDD to manage trajectories as a set of segments, and exploits a global and local indexing mechanism to accelerate trajectory queries. Furthermore, to alleviate the essential partitioning overhead as the increase of data volume, it adopts the time-decay model to monitor the change of data distribution and updates the data-partition structure adaptively. This model avoids repartitioning existing data when new batch of data arrives.

Next, we focus on the k nearest neighbor query over the distributed trajectory data. Although this query can be solved by directly sending the query reference to all remote sites, in which the pairwise distances are computed precisely. However, the overall communication and computation cost is huge. Besides, there are a number of trajectory distance measures and existing work only focus on one of them, thus these work are short of generality. To address above challenges, we devise some communication-saving ways

to estimate pairwise distance by using sketch data, which allow filtering some trajectories in advance without precise computation. Besides, we devise two general frameworks to deal with different distance measures. The first one prunes candidates by using both distance lower and lower bounds, while the second one only uses distance lower bound. So, the former one suites for cases where both distance lower and upper bounds can be computed by using sketch data, and the latter one suites for cases that only lower bound can be computed. Finally, we embed Euclidean distance into the first framework and embed Dynamic Time Warping distance into the second one, to evaluate the practicality of these frameworks. The research questions and technical contributions in this thesis can be summarized as follows:

1. **TrajSpark system is proposed to support low-latency query over big trajectory data.** Existing trajectory systems are built on top of disk-based systems and suffer the problem of I/O overhead. So, we design the TrajSpark by utilizing distributed memory. TrajSpark builds IndexTRDD, an RDD of trajectory segments, to support efficient data storage and management by incorporating a global and local indexing strategy. Besides, TrajSpark monitors the change of data distribution by importing a time decay model which alleviates the repartitioning overhead occurred in existing distributed systems and gets a good partition result at the same time. Extensive experiment results demonstrate the superiority of TrajSpark over other Spark-based systems.
2. **Pruning strategy FTB is proposed by using distance lower and upper bounds to solve the knn query, and algorithm ED-FTB is designed to embed Euclidean distance onto FTB.** Trajectory compression is a practical way to reduce the communication cost. Using distance bound(s), which are computational cheaper than original distance, to prune candidates is an efficient way to improve query efficiency. In this paper, we combine these two methods and design the pruning strategy FTB by computing lower and upper bound from sketch data which is computed from proper

compression method. To evaluate the efficiency of FTB, we implement Euclidean distance based knn query on top of FTB by proposing algorithm ED-FTB. ED-FTB adopts Haar wavelet coefficients as multi-resolution sketch data. Both new lower and upper bounds are designed for ED-FTB, and we prove that these bounds get tighter when finer-grained sketch data is used. Theoretical analysis and extensive experimental results show that ED-FTB outperforms the state-of-the-art algorithm.

3. **Pruning strategy FLB is proposed by using distance lower bound to solve the knn query, and algorithm DTW-FTB is designed to embed DTW distance onto FLB.** For the case when upper bound cannot be computed from sketch data, we propose the pruning strategy FLB by using only distance lower bound. FLB computes a global pruning threshold to approximate the k -th smallest distance. To evaluate the efficiency of FLB, we implement DTW distance based knn query on top of FLB by proposing algorithm DTW-FLB. In DTW-FLB, we adopt bounding envelopes as multi-resolution sketch data and design a lower bound for DTW distance by using such data. The lower bound gets tighter when finer-grained bounding envelope is used. Early-abandoning pruning policy is adopted in DTW-FLB to improve the query efficiency. Extensive experimental results show that DTW-FLB algorithm is efficient and scalable.

Keywords: *Trajectory; Data Management; knn query; Distance Bounds; Communication Cost; Time Efficiency.*

目录

第一章 绪论	1
1.1 研究背景	1
1.2 研究内容与挑战	5
1.3 主要贡献	7
1.4 章节安排	9
第二章 研究现状	12
2.1 数据模型及问题定义	12
2.1.1 轨迹数据	12
2.1.2 分布式 k 近邻轨迹查询	13
2.2 轨迹数据管理	14
2.2.1 轨迹索引	14
2.2.2 集中式轨迹数据管理	14
2.2.3 基于集群的分布式轨迹数据管理	15
2.3 轨迹压缩	17
2.3.1 传统离线时间序列压缩算法	17
2.3.2 传统在线时间序列压缩算法	19
2.3.3 基于路网结构的轨迹压缩算法	20
2.3.4 语义压缩算法	22
2.4 轨迹距离度量	22
2.4.1 传统时间序列距离	23
2.4.2 时空轨迹距离	24
2.4.3 语义轨迹距离	25
2.5 k 近邻轨迹查询	26
2.5.1 集中式环境下查询	26

2.5.2	分布式环境下查询	30
第三章	基于集群的分布式轨迹管理	34
3.1	引言	34
3.1.1	背景知识	34
3.1.2	基于 Spark 的时空管理系统	35
3.2	TrajSpark 系统设计及实现	36
3.2.1	系统架构	37
3.2.2	数据表示层	38
3.2.3	辅助数据层	41
3.2.4	查询处理层	43
3.3	实验分析	47
3.3.1	实验设置	47
3.3.2	实验结果及分析	48
3.4	本章小结	54
第四章	基于上、下界剪枝的分布式 k 近邻查询	56
4.1	引言	56
4.1.1	研究思路	56
4.1.2	接口函数	57
4.2	基于上、下界的逐步剪枝策略	58
4.2.1	策略设计原理	58
4.2.2	FTB 策略实现方案	59
4.3	欧氏距离的概要数据及上、下界计算	62
4.3.1	基于欧氏距离的轨迹相似度度量	62
4.3.2	基于哈尔小波的轨迹概要数据抽取	63
4.3.3	基于哈尔小波系数的欧氏距离表示	64
4.3.4	基于哈尔小波的欧氏距离上、下界	67
4.4	基于欧氏距离的查询算法:ED-FTB	71
4.4.1	ED-FTB 算法实现	71

4.4.2	ED-FTB 算法性能分析	73
4.5	实验分析	74
4.5.1	实验设置	75
4.5.2	算法有效性	75
4.5.3	算法可扩展性	80
4.6	本章小结	81
第五章	基于下界剪枝的分布式 k 近邻查询	83
5.1	基于下界的逐步剪枝策略	83
5.1.1	策略设计原理	83
5.1.2	策略实现方案	84
5.2	基于动态时间弯曲距离的概要数据	88
5.2.1	基于动态时间弯曲距离的轨迹相似度量	88
5.2.2	基于包围信封的概要数据	90
5.2.3	满足 DTW 距离约束的包围信封及下界	91
5.2.4	基于多粒度包围信封的下界	93
5.3	基于动态时间距离的查询算法:DTW-FLB	98
5.3.1	DTW-FLB 算法实现	98
5.3.2	DTW-FLB 算法性能分析	100
5.4	实验分析	101
5.4.1	实验设置	101
5.4.2	算法有效性	101
5.4.3	算法可扩展性	104
5.5	本章小结	105
第六章	总结与展望	107
6.1	研究总结	107
6.2	研究展望	108
参考文献	111
附录一	124

致谢	126
发表论文和科研情况	127

插图

图 1.1	两种常见分布式架构	3
图 1.2	基于协调者-远程结点架构的查询展示	6
图 1.3	本文的组织结构	9
图 2.1	豪斯多夫距离	25
图 2.2	弗雷歇距离	25
图 3.1	剪枝示例	36
图 3.2	轨迹片段表示	38
图 3.3	轨迹索引构建	39
图 3.4	Join 查询示例	46
图 3.5	数据加载开销	49
图 3.6	数据存储开销	49
图 3.7	基于给定对象的查询性能	50
图 3.8	时空范围查询性能	51
图 3.9	k 近邻轨迹查询性能	52
图 3.10	Join 查询性能	53
图 4.1	利用多粒度概要数据不断逼近真实距离	59
图 4.2	剪枝示例	60
图 4.3	哈尔小波变换-误差树	64
图 4.4	基于 Haar 小波系数的距离计算示例	66
图 4.5	欧氏距离上、下界示例	70
图 4.6	ED-FTB 剪枝效果图	76
图 4.7	下界对剪枝结果的影响	77
图 4.8	算法执行策略详细比较	78

图 4.9	n, k 和 M 对 ED-FTB 算法通信开销的影响	79
图 4.10	算法性能比较	80
图 4.11	ED-FTB 算法可扩展性 (T-Big 数据集)	81
图 5.1	利用下界和全局阈值的剪枝过程	84
图 5.2	带约束的动态时间弯曲	88
图 5.3	包围信封	90
图 5.4	DTW-FLB 剪枝效果	102
图 5.5	DTW-FLB 通信开销与 n, M 和 k 之间的关系	103
图 5.6	DTW-FLB 性能与 δ 间的关系	104
图 5.7	DTW-FLB 可扩展性	105

表格

表 3.1	TrajSpark 数据集描述	48
表 4.1	实验数据集描述	75
表 5.1	时间序列 \mathcal{S} 的多粒度包围盒	91

第一章 绪论

轨迹大数据是移动互联网时代的产物，记录了移动对象的行为信息。通过轨迹数据分析技术可以挖掘人类活动规律与行为特征、城市车辆的移动模式、大气环境变化规律等信息。这些挖掘成果为研究城市发展、社会变迁和自然环境演变提供重要的参考价值。本文重点研究了轨迹大数据的管理和 k 近邻轨迹查询问题。本章中，章节 1.1 阐述了轨迹数据管理和 k 近邻轨迹查询的研究背景；章节 1.2 详述了本文的具体研究内容以及所遇到的挑战；章节 1.3 简述了本文的主要研究方法和研究贡献。

1.1 研究背景

随着卫星定位导航、无线通信和普适计算等高新技术的不断发展，带有定位功能的移动智能设备已被广泛应用，并成为社会生产、生活的重要组成部分。移动对象（含人、动物和车辆等）在主动或被动使用这些设备的同时，产生了大量记录其移动历史行为的轨迹大数据。例如：滴滴出行，全球最大的一站式多元化出行平台，在 2017 年 10 月发布的《第三季度全国重点城市交通出行报告》中称其每天新增的定位轨迹数据超过 70TB¹。轨迹大数据具有数据量大、数据增长快等特性。轨迹是地理空间在时间轴上形成的多维空间中的一条曲线，表示了移动对象在一段时间内时空信息等移动行为的变化。每条轨迹可看作一条时空采样点构成的序列，其中每个采样点记录了时间、位置、速度、方向等信息。从微观角度讲，轨迹数据蕴含了移动对象的移动模式与规律 [1–3]，例如从轨迹数据中我们可以发现市民的居住地、工作地和消费娱乐场所 [4]。从宏观角度来讲，海量的轨迹大数据蕴含了群体的移动迁徙和社会发展的变化，如城市的发展、交通的演化以及社会的变迁 [5]。通过轨迹分析等手段进行知识发现，并将它们运用在各种交通和服务应用系

¹<http://www.xiaojukeji.com/index/index>

统中，包括交通导航 [6, 7]、交通智能指挥 [8]、车辆异常监控 [9, 10]、物流配送、城市规划、军事调度等 [11–13]。

海量的轨迹数据具有重要的社会和应用价值，不仅为解决拥堵、改善交通服务、缓解能源紧缺和降低大气污染等社会问题提供了新的机遇，而且对认知人民的社会活动、优化公共资源配置、为建立新型共享经济有着特殊的意义。在工业界，百度地图能根据实时轨迹数据进行路径规划。摩拜进行了基于自行车骑行目的地预测的研究²。美团和饿了么等外卖公司设计了基于轨迹数据的智能派单系统³。上海电信根据手机信令数据研究了人口流动分析 [14]。学术界也出现一些针对轨迹数据挖掘的热点研究工作，包括轨迹管理系统研究 [15–18] 查询可伸缩的快速轨迹聚类 [19–22]、轨迹流的连续查询 [23]、路径规划及路径发现 [24]、汇集模式发现 [25]、旅伴模式发现 [26, 27] 以及实时共享乘车 [28] 等。

轨迹相似性研究是轨迹挖掘和分析的核心内容之一。现有的轨迹数据间的相似性通常通过它们之间的距离来度量，即两条轨迹数据之间的距离越小，则认为它们之间越相似。根据这一准则，研究者们已经将广泛应用于时间序列分析的距离度量函数应用到轨迹数据中，这些度量准则包括：欧氏距离 (Euclidean Distance, ED)[29]、动态时间弯曲 (Dynamic Time Warping, DTW)[30]、最长公共子序列 (Longest Common Subsequence, LCSS)[31, 32]、实值序列上的编辑距离 (Edit Distance on Real sequence, EDR) [33] 和带有惩罚项的编辑距离 (Edit Distance with Real Penalty, ERP) [34]。此外，部分研究者针对轨迹数据特性设计了豪斯多夫距离 (Hausdorff Distance, HD)[22]、弗雷歇距离 (Fréchet Distance, FD) [35, 36]、一路距离 (One Way Distance, OWD) [37]、带投影的编辑距离 (Edit Distance with Projections, EDwP) [38] 等。此外，还有部分学者针对轨迹的语义特性设计了用于计算轨迹语义相似度的距离 [39–41]。尽管这些距离度量函数可以较好地捕捉到轨迹之间的相似度，但它们的计算复杂度高，要想应用到大规模轨迹数据集上需要进一步研究如何提高计算效率。此外，如何针对特定问题选择合适的相似度度量函数也是重要的研究内

²<https://www.biendata.com/competition/mobike/>

³<http://blog.csdn.net/jinjin603/article/details/78793243>

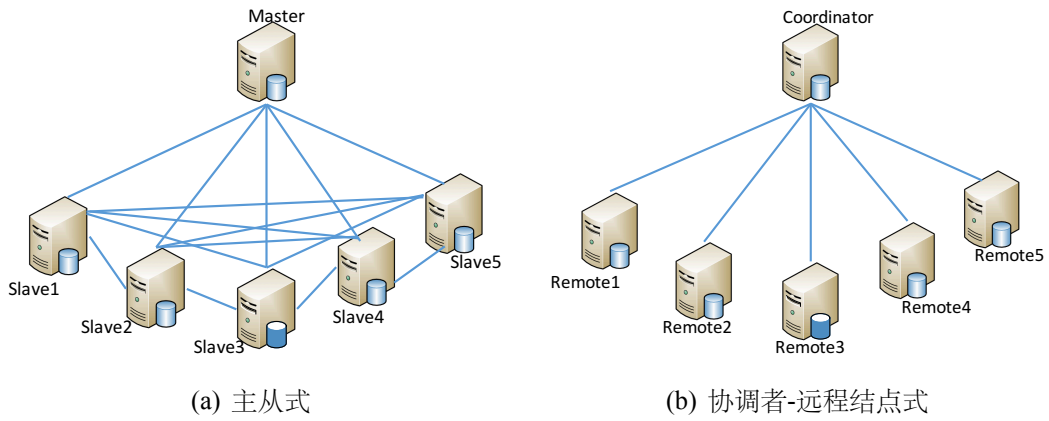


图 1.1: 两种常见分布式架构

容 [42, 43]。目前，仍然缺乏统一的查询处理框架以支持多种距离度量准则的应用。

本文主要研究了 k 近邻轨迹查询，即根据一条查询轨迹 Q 找出 k 条与之距离最近（即最相似）的轨迹，是数据分析和挖掘的重要内容。其有着广泛的应用场景。比如说，交通部门想要开辟一条新的公交线路。其可以根据海量的人、车轨迹数据查看早/晚上下班高峰期间是否有 k 条与该路线相似的轨迹。此外，旅行平台可以通过查找某旅客的相似旅行轨迹给他推荐旅行伙伴或路线。此外该查询也有着广泛的科研价值，其是用户移动模式分析和异常轨迹检测等问题的核心内容。传统的集中式查询算法专注于查询时间效率的提高，常通过轨迹索引 [38, 44–47]、压缩 [48–50] 和上、下界计算 [51, 52] 等方法来进行预剪枝从而避免过多的距离计算。而海量的轨迹往往以分布式形式存储，即轨迹数据会分布在不同的数据节点上。因而已有的集中式算法无法满足分布式轨迹数据的查询需求。

常用的分布式场景分为两种（如图1.1所示）：一种是基于主-从架构的。Hadoop 和 Spark 系统就是典型基于主-从架构实现的系统，并随着这些系统的流行，该架构得到了广泛的应用。在主-从式中，系统包含一个主节点 (Master) 和多个从节点 (Slave)。其中从节点存放具体数据并负责对本地图任务的执行，主节点存放数据的元数据并负责任务的分发和调度。该架构往往要求所有物理结点在同一个集群内部，结点间通过高速网络连通。使用该框架进行轨迹处理分析的前提是，所有的轨迹都能被采集到一个数据中心中。在这种架构中，降低问题的计算开销是设计方法

的首要考虑。另一种是基于协调者 (Coordinator) - 远程结点 (Remote) 架构的。该架构中存在一个协调者和多个远程节点, 其中协调者结点不存储任何数据只负责跟远程结点通信, 而远程结点负责存放本地数据, 且各远程结点间互不通信。该架构一个典型应用就是众包 (Crowdsourcing)。即个人或组织 (协调者) 将一个问题提交给一个群体 (所有远程结点), 群体中的各个单位, 根据自身的知识来给出问题的结果。最终, 提问者根据所有用户的答案获得最终结果。而在此架构中, 降低问题的通信开销是解决问题的首要考虑。从图中, 我们可以看出这两个架构的主要结构区别是后一种架构数据节点间彼此隔离互不通信, 而第二种架构要求数据节点间能互相通信。这导致它们使用的应用场景也有所区别, 第一种常用于集群内部, 而第二种架构往往用于多个数据结点间构建连接。针对海量轨迹数据, 若都能被采集到某一数据中心 (常见于滴滴等轨迹数据拥有者), 则采用第一种架构进行管理。若由于采集、通信、隐私等原因, 数据不能被统一存放到单一数据中心, 往往以第一种架构进行管理。如每个智能手机拥有者, 在各自的手机中保存了自己的轨迹数据, 其他任何人都无法知晓其内容。针对这两种分布式场景, 急需提出相应的 k 近邻查询方案。

首先, 使用主-从式架构进行 k 近邻查询前提是, 我们已经采集了海量的轨迹数据并保存在集群中, 且采集的数据集可能会不断增大。传统的轨迹数据库已无法满足需求 [53–55], 而直接将海量轨迹以分布式文件 (如 HDFS) 的方式存放在系统中, 虽然简单、易行, 但无法满足用户的查询分析的时效性需求。为此, 我们首先需要对海量轨迹进行有效地管理以便提供低延时的查询结果。目前, 已经有许多改造 Hadoop、Spark 等分布式系统, 以构建满足轨迹数据的管理系统。这些系统根据存储介质的不同可分为两类, 一类是基于磁盘的时空数据管理系统, 典型的有 TRUSTER[56, 57]、SpatialHadoop[58]、AQWA[59] 等。另一类是基于内存的管理系统, 典型的有 SpatialSpark[60]、GeoSpark[61]、LocationSpark[62]、Simba[63] 以及针对 k 近邻轨迹查询专门设计的 TrajSimba[18] 等系统。已有基于磁盘的系统往往不能提供实时查询结果, 而基于内存的管理不能满足数据不断增加的场景。为此,

仍缺乏满足大数据要求的轨迹管理系统以提供低延时的查询结果。

其次，在协调者-远程结点架构中，每个远程结点各自管理着本地的轨迹数据，且对于协调者和其他远程结点来说，其相当于一个黑盒，无法知道其保存数据的信息。而协调者结点扮演查询引擎的角色，其接受用户提交的查询，并将查询相关数据发送给远程结点。远程结点根据接收到的数据或请求，返回相应的查询结果或执行相应的操作。基于该架构的一种直观 k 近邻查询实现方法如下：协调者将接收到的查询轨迹直接发送给所有远程结点。远程结点根据接收到的查询轨迹，对其本地轨迹进行两两距离计算并将距离值返回给协调者结点。但该方案面临着通信开销过大的问题，其中通信开销值达到 $O(n * M)$ ，其中 n 表示带查询轨迹的长度， M 表示远程结点的数量。此外，由于该方案需要跟所有轨迹进行距离计算，从而导致时间开销也很高。为此，Smart Trace[32] 和 Smart Trace⁺ [31] 等工作通过发送时间序列包围信封以计算最长公共子串距离上、下界并剪枝，从而降低计算开销。BandWidth[30] 通过多粒度包围信封计算动态时间弯曲距离的下界来逐步剪枝。LEEWAVE[64] 使用小波变换来降维时间序列数据并根据降维数据计算欧氏距离的上、下界来剪枝。以上优化都是针对一维时间序列数据，而轨迹数据是天然的多维数据。此外，它们都是针对某一具体距离度量方式进行针优化，优化方法不能迁移到其他距离下的查询中。为此，仍然缺乏通用查询方案。

1.2 研究内容与挑战

本文针对海量的轨迹数据进行了分布式数据管理及分布式 k 近邻查询研究。首先，针对主-从式架构下的海量数据管理及查询，本文的研究目标是如何对获得的海量轨迹数据进行有效管理，并对以 k 近邻为代表的轨迹查询提供高效的查询接口。传统的集中式轨迹数据管理系统不能满足轨迹大数据的管理要求，而现有的基于集群的轨迹数据管理系统往往是基于磁盘的管理系统，存在着 I/O 开销较大的问题。它们不能满足实时查询的需求，即用户给出查询轨迹后，需要在较短的时间内给出查询结果。目前，Spark 等分布式计算平台因其良好的内存利用特性，成为

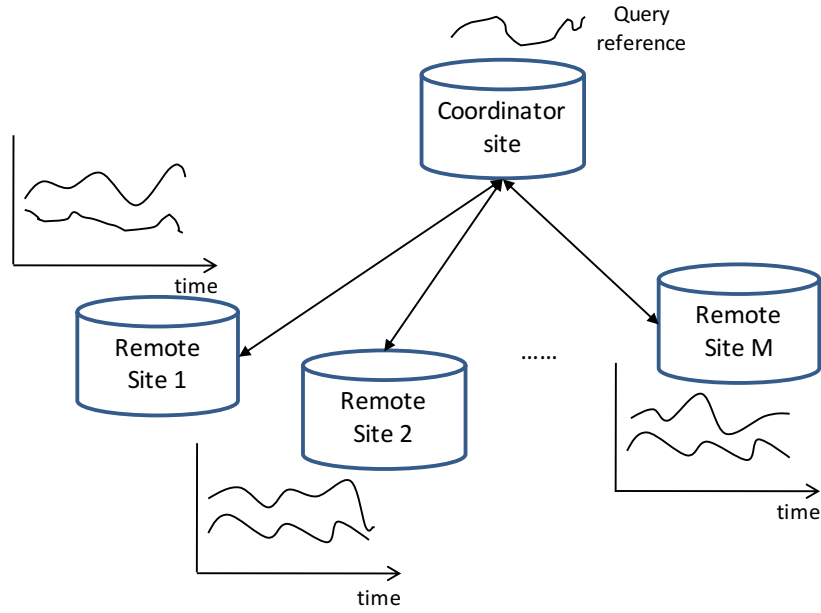


图 1.2: 基于协调者-远程结点架构的查询展示

提供实时数据分析的有效供给。本文的研究思路就是如何利用这些开源分布式内存平台，构建高效的轨迹数据管理系统。

其次，针对协调者-远程节点结构下的 k 近邻查询进行了研究。该结构下的轨迹数据管理如图1.2所示，每个远程结点各自管理着一批轨迹数据。协调者负责接收用户提交的查询和与远程结点进行通信。在该框架中，协调者结点接收到用户的查询轨迹后，由于不知道各个远程结点中数据的任何信息。因此，一个简单的查询方案就是，协调者结点直接将查询轨迹直接发送给所有远程结点，并在远程结点计算查询轨迹与所有候选轨迹的距离。最后，协调者结点根据接收到的距离值，找出距离最近的 k 条轨迹。该方案简单易行，但由于需要将查询轨迹发送给所有远程结点，当查询轨迹数据量较大或远程结点较多时，存在着通信开销较大的问题。此外，由于需要计算查询轨迹与所有候选的距离，而距离计算往往复杂度较高。因此，该方案的计算开销也较大。为此，在该框架下，我们需要考虑如何设计出通信和计算开销都低的解决方案。

在处理 k 近邻查询时，还存在着一个不可忽略的问题即轨迹距离度量方法的选择。现有的轨迹查询和分析的工作往往都是针对某一选择或精心设计的度量方

法以满足给定应用场景的需求。因此，现有的方法缺乏通用性。综上所述，

本文研究面临的挑战主要表现为以下四点：

- **首先，针对获得的海量轨迹数据，缺乏高效的管理系统以提供低延时的查询结果。**基于集群的分布式数据管理方式，是处理轨迹大数据的必由之路。目前已经有许多基于分布式平台的轨迹数据管理方案，但这些方案大都是基于磁盘的管理系统，无法提供实时的查询结果。此外，这些系统大都是针对静态轨迹数据设计，不能满足轨迹数据不断增加的需求。为此，需要设计出满足能够对不断增长的轨迹大数据提供实时查询需求的管理系统。
- **其次，针对时间序列上已有的 k 近邻轨迹查询技术往往只针对某一距离准则设计，缺乏通用性。**距离度量准则是轨迹数据挖掘分析的核心内容之一，选用不同的度量函数，导致的挖掘结果以及对结果的理解往往大不相同。现有的工作都是针对某一研究问题，精心挑选或自定义一距离函数以满足研究问题的需要。因此，所提出的解决方案具有较高的局限性，很难推广到其他的问题中。为此，需要提出通用或对一类问题适用的解决方案。
- **最后，现有协调者-远程结点架构下的 k 近邻查询面临着通信和计算开销过大的问题。**直接将原始查询轨迹发送到所有远程结点的方式，其通信开销随轨迹长度和远程结点的个数的增加而快速增大。而现有的数据降维方案虽然降低了数据的维度，但也造成了数据信息损失，使得数据不再直观且不能保证查询结果的准确性。此外，这些方法常用于处理一维时间序列数据，而轨迹是天然的多维时间序列数据。为此，需要提出新的轨迹数据降维方法，在保证查询结果正确性的同时，降低通信和计算开销。

1.3 主要贡献

本文围绕分布式 k 近邻轨迹查询这一问题，系统性的提出了解决方案。首先，针对采集得到的轨迹大数据提出了基于主-从式架构的管理系统，该系统能对 k 近

邻等查询提供低延时的查询结果。接着，针对协调者-远程结点架构设计了两种通用查询处理策略，以解决通信开销过大和距离准则多的问题。这两个策略区别是第一种策略针对利用概要数据能同时计算距离上、下界的场景，第二种针对仅能利用概要数据设计出下界的场景。在这两种策略中，我们均使用多粒度概要数据来进行剪枝，并通过引入具体的度量准则验证了两个策略的有效性。最后，使用真实轨迹数据集验证了算法的性能。因此，本文的主要贡献分为以下 3 点：

- 针对集群内海量轨迹数据的管理问题，设计了基于分布式内存的轨迹数据管理系统 **TrajSpark**。TrajSpark 构建在分布式开源系统 Spark 之上，以充分利用分布式集群的内存资源并减少查询的 I/O 开销。为适应非结构化轨迹数据的管理需求，TrajSpark 设计了适用于轨迹数据的表达方式 **IndexTRDD**，并使用局部和全局索引相结合的轨迹索引策略此提供查询效率。此外，TrajSpark 引入了时间衰减模型以适应轨迹数据不断增加所导致的数据划分开销较大问题。最后，针对三种典型轨迹查询设计了查询算法，并在真实海量轨迹数据集上验证了算法的性能。
- 针对根据概要数据能同时设计出上、下界的距离函数，设计了同时使用上、下界针对协调者-远程结点架构下 k 近邻查询进行逐步剪枝的处理策略。基于该策略，实现了基于欧氏距离的查询算法。协调者-远程结点架构下的 k 近邻查询的瓶颈在于通信和计算开销，本文使用降维技术以获取概要数据，并通过概要数据来进行距离估计并剪枝，从而达到降低数据通信开销的目的。针对根据概要数据能够同时计算出上、下界的场景，设计了使用多粒度数据逐步逼近上、下界的剪枝策略。接着，针对欧氏距离，使用哈尔小波得到多粒度概要数据，并设计了适用于轨迹数据的距离上、下界。然后，将欧氏距离应用到我们所提策略中，设计了查询算法 **ED-FTB**。最后，我们通过理论分析和大量实验验证了 ED-FTB 算法的有效性和可扩展性。
- 针对根据概要数据仅能设计出下界的距离函数，设计了仅使用下界针对协调

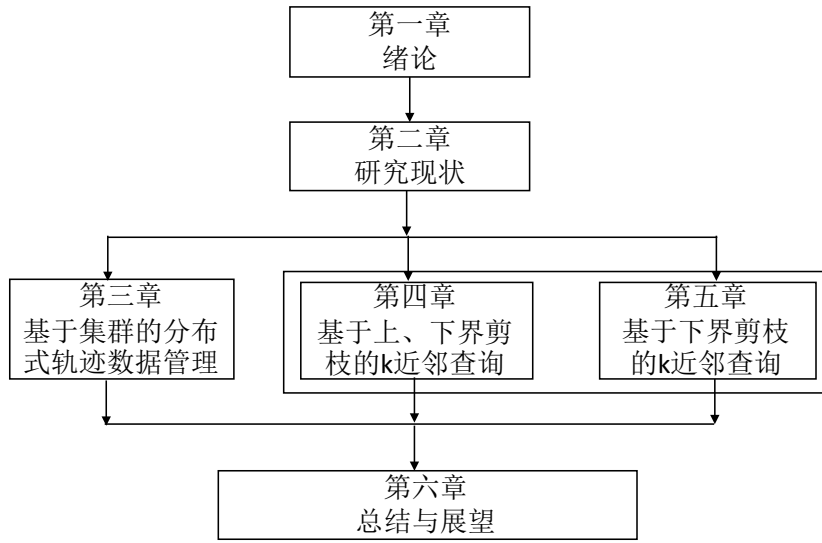


图 1.3: 本文的组织结构

者-远程结点架构下 k 近邻查询进行逐步剪枝的处理策略。基于该策略，实现了基于动态时间弯曲距离的查询算法。本文针对仅能根据概要数据计算出下界的场景，设计了使用多粒度数据进行逐步逼近距离下界和全局剪枝阈值的剪枝策略。接着，针对动态时间弯曲距离，设计了基于包围盒的多粒度概要数据，同时设计了适用于轨迹数据的距离下界。然后，将动态时间弯曲距离应用到该策略中，设计了查询算法 DTW-FLB。最后，我们通过大量实验验证了 FLB-DTW 算法的有效性和可扩展性。

1.4 章节安排

本文一共分为六章，章节安排如图 1.3所示：

- 第二章从问题定义、轨迹数据管理、轨迹降维以及轨迹数据相似性查询三个方面介绍了研究背景知识和研究现状。
- 第三章介绍了基于集群分布式内存的轨迹管理系统 TrajSpark，这一工作是基于主-从架构实现。
- 第四章介绍了使用距离上、下界并逐步剪枝的查询策略，并实现了基于欧氏

距离的查询算法。

- 第五章介绍了仅使用距离下界进行逐步剪枝的查询策略，并实现了基于动态时间弯曲距离的查询算法。第四、五两章内容都是基于协调者-远程结点架构实现。
- 第六章对上述已有工作进行了总结，并展望了未来的研究方向和内容。

第二章 研究现状

本章第一节首先对数据模型和要解决的查询问题进行了定义，然后从以下三方面介绍了相关的研究工作：轨迹数据管理、轨迹压缩、轨迹距离度量和 k 近邻轨迹查询，最后对本章内容进行小结。

2.1 数据模型及问题定义

本文的研究目标是给定一条查询轨迹，从存储在若干彼此独立的远程结点中的轨迹中找出 k 条距离最短或相似度最高的轨迹。为此，本节首先介绍了轨迹数据模型，接着形式化给出查询定义。

2.1.1 轨迹数据

轨迹是描述移动对象移动行为的数据。通常来说，一条轨迹 T 可看做包含 n 个元素（即轨迹点）的时间序列。每个轨迹点 p 包含了时间、位置等维度的信息。因此轨迹可被形式化定义为如下：

定义 2.1.1 (轨迹 (Trajectory)). 轨迹形式化表示为: $T = \{p_0, p_1, \dots, p_{n-1}\}$ 。

$$T = \{p_0, p_1, \dots, p_{n-1}\} \quad (2.1)$$

其中 $|T| = n$ 代表轨迹所包含的点数，即轨迹的长度。每个轨迹点 p 包含时间 (t)、位置 (l) 等维度的信息。因而 $|p| = d$ 称为轨迹的维度。此外轨迹中的点严格按照时间升序排列，即 $\forall i, j, 0 \leq i \leq j < n$ 则 $p_i.t \leq p_j.t$ 。

轨迹数据的来源多样且复杂。根据移动对象的划分可分为如下几类 [13]:

- **人类活动轨迹数据:** 该类数据分为主动式和被动式。主动式数据是人们主动利用移动定位设备分享或汇报自己的位置等信息。典型的有社交网络中的数

据, 用户提交位置获得服务的数据。被动式数据是人们无意间使用各种服务时所产生的轨迹数据。典型的有公交刷卡轨迹 [65] 和手机的信令轨迹数据 [4]。

- **交通工具轨迹数据:** 这类数据主要是交通工具使用车载 GPS 设备所产生的移动轨迹数据。例如, 出租车、公交车的活动轨迹数据 [29]。
- **动物活动轨迹数据:** 这类数据是为了研究动物生活、迁徙等行为和习惯而捕获的数据 [66]¹。
- **自然现象活动轨迹数据:** 这类数据典型的有台风、冰山、海洋事件等的轨迹数据², 用以探索自然现象的活动规律。

轨迹数据符合大数据时代的 3V 特征, 即量大、实时、多样。但是由于受设备、采样频率等因素影响, 数据质量较低且各个轨迹的采样间隔差异显著。这些问题导致原始轨迹数据的可用性较低。这为我们对其管理和有效利用造成了挑战。

2.1.2 分布式 k 近邻轨迹查询

轨迹数据往往是分布式采集并存储的。为此假设有 M 个远程结点, 每个远程结点 i 包含轨迹数据集 \mathcal{D}_i 。那么整个分布式轨迹数据集 $\mathcal{D} = \bigcup_{i=1}^M \mathcal{D}_i$ 。我们的目标是给定查询轨迹, 从分布式轨迹数据集 \mathcal{D} 中, 找出与其距离最近的 k 条轨迹。下面我们将给出查询的形式化定义:

定义 2.1.2 (分布式 k 近邻轨迹查询). 该查询形式为 $kNN(Q, \mathcal{D}, DM, k)$, 其中 Q 为给定查询轨迹, \mathcal{D} 为分布式轨迹数据集, DM 为距离度量准则以及 k 为返回结果集大小。查询的目标是返回满足如下条件的轨迹集 \mathcal{S} : (1) $\mathcal{S} \subseteq \mathcal{D}$; (2) $|\mathcal{S}| = k$; (3) $\forall C \in \mathcal{S}, C' \in \mathcal{D} - \mathcal{S}, DM(Q, C) \leq DM(Q, C')$ 。

¹<http://www.fs.fed.us/pnw/starkey/data/tables/>

²<http://weather.unisys.com/hurricane/atlantic/>

与传统的集中式环境下 k 近邻轨迹查询相比, 分布式场景下的查询不仅注重查询效率, 而且尤其注重通信开销。这是由于分布式场景中, 远程结点和协调者结点的带宽资源往往是有限的。而高的通信开销, 意味着用户可能要花费更多的金钱。为此, 我们应该在不牺牲太多计算开销的基础上降低通信开销。

2.2 轨迹数据管理

本节将介绍如何对轨迹数据进行有效的管理, 并介绍现有的轨迹管理系统。

2.2.1 轨迹索引

一个轨迹数据库需要能够在任意数据量下对提供对轨迹的有效查询, 而选用合适的轨迹索引策略往往很查询效率有着很大的影响。不同与其他类型数据库, 轨迹数据库由于其时间和空间维度特性, 需要专门的索引技术, 以满足轨迹查询的需求。

现有的轨迹索引技术大致可以分为三类: 第一类, 使用传统多维索引技术如 R 树增加时间维度后的改进索引, 典型的有三维 R 树和 STR 树 [67]; 第二类就是多版本结构的索引, 典型的有 RT 树 [68]、 HR 树 [69]、 $HR+$ 树 [70] 和 $MV3R$ 树 [71]。这类方法为每个时间戳构建一个单独的 R 树并为两颗连续的树共享共同的部分以减少冗余; 第三类就是讲空间维度进行划分得到一系列的方格, 然后为每个方格内的数据在时间维度构建索引。这类工作包含 $SETI$ [54] 和 $MTSB$ 树 [72]。这些算法大都针对静态轨迹数据设计, 无法支持轨迹数据动态增加的需求。为此, SEB 树 [73] 和 QU -Trade 两颗 R 树的改进以提供数据动态增加。

2.2.2 集中式轨迹数据管理

目前已经有一些针对历史数据的轨迹管理系统以支持轨迹查询 [53–55]。Botea 等首先构建了针对轨迹数据的管理系统 $PIST$ [53]。该系统将轨迹数据以点为基本单位构建空间索引。它首先统计轨迹数据在空间维度的分布, 然后构建空间索引划分数据点。最后对每个划分内部的数据根据时间维度构建索引。Chakka 等提出

了以子轨迹为基本单元的管理系统 SETI[54]。该系统同样首先根据统计数据构建空间索引。然后根据空间索引划分轨迹数据，并将每个划分内同一移动对象的子轨迹作为基本处理单元构建时空索引。该工作表明，基于子轨迹的索引策略比基于点的策略查询效率要高。进一步地，Philippe 等提出了基于 I/O 开销模型的轨迹数据管理系统 TrajStore 以支持动态轨迹划分 [55]。此外，TrajStore 对每个划分内的子轨迹进行聚类并用簇中心来代表簇内所有轨迹，从而降低数据的存储开销。它还使用了多种数据压缩技术以进一步降低存储开销。Wang 等提出针对大内存服务器的轨迹数据系统 SharkDB[74–76]。该系统提出了 3 种存储框架用以在降低数据存储开销同时增大内存命中率。此外，它还引入了分层索引以提高查询效率。上述轨迹系统需进行功能扩展以实现近邻轨迹查询。

2.2.3 基于集群的分布式轨迹数据管理

近 10 年来，随着 Hadoop³、OpenStack⁴和 Spark⁵ 等开源大数据处理系统的普及，在这些系统上构建轨迹数据管理分析系统成为新的方向。目前，已经出现了大量基于这些开源分布式平台的轨迹数据管理系统以支持轨迹查询。Lu 等设计了并行的空间数据 DBMS 系统 Parallel-Secondo[77]。该系统在每台节点上使用传统的空间 DBMS 来实现数据存储和查询实现，仅使用 Hadoop 作为结点间任务调度器。Ma 等首先提出了基于 Hadoop 的轨迹数据管理系统 [56, 57]。为提高查询效率，其构建了两类索引。第一类是基于空间的索引，用于将数据按空间维度划分。此类索引能加速带有空间约束的查询，但如果查询某个移动对象的轨迹，则需要遍历许多数据分区。为此设计了第二类基于移动对象的倒排索引，该索引中存储了每个移动对象的轨迹数据是存储在哪些数据分区上。Clost[16] 系统通过建立一个多层索引用以数据划分。同时，其支持数据的批量增加。SpatialHadoop[58] 系统是基于 Hadoop 的典型空间数据管理系统。该系统从以下四个方面对 Hadoop 进行了改进

³<http://hadoop.apache.org/>

⁴<https://www.openstack.org/>

⁵<http://spark.apache.org/>

以满足空间数据查询的需求。(1) 提供了基于多种索引树的存储策略。首先根据全局索引以完成数据划分,其次在每个数据块内支持块内局部空间索引。这种全局-局部的索引方式能大大减少 I/O 开销;(2) 设计了新的文件读写方式,以支持对块内索引数据的读写;(3) 设计了多种查询接口,以方便开发这进行查询调用;(4) 设计了基于 **Pig Latin** 的查询语言,以方便普通用户实现查询功能。但 **SpatialHadoop** 只能用来管理静态数据,当新的数据到达时,需要将整个数据集重新划分,因而开销较大。**AQWA**[59] 系统是其改进版本,允许数据动态添加。**AQWA** 建立了新的数据划分模型,同时考虑了用户查询和数据分布对划分的影响。以上系统最终将查询需求,变成定制的 **Map/Reduce**[78, 79] 任务。

此外,还有一些基于 **Hadoop** 组件或类 **Hadoop** 系统的时空数据管理系统。**Ablimit** 等等提出了基于 **Hive**⁶的轨迹管理仓库 **Hadoop-Gis**[80],其通过网格索引以提高范围和 **join** 查询的效率。**MD-HBase**[81] 和 **R-HBase**[82] 是两个基于 **HBase**⁷设计的实时时空数据管理系统。**MD-HBase** 使用 **Z-Curve** 空间填充曲线来划分数据空间。接着对所划分子空间使用四叉树或 **k-d** 树索引来构建空间索引。**R-HBase** 使用了局部性更好的希尔伯特空间填充曲线来划分数据空间,并使用 **R** 树来构建空间索引。这两个系统借助了 **HBase** 的特性可支持数据的实时插入和修改。**Geomesa** 系统[83] 与以上两个系统类似,可以快速部署到类似 **HBase** 的键值对存储系统中,且其从语言层定义了空间查询操作接口以方便用户调用。相较于以上系统,**Elite** 系统[84] 专门设计用于处理不确定轨迹数据并提供了丰富的不确定查询接口。

尽管基于 **Hadoop** 生态圈内系统构建的分布式轨迹管理系统能满足轨迹查询的需求,但也面临着由于 I/O 开销较大,无法提供实时、高吞吐率查询服务的问题。为此,出现了许多基于 **Spark** 等分布式内存的时空数据管理系统。**SpatialSpark**[60]、**GeoSpark**[61] 和 **LocationSpark**[62] 是最早的几个空间查询系统,但这些系统仅能处理空间维度上的查询,没有考虑时间维度的影响。最新地,**Dong** 等人提出了新的基于 **SparkSQL** 的多维数据管理系统 **Simba** [63]。并从查询解析到查询执行

⁶<https://hive.apache.org/>

⁷<https://hbase.apache.org/>

等方面作了优化。其中最显著的是在其查询执行计划设计方面使用了空间索引等技术来提高查询效率。此外，它还从语言层提供了用户查询接口。相比以上系统，TrajSpark[17] 是专门设计用来处理轨迹数据的系统。其实现了 3 类典型的轨迹查询接口，并利用全局-局部索引策略以提高查询效率。此外，目前还存在着专门用于对时空数据进行复杂分析的系统 [85, 86]。STORM[86] 系统设计用来处理交互式近似查询。相比于已有的设计用来精确查找的系统，该系统能在查询提出后，通过逐渐增加采样数据以不断以一定置信度给出查询结果。用户根据系统不断精细的结果随时可以停止查询。OceanST[85] 系统同样适用采样技术来获得查询近似结果，且能处理数据不断增加的场景（STORM 仅能处理静态历史数据）。此外，它还提供了丰富的查询接口用以满足各种查询需求。

2.3 轨迹压缩

将查询轨迹数据压缩是降低通信开销的有效方式。现有轨迹压缩的方法根据处理数据类型的不同可分为 3 类 [87]：第一类是传统的时间序列压缩算法，其根据应用场景又可以分为离线和在线压缩两类算法。第二类是基于路网结构的轨迹压缩算法，该类算法依赖于所给路网数据。第三类是语义压缩，其目标是将原始数值型轨迹数据转换为人们能理解的语义信息。下面将详细介绍这三类压缩算法。

2.3.1 传统离线时间序列压缩算法

离线轨迹压缩算法主要的应用场景是，给定一静态历史轨迹数据库，对其中的轨迹进行压缩。该类压缩算法不考虑数据集的增加、删除和修改。该类算法的提出主要是为了解决时间序列数据的降维（即降低序列的长度）。由于轨迹本质上也是时间序列数据，因此该类方法也可用于轨迹的压缩。尽管相比于传统时间序列数据，轨迹数据属于多维数据。但其各属性维度上数据一般互相独立，因此只需将降维方法应用到各个维度上分别进行压缩即可。本小节主要介绍几种典型的离线时间序列压缩算法。

道格拉斯-普克算法：道格拉斯-普克 (Douglas-Peucker, DP) 算法将原始时间序列看作一条多维曲线，其目标是找出一条与原始曲线相似但使用更少点构成的简化曲线。此外，构成简化曲线的点必须来自原来曲线。该算法中原始轨迹与简化轨迹间的“不相似性”是通过两条曲线间的豪斯多夫 (Hausdorff) 距离表示。简化曲线中的点即为压缩后的数据。DP 算法的思想简单且性能较好，在各个领域都有广泛的应用。但其缺点就是算法复杂度较高达到 $O(n^2)$ ， n 是轨迹长度。文献 [88] 提出了借助外部存储结构的改进算法，把时间复杂度降低为 $O(n \log n)$ 。此外，文献 [89] 和 [90] 分别提出了同时时间和空间维度的改进距离函数 (Time-Distance Ratio, TDR) 和 (Top-Down Time Ratio, TD-TR)，克服了原始 DP 算法仅考虑空间维度的缺点。

离散傅里叶变换：离散傅里叶变换 (Discrete Fourier Transform, DFT)[91] 是时间序列数据降维的常用方法，并且已有许多扩展和改进方案被提出 [92–94]。其基本思想是任何一个信号，不管其多复杂，都可以分解为有限个正弦和余弦曲线的叠加。每条曲线可以由一称为傅里叶系数的复数表示 [95]。此时，我们时间序列由时域变换到频域了。使用频域的好处有很多，其最重要的就是能进行数据压缩。一条长度为 n 的信号（即时间序列）可以被分解为 n 个正/余弦曲线，且这 n 个曲线能重新组合成原来的信号序列。但由于 n 个曲线中存在着许多振幅较低的，这些低振幅曲线对重新构建原来的信号的贡献较低。因此，其对应的低振幅系数可以被丢弃掉，而保留那些振幅较高的系数。这些保留的系数重新组合所构成的信号数据与原始信号数据相比，并没有太多的信息丢失。因而，达到了使用少量数据来表示原始信息的目的。此外，文献 [95] 中提出了帕斯瓦尔定理 (Parseval's law)，即原始时域内信号数据间的欧氏距离等于变换后频域内系数间的欧氏距离。因此，若经离散傅里叶变换且将低振幅的系数丢掉后，使用剩下的系数计算距离，所得到的结果必然是原始欧氏距离的下界（丢失的数据都是正数）。文献 [96] 提出了基于该下界的剪枝方法。

小波变换：小波变换 (Wavelet Transform) 将信号分解为一系列基/母小波函

数的和与差的组合，这一点与离散傅里叶变换类似。但它们之间也有许多的不同点。其中一个重要的区别是，小波在时间上是局部的，即小波系数只代表原始数据中的一小分子段。这也是“小波”这一名词的由来，“小波”就是小区域、长度有限、均值为 0 的波形。而傅里叶系数总是代表着数据的全局信息。对于一些需要研究数据局部信息的应用来说，具有多解析度特性的小波变换更为适用。当原始信号/时间序列数据被小波分解之后，会得到一个小波系数序列。该序列的前几个系数包含了数据的总体信息，是对原始信号数据的粗粒度估计。而其余的系数则包含着数据局部的细节信息。因此使用小波变换压缩数据，就是通过只保留前几个系数，以大体上逼近原始的数据，达到降维的目的。其代价是损失一些局部的细节信息。这一点跟傅里叶变换进行压缩一样，两者都是有损压缩。近年来，小波变换由于其同时保持了时域和频域的特征（傅里叶只包含了频域特性），在数据压缩、过滤、分析等领域得到广泛应用。

奇异值分解： 尽管奇异值分解 (Singular Value Decomposition, SVD) 方法已经成功应用于图像和多媒体数据的分析中。最近的工作表明，其也可以被应用于时间序列数据的索引和压缩中。相比于前面介绍的几种方法都是对单一轨迹进行压缩，各个轨迹之间相互独立。奇异值分解不是针对单个时间序列对象进行压缩，而是对全局数据进行整体变换，以得到整体数据的压缩结果。

2.3.2 传统在线时间序列压缩算法

离线的轨迹压缩算法可以充分根据轨迹的所有信息来获取较好的压缩效果。但对于实时在线场景，我们无法事先得到完整的轨迹数据，因而需要在线算法来进行实时压缩。相比于离线压缩算法，降低计算开销是在线算法的重要内容，它需要快速确定是否要保留刚刚到来的轨迹点。为此，我们将介绍几种典型的在线算法。

蓄水池算法： 蓄水池算法 (Reservoir Sampling Algorithm) [97] 的思想是为每个时间序列维持一个大小为 R 的缓冲区，并用这 R 个点来近似原始轨迹。由于轨迹数据是持续增加的，无法知道轨迹的最终长度。因而，该算法所要解决的核心问

题就是无放回替换，即当一个点从缓冲区删除掉后，不能再将它取回。它的做法是，首先将前 R 个轨迹点放入缓冲区，当新的轨迹点到来时，判断该点是否需要放入缓冲区。具体地，当第 k 个点到来时 ($k > R$)，该算法以 R/k 的概率判断该点是否需要加入缓冲区。如果判断结果正确，则从现有的轨迹点中随机删除一个点，并将新点加入。该算法的时间复杂度为 $O(R \cdot (1 + \log N/R))$ ，因此效率较高。但它没有考虑轨迹的时间或空间属性，因而近似轨迹无法保留轨迹的原始特征。

滑动窗口算法：与蓄水池算法不同的是，滑动窗口算法 [98, 99] 的思想是维护一个不断增大的缓冲区，并用缓冲区中的轨迹来近似原始轨迹。它的算法过程是，首先选取第一个点作为锚点，用 p_a 来标识，然后将下一个点加入滑动窗口中。当新点 vp_i 加入后使用 $\overline{p_a p_i}$ 来拟合窗口中所有点组成的字轨迹。若 $\overline{p_a p_i}$ 拟合误差小于给定的阈值，则继续加入新点。否则，将 p_a 与 p_{i-1} 之间的点从缓冲区删除，并使用 p_{i-1} 作为新的锚点。该算法能有效保留轨迹的原始时空特征。

基于安全区的算法：以上两个压缩算法没有考虑轨迹中所蕴含的速度和方向等特征，为此 Potamias[100] 等认为，一个轨迹点只要能表示轨迹发生了改变，就应该包含到最终的压缩轨迹中。而如果一个点可以从之前的运动中通过插值或位置预测等方法计算出来，那么就可以将其舍弃。因此，他们认为在轨迹压缩过程中应该考虑速度和方向这些特征，因为这些可以用来预测轨迹的位置。因而中提出了一种阈值导向的采样算法，即基于安全区的算法，来减少轨迹中的冗余数据点。该算法基本思路是使用最后两个轨迹点以及速度及方向预设的阈值来计算一个安全区域，以此判断一个新采集到的轨迹点是否含有重要的信息。如果该点位于安全区域之内，则认为它是多余的，可以被舍弃；如果该点位于安全区域之外，则将其保留在压缩轨迹中，因为此时运动方向或者速度已经发生了改变。

2.3.3 基于路网结构的轨迹压缩算法

上述传统时间序列算法受限于基于欧氏空间，而人、车辆等运动轨迹往往受限于路网结构。简单使用欧氏距离的度量方式，无法有效刻画这些对象的移动行为。

随着城市路网结构数据的完善, 结合路网数据的轨迹压缩方法不断被提出 [101–104]。这些算法往往需要将轨迹数据进行地图匹配预处理, 以将原始轨迹点映射到路网中。这样做的好处有: (1). 轨迹点落在路网结构上更能直观表达位置信息。(2). 连续变动的多个轨迹点可能对应同一个路段, 因而使用路段表达能有效的降低数据存储开销。此外, 一个城市中的路网线段是有限的, 而采集的轨迹点时无限的。使用路网数据表达原始位置, 能有效降低数据表示的复杂度。

Non-materialized 算法文献 [101] 提出了 Non-materialized 算法来压缩数据, 他首先将原始轨迹变成路网序列数据, 接着将上述结果转换成路口的序列数据, 从而压缩轨迹。文献 [102] 提出了基于最短路径和链接相结合的压缩算法。所谓最短路径就是, 如果一条移动轨迹与其起始和终点的最短路径相一致, 那么可以用头尾两条路段来表示整条轨迹。链接的思想是, 假设司机在两个路段间的行驶轨迹都是沿着最小转角的路段行驶, 并将相邻两路段的头尾连接起来拼凑成更长的轨迹。

基于 MDL 模型的算法文献 [103, 104] 把轨迹压缩问题转换成一个最优化的问题, 其目标是寻找一条近似轨迹, 使得压缩率尽可能高, 且该轨迹与原始轨迹相似度尽可能高。文章引入信息论中的最小描述长度模型 (Minimal Description Length, MDL), 来表示最优化压缩率和相似度组成的目标函数。MDL 模型是信息论中的常用模型, 有两个部分组成, 分别是 $L(H)$ 和 $L(D|H)$ 。 $L(H)$ 为假设条件; $L(D|H)$ 为在此假设条件下, 数据的描述情况。在对应到轨迹数据压缩中, $L(H)$ 表示压缩后的轨迹; $L(D|H)$ 表示近似轨迹与原始轨迹之间的误差。根据 MDL 原则, 只要找到一条路径, 使得 $L(H) + L(D|H)$ 最小, 那么这条路径就是满足目标的路径。该算法在高频率的采样数据集和稀疏的路网结构下有较好的效果。此外针对估计分布不均的情况, [105] 利用哈夫曼思想编码思想的轨迹压缩方法, 即出现频率越高的轨迹段编码越长。

2.3.4 语义压缩算法

原始轨迹和路网轨迹尽管能很好地记录移动物体的运动踪迹,但是对于人们理解轨迹的含义却帮助不大。因为人们阅读轨迹时,无法直观了解经纬度坐标代表的意义。因此出现了利用 POI、标志物、路口和路段来表示车辆的行驶过程,人们阅读语义轨迹时,能明确知道轨迹的起点、终点以及行驶经过的路段。

基于事件的压缩算法文献 [106, 107] 提出了仅保存有意义的状态和事件的语义轨迹压缩。它把一条轨迹拆分为若干连续的事件。事件包括行驶的路段,如从边 a 直走到达边 b , 以及方向的改变,如在 b 路口左转到达边 c 。这种描述方法,会用一条边来表示若干个点,以而达到压缩的目的。

基于摘要的压缩算法文献 [108] 针对现有语义压缩方法未考虑速度、方向等问题,提出了利用轨迹摘要数据的压缩框架。在文章中,他们考虑了道路等级、路宽、速度、停车次数和 U 形转弯次数这六个特征表示摘要数据。其框架包含“划分”和“汇总”两个阶段。在划分阶段,根据之前提取的 6 个特征,把轨迹切分为几段子轨迹以使得每段子轨迹内部具有相似的特征,同时子轨迹之间的特征差异较大。在每段内,选取最具代表性的特征来描述这段的移动过程。在汇总阶段,使用每段的特征来描述轨迹形式过程。该通过提取轨迹概要数据,不仅压缩了数据量,而且也便于人们理解轨迹的行为。文献 [109] 介绍了该算法的演示系统,要求输入一条原始轨迹序列,并输出一段文字,用以描述轨迹的行驶特征以及经过的重要位置。

综上所述,语义压缩后得到的轨迹,便于阅读理解且显著地减少了空间开销。但缺点是丢失了具体的经纬度信息,从而不能支持具体的点查询。

2.4 轨迹距离度量

轨迹距离的定义是轨迹挖掘中的核心内容之一。其度量方式可以分为 3 大类:传统时间序列距离、针对轨迹时空特性专门设计的距离(简称为时空轨迹距离)、以及针对轨迹语义属性设计的距离(简称为语义轨迹距离)。下面我们将介绍每一类的经典度量方式及其特点。

2.4.1 传统时间序列距离

传统时间序列距离广泛应用于时间序列数据分析上如：语音识别、股票期货等价格分析。其典型距离有 L_p 距离、动态时间弯曲距离和最长公共子串距离。

L_p 及欧氏距离: L_p (闵可夫斯基距离) 距离是 L_1 (曼哈顿距离)、 L_2 (欧氏距离) 以及 L_∞ (切比雪夫距离) 等多个距离的总称。使用这一类距离的前提是, 任意两条轨迹具有相同的长度。在该类距离中欧氏距离是最常用的一个, 其计算方式是对应轨迹点间欧氏距离的累加和。特别的, 给定两条轨迹分别为 $\mathcal{Q} = \{q_1, q_2, \dots, q_n\}$ 和 $\mathcal{P} = \{p_1, p_2, \dots, p_n\}$, 其对应的任意一对点 q_i 和 p_i 间的欧氏距离定义如公式2.3所示。在此基础上, 公式2.3定义了两条轨迹间的欧氏距离。轨迹间欧氏距离是对应点欧氏距离的累加和。拓展到 L_p 距离, 其距离为对应点的 p 范数距离的累加。以欧氏距离为代表的 L_p 距离具有计算简单的优点。但也存在着查询结果易受噪声数据影响的缺点。

$$ED(q_i, p_i) = \|q_i - p_i\| \quad (2.2)$$

$$ED(\mathcal{Q}, \mathcal{P}) = \sum_{i=1}^n ED(q_i, p_i) \quad (2.3)$$

动态时间弯曲距离: 相比于欧氏距离, 动态时间弯曲距离能处理两条轨迹长度不同的情况。假设给定两条轨迹分别为 $\mathcal{Q} = \{q_1, q_2, \dots, q_n\}$ 和 $\mathcal{P} = \{p_1, p_2, \dots, p_m\}$ 它们之间的动态时间弯曲距离可由定义2.4中的方式递归计算。其中 $\mathcal{Q}_{1,n-1}$ 代表轨迹 \mathcal{Q} 从第 1 到第 $n-1$ 个点构成的子轨迹。由于动态时间弯曲距离不满足三角不等式, 因此它不是一个度量准则。其尝试将两条轨迹的点以最小的距离和对应起来。

$$DTW(\mathcal{Q}_{1,n}, \mathcal{P}_{1,m}) = \|q_n - p_m\| + \min \begin{cases} DTW(\mathcal{Q}_{1,n-1}, \mathcal{P}_{1,m-1}) \\ DTW(\mathcal{Q}_{1,n-1}, \mathcal{P}_{1,m}) \\ DTW(\mathcal{Q}_{1,n}, \mathcal{P}_{1,m-1}) \end{cases} \quad (2.4)$$

最长公共子串距离: 最长公共子串距离也不要求两条轨迹具有相同的长度。其定义如公式2.5所示。与前两个距离相比, 最长公共子串距离对于数据含噪声的情况处理结果更加鲁棒。但它需要额外参数 ϵ 来约束两个点空间的相似性。最近提出的 EDR 和 EDP 距离同样也是设计出来处理带噪声的数据, 其处理噪声的效果比最长公共子串距离还好。但由于所使用的广泛性不高, 故不作详细介绍。

$$LCSS(Q_{1,n}, P_{1,m}) = \begin{cases} 0 & \text{if } n = 0 \vee m = 0 \\ 1 + LCSS(Q_{1,n-1}, P_{1,m-1}) & \text{if } \|q_n - p_m\| < \epsilon \\ \max\{LCSS(Q_{1,n-1}, P_{1,m}), \\ LCSS(Q_{1,n}, P_{1,m-1})\} & \text{otherwise} \end{cases} \quad (2.5)$$

2.4.2 时空轨迹距离

相比传统时间序列数据, 轨迹数据独有空间属性。为刻画空间属性的影响, 许多工作提出了新的距离。其中典型的有基于最小包围盒的距离、豪斯多夫距离和弗雷歇距离。

基于最小包围盒距离: 基于最小包围盒距离利用给定轨迹线段间的最小包围盒来快速计算轨迹线段间的距离。假设 B_1 和 B_2 分别代表轨迹线段 L_1 和 L_2 间的最小包围盒。 $D_{min}(B_1, B_2)$ 距离代表 B_1 内任意一轨迹线段与 B_2 内任一轨迹线段间距离的下界。其计算方式如公式2.6所示。

$$D_{min}(B_1, B_2) = \sqrt{(\Delta(B_1.[x_l, x_u], B_2.[x_l, x_u]))^2 + (\Delta(B_1.[y_l, y_u], B_2.[y_l, y_u]))^2} \quad (2.6)$$

其中两个区间的距离定义如下:

$$\Delta([l_1, u_1], [l_2, u_2]) = \begin{cases} 0 & \text{if } [l_1, u_1] \cap [l_2, u_2] \neq \emptyset \\ l_2 - u_1 & \text{if } u_1 < l_2 \\ l_1 - u_2 & \text{if } u_2 < u_1 \end{cases} \quad (2.7)$$

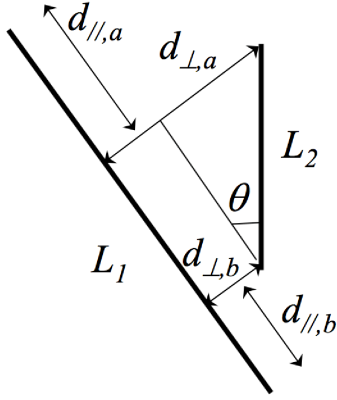


图 2.1: 豪斯多夫距离

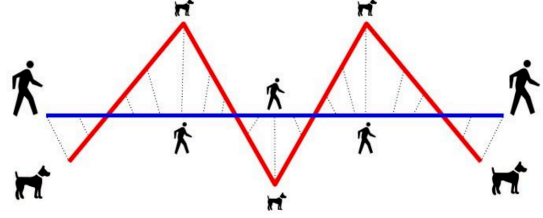


图 2.2: 弗雷歇距离

豪斯多夫距离: 豪斯多夫距离是用来衡量两个轨迹段的相似度。给定轨迹段 L_1 和 L_2 ，豪斯多夫距离定义如公式2.8所示，表示为 3 个带权重距离的累加和。其中 d_{\perp} 表示两个轨迹段的垂直距离， d_{\parallel} 代表轨迹段间的平行距离， d_{θ} 代表轨迹段间的夹角距离。这些几何意义如图2.1所示。而 w_{\perp} 、 w_{\parallel} 和 w_{θ} 为上述三个距离在距离的权重，这些权重值可以随着应用场景的不同而调整相应的值。

$$Hausdorff(L_1, L_2) = w_{\perp} \cdot d_{\perp} + w_{\parallel} \cdot d_{\parallel} + w_{\theta} \cdot d_{\theta}$$

$$d_{\perp} = \frac{d_{\perp,a}^2 + d_{\perp,b}^2}{d_{\perp,a} + d_{\perp,b}}, \quad d_{\parallel} = \min\{d_{\parallel,a}, d_{\parallel,b}\}, \quad d_{\theta} = \|L_2\| \cdot \sin \theta \quad (2.8)$$

在上述公式中 $d_{\perp,a}$ 和 $d_{\perp,b}$ 分别表示 L_1 和 L_2 间的垂直距离， $d_{\parallel,a}$ 和 $d_{\parallel,b}$ 分别表示 L_1 和 L_2 间的平行距离， θ 代表两线段间的夹角。

弗雷歇距离: 弗雷歇距离通俗的讲就是狗绳距离。如图2.2所示人和狗的轨迹，两者之间有一条狗绳约束。主人走路径 A，狗走路径 B，各自走完这两条路径过程中所需要的最短狗绳长度就是弗雷歇距离。

2.4.3 语义轨迹距离

语义轨迹的特点是将原始轨迹中的坐标映射到一个语义信息上，如给点标注为“学校”。从而使得原始的坐标序列变为语义信息的序列。现有语义轨迹度量方式的思想就是从语义和空间两个角度来考虑相似性，但它们的实现方式各不相同。

文献 [110] 提出了基于位置分类属性的语义轨迹分析方法。该方法首先将 POI 分成博物馆、医院、学校等若干个类别，然后给原始轨迹的每个点赋予类别属性。最后使用类别属性的序列进行相似度计算。文献 [39] 利用轨迹空间距离和语义距离的比值来作为最终两个轨迹的相似度。其计算方式如公式 2.9 所示，其中 $geoDist$ 代表了两条轨迹的空间距离， $semRatio$ 代表了两条轨迹间的语义距离。其空间距离考虑了轨迹中心点间的距离、轨迹长度的差别和轨迹间的角度。其语义距离使用最长公共子串距离除以短轨迹长度来表示。 α 用于调节着空间和语义距离的比重。

$$totalDist(t_1, t_2) = geoDist(t_1, t_2) \cdot \frac{1}{1 + \alpha \cdot semRatio(t_1, t_2)} \quad (2.9)$$

文献 [41, 111, 112] 给空间距离和语义距离以不同权重，并用它们的权重和作为轨迹或轨迹点的相似度。其定义形式如公式 2.10 所示。其中文献 [111] 中要求被查询轨迹要完全包含待查询轨迹的所有语义信息。文献 [112] 不是给每个点一个标签属性，而是给每条轨迹一个标签信息。文献 [41] 提出了快速求解近似查询结果的方法。

$$totalDist(t_1, t_2) = \alpha \cot geoDist(t_1, t_2) + (1 - \alpha) \cdot semRatio(t_1, t_2) \quad (2.10)$$

2.5 k 近邻轨迹查询

关于 k 近邻轨迹查询的研究工作有很多，我们将这些工作分为两大类：集中式环境下查询和分布式环境下查询，并分别作介绍。

2.5.1 集中式环境下查询

轨迹数据上的 k 近邻查询，已经得到广泛研究。我们将这些工作分为：历史数据上的查询，轨迹流上的连续查询以及不确定轨迹上的查询。

历史数据上的查询

集中式邻轨迹查询算法也得到广泛研究。Agrawal 等首先使用离散傅里叶变换来转换轨迹数据并保留最开始的几个频率作为轨迹特征 [48]。原始轨迹上的 k 近

邻查询，可通过先在特征值上的查询结果来进行剪枝，以提高查询效率。**Refiei** 等使用傅里叶描述子来表示轨迹形状的边界，接着对每个形状计算出多维指纹信息并对指纹数据构建 **R** 树索引 [49]。最后使用指纹间的距离来近似原始轨迹间的距离。该方法计算出来的距离不受位置，方向和轨迹起始点的影响。**Pelekis** 等针对仅考虑空间维度和同时考虑时空维度两个方面定义了轨迹相似度，并分别提出了处理方法 [50]。此外，他们还提出了考虑速度、加速度和方向的方法。

进一步地，**Frentzos** 等提出了基于 **R** 树索引的查询算法用以处理近邻轨迹查询，并比较了当使用不同类型 **R** 树索引对查询效率的影响 [44]。**Ralf** 等提出了基于剪枝-提炼的解决方案 [45]。其方法在剪枝阶段其同样利用 **R** 树索引来进行剪枝。具体地，其为树的每个叶子节点设计一个时间独立且收敛的覆盖函数，并在查询过程中通过计算查询对象与节点包围盒的距离更新该函数值。最终返回满足查询条件的轨迹段。在提炼阶段，将上一步获得的轨迹段按照到达时间进行排序并合成完整的轨迹。文献 [46] 提出了最近邻轨迹预测问题，其目标是预测接下来一段时间内，与给定查询轨迹最相似的轨迹。其使用 **kd** 树和 **B+** 树来构建对偶索引以提高预测速度。**Ranu** 等针对轨迹数据的异频采样和无固定采样频率的问题，设计了新的轨迹距离计算方法 **EDwP**，并根据该距离设计了新的索引策略用于剪枝查询结果 [38]。**Chen** 等研究了 k 最优连接轨迹查询问题 [47]。其目标是针对给定的一组有序或无序位置信息，找出 k 条满足其采样子轨迹与给定查询最匹配的轨迹。该工作首先给出了满足查询条件的轨迹距离定义方式，然后基于点最优匹配的查询算法。其算法针对索引树，提出了基于查询开销的最优优先和深度优先两种查询策略，并比较验证了两种策略的性能。

此外，还有一些基于距离上、下界的优化方案。**Skoumas** 等使用豪斯多夫距离来计算轨迹段的相似度，并用轨迹段的累加值作为最终轨迹间的相似度 [51]。为此，其首先提出了一个基于优先队列的剪枝算法。该算法通过前几个时间戳上的数据来构建相似度上、下界，并维护一个上界的优先堆（大堆），当堆顶元素下界大于次小元素的上界时，则该元素加入最终的结果集并从堆中移除。否则，不段对

堆顶元素增加更多时间戳内地数据以更新上、下界。该算法直到找出 k 个结果才会停止。该方法，虽然能找出精确的结果但也存在着迭代次数多的问题。为此，提出了两个近似算法以提高查询效率。第一个是基于简化轨迹的算法。其首先使用文献 [52] 所提技术简化轨迹，简化后的轨迹近保留有限个时间戳上的数据。然后对简化轨迹调用文献 [51] 中剪枝算法实现查询。该方法虽然会导致部分结果不正确，但能大大降低计算开销。第二个是基于先验概率的加速算法。其首先计算出轨迹段的分布概率，接着在计算轨迹段距离上、下界的过程中引入该分布。使得包含先验概率高的轨迹段的轨迹得到优先扩展。

目前，还有一些针对语义轨迹的 k 近邻查询研究 [110, 111, 113]。Xiao 等提出了基于语义轨迹的 k 近邻查询 [110]。其首先将原始的位置序列轨迹通过停留区查找算法变成停留区序列 (即语义轨迹)。接着对语义轨迹提出了匹配的定义并基于次给出两条语义轨迹的距离度量。最后，将语义轨迹间距离度量的问题转化为图的构建问题。Zheng 等提出了 k 近邻活动轨迹的查询研究。其目标是给定一组位置序列，每个位置带有多种活动标签 (这样的序列称为活动轨迹) [111]。其目标是从用户活动轨迹数据集中找出与给定序列距匹配度最高的 k 条。为此，其分别定义了基于点和轨迹的匹配度量方式，接着给出了匹配距离的下界并证明了通过下界能快速剪枝候选。最后给出了两种算法以分别处理查询点集合有序和无序的场景。该工作所提查询要求返回的轨迹必须完全包含所有待查询活动标签的类别，只要有一个标签不包含，则认为返回的轨迹与查询不相似 (相似度为 0)。Wang 等在此基础上进行了改进，认为一条轨迹包含待查询轨迹内容标签越多，则相似度越高 [113]。此外，该工作还给每个位置上的标签赋予一定的权重以满足不同用户的需求。为解决所提查询，他们首先提出了相似度上、下界的增量式算法。但该方法存在着迭代次数多的问题，为此提出了动态扩展的策略以给每个查询点以不同的扩展范围。进一步地，为解决每次扩展中需要对索引树进行深度遍历的问题，提出了两层阈值算法，以保证一次搜索就能找出满足条件的候选。

轨迹流上的连续查询

Sacharidis 等研究了轨迹数据流上的 k 近邻轨迹查询 [114]。该工作使用滑动窗口来处理轨迹数据流, 并使用窗口内每个时段点的距离的聚集值 (最大、最小、累加、均值等) 作为轨迹相似度。其首先提出了基于事情模型驱动的通用处理方法 BSL。该方法关注如下三类事件: (1) 待查询移动对象的位置变化 (Query location updates, $QUpd$), (2) 被查询移动对象位置的变化 (Object location updates, $OUpd$) 含新位置到达和旧位置过期, (3) 移动对象与被查询移动对象位置过期 (Object distance expiration, $OExp$)。BSL 为查询对象保留最后时刻的位置, 为其他移动对象保持最后的位置和窗口内的距离序列 (每个时间戳对应一个其余查询对象的距离)。此外还维护了一个事件列表以便及时删除过期的位置信息。当事件某一类型事件到达时, 按照其设计的方案更新轨迹间的距离。接着, 提出了改进算法 XTR 用以去出无用点 (不会影响到最终轨迹间距离值的点) 以减少遍历时间。最后, 提出了基于移动对象最大以速度改进算法 HRZ。该方法利用最大移动速度约束, 以剪枝候选。XTR 和 HRZ 这两个改进方法只适用于使用最大和最小距离作为轨迹距离的情况。

此外, Bakalov 等研究了轨迹数据流上的连续 join 查询 [115]。其目标是针对两个不断变化的轨迹数据集, 数据集间的轨迹进行两两相似度匹配并返回最相似的, 且相似度由轨迹空间邻近度决定。其所提解决方案首先使用轨迹近似技术来降低数据的维度并对近似轨迹构建空间索引。接着对近似轨迹设计了距离下界用以进行剪枝。针对连续查询, 设计了两阶段处理方法。第一阶段根据当前窗口内的数据初始化查询结果。在该阶段使用索引来降低匹配的个数并利用下界得到近似结果。第二阶段针对数据的变换, 持续检查结果集并作出更新。

不确定轨迹上的查询

不确定数据流上的概率相似度查询上的查询已经被广泛研究 [116, 117]。这类工作首先提出概率距离的计算方式, 其需要设置两个阈值参数。一个是距离阈值 γ 另一个是概率阈值 τ 。其查询目标是返回与查询轨迹距离小于 γ 的概率超过 τ 的轨迹。因此, 查询结果对这两个阈值比较敏感。阈值作轻微改变可能查询结果的大小和内容发生较大变化。

此外, 一些专门针对不确定轨迹数据上的查询也被提出 [118–120]。文献 [118, 119] 研究了给定不确定轨迹数据集上的连续近邻查询。其假设查询与被查询的轨迹都是不确定的, 但轨迹在给定空间范围内满足一定的概率分布。其目标是不断返回与待查询轨迹概率距离最高的 k 条轨迹, 并找出结果集内容或顺序发生变化的时间点。其主要思想是构建一棵几何对偶的索引树以剪枝候选。Ma 等研究了不确定轨迹数据上的 k 近邻轨迹查询 [120]。该工作与文献 [118] 的研究内容差别是其给定的查询轨迹是确定的。其假设所处理的轨迹数据位置具有一定的误差, 且误差满足一定的概率分布。为度量两条不确定轨迹的相似度, 其首先提出了 p 距离。给定轨迹 x 与查询 Q 的 p 距离由其他比 x 离 Q 更近的轨迹的概率距离和表示。即 x 离 Q 越近, 越少的轨迹比 x 更近。接着提出了基于空间网格划分的 *UTgrid* 索引, 并对每个网格内的轨迹构根据时间维度构建 1 维 R 树索引。最后设计了针对不确定轨迹的查询算法, 该算法利用 *UTgrid* 进行剪枝以达到提高查询效率。与以上工作不同地是, Li 等还研究了基于路网的不确定轨迹查询 [121]。该工作中假设移动对象在路网中的移动速度未知。因此如何预测哪些轨迹会成为将来的候选集成为难点。

2.5.2 分布式环境下查询

为处理轨迹大数据, 分布式环境下的 k 近邻轨迹查询得到越来越多的研究。我们将从分布式轨迹数据查询系统和分布式近邻轨迹查询两方面进行介绍。

分布式近邻轨迹查询

目前, 针对现有不同并行计算框架设计时间序列相似性查询算法的工作。文献 [122] 研究了基于 Map/Reduce 框架的集合数据的 join 查询问题, 其目标是减少数据节点间的通信问题。文献 [122] 针对 Map/Reduce 框架下数据集的 top- k 连接查询问题进行了研究。其首先提出了基于分治法和分支限界法的两种实现策略。接着提出了全部配对划分和有限配对划分的数据划分策略以减少 map 和 reduce 任务间的通信开销。此外, 还有针对通用计算图形处理器 (GPGPUs) 并行计算框架的

工作 [123–125]。其中, 文献 [123] 研究了范围查询即找出与查询轨迹距离小于给定阈值的轨迹。其提出了新的适用于 GPU 的索引结构有一替代传统的 R 树索引。给定一个查询集合, 它提出了多个方法以将查询分成若干互相独立的批次以得到更好的内容命中率和降低计算的开销。文献 [124] 构建了针对轨迹数据的管理系统并研究了基于豪斯多夫距离的轨迹相似度 join 查询。其主要贡献在于设计了四层轨迹数据表现方式并设计了一个 3 层所以架构用以提高查询效率。Leal 等 [125] 首先提出了基于豪斯多夫距离的 k 近邻轨迹查询算法 TKSIMGPU, 其主要研究目标是使得各个计算线程负载均衡。Top-KaBT[126] 是 TKSIMGPU 的改进版本, 其设计了轨迹间豪斯多夫距离上、下界, 用以减少需要计算精确距离的候选数。

针对协调者-远程结点分布式架构, 一些轨迹序列 top- k 查询算法已经被提出 [31, 32, 127]。文献 [127] 针对手机信令轨迹数据进行了 k 近邻研究。在该工作应用场景中, 每个手机基站被看作远程结点, 用户在移动过程中会经过不同的基站, 因而其轨迹数据会分成若干段并保留在不同的远程节点上。为处理针对这一场景下的查询, 其提出了在每个远程结点计算子轨迹距离的上、下界, 以提高查询的效率。Smart Trace[32] 和 Smart Trace+[31] 两个系统针对众包场景下的 k 近邻查询进行了研究。其这两个系统中, 每个用户的手机被看作远程结点, 且每个远程结点仅包含一条轨迹数据。针对这一场景, Smart Trace 提出了基于最长公共子串距离的上界用以查询时间和能量消耗。进一步地, Smart Trace+ 还提出了新的下界并提出了结合上、下界的处理算法。以上工作均是以提高查询效率为首要目标, 并未深入分析所提算法在通信问题上的开销。

为解决分布式 k 近邻轨迹查询中通信开销过大的问题, 基于多粒度概要数据的方法已经被提出 [29, 30, 30, 64, 128]。Papadopoulos 等 [128] 首先对分布式时间序列进行了 k 近邻查询研究, 并提出了 4 种基于协调者-远程结点的方案以求在提高时间效率的同时降低通信开销。但该方法仅能降低从远程结点返回给协调者结点的数据开销, 协调者结点仍需要将待查询时间序列发送给所有远程结点, 而该开销是所有算法通信开销的主要部分。因此, 其并未彻底解决通信开销过大的问

题。Yeh 等 [64] 提出了用于降低通信开销的算法 **LEEWAVE**。其使用小波变换的数据压缩放，将原始时间序列数据变换成多粒度的概要数据。**LEEWAVE** 算法中协调者结点将概要数据由粗到细发送给远程结点，远程结点根据概要数据计算候选轨迹与查询轨迹欧氏距离范围的必要参数，并将参数发送给协调者结点用以计算距离上、下界并进行剪枝。其根据多粒度数据不断缩小距离范围的方法能大大降低通信开销。文献 [29, 129] 提出了更紧的下界，因而能打到更好的剪枝效果。文献 [29, 64, 129] 都是基于哈尔小波变换的压缩方式，而该方式仅适用于距离度量方式是欧氏距离的场景。为此，文献 [30] 提出了针对动态时间弯曲距离的算法，该算法引入多粒度包围信封以作为概要数据，并根据该概要数据计算距离下界。此外，使用级联下界的方式以进一步降低计算开销。而文献 [30] 在其基础上提出了新的上界，并引入了同时根据上、下界进行剪枝的算法 **MDTK**。**MDTK** 算法引入了边剪枝边过滤的思想用以加快下界的计算速度。以上工作都是针对某一具体距离度量方法提出相应的基于概要数据的距离上、下界，并未考虑如何扩展到其他轨迹距离的度量方式上。为此，亟需提出通用的方法，以解决分布式 k 近邻轨迹查询。

第三章 基于集群的分布式轨迹管理

本章主要介绍如何在集群内构建轨迹数据管理系统并提供实时的查询结果。首先，章节3.1介绍了背景知识。其次，章节3.2介绍了本文设计的系统架构及系统的实现。然后，章节3.3实验验证了系统的性能。最后，小结本章的研究内容。

3.1 引言

3.1.1 背景知识

基于分布式集群的数据管理和分析方法是当前解决大数据问题的首选方案。分布式集群内的存储和计算方案通常采用主从式架构，即主节点负责管理数据的元数据和任务的分发，从节点负责存储数据的一部分并负责对这一部分数据进行分析。Hadoop 和 Spark 是基于主从式架构设计的当前最流行的大数据处理系统。这两个系统的最大区别是 Hadoop 是基于磁盘的数据处理方式。每当有新的查询或分析任务产生时，存储在磁盘上的数据需要先被读取到内存上再做分析。因此，基于 Hadoop 构建的轨迹数据管理系统如 Truster [57] 和 Clost [16] 等都面临着 I/O 开销过大问题，因而无法提供实时的查询分析结果。而 Spark 是基于内存的处理方式，在进行数据分析前，所有数据可以被加载进入集群的内存中 (Spark 通过 off-heap 技术处理数据内存放不下的情况)。新的查询过来后，可以直接从内存里面查询数据，有效地避免了 I/O 开销。因此，Spark 成为当前提供快速数据分析的主要工具。

鉴于 Spark 的高效性能，本文考虑使用这一系统来进行轨迹数据管理和查询实现。在此之前，我们将首先对 Spark 作一个概括介绍。Spark 是一个通用的并得到广泛使用的用于处理大数据的集群计算引擎。其提供了丰富的 Scala、Java 和 Python 调用 API 接口，以方便用户操作数据。该系统一经提出，围绕其展开的用于进行内存数据分析的生态圈系统便得到广泛关注。这些系统提供了丰富的数据处理功能，包含数据仓库 (Spark SQL)、流数据处理 (Spark Streaming)、图数据处理 (GraphX)

和机器学习 (MLlib)。

Spark 提供了一个高效的基于分布式集群内存的数据抽象：弹性分布式数据集 (Resilient Distributed Dataset, RDD)。每个 RDD 可以看做是一个 Java 或 Python 中的对象集合，这些对象按照一定的方式分布在集群内的机器上。用户可以通过函数式编程 API（如 `map`、`filter`、`reduce`）来实现对数据的操作。RDD 具有较好的容错性，因其可以通过依赖图以重新计算部分算子以恢复丢失的分区内的数据。所有的 RDD 可以既可以被缓存到内存中又可以被显示的保持到磁盘中以加速数据的重复利用并支持迭代计算。RDD 上的算子可以分为两类：一类是转换 (Transform)，其操作在一个 RDD 上得到另一个 RDD，如 `map`、`filter`、`union` 等。另一类就是行动 (Action)，其返回结果或是把结果持久化起来，如 `count`、`collect`、`save`。其中转换类算子采用的是懒策略，即其提交后不会立马执行，而是等到行动算子被提交时才会触发。

3.1.2 基于 Spark 的时空管理系统

目前，已经出现了些利用 Spark 构建时空数据管理系统。本小节将对这些系统作简要介绍。

GeoSpark.GeoSpark 扩展了 Spark 以处理空间数据。它提供了新的适用于空间数据的抽象模型 SRDDs 以及对空间对象的操作算子。GeoSpark 提供空间范围查询、 k 近邻查询和空间 join 查询。除此之外，它允许在每个数据分区提供一个局部索引（四叉树和 R 树）。值得注意的是，Spark 允许开发者构建任意对象类型的 RDD（如关于点、多边形和圆等对象）并提供对应的空间操作算子。尽管 GeoSpark 能够使用局部索引提高查询速度，但其不支持全局索引。此外，它支持 2 维空间数据，不能处理带有额外属性的空间对象（如带有用于描述信息的对象）。

SpatialSpark.SpatialSpark 在 Spark 之上实现了一些用于分析空间数据的接口。特别地，SpatialSpark 提供了空间范围查询和 join 查询（使用如“相交”和“包含”作为约束）的接口。其使用一个确定的 kd 树或网格索引来划分数据和并提高查询

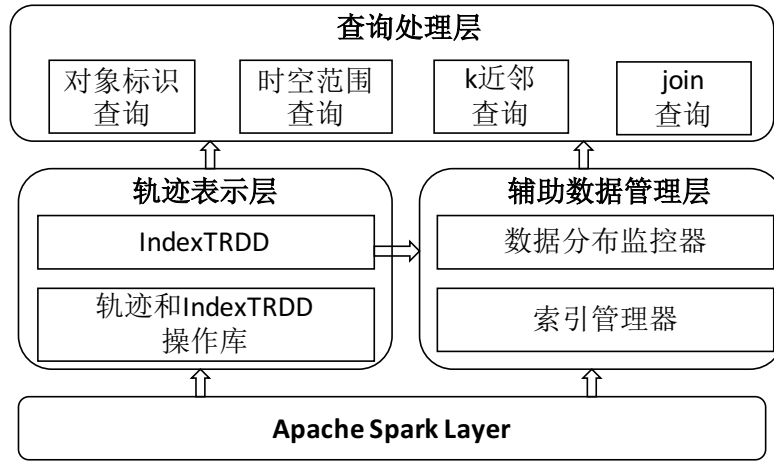


图 3.1: 剪枝示例

效率。尽管如此，SpatialSpark 只提供 2 维数据上的操作，且不支持对 RDD 构建索引。

Simba. Simba 扩展了 SparkSQL 以提供类 SQL 空间查询。特别地，它提出了 IndexRDD 以管理多维空间数据。IndexRDD 在每个分区包含一个局部空间索引，并为所有分区构建了一个全局索引。Simba 提供了空间范围查询、 k 近邻查询和空间 join 查询等查询接口。为进一步提高查询效率，Simba 从逻辑层（查询解析）和物理层（查询计划）两方面进行了优化。相比于前面两个系统，Simba 能处理多维数据。

注意. 上述系统都是针对空间对象（点、线、面）数据进行管理。而轨迹数据除了空间维度还有时间维度以及移动对象标识这两个重要属性。这使得我们在存储数据时，不但要把空间靠的近的轨迹放到一起，还要把时间邻近的以及同一个移动对象的数据放到一起。

3.2 TrajSpark 系统设计及实现

本节将介绍在 Spark 之上构建的轨迹数据管理系统 TrajSpark。

3.2.1 系统架构

TrajSpark 系统架构如图3.1所示, 其由如下四层构成: (1) **Apache Spark 层**, 该层由 Apache Spark 提供的 RDD 及其常用操作构成; (2) **轨迹数据表示层**, 该层介绍了对轨迹数据的抽象表示结构 **IndexTRDD**; (3) **辅助数据层**, 该层监控管理数据添加导致的数据分布变化, 并负责指导对新数据的划分; (4) **查询处理层**, 该层对三种典型轨迹查询进行了算法设计。具体地:

- **Apache Spark 层:** 该层是整个系统的基础, 其继承自 Apache Spark, 故不作详细介绍。
- **数据表示层:** 在该层, 我们首先为轨迹设计了结合列存储形式的轨迹表示方法以达到压缩数据的目的。在 TrajSpark 系统中, 轨迹以轨迹片段的形式存储, 一条长轨迹将被切分成多条轨迹片段, 而时空相邻的轨迹片段将被放到同样的数据分区中。为此, 我们设计了基于轨迹片段的数据表示结构 **IndexTRDD**, 并提供了丰富的轨迹查询算子。为提高轨迹查询效率, TrajSpark 为 **IndexTRDD** 维护了全局和局部两层轨迹索引。其中全局索引维护了所有分区的时空信息, 而局部索引维护了每个分区内部轨迹的信息。该层是整个系统的核心。
- **辅助数据层:** 这一层维护了两种系统的全局信息。第一种就是轨迹数据的分布信息, 其由数据分布监控器 (**Data Distributor Monitor**) 管理。随着数据不断装载入系统中, 数据的分布会随之发生变化。TrajSpark 通过监控这一变化并设计合理的数据划分方式以达到数据均衡的目的。第二种就是全局索引, 其由索引管理者 (**Index Manager**) 维护。索引管理者运行在主节点上, 用以管理数据的全局索引。当数据加载进后, 索引管理这将更新全局索引。
- **查询处理层:** 这一层主要实现了两类基本轨迹查询以及轨迹 k 近邻查询。此外, 用户可以通过基本函数接口实现更加复杂的查询。

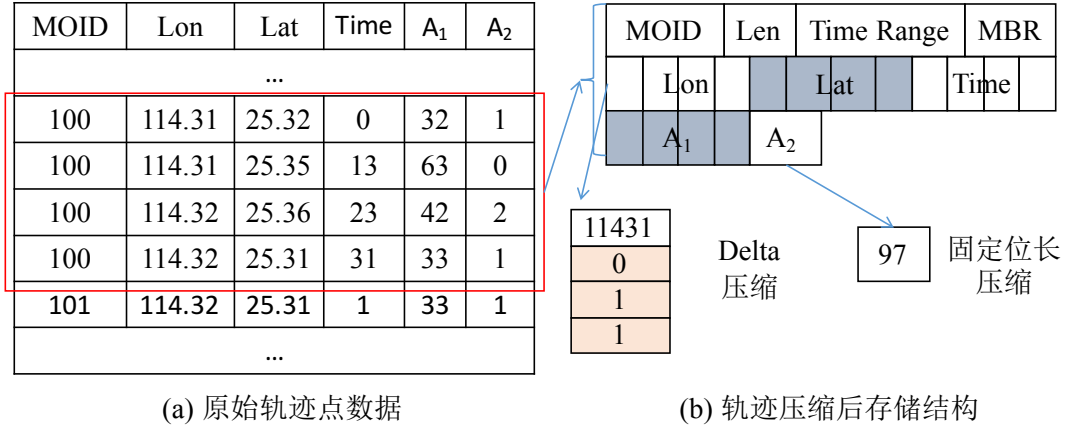


图 3.2: 轨迹片段表示

3.2.2 数据表示层

轨迹片段表示

原始来自轨迹 GPS 日志中的轨迹数据往往以轨迹点集的形式保存。因而，原始信息可以看作包含如下属性的表格 ($MOID, Location, Time, A_1, \dots, A_n$)。其中 $MOID$ 是移动对象的标识, $Time$ 记录了轨迹点采集的时间、 $Location$ 代表轨迹点的位置。其他属性根据数据源的不同而变化。基于 Spark 的一个简单轨迹数据管理实现方案就是将轨迹数据看作由轨迹点构成的 RDD。但该方案会导致数据的冗余, 比如一个移动对象出现 100 次, 则其标识将会重复 100 次。还有当连续采集的过程中, 移动对象位置没有发生变化时, 该位置仍然会被保存多次。除了数据冗余外, 文献 [54] 表明基于轨迹片段的表示方式相比基于点的方式能显著提高查询效率。

TrajSpark 采用基于轨迹片段的方式管理轨迹数据。其将时空相邻的轨迹点聚集到统一数据分区中。此时, 一条长轨迹可能被分割成若干段并存储在不同的分区上。在每一个分区内部, 可以使用基于列存储的数据表达方式并结合数据压缩手段。实现这一目的存在两点挑战: (1) 如何为非结构化的轨迹数据设计列存储的数据结构; (2) 如何对列存储的轨迹数据进行压缩且不损害数据查询的效率。

TrajSpark 的实现方法是: 在每一个分区内, 先将所有轨迹点按照移动对象标识进行分组。接着属于同一移动对象的轨迹点按时间顺序进行排列, 并得到一个轨

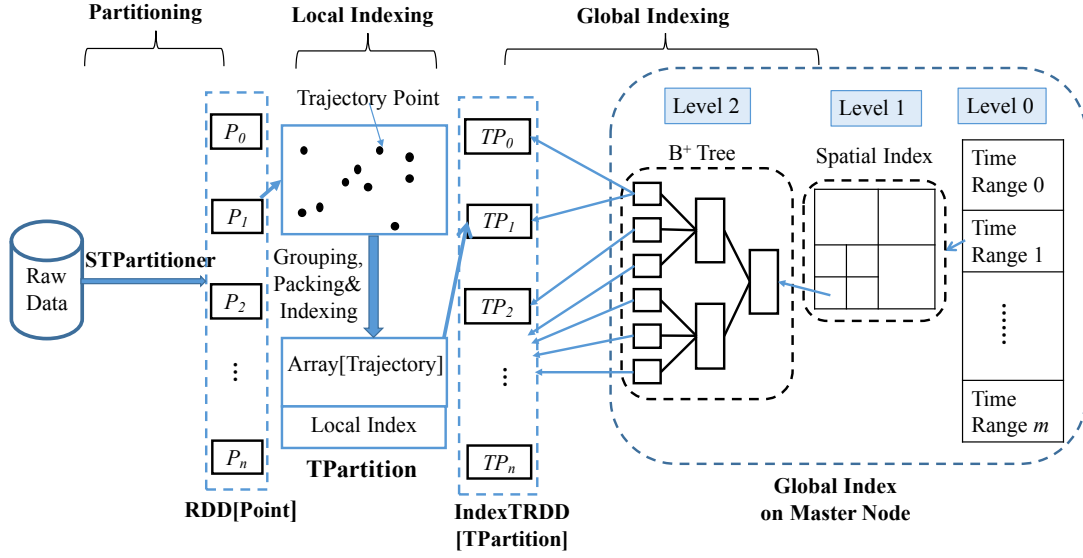


图 3.3: 轨迹索引构建

迹片段。接着，将对每一个轨迹片段按图3.2(b)所设计的表示方式进行压缩处理。在这种方式中，同一属性的值将被连续存放并压缩存储。其中针对数值型熟悉，如坐标位置等，使用差值压缩；对于枚举类型属性，使用固定为压缩。如图中 A_2 属性，其只有 3 个可能性的值即 0、1 和 2。此时我们只需使用两个二进制位就能完成数据的存储。对于字符串属性，使用 *gzip* 进行压缩。压缩完所有数值后，我们为一条轨迹保留额外的概要数据，概要数据包含移动对象的标识、轨迹长度和（时空）最小包围盒信息。使用概要数据能对查询进行有效剪枝。最终，分区内轨迹点被转换为压缩好的轨迹片段，并使得整个轨迹数据集成为以轨迹片段构成的 RDD（称之为 TRDD）。

轨迹片段索引

前面，我们介绍了如何将原始的 GPS 记录转换成基于轨迹片段的 TRDD。但 TRDD 对于给定的任意查询需要遍历整个数据集才能找到结果，因而 CPU 开销较大。为此，我们需要引入索引结构以提高查询效率。TrajSpark 提出了 IndexTRDD 这一结构以支持索引。IndexTRDD 首先通过在 TRDD 每个分区内引入针对轨迹标识的局部哈希索引，从而使得针对给定轨迹的查询复杂度降为常数级。进一步地，我们在 IndexTRDD 的分区之上构建了全局索引，以便查询执行时对不必要

的数据分区进行剪枝。图3.3介绍了构建索引的整个数据流程，其包含：数据划分 (*Partitioning*)，局部索引构建 (*Local Indexing*) 和全局索引构建 (*Global Indexing*) 三个阶段。接下来将对这三阶段进行详细介绍。

数据划分：该阶段，TrajSpark 将原始轨迹日志数据从磁盘装载进入磁盘形成一包含轨迹点对象的 RDD。接着，我们需要对其进行数据划分以达到如下效果：(1) 数据局部性，即时空相邻的轨迹点需要划分到同一个数据分区中；(2) 负载均衡，各数据分区的数据量尽量相等；(3) 分区大小约束，每个分区数据大小要适中，以避免内存过载。尽管 Spark 本身提供了范围和哈希两种数据划分方法，但它们都是针对一维的数据进行划分。而轨迹是多维数据，这两种方式无法简单应用到轨迹划分中。为此，我们设计了新的针对轨迹的划分方法 STPartitioner，该方法首先使用一颗四叉树或 kd 树来划分轨迹点。该索引树从原始轨迹的空间分布学习出来，以保证树的叶子节点所包含的轨迹点数量比较接近。当索引树构建好后，STPartitioner 使用叶节点的边界进行数据划分。划分后，同一时空范围内的点被聚集到一起。为了保证每个分区大小适中，TrajSpark 将聚集在一起的轨迹点根据移动对象标识或时间划分成若干大小相近的数据分区。

局部索引构建：经过数据划分后，轨迹数据集可看作是一个按时空划分好的点对象构成的 RDD。在本阶段，我们首先将此 RDD 经过分组、排序和包裹以构建成以轨迹片段组织的 TRDD。接着，为每个数据分区的头部添加一个局部哈希索引，用于将移动对象标识跟对应的轨迹片段映射起来。我们将这种包含索引及轨迹片段的分区称为 TPartition。此时，整个数据集可看作由 TPartition 构成的 RDD。我们称此时的 RDD 为 IndexTRDD。最后，主节点收集每个分区的标识（每个数据分区都有一个特别的标识符）、分区所含数据的空间范围和时间范围等信息以便构建全局索引。

全局索引构建：本阶段，根据收集到的分区信息在主节点构建全局索引。全局索引的格式如图3.3所示，是一个三层混合索引。数据首先根据其时间维度信息被第 0 层粗粒度时间范围所划分。同时，每个粗粒度时间范围对应着一个第 1 维空

间索引 (STPartitioner 就是根据这个空间索引对该时间范围内数据进行空间划分)。每个叶子节点所对应的空间范围可能包含若干数据分区, 我们使用第 2 层 B+ 树索引对这些分区根据时间维度进行管理。当第一批数据加载入系统后, 第 0 层索引只包含一个时间值 (时间段的起始时间), 第 1 层空间索引即为数据划分所用的索引。我们收集每个分区的时间和空间信息以构建第 2 层索引。TrajSpark 将全局索引保存在主节点的内存中并维护该索引的更新。即使当轨迹数据量很大时, 我们的全局索引仍然只需消耗较少的存储空间, 因而能以较低的内存开销放在主节点上。

3.2.3 辅助数据层

数据分布管理器

在真实的应用场景中, 轨迹数据集会随着数据周期性的加载而不断变大 [59], 从而导致数据的空间分布也发生变化。如何对新来的数据进行有效划分成为新的难点。一方面如果始终使用一个静态数据划分方式来划分, 这样划分的开销虽低, 但会导致数据不均衡从而影响查询的效率。另一方面, 如果将新来的和已有的数据一起重新划分以得到均衡的结果 (方法被文献 [60–63] 采用), 但这样的划分开销太大 (主要在数据 Shuffle 上)。并且重新划分已有的数据没有太大的必要, 因为新数据的价值总是大于旧数据的, 且用户的查询往往也是集中在新数据上。所以当一批新数据到来时, TrajSpark 尝试只均衡划分新数据而不改变已有数据的划分。这一点与 AQWA 系统 [59] (需要对新的和已有的数据进行划分) 有所区别。此外, TrajSpark 只关注长时间段内的数据分布的变化, 而尽量减轻短时间内数据分布的变化对数据划分的影响。比如周末与平时城市的轨迹分布会大不相同, 但我们不应该应该这样的不同而对数据划分作太大变化。

基于以上考虑, 数据分布管理器 (Data Distribuion Monitor) 采用了时间衰减模型来监控数据分布的变化。该模型中给新来的数据更高的权重。具体做法是: 首先, 将整个空间维度划分成 $m * m$ 个细粒度的格子, 统计数每个格子内轨迹点的个数。当一批新数据到达时, 数据分布管理器维护了两个矩阵: $A_{existing}$ 和 A_{new} ,

用以分别记录已加载和新到达数据的分布。其中 A_{new} 为新数据在每个格子内点的个数除以总的点的个数得到。当新数据达到后, $A_{existing}$ 通过将其每个值除以 γ (γ 称为衰减因子, 其给已有数据更低的权重) 以降低自身的权重。然后, A_{new} 加入到 $A_{existing}$ 中, 并将其值赋为 0。因此, 每当新数据加入后, 早期加入的数据权重会不断降低。为了更好的描述 $A_{existing}$ 的变化过程, 我们使用 $A_{existing}^n$ 来表示当地 n 次数据加入后数据的分布。

为自适应的调整数据划分策略, 我们使用矩阵 PA_c 来描述当前划分方式所依赖的数据分布, 其用 $A_{existing}^0$ 来初始化。STPartitioner 就是从 PA_c 中学习出数据划分的索引并将数据按照空间维度进行划分。当第 n 批数据到达后, PA_c 与 $A_{existing}^n$ 的分布差别超过一个给定的阈值, 则认为数据发生了较大变化。此时, 需要更新数据划分策略。我们的做法是将 $A_{existing}^n$ 的值赋予给 PA_c , 并从 PA_c 中学出新的索引用以替换 STPartitioner 中用于数据划分的索引。在 TrajSpark, 我们使用 JSD 距离来度量两个分布的差别 (两个分布均需归一化后才能计算)。本文使用的时间窗口衰减模型具有延迟更新特性, 能很好的应对数据短时间内发生较大变化的情况。

索引管理器

索引管理器 (Index Manager) 主要维护索引的更新和持久化。在 Spark 中存在着两种情况会导致全局索引的变化。第一种情况是当 STPartitioner 更新了其用于数据划分的空间索引。在这种情况下, 我们会在全局索引的第 0 层的最后添加新的时间范围起点, 同时为该范围添加对应的第 1 层空间索引。第二种情况是当新到达数据的所有分区加入到 IndexTRDD 中后, 这些分区所对应的信息也会被添加到全局索引中。索引管理器将全局索引存储在主节点内存上。此外, 它可以选择将索引持久化到文件系统中也能将其再次从文件读入到内存中。这使得系统能够在出现异常下进行快速恢复。值得注意的是, TrajSpark 对全局索引提供了一系列的操作接口, 如交集运算 (intersect)、范围覆盖计算 (overlap) 等, 以满足用户对全局索引的操作需求。

算法 1 基于对象标识的查询算法**输入:** $moid, tRange$;**输出:** one trajectory;

```

1:  $pids = gIndex.intersect(tRange)$ ;
2:  $ts = IndexTRDD.PartitionPruningRDD(pids)$ 
    $.getTraWithID(moid).mapValues(sub(tRange))$ ;
3: return  $ts.reduceByKey(merge).collect()$ ;

```

3.2.4 查询处理层

在这一层，我们将介绍 TrajSpark 如何处理三种典型的轨迹查：基于给定对象标识的查询、时空范围查询以及 k 近邻查询。

基于对象标识的查询

基于给定对象的查询通过接收待查询轨迹的标识 $moid$ 和时间约束范围 $tRange$ ，以返回该移动对象在指定时间范围内的轨迹。尽管 Spark 可以通过最基本的 filter 算子实现查找，但该方式需要遍历整个数据集，因而 CPU 开销较高。TrajSpark 通过使用内部两层索引机制能进行快速的剪枝。其处理方法基于以下三点观察：(1) 全局索引的第 0 层能够快速剪枝掉不满足时间约束的分区，并且第 2 层时间范围索引能够快速找出与给定时间范围相交的分区；(2) 对于每个数据分区，可以通过局部的哈希索引快速找出指定移动对象的轨迹；(3) 指定对象轨迹的概要数据包含时间范围，利用该范围可以快速判断，该移动对象在指定时间是否有数据。

基于以上观察，我们设计了查询算法1。TrajSpark 首先遍历全局索引找出时间范围与给定时间约束相交的分区（第 1 行）。该操作可以通过全局索引的 intersect 接口实现。接着，IndexTRDD 调用 Spark 系统 API— PartitionPruningRDD，来找出这些分区内容。然后，TrajSpark 通过局部索引找到给定移动对象的轨迹片段，并找出指定时间内的数据（第 2 行）。最后，汇总每个分区所找出的子轨迹片段，并合并成一条完整的轨迹（第 3 行）。TrajSpark 提供了 merge 算子用于合并同一对象的两条轨迹并将结果按时间序排列。

时空范围查询

时空范围查询通过接收时间范围参数 $tRange$ 和空间范围参数 $sRange$ 返回指

算法 2 时空范围查询算法**输入:** $tRange, sRange$;**输出:** a set of trajectories;

```

1:  $pids = gIndex.intersect(tRange, sRange)$ ;
2:  $ts = IndexTRDD.PartitionPruningRDD(pids)$ 
    $.filter(tRange, sRange)$ 
    $.mapValues(sub(tRange, sRange))$ ;
3: return  $ts.reduceByKey(merge).collect()$ ;
```

定时空范围内的轨迹。需要指出的是，这两个查询参数可以不全部给出。通过使用索引，TrajSpark 也能为该查询提供远胜于遍历操作的查询效果。算法2 介绍了该查询的实现方法。其首先通过全局索引找出与给定时空范围相交的分区（第 1 行）。接着，对每个分区内的轨迹通过访问其概要数据以剪枝掉与给定时间范围没有交集的轨迹片段。然后对满足约束的片段进一步找出满足时空约束的子片段（第 2 行）。最后，将满足时空范围内的子片段按照移动对象标识分组，同一个移动对象的子片段进行合并得到完整的轨迹（第 3 行）。

算法 3 k 近邻查询算法**输入:** $IndexTRDD, disM$;**输出:** the k most similar trajectories to tr ;

```

1:  $mbr = tr.MBR, tRange = tr.TimeRange$ ;
2: repeat
3:    $pids = gIndex.intersect(mbr, tRange)$ ;
4:    $ts = IndexTRDD.PartitionPruningRDD(pids)$ 
      $.filter(mbr, tRange)$ 
      $.reduceByKey(merge)$ ;
5:    $mbr.expand(1 + \alpha)$ ;
6: until ( $ts.size > k$ )
7:  $candidate = ts.collect()$ ;
8: return  $candidate.map(t \rightarrow (distance(t, tr), t))$ 
      $.sortByKey.top(k)$ ;
```

 k 近邻查询

k 近邻轨迹查询,即给定查询轨迹,从轨迹数据集中找出与之距离最近的 k 条轨迹。该查询接受两个参数:一个是待查询轨迹 tr ,另一个是距离度量方式 $distance$ 。尽管轨迹距离度量准则有很多,但都遵循着时空范围越接近,则距离越近原则。为

此设计了算法 3 以支持任一距离在 k 近邻查询的使用。首先 **TrajSpark** 获得待查询轨迹的空间包围信封和时间范围（第 1 行）。接着，利用全局索引，找出与查询时空范围相交的分区，然后在每个分区内找出与轨迹时空范围相临交的轨迹片段，并将这些片段按照标识和时间拼接成完整的轨迹（第 2 行）。此时，需要检查拼接后的轨迹数目是否有 k 个，如果不足 k 个，则进一步扩大搜索空间，直到找到超过 k 个为止。其中扩大搜索空间的方法是将搜索范围的长度和宽度同时变为原来的 $1 + \alpha$ ($0 < \alpha < 1$) 倍。最后，对搜集到的轨迹进行真实距离计算，并找出距离最近的 k 条。

Join 查询

轨迹的 **join** 查询被广泛应用在社交网络和推荐系统中。一个典型的查询应用就是，给定一个轨迹数据集和一个带语义信息的空间对象集合（如 POIs、道路和区域），研究者们将这两个数据集关联起来以得到语义轨迹。这样的查询可表示为 **Join**(*IndexTRDD*, *SP*, ϵ)，其中 *IndexTRDD* 代表轨迹数据，*SP* 代表空间对象数据集， ϵ 是查询的空间约束。这个查询将轨迹中的点 p 映射到 *SP* 中距离最近的点 e 上，且需满足 p 到 e 的距离需小于 ϵ 。如果点 p 找不到满足条件的映射点，则会将其忽略。图 3.4 给出了查询示例，给定查询轨迹 $\langle p_1, p_2, p_3 \rangle$ 和由 10 个点对象构成的空间对象集合（由 e_1 到 e_{10} 表示）。根据查询， p_1 不会被映射到任何点上， p_2 会被映射到 e_2 上，因为 e_2 是 ϵ 范围内最近的点， p_3 将会被映射到 e_9 上。

由于轨迹数据集和空间数据集都比较大，暴力查找开销较高。为此，我们需要对 *SP* 数据集进行预处理。由于整个系统空间被划分成了 $m * m$ 个格子，我们首先对 *SP* 数据集统计每个格子里点的个数。假设 *SP* 将被划分成 S 个分区（每个分区对应一个网格，且 S 值由 *SP* 和分区大小约束两方面决定）。然后，我们构建四叉树或 **R** 树来构建索引管理这些格子，使得最终索引树的叶子节点为 S 个。接着，用构建好的树索引划分 *SP* 得到一个 **PairRDD**，**PairRDD** 的每个元素的 **value** 值为对应的空间点，**key** 值为与该点 ϵ 范围相交的网格标识。由于点的 ϵ 范围可能与多个网格相交，因此一个点可能会形成多个键值对。Fig. 3.4(b)介

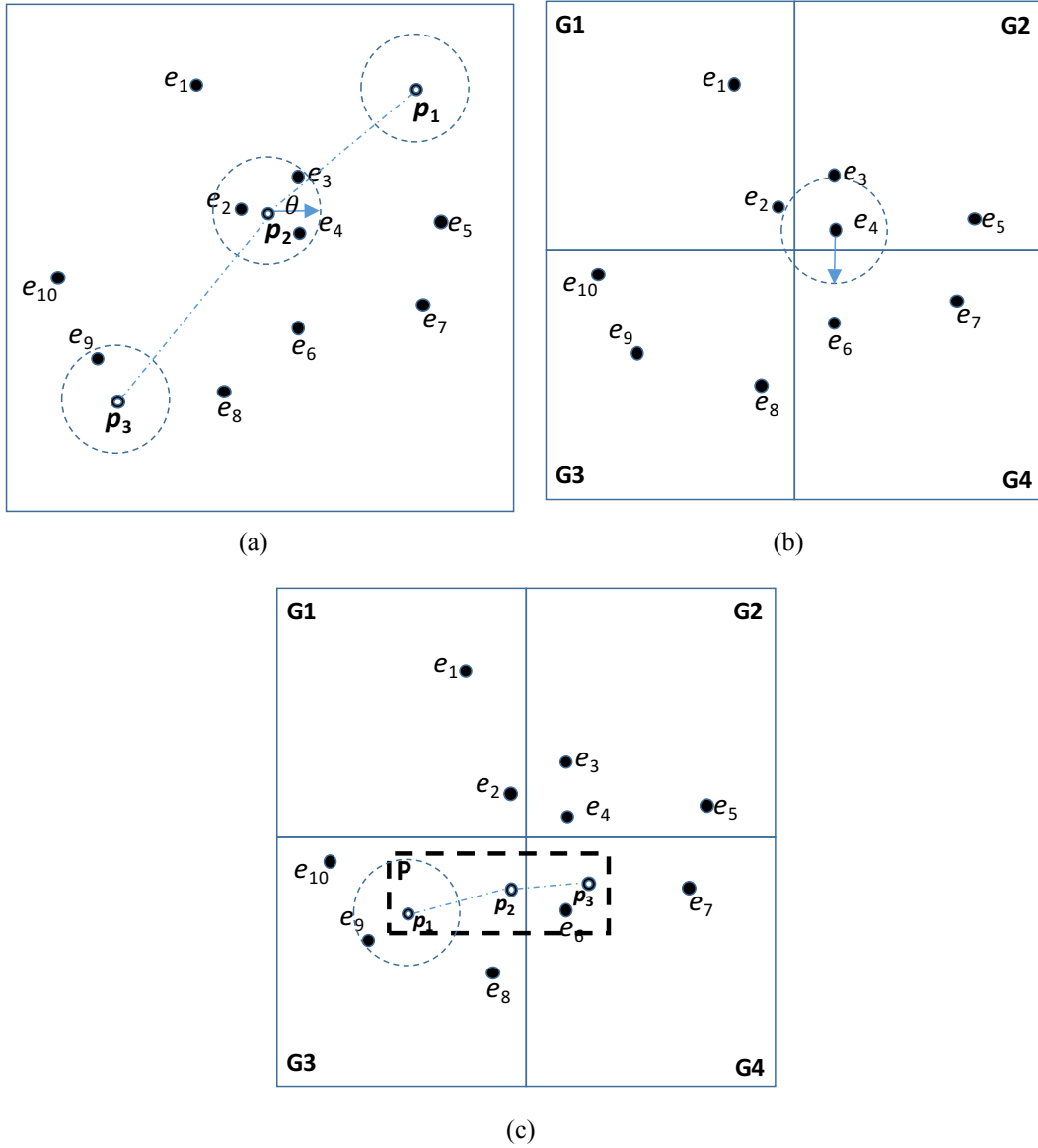


图 3.4: Join 查询示例

绍了当使用四叉树来划分 SP 时的案例。在该图中，点 e_4 被映射成 4 个键值对 $\langle G1, e_4 \rangle, \langle G2, e_4 \rangle, \langle G3, e_4 \rangle, \langle G4, e_4 \rangle$ 。最后，我们对这样的 PairRDD 中键相同的元组看作为一个分区，并对该分区内部空间点构建局部索引。这样 SP 就被转换成具有局部和全局两层索引。完成以上对空间数据集的预处理后，我们设计了如下 join 查询算法，该算法包含三个主要步骤：全局 join，局部 join 和合并。下面将对这三个部分进行详细介绍。

全局 join. 在该阶段，根据 $IndexTRDD$ 和 SP 的全局索引，将两者有交集的

分区形成分区对。如图3.4(c)所示, 带有虚线的方框 P 代表了 $IndexTRDD$ 的一个分区的包围矩形。 P 跟 SP 数据集的 $G3$ 和 $G4$ 相交, 因此会形成两个新的数据分区 $\langle P, G3 \rangle$ 和 $\langle P, G4 \rangle$ 。其中 $\langle P, G3 \rangle$ 包含了 $IndexTRDD$ 的 P 分区和 SP 的 $G4$ 分区的数据。其中由于 $G3$ 和 $G4$ 所代表的分区包含了除自身范围外还有其边界外 ϵ 范围内的点 (图中 e_4 存在于 $G3$ 和 $G4$ 所对应的分区中), 所以构成的分区对会包含 join 查询结果的完备集。因此, 我们对两个数据集的全局索引使用传统的空间 join 算法, 得到相交的分区对。最终, 将所有分区对的数据整合在一起, 形成新的数据集。

局部 join. 根据上一步得到的结果, 对每一个分区内的属于轨迹的那部分数据, 逐步遍历每条轨迹, 并对每个点从 SP 部分的索引中找出满足 ϵ 约束的最近邻点。若每个轨迹点找到满足条件的点, 则将原始轨迹的位置信息, 替换为对应匹配点的信息。若找不到满足条件的点, 则忽略该轨迹点。局部 join 产生的数据即为映射后的轨迹数据, 只是将原来的位置信息维度替换为空间数据的标识。由于某个空间点会存在多个分区中, 因此局部匹配后的结果会出现对一个轨迹

合并. 由于 $IndexTRDD$ 的一个分区会与 SP 的多个分区相交, 所以存在着某个轨迹点在一个 SP 分区中没有匹配点, 但在另一个 SP 分区中有匹配点的情况。如图 3.4(c)中, $\langle p_1, p_2, p_3 \rangle$ 是一段位于分区 P 中的片段。当 P 和 $G4$ 产生分区对时, p_1 将不会被匹配到任何点, 但是当 P 和 $G3$ 构成的分区对时, p_1 会被匹配到 e_9 上。因此, 在该阶段, 我们需要对局部 join 的结果进行合并以形成最终的结果集。

3.3 实验分析

3.3.1 实验设置

在本章节我们将验证系统的性能。所有的实验是在由 12 个节点构建的分布式集群上实现。每个节点包含一个 8 核 Intel E5335 2.0GHz 处理器和 16G 内存, 每个节点的操作系统为 Ubuntu 12.0.4, 并运行着 Spark 1.5.2。Spark 集群以 standalone 模

数据集	时间	记录数	大小 (Gb)
真实数据集	2013.10.1-12.31	约 250 万	190
人工数据集	2013.10.1-12.31	约 1.8 亿	1400

表 3.1: TrajSpark 数据集描述

式部署。

我们使用了两个轨迹数据集来验证系统的性能 [17], 包含一个真实轨迹数据集合一个人工合成的轨迹数据集。数据集描述如表3.1所示, 这两个数据集为 13,007 量北京出租车在 3 个月内 (2013 年 10 月至 12 月) 产生的轨迹数据。每条记录包含了如下信息: 车的标识、采集时间、经度、纬度、速度和角度等其他描述信息。真实数据集含约 250 万条轨迹日志记录, 数据量为 190Gb。为了更好的演示 TrajSpark 系统的可扩展性, 我们根据真实数据集生成了人工数据集。生成方法为将出租车在运行过程中的点通过差值法增加数据使得每 5 秒种有一个点产生。生成的人工数据集含约 1.8 亿条记录, 数据量为 1.4Tb。需要指出的是, 真实数据集可以完全被加载入系统内存, 而人工数据集无法被完全加载。

在 TrajSpark 系统中我们将整个北京空间划分成 $1,000 * 1,000$ 个细格子, 使得每个格子覆盖的范围为 $180m * 180m$ 。我们将 TrajSpark 跟最新的两个空间数据管理系统 GeoSpark 和 Simba 从查询延迟和可扩展性两方面进行对比分析。其中延时指标使用 10 个查询的平均时间进行计算。

3.3.2 实验结果及分析

首先, 我们研究了系统加载数据的时间开销。图3.5介绍了当所有系统随着数据不断导入到系统, 系统更新一批数据的时间开销, 其中每一批数据的大小为 32Gb。从图中可以发现 GeoSpark 和 Simba 都随着已转入数据量的增加导致新来的数据加载时间呈线性增加。这是由于这两个系统会将已有数据和新来的数据结合起来统一考虑, 然后将两者作为整体重新划分。划分的过程会导致数据的洗牌, 占用较多的网络通信和磁盘 I/O 开销。而 TrajSpark 由于只需对新来的数据进行数据划分, 然后将划分后的分区信息汇总入全局索引中。因此, 划分开销并不会随着数据的

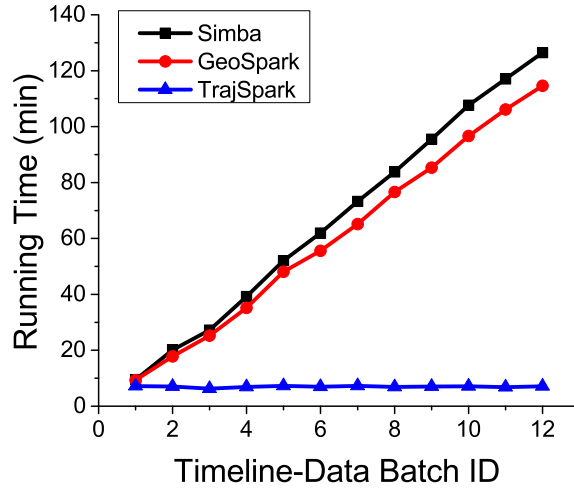
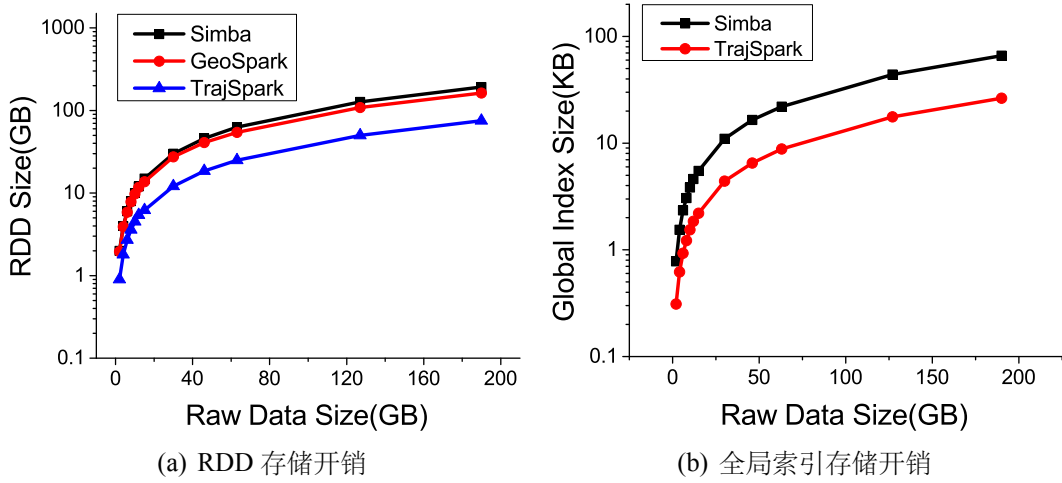


图 3.5: 数据加载开销



(a) RDD 存储开销

(b) 全局索引存储开销

图 3.6: 数据存储开销

增加而发生较大变化。综上所述, TrajSpark 具有较好的数据可扩展特性。

接着, 我们研究了从 IndexTRDD 和全局索引两部分的存储开销随数据量的变化情况。图3.6展示了 3 个系统当不同数据量级的轨迹数据加载后, 数据存储和全局索引两者存储开销的变化情况。其中图3.6(a)显示当数据加载入 TrajSpark 系统后, 所消耗的存储开销最低。而当加载入 Simba 和 GeoSpark 后消耗的存储开销是 TrajSpark 的 2-3 倍。这是因为 TrajSpark 会将原始轨迹数据按列存储形式存放并进行压缩, 而后两者直接存放原始数据, 数据冗余较多。图3.6(b)介绍了 3 个系统的全局索引存储开销。从图中可以发现这一开销在三个系统内部均较少(数量级为

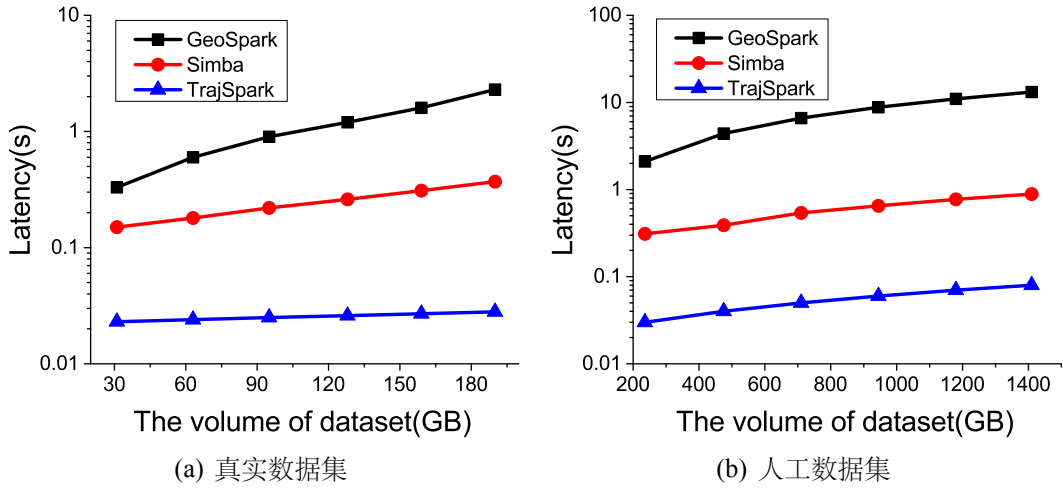


图 3.7: 基于给定对象的查询性能

Kb), 因而都可以轻松存放内存中。其中 TrajSpark 系统的全局索引开销最少 (约是 Simba 的 1/3), 这是因为其数据压缩后, 每个分区能放入跟多的数据, 从而导致分区数较少。而全局索引是构建在分区之上的, 所以 TrajSpark 会消耗更少的内存来存放全局索引。

然后, 我们验证了 TrajSpark 在基于给定对象的查询在不同数据量下的效果。图3.7 展示了查询性能。从该图中我们可以发现 TrajSpark 比 Simba 快了一个数量级, 且比 GeoSpark 快了两个数量级。这是由于 GeoSpark 需要遍历整个数据集, 所以其效果最慢。而 Simba 虽然能通过全局索引剪枝掉不相关的数据分区, 但它仍然需要遍历每个分区内部的所有数据。相比较而言, TrajSpark 既使用了全局索引剪枝分区, 又在每个分区内使用局部哈希索引进行随机访问。需要注意的是, 这些系统在真实数据集上的延时比人工数据集下的低。这是由于内存容量可以存放大部分真实数据集, 而只能转载的下少部分人工数据集。所以人工数据集下的查询会有额外的 I/O 开销。尽管如此, 这些系统在人工数据集下仍取得较好的结果。这是由于以下两点原因: (1) 我们将数据以 “MEM_AND_DISK_SER” 形式存放, 经常被访问的数据会被存放到内存中; (2) 使用全局索引, 能避免很多 I/O 开销。

其次, 我们验证了 TrajSpark 在时空范围下的查询性能。由于数据分布的不均匀性, 查询空间范围的选取对查询效率有着较高的影响。为此, 我们随机选择了

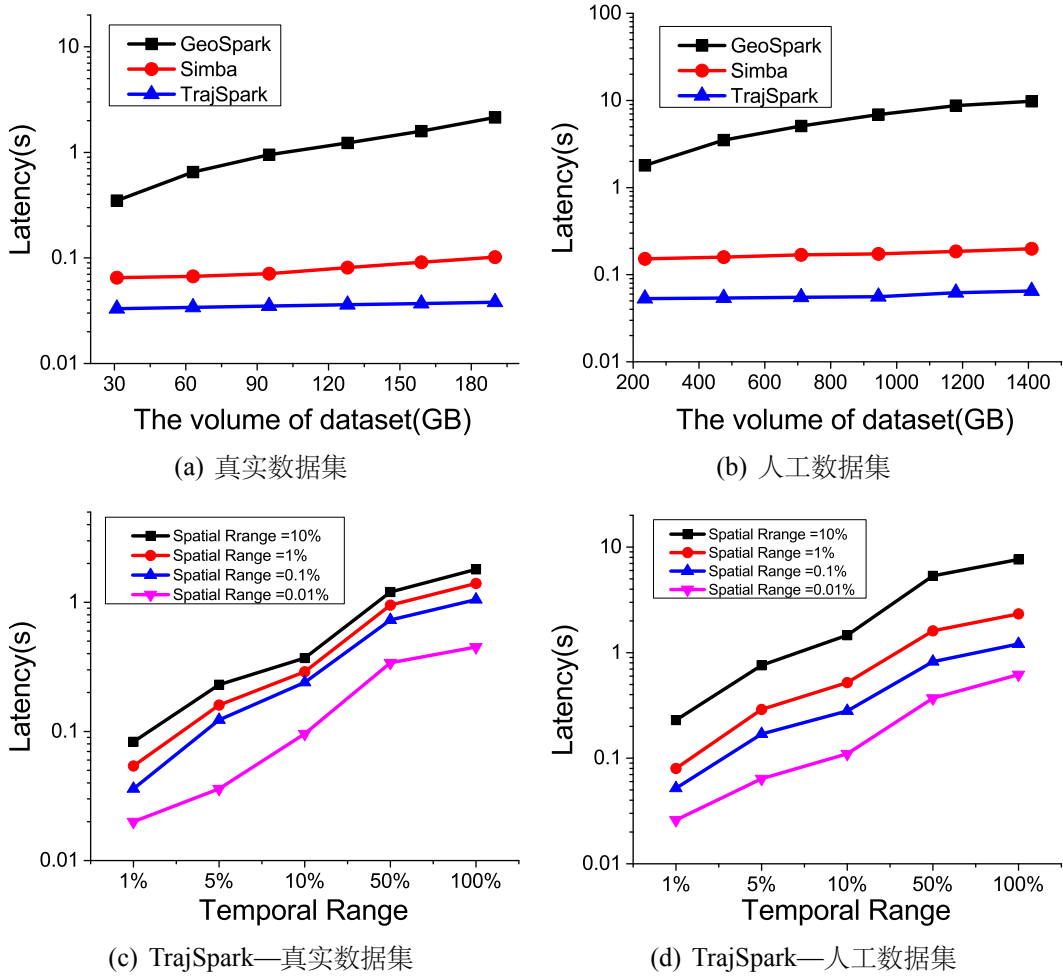
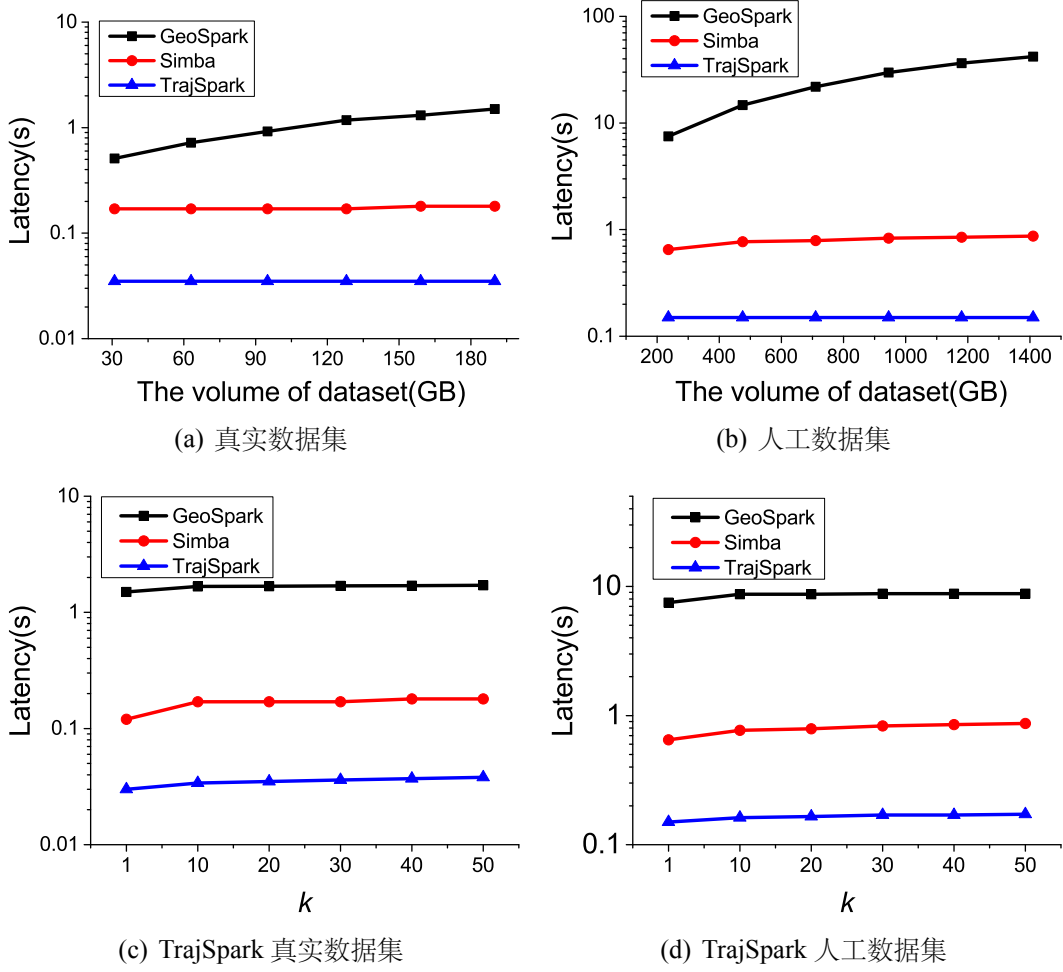


图 3.8: 时空范围查询性能

100 个包含 20×20 个方格的区域作为查询的空间约束, 选取最近的一周作为当前数据的查询时间约束。查询延时使用 100 个查询时间的平均值来计算。实验结果如图 3.8 所示, 其中图 3.8(a) 和 3.8(b) 介绍了查询时间随数据量大小的变化情况。我们可以发现, 随着数据量的增加, TrajSpark 和 Simba 上的查询效率并没有发生较大变化, 这是由于其全局和局部两层索引剪枝的效果。而 GeoSpark 由于需要遍历数据集, 导致随着数据的增加而查询时延逐渐增大。此外, 可以发现 TrajSpark 比 Simba 快了 3-5 倍, 这是由于其可以使用轨迹的 MBR 和时间范围来快速剪枝轨迹。此外, 我们验证了查询时间随空间和时间约束变化的情况。我们将空间范围选取了整个空间的 10%, 1%, 和 0.1%, 时间范围选取了三个月的后 100%, 50%, 10%, 5% 和 1%

图 3.9: k 近邻轨迹查询性能

。从图3.8(c)和3.8(d)可以发现随着查询时间和空间范围的增加，查询的时间也随之增加。但增加的方式并不是线性的。比如，当空间范围设置为 0.01% 时，我们变换时间范围从 1% 变化到 100%，空间范围扩大了 100 倍，但查询时间只延长了 30 倍。因此，TrajSpark 处理大的时空范围查询仍能取得较好的效果。

再接着，我们验证了 TrajSpark 处理欧氏距离下 k 近邻查询的性能。图3.9(a)和3.9(b)展示了当 k 设置为 10 时，查询时间随数据量大小的变化情况。我们可以发现 TrajSpark 比 GeoSpark 快了两个数量级，比 Simba 快了 4-6 倍。这是由于 TrajSpark 和 Simba 使用全局索引剪枝时空不相关数据分区，而 GeoSpark 需要遍历数据集的结果。此外 TrajSpark 由于不需要对同一个移动对象的轨迹点重新排序，

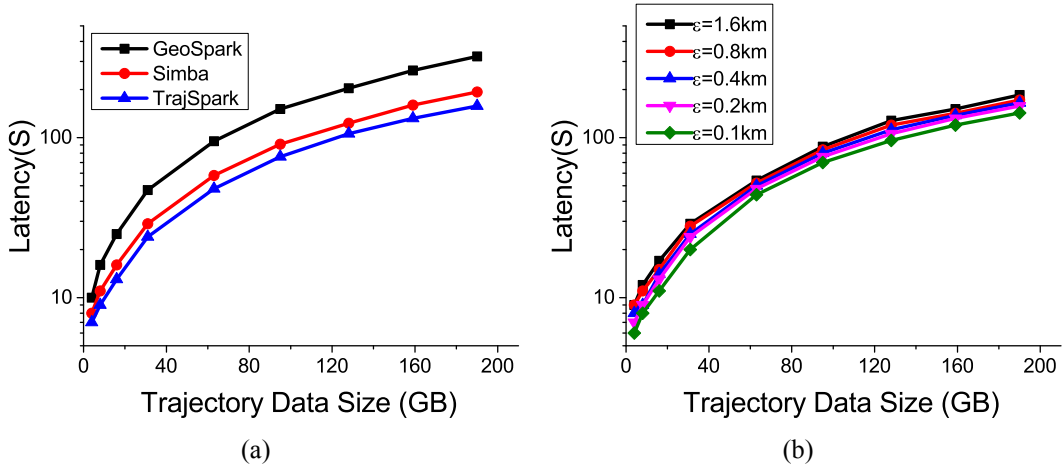


图 3.10: Join 查询性能

因而查询效率更高。这一查询结果与时空范围类似，这是因为 k 近邻查询首先会执行一个迭代的时空范围查询。进一步地，我们将 k 值从 1 变化到 50 以查看 k 值对查询效率的影响。从图3.9(c)和3.9(d)可以看出 k 值对查询结果影响不大。这是距离较近的轨迹大都放在跟查询轨迹相同的分区内，而每个分区存放的轨迹数远超 50 个。

最后，我们验证了 TrajSpark 处理 join 查询的性能。实验所用空间数据集包含两部分数据：北京路网数据结合 [130] 和北京 POI 数据集 [130]。其中路网上界集包含 148,110 个点和 196,307 条边，而 POI 数据集包含 273,165 个 POI 点。为保持结果简洁，我们只展示了当使用四叉树来索引空间数据的情况（其他索引下结果趋势相同）。Simba 实现本文 join 查询需要结合其自身提供的 k 近邻 join 和距离约束 join 两种接口。Fig. 3.10展示了 join 查询的性能与数据量之间的关系。我们可以发现随着数据量的线性增加，查询时间也呈线性增加。进一步地，图3.10(a)对比了当 ϵ 设置为 0.2km 时，不同系统间的性能比较。这三个系统都支持对轨迹点通过查找空间对象数据集的局部索引以便能快速找到满足 ϵ 约束的点，所以都能取得较好的查询结果。但 TrajSpark 和 Simba 比 GeoSpark 具有更好的加速比，这是因为前两者能通过全局索引，能够剪枝掉不必要的分区对。而 GeoSpark 由于缺乏全局索引，故需要对其每个 SRRD 分区匹配空间数据集的每个分区，造成过多不必要

的匹配开销。而 TrajSpark 比 Simba 性能好的原因是，其对匹配完后的轨迹点无需排序，而 Simba 是以点的形式管理数据，为形成结果轨迹，需要对属于同一移动对象的点排序。最后，我们验证了算法性能与 ϵ 的关系。显然地， ϵ 值越大，则意味着轨迹点搜索最近邻点的空间越大，从而需要与更多的候选进行距离计算。所以查询时间随着 ϵ 的增大而增加。但是，由于使用局部索引来搜索候选，导致 ϵ 增加后仍能快速找到候选。最终，导致 ϵ 的增加对查询开销的影响不大。

3.4 本章小结

本章节介绍了使用分布式集群对轨迹大数据进行有效管理并提供实时查询结果。本章节，首先介绍了分布式内存处理系统 Spark 的相关知识。接着，介绍了基于 Spark 轨迹数据管理系统及其上经典查询的设计。TrajSpark 基于内存的计算思想并引入了全局和局部两层索引，因此设计的查询具有较低的延时。通过在海量轨迹数据集进行的实验，表明 TrajSpark 具有较高的查询效果和可扩展性。

第四章 基于上、下界剪枝的分布式 k 近邻查询

本章主要介绍如何使用距离上、下界剪枝策略来实现分布式 k 近邻查询，并达到降低通信开销的目的。首先，章节4.1介绍了本章的研究思路和预备知识。然后，章节4.2介绍了使用多粒度概要数据计算距离上、下界并进行逐步剪枝的策略。接着，章节4.3针对欧氏距离，使用哈尔小波变换得到多粒度概要数据，并利用部分小波系数的多粒度特性设计了逐步变紧的距离上、下界。其次，章节4.4介绍了算法ED-FTB，以处理当使用欧氏距离作为距离度量准则时的查询。再接着，章节4.5实验展示了ED-FTB算法的有效性和可扩展性。最后，章节4.6小结本章的研究内容。

4.1 引言

4.1.1 研究思路

本文的研究内容之一就是处理协调者-远程结点架构下的 k 近邻查询。正如第一章所述，直接将查询轨迹发送到所有远程结点进行距离计算的方式虽能保证结果的正确性，但网络传输开销较大。为此，在保证结果正确性的前提下，降低通信开销成为查询的首要挑战。本文的研究思路是：协调者结点首先选用合适的降维策略先对轨迹数据进行降维处理以得到概要数据，接着将该概要数据发送给所有远程结点。远程结点在获取到概要数据后，能够对其保存的局部轨迹快速计算出与查询轨迹距离的范围，并根据距离范围进行剪枝。除此之外，我们希望通过降维得到的概要数据具有多粒度特性，即粒度越细，所对应的数据量也越多，同时所包含的信息量也越大。与之对应的，所计算出来的距离范围也越精确。通过不断精确的距离范围，我们能进一步剪枝不相关的候选。有两点需要注意：(i) 粒度越高，对应的概要数据所能表示的信息量也越接近原始轨迹数据；(ii) 在根据由粗到细粒度概要数据剪枝过程中，可能不需要到最细粒度，结果已经找出。

从以上思路出发，本章从所得到的距离范围出发考虑了根据概要数据能同时

计算出所给距离函数的上界和下界情况。下一章考虑了仅能计算出下界的情况。此外，需要注意的是仅能计算出上界的情况是无意义的。这是因为查询目标是找距离最小的 k 条轨迹。针对能同时计算出上、下界的场景，我们首先给出界特征定义。

定义 4.1.1 (界特征 (Bound Feature, BF)). 三元组 $\langle id, ub, lb \rangle$ 构成了界特征，记录了一条轨迹的标识 (id)，以及它与待查询轨迹的距离上界 (ub) 和下界 (lb)。

对任意一条轨迹其默认的距离下界为 0，上界为正无穷。我们的目标是根据概要数据不断提高界特征的下界值和降低界特征的上界值。

4.1.2 接口函数

在介绍如何进行查询处理之前，我们将首先介绍查询处理所用到的基本通信和处理接口函数。这些接口均为抽象函数，在应用中，用户可以自定义其内容（接下来两章将会介绍这些接口函数在使用欧氏和动态时间弯曲距离下的具体实现）。我们将这些接口函数按照运行所处位置分为两类。一类是协调者结点上接口函数，另一类是远程结点接口函数。

协调者结点函数接口：

- **coordinatorInit(\mathcal{Q}, \mathcal{R})**. 协调者结点运行该函数以实现查询初始化。该函数涉及的内容包括对查询轨迹 \mathcal{Q} 进行概要数据计算以及协调者结点跟所有远程结点的信息传递等。
- **generateInfo()**. 协调者结点生成所需要发送的数据。具体的，该函数产生指定粒度的概要数据。
- **sendToRemoteSites(\mathcal{R}, x)**. 协调者节点将数据 x 发给所有在集合 \mathcal{R} 中的远程节点。
- **getFromRemoteSites(\mathcal{R})**. 协调者结点从集合 \mathcal{R} 中的每个远程结点获取信息，并阻塞等待所有结点的信息都接收完毕后返回。

远程结点函数接口：

- $\text{remoteInit}(TS_r, S_r)$. 远程结点运行该函数以实现初始化。该函数主要工作为保存在集合 TS_r 中的每条轨迹初始化界特征，并将这些特征值放在集合 S_r 中。此外，该函数还可能涉及到对每条轨迹数据进行概要数据计算。
- $\text{getFromCoordinator}()$. 该函数接收协调者结点运行 sendToRemoteSites 函数后所发数据。
- $\text{sendToCoordinator}(x)$. 远程结点将信息 x 发送给协调者结点。协调者节点则通过 $\text{getFromRemoteSites}$ 函数接收所发消息。
- $\text{updateBounds}(S_r, x)$. 远程结点根据获取到的信息 x 更新界特征集合中值。

4.2 基于上、下界的逐步剪枝策略

本节将介绍使用上、下界进行逐步剪枝的策略 FTB 的设计原理及实现方案。

4.2.1 策略设计原理

我们假设能对查询对象选用合适的降维策略得到存储空间相对较小的多粒度概要数据，利用该概要数据可以同时计算出距离的上、下界。FTB 框架核心思想就是从这一假设出发，具体实现是：协调者结点由粗到细粒度的发送概要数据，远程结点则根据该数据不断为本地候选轨迹计算更加紧凑的距离上、下界，并根据界特征进行剪枝。图4.1给出了使用多粒度概要数据不断逼近与某一候选轨迹真实距离的过程。图中，横坐标为概要数据的粒度，纵坐标表示轨迹间距离的值。当概要数据粒度，不断增加的过程中，所计算出的相似度上界（红色实线）和下界（蓝色虚线）不断逼近距离的真实值（黑色实线），并最终等于真实值。当获得了所有轨迹的界特征后，就可以根据如下剪枝原理对候选进行剪枝。

引理 4.2.1 (剪枝原理). 给定一界特征集合 S ，若某界特征的下界大于集合中第 k 小的上界，则该界特征所对应轨迹不会进入最终结果集，即可以被剪枝掉。

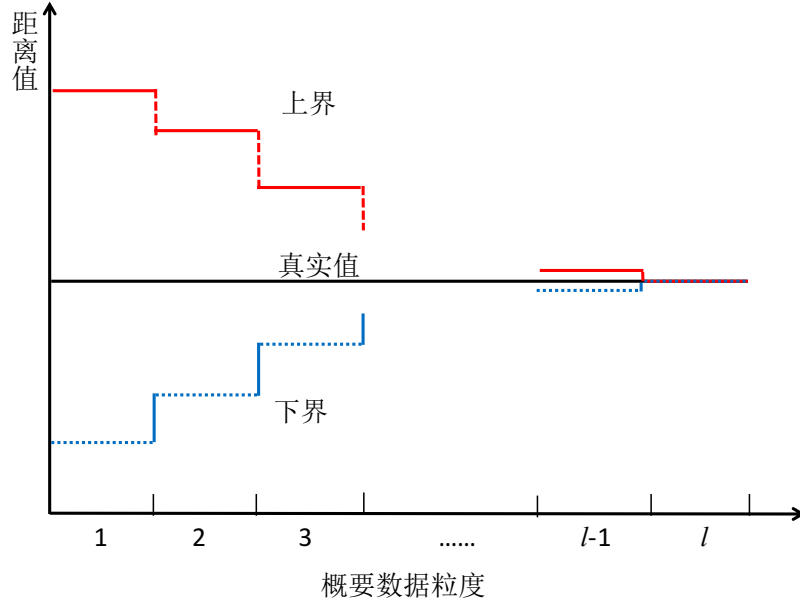


图 4.1: 利用多粒度概要数据不断逼近真实距离

证明. 对 S 中的界特征按上界值由小到大排序, 并记排序后的第 k 个界特征为 $S[k]$ 。假设某轨迹的特征为 $S[c]$ ($c > k$), 且 $S[c].lb > S[k].ub$ 。由于 $S[c]$ 所对应的真实值大于 $S[c].lb$, 所以有 $S[c]$ 的真实值大于 $S[k].ub$ 。此外, 又由于 $S[k].ub$ 大于前 $k-1$ 个界特征的上界, 则 $S[k].ub$ 大于前 $k-1$ 个界特征所对应轨迹的真实距离值。因此, $S[c]$ 所对应的轨迹真实距离值大于前 $k-1$ 个界特征所对应轨迹的真实距离值。故 $S[c]$ 可以被剪枝掉。□

图4.2示例介绍了剪枝原理。首先, 将界特征按上界由小到大排完序后。对于第 c 个轨迹 ($c > k$), 若其下界大于第 k 个的上界。则该轨迹的真实值必然大于前 k 个轨迹的真实值。由于我们的目标是查找距离最小的 k 条轨迹, 而第 c 个轨迹不可能存在于结果集中。此时, 根据对其计算出来的界特征就可以将其从候选集合中移除。

4.2.2 FTB 策略实现方案

根据上节介绍, 我们知道使用 FTB 框架需要满足两点条件: (i) 协调者节点能根据给定的距离函数, 设计出满足多粒度属性的概要数据; (ii) 远程结点能根据

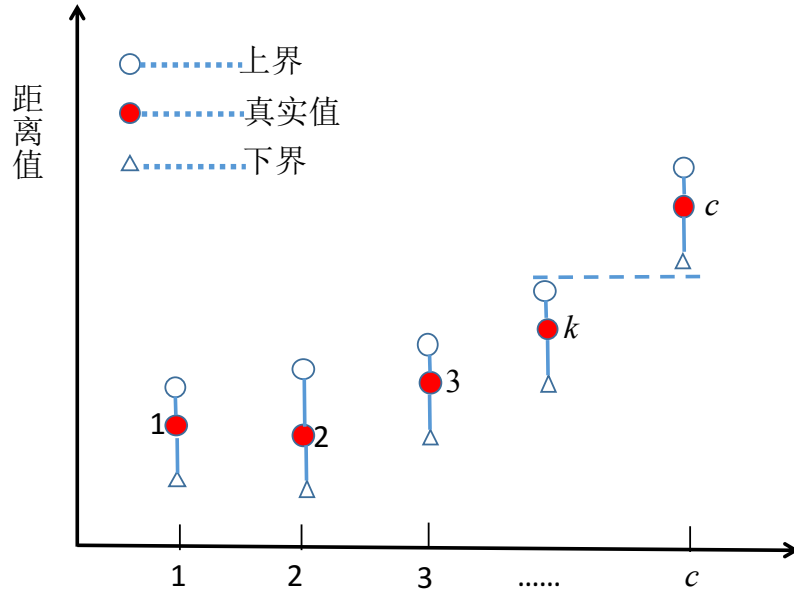


图 4.2: 剪枝示例

概要数据计算出距离上、下界，且能保证该界收敛。基于以上分析，本文所设计的 FTB 框架分为两个部分，一部分是运行在协调者结点的算法4，另一部分是运行在所有远程结点的算法5。在框架运行过程中，这两个算法互相通信并协调工作。框架运行过程可分为三个阶段：初始化阶段，迭代交互式剪枝阶段，最终结果获取阶段。下面将对这三个部分分别进行介绍：

初始化阶段：协调者结点在此阶段执行多种操作，如获取远程结点的列表、概要数据计算以及可能预发送一些信息。在此过程中，我们使用 \mathcal{R} 来表示候选远程结点的集合（算法 4:1-2 行）。与之对应的，远程结点在该阶段的初始化主要工作是为保存在本地数据集合 TS_r 中轨迹初始化界特征并将结果存在局部候选集 S_r 中。此外，它也会接收协调者结点发过来的数据（算法 2:1-2行）。在该阶段，所有轨迹都是候选结果。我们把包含候选的远程结点称为候选（远程）结点。

迭代交互式阶段：在此阶段，协调者结点与候选远程结点交互通信直到剪枝完毕。首先，远程结点生成概要数据并发送给所有候选结点（算法 4:5-6行）。接收到概要数据后，候选远程结点利用该概要数据为每个候选计算距离上、下界并更新界特征。需要注意的是：随着迭代次数的增加，所生成的概要数据粒度越细。这使

算法 4 FTB 之协调者结点**输入:** 查询轨迹 Q , 结果集大小 k ;**输出:** k 条最相似轨迹的 ID;

```

1:  $\mathcal{R} \leftarrow$  远程结点集合;
2: coordinatorInit( $Q, \mathcal{R}$ );
3: while true do
4:   /* 生成全局第  $k$  小上界 */
5:    $Info \leftarrow$  generateInfo(); # 准备概要数据
6:   sendToRemoteSites( $\mathcal{R}, Info$ );
7:    $GUBS \leftarrow$  getFromRemoteSites( $\mathcal{R}$ );
8:    $gkub \leftarrow \operatorname{argmin}_{\tau}(|x \in GUBS, x < \tau| \geq k)$ ;
9:   sendToRemoteSites( $\mathcal{R}, gkub$ );
10:  /* 获取候选集大小  $\mathcal{R}$  */
11:   $CSS \leftarrow$  getFromRemoteSites( $\mathcal{R}$ );
12:   $\mathcal{R} \leftarrow \{x.r | x \in CSS, x.|S_r| > 0\}$  # 剪枝远程结点
13:   $sum \leftarrow \sum_{x \in CSS} x.|S_r|$ ; # 候选集大小
14:  if  $sum = k$  then
15:    sendToRemoteSites( $\mathcal{R}, finish$ );
16:    break;
17:   $ids \leftarrow$  getFromRemoteSites( $\mathcal{R}$ );
18: return  $ids$ ;
```

得我们计算出来的上、下界越来越紧。根据最新的界特征，每个候选结点找出局部最小的 k 个上界并将它们发送给协调者结点 (算法 5:5-10行)。当协调者结点获取到所有候选远程结点发送来的上界后，它从中选择第 k 小的上界，记为 $gkub$ ，并将其值发送给那些候选远程结点 (算法 4 :7-9行)。候选结点在接收到 $gkub$ 后，就可以剪枝局部候选轨迹。具体剪枝操作是，比较每个局部候选的下界与 $gkub$ 的值。如下界值大于 $gkub$ ，则将该轨迹从局部候选中删除。剪枝后，候选结点将其所剩候选的个数发给协调者结点。若候选结点的所有轨迹都被剪枝掉，则该结点运行结束 (算法 5:11-15行)。在此之后，协调者结点收集候选远程结点所发送的局部候选的个数，并求和计算总的个数。若发现某结点不再包含候选，则将其从候选结点集中移除。此外，若发现当前所剩候选总数正好为 k 个，则迭代终止。协调者将向剩下的候选结点发送终止信号 *finish*。若所剩个数大于 k ，则迭代继续 (算法 4:11-16行)。

最终结果获取阶段: 经过上一阶段，最终会剩余 k 个候选。这些候选即为最终

算法 5 FTB 之远程结点**输入:** 局部轨迹集合 TS_r ;**输出:** 属于 top- k 结果集的轨迹 ID;

```

1:  $S_r \leftarrow$  局部轨迹界特征集;
2: remotelnit( $TS_r, S_r$ );
3: while true do
4:    $m \leftarrow$  getFromCoordinator();           # 从协调者结点获取信息
5:   if  $m$  is Info then
6:      $S_r \leftarrow$  UpdateBounds( $S_r, m$ );
7:     /* 产生局部最小  $k$  个上界 */
8:      $\hat{ub} \leftarrow \operatorname{argmin}_\tau (|x \in S_r, x.ub \leq \tau| \geq k)$ ;
9:      $S' \leftarrow \{\alpha.ub \mid \alpha \in S_r, \alpha.ub \leq \hat{ub}\}$ ;
10:    SendToCoordinator( $S'$ );           # 将最小的  $k$  个上界返回给协调者结点
11:   else if  $m$  is gkub then
12:      $S_r \leftarrow \{\beta \mid \beta \in S_r, \beta.lb \leq m\}$            # 局部剪枝
13:     SendToCoordinator( $\langle r, |S_r| \rangle$ );
14:     if  $|S_r| = 0$  then
15:       return ;           # 当远程结点无候选, 则停止运行
16:   else
17:     /*  $m$  is finish */
18:      $ids \leftarrow \{a.id \mid a \in S_r\}$ ;
19:     SendToCoordinator( $ids$ );
20:   return ;

```

的结果。所剩的候选结点在收到上一阶段所发的结束信号后, 会将本地所剩候选的 ID 发送给协调者节点。发送成功后, 自身会结束查询 (算法5: 18-20行)。而协调者节点则会收集所有候选结点发送过来的 ID, 并返回给用户 (算法4: 17-18行)。

4.3 欧氏距离的概要数据及上、下界计算**4.3.1 基于欧氏距离的轨迹相似度度量**

欧氏距离用于处理两条长度相同的轨迹。假设待查询轨迹 Q 表示为 $Q = \{q_0, q_1, \dots, q_{n-1}\}$, 一条候选轨迹 C 表示为 $C = \{c_0, c_1, \dots, c_{n-1}\}$, 其中 n 为轨迹长度, 每个轨迹点来自 d 维空间, 即 $q_i, c_i \in R^d$ 。我们首先给出点之间的距离定义, 不失一般性的, 我们使用欧氏距离来度量。

$$ED(q_i, c_i) = \|q_i - c_i\| = \sqrt{\|q_i\|^2 + \|c_i\|^2 - 2q_i \cdot c_i} \quad (4.1)$$

其中 $\|q_i\|$ (或 $\|c_i\|$) 表示 q_i (或 c_i) 的二范数的值, $q_i \cdot c_i$ 表示向量 q_i 和 c_i 之间的点 (内) 积。在此基础上, 我们给出轨迹间的欧氏距离定义。

$$ED(Q, C) = \sqrt{\sum_{i=0}^{n-1} ED(q_i, c_i)^2} \quad (4.2)$$

为简化计算和表示, 我们使用欧氏距离的平方 (Squared Euclidean Distance, SED) 来替换距离, 即 $SED(q_i, c_i) = ED(q_i, c_i)^2$ and $SED(Q, C) = ED(Q, C)^2$ 。此时, 轨迹距离的欧氏距离可以看作是点距离的累加和。

4.3.2 基于哈尔小波的轨迹概要数据抽取

哈尔小波变换是降维时间序列、图像数据等的有效方法。它将原始数据从多个解析度来展示, 每个解析度代表了不同频域下的信息。其变换过程又可以被抽象成如图4.3(a)所示的自底向上构建一颗二叉树 (称为误差树, Error-tree) 的过程。在该图的最底层 (叶子层) 自左往右是时间序列原始数据, 非叶子节点保留两个值 a_i^j 和 d_i^j 。 a_i^j 记录了该结点两个孩子结点的均值的正则化均值, 即 $a_{i,j} = (a_{i+1,2j} + a_{i+1,2j+1})/nf$ 。 d_i^j 记录了该结点两个孩子结点均值的正则化差值信息, 即 $d_{i,j} = (a_{i+1,2j} - a_{i+1,2j+1})/nf$ 。 nf 称为正则化因子 (normalization factor), 其取值为 $1/2$ 或 $1/\sqrt{2}$, 具体是应用场景而定。由于最底层 (叶子节点层) 结点仅包含原始数据, 故倒数第二层结点直接从原始值计算出来。在自底向上构建的过程中, 每层结点个数减半, 且直到某层仅包含一个节点为止 (第 0 层)。

在误差树结构中, 每层的均值序列可以看做对原始时间序列的一层概要数据, 自上而下粒度越来越细。然而, 哈尔小波并没有直接保存这些均值, 它依次保留了最终的均值及由上到下每层的差值, 并将保留的值称为 (哈尔小波) 系数。我们能根据系数值能恢复出每层的均值。具体的, 如图4.3(b)所示, 给定时间序列 $f(t) = \{8, 6, 2, 4, 5, 9, 17, 13\}$ 且 $nf = 1/2$ 时, 经过哈尔小波变换后的系数为 $H(f(t)) = \{8, -3, 2, -4, 1, -1, -2, 2\}$ 。给定第 0 层的系数 $8, -3$, 我们能计算出第 1 层的均值分别为 $8 + (-3) = 5, 8 - (-3) = 11$ 。此时, 我们仅使用了 2 个值就表达了与 3 个

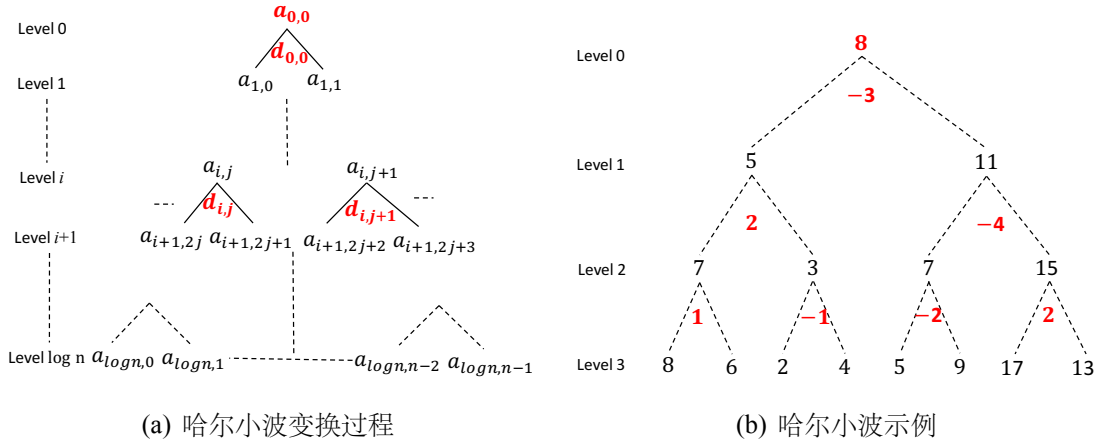


图 4.3: 哈尔小波变换-误差树

值（第 0 和第 1 层均值）同等量的信息。此外，若继续给出第 1 层的系数，我们使用相同的方法能恢复出第 2 层的均值。值得注意的是，为方便解释，示例中正则化因子为 $1/2$ ，本章接下来的应用中将使用 $1/\sqrt{2}$ 来进行变换。

前面介绍了哈尔小波变换以及如何使用它处理一维时间序列数据。但轨迹数据天然是多维时间序列数据。本节将介绍如何将其推广到多维的轨迹数据中。首先，给定查询轨迹 \mathcal{Q} 和候选候选轨迹 \mathcal{C} 。它们的长度均为 n ，且假设 n 是 2 的正整数次方。由于长度为 n 的时间序列，其对应误差树的深度为 $L+1$ ($L = \log_2 n$)。对于 \mathcal{Q} 和 \mathcal{C} 的哈尔小波系数分别为 $H\mathcal{Q} = \{a_{0,0}^{\mathcal{Q}}, d_{0,0}^{\mathcal{Q}}, d_{1,0}^{\mathcal{Q}}, \dots, d_{L-1,n/2-1}^{\mathcal{Q}}\}$ 和 $H\mathcal{C} = \{a_{0,0}^{\mathcal{C}}, d_{0,0}^{\mathcal{C}}, d_{1,0}^{\mathcal{C}}, \dots, d_{L-1,n/2-1}^{\mathcal{C}}\}$ ，其中 $a_{0,0}^{\mathcal{Q}}$ 和 $a_{0,0}^{\mathcal{C}}$ 分别是 $H(\mathcal{Q})$ 和 $H(\mathcal{C})$ 变换后的最终的正则化的均值， $d_{i,j}^{\mathcal{Q}}$ 和 $d_{i,j}^{\mathcal{C}}$ 是正则化后的差值，且 $a_{i,j}^{\mathcal{Q}}, a_{i,j}^{\mathcal{C}}, d_{i,j}^{\mathcal{Q}}, d_{i,j}^{\mathcal{C}} \in R^d$ 。根据正则化小波变换过程， \mathcal{Q} 的误差树中第 i 层第 j 个非叶子节点的内容 a_i^j 和 d_i^j 可以由如下公式计算。

$$a_{i,j}^{\mathcal{Q}} = \frac{a_{i+1,2j}^{\mathcal{Q}} + a_{i+1,2j+1}^{\mathcal{Q}}}{\sqrt{2}}, \quad d_{i,j}^{\mathcal{Q}} = \frac{a_{i+1,2j}^{\mathcal{Q}} - a_{i+1,2j+1}^{\mathcal{Q}}}{\sqrt{2}} \quad (4.3)$$

4.3.3 基于哈尔小波系数的欧氏距离表示

尽管哈尔小波被广泛应用于时间序列数据分析，但现有工作都是针对一维时间序列。本节将介绍如何将其推广到多维的轨迹数据中。首先，对于给定的 \mathcal{Q} 的误差树中的任意兄弟结点对 $\{a_{i+1,2j}^{\mathcal{Q}}, a_{i+1,2j+1}^{\mathcal{Q}}\}$ ，以及与之对应的 \mathcal{C} 的误差树中节点

对 $\{a_{i+1,2j}^Q, a_{i+1,2j+1}^Q\}$, 两节点对相应元素的欧氏距离和可以如下表示:

$$\begin{aligned}
 & SED(a_{i+1,2j}^Q, a_{i+1,2j}^C) + SED(a_{i+1,2j+1}^Q, a_{i+1,2j+1}^C) \\
 = & SED\left(\frac{a_{i,j}^Q + d_{i,j}^Q}{\sqrt{2}}, \frac{a_{i,j}^C + d_{i,j}^C}{\sqrt{2}}\right) + SED\left(\frac{a_{i,j}^Q - d_{i,j}^Q}{\sqrt{2}}, \frac{a_{i,j}^C - d_{i,j}^C}{\sqrt{2}}\right) \\
 = & \left\| \frac{a_{i,j}^Q + d_{i,j}^Q}{\sqrt{2}} \right\|^2 + \left\| \frac{a_{i,j}^C + d_{i,j}^C}{\sqrt{2}} \right\|^2 - 2 \frac{a_{i,j}^Q + d_{i,j}^Q}{\sqrt{2}} \cdot \frac{a_{i,j}^C + d_{i,j}^C}{\sqrt{2}} \\
 & + \left\| \frac{a_{i,j}^Q - d_{i,j}^Q}{\sqrt{2}} \right\|^2 + \left\| \frac{a_{i,j}^C - d_{i,j}^C}{\sqrt{2}} \right\|^2 - 2 \frac{a_{i,j}^Q - d_{i,j}^Q}{\sqrt{2}} \cdot \frac{a_{i,j}^C - d_{i,j}^C}{\sqrt{2}} \\
 = & SED(a_{i,j}^Q, a_{i,j}^C) + SED(d_{i,j}^Q, d_{i,j}^C)
 \end{aligned} \tag{4.4}$$

公式4.4说明, 细粒度的概要数据间的距离可以由粗粒度的数据计算出来。为进一步表示, 我们用 $S_i(Q, C)$ 表示 Q 和 C 误差树的第 i 层的对应 (正则化) 均值间的欧氏距离的平方和, 用 $SED_i(Q, C)$ 表示 Q 和 C 误差树的第 i 层的对应 (正则化) 差值间的欧氏距离的平方和。 $S_i(Q, C)$ 和 $SED_i(Q, C)$ 的定义如下所示:

$$S_i(Q, C) = \sum_{j=0}^{2^i-1} SED(a_{i,j}^Q, a_{i,j}^C) \tag{4.5}$$

$$SED_i(Q, C) = \sum_{j=0}^{2^i-1} SED(d_{i,j}^Q, d_{i,j}^C) \tag{4.6}$$

此外, 由于原始轨迹间的距离可以根据两个误差树的所有叶子结点计算出来, 即 $SED_L(Q, C) = SED(Q, C)$ 。根据这一结论和以上定义, 我们得到如下定理。

引理 4.3.1. 给定两条轨迹 Q 和 C , HQ 和 HC 分别表示 Q 和 C 经过哈尔小波变换后的系数序列。我们有如下结论: $SED(Q, C) = SED(HQ, HC)$ 。

证明. 首先, 根据 $S_i(Q, C)$ 和 $SED_i(Q, C)$ 定义 (定义4.5) 我们有

$$\begin{aligned}
 S_{i+1}(Q, C) &= \sum_{j=0}^{2^{i+1}-1} SED(a_{i+1,j}^Q, a_{i+1,j}^C) \\
 = & SED(a_{i+1,0}^Q, a_{i+1,0}^C) + SED(a_{i+1,1}^Q, a_{i+1,1}^C) + \cdots + \\
 & SED(a_{i+1,2^{i+1}-2}^Q, a_{i+1,2^{i+1}-2}^C) + SED(a_{i+1,2^{i+1}-1}^Q, a_{i+1,2^{i+1}-1}^C) \\
 = & SED(a_{i,0}^Q, a_{i,0}^C) + SED(d_{i,0}^Q, d_{i,0}^C) + \cdots +
 \end{aligned}$$

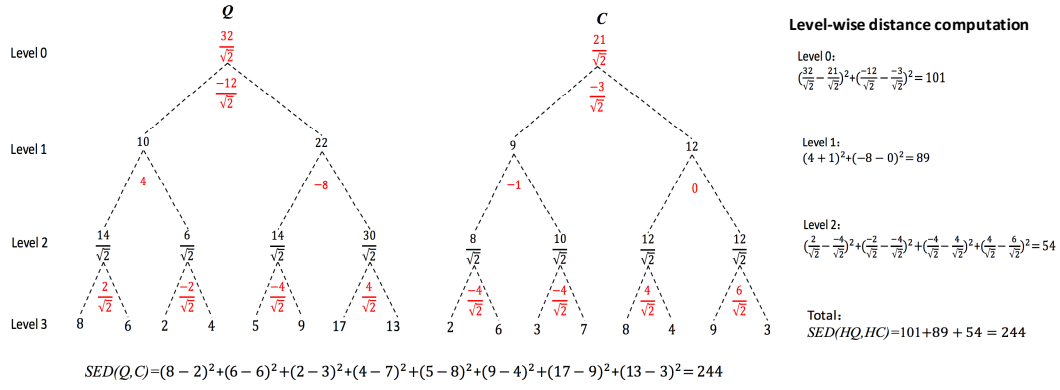


图 4.4: 基于 Haar 小波系数的距离计算示例

$$\begin{aligned}
 & SED(a_{i,2^i-1}^Q, a_{i,2^i-1}^C) + SED(d_{i,2^i-1}^Q, d_{i,2^i-1}^C) \\
 &= \sum_{j=0}^{2^i-1} SED(a_{i,j}^Q, a_{i,j}^C) + \sum_{j=0}^{2^i-1} SED(d_{i,j}^Q, d_{i,j}^C) \\
 &= S_i(Q, C) + SED_i(Q, C)
 \end{aligned} \tag{4.7}$$

根据如上公式，对于叶子层结点，我们有：

$$S_L(Q, C) = S_0(Q, C) + \sum_{i=0}^{L-1} SED_i(Q, C) = SED(HQ, HC) \tag{4.8}$$

此外，第 L 层为叶子节点层，包含了轨迹的原始信息。所以我们又有 $SED(Q, C) = S_L(Q, C)$ 。此时结合公式 4.8 我们得到 $SED(Q, C) = SED(HQ, HC)$ 。原问题得证。 \square

引理 4.3.1 既说明了原始轨迹间的欧氏距离等于哈尔小波系数间距离（证明见本章附件），又说明了通过对查询轨迹哈尔变换后的系数可以替换原始轨迹用于距离计算。此外，对于轨迹长度不是 2 的次方的情况，我们可以通过将轨迹切分成若干个子轨迹，只需保证每个轨迹是 2 的次方。然后，可以对子轨迹进行哈尔小波变换，并将所有子轨迹系数的欧氏距离累加等到整体轨迹间的距离。因此，哈尔小波可以被用来分解任意长度的轨迹。

图 4.4 针对引理 4.3.1 给出了一维时间序列下的示例说明。假定给定查询时间序列 $Q = \{8, 6, 2, 4, 5, 9, 17, 13\}$ ，另一条序列为 $C = \{2, 6, 3, 7, 8, 4, 9, 3\}$ 。通过计算可

得，这两条序列间的欧氏距离为 244。接着，我们对这两条序列分别进行哈尔小波变换分别得到如图所示的两颗误差树。对每一层的系数进行欧氏距离计算分别得到距离值 101、89 和 54。其中第 0 层计算时把最终的均值也算进去了。通过计算发现系数的累加和确实等于原始序列的欧氏距离。

4.3.4 基于哈尔小波的欧氏距离上、下界

上一节介绍了，利用完整的哈尔小波系数，即全部的概要数据，能够用来计算原始的轨迹间欧氏距离。但全部概要数据的数据量于原始轨迹数据量相同，不能达到降低通信开销的目的。为此本节将介绍如何利用部分概要来计算轨迹的上、下界。首先根据引理4.3.1, 我们有:

$$\begin{aligned} SED(Q, C) &= S_0(Q, C) + \sum_{i=0}^{L-1} SED_i(Q, C) \\ &= S_0(Q, C) + \sum_{i=0}^l SED_i(Q, C) + \sum_{i=l+1}^{L-1} SED_i(Q, C) \end{aligned} \quad (4.9)$$

假设远程结点由粗到细已经获取了查询轨迹 Q 的前 $l+1$ 层概要数据，我们的目标就是根据这些已有的数据计算出 Q 与任一候选 C 的欧氏距离上、下界。公式 4.9 有半部分分为两部分，前半部分可以根据已有概要数据计算出来，后半部分无法直接计算。为此，我们将后半部分继续展开，得到如下：

$$\sum_{i=l+1}^{L-1} SED_i(Q, B) = \sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} (\|d_{i,j}^Q\|^2 + \|d_{i,j}^C\|^2 - 2d_{i,j}^Q \cdot d_{i,j}^C) \quad (4.10)$$

公式 4.10 右半部分含有 3 个元素，第一个元素为查询轨迹 Q 剩下概要数据的和，该值可根据已有概要数据计算出来，计算方法为 $SSQ - \sum_{i=0}^l \sum_{j=0}^{2^i-1} \|d_{i,j}^Q\|^2$ ，其中 SSQ 为 Q 所有系数的累加和。远程结点只需一开始就把 SSQ 发给所有远程结点即可。同理，第二个元素可在远程结点计算出来。难点在于对第三个关于系数内积累加和的计算。本文的做法是使用两次柯西—施瓦茨不等式估计其区间。第一

次使用具体过程如下：

$$-2 \sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^Q\| \cdot \|d_{i,j}^C\| \leq \sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} -2d_{i,j}^Q \cdot d_{i,j}^C \leq 2 \sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^Q\| \cdot \|d_{i,j}^C\| \quad (4.11)$$

接着我们对 $\sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^Q\| \cdot \|d_{i,j}^C\|$ 进行再次使用柯西-施瓦茨不等式放缩：

$$\sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^Q\| \cdot \|d_{i,j}^C\| \leq \sqrt{\sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^Q\|^2} \cdot \sqrt{\sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^C\|^2} \quad (4.12)$$

结合公式4.11和4.12, 我们得到如下完整的对 $\sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} -2d_{i,j}^Q \cdot d_{i,j}^C$ 计算出如下上、下界。

$$\begin{aligned} & -2 \sqrt{\sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^Q\|^2} \cdot \sqrt{\sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^C\|^2} \\ & \leq \sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} -2d_{i,j}^Q \cdot d_{i,j}^C \leq 2 \sqrt{\sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^Q\|^2} \cdot \sqrt{\sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^C\|^2} \quad (4.13) \end{aligned}$$

最终，我们对欧氏距离获得如下上、下界：

$$HLB_l(Q, C) = \sum_{i=0}^l SED_i(Q, C) + \sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} (\|d_{i,j}^Q\|^2 + \|d_{i,j}^C\|^2) + S_0(Q, C) \quad (4.14)$$

$$\begin{aligned} & -2 \sqrt{\sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^Q\|^2} \cdot \sqrt{\sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^C\|^2} \\ HUB_l(Q, C) &= \sum_{i=0}^l SED_i(Q, C) + \sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} (\|d_{i,j}^Q\|^2 + \|d_{i,j}^C\|^2) + S_0(Q, C) \\ & + 2 \sqrt{\sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^Q\|^2} \cdot \sqrt{\sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^C\|^2} \quad (4.15) \end{aligned}$$

进一步的，我们提出如下两个说明本文所提上、下界会随着概要数据的增加而越来越紧。

定理 4.3.2. HLB 会随着粒度的概要数据粒度的增加而逐渐上升，即 $HLB_l \leq$

HLB_{l+1} 。

证明. 我们的策略是证明 $HLB_{l+1} - HLB_l \geq 0$ 。我们首先将其左半部分展开：

$$HLB_{l+1} - HLB_l = SED_{l+1}(\mathcal{Q}, \mathcal{C}) - \sum_{j=0}^{2^{l+1}-1} (\|d_{l+1,j}^{\mathcal{Q}}\|^2 + \|d_{l+1,j}^{\mathcal{C}}\|^2) + 2\left(\sqrt{\sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^{\mathcal{Q}}\|^2} \cdot \sqrt{\sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^{\mathcal{C}}\|^2} - \sqrt{\sum_{i=l+2}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^{\mathcal{Q}}\|^2} \cdot \sqrt{\sum_{i=l+2}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^{\mathcal{C}}\|^2}\right)$$

接着, 我们将 $SED_{l+1}(\mathcal{Q}, \mathcal{C})$ 展开得到 $SED_{l+1}(\mathcal{Q}, \mathcal{C}) = \sum_{j=0}^{2^{l+1}-1} (\|d_{l+1,j}^{\mathcal{Q}}\|^2 + \|d_{l+1,j}^{\mathcal{C}}\|^2 - 2d_{l+1,j}^{\mathcal{Q}} \cdot d_{l+1,j}^{\mathcal{C}})$ 。我们的问题变为证明如下不等式成立。

$$\begin{aligned} \sum_{j=0}^{2^{l+1}-1} d_{l+1,j}^{\mathcal{Q}} \cdot d_{l+1,j}^{\mathcal{C}} &\leq \sqrt{\sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^{\mathcal{Q}}\|^2} \cdot \sqrt{\sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^{\mathcal{C}}\|^2} \\ &\quad - \sqrt{\sum_{i=l+2}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^{\mathcal{Q}}\|^2} \cdot \sqrt{\sum_{i=l+2}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^{\mathcal{C}}\|^2} \end{aligned} \quad (4.16)$$

对于不等式 4.16 的左半部分我们有 $\sum_{j=0}^{2^{l+1}-1} d_{l+1,j}^{\mathcal{Q}} \cdot d_{l+1,j}^{\mathcal{C}} \leq \sqrt{\sum_{j=0}^{2^{l+1}-1} \|d_{l+1,j}^{\mathcal{Q}}\|^2} \cdot \sqrt{\sum_{j=0}^{2^{l+1}-1} \|d_{l+1,j}^{\mathcal{C}}\|^2}$ 。所以我们的目标变为证明 $\sqrt{\sum_{j=0}^{2^{l+1}-1} \|d_{l+1,j}^{\mathcal{Q}}\|^2} \cdot \sqrt{\sum_{j=0}^{2^{l+1}-1} \|d_{l+1,j}^{\mathcal{C}}\|^2}$ 小于不等式 4.16 的右半部分。为方便表示, 我们令 $x = \sum_{j=0}^{2^{l+1}-1} \|d_{l+1,j}^{\mathcal{Q}}\|^2$, $y = \sum_{j=0}^{2^{l+1}-1} \|d_{l+1,j}^{\mathcal{C}}\|^2$, $\alpha = \sum_{i=l+2}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^{\mathcal{Q}}\|^2$, $\beta = \sum_{i=l+2}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^{\mathcal{C}}\|^2$ 。则不等式 4.16 等价于：

$$\sqrt{x \cdot y} + \sqrt{\alpha \cdot \beta} \leq \sqrt{(\alpha + x) \cdot (\beta + y)} \quad (4.17)$$

我们将如上不等式两边平方得到如下不等式：

$$2\sqrt{x \cdot y \cdot \alpha \cdot \beta} \leq \alpha \cdot y + \beta \cdot x \quad (4.18)$$

根据算数-几何平均不等式, 我们得不等式 4.18 成立。原问题得证。 \square

定理 4.3.3. HUB 会随着粒度的概要数据粒度的增加而逐渐下降, 即 $HUB_l \geq HUB_{l+1}$ 。

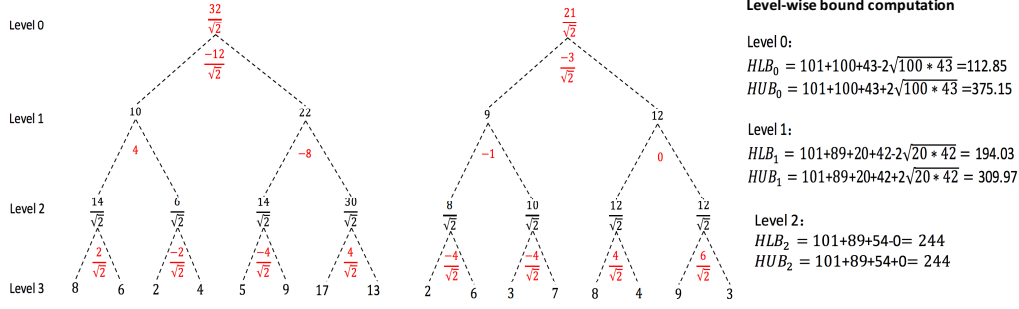


图 4.5: 欧氏距离上、下界示例

证明. 我们的策略是证明 $HUB_{l+1} - HUB_l \leq 0$. 我们首先将其左半部分展开:

$$HUB_{l+1} - HUB_l = SED_{l+1}(\mathcal{Q}, \mathcal{C}) - \sum_{j=0}^{2^{l+1}-1} (\|d_{l+1,j}^{\mathcal{Q}}\|^2 + \|d_{l+1,j}^{\mathcal{C}}\|^2) -$$

$$2\left(\sqrt{\sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^{\mathcal{Q}}\|^2} \cdot \sqrt{\sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^{\mathcal{C}}\|^2} - \sqrt{\sum_{i=l+2}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^{\mathcal{Q}}\|^2} \cdot \sqrt{\sum_{i=l+2}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^{\mathcal{C}}\|^2}\right)$$

接着, 我们将 $SED_{l+1}(\mathcal{Q}, \mathcal{C})$ 展开得到 $SED_{l+1}(\mathcal{Q}, \mathcal{C}) = \sum_{j=0}^{2^{l+1}-1} (\|d_{l+1,j}^{\mathcal{Q}}\|^2 + \|d_{l+1,j}^{\mathcal{C}}\|^2 - 2d_{l+1,j}^{\mathcal{Q}} \cdot d_{l+1,j}^{\mathcal{C}})$. 我们的问题变为证明如下不等式成立。

$$\sum_{j=0}^{2^{l+1}-1} d_{l+1,j}^{\mathcal{Q}} \cdot d_{l+1,j}^{\mathcal{C}} \geq \sqrt{\sum_{i=l+2}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^{\mathcal{Q}}\|^2} \cdot \sqrt{\sum_{i=l+2}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^{\mathcal{C}}\|^2}$$

$$- \sqrt{\sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^{\mathcal{Q}}\|^2} \cdot \sqrt{\sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^{\mathcal{C}}\|^2} \quad (4.19)$$

由于 $\sum_{j=0}^{2^{l+1}-1} d_{l+1,j}^{\mathcal{Q}} \cdot d_{l+1,j}^{\mathcal{C}} \geq -\sqrt{\sum_{j=0}^{2^{l+1}-1} \|d_{l+1,j}^{\mathcal{Q}}\|^2} \cdot \sqrt{\sum_{j=0}^{2^{l+1}-1} \|d_{l+1,j}^{\mathcal{C}}\|^2}$, 所以我们的问题变为证明 $-\sqrt{\sum_{j=0}^{2^{l+1}-1} \|d_{l+1,j}^{\mathcal{Q}}\|^2} \cdot \sqrt{\sum_{j=0}^{2^{l+1}-1} \|d_{l+1,j}^{\mathcal{C}}\|^2}$ 大于公式4.19右半部分。与定理4.3.2证明类似, 我们令 $x = \sum_{j=0}^{2^{l+1}-1} \|d_{l+1,j}^{\mathcal{Q}}\|^2$, $y = \sum_{j=0}^{2^{l+1}-1} \|d_{l+1,j}^{\mathcal{C}}\|^2$, $\alpha = \sum_{i=l+2}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^{\mathcal{Q}}\|^2$, $\beta = \sum_{i=l+2}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^{\mathcal{C}}\|^2$. 则不等式4.19等价于:

$$-\sqrt{x \cdot y} \geq \sqrt{\alpha \cdot \beta} - \sqrt{(\alpha + x) \cdot (\beta + y)} \quad (4.20)$$

不等式经过变换可得到不等式4.17. 因此可得不等式4.19成立, 原问题得证. \square

算法 6 *ED-FTB*在协调者结点

```

/* ED-FTB 协调者结点函数接口实现 */
coordinatorInit( $\mathcal{Q}, \mathcal{R}$ )
1:  $l \leftarrow -1$ ;
2:  $HQ \leftarrow$  Haar wavelet coefficients of  $\mathcal{Q}$ ;           # 待查询轨迹获取哈尔小波系数
3:  $SSQ \leftarrow \sum_{i=0}^{n-1} \|HQ_i\|^2$ ;                 # 所有系数的平方和
4: sendToRemoteSites( $\mathcal{R}, SSQ$ );

generateInfo()
1:  $l \leftarrow l + 1$ ;
2: return  $\hat{Q}_l$ ;                                         # 返回第  $l$  层哈尔小波系数

```

图4.5介绍了使用概要数据得到越来越紧的上、下界，并直至逼近到真实距离的过程。其使用的数据与图4.4一样。当查询对象的第0层系数（数据量为1）发送给远程结点后，远程结点可以对其局部时间序列 \mathcal{C} 根据公式4.14, 计算得到上界和下界分别为 375.15 和 112.85。当第1层系数（数据量为2）发过去后得到更紧的上界和下界分别为 309.97 和 194.03。当第2层系数（即最后一层）发过去后，可以发现此时的上、下界等于原始序列间的欧氏距离值。

4.4 基于欧氏距离的查询算法:ED-FTB

4.4.1 ED-FTB 算法实现

在上一节，我们已经利用多粒度哈尔小波系数设计了逐步变紧的欧氏距离上、下界。这使得在 FTB 框架中插入欧氏距离成为可能。本节将详细介绍将欧氏距离跟 FTB 相结合的查询算法 ED-FTB。ED-FTB 维持了 FTB 框架的主要结构，只对本章第一节所介绍的接口进行了具体实现。

算法6介绍了协调者节点上的函数接口的实现方法。在 **coordinatorInit** 函数中，我们对待查询轨迹进行哈尔小波变换，并计算出其所有系数的平方和。接着将该平方和发送给所有远程结点。远程结点在将来将会利用该值进行上、下界计算。由于 FTB 框架是迭代式由粗到细的通信计算框架，在每轮的迭代中会将某一层的概要数据（哈尔小波系数）发送给远程结点。因此，在初始化过程中我们用 l 来记录当前已经发送到哪一层的数据，并将 l 初始化为 -1 以便从第0层开始。此外，我

算法 7 *ED-FTB*在远程结点

```

/* ED-FTB 远程结点函数接口实现 */
remoteInit( $TS_r, S_r$ )    集合  $S_r$  维护本地候选的界特征
1: if 第一次接收查询 then
2:   for all  $C$  in  $TS_r$  do
3:      $HC \leftarrow$  Haar wavelet coefficients of  $C$ ;           # 候选轨迹获取哈尔小波系数
4:      $SSC \leftarrow \sum_{i=0}^{n-1} \|HC_i\|^2$ ;                     # 所有系数的平方和
5:   for all  $C$  in  $TS_r$  do
6:      $C\_BF \leftarrow \langle C, 0, \infty \rangle$ ;                 # 为  $C$  初始化界特征
7:      $S_r \leftarrow S_r \cup C\_BF$ ;
8:    $SSQ \leftarrow$  getFromCoordinator();
updateBounds( $S_r, m$ )     $m$  为第  $i$  层哈尔小波系数
1: for all  $C$  in  $TS_r$  do
2:    $a \leftarrow \sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^Q\|^2$ ;  $b \leftarrow \sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} \|d_{i,j}^C\|^2$ ;
3:    $tmp \leftarrow a + b + S_0(Q, C)$ ;
4:    $lb \leftarrow tmp - 2\sqrt{ab}$ ;  $ub \leftarrow tmp + 2\sqrt{ab}$ ;
5:   if  $lb < gkub$  &&  $(lb \leftarrow lb + \sum_0^l SED_i(Q, C)) < gkub$  then
6:     将轨迹  $C$  的界特征的下界更新为  $lb$ ;
7:   if  $ub < gkub$  &&  $(ub \leftarrow ub + \sum_0^l SED_i(Q, C)) < gkub$  then
8:      $ub \leftarrow ub + \sum_0^l SED_i(Q, C)$ ;
9:     将轨迹  $C$  的界特征的上界更新为  $ub$ ;
10:  else
11:    将  $C$  对应界特征从  $S_r$  中移除;

```

们在 `generateInfo` 中准备将要发送的下一层小波系数，以便协调者结点发送。

接着，算法7介绍运行在远程结点的函数。对于 `RemoteInit` 函数，若远程结点是第一次接收查询，则会为每条候选轨迹进行哈尔小波变换并计算系数的平方和。若不是，则为该结点所包含的每条轨迹初始化界特征信息（下界初始化为 0，上界初始化为正无穷），并接收查询轨迹的系数平方和。在查询执行过程中 `UpdateBounds` 函数根据接收到的小波系数 m ，为界特征集合 S_r 中的每个界特征更新上、下界。一个直接的实现方式为根据上、下界的计算公式，计算出这两个值。但由于查询过程中的主要计算开销就是对所有候选更新界特征。所以，降低该过程的计算开销很有必要。

为降低更新界特征的计算开销，本文引入了在更新界过程中进行剪枝的思想。在介绍此方法前，我们再次回顾下我们的下界。我们的下界计算主要包含三个算

子: 首先是 $\sum_{i=0}^l SED_i(\mathcal{Q}, \mathcal{C})$, 其计算复杂度为 $O(2^l)$ 。其次是 $\sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} \|\mathbf{d}_{i,j}^{\mathcal{Q}}\|^2$, 由于该部分可以通过 $SSQ - \sum_{i=0}^l \sum_{j=0}^{2^i-1} \|\mathbf{d}_{i,j}^{\mathcal{Q}}\|^2$ 计算, 而且 $\sum_{i=0}^{l-1} \sum_{j=0}^{2^i-1} \|\mathbf{d}_{i,j}^{\mathcal{Q}}\|^2$ 的值已知。故该部分的计算复杂度也为 $O(2^l)$ 。同理, 第三个 $\sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} \|\mathbf{d}_{i,j}^{\mathcal{C}}\|^2$ 计算复杂度也为 $O(2^l)$ 。除这三个算子之外, 剩余部分的计算复杂度为 $O(1)$ 。

基于以上分析, 我们设计了在更新下界的同时进行剪枝的下界更新函数 (见算法7: `UpdateBounds` 函数)。具体的做法是我们首先计算出上述第二和第三两个算子, 并根据这两个算子的值计算出下界中除 $\sum_{i=0}^l SED_i(\mathcal{Q}, \mathcal{C})$ 以外部分的值。若此时计算出来的值已超过 $gkub$, 则根据剪枝原理, 无需继续求解下界和上界, 可以直接将该候选删除。若仍小于 $gkub$, 则计算出完整的下界, 并判断此时下界值是否小于 $gkub$, 若小于则停止计算上界并将该轨迹从候选中移除。当更新完下界后, 我们先计算出上界中除 $\sum_{i=0}^l SED_i(\mathcal{Q}, \mathcal{C})$ 以外部分的值。若该部分值超过 $gkub$, 则说明该轨迹的上界对选取新一轮的全局最小的 k 个上界并没有帮助, 无需计算该轨迹的具体上界值。否则, 计算出具体的上界并更新该对象的界特征。此时, 需要注意的是, 在 FTB 框架算法的第 7-9 行中, 我们仅需传递上界小于 $gkub$ 的 k 个最小上界。若个数不足 k 个, 则只传递那些小于 $gkub$ 的上界值。

4.4.2 ED-FTB 算法性能分析

在分析前, 我们先介绍对比算法 LEEWAVE 和 LEEWAVE-CL。LEEWAVE 算法也是迭代式算法, 在它的每次迭代中, 远程结点根据获取的概要数据, 为每个候选计算如下两个算子: $\sqrt{\sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} \|\mathbf{d}_{i,j}^{\mathcal{C}}\|^2}$ 和 $S_0(\mathcal{Q}, \mathcal{C}) + \sum_{i=0}^l SED_i(\mathcal{Q}, \mathcal{C})$ 。然后协调者节点获取这些参数并为每个候选计算上、下界, 并利用界特征进行过滤。然后, 将候选列表发给对应远程结点。LEEWAVE 方法与本文方法的最大不同就是, 它在协这结点进行界特征的计算和过滤, 而 ED-FTB 是在所有远程结点进行。LEEWAVE-CL 是在 LEEWAVE 算法的基础上使用了本文所提下界 (比 LEEWAVE 算法所使用下界更紧)。接下来, 我们将从时间和通信两个方面对 ED-FTB 和 LEEWAVE-CL 进行对比分析。LEEWAVE 和 LEEWAVE-CL 在这两个方面复杂度一样, 只是由于

LEEWAVE-CL 使用了更紧的下界，因此其在这两方面性能都比 LEEWAVE 更优。在此之前，我们使用如下标记：(i) 使用 $|C_i|$ 和 $|CS_i|$ 分别表示第 i ($i \geq 0$) 次迭代前候选轨迹的数量和包含候选轨迹的远程结点数量。由于采用相同的上、下界，这两个算法的迭代次数相同，我们假定迭代次数为 λ ($\lambda \leq \log_2 n$)。

时间复杂度：不考虑系统初始化时对所有候选进行哈尔小波变换的时间开销，ED-FTB 和 LEEWAVE-CL 两者的时间开销来自迭代式计算时更新上、下界，所以他们的时间复杂度一样。计算复杂度最坏的情况就是所有候选都集中在一个结点上，则更新上、下界的时间复杂度为 $O(\sum_{i=0}^{\lambda-1} d \cdot |C_i| \cdot 2^i)$ 。此外，由于 k 值一般较小，算法运行过程中涉及到的查找局部最小的 k 个上界耗时较低，可以忽略。总的来说，ED-FTB 和 LEEWAVE-CL 的时间复杂度均为 $O(\sum_{i=0}^{\lambda-1} d \cdot |C_i| \cdot 2^i)$ 。但 ED-FTB 由于在各个结点进行界特征的计算，且采用了边计算边过滤的策略，故其实际计算开销低于 LEEWAVE-CL。

通信复杂度：ED-FTB 和 LEEWAVE-CL 的通信开销主要集中在迭代式计算过程中。ED-FTB 在第 i 轮迭代时，需要花费 $O(d \cdot 2^i \cdot |CS_i| + |C_i|)$ 代价以将第 i 层概要数据发送给候选结点，并至多接收 $|C_i|$ 个上界（用于计算全局的剪枝阈值）。在 λ 次迭代后，总的通信开销为 $O(\sum_{i=0}^{\lambda-1} (d \cdot 2^i \cdot |CS_i| + |C_i|))$ 。LEEWAVE-CL 在第 i 轮迭代需要花费 $O(|CS_i| \cdot (d \cdot 2^i + |C_{i+1}|))$ 来讲概要数据和所有候选的列表发送给所有候选结点。此外，还需要花费 $O(d \cdot |C_{i+1}|)$ 代价以接收每个候选轨迹的两个算子值以便在协调者结点计算出上、下界。所以，当 λ 次迭代后，其总的通信开销为 $O(|CS_i| \cdot (d \cdot 2^i + |C_{i+1}|) + d \cdot |C_{i+1}|)$ 。对比两个算法的开销值，我们可以发现 ED-FTB 的能节省更多通信开销。

4.5 实验分析

本节将在真实世界的轨迹数据上评估本章节所提出的算法。本小节首先介绍使用的真实世界轨迹数据集及实验设置。接着从分别验证了算法的有效性和可扩展性。

数据集	时间	轨迹数	轨迹长度	大小 (Gb)
<i>T-Small</i>	2013.10.1-10.7	1 万	>4,096	0.8
<i>T-Big</i>	2013.11.1-12.31	100 万	>4,096	76

表 4.1: 实验数据集描述

4.5.1 实验设置

本章工作在实验中使用北京出租车数据集，该数据集已被广泛应用于轨迹数据分析。该数据集采集了北京市三万多辆出租车在 2013 年 10 月份到 12 月份三个月内的行驶 GPS 轨迹。每个 GPS 轨迹点包含的数据维度较多，我们仅选取了位置(经度和纬度)、时间、速度和角度这五个维度的值，并进行了归一化预处理。我们从该数据集中截取了两个子数据集：*T-Small* 和 *T-Big* 以分别验证算法的有效性和可扩展性。数据集描述见表4.1，其中 *T-Small* 截取了 10 月 1 至 7 号间上午 8 点到 10 点的轨迹数据，并从中选出最长的 1 万条轨迹进行分析。*T-Big* 截取了 11 月 1 号至 12 月 31 号间每天上午 8 至 10 点和下午 5 至 7 点两个时间段的 1 百万条轨迹数据。为满足实验需求，这两个数据集的每条轨迹长度均超过 4,096。

在本章的实验中,我们将比较 ED-FTB 和 LEEWAVE-CL 算法的性能。LEEWAVE-CL 算法在已有 LEEWAVE 算法的基础上使用了本文所提供的下界。两个算法均用 JAVA 实现,并运行在一个包含 12 个结点的 Spark 集群上。每个结点包含 8 核英特尔 E5335 2.0 GHz 中央处理器和 16GB 的内存。我们在 *T-Small* 数据集上验证了算法的有效性,在 *T-Big* 数据集上验证了算法的可扩展性。

4.5.2 算法有效性

首先,我们通过观察迭代过程中每轮结束后的剩余候选数来研究 ED-FTB 算法的剪枝效果。在该组实验中,我们将 M 设置为 10,000,并研究了不同长度轨迹剪枝的效果。图 4.6 介绍了前 8 轮迭代中每轮迭代后的候选数。我们可以看出候选数在前 5 轮迭代中下降的较快且已经过滤掉绝大多数候选。这说明了我们的上、下界收敛速度较快,具有较好的剪枝效果。此外,我们可以发现短轨迹在前几轮中候

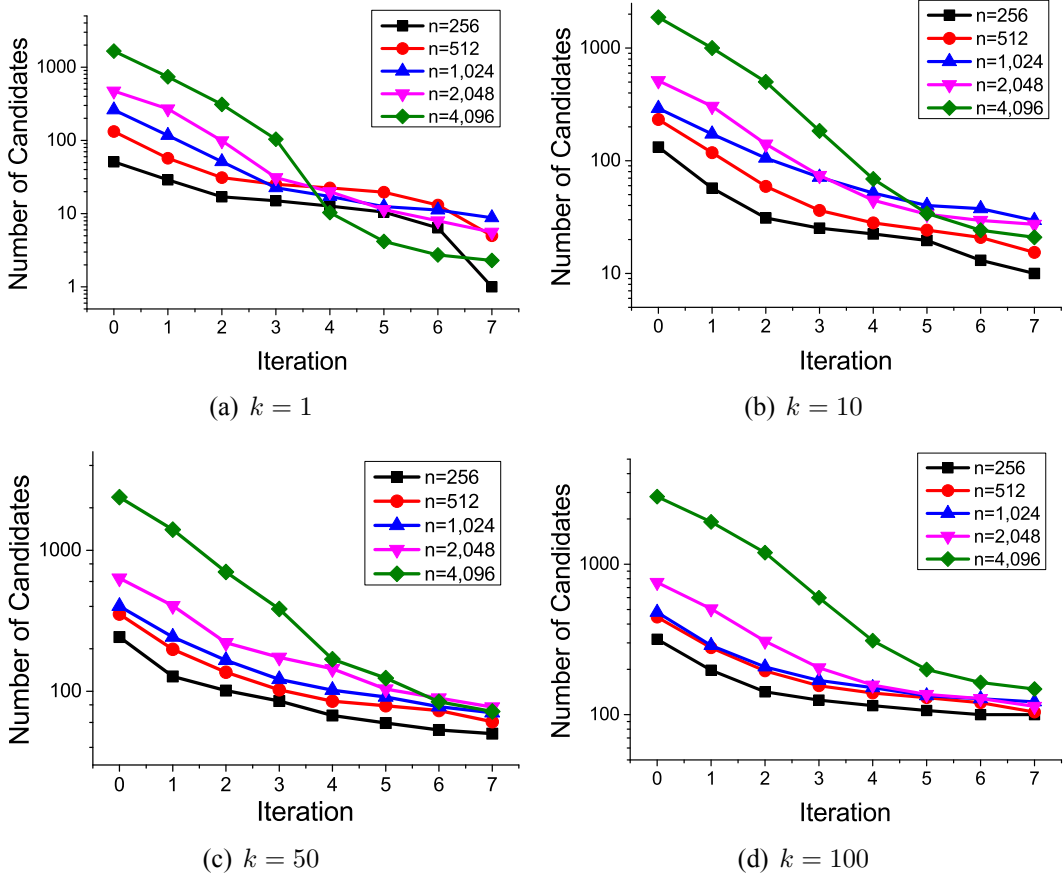


图 4.6: ED-FTB 剪枝效果图

选数下降的比长轨迹快。这是由于相同数据量的概要数据下短轨迹比长轨迹包含更多的原始轨迹的信息。最后，对比不同的 k 值，我们发现 k 值越小，每轮所剩候选数越少。即 k 越小，剪枝效果越好。这是由于 k 值越小，我们所获得的全局第 k 小上界值就越小。而无论 k 值如何选取，每个候选的下界值不变。因此，越小的全局上界能通过下界剪枝掉更多的候选。同时，由于实际应用中 k 值通常都是一个很小的数。因而我们的算法能往往取得较好的效果。

然后，我们对比研究了 ED-FTB 的剪枝效果和 ED-FTB 使用 LEEWAVE 中的下界后的剪枝效果，以分析下界对剪枝效果的影响。在该组实验中，我们将 M 值设置为 10,000，轨迹长度设为 1,024。图4.7对比展示了不同 k 值下，不同下界对剪枝的影响。其中 LEEWAVE 代表 ED-FTB 算法使用 LEEWAVE 算法中的下界后的结果。从图中，我们可以看出由于本文所提出的下界比已有算法的下界更紧，导

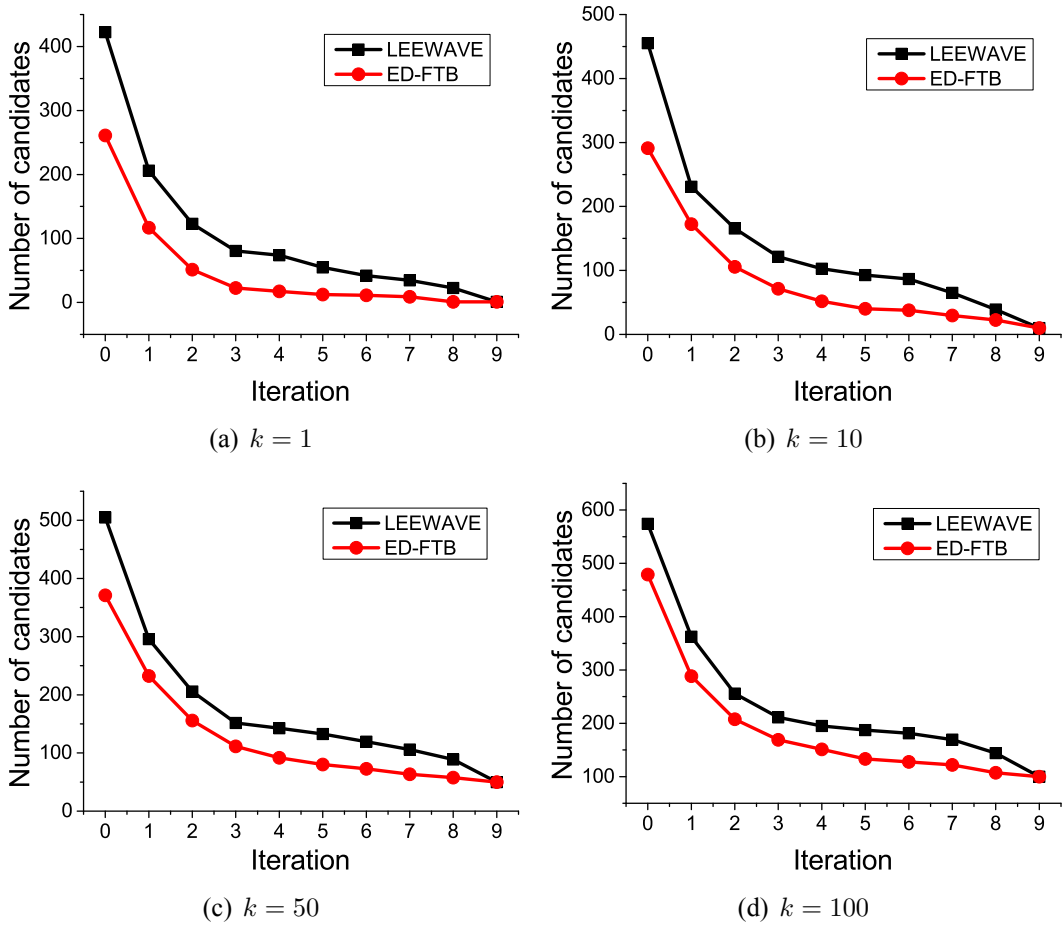


图 4.7: 下界对剪枝结果的影响

致每轮迭代结束后 ED-FTB 所剩的候选数更少。这说明了本文所提下界的优越性。此外我们可以发现 k 值越小，两条曲线间的间隔越大。更能说明本文下界所取得的优越性越明显。

接着，我们使用相同的界，对比了分析了本文所提查询策略与 LEEWAVE 所用查询策略对通信性能的影响。我们从协调者节点出发，研究了两种策略下它发送和接收的数据量以及总的数据通信开销。图 4.8 介绍了当 M 取 50 和 100， k 取 1 和 1000 下的实验结果。图中 LEEWAVE-CL 代表结合了 LEEWAVE 的查询策略与本文的界特征后的算法。从图中我们可以发现，协调者结点在两种策略下，发送的数据量差别不大且都较小。这是因为，两个策略都有减少协调者数据发送的数据量这一共同目标。且两者使用的使用概要数据剪枝的策略很好的达成了这一目

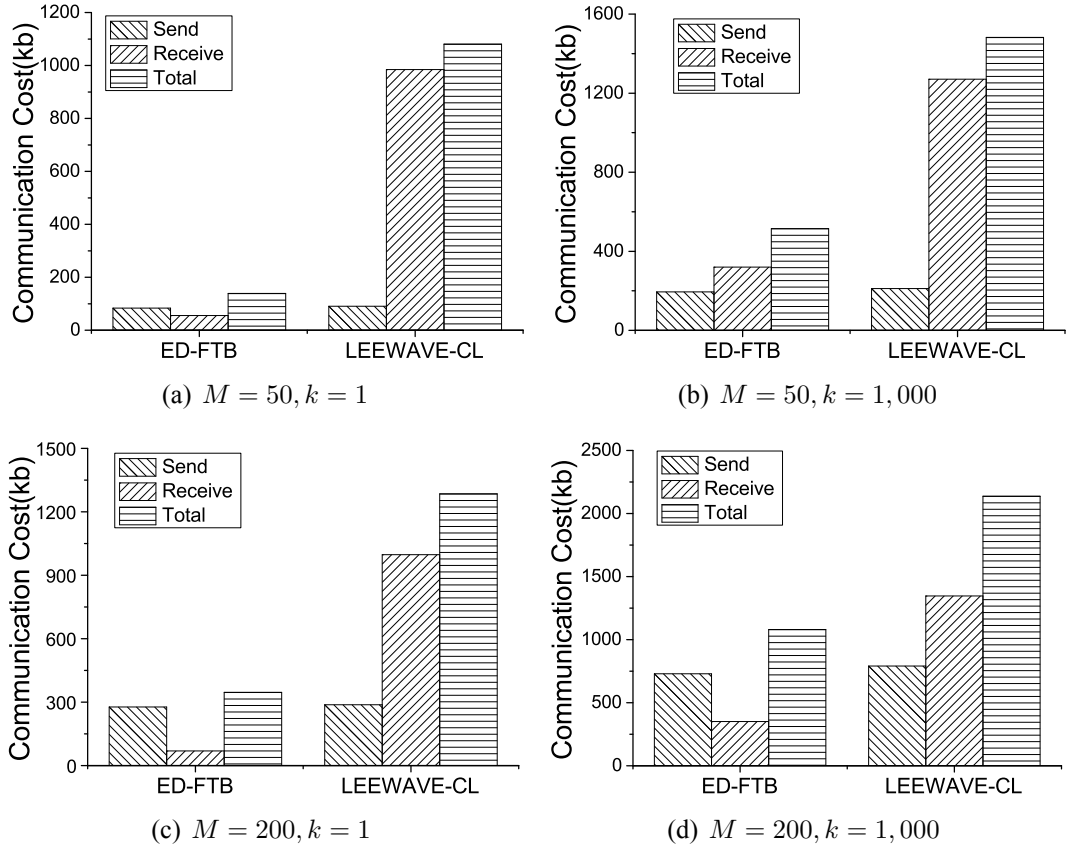
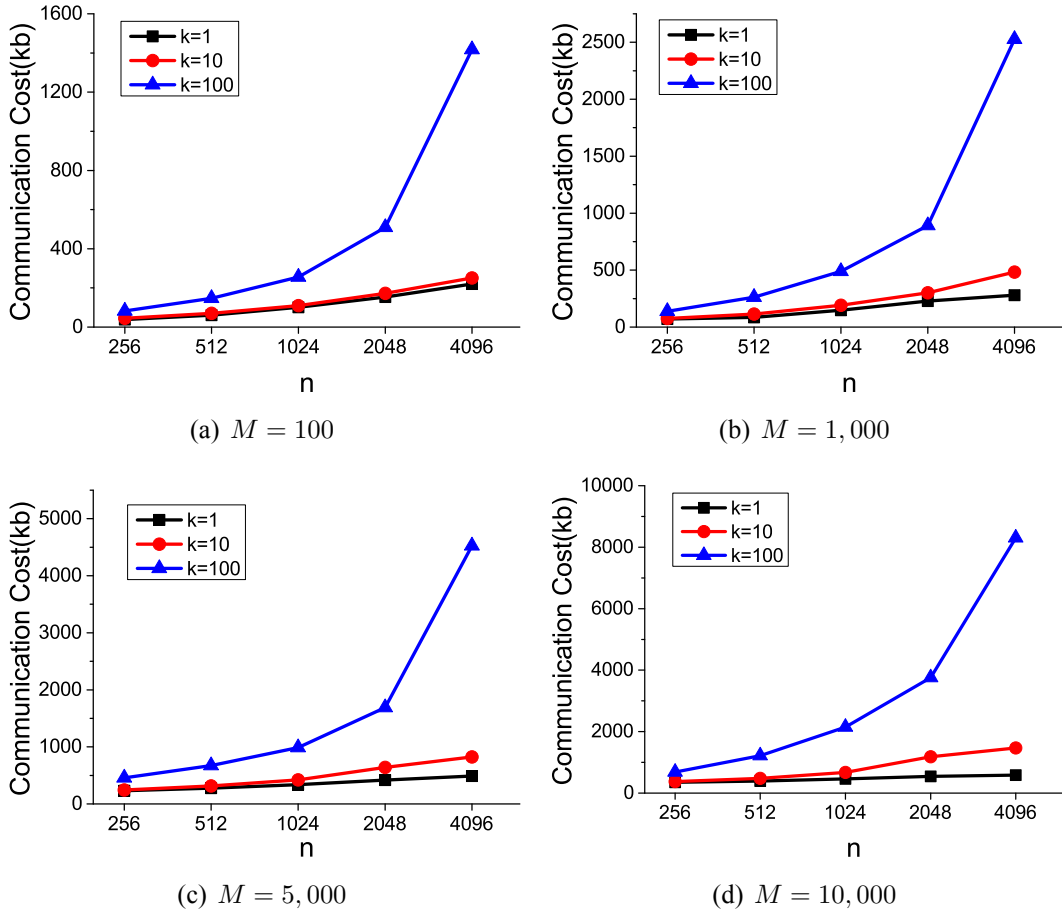


图 4.8: 算法执行策略详细比较

标。但接收的数据量在两种策略下差别很大。ED-FTB 所用的查询策略是在每个远程结点剪枝数据。每轮迭代完后远程结点仅需将所计算出来的局部最小的 k 个上界返回。而 LEEAVE-CL 需要将所有候选的跟计算界信息的两个算子返回。当远程结点所包含的数据越多，则两者的区别越大。特别地，当 k 值较小时（图 4.8(a) 和 4.8(c)），ED-FTB 所收到的数据不到 LEEWAVE-CL 的十分之一。

其次，我们研究了 n , k 和 M 对 ED-FTB 算法通信开销的影响。在该组实验中，我们将 n 的值从 256 变化到 4,096，将 k 的值从 1 变化到 100。图 4.9 展示了不同 M 值下（ M 从 100 编化到 10,000），ED-FTB 算法通信开销随参数变化的结果。首先，我们看出随着 k 值的增加，通信开销逐步增加。这是由于 k 值越大，每轮所剩候选数越多，候选所在的远程结点也就越多。从而每轮需要将概要数据发送到更多的候选结点中。其次，随着 n 值的增加通信开销也在增加。这是由于长轨迹需要发送更多的数据才能达到与短轨迹相同的剪枝效果。最后，对比三幅图可以发现，

图 4.9: n, k 和 M 对 ED-FTB 算法通信开销的影响

随着远程结点数据的增加, 通信开销也在增加。这是由于在每轮迭代中, 远程结点增加后, 候选轨迹会分布到更多的结点中。因而, 算法需要将概要数据发送到更多的结点中。因而总的通信开销也会增加。但是, 对于极端情况 $M = 10,000$ 时, 此时每个远程结点仅包含一条轨迹时我们的通信开销仍不到 10Mb, 远小于直接方法的开销 (约 640Mb)。因此, ED-FTB 算法具有较高的通信节约性能。

接下来, 我们将 ED-FTB 与 LEEWAVE 和 LEEWAVE-CL 进行通信性能比较。在该组实验中我们将轨迹长度 n 从 256 变化到 4,096。图4.10展示了当 k 取值为 1, 10 和 100 下的结果。从该组图中我们可以看出, 本文所提 ED-FTB 算法性能最好, LEEWAVE-CL 其次, 最差的是 LEEWAVE。LEEWAVE-CL 比 LEEWAVE 好的原因是它使用了本文所提下界进行剪枝。本文所提下界由于比 LEEWAVE 中的更加紧凑, 因而剪枝效果更好。最终导致通信开销更低。而 ED-FTB 算法比 LEEWAVE-

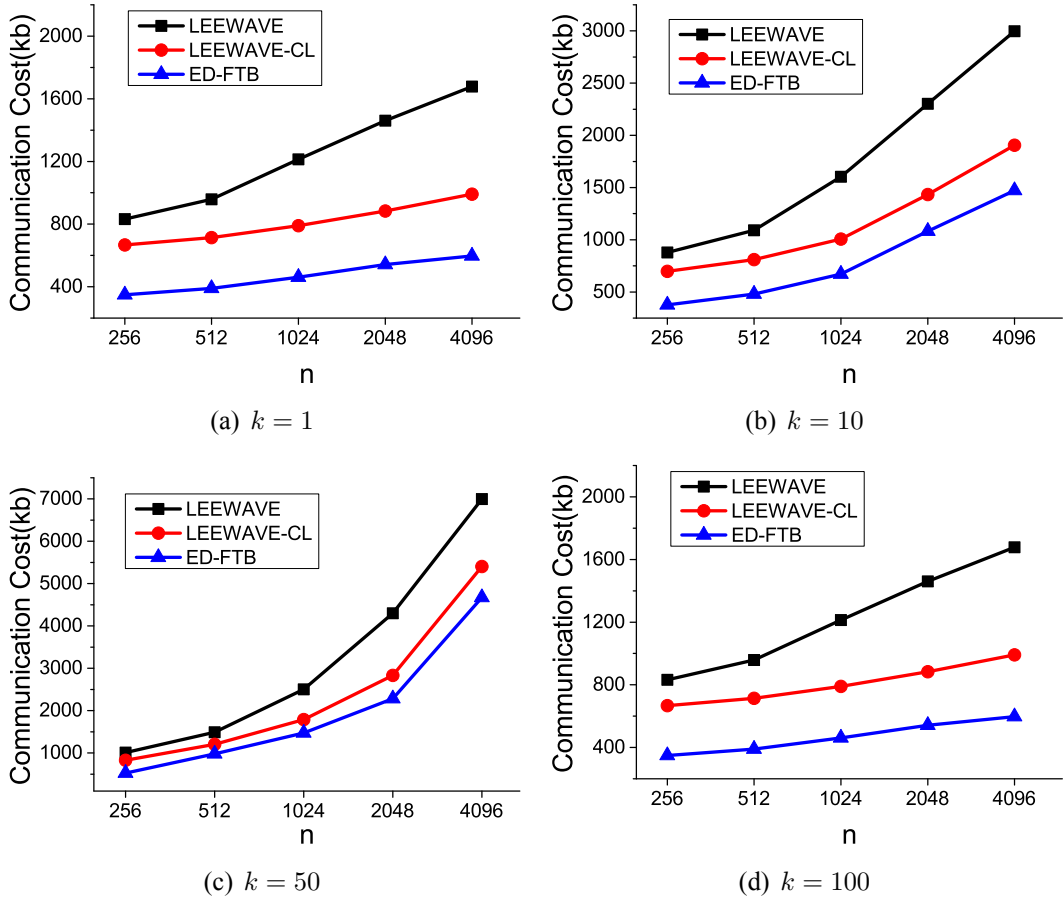


图 4.10: 算法性能比较

CL 好的原因是，本文采用的是在远程结点剪枝的策略，只需发送概要数据。而 LEEWAVE 和 LEEWAVE-CL 均是采用在协调者结点剪枝的策略，除了发概要数据还要发送和接收其他额外数据。因而开销更高。最后，我们可以发现，随着轨迹长度 n 和 k 值的增加，EDD-FTB 算法所节省的通信开销逐步增大。这进一步说明了本文算法的优越性。

4.5.3 算法可扩展性

本小节我们将在 *T-Big* 数据集上从时间和通信两个角度研究了 ED-FTB 算法的可扩展性。首先，考虑了轨迹数量 N （即数据量大小）和轨迹长度 n 对可扩展性的影响。在该组实验中，我们将 k 设置为 1， M 设置为 10,000。我们将轨迹长度从 256 到 4096 进行指数级变化，同时将轨迹数量从 10 万到 1 百万进行线性增加。

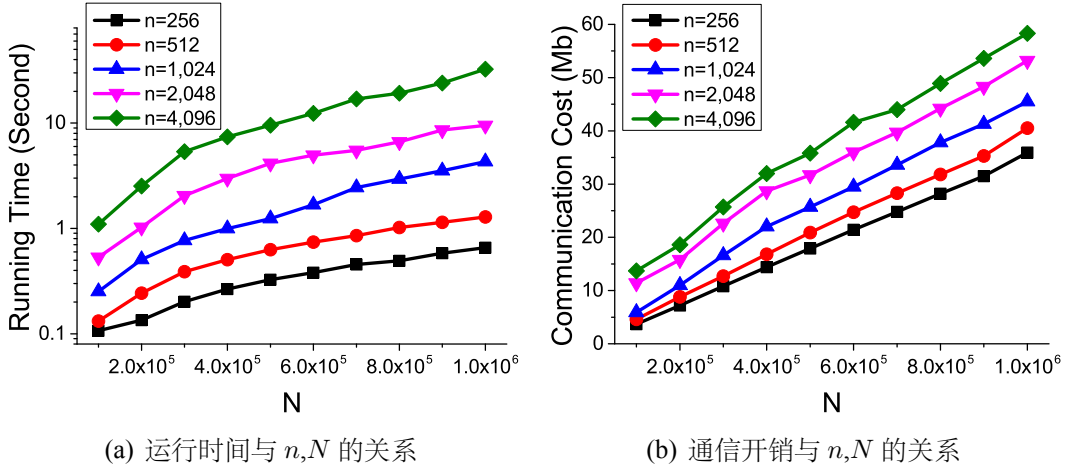


图 4.11: ED-FTB 算法可扩展性 (T-Big 数据集)

图4.11分别对这两个角度的结果进行了介绍，其中图4.11(a)介绍了时间性能受 n 和 N 的影响 (时间包系统初始化过程中对数据进行哈尔小波转换的开销)。我们可以看到对于任意长度的轨迹数据集，ED-FTB 算法的运行时间都随着轨迹数据量的增加而线性增加。此外，随着轨迹长度的指数增加，算法的运行时间也指数增加，即运行时间与轨迹长度呈一定的线性关系。这一结果反映了由于 ED-FTB 的剪枝效果较好，导致迭代结束后，所剩候选数较少。即我们对 ED-FTB 算法时间复杂度分析部分的 N' 较少。结合以上两点，我们可以看出 ED-FTB 算法运行时间随着数据集的大小而线性变换，因此运行效率具有较好的可扩展性。

4.6 本章小结

本章节介绍了利用距离上、下界进行剪枝的思想实现分布式 k 近邻查询的同时，以降低通信开销。本章节，首先设计了利用多粒度概要数据不断缩紧距离上、下界，并利用界特征进行剪枝的 FTB 策略。接着，针对欧式距离，使用哈尔小波系数作为多粒度概要数据并提出了不断变紧的距离上、下界。其次，将欧氏距离应用于 FTB 策略并设计了 ED-FTB 算法。最后，真实数据集上的实验结果表明所提方法优于现有方案。

第五章 基于下界剪枝的分布式 k 近邻查询

本章主要介绍针对仅使用距离下界进行的剪枝策略来实现分布式 k 近邻查询，并达到降低通信开销的目的。首先，章节5.1介绍了使用概要数据计算距离下界并进行逐步剪枝的策略。接着，章节5.2针对动态时间弯曲距离，设计了基于包围信封的多粒度概要数据，并设计了逐渐变紧的距离下界。然后，章节5.3介绍了算法 DTW-FLB，以处理当使用动态时间弯曲距离作为距离度量准则时的查询。接着，章节5.4展示了 DTW-FLB 算法的有效性和可扩展性。最后，章节 5.5小结本章的研究内容。

5.1 基于下界的逐步剪枝策略

上一章介绍了同时使用距离上、下界进行剪枝的策略，但存在着某些距离仅能根据概要数据计算出下界的情况。为此，本章针对这些仅能设计出下界的距离函数设计了仅使用下界进行剪枝的策略。在此之前首先，介绍下界特征：

定义 5.1.1 (下界特征 (Lower Bound Feature, LBF)). 二元组 $\langle id, lb \rangle$ 构成了下界特征，记录了一条轨迹的标识 (id)，以及它与待查询轨迹的距离下界 (lb)。

5.1.1 策略设计原理

针对那些不能同时获得上界的距离度量准则，FLB 在使用由粗到细粒度的概要数据以不断获取更加紧凑的距离下界的同时，还不断计算一个越来越紧的全局阈值以剪枝数据。假设与待查询轨迹距离第 k 小的距离值为 gk ，若某轨迹的下界大于该值，则该轨迹不会成为最终结果。但 gk 值很难一开始就计算出来。为此，本文的做法是以最小的代价（通信和时间开销）计算 gk 的上界 θ ，并不断将 θ 逼近 gk 。然后，在每轮迭代中，用 θ 来剪枝候选。若某条轨迹计算出来的下界大于 θ ，则该轨迹从候选中移除。 θ 就是 FLB 用来剪枝的全局阈值。需要区别的是，FTB 框

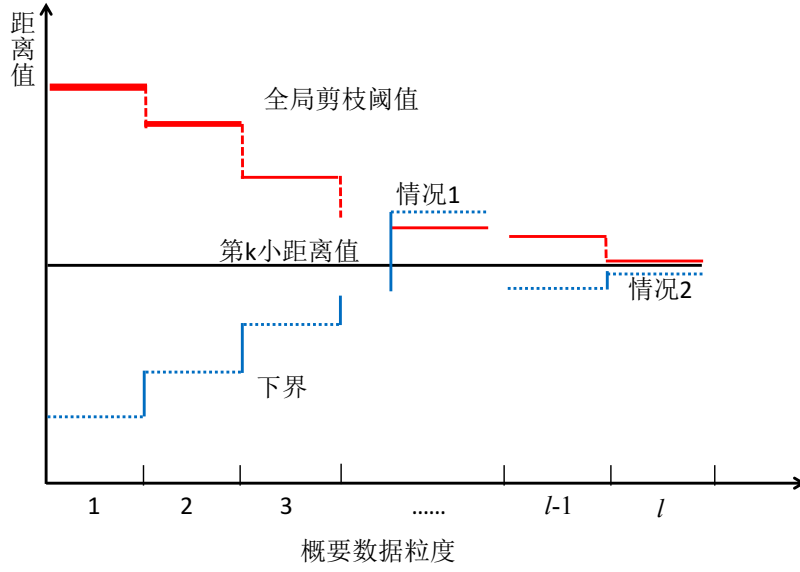


图 5.1: 利用下界和全局阈值的剪枝过程

架是利用距离上界值进行剪枝。

图5.1示例介绍了 FLB 框架对某候选轨迹的剪枝过程。图中，中间黑色直线表示查询轨迹与所有轨迹距离中第 k 小的距离值 (gk)。蓝色分段虚线介绍了轨迹距离下界随着概要数据粒度增加，该下界值越来越逼近其真实距离的过程。红色分段实线介绍了全局阈值 θ 由高到低不断逼近 gk 值。随着概要数据的不断精细，下界和 θ 在不断逼近真实值的过程中会出现两种情况：第一种情况就是不断逼近过程中，轨迹的下界大于阈值 θ 。此时，我们可以将轨迹剪枝掉。第二种情况就是当最细粒度的概要数据后，轨迹距离下界仍小于阈值 θ 。此时，我们将对该轨迹继续保留为候选以待进一步分析。值得注意的是，当根据最细粒度的概要数据进行剪枝后，若所保留的候选数仍超过 k 。此时，FLB 框架将向包含这些候选的远程结点发送原始轨迹，远程结点则可以计算出真实的距离值并找出最终的 k 个距离最近的轨迹。

5.1.2 策略实施方案

根据上节介绍，我们知道使用 FLB 框架也需要满足三点条件：(i) 协调者节点能根据给定的距离函数，设计出满足多粒度属性的概要数据；(ii) 远程结点能

算法 8 FLB 之协调者结点**输入:** 查询轨迹 Q , 结果集大小 k ;**输出:** k 条最相似轨迹的 ID;

```

1:  $\mathcal{R} \leftarrow$  远程结点集合,  $\mathcal{R}' \leftarrow null$ ;
2: coordinatorInit( $Q, \mathcal{R}$ ); # 协调者节点初始化
3: while true do
4:    $Info \leftarrow$  generateInfo(); # 生成待发送概要数据
5:   if  $Info \neq null$  then
6:     sendToRemoteSites( $\mathcal{R}, Info$ ); # /*(6-13行): 更新全局阈值  $\theta$ */
7:      $GLBS \leftarrow$  getFromRemoteSites( $\mathcal{R}$ ); # 从远程结点获取局部 top- $k$  下界
8:      $gklbs \leftarrow argmin_{\tau}(|x \in GLBS, x.lb < \tau| \geq k)$ ; # 选取下界最小的  $k$  条轨迹
9:      $\mathcal{R}'' \leftarrow$  the remote sites that contain  $gklbs$ ; # 找出包含上述轨迹的远程结点
10:    sendToRemoteSites( $\mathcal{R}'' - \mathcal{R}', \langle Q, gklbs \rangle$ ); # 对未收到  $Q$  的结点进行处理
11:    sendToRemoteSites( $\mathcal{R}'' \cap \mathcal{R}', \langle null, gklbs \rangle$ ); # 对已收到  $Q$  的结点进行处理
12:     $sv \leftarrow sv \cup getFromRemoteSites(\mathcal{R}'')$ ;
13:     $\theta \leftarrow argmin_{\tau}(|x \in sv, x < \tau| \geq k)$ ; # 选取第  $k$  小相似度值
14:    sendToRemoteSites( $\mathcal{R}, \theta$ );
15:     $\mathcal{R}' \leftarrow \mathcal{R}' \cup \mathcal{R}''$ ;
16:    /*Step 2: 剪枝候选远程结点 */
17:     $CSS \leftarrow$  getFromRemoteSites( $\mathcal{R}$ ); # /*(17-22行): 剪枝并反馈 */
18:     $\mathcal{R} \leftarrow \{x.r | x \in CSS, x.|S_r| > 0\}$ ;
19:     $sum \leftarrow \sum_{x \in CSS} x.|S_r|$ ;
20:    if  $sum = k$  then
21:      sendToRemoteSites( $CSS, finish$ ); # top- $k$  结果已经找到, 发结束信号
22:      return getFromRemoteSites( $\mathcal{R}$ );
23:    else
24:      sendToRemoteSites( $\mathcal{R} - \mathcal{R}', \langle null, Q \rangle$ ); # /*(24-28行): 结果提炼 */
25:      sendToRemoteSites( $\mathcal{R} \cap \mathcal{R}', \langle null, null \rangle$ );
26:       $CSet \leftarrow$  getFromRemoteSites( $\mathcal{R}$ );
27:       $gkSim \leftarrow argmin_{\tau}(|x \in CSet, x.dis < \tau| \geq k)$ ;
28:      return  $\{x.id | x \in CSet, x.dis \leq gkSim\}$ ;

```

根据概要数据计算出距离下界, 且能保证其不断逼近真实值。(iii) 不断逼紧的全局阈值。基于以上分析, 本文设计的 FLB 框架分为两个部分, 一部分是运行在协调者结点的算法8, 另一部分是运行在所有远程结点的算法9。在框架运行过程中, 这两个部分互相通信并协调工作。整个过程, 可以分为四个阶段: 初始化阶段, 全局阈值计算阶段, 剪枝阶段和结果提炼阶段。下面将对这四个阶段分别进行介绍:

初始化阶段: 协调者结点在此阶段执行多种操作, 如获取远程结点的列表, 概要数据计算以及可能预发送一些信息。在此过程中, 我们使用 \mathcal{R} 来表示远程结点

算法 9 FLB 之远程结点**输入:** 局部轨迹集合 TS_r ;**输出:** 属于 top- k 结果集的轨迹 ID;

```

1:  $S_r \leftarrow$  局部轨迹下界特征集;
2: remotelnit( $TS_r, S_r$ );
3: while true do
4:    $m \leftarrow$  getFromCoordinator(); # 从协调者结点获取信息
5:   if  $m$  is Info then
6:      $S_r \leftarrow$  UpdateLowerBound( $S_r, m$ );
7:      $\hat{lb} \leftarrow \operatorname{argmin}_\tau(|x \in S_r, x.lb < \tau| \geq k)$ ;
8:      $S' \leftarrow \{x \mid x \in S_r, x.lb \leq \hat{lb}\}$ ;
9:     SendToCoordinator( $\langle r, S' \rangle$ );
10:  else if  $m = \langle Q/null, gklbs \rangle$  then
11:     $sv \leftarrow \{distance(Q, x) \mid x \in gklbs\}$ ;
12:    SendToCoordinator( $sv$ );
13:  else if  $m = \theta$  then
14:     $S_r \leftarrow \{x \mid x \in S_r, x.lb \leq \theta\}$ ; # 局部剪枝
15:    SendToCoordinator( $\langle r, |S_r| \rangle$ );
16:    if  $|S_r| = 0$  then
17:      return ; # 结点不包含候选, 则结束
18:  else if  $m = finish$  then
19:     $ids \leftarrow \{x.id \mid x \in S_r\}$ ; # 收到结束信号, 则返回候选 ID 并结束
20:    SendToCoordinator( $ids$ );
21:    return;
22:  else
23:    SendToCoordinator( $\{\langle x.id, distance(x, Q) \rangle \mid x.id \in S_r\}$ );
24:  return ;

```

的集合, \mathcal{R}' 表示那些已经接收了查询轨迹的远程结点 (算法 8:1-2 行)。与之对应的, 远程结点在该阶段的初始化主要工作是为保存在本地数据集合 TS_r 中轨迹初始化下界特征并将结果存在局部候选集 S_r 中。此外, 它也会接收协调者结点发过来的数据 (算法 2:1-2行)。

全局阈值计算阶段: 该阶段, 我们的主要目标是通过迭代通信交互, 逐步获取更加精确的全局阈值 θ 。在每次迭代中, 协调者结点首先发送概要数据给所有候选远程结点 (算法 8 6行)。候选远程结点在接收到概要数据后, 更新每个候选轨迹下界特征。接着选取局部下界最小的 k 个下界特征返回给协调者节点 (算法 9: 6-9行)。其次, 协调者选取包含最小的 k 个下界特征集合 $gklbs$, 及包含 $gklbs$ 的

远程结点集合 \mathcal{R}'' (算法 8: 7-8行)。然后, 它将待查询轨迹 Q 发送给 \mathcal{R}'' 中的远程结点。在发送前, 算法将 \mathcal{R}'' 中的结点分为两类: 第一类是未接收过 Q 的结点, 此时我们需将 Q 以及 $gklbs$ 中属于该结点的轨迹发送给对应的结点。第二类是已接收过 Q 的结点, 此时, 仅需将对应结点的轨迹标识发送给该远程结点 (算法 8: 10-11)。当候选远程结点接收到信息这样的成对信息后, 针对本地出现在 $gklbs$ 中的轨迹计算出距离值并返回给用户 (算法 9: 11-12 行)。需要注意的是, 对于接收成对信息的结点, 我们并没有对本地所有候选轨迹计算真实值。其原因是尽管使用全部候选计算出来的距离能有更好的逼近第 k 小距离值, 但这样做可能导致时间开销较大, 尤其是当数据集中在这些节点上时。远程结点在接收到那些包含 $gklbs$ 的结点发送过来的距离值后, 选择最小的值作为全局阈值, 并将该值发送给候选结点 (算法 8: 13-14行)。

剪枝阶段: 该阶段主要利用 θ 和下界进行剪枝。当候选远程结点接收到 θ 后, 将会将下界小于所接收的阈值的轨迹从候选集中删除。剪枝完毕后, 该结点会将所剩候选数发送给协调者结点。此外, 当该结点不再包含任何候选时, 则停止运行 (算法 9: 13-17行)。协调者结点在接收到所有候选远程结点发送过来的候选数后, 计算出总的剩余候选数。若所剩候选总数正好为 k 个, 则说明所有候选已经找出。协调结点向候选远程结点发结束信号并等待接收最终结果的标识 (算法 8: 21-22行)。候选远程在接收到结束信号后, 则将本地候选的标识发送给协调者结点 (算法 9: 18-21行)。若所剩候选总数仍超过 k , 则迭代执行第二和本阶段任务直到 k 个候选被找出或概要数据发送完毕。

结果提炼阶段: 由于距离下界不一定能最终能逼近到真实距离值, 且全局阈值不一定能逼近到第 k 小距离值。所以存在着当概要数据发送完毕, 所剩候选数仍超过 k 个的情况。此时, 需要对剩下的候选进行甄别。FLB 框架的做法是将查询轨迹 Q 发送到剩下的候选结点中, 并对所有剩余候选计算真实的距离值以找出最终的 k 个结果 (算法 9: 24-28行)。需要注意的是在发送 Q 时, 若某候选已经接收过 Q 则无需再次发送。

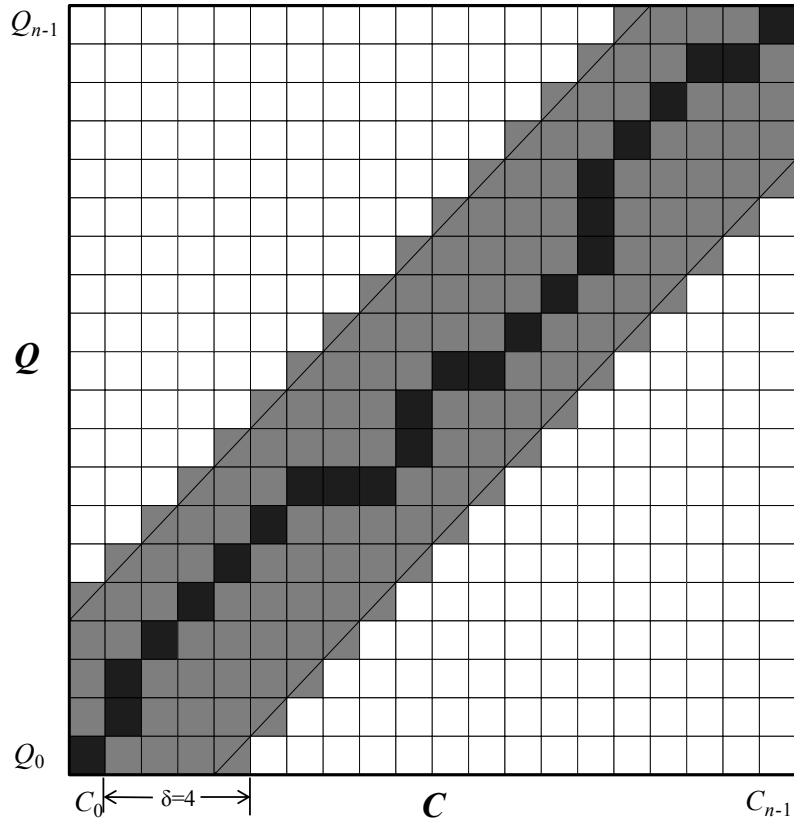


图 5.2: 带约束的动态时间弯曲

5.2 基于动态时间弯曲距离的概要数据

5.2.1 基于动态时间弯曲距离的轨迹相似度量

动态时间弯曲 DTW 距离是度量轨迹等时间数据距离的重要度量方式 [131], 为此, 本章节将研究如何将该距离引入到我们所提框架中。给定查询轨迹 Q 表示为 $Q = \{q_0, q_1, \dots, q_{n-1}\}$, 以及一条候选轨迹 C 表示为 $C = \{c_0, c_1, \dots, c_{n-1}\}$, 其中 n 为轨迹长度, 每个轨迹点来自 d 维空间, 即 $q_i, c_i \in R^d$ 。为使用动态时间弯曲距离匹配 Q 和 C , 我们构建了一个 $n \times n$ 的代价矩阵 $cost$ 。其中第 i 行第 j 列的元素记录了使用欧氏距离匹配 q_i 和 c_j 两点之间的代价。动态时间弯曲距离的目的就是发现一条从如图5.2所示左下角出发到右上角结束的代价最小的路线/弯曲路径。一条弯曲路径需满足如下 3 个特别的约束:

- **边界条件:** 路径从代价矩阵的 $[0, 0]$ 号格子出发并在 $[n-1, n-1]$ 号格子结束。

- **连续性:** 如果路径中相邻的两次权重值分别取自代价矩阵的 $[i, j]$ 和 $[i', j']$ 两个单元, 那么这两个单元的位置关系必满足如下条件: $i' - i \leq 1$ 和 $j' - j \leq 1$ 。这个约束使得路径在扩展过程中只能从一个格子扩展到其半径为 1 的格子。
- **单调性:** 如果路径中相邻的两次权重值分别取自代价矩阵的 $[i, j]$ 和 $[i', j']$ 两个单元, 那么这两个单元的位置关系还须满足如下条件: $i' - i \geq 0$ 和 $j' - j \geq 0$ 。这个约束使得路径搜索过程中只能前进不能后退。

根据以上描述, 我们形式化给出动态时间弯曲距离的另一种更加直观的定义方式:

$$DTW(Q, \mathcal{C}) = \min \sum_{k=1}^K w_k \quad (n \leq K < 2n) \quad (5.1)$$

其中, w_k 是弯曲路径 \mathcal{W} 的第 k 个元素, 其值来源于代价矩阵的第 $[i, j]$ 个格子。 $\mathcal{W} = \{w_1, w_2, \dots, w_K\}$, 代表了一条匹配 Q 和 \mathcal{C} 弯曲路径。

$$DTW = \gamma(n-1, n-1) \quad (5.2)$$

$$\begin{aligned} \gamma(i, j) = SED(\mathbf{q}_i, \mathbf{c}_j) + \min(\gamma(i-1, j-1), \gamma(i-1, j), \\ \gamma(i, j-1)) \quad s.t. \quad |i-j| \leq \delta \end{aligned} \quad (5.3)$$

动态时间弯曲距离的计算过程可通过动态规划算法实现, 因为其计算过程可以看作公式5.2最优子结构的递归定义。其中 δ 是一个全局约束, 用于限制查询路径的可以偏离对角线的范围。图5.2中显示了当 $\delta = 4$ 时, 灰色部分即为弯曲路径的查询范围。使用全局约束的原因有两个: (i) 该约束可以提高查询的效率, δ 使得动态时间弯曲距离的计算复杂度由原来的 n^2 降低为 n^2/δ 。 (ii) 该约束还能有效抑制无效的弯曲路径的产生。比如, 能够防止一轨迹的一小段匹配到另一轨迹的一大段上情况。

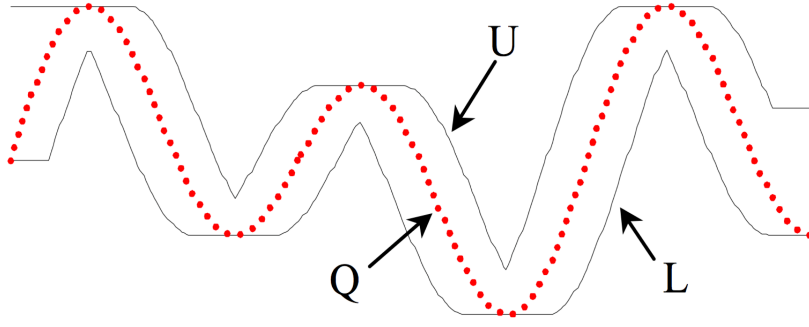


图 5.3: 包围信封

5.2.2 基于包围信封的概要数据

包围信封 (Bounding Envelope, BE) 概念来源于时间序列数据分析, 其目的是使用一上一下两条边界曲线来包住给定的时间序列。如图5.3所示, 红色虚线为一条时间序列, U 和 L 两条线分别代表了这条时间序列的上界和下界。给定时间序列, 构造其包围信封的方法有许多。由于本文的目的是利用包围信封构造概要数据, 故所构造的包围信封包含的数据量要低。此外, 所构造的包围信封还应具有多粒度特性, 以适应 FLB 框架的需求。

为此, 引入基于时间序列划分 (Time Series Segmentation) 的包围信封。其做法是给定一条时间序列 \mathcal{S} , 将其均匀划分成多个等长度的片段。我们使用 $s_l^p = [lb, ub, len]$ 来表示其中一条片段 $\{q_i, q_{i+1}, \dots, q_j\}$ 的最小包围矩形 (Minimum Bounding Rectangle, MBR)。其中 l 代表划分的粒度 (粒度越细, 每个片段的长度越短), p 代表该片段在当前划分下所对应的位置。 lb 和 ub 分别代表该片段的最小值和最大值, len 代表该片段的长度。由 $\{s_l^1, \dots, s_l^{n'}\}$ 所构成的列表代表了该次划分所对应的包围信封, 其用 \mathcal{S}_l 表示。在第 0 层, 整个时间序列被看做一个划分, 接着将其均匀切成 R 个不相交的相连子片段。这个过程可以一直持续下去, 直到每个片段的长度为 1 截止。

表格5.1介绍了针对给定时间序列 $\mathcal{S} = \{5, 2, 4, 3\}$, 计算其不同粒度包围盒的过程。其中, R 的值设置为 2。即从粗粒度到细粒度切分的过程中, 每次将一个片段均匀划分成 2 个子片段。不失一般性, 接下来的示例和实验室中, 我们都将 R 的

\mathcal{S}_0	$s_0^0 = [2, 5, 4]$			
\mathcal{S}_1	$s_1^0 = [2, 5, 2]$		$s_1^1 = [3, 4, 2]$	
\mathcal{S}_2	$s_2^0 = [5, 5, 1]$	$s_2^1 = [2, 2, 1]$	$s_2^2 = [4, 4, 1]$	$s_2^3 = [3, 3, 1]$
Q	5	2	4	3

 表 5.1: 时间序列 \mathcal{S} 的多粒度包围盒

值设置为 2。值得注意的是，本文采用的划分为均匀划分，而相关文章中所提非均匀划分的方法也可以被使用。此外，由于是均匀划分，我们可以推算出每个片段的长度，故其值无需保存。

5.2.3 满足 DTW 距离约束的包围信封及下界

本节将介绍如何根据包围信封计算动态时间弯曲距离的下界。首先将介绍点的上、下界，然后再介绍如何利用点的范围计算轨迹的下界。令 \mathbf{q} 和 \mathbf{c} 表示两个来自 d 维实数空间的点，其中 \mathbf{c} 的值已知， \mathbf{q} 的值不知道，但知道其在一个 d 维矩形 $\{\mathbf{u}, \mathbf{l}\}$ 里，即 $\forall j \in [0, d), \mathbf{l}^j \leq \mathbf{q}^j \leq \mathbf{u}^j$ 。对 \mathbf{q} 和 \mathbf{c} 之间的实际欧氏距离我们无法计算出来，但可以通过如下引理计算出距离的下界。

引理 5.2.1. 给定 \mathbf{q} 、 \mathbf{c} 以及 \mathbf{q} 的包围矩形 $\{\mathbf{u}, \mathbf{l}\}$ ，我们有 $SED_LB(\{\mathbf{u}, \mathbf{l}\}, \mathbf{c}) \leq SED(\mathbf{q}, \mathbf{c})$ ，其中 $SED_LB(\{\mathbf{u}, \mathbf{l}\}, \mathbf{c})$ 的计算过程如下：

$$SED_LB(\{\mathbf{u}, \mathbf{l}\}, \mathbf{c}) = \sum_{j=0}^{d-1} \begin{cases} (\mathbf{c}^j - \mathbf{u}^j)^2 & \text{if } \mathbf{u}^j \leq \mathbf{c}^j, \\ 0 & \text{if } \mathbf{l}^j \leq \mathbf{c}^j \leq \mathbf{u}^j, \\ (\mathbf{l}^j - \mathbf{c}^j)^2 & \text{if } \mathbf{c}^j \leq \mathbf{l}^j. \end{cases} \quad (5.4)$$

证明. 对于点的任意维度，如第 j 维，我们有如下结论

$$\begin{cases} (\mathbf{u}^j - \mathbf{c}^j)^2 \leq (\mathbf{q}^j - \mathbf{c}^j)^2 & \text{if } \mathbf{u}^j \leq \mathbf{c}^j, \\ 0 \leq (\mathbf{q}^j - \mathbf{c}^j)^2 & \text{if } \mathbf{l}^j \leq \mathbf{c}^j \leq \mathbf{u}^j, \\ (\mathbf{c}^j - \mathbf{l}^j)^2 \leq (\mathbf{q}^j - \mathbf{c}^j)^2 & \text{if } \mathbf{c}^j \leq \mathbf{l}^j. \end{cases} \quad (5.5)$$

由于 $SED_LB(\{\mathbf{u}, \mathbf{l}\}, \mathbf{c})$ 可看做上式左边部分在各个维度的累加, $SED(\mathbf{q}, \mathbf{c})$ 是上式右半部分的累加。故累加所有维度的值后, 我们可以得到 $SED_LB(\{\mathbf{u}, \mathbf{l}\}, \mathbf{c}) \leq SED(\mathbf{q}, \mathbf{c})$ 。 \square

接着介绍如何构建满足动态时间弯曲距离的包围信封。根据包围信封的介绍, 我们知道给定查询轨迹 \mathcal{Q} , 构建包围信封的最重要就是构造轨迹的上、下边界。同时, 考虑到动态时间弯曲距离的全局约束特性。我们使用如下方法构建轨迹的上、下边界:

$$\begin{aligned} \mathcal{U} &= \{\mathbf{u}_0, \mathbf{u}_1, \dots, \mathbf{u}_{n-1}\} \quad , \quad \forall j \mathbf{u}_i^j = \max(\mathbf{q}_{i-\delta}^j : \mathbf{q}_{i+\delta}^j) \\ \mathcal{L} &= \{\mathbf{l}_0, \mathbf{l}_1, \dots, \mathbf{l}_{n-1}\} \quad , \quad \forall j \mathbf{l}_i^j = \min(\mathbf{q}_{i-\delta}^j : \mathbf{q}_{i+\delta}^j) \end{aligned} \quad (5.6)$$

公式5.6中, \mathcal{U} 和 \mathcal{L} 分别代表轨迹的上边界和下边界, δ 为动态时间弯曲距离中的全局约束。根据如上方式计算出来的包围盒, 我们称之为满足动态时间弯曲约束的包围信封 (Dynamic Time Warping constrained Bounding Envelope, DTWBE)。此时, 根据 $DTWBE\{\mathcal{U}, \mathcal{L}\}$ 和 \mathcal{C} , 我们可以给出如下距离定义:

$$LB_Keogh(\{\mathcal{U}, \mathcal{L}\}, \mathcal{C}) = \sum_{i=0}^{n-1} SED_LB(\{\mathbf{u}_i, \mathbf{l}_i\}, \mathbf{c}_i) \quad (5.7)$$

该定义可以看做是原始一维的 LB_Keogh 下界在多维情况下的扩展, 并得到如下下界定理:

定理 5.2.2. 给定轨迹 \mathcal{C} 和待查询轨迹 \mathcal{Q} 满足动态时间弯曲约束的包围信封 $\{\mathcal{U}, \mathcal{L}\}$, 我们有如下结论: $LB_Keogh(\{\mathcal{U}, \mathcal{L}\}, \mathcal{C}) \leq DTW(\mathcal{Q}, \mathcal{C})$ 。

证明. 根据 $LB_Keogh(\{\mathcal{U}, \mathcal{L}\}, \mathcal{C})$ 和 $DTW(\mathcal{Q}, \mathcal{C})$ 的定义, 我们的目标即是证明如下不等式成立。

$$\sum_{i=0}^{n-1} SED_LB(\{\mathbf{u}_i, \mathbf{l}_i\}, \mathbf{c}_i) \leq \sum_{k=1}^K \mathbf{w}_k \quad (5.8)$$

此外我们还有 $n \leq K$, 我们接下来的策略是证明上述不等式左边的每个值都能在

右边找到一个比它大或相等的元素：

$$\begin{aligned}
 SED_LB(\{\mathbf{u}_i, \mathbf{l}_i\}, \mathbf{c}_i) &\leq \mathbf{w}_k \\
 \Leftrightarrow SED_LB(\{\mathbf{u}_i, \mathbf{l}_i\}, \mathbf{c}_i) &\leq \|\mathbf{c}_i^j - \mathbf{q}_x^j\|^2 \\
 \Leftrightarrow \sum_{j=0}^{d-1} SED_LB(\{\mathbf{u}_i^j, \mathbf{c}_i^j\}, \mathbf{c}_i^j) &\leq \sum_{j=0}^{d-1} (\mathbf{c}_i^j - \mathbf{q}_x^j)^2
 \end{aligned} \tag{5.9}$$

其中 x 与 i 满足如下不等式约束 $i - \delta \leq x \leq i + \delta$ 。同时，根据 \mathbf{u}_i 和 \mathbf{l}_i 的定义，我们有 $\forall j \mathbf{l}_i^j \leq \mathbf{q}_x^j \leq \mathbf{u}_i^j$ 。因此，我们进一步的得到如下不等式：

$$\begin{aligned}
 &\begin{cases} (\mathbf{u}_i^j - \mathbf{c}_i^j)^2 \leq (\mathbf{c}_i^j - \mathbf{q}_x^j)^2 & \text{if } \mathbf{u}_i^j \leq \mathbf{c}_i^j, \\ 0 \leq (\mathbf{c}_i^j - \mathbf{q}_x^j)^2 & \text{if } \mathbf{l}_i^j \leq \mathbf{c}_i^j \leq \mathbf{u}_i^j, \\ (\mathbf{c}_i^j - \mathbf{l}_i^j)^2 \leq (\mathbf{c}_i^j - \mathbf{q}_x^j)^2 & \text{if } \mathbf{c}_i^j \leq \mathbf{l}_i^j. \end{cases} \\
 \Leftrightarrow SED_LB(\{\mathbf{u}_i^j, \mathbf{c}_i^j\}, \mathbf{c}_i^j) &\leq (\mathbf{c}_i^j - \mathbf{q}_x^j)^2 \\
 \Leftrightarrow \sum_{j=0}^{d-1} SED_LB(\{\mathbf{u}_i^j, \mathbf{c}_i^j\}, \mathbf{c}_i^j) &\leq \sum_{j=0}^{d-1} (\mathbf{c}_i^j - \mathbf{q}_x^j)^2
 \end{aligned} \tag{5.10}$$

因此，不等式 5.9 成立。原问题得证。 \square

5.2.4 基于多粒度包围信封的下界

为计算上一小节介绍的 LB_Keogh 下界，需要传递 DTWBE，其数据量为 $2 \cdot n$ ，是原始轨迹数据量的两倍。因此，不满足我们降低通信的目标。为此，本文首先设计了基于 DTWBE 的多粒度包围信封。多粒度信封的第 l 层边界计算方式如下：

$$\begin{aligned}
 \hat{\mathcal{U}}_l &= \{\hat{\mathbf{u}}_{l,0}, \hat{\mathbf{u}}_{l,1}, \dots, \hat{\mathbf{u}}_{l,2^l-1}\} \quad , \quad \forall j \hat{\mathbf{u}}_{l,i}^j = \max(\mathbf{u}_{i,2^{L-l}}^j : \mathbf{u}_{(i+1),2^{L-l}-1}^j) \\
 \hat{\mathcal{L}}_l &= \{\hat{\mathbf{l}}_{l,0}, \hat{\mathbf{l}}_{l,1}, \dots, \hat{\mathbf{l}}_{l,2^l-1}\} \quad , \quad \forall j \hat{\mathbf{l}}_{l,i}^j = \min(\mathbf{l}_{i,2^{L-l}}^j : \mathbf{l}_{(i+1),2^{L-l}-1}^j)
 \end{aligned} \tag{5.11}$$

根据公式 5.11，第 l 层的包围信封计算过程可看作将原来的 DTWBE 均匀切分成 2^l 块，然后对每个信封块计算仅使用两个 d 维点来构成新的包围信封。根据对公式 5.11 的观察，我们得到两种结论：(i) 第 l 层的包围信封 $\{\hat{\mathcal{U}}_l, \hat{\mathcal{L}}_l\}$ 数据量是第

$l-1$ 层的两倍；(ii) 第 $l-1$ 层的数据被第 l 层包含。因此，在由粗到细发送包围信封这一概要数据时，除第 0 层外，其他层只需要发送一半的信息量（这一半数据的每个值需要用 0 和 1 标注其位置是在左边还是右边，但位置信息消耗的开销较小可忽略）。那么假定远程结点已获取到查询轨迹 Q 的第 l 层包围信封 $\{\hat{\mathcal{U}}_l, \hat{\mathcal{L}}_l\}$ ，我们可通过如下公式来计算给定候选与 Q 的 DTW 距离下界。

定理 5.2.3. 给定轨迹 Q 的第 l 层包围信封 $\{\hat{\mathcal{U}}_l, \hat{\mathcal{L}}_l\}$ 和候选轨迹 \mathcal{C} ，我们得到如下结论： $LB_HPAA(\{\hat{\mathcal{U}}_l, \hat{\mathcal{L}}_l\}, \bar{\mathcal{C}}_l) \leq DTW(Q, \mathcal{C})$ 。其中 LB_HPAA 的计算公式如下：

$$LB_HPAA(\{\hat{\mathcal{U}}_l, \hat{\mathcal{L}}_l\}, \bar{\mathcal{C}}_l) = 2^{L-l} \cdot \sum_{i=0}^{2^l-1} SED_LB(\{\hat{\mathbf{u}}_{l,i}, \hat{\mathbf{l}}_{l,i}\}, \bar{\mathbf{c}}_{l,i}) \quad (5.12)$$

证明. 从定理 5.2.2 可知 LB_keogh 是原始 DTW 距离的下界。因此，若我们能得到 $LB_HPAA(\{\hat{\mathcal{U}}_l, \hat{\mathcal{L}}_l\}, \bar{\mathcal{C}}_l) \leq LB_Keogh(\{\mathcal{U}, \mathcal{L}\}, \mathcal{C})$ ，则原问题得证。根据 LB_HPAA 的定义，我们可知第 l 层包围信封的每个元素对应着 LB_Keogh 的 s ($s = 2^{L-l}$) 个元素。如果 LB_HPAA 的每个元素满足如下不等式，则原问题得证。

$$s \cdot SED_LB(\{\hat{\mathbf{u}}_{l,i}, \hat{\mathbf{l}}_{l,i}\}, \bar{\mathbf{c}}_{l,i}) \leq \sum_{p=i \cdot s}^{(i+1) \cdot s - 1} SED_LB(\{\mathbf{u}_p, \mathbf{l}_p\}, \mathbf{c}_p) \quad (5.13)$$

根据不等式 5.14，由于 $\forall j, \hat{l}_{l,i}^j \leq l_p^j \leq u_p^j \leq \hat{u}_{l,i}^j$ ，我们有 $SED_LB(\{\hat{\mathbf{u}}_{l,i}, \hat{\mathbf{l}}_{l,i}\}, \mathbf{c}_p) \leq SED_LB(\{\mathbf{u}_p, \mathbf{l}_p\}, \mathbf{c}_p)$ 。

$$\begin{cases} (\hat{u}_{l,i}^j - c_p^j)^2 \leq (u_p^j - c_p^j)^2 & \text{if } u_p^j \leq \hat{u}_{l,i}^j \leq c_p^j, \\ 0 \leq (u_p^j - c_p^j)^2 & \text{if } u_p^j \leq c_p^j \leq \hat{u}_{l,i}^j, \\ 0 \leq 0 & \text{if } l_p^j \leq c_p^j \leq u_p^j, \\ 0 \leq (l_p^j - c_p^j)^2 & \text{if } \hat{l}_{l,i}^j \leq c_p^j \leq l_p^j, \\ (\hat{l}_{l,i}^j - c_p^j)^2 \leq (l_p^j - c_p^j)^2 & \text{if } c_p^j \leq \hat{l}_{l,i}^j \leq l_p^j. \end{cases} \quad (5.14)$$

所以，我们的问题变为求证 $s \cdot SED_LB(\{\hat{\mathbf{u}}_{l,i}, \hat{\mathbf{l}}_{l,i}\}, \bar{\mathbf{c}}_{l,i}) \leq \sum_{p=i \cdot s}^{(i+1) \cdot s - 1} SED_LB(\{\hat{\mathbf{u}}_{l,i}, \hat{\mathbf{l}}_{l,i}\}, \mathbf{c}_p)$ 成立。为简化证明，我们只证明第一个轨迹片段（即 $i = 0$ 时）成立。此外我们使

用 $\hat{\mathbf{u}}, \hat{\mathbf{l}}$ 和 $\bar{\mathbf{c}}$ 来分别表示 $\hat{\mathbf{u}}_{l,0}, \hat{\mathbf{l}}_{l,0}$ 和 $\bar{\mathbf{c}}_{l,0}$ 。对于其他片段，证明方法相同。此时，我们的目标就是证明如下不等式：

$$s \cdot SED_LB(\{\hat{\mathbf{u}}, \hat{\mathbf{l}}\}, \bar{\mathbf{c}}) \leq \sum_{i=0}^{s-1} SED_LB(\{\hat{\mathbf{u}}, \hat{\mathbf{l}}\}, \mathbf{c}_i) \quad (5.15)$$

根据 SED_LB 定义，我们展开不等式5.15。此时，若对于任一维度 j 有如下不等式成立则不等式5.15也成立。

$$s \cdot SED_LB(\{\hat{\mathbf{u}}^j, \hat{\mathbf{l}}^j\}, \bar{\mathbf{c}}^j) \leq \sum_{i=0}^{s-1} SED_LB(\{\hat{\mathbf{u}}^j, \hat{\mathbf{l}}^j\}, \mathbf{c}_i^j) \quad (5.16)$$

对于以上不等式的右半部分， \mathbf{c}_i^j 和 $\{\hat{\mathbf{u}}^j, \hat{\mathbf{l}}^j\}$ 之间存在 3 种大小关系。不失一般性，我们假设这 3 种关系均出现在这 s 对元组中。为此，我们首先将所有 \mathbf{c}_i 进行重新排序，使得排序后的结果满足：(i) 从 \mathbf{c}_0 到 \mathbf{c}_{p-1} 中的元素，满足 $\mathbf{c}_i^j \leq \hat{\mathbf{u}}^j$ ；(ii) 从 \mathbf{c}_p 到 \mathbf{c}_{q-1} 中的元素，满足 $\hat{\mathbf{l}}^j \leq \mathbf{c}_i^j \leq \hat{\mathbf{u}}^j$ ；(iii) 从 \mathbf{c}_q 到 \mathbf{c}_{s-1} 中的元素满足 $\mathbf{c}_i^j \leq \hat{\mathbf{l}}^j$ ，其中 $0 \leq p \leq q \leq s$ 。那么根据 $\bar{\mathbf{c}}$ 的定义，我们得到下述结论：

$$\sum_{i=0}^{p-1} (\mathbf{c}_i^j - \bar{\mathbf{c}}^j) = \sum_{i=p}^{s-1} (\bar{\mathbf{c}}^j - \mathbf{c}_i^j) = \sum_{i=p}^{q-1} (\bar{\mathbf{c}}^j - \mathbf{c}_i^j) + \sum_{i=q}^{s-1} (\bar{\mathbf{c}}^j - \mathbf{c}_i^j) \quad (5.17)$$

然后，我们考虑不等式5.16的左半部分。其根据 $\hat{\mathbf{u}}^j$ 和 $\bar{\mathbf{c}}^j$ 之间的大小关系可分为 3 种情况。(i) $\hat{\mathbf{u}}^j \leq \bar{\mathbf{c}}^j$ 的情况，此时左半部分的值为 $s(\bar{\mathbf{c}}^j - \hat{\mathbf{u}}^j)^2$ 。此时，对于右半部分我们有

$$\begin{aligned} \sum_{i=0}^{s-1} (\{\hat{\mathbf{u}}^j - \hat{\mathbf{l}}^j\}, \mathbf{c}_i^j)^2 &= \sum_{i=0}^{p-1} (\mathbf{c}_i^j - \hat{\mathbf{u}}^j)^2 + \sum_{i=q}^{s-1} (\hat{\mathbf{l}}^j - \mathbf{c}_i^j)^2 \\ &\geq \frac{1}{s-q+p} \left(\sum_{i=0}^{p-1} (\mathbf{c}_i^j - \hat{\mathbf{u}}^j) + \sum_{i=q}^{s-1} (\hat{\mathbf{l}}^j - \mathbf{c}_i^j) \right)^2 \quad (AM - GM) \\ &\geq \frac{1}{s-q+p} \left(\sum_{i=p}^{q-1} (\bar{\mathbf{c}}^j - \mathbf{c}_i^j) + p(\bar{\mathbf{c}}^j - \hat{\mathbf{u}}^j) + \sum_{i=q}^{s-1} (\hat{\mathbf{l}}^j - 2\mathbf{c}_i^j + \bar{\mathbf{c}}^j) \right)^2 \\ &\geq \frac{1}{s-q+p} \left(\sum_{i=p}^{q-1} (\bar{\mathbf{c}}^j - \hat{\mathbf{u}}^j) + p(\bar{\mathbf{c}}^j - \hat{\mathbf{u}}^j) + \sum_{i=q}^{s-1} (\bar{\mathbf{c}}^j - \hat{\mathbf{l}}^j) \right)^2 \end{aligned}$$

$$= \frac{s^2}{s-q+p}(\bar{\mathbf{c}}^j - \hat{\mathbf{u}}^j)^2 \geq s(\bar{\mathbf{c}}^j - \hat{\mathbf{u}}^j)^2 \quad (5.18)$$

因此，第一种情况下原问题得证。(ii) $\bar{\mathbf{c}}^j \leq \hat{\mathbf{l}}^j$ 的情况，此时证明过程与第一种情况类似。(iii) $\hat{\mathbf{l}}^j \leq \bar{\mathbf{c}}^j \leq \hat{\mathbf{u}}^j$ 的情况，此时不等式5.16成立，因其左半部分值为0，而有半部分是在大于0。结合以上3种情况，原问题得证。□

以上定理说明了给定查询轨迹的第 l 层包围信封和候选误轨迹差树的第 l 层均值，就能为原始轨迹和候选轨迹计算出动态时间弯曲距离的下界。同时该下界的计算复杂度是线性的，远小于距离本身二次的复杂度。因此，通过计算下界可为剪枝候选提供帮助。此外，为配合 FLB 框架，我们需要证明， LB_HPAA 能随着粒度的增加而逐渐变紧。为此我们给出如下定理：

定理 5.2.4. LB_HPAA 下界能随着查询轨迹包围信封层次的增加而逐渐变紧。即： $LB_HPAA(\{\hat{\mathcal{U}}_l, \hat{\mathcal{L}}_l\}, \bar{\mathcal{C}}_l) \leq LB_HPAA(\{\hat{\mathcal{U}}_{l+1}, \hat{\mathcal{L}}_{l+1}\}, \bar{\mathcal{C}}_{l+1})$ 。

证明. 根据多粒度包围信封的计算法方式我们可知，当包围信封从第 l 层转到第 $l+1$ 层时，其每个元素分为两个元素用于分别表示左右两边的最值。所以，我们只要证明如下不等式即可：

$$\begin{aligned} 2 \cdot SED_LB(\{\hat{\mathbf{u}}_{l,i}, \hat{\mathbf{l}}_{l,i}\}, \bar{\mathbf{c}}_{l,i}) &\leq SED_LB(\{\hat{\mathbf{u}}_{l+1,2i}, \hat{\mathbf{l}}_{l+1,2i}\}, \bar{\mathbf{c}}_{l+1,2i}) \\ &+ SED_LB(\{\hat{\mathbf{u}}_{l+1,2i+1}, \hat{\mathbf{l}}_{l+1,2i+1}\}, \bar{\mathbf{c}}_{l+1,2i+1}) \end{aligned} \quad (5.19)$$

为简化问题，我们分别使用 $\hat{\mathbf{u}}$, $\hat{\mathbf{u}}_L$ 和 $\hat{\mathbf{u}}_R$ 来分别表示 $\hat{\mathbf{u}}_{l,i}$, $\hat{\mathbf{u}}_{l+1,2i}$ 和 $\hat{\mathbf{u}}_{l+1,2i+1}$ 。并且使用 $\bar{\mathbf{c}}$, $\bar{\mathbf{c}}_L$ 和 $\bar{\mathbf{c}}_R$ 来分别代表 $\bar{\mathbf{c}}_{l,i}$, $\bar{\mathbf{c}}_{l+1,2i}$ 和 $\bar{\mathbf{c}}_{l+1,2i+1}$ 。那么不等式可转换为如下形式：

$$\begin{aligned} 2 \cdot SED_LB(\{\hat{\mathbf{u}}, \hat{\mathbf{l}}\}, \bar{\mathbf{c}}) &\leq SED_LB(\{\hat{\mathbf{u}}_L, \hat{\mathbf{l}}_L\}, \bar{\mathbf{c}}_L) \\ &+ SED_LB(\{\hat{\mathbf{u}}_R, \hat{\mathbf{l}}_R\}, \bar{\mathbf{c}}_R) \end{aligned} \quad (5.20)$$

此时如果我们能证明对任一维度 j ，都能满足如下不等式，则不等式5.20成立。

$$\begin{aligned}
 2 \cdot SED_LB(\{\hat{\mathbf{u}}^j, \hat{\mathbf{l}}^j\}, \bar{\mathbf{c}}^j) &\leq SED_LB(\{\hat{\mathbf{u}}_L^j, \hat{\mathbf{l}}_L^j\}, \bar{\mathbf{c}}_L^j) \\
 &+ SED_LB(\{\hat{\mathbf{u}}_R^j, \hat{\mathbf{l}}_R^j\}, \bar{\mathbf{c}}_R^j)
 \end{aligned} \tag{5.21}$$

对于不等式5.21左半部分, $\bar{\mathbf{c}}^j$ 和 $\{\hat{\mathbf{u}}^j, \hat{\mathbf{l}}^j\}$ 之间存在以下三种关系:

(i) $\bar{\mathbf{c}}^j \geq \hat{\mathbf{u}}^j$, 此时左半部分等价于 $2 \cdot (\bar{\mathbf{c}}^j - \hat{\mathbf{u}}^j)^2$ 。而右半部分, 我们首先假设 $\bar{\mathbf{c}}_L^j \leq \bar{\mathbf{c}}^j \leq \bar{\mathbf{c}}_R^j$ 。此时我们得到如下结论 $SED_LB(\{\hat{\mathbf{u}}_L^j, \hat{\mathbf{l}}_L^j\}, \bar{\mathbf{c}}_L^j) \geq (\bar{\mathbf{c}}_L^j - \hat{\mathbf{u}}^j)^2$ 。因此, 不等式5.21 左半部分等于 $2 \cdot SED(\bar{\mathbf{c}}, \hat{\mathbf{u}})$ 。对于其右半部分, 我们有

$$\begin{aligned}
 &SED_LB(\{\hat{\mathbf{u}}_L^j, \hat{\mathbf{l}}_L^j\}, \bar{\mathbf{c}}_L^j) + SED_LB(\{\hat{\mathbf{u}}_R^j, \hat{\mathbf{l}}_R^j\}, \bar{\mathbf{c}}_R^j) \\
 &= SED_LB(\{\hat{\mathbf{u}}_L^j, \hat{\mathbf{l}}_L^j\}, \bar{\mathbf{c}}_L^j) + (\bar{\mathbf{c}}_R^j - \hat{\mathbf{u}}^j)^2 \quad (\hat{\mathbf{u}}_R^j \leq \hat{\mathbf{u}}^j \leq \bar{\mathbf{c}}^j \leq \bar{\mathbf{c}}_R^j) \\
 &\geq (\bar{\mathbf{c}}_L^j - \hat{\mathbf{u}}^j)^2 + (\bar{\mathbf{c}}_R^j - \hat{\mathbf{u}}^j)^2 \\
 &\geq (\bar{\mathbf{c}}_L^j - \hat{\mathbf{u}}^j)^2 + (\bar{\mathbf{c}}_R^j - \hat{\mathbf{u}}^j)^2 \quad (\bar{\mathbf{u}}_R^j \leq \hat{\mathbf{u}}^j \leq \bar{\mathbf{c}}_R^j) \\
 &\geq \frac{1}{2} \cdot (\bar{\mathbf{c}}_L^j - \hat{\mathbf{u}}^j + \bar{\mathbf{c}}_R^j - \hat{\mathbf{u}}^j)^2 \quad (AM - GM) \\
 &= \frac{1}{2} \cdot (2\bar{\mathbf{c}}^j - 2\hat{\mathbf{u}}^j)^2 \quad (\bar{\mathbf{c}}_L^j + \bar{\mathbf{c}}_R^j = 2\bar{\mathbf{c}}^j) \\
 &= 2 \cdot (\bar{\mathbf{c}}^j - \hat{\mathbf{u}}^j)^2
 \end{aligned} \tag{5.22}$$

对于 $\bar{\mathbf{c}}_R^j \leq \bar{\mathbf{c}}^j \leq \bar{\mathbf{c}}_L^j$ 的情况, 我们通过相同的方法得到同样的结论。因此, 此种情况下不等式 5.20 成立。

(ii) $\bar{\mathbf{c}}^j \leq \hat{\mathbf{l}}^j$, 可通过跟第一种相同的方法证明不等式 5.21 成立。

(iii) $\hat{\mathbf{l}}^j \leq \bar{\mathbf{c}}^j \leq \hat{\mathbf{u}}^j$, 此时不等式5.21 的左半部分为 0, 其右半部分始终大于 0。

因此, 结论也成立。综合以上几种情况, 不等式5.20 始终成立。原问题得证。 \square

算法 10 DTW-FLB 在协调者节点

```

/* DTW-FLB 协调者结点函数接口实现 */
coordinatorInit( $\mathcal{Q}, \mathcal{R}$ )
1: 根据公式5.6构建 DTWBE  $\{\mathcal{U}, \mathcal{L}\}$ ;
2: for  $i = 1, i < \log_2 n; i++$  do
3:   根据公式5.11构建第  $i$  层包围信封  $\{\hat{\mathcal{U}}_i, \hat{\mathcal{L}}_i\}$ ;
4:  $l \leftarrow -1$ ;
generateInfo()
1:  $l \leftarrow l + 1$ ;
2: return  $\{\hat{\mathcal{U}}_l, \hat{\mathcal{L}}_l\}$ ;

```

5.3 基于动态时间距离的查询算法:DTW-FLB**5.3.1 DTW-FLB 算法实现**

在上一节，我们已经利用多粒度概要数据得到越来越紧的动态时间弯曲距离下界。这使得在 FLB 框架中插入欧氏距离成为可能。本节将详细介绍动态时间弯曲距离跟 FLB 相结合的查询算法 DTW-FLB。DTW-FLB 维持了 FLB 框架的主要结构，只对本章第一节所介绍的接口进行了具体实现。

我们首先介绍如何实现那些运行在协调者节点上的函数。在 **coordinatorInit** 函数中，我们首先对待查询轨迹计算出对应的 DTWBE，接着基于 DTWBE，我们构造多粒度的包围信封（即概要数据）。由于 FLB 框架也是迭代式由粗到细的通信计算框架，在每轮的迭代中会将某一层的概要数据（包围信封）发送给远程结点。因此，在初始化过程中我们用 l 来记录当前已经发送到哪一层的数据，并将 l 初始化为 -1 ，以便从第 0 层开始发送。此外，我们在 **generateInfo** 中准备将要发送的下一层概要数据，以便协调者结点发送。在发送数据时需要注意的是，由于 $\{\hat{\mathcal{U}}_l, \hat{\mathcal{L}}_l\}$ 的一半数据量已经包含在 $\{\hat{\mathcal{U}}_{l-1}, \hat{\mathcal{L}}_{l-1}\}$ 中，我们只需传输剩下的一半数据。

接着，介绍运行在远程结点的函数。对于 **RemoteInit** 函数，若远程结点是第一次接收查询，则会为每条候选轨迹进行哈尔小波变换。若不是，则为该结点所包含的每条轨迹初始化下界特征信息（下界初始化为 0）。在查询执行过程中 **UpdateBounds** 函数为候选更新下界。直接实现方式为根据下界的计算公式直接算出该值。但由

算法 11 DTW-FLB 在远程结点

```

/* DTW-FLB 远程结点函数接口实现 */
remoteInit( $TS_r, S_r$ )    集合  $S_r$  维护本地候选的下界特征
1: if 第一次接收查询 then
2:   构建 R-Tree 索引。
3:   for all  $C$  in  $TS_r$  do
4:      $\bar{C} \leftarrow C$  的均值序列;           # 候选轨迹获取所有层的均值构成的序列
5:   for all  $C$  in  $TS_r$  do
6:      $C\_LBF \leftarrow \langle C, 0 \rangle$ ;       # 为  $C$  初始化下界
7:      $S_r \leftarrow S_r \cup C\_BF$ ;
updateBounds( $S_r, m$ )    / *  $m = \{\hat{u}_{l,i}, \hat{l}_{l,i}\} * /$ 
1: if  $l = 0$  then
2:   使用索引树进行剪枝, 将不满足要求的轨迹从  $S_r$  中移除。
3: else
4:   for all  $C$  in  $TS_r$  do
5:      $factor \leftarrow 2^{L-l}$ ;
6:      $sum \leftarrow 0$ ;
7:     for  $i = 0, i < 2^l; i++$  do
8:        $p\_lb \leftarrow SED\_LB(\{\hat{u}_{l,i}, \hat{l}_{l,i}\}, \bar{c}_{l,i})$ ;
9:        $sum \leftarrow sum + p\_lb$ ;
10:    if  $sum > \theta$  then
11:      将  $C$  对应下界特征从  $S_r$  中移除;
12:      将  $C$  对应下界特征中下界值更新为  $sum$ ;

```

于查询过程中的主要计算开销就是对所有候选更新下界特征。因此,降低该过程的计算开销很有必要。为降低更新下界特征的计算开销,与上一章一样采用在更新界过程中进行剪枝的思想。根据 LB_HPAA 的定义,我们可知该下界是一系列点距离下界的累加和。由于每个点距离下界都是大于 0 的数,因此我们可以在累加的过程中,通过判断当前已经累加的值是否已经超过剪枝的阈值。若超过了则可提前将该值过滤掉。

除了以上步骤外,我们引入了 R 树来对轨迹构建索引以提高查询效率。具体地:首先,每个远程结点对自己局部轨迹数据构建 R 树索引。接着,当该结点接收到第 0 层包围信封时,使用 R-tree 来剪枝。剪枝具体过程入下:(i) 遍历 R 树,找到跟查询轨迹包围信封相交的叶子节点。(ii) 遍历这些叶子节点所包含的轨迹,并将这些轨迹列为候选。

5.3.2 DTW-FLB 算法性能分析

本章节将从时间复杂度和空间复杂度量方面对 DTW-FLB 进行性能分析。在此之前，我们仍使用 $|C_i|$ 和 $|CS_i|$ 分别表示第 i ($i \geq 0$) 次迭代前候选轨迹的数量和包含候选轨迹的远程结点数量，并假定迭代次数为 λ ($\lambda \leq \log_2 n$)。由于仅使用下界进行剪枝，因此存在着当概要数据发送完毕后，仍然存在着超过 k 个候选的情况。此时，我们使用 N' 来表示迭代计算结束后剩余候选的个数。

时间复杂度：不考虑系统初始化时对所有候选进行哈尔小波变换的时间开销，DTW-FLB 算法的时间主要来自两个方面：(i) 算法 8 第 2 阶段的迭代更新的时间；(ii) 对迭代结束后的候选计算真实动态时间弯曲距离的时间。第一方面的时间复杂度为 $\sum_{i=0}^{\lambda} d \cdot |C_i| \cdot 2^i$ ，该值与算法 ED-FTB 的迭代计算复杂度相同。第二方面的计算复杂度为 $O(d \cdot \delta \cdot N' \cdot n^2)$ 。因此，总的时间复杂度为 $O(d \cdot (\sum_{i=0}^{\lambda} |C_i| \cdot 2^i + \delta \cdot N' \cdot n^2))$ 。

通信复杂度：DTW-FLB 算法的通信开销跟时间开销一样也是来自迭代发送包围信封和迭代结束后发送轨迹计算真实距离两个阶段。在迭代式发送包围阶段，第 i 次迭代中，协调者需要将第 i 层包围信封发送到候选结点中。由于，第 i 层包围信封已经有一半数据存在于第 $i-1$ 层包围信封中且已被候选结点接收。因此，发送的数据量为 $O(d \cdot 2^i \cdot |CS_i|)$ 。远程结点在接收到包围信封后，便更新下界并将局部最小的 k 个下界发送给协调者结点。这一过程发送的数据量最多为 $O(|C_i|)$ 。总的来说，迭代过程中产生的通信开销为。假设迭代结束后，所剩 N' ($N' > k$) 个结点分布在 m' 个远程结点中。此时最多需要将原始轨迹发送到 m' 个结点中。其所对应的开销为 $O(d \cdot n \cdot m')$ 。结合这两个阶段，DTW-FLB 的总通信开销为 $O(\sum_{i=0}^{\log_2 n} (d \cdot 2^i \cdot |CS_i| + |C_i|) + d \cdot n \cdot m')$ 。

通过以上对 DTW-FLB 的分析，我们可以发现该算法对迭代剪枝的效果十分敏感。即剪枝结束后所剩候选的个数即包含这些候选的节点的个数，会严重影响到计算时间和通信开销。由于，DTW-FLB 会在每一轮迭代中都会通过对部分候选轨迹计算真实的 DTW 距离用以更新全局过滤的阈值。该方法能保证该过滤阈值会不断逼近真实的第 k 小距离，因此最终会取得较好的剪枝效果。

5.4 实验分析

5.4.1 实验设置

本章工作在实验中使用北京出租车数据集，该数据集已被广泛应用于轨迹数据分析。该数据集采集了北京市三万多辆出租车在 2013 年 10 月份到 12 月份三个月内的行驶 GPS 轨迹。每个 GPS 轨迹点包含的数据维度较多，我们仅选取了位置(经度和纬度)、时间、速度和角度这五个维度的值，并进行了归一化预处理。我们从该数据集中截取了两个子数据集：*T-Small* 和 *T-Big* 以分别验证算法的有效性和可扩展性。数据集描述见表 4.1，其中 *T-Small* 截取了 10 月 1 至 7 号间上午 8 点到 10 点的轨迹数据，并从中选出最长的 1 万条轨迹进行分析。*T-Big* 截取了 11 月 1 号至 12 月 31 号间每天上午 8 至 10 点和下午 5 至 7 点两个时间段的 1 百万条轨迹数据。为满足实验需求，这两个数据集的每条轨迹长度均超过 4,096。

在本章的实验中，DTW-FLB 算法使用 JAVA 实现。需要注意的是，在我们介绍中 δ 的取值为整数来表示约束的范围。但为适应不同长度轨迹的需求，我们在实验中使用轨迹长度的百分比来表示约束范围。此外，我们将 l 的初值设置为 3，即 DTW-FLB 算法从第 3 层包围盒发起。实验运行在一个包含 12 个结点的 Spark 集群上。每个结点包含 8 核英特尔 E5335 2.0 GHz 中央处理器和 16GB 的内存。

5.4.2 算法有效性

首先，我们通过汇报迭代过程中每轮结束后所剩的候选的个数以展示 DTW-FLB 的剪枝效果。在该组实验中，我们设置轨迹长度 n 从 512 变化到 4,096， k 的值从 1 变化到 100。实验结果，为 10 次查询的平均值。图 5.4 展示了随着 k 和 n 变化后的剪枝的效果。从该图中我们可以看出在最开始的两轮迭代中，候选的个数急剧下降。此外，两轮迭代后所剩候选数较少。这说明，使用粗粒度的概要数据索引和剪枝策略的混合使用能起到较好的剪枝效果。在接下来的迭代中，我们可以发现候选个数降低比较平缓。由此可见，本文提出的由粗到细使用多粒度概要数据进行剪枝的思想具有较高的优越性，即它能够使用较少的数据剪枝掉大多数候选，

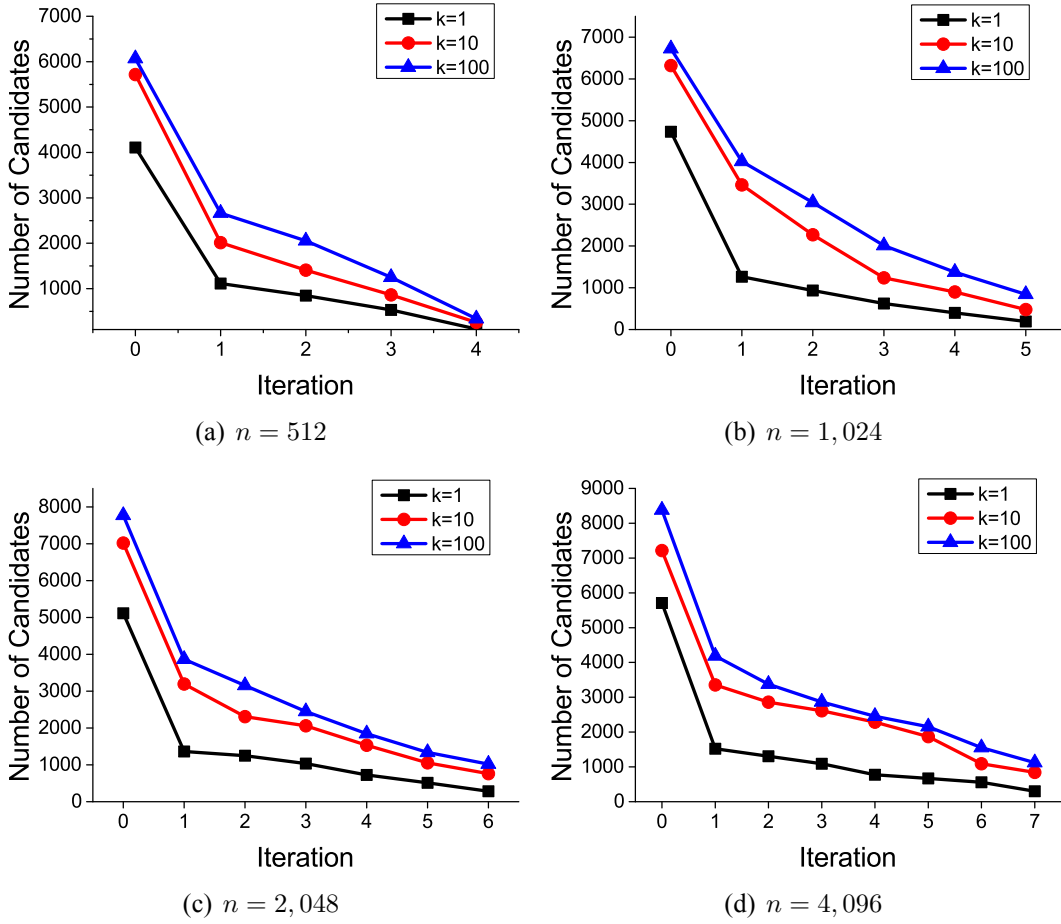
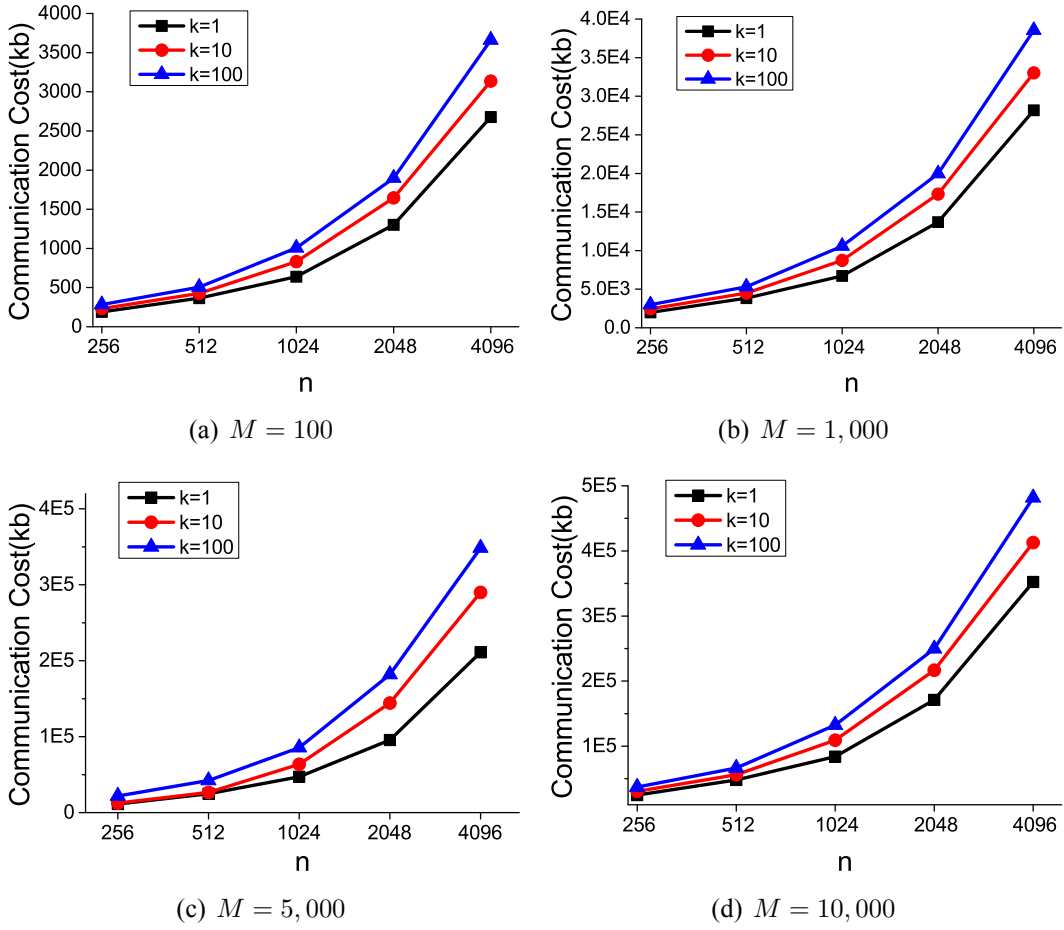


图 5.4: DTW-FLB 剪枝效果

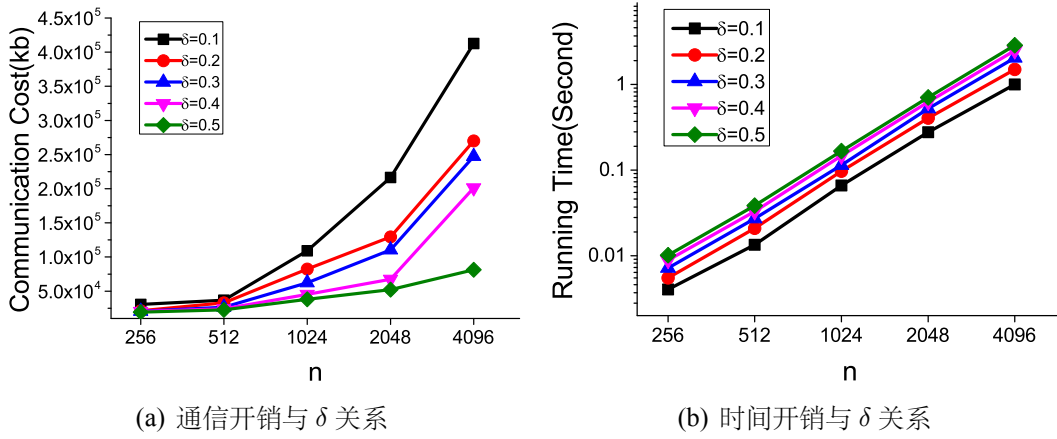
从而达到了降低通信开销的目的。此外，可以发现随着 k 值的降低，我们能取得更好的剪枝效果。究其原因， k 值较小时，所选取的全局过滤阈值随之降低即更接近真实的第 k 小真实距离值。

接着，我们研究了 n ， k 和 M 对 DTW-FLB 算法通信开销的影响。在该组实验中，我们将 M 值从 100 变化到 10,000，将 k 从 1 变化到 100，并将 n 从 256 变化到 4,096。实验结果如图 5.5 所示，我们可以看到随着 n 和 M 的指数增加，通信开销随之指数增长。这实际反应了通信开销与 n 和 M 之间存在着线性关系。究其原因，随着 M 的增加，候选轨迹会存在于更多的远程结点中。所以协调者结点需要将概要数据发送到更多的节点中，这导致了通信开销的增加。此外，当 n 增加时，长轨迹需要传递更多的数据才能跟短轨迹达到相同的剪枝效果。最后，随着 k 值

图 5.5: DTW-FLB 通信开销与 n , M 和 k 之间的关系

的增加, 通信开销也会随之增长。这一结论跟图5.4所得到的结果一致, 反应了随着 k 的增加, 剪枝阈值越松, 进而每轮迭代中所剩候选越多。

然后, 我们从时间和通信开销两个角度研究了 DTW 距离计算中 δ 的值对 DTW-FLB 算法的影响。在该组实验中, 我们将 k 和 M 的值分别固定为 10 和 10,000。实验结果如图5.6所示。我们可以发现随着 δ 的增加通信开销逐步减小。其原因是 δ 值的增加会导致我们的包围信封更加紧凑, 从而更利于剪枝。同时, 发现随着 δ 的增加时间开销逐步增加。这是因为 δ 值越大会导致最后一步计算 DTW 真实距离时的搜索空间越大。此外, 结合前面对 DTW-FLB 的时间复杂度分析部分, 我们可以发现这一实验结果跟我们的理论分析是一致的。

图 5.6: DTW-FLB 性能与 δ 间的关系

5.4.3 算法可扩展性

本小节我们将从时间和通信两个角度研究了 DTW-FLB 算法的可扩展性。首先，考虑了轨迹数量 N （即数据量大小）和轨迹长度 n 对可扩展性的影响。在该组实验中，我们将 k 设置为 1， M 设置为 10,000。我们将轨迹长度从 256 到 4096 进行指数级变化，同时将轨迹数量从 10 万到 1 百万进行线性变换。图 5.7 分别对这两个角度的结果进行了介绍。其中图 5.7(a) 介绍了时间性能受 n 和 N 的影响。我们可以看到，对于任意长度的轨迹数据集，DTW-FLB 算法的运行时间都随着轨迹数据量的增加而线性增加。此外，随着轨迹长度的指数增加，算法的运行时间也指数增加，即运行时间与轨迹长度呈一定的线性关系。这一结果反应了由于 DTW-FLB 的剪枝效果较好，导致迭代结束后，所剩候选数较少。即我们对 DTW-FLB 算法时间复杂度分析部分的 N' 较少。结合以上两点，我们可以看出 DTW-FLB 算法运行时间随着数据集的大小而线性变换，因此运行效率具有较好的可扩展性。

接着，我们在图 5.7(b) 中介绍了通信性能受 n 和 N 的影响。从图中可以看出，通信量随着轨迹的数据的线性增加而呈指数增加。这是由于 DTW-FLB 具有较好的剪枝效果，导致迭代结束后，不仅所剩候选较少，而且所剩下的候选结点也较少。因此，迭代结束后的通信较少。而根据我们前面对 DTW-FLB 的通信开销分析可知通信开销主要来自迭代式计算部分和迭代后将轨迹发送给所有的候选结点部

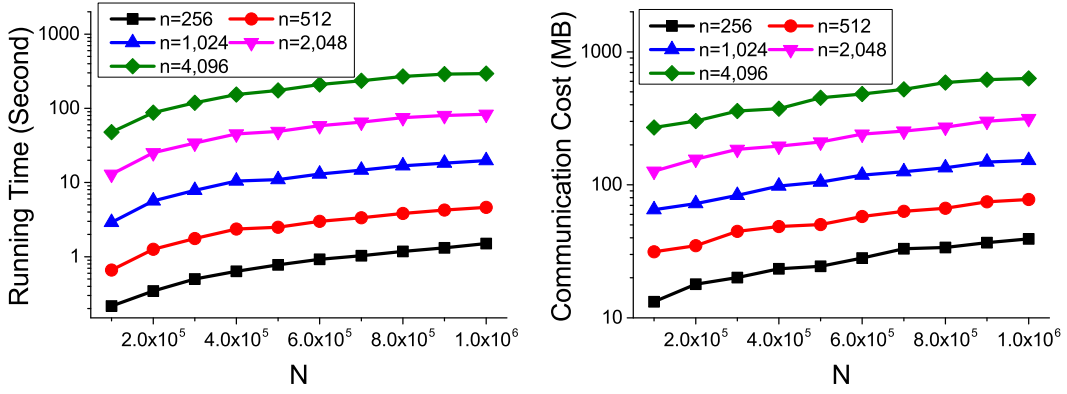
(a) 时间可扩展与 n 和 N 之间的关系(b) 通信可扩展性与 n 和 N 之间的关系

图 5.7: DTW-FLB 可扩展性

分。由于后一部分开销较少，故主要开销集中在第一部分，即迭代式通信部分。根据图5.4分析可知，前两次迭代已经过滤掉很多结点，后面的候选变化较小。此时，第一部分的开销主要跟包围盒大小相关。而包围盒大小是呈指数变换（当前大小是上一次的2倍）。所以，我们的实验结果跟我们的理论分析一致。此外，从图中可以看出通信开销随着轨迹长度的指数变化而指数增加，即通信开销与轨迹长度呈线性关系。这反应了对于长轨迹需要更多的概要数据才能达到与短轨迹相同的剪枝效果。

5.5 本章小结

本章节介绍了利用距离下界进行剪枝的思想实现分布式 k 近邻查询的同时，以降低通信开销。本章节，首先设计了利用多粒度概要数据不断缩紧距离下界，并利用下界特征进行剪枝的 FLB 策略。接着，针对动态时间弯曲距离，设计了多粒度包围信封作为概要数据，并提出了不断变紧的下界。其次，将动态时间弯曲距离应用于 FLB 策略并设计了 DTW-FLB 算法。该算法引入了多种优化措施以提高查询效率。最后，通过在真实数据集上进行的实验，表明 DTW-FLB 算法效率高且具有较好的可扩展性。

第六章 总结与展望

本文重点研究了分布式轨迹数据上的 k 近邻查询。本章对全文研究内容和技术模型进行总结归纳，并展望未来的重点研究工作。

6.1 研究总结

随着移动设备的广泛应用，各行各业产生了海量轨迹大数据，随之而来面临着如何管理和利用该数据的问题。首先，本文研究了如何对采集到的轨迹数据进行有效管理并提供低延时的查询问题。本文的思路是利用现有分布式集群处理系统，构建基于分布式内存的轨迹管理系统。接着，本文针对海量轨迹数据散布在各个数据节点且无法被采集到集群内的场景，研究了 k 近邻查询的实现。本文的研究目标是在保证结果正确性的前提下，如何降低该查询的通信和时间开销。此外，目前存在着许多轨迹间距离度量方式。现有的查询研究工作往往只是针对某种距离进行查询优化，从而缺乏通用性。所以本文的另一个目标就是提出通用的 k 近邻查询处理方法。为此，本文从以下三个方面解决相关问题：（1）如何对采集到的轨迹大数据进行有效管理并提供低延时的轨迹查询服务；现有的分布式轨迹管理系统尽管能提供轨迹管理和查询服务，但由于这些系统都是基于磁盘的，导致查询执行期间 I/O 开销较大，无法提供低延时的查询效果。（2）尽管目前存在着许多轨迹距离度量方式，但缺乏通用的查询处理策略。设计统一的处理框架可以满足不同应用场景的需求，使得处理的问题不再受限。（3）针对轨迹数据分布在各个数据节点上的 k 近邻查询，面临着通信和计算开销较大的问题；相对于传统集中式环境下的查询，分布式环境下通信开销成为算法的首要瓶颈。此外，在降低通信开销的同时，仍需要保证查询效率。具体地，本文的研究问题和技术贡献总结如下。

首先，针对现有系统无法对海量轨迹数据提供实时查询服务问题，本文设计

了基于分布式内存的管理系统 TrajSpark。TrajSpark 首先提出了基于分布式内存的轨迹数据表示结构 IndexTRDD。该结构在引入了列存储的思想来表示轨迹片段并引入压缩技术以减少存储开销。此外，我们为 IndexTRDD 引入了全局和局部相结合的两层索引策略以降低查询的搜索开销。进一步的，为满足数据不断添加需求，TrajSpark 引入了时间衰减模型以描述轨迹数据分布的变化，并使得系统能够自适应的动态划分新到来的数据。最后，TrajSpark 针对三种典型轨迹查询设计了查询算法。海量轨迹数据集上的查询结果表明，TrajSpark 相比已有分布式内存时空系统具有较高的优越性。

其次，针对现有轨迹距离度量方法多，导致缺乏通用 k 近邻查询处理方法的问题，本文设计了两种基于多粒度概要数据的剪枝查询策略：一种是同时使用距离上、下界的剪枝策略，另一种是仅使用下界剪枝的策略。其中第一种策略适用于那些能够根据概要数据同时计算出上、下界的轨迹距离函数，第二种适用于那些仅能根据概要数据计算出下界的距离函数。针对具体的距离度量方式，我们只需研究出适合的概要数据并提供界信息，便能应用到对应的框架中。最后，研究了通过欧氏和动态时间弯曲两种距离在这两种策略下的应用，验证了策略的有效性。

最后，针对分布式 k 近邻查询中通信和计算开销过大的问题，本文设计的两种使用多粒度概要数据进行剪枝的策略能大大降低通信开销。由于概要数据占用空间较少，通过发送概要数据来剪枝能够避免将数据发送到不必要的节点上。此外，随着数据粒度的增加，我们能够得到不断变紧的距离范围，从而进一步剪枝候选。在这两个策略中应用到欧氏和动态时间弯曲距离的过程，由于计算这两个距离的范围的开销远小于真实距离的计算开销，因而能同时达到降低计算开销的目的。

6.2 研究展望

本文重点研究了分布式 k 近邻轨迹查询问题。作者认为还可以从下面三个方面进行扩展。

首先，可以将本文系统进行更多的查询和应用扩展。本文所提出的系统 TrajS-

park 目前支持基本的基于移动对象标识的查询和时空范围查询, 以及高级的 k 近邻查询。在未来的工作中, 我们将在 **TrajSpark** 上实现更多的查询及应用。

其次, 可以针对语义轨迹上的查询进行拓展研究。本文所研究的轨迹数据来自欧氏空间, 如何结合含非欧空间的语义信息进行近邻查询会是一个有趣的问题。一些前人工作 [110, 111, 113] 已经考虑了对语义轨迹上的查询。因此, 基于轨迹的查询也是未来研究的重点。

最后, 本文所提算法是针对协调者-远程结点这一分布式场景设计的查询算法。随着区块链¹、以太坊²等基于 **P2P** 系统和概念的兴起, 基于 **P2P** 这一分布式架构的轨迹等时间序列上的近邻查询会成为未来的热点。

¹<https://blockchain.info/>

²<https://www.ethereum.org/>

参考文献

- [1] GONZÁLEZ M C, HIDALGO C A, BARABÁSI A L. Understanding individual human mobility patterns.[J]. Nature, 2008, 453(7196): 779.
- [2] SONG C, BARABÁSI A L. Limits of Predictability in Human Mobility[J]. Science, 2010, 327(5968): 1018.
- [3] LI Z, DING B, HAN J, et al. Mining periodic behaviors for moving objects[C] // Proceedings of the 16th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD). 2010: 1099–1108.
- [4] 章志刚, 金澈清, 王晓玲, et al. 面向海量低质手机轨迹数据的重要位置发现 [J]. 软件学报, 2016, 27(7): 1700–1714.
- [5] ZHENG Y. Trajectory Data Mining: An Overview[J]. ACM TIST, 2015, 6(3): 29:1–29:41.
- [6] 乔少杰, 金琨, 韩楠, et al. 一种基于高斯混合模型的轨迹预测算法 [J]. 软件学报, 2015, 26(5): 1048–1063.
- [7] 乔少杰, 韩楠, 李天瑞, et al. 基于前缀投影技术的大规模轨迹预测模型 [J]. 软件学报, 2017, 28(11): 3043–3057.
- [8] 李晓旭, 于亚新, 张文超, et al. 基于 MapReduce 的 Coteries 轨迹模式挖掘及个性化旅游路线推荐 [J]. 软件学报, 2018, 29(3): 587–598.
- [9] 陈锦阳, 刘良旭, 宋加涛, et al. 基于 R-tree 的高效异常轨迹检测算法 [J]. 计算机应用与软件, 2011, 20(10): 2426–2435.
- [10] 毛嘉莉, 金澈清, 章志刚, et al. 轨迹大数据异常检测: 研究进展及系统框架 [J]. 软件学报, 2017, 28(1): 17–34.
- [11] 齐观德, 潘遥, 李石坚, et al. 基于出租车轨迹数据挖掘的乘客候车时间预测 [J]. 软件学报, 2013, 24(S2): 14–23.
- [12] 许佳捷, 郑凯, 池明旻, et al. 轨迹大数据: 数据、应用与技术现状 [J]. 通信学报, 2015, 36(12): 97–105.
- [13] 高强, 张凤荔, 王瑞锦, et al. 轨迹大数据: 数据处理关键技术研究综述 [J]. 软件学报, 2017, 28(4): 959–992.

- [14] 包婷, 章志刚, 金澈清. 基于手机大数据的城市人口流动分析系统 [J]. 华东师范大学学报 (自然科学版), 2015, 2015(5): 162–171.
- [15] WANG H, ZHENG K, XU J, et al. SharkDB: An In-Memory Column-Oriented Trajectory Storage[C] // Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM). 2014: 1409–1418.
- [16] TAN H, LUO W, NI L M. CloST: a hadoop-based storage system for big spatio-temporal data analytics[C] // Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM). 2012: 2139–2143.
- [17] ZHANG Z, JIN C, MAO J, et al. TrajSpark: A Scalable and Efficient In-Memory Management System for Big Trajectory Data[C] // Proceedings of the First International APWeb-WAIM Joint Conference on Web and Big Data, Part I. 2017: 11–26.
- [18] XIE D, LI F, PHILLIPS J M. Distributed Trajectory Similarity Search[J]. PVLDB, 2017, 10(11): 1478–1489.
- [19] DENG Z, HU Y, ZHU M, et al. A scalable and fast OPTICS for clustering trajectory big data[J]. Cluster Computing, 2015, 18(2): 549–562.
- [20] COSTA G, MANCO G, MASCIARI E. Dealing with trajectory streams by clustering and mathematical transforms[J]. Journal of Intelligent Information Systems, 2014, 42(1): 155–177.
- [21] YU Y, WANG Q, WANG X, et al. Online clustering for trajectory data stream of moving objects[J]. Computer Science and Information Systems, 2013, 10(3): 1293–1317.
- [22] MAO J, SONG Q, JIN C, et al. TScluWin: Trajectory Stream Clustering over Sliding Window[C] // Proceedings of the 21st International Conference on Database Systems for Advanced Applications (DASFAA), Part II. 2016: 133–148.
- [23] NEHME R V, RUNDENSTEINER E A. SCUBA: Scalable Cluster-Based Algorithm for Evaluating Continuous Spatio-temporal Queries on Moving Objects[C] // Proceedings of the 10th International Conference on Extending Database Technology (EDBT). 2006: 1001–1019.
- [24] SACHARIDIS D, PATROUMPAS K, TERROVITIS M, et al. On-line discovery of hot motion paths[C] // Proceedings of the 11th International Conference on Extending Database Technology (EDBT). 2008: 392–403.

- [25] ZHENG K, ZHENG Y, YUAN N J, et al. On discovery of gathering patterns from trajectories[C] // Proceedings of the 29th IEEE International Conference on Data Engineering (ICDE). 2013 : 242 – 253.
- [26] TANG L A, ZHENG Y, YUAN J, et al. On Discovery of Traveling Companions from Streaming Trajectories[C] // Proceedings of the 28th IEEE International Conference on Data Engineering (ICDE). 2012 : 186 – 197.
- [27] LI X, CEIKUTE V, JENSEN C S, et al. Effective Online Group Discovery in Trajectory Databases[J]. IEEE Transaction on Knowledge and Data Engineering (TKDE), 2013, 25(12) : 2752 – 2766.
- [28] DUAN X, JIN C, WANG X, et al. Real-Time Personalized Taxi-Sharing[C] // Proceedings of the 21st International Conference on Database Systems for Advanced Applications (DASFAA), Part II. 2016 : 451 – 465.
- [29] ZHANG Z, WANG Y, MAO J, et al. DT-KST: Distributed Top-k Similarity Query on Big Trajectory Streams[C] // Proceedings of the 22nd International Conference on Database Systems for Advanced Applications (DASFAA), Part I. 2017 : 199 – 214.
- [30] HSU C-C, KUNG P-H, YEH M-Y, et al. Bandwidth-efficient distributed k-nearest-neighbor search with dynamic time warping[C] // Proceedings of the 2015 International Conference on Big Data (ICBD). 2015 : 551 – 560.
- [31] ZEINALIPOUR-YAZTI D, LAOUDIAS C, COSTA C, et al. Crowdsourced trace similarity with smartphones[J]. IEEE Transaction on Knowledge and Data Engineering (TKDE), 2013, 25(6) : 1240 – 1253.
- [32] COSTA C, LAOUDIAS C, ZEINALIPOUR-YAZTI D, et al. SmartTrace: Finding similar trajectories in smartphone networks without disclosing the traces[C] // Proceedings of the 27th IEEE International Conference on Data Engineering (ICDE). 2011 : 1288 – 1291.
- [33] CHEN L, OZSU M T, ORIA V. Robust and fast similarity search for moving object trajectories[C] // Proceedings of the 2005 ACM International Conference on Management of Data (SIGMOD). 2005 : 491 – 502.
- [34] CHEN L, NG R T. On The Marriage of Lp-norms and Edit Distance[C] // Proceedings of the 30th International Conference on Very Large (VLDB). 2004 : 792 – 803.

- [35] ZHU H, LUO J, HANG, et al. Mining Trajectory Corridors Using Frechet Distance and Meshing Grids[C] // Proceedings of the 14th Pacific-Asia Advances in Knowledge Discovery and Data Mining (PAKDD). 2010 : 228 – 237.
- [36] GUO N, MA M, XIONG W, et al. An Efficient Query Algorithm for Trajectory Similarity Based on Frechet Distance Threshold[J]. International Journal of Geo-Information, 2017, 6(11) : 326.
- [37] LIN B, SU J. One Way Distance: For Shape Based Similarity Search of Moving Object Trajectories[J]. GeoInformatica, 2008, 12(2) : 117 – 142.
- [38] RANU S, P D, TELANG A D, et al. Indexing and matching trajectories under inconsistent sampling rates[C] // Proceedings of the 31st IEEE International Conference on Data Engineering (ICDE). 2015 : 999 – 1010.
- [39] LIU H, SCHNEIDER M. Similarity measurement of moving object trajectories[C] // Proceedings of the 2012 ACM International Workshop on GeoStreaming (SIGSPATIAL). 2012 : 19 – 22.
- [40] ZHAO X, XU W. A New Measurement Method to Calculate Similarity of Moving Object Spatio-Temporal Trajectories by Compact Representation[J]. International Journal of Computational Intelligence Systems, 2011, 4(6) : 1140 – 1147.
- [41] ZHENG B, YUAN N J, ZHENG K, et al. Approximate keyword search in semantic trajectory database[C] // Proceedings of the 31st IEEE International Conference on Data Engineering (ICDE). 2015 : 975 – 986.
- [42] NEHAL M, A. S M, MOSTAFA T, et al. Review on trajectory similarity measures[C] // Proceedings of the 7th IEEE International Conference on Intelligent Computing and Information Systems (ICICIS). 2016 : 613 – 619.
- [43] TOOHEY K, DUCKHAM M. Trajectory similarity measures[J]. SIGSPATIAL Special, 2015, 7(1) : 43 – 50.
- [44] FRENTZOS E, GRATSIAS K, PELEKIS N, et al. Algorithms for Nearest Neighbor Search on Moving Object Trajectories[J]. GeoInformatica, 2007, 11(2) : 159 – 193.
- [45] GÜTING R H, BEHR T, JIANQIU. Efficient k-nearest neighbor search on moving object trajectories[J]. VLDB Journal, 2010, 19(5) : 687 – 714.
- [46] KOLLIOS G, GUNOPULOS D, TSOTRAS V J. Nearest Neighbor Queries in a Mobile Environment[C] // Proceedings of the International Workshop on Spatio-Temporal Database Management (STDBM). 1999 : 119 – 134.

- [47] CHEN Z, SHEN H T, ZHOU X, et al. Searching trajectories by locations: an efficiency study[C] // Proceedings of the 2010 ACM International Conference on Management of Data (SIGMOD). 2010 : 255 – 266.
- [48] AGRAWAL R, FALOUTSOS C, SWAMI A N. Efficient Similarity Search In Sequence Databases[C] // Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms (FODO). 1993 : 69 – 84.
- [49] RAFIEI D, MENDELZON A O. Efficient retrieval of similar shapes[J]. VLDB Journal, 2002, 11(1): 17 – 27.
- [50] NIKOS P, IOANNIS K, GERASIMOS M, et al. Similarity Search in Trajectory Databases[C] // Proceedings of the 14th International Symposium on Temporal Representation and Reasoning (TIME). 2007 : 129 – 140.
- [51] GEORGIOS S, DIMITRIOS S, ALEXANDRA V. Efficient Identification and Approximation of K-nearest Moving Neighbors[C] // Proceedings of the 21st ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL). 2013 : 264 – 273.
- [52] ZHAO Z, SAALFELD A. Linear-time Sleeve-fitting Polyline Simplification Algorithms[C] // Proceedings of Autocarto 13. 1997 : 214 – 223.
- [53] BOTEVA V, MALLET D, NASCIMENTO M A, et al. PIST: An Efficient and Practical Indexing Technique for Historical Spatio-Temporal Point Data[J]. Geoinformatica, 2008, 12(2) : 143 – 168.
- [54] CHAKKA V P, EVERSPAUGH A, PATEL J M. Indexing large trajectory data sets with SETI[C] // Proceedings of the 1st Biennial Conference on Innovative Data Systems (CIDR). 2003.
- [55] CUDRÉ-MAUROUX P, WU E, MADDEN S. TrajStore: An adaptive storage system for very large trajectory data sets[C] // Proceedings of the 26th IEEE International Conference on Data Engineering (ICDE). 2010 : 109 – 120.
- [56] MA Q, YANG B, QIAN W, et al. Query processing of massive trajectory data based on mapreduce[C] // Proceedings of the First International CIKM Workshop on Cloud DataManagement (CloudDb). 2009 : 9 – 16.
- [57] YANG B, MA Q, QIAN W, et al. TRUSTER: TRajjectory Data Processing on CLUSTERS[C] // Proceedings of the 14th International Conference Database Systems for Advanced Applications (DASFAA). 2009 : 768 – 771.

- [58] ELDAWY A, MOKBEL M F. SpatialHadoop: A MapReduce framework for spatial data[C] // Proceedings of the 31st IEEE International Conference on Data Engineering (ICDE). 2015 : 1352 – 1363.
- [59] ALY A M, MAHMOOD A R, HASSAN M S, et al. AQWA: Adaptive Query-Workload-Aware Partitioning of Big Spatial Data[J]. PVLDB, 2015, 8(13) : 2062 – 2073.
- [60] YOU S, ZHANG J, GRUENWALD L. Large-scale spatial join query processing in Cloud[C] // Proceedings of the 31st IEEE International Conference on Data Engineering (ICDE) Workshops. 2015 : 34 – 41.
- [61] YU J, WU J, SARWAT M. GeoSpark: a cluster computing framework for processing large-scale spatial data[C] // Proceedings of the 23rd International Conference on Advances in Geographic Information Systems (SIGSPATIAL). 2015 : 70:1 – 70:4.
- [62] TANG M, YU Y, MALLUHI Q M, et al. Locationspark: a distributed in-memory data management system for big spatial data[J]. PVLDB, 2016, 9(13) : 1565 – 1568.
- [63] XIE D, LI F, YAO B, et al. Simba: Efficient In-Memory Spatial Analytics[C] // Proceedings of the 2016 ACM International Conference on Management of Data (SIGMOD). 2016 : 1071 – 1085.
- [64] YE H M Y, WU K L, YU P S, et al. LeeWave: level-wise distribution of wavelet coefficients for processing kNN queries over distributed streams[J]. PVLDB, 2008, 1(1) : 586 – 597.
- [65] 王艺霖, 章志刚, 金澈清. 智能交通刷卡记录中的公交站点恢复方法 [J]. 华东师范大学学报 (自然科学版), 2017, 2017(5) : 201 – 212.
- [66] LEE J, HAN J, WHANG K. Trajectory clustering: a partition-and-group framework[C] // Proceedings of the 2007 ACM International Conference on Management of Data (SIGMOD). 2007 : 593 – 604.
- [67] PFOSER D, JENSEN C S, THEODORIDIS Y. Novel Approaches to the Indexing of Moving Object Trajectories[C] // Proceedings of the 26th International Conference on Very Large Data Bases (VLDB). 2000 : 395 – 406.
- [68] XU X, HAN J, LU W. Rt-tree: an improved r-tree index structure for spatiotemporal data[C] // Proceedings of the 4th International Symposium on Spatial Data Handling. 1990.

- [69] NASCIMENTO M A, SILVA J R O. Towards historical R-trees[C] // Proceedings of the 1998 ACM Symposium on Applied Computing (SAC). 1998 : 235 – 240.
- [70] TAO Y, PAPADIAS D. Efficient Historical R-trees[C] // Proceedings of the 13th International Conference on Scientific and Statistical Database Management (SS-DBM). 2001 : 223 – 232.
- [71] 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 (VLDB). 2001 : 431 – 440.
- [72] ZHOU P, ZHANG D, SALZBERG B, et al. Close pair queries in moving object databases[C] // Proceedings of the 13th ACM International Workshop on Geographic Information Systems (ACM-GIS). 2005 : 2 – 11.
- [73] SONG Z, ROUSSOPOULOS N. SEB-tree: An Approach to Index Continuously Moving Objects[C] // Proceedings of the 4th International Conference on Mobile Data Management (MDM). 2003 : 340 – 344.
- [74] WANG H, ZHENG K, ZHOU X, et al. SharkDB: An In-Memory Storage System for Massive Trajectory Data[C] // Proceedings of the 2015 ACM International Conference on Management of Data (SIGMOD). 2015 : 1099 – 1104.
- [75] WANG H, ZHENG K, XU J, et al. SharkDB: An In-Memory Column-Oriented Trajectory Storage[C] // Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM). 2014 : 1409 – 1418.
- [76] ZHENG B, WANG H, ZHENG K, et al. SharkDB: an in-memory column-oriented storage for trajectory analysis[J]. World Wide Web, 2018, 21(2) : 455 – 485.
- [77] LU J, GÜTING R H. Parallel Secondo: Boosting Database Engines with Hadoop[C] // Proceedings of the 18th IEEE International Conference on Parallel and Distributed Systems (ICPADS). 2012 : 738 – 743.
- [78] DEAN J, GHEMAWAT S. MapReduce: Simplified Data Processing on Large Clusters[C] // Proceedings of the 6th Symposium on Operating System Design and Implementation (OSDI). 2004 : 137 – 150.
- [79] DEAN J, GHEMAWAT S. MapReduce: simplified data processing on large clusters[J]. ACM Communication, 2008, 51(1) : 107 – 113.

- [80] AJI A, WANG F, VO H, et al. Hadoop-GIS: A High Performance Spatial Data Warehousing System over MapReduce[J]. PVLDB, 2013, 6(11): 1009–1020.
- [81] NISHIMURA S, DAS S, AGRAWAL D, et al. MD-HBase: design and implementation of an elastic data infrastructure for cloud-scale location services[J]. Distributed and Parallel Databases (DPD), 2013, 31(2): 289–319.
- [82] HUANG S, WANG B, ZHU J, et al. R-HBase: A Multi-dimensional Indexing Framework for Cloud Computing Environment[C] // Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM). 2014: 569–574.
- [83] HUGHES J N, ANNEX A, EICHELBERGER C N, et al. Geomesa: a distributed architecture for spatio-temporal fusion[C] // SPIE Defense+ Security. 2015: 94730F–94730F.
- [84] XIE X, MEI B, CHEN J, et al. Elite: an elastic infrastructure for big spatiotemporal trajectories[J]. VLDB Journal, 2016, 25(4): 473–493.
- [85] YUAN M, DENG K, ZENG J, et al. OceanST: a distributed analytic system for large-scale spatiotemporal mobile broadband data[J]. PVLDB, 2014, 7(13): 1561–1564.
- [86] CHRISTENSEN R, WANG L, LI F, et al. STORM: Spatio-Temporal Online Reasoning and Management of Large Spatio-Temporal Data[C] // Proceedings of the 2015 ACM International Conference on Management of Data (SIGMOD). 2015: 1111–1116.
- [87] 江俊文, 王晓玲. 轨迹数据压缩综述 [J]. 华东师范大学学报 (自然科学版), 2015(5): 61–76.
- [88] HERSHBERGER J, SNOEYINK J. Speeding Up the Douglas-Peucker Line-Simplification Algorithm[C] // Proceedings of International Symposium on Spatial Data Handling. 1992: 134–143.
- [89] MERATNIA N, de BY R A. Spatiotemporal Compression Techniques for Moving Point Objects[C] // Proceedings of the 9th International Conference on Extending Database Technology (EDBT). 2004: 765–782.
- [90] LIU J, ZHAO K, SOMMER P, et al. Bounded Quadrant System: Error-bounded trajectory compression on the go[C] // Proceedings of the 31st IEEE International Conference on Data Engineering (ICDE). 2015: 987–998.

- [91] AGRAWAL R, FALOUTSOS C, SWAMI A N. Efficient Similarity Search In Sequence Databases[C] // Proceedings of 4th International Conference on Foundations of Data Organization and Algorithms (FODO). 1993 : 69 – 84.
- [92] FALOUTSOS C, RANGANATHAN M, MANOLOPOULOS Y. Fast subsequence matching in time-series databases[C] // Proceedings of the 1994 ACM International Conference on Management of Data (SIGMOD). 1994 : 419 – 429.
- [93] RAFIEI D, MENDELZON A O. Similarity-based queries for time series data[C] // Proceedings of the 1997 ACM International Conference on Management of Data (SIGMOD). 1997 : 13 – 25.
- [94] RAFIEI D. On similarity-based queries for time series data[C] // Proceedings of the IEEE 15th International Conference on Data Engineering (ICDE). 1999 : 410 – 417.
- [95] SHATKAY H. The Fourier Transform - A Primer[M]. Providence, Rhode Island 02912, USA : Brown University, 1995 : 186 – 195.
- [96] KEOGH E J, CHAKRABARTI K, PAZZANI M J, et al. Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases[J]. Knowledge and Information System (KIS), 2001, 3(3) : 263 – 286.
- [97] VITTER J S. Random Sampling with a Reservoir[J]. ACM Transactions on Mathematical Software (TOMS), 1985, 11(1) : 37 – 57.
- [98] KEOGH E J, CHU S, HART D M, et al. An Online Algorithm for Segmenting Time Series[C] // Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM). 2001 : 289 – 296.
- [99] MERATNIA N, de BY R A. Spatiotemporal Compression Techniques for Moving Point Objects[C] // Proceedings of the 2004 International Conference on Extending Database Technology (EDBT). 2004 : 765 – 782.
- [100] POTAMIAS M, PATROUMPAS K, SELLIS T K. Sampling Trajectory Streams with Spatiotemporal Criteria[C] // Proceedings of the 18th International Conference on Scientific and Statistical Database Management (SSDBM). 2006 : 275 – 284.
- [101] YIN H, WOLFSON O. A Weight-based Map Matching Method in Moving Objects Databases[C] // Proceedings of the 16th International Conference on Scientific and Statistical Database Management (SSDBM). 2004 : 437 – 438.
- [102] LERIN P M, YAMAMOTO D, TAKAHASHI N. Encoding Travel Traces by Using Road Networks and Routing Algorithms[M]. 2012 : 233 – 243.

- [103] KELLARIS G, PELEKIS N, THEODORIDIS Y. Trajectory Compression under Network Constraints[C] // Proceedings of the 11th International Symposium on Advances in Spatial and Temporal Databases (SSTD). 2009 : 392 – 398.
- [104] KELLARIS G, PELEKIS N, THEODORIDIS Y. Map-matched trajectory compression[J]. Journal of Systems and Software, 2013, 86(6) : 1566 – 1579.
- [105] SONG R, SUN W, ZHENG B, et al. PRESS: A Novel Framework of Trajectory Compression in Road Networks[J]. PVLDB, 2014, 7(9) : 661 – 672.
- [106] SCHMID F, RICHTER K, LAUBE P. Semantic Trajectory Compression[C] // Proceedings of the 11th International Symposium on Advances in Spatial and Temporal Databases (SSTD). 2009 : 411 – 416.
- [107] RICHTER K, SCHMID F, LAUBE P. Semantic trajectory compression: Representing urban movement in anutshell[J]. Journal of Spatial Information Science, 2012, 4(1) : 3 – 30.
- [108] SU H, ZHENG K, ZENG K, et al. STMaker - A System to Make Sense of Trajectory Data[J]. PVLDB, 2014, 7(13) : 1701 – 1704.
- [109] SU H, ZHENG K, ZENG K, et al. Making sense of trajectory data: A partition-and-summarization approach[C] // Proceedings of the 31st IEEE International Conference on Data Engineering (ICDE). 2015 : 963 – 974.
- [110] XIANGYE X, YU Z, QIONG L, et al. Finding Similar Users Using Category-based Location History[C] // Proceedings of the 18th International Conference on Advances in Geographic Information Systems (SIGSPATIAL). 2010 : 442 – 445.
- [111] ZHENG K, SHANG S, YUAN N J, et al. Towards efficient search for activity trajectories[C] // Proceedings of the IEEE 29th International Conference on Data Engineering (ICDE). 2013 : 230 – 241.
- [112] SHANG S, DING R, YUAN B, et al. User oriented trajectory search for trip recommendation[C] // Proceedings of the 15th International Conference on Extending Database Technology (EDBT). 2012 : 156 – 167.
- [113] WANG S, BAO Z, CULPEPPER J S, et al. Answering Top-k Exemplar Trajectory Queries[C] // Proceedings of the 33rd IEEE International Conference on Data Engineering (ICDE). 2017 : 597 – 608.

- [114] SACHARIDIS D, SKOUTAS D, SKOUMAS G. Continuous monitoring of nearest trajectories[C] //Proceedings of the 22nd ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL). 2014 : 361 – 370.
- [115] BAKALOV P, TSOTRAS V J. Continuous Spatiotemporal Trajectory Joins[C] //Proceedings of the 2nd International Conference on GeoSensor Networks (GSN). 2006 : 109 – 128.
- [116] LIAN X, CHEN L, YU J X. Pattern Matching over Cloaked Time Series[C] //Proceedings of the 24th International Conference on Data Engineering (ICDE). 2008 : 1462 – 1464.
- [117] YE H M, WU K, YU P S, et al. PROUD: a probabilistic approach to processing similarity queries over uncertain data streams[C] // Proceedings of the 12th International Conference on Extending Database Technology (EDBT). 2009 : 684 – 695.
- [118] TRAJCEVSKI G, TAMASSIA R, DING H, et al. Continuous probabilistic nearest-neighbor queries for uncertain trajectories[C] //Proceedings of the 12th International Conference on Extending Database Technology (EDBT). 2009 : 874 – 885.
- [119] GOCE T, ROBERTO T, F C I, et al. Ranking continuous nearest neighbors for uncertain trajectories[J]. VLDB Journal, 2011, 20(5) : 767 – 791.
- [120] CHUNYANG M, HUA L, LIDAN S, et al. KSQ: Top- k Similarity Query on Uncertain Trajectories[J]. IEEE Transaction on Knowledge and Data Engineering (TKDE), 2013, 25(9) : 2049 – 2062.
- [121] LI G, LI Y, SHU L, et al. CkNN Query Processing over Moving Objects with Uncertain Speeds in Road Networks[C] // Proceedings of the 13th Asia-Pacific Web Conference on Web Technologies and Applications (APWeb). 2011 : 65 – 76.
- [122] KIM Y, SHIM K. Parallel top- k similarity join algorithms using MapReduce[C] //Proceedings of the IEEE 28th International Conference on Data Engineering (ICDE). 2012 : 510 – 521.
- [123] GOWANLOCK M G, CASANOVA H. Distance threshold similarity searches on spatiotemporal trajectories using GPGPU[C] //Proceedings of the 21st International Conference on High Performance Computing (HiPC). 2014 : 1 – 10.
- [124] ZHANG J, YOU S, LE G. U2STRA:high-performance data management of ubiquitous urban sensing trajectories on GPGPUs[C] // Proceedings of the 2012 ACM Workshop on City Data Management Workshop (CDMW). 2012 : 5 – 12.

- [125] LEAL E, GRUENWALD L, ZHANG J, et al. TKSImGPU: A parallel top-K trajectory similarity query processing algorithm for GPGPUs[C] // Proceedings of the 2015 IEEE International Conference on Big Data (ICBD). 2015 : 461 – 469.
- [126] LEAL E, GRUENWALD L, ZHANG J, et al. Towards an Efficient Top-K Trajectory Similarity Query Processing Algorithm for Big Trajectory Data on GPGPUs[C] // Proceedings of the 2016 IEEE International Conference on Big Data (ICBD). 2016 : 206 – 213.
- [127] ZEINALIPOUR-YAZTI D, LIN S, GUNOPULOS D. Distributed spatio-temporal similarity search[C] // Proceedings of the 15th ACM International Conference on Information and Knowledge Management (CIKM). 2006 : 14 – 23.
- [128] PAPADOPOULOS A, MANOLOPOULOS Y. Distributed Processing of Similarity Queries[J]. Distributed and Parallel Databases, 2001, 9(1) : 67 – 92.
- [129] KASHYAP S, KARRAS P. Scalable kNN search on vertically stored time series[C] // Proceedings of the 17th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD). 2011 : 1334 – 1342.
- [130] SHANG J, ZHENG Y, TONG W, et al. Inferring gas consumption and pollution emission of vehicles throughout a city[C] // Proceedings of the 20th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD). 2014 : 1027 – 1036.
- [131] 龚旭东. 轨迹数据相似性查询及其应用研究 [D]. 合肥: 中国科学技术大学: 中国科学技术大学, 2015.

附录 主要缩写符号对照表

ED	欧氏距离 (Euclidean Distance)
DTW	动态时间弯曲 (Dynamic Time Warping)
LCSS	最长公共子串 (Longest Common Subsequence)
EDR	实值序列上的编辑距离 (Edit Distance on Real sequence)
ERP	带有惩罚项的编辑距离 (Edit Distance with Real Penalty)
HD	豪斯多夫距离 (Hausdorff Distance)
FD	弗雷歇距离 (Fréchet Distance)
OWD	一路距离 (One Way Distance)
DEwP	带投影的编辑距离 (Edit Distance with Projections)
P2P	点对点 (Peer-To-Peer)
DP	道格拉斯-普克 (Douglas-Peucker)
SVD	奇异值分解 (Singular Value Decomposition)
DFT	离散傅里叶变换 (Discrete Fourier Transform)
HWT	哈尔小波变换 (Haar Wavelet Transform)
PAA	分段聚合近似 (Piecewise Aggregate Approximation)
MDL	最小描述长度 (Minimal Description Length)
APCA	自适应分段常量近似 (Adaptive Piecewise Constant Approximation)
SED	欧氏距离的平方 (Squared Euclidean Distance)
BE	包围信封 (Bounding Envelope)
MBR	最小包围矩形 (Minimum Bounding Rectangle)
BF	界特征 (Bound Feature)
LBF	下界特征 (Lower Bound Feature)

致 谢

回首十一年的本、硕、博三阶段求学时光，经历了失落与荣耀。求学路上，曾目睹了一届届同学离开校园走上工作岗位的背影。此刻的我，即将博士毕业，感慨良多。在此期间，有幸得到老师、同学和亲友们的指导与帮助，在此谨对他们表示衷心的感谢！

首先，我要郑重感谢我人生中的三位引路人。他们分别是吉根林、周傲英和金澈清三位教授。吉根林教授是我在南京师范大学本、硕阶段的指导老师。他对待学生视若己出，为我们创造了良好的学习环境与和热烈活跃的团队氛围。并在我即将硕士毕业的十字路口，为我介绍了人生中第二位引路人周傲英教授。周老师是国内首屈一指的数据库领域专家，他在数据库领域的远瞩高瞻、对新兴学科和领域也能高屋建瓴。为此，我毅然来到他在华东师范大学的团队，拜师学艺。进入团队，我很快遇到了我的第三位引路人金澈清教授。金老师对待学术事无巨细且严谨有方，教导学生谆谆善诱。难以忘怀，在多个节假日期间，金老师牺牲自己跟家庭团聚的时间帮我们修改论文。在周、金两位老师的指导下，我不仅学术水平突飞猛进发表若干学术论文，而且项目管理和实践能力也得到较大提高，成功带领实验室同学完成两项校企合作项目。除此之外，周、金两位老师在我母亲生病期间给我很大安慰和温暖，帮我度过了人生中最艰难的时刻。

其次，我十分感谢毛嘉莉老师，毛老师在我读博四年时光里，始终像知心大姐一样给我关心和鼓励。此外，在我几次投稿期间花费许多精力和时间与我讨论研究问题和研究方法，修改学术论文，让我学习了大量的学术论文写作技巧。然后，我还要感谢南京师范大学赵斌老师。感谢赵老师始兄长般的热情和帮助。也特别感谢华东师范大学钱卫宁教授、王晓玲教授、何晓丰教授、张蓉教授、高明副教授和张召副教授在日常学习与研究生课程中给予的多方面帮助和指导。

另外，感谢所有读书期间陪伴我的同学和朋友，你们是我的美好记忆。感谢已经毕业的立、超、俊、科等师兄，以及建松、康强、小艺、扬鑫、秋革、包婷、杨小林和廖春和等师弟师妹，感谢你们在我研究生前期给予的学习上的帮助和生活上的快乐，难忘一起写论文和一起做项目的点点滴滴。感谢王艺霖、杨小林、咸晓冬，你们为我的论文提出了宝贵的修改意见。特别感谢陪

我度过研究生时光的李财政、方祝和和肖冰，谢谢你们几年来对我的关心和包容。感谢一起毕业的毛嘉莉博士、郭敬伟博士、刘辉平博士和朱涛博士，与你们一起毕业是我的荣幸。也祝尚在奋斗的乔宝强、戚晓冬、申弋斌、方祝和等博士顺利完成学业，早日毕业。感谢在 LBS 课题组一起学习的陈鹤、王婧、李敏茜、施晋、华嘉逊、濮敏等师弟师妹们。此外，我还想感谢燕存、李文俊、丁敬恩和徐寅等朋友，谢谢你们这些年一直对我的关心和照顾。

最后，着重感谢我的家人，对我无私的爱以及对我攻读博士学位的大力支持。

章志刚

二零一八年四月

攻读博士学位期间发表论文

■ 已公开发表论文

- [1] **Zhigang Zhang**, Xiaodong Qi, Yilin Wang, Cheqing Jin, Jiali Mao, Aoyin Zhou. Distributed Top- k Similarity Query on Big Trajectory Streams. *Frontiers of Computer Science* (FCS), to be published, 2018.
- [2] **Zhigang Zhang**, Jiali Mao, Cheqing Jin, Aoying Zhou. MDTK: Bandwidth-saving Framework for Distributed Top- k Similar Trajectory Query. In Proceedings of *the 23rd Database Systems for Advanced Applications (DASFAA)*, to be published, 2018.
- [3] **Zhigang Zhang**, YilinWang, Jiali Mao, Cheqing Jin, Aoying Zhou. DT-KST: Distributed Top- k Similarity Query on Big Trajectory Streams. In Proceedings of *the 22nd Database Systems for Advanced Applications (DASFAA)*, pages 199-214, 2017.
- [4] **Zhigang Zhang**, Cheqing Jin, Jiali Mao, Xiaolin Yang, Aoying Zhou. TrajSpark: A Scalable and Efficient in-memory Management System for Big Trajectory Data. In Proceedings of *Web and Big Data - First International Joint Conference, Part I (APWeb-WAIM)*, pages 11-26, 2017.
- [5] 章志刚, 金澈清, 王晓玲, 周傲英. 面向海量低质手机轨迹数据的重要位置发现. *软件学报*, 27 卷, 7 期, 1700-1714, 2016.
- [6] Jiali Mao, Qiuge Song, Cheqing Jin, **Zhigang Zhang**, Aoyin Zhou. Online clustering of streaming trajectories. *Frontiers of Computer Science* (FCS), volume 12, pages 245-263, 2018.
- [7] Xiaolin Yang, **Zhigang Zhang**, Yilin Wang, Cheqing Jin. Distributed Continuous KNN Query over Moving Objects. *Internation Journal on High Performance Computing and Networking*, to be published, 2017.
- [8] 毛嘉莉, 金澈清, 章志刚, 周傲英. 轨迹大数据异常检测: 研究进展及系统框架. *软件学报*, 28 卷, 1 期, 17-34, 2017.
- [9] 王艺霖, 章志刚, 金澈清. 智能交通卡记录中的公交站点恢复方法. *华东师范大学学报 (自然科学版)*, 5 卷, 201-212, 2017.

- [10] Cheqing Jin, Ashwin Lall, Jun(Jim) Xu, **Zhigang Zhang**, Aoying Zhou. Distributed Error Estimation of Functional Dependency. *Information Scineces* , volume 345, pages 11-26, 2017.
- [11] Jiali Mao, Qiuge Song, Cheqing Jin, **Zhigang Zhang**, Aoying Zhou. TSCluWin: Trajectory Stream Clustering over Sliding Window. In Proceedings of *the 21th Database Systems for Advanced Applications (DASFAA)*, pages 133-148, 2016.
- [12] 包婷, **章志刚**, 金澈清. 基于手机大数据的城市人口流动分析系统. 华东师范大学学报(自然科学版), 5 卷, 162-171, 2015.