分类号：TP391　520.4050　　密级：

天津理工大学研究生学位论文

基于路网的移动对象轨迹相似性查询方法研究

（申请工程硕士学位）

工程领域：计算机技术

作者姓名：沙文强

指导教师：肖迎元　教授

2015 年 1 月

Thesis Submitted to Tianjin University of Technology for the Master's Degree

Research on Trajectory Similarity Search of Moving Object based on Road Network

By

Sha Wenqiang

Supervisor

Xiao Yingyuan

January, 2015

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作和取得的研究成果，除了文中特别加以标注和致谢之处外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 天津理工大学 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文作者签名：　　签字日期：　　年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解 天津理工大学 有关保留、使用学位论文的规定。特授权 天津理工大学 可以将学位论文的全部或部分内容编入有关数据库进行检索，并采用影印、缩印或扫描等复制手段保存、汇编，以供查阅和借阅。同意学校向国家有关部门或机构送交论文的复本和电子文件。

（保密的学位论文在解密后适用本授权说明）

学位论文作者签名：　　导师签名：

签字日期：　　年　　月　　日　　签字日期：　　年　　月　　日

摘要

近年来，随着智能手机、GPS (Global Positioning System) 的广泛使用以及基于位置服务 (Location-Based Service) 的普及，大量的轨迹数据正在日益积累并为各种各样的应用服务。同时海量的轨迹数据也对存储、运算、处理和管理带来了巨大的挑战。

轨迹可以看作是移动对象随着时间的变化在空间中留下的印迹。轨迹相似性查询已逐渐成为数据库研究领域中一个新的研究热点。本文中在交通路网下针对两种轨迹查询类型：轨迹时空相似性查询和轨迹空间-文本相似性查询展开研究，主要工作如下：

(1) 提出一种路网环境下移动对象轨迹时空相似性查询算法，该算法首先使用网络 Voronoi 图对路网空间进行划分，进行部分网络距离的预计算；然后采用两阶段处理策略：空间过滤阶段和时间提纯阶段。在空间过滤阶段，以查询点为中心进行网络 Voronoi 扩张，将扩张范围内的轨迹加入到候选轨迹集合中；在时间提纯阶段，对于候选轨迹集合中的每一条轨迹计算时间相似性值，并最终计算出时空相似性值，找到时空相似性值最小的轨迹并作为返回结果。

(2) 提出一种路网环境下移动对象轨迹空间-文本相似性查询算法，该算法主要包括三个阶段：最小点匹配计算阶段，候选轨迹生成阶段和候选轨迹验证阶段。在最小点匹配计算阶段，以查询中的每一个查询点为中心逐渐进行网络范围扩张，找到范围内轨迹的最小点匹配，并计算出最小点匹配距离；在候选轨迹生成阶段，根据计算出的最小点匹配距离，生成候选轨迹集合；在候选轨迹验证阶段，对候选轨迹集合中的每一条轨迹按照最小点集匹配距离升序排列，选出前 k 条轨迹作为最终返回结果。

(3) 对上述提出的算法进行了广泛的实验，验证了算法的有效性。

关键词： 轨迹相似性　路网　网络 Voronoi 图　移动对象

Abstract

In recent years, with wide use of smart phones and GPS (Global Positioning System) as well as population of Location-Based Services, a large amount of trajectory data has been increasingly getting its accumulation to serve for a variety of applications. At the same time, trajectory data also brings about great challenge on storage, operation, processing and management due to its large scale.

Trajectory can be regarded as a track of a moving object with the change of time left in the space. Trajectory similarity search has gradually become a new research hot issue in the database field. In this paper, we consider two types of trajectory search: trajectory spatio-temporal similarity search and trajectory spatial-textual similarity search under the road network. The main work is as follows:

(1) A novel algorithm of trajectory spatio-temporal similarity search of moving objects based road network is put forward. The algorithm first uses network Voronoi diagram to divide the network space and pre-computes some necessary network distance. It adopts the two-phase strategy: spatial filtering step and temporal refinement step. In the spatial filtering step, it uses the query point as the center for network Voronoi expansion and then adds the trajectories within the expansion range into the trajectory candidate set; in the temporal refinement step, for every trajectory in the candidate set, it computes the temporal similarity and spatio-temporal similarity. Finally the trajectory with smallest spatio-temporal similarity will be returned as a result.

(2) A novel algorithm of trajectory spatial-textual similarity search of moving objects based road network is put forward. The algorithm mainly includes three steps: minimum point matching calculation step, trajectory candidate generation step and trajectory candidate verification step. In the minimum point matching calculation step, it first uses the query point as the center for network expansion, then finds the minimum point match to the trajectory within the expansion and computes the minimum point match distance; in the trajectory candidate generation step, it generates the trajectory candidate set based on the minimum point match distance computed; in the trajectory candidate verification step, for every trajectory in the candidate set, it computes the minimum point set match distance and sorts the trajectories in ascending order according to the distance. Finally the first k trajectories will be returned as a result set.

(3) Extensive experiments based on the two algorithms proposed above are performed and verify good effectiveness of the algorithms.

Key words： Trajectory Similarity, Road Network, Network Voronoi Diagram, Moving Object

目　录

第一章 绪论

1.1 研究背景及意义

我们正处在一个信息爆炸的时代。数据，作为推动这场巨大变革的原材料，正通过GPS、传感器以及其它的数据采集设备源源不断的被收集起来。这其中一类很重要的数据就是轨迹数据。随着移动通讯设备的快速发展，人们对移动对象轨迹的分析和研究越来越重视，如何高效管理移动对象轨迹数据成为目前的研究热点之一。

轨迹可以看作是移动对象随着时间的变化在空间中留下的印迹[1]。近年来，随着智能手机，全球定位系统 GPS (Global Positioning System) 等移动终端的广泛使用以及基于位置服务 LBS (Location-Based Service)的普及，大量的轨迹数据日益积累起来，并服务于各种各样的应用。例如，在互联网上，人们在浏览网页时随之产生的超链接点击序列的轨迹；在 Bikely，Share-My-Routes 等社交网站中，用户在这些网站上分享自己的出行路线，并在各地点分享自己的生活体验；气象学家为寻找气候变化的规律，收集大量的飓风运动轨迹数据；生物学家为研究动物的行为活动，采集了动物的运动数据。因而我们知道，海量的移动对象轨迹数据充斥在人们周围，这些数据在为我们提供了便利的生活，也为数据的应用和管理提出了巨大的挑战。

面对浩瀚如烟的移动对象轨迹数据，如何从中挖掘出隐藏的有效信息越来越受到研究者们的重视，越来越多的人投身到移动对象轨迹数据的研究上。其中，分析和研究移动对象轨迹一个常见的应用如下：“对于用户给定的一条移动对象轨迹，从轨迹数据库中查找出与其最为相似的一条或者多条轨迹”。这样的应用有着广泛的应用场景。例如，旅行者们可以通过参考其他人的旅行路线来提高自己的出行体验，做好出行计划；警方部门可以通过分析事故现场的历史时空轨迹来找到潜在的肇事者和目击者。

由此看来，移动对象轨迹数据有着广阔的应用场景，在交通管制、移动电子商务、物流配送、决策支持、数据挖掘等领域，移动对象轨迹扮演着重要的角色。因此，移动对象轨迹数据的高效利用成为时空数据库研究中亟待解决的问题，也是当前数据库领域中的研究热点。

近些年来，大多数的研究者们因移动对象轨迹的定义及计算简单易行，选用在欧氏空间下进行研究与分析。但是在实际的应用中，大多数移动对象都是受限于道路交通网络，移动对象在路网环境下的移动不再是任意方向的，它将沿着指定的路线运动，之前的基于欧氏空间的相似性度量方法已经不再适用于路网环境下，因此研究路网环境条件下的移动对象轨迹相似性更加具有实际意义。

由于道路交通网络[2-5]的特性，可以通过语义信息对道路网络空间中的轨迹进行建模。本文的研究思路基于以下两点考虑：第一，为保证路网空间中轨迹处理的正确性，

应该对真实路网进行很好的表示，同时兼顾移动对象轨迹在路网环境下的表示，这样才能更准确的计算轨迹之间的相似性。第二，不仅仅考虑移动对象轨迹的空间特性，还应该同时考虑轨迹中其他的语义信息，如：时间特性和文本特性。基于此，应该提出基于轨迹空间-时间相似性和轨迹空间-文本相似性的度量方法。最后根据这两点，设计出一个切实可行的算法来高效地响应查询。

1.2 研究现状

移动对象轨迹之间的相似程度是由轨迹之间的距离来进行判定的，一个好的轨迹相似性度量方法可以更好的对轨迹之间的相似性进行判定。基于相似性度量方法，设计一种查询处理算法高效解决移动对象轨迹相似性问题。目前研究者们对于轨迹相似性的研究环境主要集中在欧氏空间、路网环境和网格空间。

1.2.1 欧氏空间下的研究现状

欧式空间的移动对象轨迹相似性度量方法已经得到了的广泛的研究，比如 EU (Euclidean Distance)，DTW (Dynamic Time Warping Distance)[6]，LCSS (Longest Common Subsequence)[7]，ERP (Edit Distance with Real Penalty)[8]和 EDR (Edit Distance in Real Sequence)[9]。

对于给定的拥有相等长度的两条轨迹，EU 计算两条轨迹之间的坐标点的欧氏距离。这种方法使得计算更加简单也更快，然而它要求两条轨迹的长度必须相等，因此它的应用范围有限。DTW 的测量是通过复制前一个时空坐标点值来实现两条轨迹的局部时间转换，尽管这种方法没有两条轨迹长度一致的限制，但是这种方法容易受到噪声的影响。ERP 引入一个常量值(gap)，它在两条轨迹之间的元素使用编辑距离来作为处理局部时间转换的惩罚值，像 DTW 一样，ERP 也可以处理不同长度的轨迹，因为它允许在两条轨迹之间存在间隙(gap)，然而这种方法对数据噪声非常的敏感。EU，DTW，ERP 都会受到噪声数据的影响，因而当这些方法应用到路网时，它们的准确性会很差。LCSS 需要一个阈值，这个阈值可以确定两个元素是否匹配。这种方法通过量化元素之间的距离(0 或 1)来减小数据噪声，因此去除了由噪声引起的较大的距离影响，然而这种方法并不考虑相似子序列的间隙变化，这就有可能在相似性测量的时候带来不可忽略的影响。EDR 也是基于编辑距离的，它是通过量化元素对之间的距离(0 或 1)来减去除数据噪声的影响，同时，它根据间隙的长度将惩罚值分配给两条匹配的子轨迹的间隙，这将使得 EDR 比 LCSS 方法更加精确。

另外，一些研究者们也专注于轨迹的外形相似性度量的研究。Lin[10]等人引入了一种新的相似性度量方法—one way distance，并据此设计出了一种计算相似性度量的高效算法，同时引入一种新的索引结构保证查询性能。Yanagisawa[11]等人将轨迹定义为空间中的直线数据模型，轨迹之间的距离是通过对应直线段的欧氏距离来衡量的。另外，他们为加快查询速度引入了一种新的索引结构 PAA(Piecewise Approximate Aggregate)，这种索引结构有效融合了 R+树和维度降低策略。

Wang[12]等人提出了一种面向用户的轨迹相似性查询类型，有效的考虑了用户的偏好。他们为面向用户的轨迹相似性查询查询定义了新的数据模型，采用了一种基于 LCSS 的相似性度量方法—HCSS(Heaviest Common Subsequence)。另外用户在指定查询轨迹的时候，可以对轨迹点指定权值，充分考虑用户需求。查询处理算法使用自适应过滤策略和提纯策略来加快查询速度。

以上这些基于欧氏空间的相似性度量方法并不能直接应用到路网空间中去。另一方面，这些度量方法并不能从轨迹数据库中完全地挖掘出轨迹信息。

1.2.2 路网空间下的研究现状

Hwang[13]等人首次提出在路网环境下的移动对象轨迹时空相似性查询方法，它是基于轨迹的时间和空间特性来检索移动对象的相似轨迹的。其算法主要分为两个步骤：基于空间距离的过滤阶段和基于时间距离的提纯阶段。然而这种方法中存在缺点：(1) POI 必须提前由用户来指定；(2) 在兴趣点集 P 中，一个简单点的选择错误都会影响到轨迹的空间相似性；(3) 提出的相似性度量方法无法直观的表示两条轨迹是相似性程度，因为其方法并没有考虑到相似性比例或者相似性取值范围；(4) 两条轨迹的空间相似性是基于它们之间存在的公共点，因此许多的相似的轨迹被排除在外。

E.Tiakas[14]等人在 Hwang 的基础上做出了改进，他们仍然将时间和空间特性考虑进来，他们定义了新的基于轨迹空间和时间的特征的相似性测度，同时引入了一种基于 M 树的索引来加快查询处理过程，并在查询处理方法上做出了更为详尽的分析。他们提出 (1) 轨迹中的距离是路网中两个顶点的最短的网络距离，在此基础上提出了两种度量标准来测量轨迹之间的相似性，并且满足度量空间性质；(2) 通过引入合适的索引机制来保证高效的相似性查询。

Chen[15]等人使用一系列的位置作为查询源从轨迹数据库中查找最佳匹配轨迹（Best-Connected Trajectories），他们定义了一种新的轨迹相似性度量方法，同时考虑了轨迹点顺序，并设计出了 IKNN 的查询处理算法。论文中同时对查询处理算法进行了详细的数学分析与证明。

Jung-Im Won[16]等人研究了路网环境的轨迹聚类[17]问题。他们定义，路网下轨迹表示为移动对象所经过路段的时间序列，每一个路段记录有此路段的长度。另外提出了通过考虑两条轨迹之间匹配的路段长度来判断轨迹相似性程度的度量方法，基于此设计出了一个新的聚类方法。

Shang[18]等人在移动对象轨迹相似性问题上提出了一种新的查询类型—轨迹空间-文本相似性查询，这种情况下，轨迹之间的相似性不仅考虑空间特性，而且考虑到轨迹的文本特性，分别计算轨迹之间的空间相似性值和文本相似性值，最终通过一个权值来综合衡量空间-文本相似性值，这样可以更好的满足用户的偏好。但是轨迹的文本特性是作为轨迹的一个整体描述，而非单个轨迹位置的文本描述。另外相似性度量方式并不能捕获轨迹位置与关键字二者之间的联系，也有可能不会返回一条轨迹。

Zheng[19]等人的研究的轨迹含有活动信息，他们称这样的轨迹为活动轨迹，并据此提出了一种新的查询类型—ATSQ 查询，这个查询将返回 k 条轨迹能够包含所有查询活

动并且拥有最小的匹配距离。在此基础之上又提出了另外一种查询类型—OATSQ，这种查询类型能够考虑查询轨迹与查找到的轨迹的顺序一致性。为了保证查询的高效执行，引入了一种混合索引结构—GAT，按照轨迹的分段和查询活动层次进行组织，并能够保证通过空间相近性和包含的活动来同时剪枝查找空间，提高查询速度。

1.2.3 网格空间下的研究现状

网格空间广泛应用于 GIS 应用中，是空间表示的另外一种类型。Hye-Young Kang[20]等人是在网格空间下研究轨迹相似性的，轨迹在网格空间表示为移动对象随时间变化的格子序列，移动对象在每一个格子都有一个进入时刻和离开时刻。基于网格空间他们提出了两种轨迹相似性度量方法：最大公共子序列方法和公共访问时间段方法。最后通过实验验证这两种相似性度量方法的性能。Sultan Alamri[21]等人的研究内容是在网格空间下对移动对象进行索引，他们提出了一种基于网格连通性的索引结构—indoor-tree，这种索引能够保证移动对象的高效查询处理和更新。

目前研究都假设：移动对象可以在在二维或者三维空间中随意移动，然而在现实生活中，路网条件下轨迹之间的相似性的研究更加具有实用价值。因移动对象在路网的诸多限制，轨迹之间距离的巨大计算量，如何定义更为精确的相似性度量方法以及找到一种高效率的算法是在路网环境下研究轨迹相似性查询的关键。另外，之前的研究大多注重的是轨迹之间的空间特性，并没有过多的考虑轨迹的时间特性和文本特性。然而随着人们更多的查询要求，用户会在查询轨迹中加入自己的偏好和限制，轨迹的时间特性和文本特性的作用日益凸显，查询处理算法如何保障找到的轨迹同时在空间、时间、文本特性上满足用户需求，又能够保查询效率，这将是本文的研究重点。

1.3 研究内容

本文针对基于路网的移动对象轨迹相似性查询方法展开研究，主要研究内容阐述如下：

(1) 路网环境下移动对象及其轨迹的定义方法，对移动对象轨迹相似性查询问题的形式化定义。为更好的解决路网环境下的轨迹相似性问题，需要对路网环境下的移动对象及其轨迹进行抽象并建成模型，给出相应的表示方法。

(2) 路网环境下轨迹之间的相似性度量标准。对轨迹相似性的度量主要依赖于轨迹之间的“距离”，而不同的轨迹表示形式在轨迹的“距离”计算是也会有所不同。本文根据路网空间中的移动对象轨迹的空间特性、时间特性和文本特性，定义出轨迹之间的空间相似性、时间相似性和文本相似性，在此基础上完成轨迹时空相似性和空间-文本相似性的查询定义。这一部分是论文的核心之一，直接关系着轨迹之间相似性的精确度。

(3) 基于路网的移动对象轨迹时空相似性查询方法的研究。主要包括本研究中的相关定义和算法的基本思想，算法中使用到的数据结构，设计一种新的移动对象轨迹时空相似性查询处理算法，高效解决轨迹时空相似性查询问题。

(4) 基于路网的移动对象轨迹空间-文本相似性查询方法的研究。主要介绍了本研究

中的相关定义以及算法的基本思想，然后在此基础上设计查询处理算法，高效响应用户查询。

(5) 通过实验验证所设计算法的有效性。通过在真实的路网数据集下进行实验，通过与已有的算法进行比较，验证所设计的算法的性能。

1.4 论文组织结构

本文一共分为五章，论文的组织结构如下：

第一章为绪论。主要介绍了本文的研究背景、研究意义、国内外的研究现状和研究内容。

第二章介绍了研究工作的前期准备和背景知识。该章主要包括移动对象轨迹表示方法、移动对象轨迹相似性度量方法、移动对象轨迹索引的基本概念和相关理论。

第三章提出了一种基于路网的移动对象轨迹时空相似性查询方法。该章主要介绍了相关定义和算法的基本思想以及详尽的查询处理过程，并对设计的查询处理算法进行实验验证。

第四章提出了一种基于路网的移动对象轨迹空间-文本相似性查询方法。该章主要介绍了相关定义和算法的基本思想，然后在此基础上设计查询处理算法，最后通过大量实验对设计的查询处理算法进行验证。

第五章是总结与展望。对本文的主要内容及贡献进行了总结，并指出了进一步的研究方向。

第二章 基本概念和相关理论

移动对象数据库基本功能主要有两个：一是对移动对象数据的存储；二是对移动对象数据查询的支持。现在的大多数传统的数据库系统对连续变化的数据并不能进行有效的处理，如移动对象位置数据。这是因为在传统数据库中数据通常是相对稳定的，而移动对象的位置信息则是随时间连续改变的。显然，传统关系数据库很难实现对移动对象位置建模和实时更新。因而如何实现对移动对象有效管理便成为亟待解决的问题。

2.1 移动对象轨迹的表示方法

移动对象轨迹的查询类型主要包含两类，一是对移动对象历史轨迹的查询，二是对移动对象当前和未来轨迹的查询。本文主要立足于第一类，针对移动对象历史轨迹之间的相似性查询方法展开研究，为移动对象轨迹数据库的查询提供给支持。

在不同的应用场景下，为了更好的适应查询需要，轨迹应该有不同的表示形式。下面将对已有的轨迹表示方法进行阐述。

一般情况下，轨迹可以表示为 $T = <p_1, p_2, \ldots, p_n>$，在欧式空间下，$p_i = <lon_i, lat_i>$, lon_i 表示经度，lat_i 表示纬度。在路网空间下，将整个路网抽象成一张图 $G = (V, E)$，那么此时 p_i 可以表示这张图的某个顶点。由于在真实的道路交通网络环境下，利用 GPS 等位置采集设备获取的轨迹采样点也是经纬度，为了能够将采样点映射到图的顶点上，可以采用 map-matching 算法[22-25]将采样点映射到图的节点当中。

如果考虑轨迹的时间特性，轨迹表示为 $T = <(p_1, t_1), (p_2, t_2), \ldots, (p_n, t_n)>$，$p_i$ 的表示如上所述，t_i 表示移动对象经过 p_i 的时刻。如果考虑轨迹的文本特性，轨迹可以表示为 $T = <(p_1, k_1), (p_2, k_2), \ldots, (p_n, k_n)>$，$p_i$ 的含义如上所述，k_i 表示在 p_i 处的关键字描述。

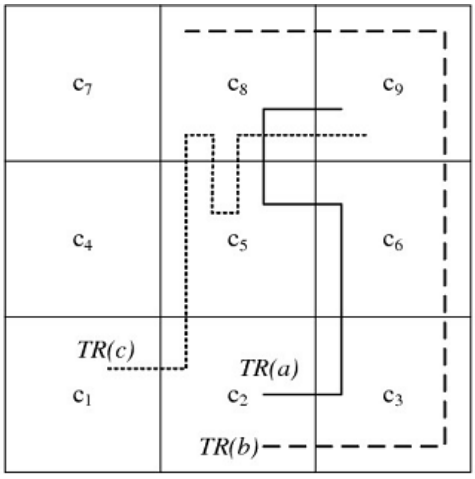


图 2-1 轨迹的网格表示法

路网环境下，轨迹的另一表示方式：$T = \{(S_1, L_1), (S_2, L_2), \ldots, (S_n, L_n)\}$，其中 S_i 代表路网中的路段，L_i 代表路段 S_i 的长度。

此外，网格表示法也是常用的轨迹表示方式：$TR = \{(c, I) \mid c \in C, I=(t_{in}, t_{out})\}$，其中，c 表示空间中的网格。$t_{in}$ 表示进入网格的时刻，t_{out} 表示离开网格的时刻。示例如图 2-1 所示，其中轨迹 $TR(a)$、$TR(b)$和 $TR(c)$可分别表示如下：

$TR(a)$={(c_2,[0,3)),(c_3,[3,6)),(c_6,[6,8)),(c_5,[8,9)),(c_8,[9,14)),(c_9,[14,18))}；

$TR(b)$={(c_2,[0,2)),(c_3,[2,4)),(c_6,[4,8)),(c_9,[8,12)),(c_8,[12,18))}；

$TR(c)$={(c_1,[0,2)),(c_2,[2,6)),(c_5,[6,8)),(c_8,[8,11)),(c_5,[11,12)),(c_8,[12,15)),(c_9,[15,18))}。

总之，轨迹在不同的应用场景下有着不同的表示方式，采用不同的形式可以更好的适用于研究问题，有助于将问题简化。另外轨迹还可以存储额外的信息，比如用户的兴趣，移动对象的速度等。

2.2 移动对象轨迹相似性度量方法

移动对象轨迹具有时间特性和空间特性，移动对象轨迹的相似性度量或者说轨迹的时空相似性度量来源于研究人员对时间序列的相似性分析，时空轨迹从本质上说也是一种时间序列的数据。在不同的应用场景下，对于轨迹之间的相似性要求也不尽相同。有的场景下要求发现轨迹中的子轨迹部分即可，有的场景应用需要发现某个时间段的相似性轨迹。因此，在度量移动对象轨迹的相似性时要根据具体的应用场景选择不同的度量方法。

2.2.1 欧氏距离

轨迹的欧氏距离是通过计算每个时间点上轨迹所对应位置上两个点的欧氏距离，然后将所有的点对的距离进行综合处理所得。处理时可以采用平均值、求和、取中值等方式。轨迹 A 与 B 的欧氏距离计算如式 2-1 和 2-2 所示：

$$EU(A, B)=\sum_{k=1}^{n} dist(p_k^A, p_k^B) \tag{2-1}$$

$$dist(p_k^A, p_k^B)=\sqrt{(p_{k,x}^A - p_{k,x}^B)^2 + (p_{k,y}^A - p_{k,y}^B)^2} \tag{2-2}$$

欧氏距离计算简单，但易受噪声影响。而且在现实条件下，轨迹之间的采样点在时间和个数上一般都不是相同的，因此采用此方法时需对移动对象轨迹数据进行预处理。

2.2.2 动态时间封装距离

位置采集设备的误差导致不能保证轨迹所有的时刻点上都能捕捉到移动对象的位置，而且不同的轨迹也不能保证所有的位置点的采集时刻是对应的。在这种情况下，需要有一种度量方法能够将时间维度进行拉伸。DTW 正是因此而提出的，计算公式如式

2-3 所示：

$$DTW(A,B)=\begin{cases}0, & \text{if } n=0 \text{ and } m=0\\ \infty, & \text{if } n=0 \text{ or } m=0\\ d(Head(A),Head(B))+\min\begin{cases}DTW(A,Rest(B))\\ DTW(Rest(A),B)\\ DTW(Rest(A),Rest(B))\end{cases}\end{cases} \quad (2\text{-}3)$$

其中 DTW(A, B)表示轨迹 A 与 B 之间的动态时间封装距离。m 和 n 分别表示轨迹 A 和 B 的采样点的数目。Rest(A)和 Rest(B)分别表示轨迹 A 和 B 上除第一个点外剩下的轨迹部分。从计算公式可以得知，当两条轨迹长度都为零时，DTW 值也为零；当两条轨迹有一条长度为零时，DTW 值为无穷大；当两条轨迹的长度都不为零时，采用递归计算的方式求解一个最小的动态时间封装距离。

2.2.3 最长公共子序列距离

最长公共子序列距离指的是两个序列中的最长的公共子序列长度。针对移动对象轨迹，可以使用两条轨迹中的最长的相似子轨迹来衡量轨迹之间的相似性。计算公式如式 2-4 所示：

$$LCSS(A,B)=\begin{cases}0, \text{if } n=0 \text{ or } m=0\\ 1+LCSS(Rest(A),Rest(B)), \text{ if } d(Head(A),Head(B))\le\varepsilon, \text{and} \mid n-m\mid<\delta\\ \max(LCSS(R\ est(A),B),LCSS(A,Rest(B))), \text{otherwise}\end{cases} \quad (2\text{-}4)$$

其中，LCSS(A, B)表示轨迹 A 和 B 之间的最长公共子序列的长度，参数 δ 是用来控制多大程度上可以实时地匹配一条轨迹的给定点到另一条轨迹的一个点，参数 ε 是一个决定是否将这个点考虑在内的空间匹配阈值。当两条轨迹的长度都为零时，LCSS 的值为零；当两条轨迹的起点的距离处于某一阈值范围内，LCSS 的值为除起点外剩余轨迹的 LCSS 值加一。否则，在一条轨迹与另一条轨迹的剩余部分中寻找一个最大的 LCSS 值。

2.2.4 编辑距离

编辑距离最早是用于比较字符串之间距离的一种方式，是指一个字符串通过增减、删除、修改操作转变成另一个字符串所需要的最小操作次数。将此种方法应用到移动对象轨迹相似性的度量上去，则是将一条轨迹转换成另一条轨迹所需要的最小开销。编辑距离的计算包括两种方式，分别是 EDR 和 ERP，它们采用递归的方式进行计算，计算公式如式 2-5 所示：

$$EDR(A,B)=\begin{cases} n, & \text{if } m=0 \\ m, & \text{if } n=0 \\ \min\{EDR(Rest(A),Rest(B))+subcost, & \text{otherwise} \\ EDR(Rest(A),B)+1, EDR(A,Rest(B))+1\} & \end{cases} \quad (2\text{-}5)$$

其中 $subcost=\begin{cases} \text{if } d(Head(A),Head(B)) \le \varepsilon \\ 1, \text{otherwise} \end{cases}$

EDR(A, B)表示轨迹 A 和 B 之间的编辑距离，m 和 n 分别表示 A 和 B 上轨迹点的数目。Rest(A)和 Rest(B)分别表示轨迹 A 和 B 上除第一个点外剩下的轨迹部分。当一条轨迹的点的数目为零时，两条轨迹之间的编辑距离就是另一条轨迹的点的数目；如果两条轨迹的起点的距离处于某个阈值 ε 下，则轨迹之间的编辑距离就是剩余的轨迹部分的编辑距离；如果没有相同的起点，则通过递归调用寻找最小开销。

ERP 是编辑距离的另外一种表示方式，计算公式如式 2-6 所示：

$$ERP(A,B)=\begin{cases} \sum_{i=1}^{n}|s_i-g|, & m=0 \\ \sum_{i=1}^{n}|r_i-g|, & n=0 \\ \min\begin{cases} ERP(Rest(A),Rest(B))+d(Head(A),Head(B)) \\ ERP(Rest(A),B)+d(Head(A),g) \\ ERP(A,Rest(B))+d(Head(B),g) \end{cases} & \text{otherwise} \end{cases} \quad (2\text{-}6)$$

EDR(A, B)表示轨迹 A 和 B 之间的编辑距离，m 和 n 分别表示 A 和 B 上轨迹点的数目。ERP 引入一个常量值 g，它在两条轨迹之间的元素使用编辑距离来作为处理局部时间转换的惩罚值。像 DTW 一样，ERP 也可以处理不同长度的轨迹，因为它允许在两条轨迹之间存在间隙。

移动对象轨迹相似性度量的方式有很多种，不同的场景应该使用不同的度量方式，这样才能更好的表示轨迹之间的相似程度，同时也可以简化将特定场景下问题的研究。其他的轨迹相似性度量方式还包括网络距离[14]、历史最近距离[26]、Earth Mover's Distance[27]等。其中本文的研究是在路网空间下的，因此轨迹的空间相似性度量选用的正是网络距离。

2.3 移动对象轨迹数据索引结构

2.3.1 R 树索引

R 树[28]是一种与 B 树相类似的高度平衡树，它被广泛应用到空间数据库系统当中，

用于解决传统数据库索引检索空间对象效率低效的问题。R 树是一种由最小包含矩形来近似表达空间对象的一种数据结构，这种索引是动态的，不需要定期进行重建。R 树的索引记录保存在叶节点中，索引记录包含指向空间数据对象的指针。

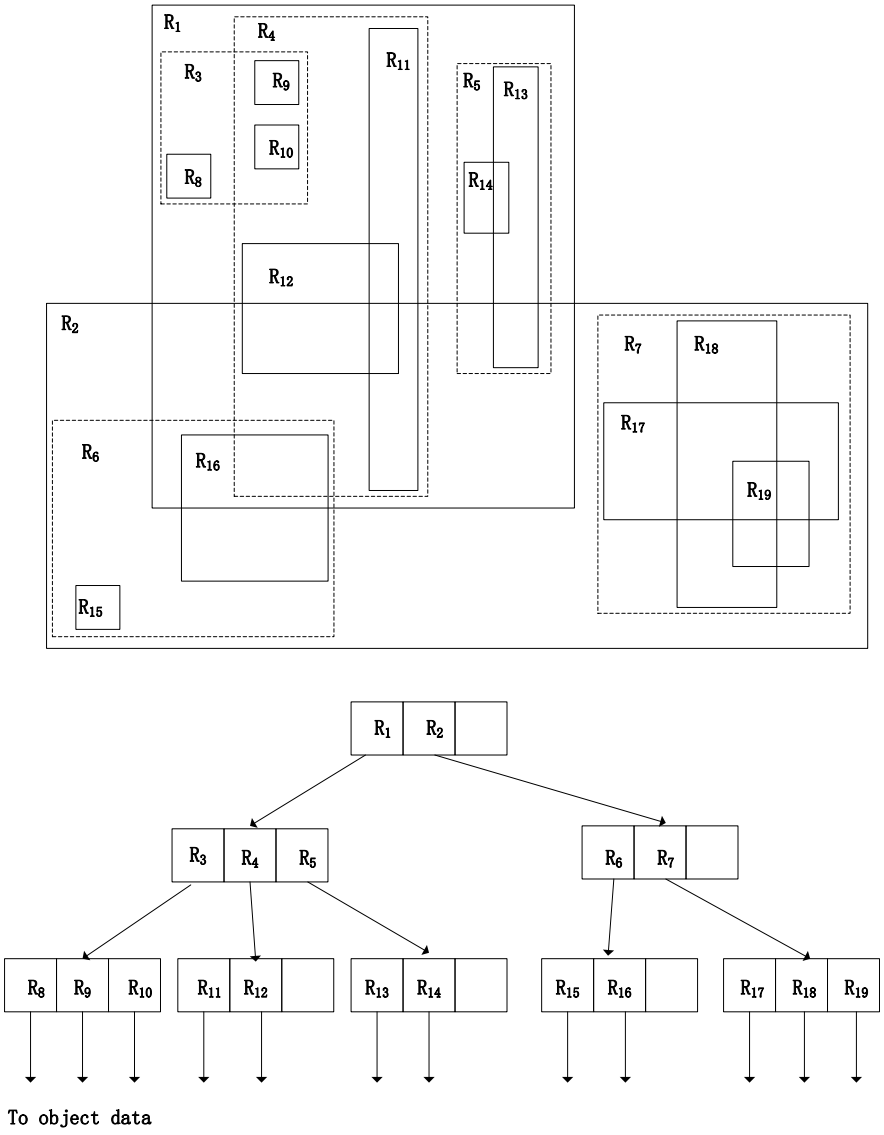


图 2-2 R 树索引结构

R 树中的叶子节点条目格式为（I，tuple-identifier）。其中，I 是一个 n 维外包矩形，表示空间对象的最小外包矩形。tuple-identifier 是一个指向空间对象的指针。R 树中的非叶子节点条目格式为（I，child-pointer）。其中，I 是其所有孩子节点的外包矩形的最小外包矩形。child-pointer 是指向其对应孩子节点的指针。

R 树的构建需要遵循一定的规则。记 M 为一个节点中的最大条目数。取 m<=M/2，约定此 m 为一个节点中的最小条目数。此时，R 树具有如下性质：

(1) 若叶子节点不是根节点，那么每个叶子节点所包含的索引记录个数介于 m 与 M 之间。

(2) 对于叶子节点中的索引记录（I，tuple-identifier），I 是其最小外包矩形。

(3) 若非叶子节点不是根节点，则其中包含的孩子节点个数介于 m 与 M 之间。

(4) 对于非叶子节点中的条目（I，child-pointer），I 表示能够包含其所有孩子节点的外包矩形的最小外包矩形。

(5) 根节点若非叶节点，则其至少有两个子节点。

(6) 所有叶子节点都在 R 树的同一层上。

如图 2-2 所示为二维空间下 R 树结构。由此图可以看出 R 树中兄弟节点对应的空间区域是可以重叠的。这样的特性可以使得 R 树比较容易进行插入和删除操作，另外也可以使得空间搜索效率降低。正是因为区域之间可以重叠，所以在搜索的时候可能存在多条路径，这就促使人们开始研究区域之间没有重叠的索引方法，例如 R^+树[29]、R*树[30]。

2.3.2 3D R 树索引

在时态数据库中存在两种基本查询类型：时间范围查询和时间戳查询。为了能够获取移动对象的时间特性，R 树可以简单的调整下，只需要将轨迹的时间维度作为 R 树的另一维度即可。因此二维的轨迹片段可以通过三维的最小外包矩形来界定，如图 2-3(a)所示。这样表示由此而带来的问题是包含轨迹片段的最小外包矩形占据了大部分的空间，而轨迹片段实际上占据了很小的一部分。因此最小包含矩形的高度重合导致了 3D R 树[31]有着很小的区分度。

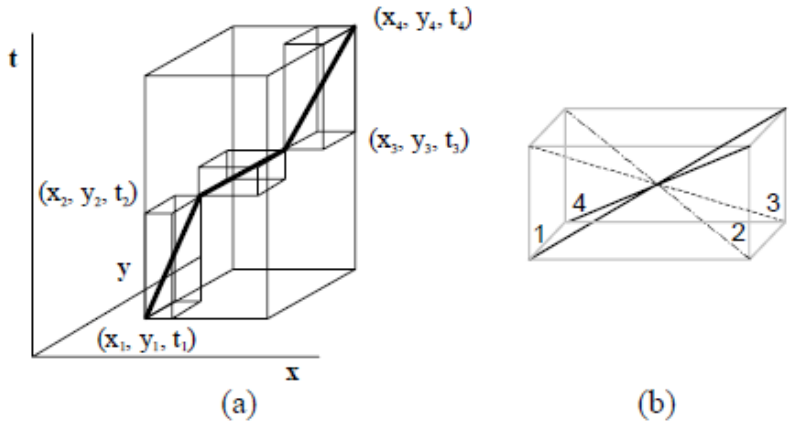


图 2-3 3D R 树索引结构

为了提高查询效率，可以对 3D R 树进行修改。修改如图 2-3(b)所示，轨迹的片段尽可以表示成四种方式。额外的信息只需通过修改条目格式为(id, MBB, orientation) 存储在叶子节点层，其中 orientation 的值域为{1,2,3,4}。假设轨迹从 0 编号到 n，叶子节点的条目的形式为(id, trajectory#, MBB, orientation)。

2.3.3 STR 树索引

R 树的插入算法是基于最小扩展策略的，而 STR[32]树不仅仅考虑空间的相近性，还

考虑到部分轨迹的保护上，它尽量将同一条轨迹的线段保存到一起，这样在插入时，就会选择其所属轨迹的上一个线段临近节点。如果这个节点还有空间，就将轨迹线段插入到这个节点。否则，这个节点将进行分裂操作。为了实现这个操作，STR 树使用算法 FindNode 来查找包含上一条线段的临近节点。插入算法引入了一个额外参数 p，当通过 FindNode 操作返回的叶子为满时，插入算法将检查 p-1 层的父节点是否为满。如图 2-4 所示，p=2，此时只需要检查非叶子节点的 1 级节点。如果它为非满，则叶子节点将被分裂。如果索引的 p-1 级父节点都为满，则采用 R 树中的选择子树算法。

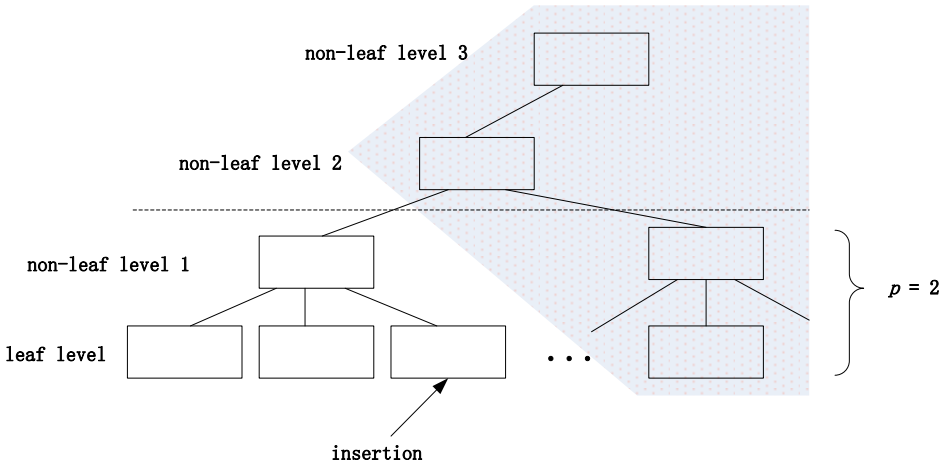


图 2-4 STR 树索引结构的插入操作

分裂策略的基本思想是将最新的线段放置到新的节点当中，分裂非叶子节点只需为新记录项创建一个新的节点。通过这种插入和分裂策略，索引结构可以同时保存轨迹的空间维度和时间维度。

2.3.4 TB 树索引

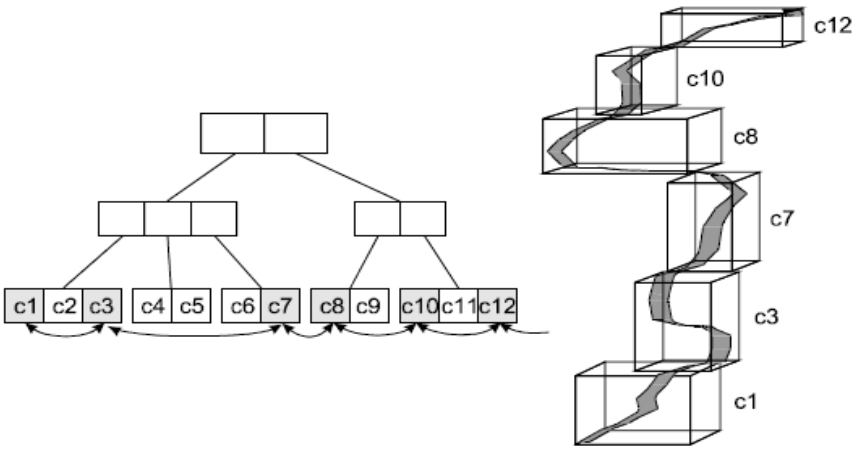


图 2-5 TB 树索引结构

TB 树[32]是一种用来严格保存移动对象轨迹的索引结构。TB 树是叶子节点的集合，每一个叶子节点包含属于同一条轨迹的线段。为了实现轨迹管理的便利，TB 树去除了 R 树中的一个重要特性：节点的重叠。这使得时空查询的效率非常高。图 2-5 展示一个 TB 树的结构。右边图中的轨迹跨越 TB 树的 6 个叶子节点，c1，c3，c7，c8，c10，c12。这些叶子节点是通过一个邻接表连接在一起的。

2.3.5 空间对象的 Voronoi 图索引

Voronoi 图[32,34]是一种基本的几何结构。Voronoi 图在地质、考古、分子化学、气象学和计算机科学等领域中都得到了深入而广泛的研究。作为空间数据库中一种重要的工具，Voronoi 图可以解决很多传统方法中不能很好解决的问题，因此有着重要的研究意义和学术价值。

(1) 欧氏空间下的 Voronoi 图

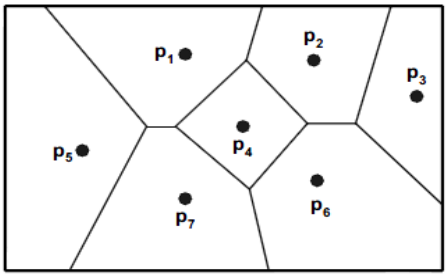


图 2-6 简单的 Voronoi 图示例

一个 Voronoi 图将一个欧氏空间划分成为 N 个不相交的多边形，在多边形内部的任何点的最近邻都是这个多边形的生成点。在欧氏空间下给定一个有限数量的点的集合，称之为生成点集，每一个生成点对应的空间下的一片点集并形成一块区域，该区域就是对应生成点的 Voronoi 多边形或者 Voronoi 格子，与所有生成点相关的 Voronoi 多边形的集合称之为关于生成点集的 Voronoi 图。每一个 Voronoi 多边形的边界称为 Voronoi 边，Voronoi 边上的点隶属于两个或者两个以上的 Voronoi 多边形，共享一条 Voronoi 边的两个 Voronoi 多边形称为相邻 Voronoi 多边形，对应的生成点称为相邻生成点。

Voronoi 多边形和 Voronoi 图可以形式化定义为：假定一个生成点集合 $P = \{p_1, p_2, \ldots, p_n\}$，区域 $VP(p_i) = \{p \mid d_E(p, p_i) \le d_E(p, p_j)$, for $i \ne j, 1 \le i, j \le n\}$表示 p_i 对应的 Voronoi 多边形，其中 $d_E(p, p_i)$表示 p 与 p_i 之间的欧氏距离。集合 $VD(P) = \{ VP(p_1), VP(p_2), \ldots, VP(p_n)\}$称之为由生成点集 P 构成的 Voronoi 图。一个二维欧氏空间的 Voronoi 图示例如图 2-6 所示。

Voronoi 图有以下重要性质，这些性质的完整数学证明参照文献[33]。

性质 1：一个生成点集 P 的 Voronoi 图 $VD(P)$是唯一的。

性质 2：距离 p_i 最近的生成点（如 p_j）做对应的 Voronoi 多边形必定与 $VP(p_i)$有公共的 Voronoi 边。

性质 3：令 n 表示生成点的数目，n_e 表示 Voronoi 图边的数目，则有 $n_e \le 3n-6$。

性质 4：每条 Voronoi 图的边只是属于两个相邻的 Voronoi 多边形，根据性质 3 可知，每一个 Voronoi 多边形的 Voronoi 边的平均数目最多为 6，亦即每个生成点有约 6 个相邻生成点。

(2) 网络 Voronoi 图

在网路 Voronoi 图[34]中，数据对象不再是随意散布在欧氏空间中，数据对象被限定在网络节点之间的边上，数据对象之间的距离定义为空间网络下两个节点的最短路径长度。空间网络可以被模型化为一个图，网络中的相交节点定义为图的节点，网络中的边定义为连接节点的边。由此，可以知道网络 Voronoi 多边形和网络 Voronoi 图可以形式化定义为：$NVP(p_i) = \{p \mid d_N(p, p_i) \le d_N(p, p_j)$ for $i \ne j, 1 \le i, j \le n\}$ 和 $NVD(P) = \{NVP(p_1), NVP(p_2), \ldots, NVP(p_n)\}$，其中 $d_N(p, p_i)$表示 p 与 p_i 之间的最短路径距离。更多详细网络 Voronoi 图的讨论见文献[34]。一个网络 Voronoi 图示例如图 2-7 所示：

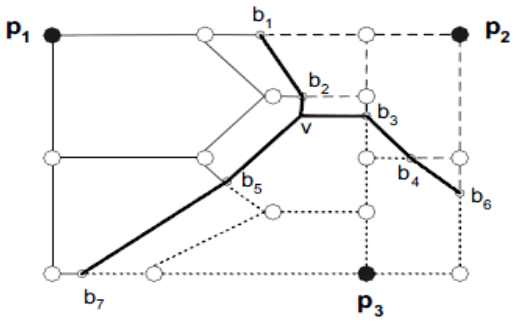


图 2-7 网络 Voronoi 图示例

2.4 本章小结

本章介绍了移动对象数据库相关概念和理论，主要包括移动对象数据库的产生背景，移动对象轨迹的多种表示方式，衡量轨迹之间相似性的几种度量标准以及移动对象轨迹的几种索引结构，为本文的后续研究奠定了基础。

第三章 基于路网的轨迹时空相似性查询

3.1 问题描述

在现实生活中，移动对象的运动主要是以轨迹数据的形式存储下来的，移动对象轨迹相似性查询广泛应用于出行推荐、好友推荐、交通规划以及数据挖掘等领域，轨迹相似性查询问题以其广泛的应用前景近年来备受研究者们的关注。然而为满足人们的需求，研究者们不断引入了一些新的查询类型。

一个简单的移动对象轨迹相似性查询的示例如图 3-1 所示：

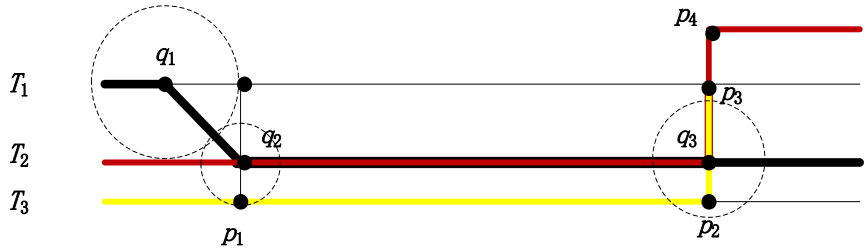


图 3-1 轨迹时空相似性查询示例

在图 3-1 中，假设在位置 q_1 处发生交通事故，肇事司机驾驶车辆逃逸，警察根据现场判断肇事司机的逃逸路线为 T_1，为更好地找到潜在的目击证人，需要从轨迹数据库中找到一条与 T_1 最为相似的轨迹。如果仅仅考虑空间因素，T_2 更符合条件。但是在考虑到现实情况，找到的轨迹需要与 T_1 在时间上也是吻合的，这样才能找到潜在的目击证人，那么此时 T_3 就符合条件，T_2 就被排除在外。有这样的一个原则：在地理空间上相近的轨迹，它们之间的相似度更高。基于每一个查询点的重要性是不同的，在查询点 q_1，q_2，q_3 处执行不同的范围查询(如图 3-1 所示)，扩张范围内的轨迹在地理上与查询轨迹相近。然后再根据时间对轨迹进行判断，最终找到与查询轨迹最为相似的轨迹。在这种情况下，轨迹的时间特性就显得尤为重要，本章立足于此类问题，在路网环境下研究移动轨迹相似性查询的时候，将轨迹的时间特性也考虑在内，为用户的多样化查询提供需求。

在路网环境下研究移动对象轨迹时空相似性查询，首先应该对路网进行建模。在这里，路网可以被模型化一张图 $G = (V, E)$，其中 V 代表一系列路网中相应路段相交的节点集合，E 是一系列 V 中节点之间的边的集合。如果有一条边连接 V 中的两个节点，那么这两个节点就是相邻的。让 $d_N(v_i, v_j)$代表两个节点 v_i 和 v_j 之间的网络距离，$d_E(v_i, v_j)$ 代表 v_i 和 v_j 之间的欧氏距离。$d_N(v_i, v_j)$通常使用 Dijkstra 算法来计算两个节点之间的最

短路径距离。很显然，$d_E(v_i, v_j) \le d_N(v_i, v_j)$，亦即两个节点对应的欧氏距离 $d_E(v_i, v_j)$是其网络距离 $d_N(v_i, v_j)$的下界。

基于路网轨迹时空相似性查询问题可以简单描述为：对于用户指定的一个查询点集合，从轨迹数据库中找到与查询点集最为“相似”的一条轨迹。这种“相似”包含两个方面：时间方面和空间方面，因此对这个问题的研究上需要定义查询与轨迹之间的相似性度量方法。由于轨迹数据的海量性，根据相似性度量方法计算查询与所有轨迹之间的相似性值不切合实际，同时计算量巨大，所以一种高效的查询处理算法也是必须的。下面就针对此问题进行形式化定义，并提出相应的查询处理解决方法。

3.2 轨迹时空相似性度量方法

为了更好的对移动对象轨迹时空相似性查询问题进行描述，下面研究中用到的相关定义进行介绍。

定义 3-1 (轨迹） 路网中的一条轨迹定义成一系列点的序列$< p_1, p_2, \ldots, p_i, \ldots, p_n >$，其中 $1 \le i \le n$, $p_i = (v_i, t_i)$，v_i 是 G 中的节点, t_i 是移动对象经过 v_i 的时刻。

为了描述方便，本文使用 $p_i(v)$来表示 v_i，使用 $p_i(t)$表示 t_i。假定 $q_i = (q_i(v), q_i(t))$为一个查询点，$q_i(v)$表示路网中 q_i 相应的节点，$q_i(t)$表示由用户指定的在 $q_i(v)$上的时刻点。

定义 3-2 (最小匹配点) 给定一条轨迹 $T = < (v_1, t_1), (v_2, t_2), \ldots, (v_n, t_n) >$和一个查询点 $q_i = (q_i(v), q_i(t))$，若称 $p_k = (v_k, t_k)$是轨迹 T 中关于 q_i 的最小匹配点，当且仅当对于任意的$(v_j, t_j) \ne (v_k, t_k)$ $(1 \le j, k \le n)$，有 $d_N(v_j, q_i(v)) \ge d_N(v_k, q_k(v))$。

定义 3-3 (匹配对) 给定一条轨迹 $T = < (v_1, t_1), (v_2, t_2), \ldots, (v_n, t_n) >$和一个查询点 $q_i = (q_i(v), q_i(t))$，如果 $p_k = (v_k, t_k)$是 q_i 关于 T 的最小匹配点，那么定义为$< p_k, q_i >$为 q_i 关于轨迹 T 的匹配对。

移动对象轨迹之间的相似程度需要有一种方法来进行量化，在本章中因需要考虑轨迹的时间特性和空间特性，所以在判断轨迹之间相似性的时候应该从这两个方面考虑。基于上面的定义，下面将分别介绍轨迹的时间相似性度量和空间相似性度量以及时空相似性度量。

3.2.1 空间相似性度量方法

定义 3-4 (轨迹与查询点之间的空间距离) 给定一条轨迹 $T = < (v_1, t_1), (v_2, t_2), \ldots, (v_n, t_n) >$和一个查询点 $q_i = (q_i(v), q_i(t))$，轨迹与查询点之间的空间距离定义如下：

$$dist_s(T, q_i) = d_N(v_i, q_i(v)) \qquad (3\text{-}1)$$

其中(v_i, t_i)是 q_i 关于 T 的匹配对。

定义 3-5 (轨迹与查询点集之间的空间距离) 给定一条轨迹 $T = < (v_1, t_1), (v_2, t_2), \ldots, (v_n, t_n) >$和一个查询点集合 $Q = \{ q_1, q_2, \ldots, q_m \}$，轨迹 T 与查询点集 Q 之间的空间距离的定义如下：

$$dist_s(T, Q) = \sum_{i=1}^{m} dist_s(T, q_i) \quad (3\text{-}2)$$

3.2.2 时间相似性度量方法

定义 3-6 (轨迹与查询点之间的时间距离) 给定一条轨迹 T = < (v_1 , t_1), (v_2 , t_2),. . . , (v_n , t_n) >和一个查询点 $q_i = (q_i(v), q_i(t))$，轨迹与查询点之间的时间距离定义如下：

$$dist_t(T, q_i) = | p_k(t) - q_i(t) | \quad (3\text{-}3)$$

其中< p_k, q_i >是 q_i 关于 T 的匹配对。

定义 3-7 (轨迹与查询点集之间的时间距离) 给定一条轨迹 T = < (v_1, t_1), (v_2 , t_2),. . . , (v_n , t_n) >和一个查询点集合 $Q = \{ q_1, q_2, \ldots, q_m \}$，轨迹与查询点集之间的时间距离定义如下：

$$dist_t(T, Q) = \sum_{i=1}^{m} dist_t(T, q_i) \quad (3\text{-}4)$$

3.2.3 时空相似性度量方法

基于上面轨迹与查询点集之间的空间距离和时间距离的定义，可以使用下面两个公式来分别定义空间相似性函数 S_s 和时间相似性函数 S_t。

$$S_s(T, Q) = \frac{1}{1 + e^{-dist_s(T,Q)}} \quad (3\text{-}5)$$

$$S_t(T, Q) = \frac{1}{m \times max\{| p_k(t) - q_i(t) |\}} \times dist_t(T, Q) \quad (3\text{-}6)$$

定义 3-8 (轨迹与查询点集的时空相似性) T 与 Q 之间的时空相似性函数定义如下：

$$S_{st}(T, Q) = S_s(T, Q) \times S_t(T, Q) \quad (3\text{-}7)$$

从上面的定义，可以得知相似性函数 S_{st} 的取值范围在 0 和 1 之间，并且 S_{st} 的值越小说明 T 和 Q 之间的时空相似性越高。在本章中，轨迹的时间相似性和空间相似性是放在同等重要的地位的，因此可以使用 S_s (T, Q)与 S_t (T, Q)乘积来表示轨迹之间的时空相似性的。基于以上定义，下面将对本章要研究的移动对象轨迹时空相似性问题进行形式化的定义。

定义 3-9（时空轨迹相似性轨迹查询）给定一个轨迹数据库 D 和一个查询点集 Q，轨迹时空相似性查询就是查找一条轨迹 $T \in D$，使得$\forall T' \in D - \{T\}$, $S_{st}(T, Q) \leq S_{st}(T', Q)$。

3.3 轨迹时空相似性查询方法

3.3.1 基本思想

在查询处理过程中，将选择网络 Voronoi 图来作为数据结构。网络 Voronoi 图在处理范围查询问题上有着高效的性能。在移动对象轨迹时空相似性查询问题当中，将采用两阶段处理策略：空间过滤阶段和时间提纯阶段，最终根据计算得到的时空相似性值 S_{st}，找到与查询最为相似的一条轨迹。轨迹相似性查询中的多个点的查询可以看成多个单点同时进行范围查询的一个扩展，因此可以将网络 Voronoi 图在处理范围查询的方法引入到空间过滤阶段。

移动对象轨迹时空相似性查询算法的基本思想是：首先使用网络 Voronoi 图将巨大的路网空间划分成为一个个小的网络 Voronoi 区域（在这里将图中所有的节点作为网络 Voronoi 图的生成点集），如图 3-2 所示。注意在图 3-2 中只有在 $NVP(P_1)$内的节点和边绘出，通过这种方式，就可以建立起所有节点和 *NVP* 之间一一映射的关系。对于单一的 NVP 而言，其边界点之间的网络距离必须预计算，并存储在一个预计算表当中。基于轨迹数据库的海量性，从轨迹数据库中的每一条轨迹都计算 S_{st} 值并找出最小的值其耗费的代价是非常巨大的。因为它必须将每一条数据轨迹从硬盘中加载到内存中来以便能够计算出精确的 S_{st} 值，这会引入太多的 I/O 并带来巨大的计算量。

存在这样一个原则：在空间上相近的轨迹，它们之间有着更好的轨迹时空相似性。因此在空间过滤阶段，可以对于每一个查询点 q，以 q 为中心逐步进行网络 Voronoi 范围扩张，直到用户指定的边界，将扩张范围内的轨迹加入到候选轨迹中去。具体步骤为，对于每一个查询点，定位 q 所在的 *NVP*，然后从 q 开始进行范围扩张，然后找出在特定区域内 q 的相邻节点，经过这些节点的轨迹就是需要处理的轨迹。这样就避免了与轨迹数据集中的所有轨迹进行时空相似性值的计算，因而减少了要处理的轨迹的数量，减少了计算量，提高了查询速度。在时间提纯阶段，对于候选轨迹集合中的每一条轨迹计算时间相似性值，并最终计算出时空相似性值，找到时空相似性值 S_{st} 最小的那条轨迹并作为返回结果。

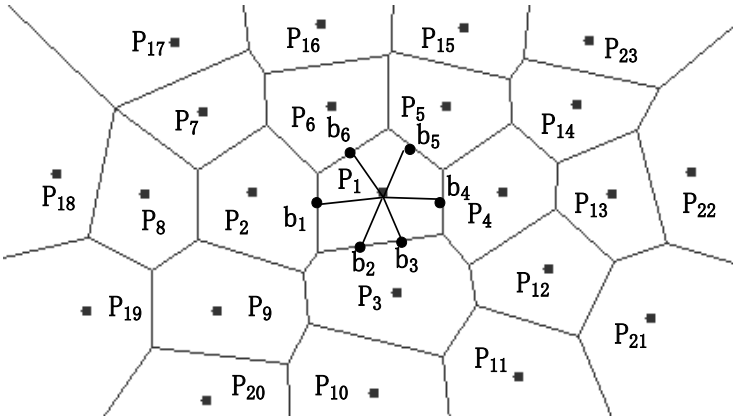


图 3-2 使用网络 Voronoi 图划分之后的地理空间

3.3.2 存储模式

为了进一步加快查询处理过程，引入一个与文献[34]类似的存储结构，但是描述 NVP 邻接关系的部分并不相同。图 3-3 直观展示了算法中使用的存储模式，它主要包括 6 个部分：NVP 部分（NVPs Component）和边界点部分（Border Points Component），这两个部分是存在于内存中的数组。邻接部分（Adjacency Component）、预计算部分（Pre-computed Component）、轨迹倒排表部分（Trajectory Inverted List）和轨迹部分（Trajectory Component）占用空间较大，存放在磁盘上。为了快速定位查询点所在的 NVP，这里需要使用 R 树来对 NVP 进行一次预处理。边界点部分为每一个边界点记录下了一些在查询处理过程中生成的关键信息。对于轨迹倒排表部分，为每一个 NVP 建立了一张倒排表，一系列经过 NVP 的生成点的轨迹 ID。预计算部分是一个距离集合，存储所有 NVP 的边界点之间的网络距离以及边界点到 NVP 生成点的网络距离。由于一个边界点属于两个相邻的 NVP，因此边界点与两个 NVP 的其他边界点都需要进行预计算。

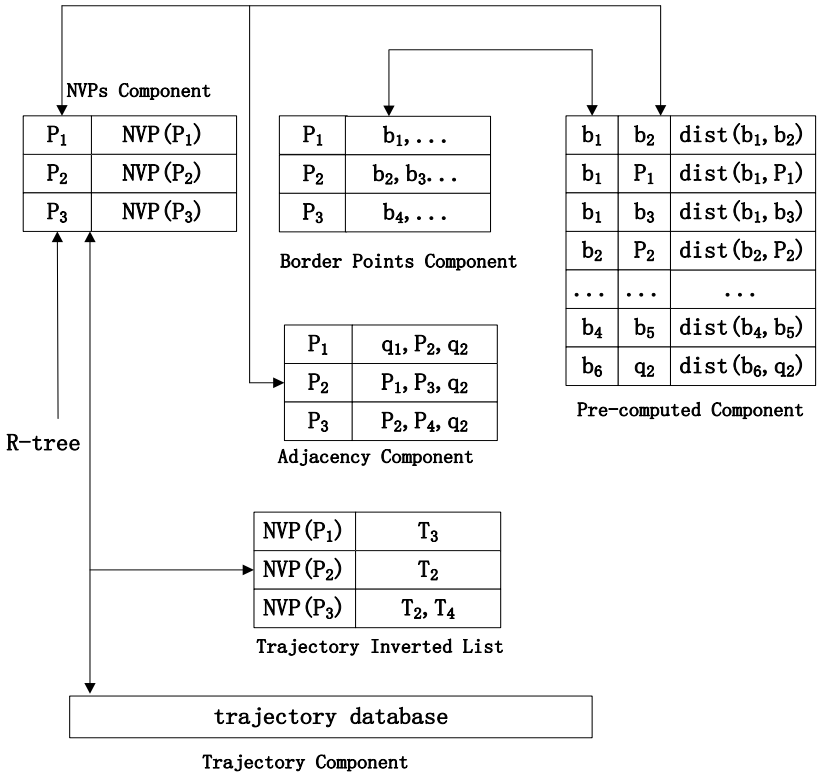


图 3-3 网络 Voronoi 图存储模式

3.3.3 算法描述

基于上面的基本思想和存储模式，本章提出的查询处理算法如下：

算法 ***STQuery*** (*D*, *Q*, *Φ*)
输入：轨迹数据集 *D*，查询点集 *Q*，距离范围阈值*Φ*

输出：一条时空相似性轨迹

```
1    construct a queue Qe_i for every query point q_i
2    for each q_i in the query point set Q do
3        form the voronoi polygon NVP(q_i) according to q_i
4            while each neighbor P_j of q_i
5                if (P_j existed in Qe_i) then
6                    neglect it
7                else
8                    add P_j and d_N(q_i,P_j) to Qe_i in the descending order of d_N(q_i,P_j)
9            until d_N(q_i,P_j) >Φ
10       add those trajectories crossing through NVP(q_i)
             and all NVP(P_j) into candidate set C
11   end for
12   for each trajectory T in C do
13       compute S_st (T, Q)
14   end for
15   return the trajectory with smallest S_st value
```

STQuery 算法首先定位包含查询点 q 所在的 *NVP*，同时为每一个查询点 q 构建一个优先队列来处理在逐步扩张过程中入队的节点。在过滤阶段，记录下处于 Φ 范围内的节点，并将经过这些节点的轨迹加入到候选集合 C 中，以便在提纯阶段对 C 进行计算。下面通过一个具体的例子来形象描述一下上面描述的算法。

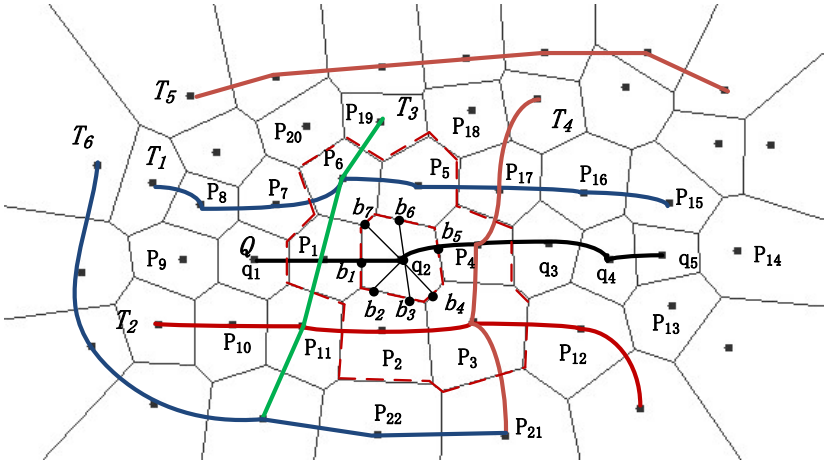


图 3-4 算法示例

如图 3-4 所示，T_1, T_2, T_3, T_4, T_5 和 T_6 表示轨迹，$Q = \{q_1, q_2, q_3, q_4, q_5\}$ 是一个由用户指定的查询点集（注意此时 Q 的时间特性并没有在图 3-4 标示）。以 q_2 为例，根据算法 1，首先定位 *NVP*(q_2)，然后从 NVP(q_2)开始进行扩张，那就是说，将 q_2 的相邻节点 P_1, P_2,

P_3, P_4, P_5, P_6 加入到扩张范围中，然后使用边界点部分和预计算部分来计算 q_2 到这些新插入节点的距离。通过比较，当前的 $d_N < \Phi$，然后继续将扩张范围内节点的相邻节点 q_1, q_3, P_7, P_{11}, P_{12}, P_{17}, P_{18}, P_{19}, P_{20}, P_{21}, P_{22} 加入到 Qe_i 中。重复这个过程，最终的查找区域如图 3-4 红色虚线部分所示。对于其他的查询位置来说其过程也是一样的。所有的查询位置扩张完毕之后，T_1, T_2, T_3 和 T_4 被加入到候选集合 C 中去。最终，为 T_1, T_2, T_3 和 T_4 计算出准确的 S_{st} 值。其中 T_1 拥有最小的 S_{st} 值，并作为最终的返回结果。

3.4 实验与分析

3.4.1 实验环境与设计

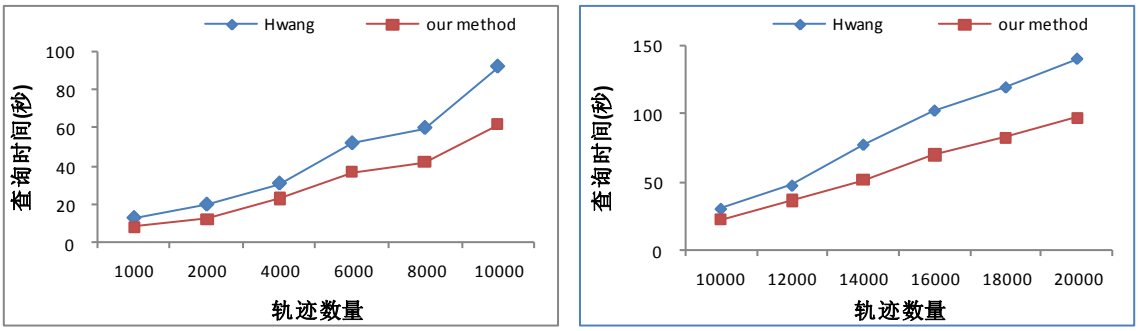
在这一部分，通过实验来验证本章所提出的方法的性能。本实验中使用的路网数据集是 Oldenburg Road Network 和 California Road Network[35]，分别包含 6104 和 21047 个节点。实验环境是在 Windows 平台下配置如下 Intel(R) Core(TM) Duo CPU E7400 处理器和 2GB 内存。所使用的轨迹数据集是由 Brinkhoff[36]生成器基于上面提到的真实路网来生成的。

另外在进行实验的时候，将选择 Hwang 的方法来作为一个基准方法来与本章的方法进行比较。原因是：Hwang 的方法同时考虑空间相似性和时间相似性。实验效果所采用的主要性能指标是查询时间和处理的轨迹数目。选择处理的轨迹数目这个参数作为标准原因有两个：一是它可以粗略表示出候选集合的数量；二是它可以在一定程度上反映出 I/O 的大小。其他主要的参数设置见表 3-1。

表 3-1 主要参数

	Oldenburg	California
轨迹数量	1000-10000(默认 6000)	10000-20000(默认 16000)
查询点数量	2-12(默认 8)	2-12(默认 8)

3.4.2 实验结果与分析

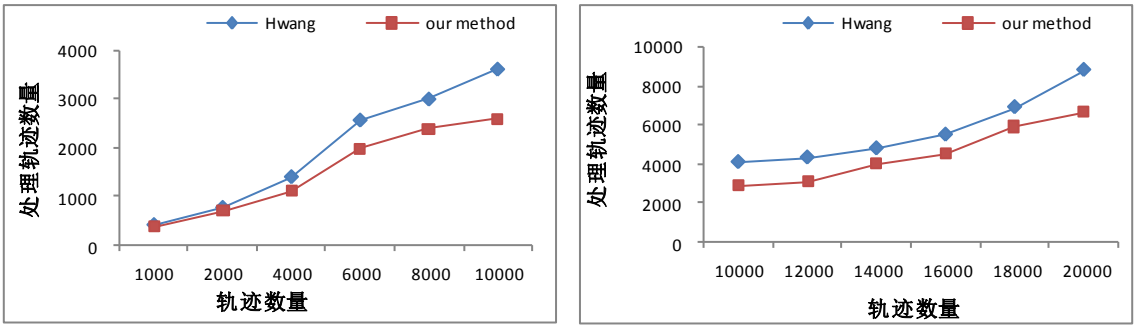


(a) Oldenburg (b) California

图 3-5 查询时间 vs. 轨迹数量

针对上一小节的实验设置，对本章提出的算法进行实验分析，并与其它方法做出性能的比较。实验结果及分析如下。

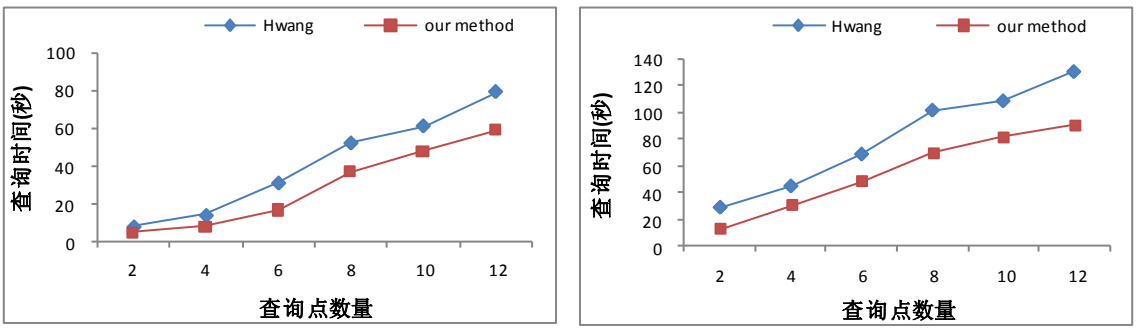
图 3-5(a)和(b)展示了两种方法随着轨迹数量的不同在查询时间上性能上的对比。由图 3-5(a)和(b)显示，在 Oldenburg 和 California 路网环境下，两种方法 CPU 时间会随着路网环境中存在的轨迹数量增多而增加。这是因为随着轨迹数量的增加，在一定范围内的轨迹数量也会增多，这会导致要处理的轨迹的数目的增加。同样地，可以看出本章提出的方法都是要优于 Hwang 的方法的。



(a) Oldenburg (b) California

图 3-6 处理轨迹数量 vs. 轨迹数量

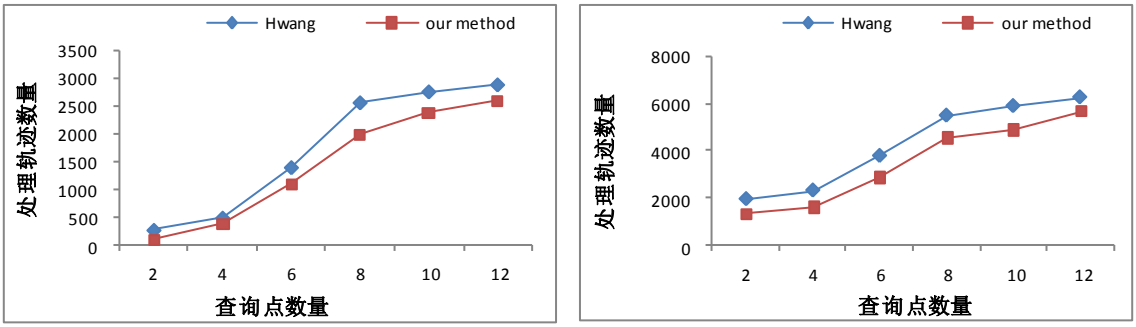
图 3-6(a)和(b)展示了两种方法在两个数据集上随着轨迹数量的不同在处理轨迹数量性能上的对比。由图 3-6(a)和(b)显示,待处理的轨迹数目随着轨迹数目的增加而增加。随着在同一区域内轨迹数目的增加，在一定区域内的轨迹数据的密度也会增加，因此在一定区域内要处理的轨迹数目必然会增加。同时可以看出，本章提出的过滤策略是要比 Hwang 的方法效果是好一些的。



(a) Oldenburg (b) California

图 3-7 查询时间 vs. 查询点数量

图 3-7(a)和(b)展示了两种方法随着查询点数量的不同在查询时间性能上的对比。由图 3-7(a)和(b)显示，查询响应时间随着查询点数量的增加而增加。这是因为随着查询点数量的增加，查询会导致需要更多的查询源来进行扩张，因此时间会在这个地方有所消耗。



(a) Oldenburg (b) California

图 3-8 处理轨迹数量 vs. 查询点数量

图 3-8(a)和(b)展示了两种方法查询点数量的不同在处理轨迹数量性能上的对比。由图 3-8(a)和(b)显示,待处理的轨迹数目随着查询点数量的增加而基本线性增加的。随着查询点数量的增加，越来越多的轨迹加入到候选轨迹集合中来，这样会增大计算量。另外可以看出，本章所采用的方法，处理的轨迹数量是要比 Hwang 的要少的，可以说明本章的算法过滤阶段是要优于 Hwang 的。

3.5 本章小结

本章首先讨论了路网环境下的移动对象轨迹时空相似性查询的问题背景，给出了轨迹时空相似性查询的几个相关定义，包括空间网络的定义、轨迹的定义等，并在此基础之上提出轨迹之间的相似性度量方法，同时对路网环境下移动对象的轨迹时空相似性查询问题进行了形式化的定义。在明确问题定义后，简要介绍了该问题解决方法的基本思想，并提出了一种解决轨迹时空相似性查询问题的高效算法，对部分算法细节进行了重点阐述。最后是对提出算法的实验验证，包括实验环境、实验数据集的介绍等。

第四章 基于路网的轨迹空间-文本相似性查询

4.1 问题描述

随着移动设备和 GPS 的不断发展，人们可以很方便地记录下自己的地理位置，并将自己的位置分享到像 Bikely，GPS-Way-points，Share-My-Routes，Microsoft GeoLife 这样的网站上，同时越来越多的社交网站像 Twitter，Four-square 和 Facebook，开始支持分享轨迹/位置的应用。其中一个很重要的应用就是轨迹查找和推荐，在这个应用中旨在从历史轨迹中找到与一系列查询位置相近的轨迹。

之前研究者的工作重点，轨迹相似性查询只考虑空间特性，这就意味着空间相似性是在查询中唯一的判别标准。然而在很多应用中，特别是在一些推荐系统中，基于用户的不同偏好，空间距离本身并不能足够表示轨迹与查询位置之间的关系。例如，系统可能推荐一条包含多个收费路段的轨迹，但是这对一些对预算比较在意的人来说并不是一个好的推荐。尽管推荐的路线与查询位置很相近，出行者依然不满意所推荐的路线，因为他们的偏好并没有得到满足。

轨迹空间-文本相似性查询在一定范围区域内查找一条轨迹，这条轨迹包含所有他们要去的位置，同时包含一个关键词集合。下面通过一个示例来说明，如图 4-1 所示：

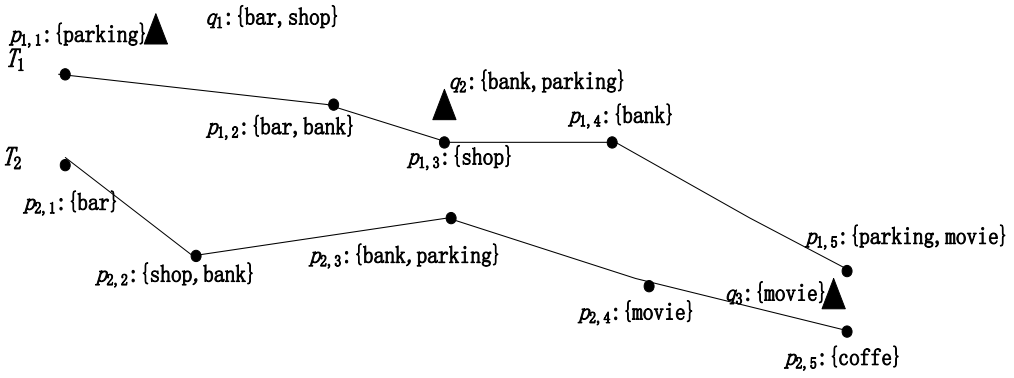


图 4-1 轨迹空间-文本相似性查询示例

假设一个出行者在一个陌生的城市想去 q_1，q_2，q_3 三个地方，并分别在这三个位置做某些活动。图 4-1 将展示这样的场景，其中两条轨迹 T_1，T_2 和一个用户查询 Q，其中 Q 包括一个的查询位置集合{q_1，q_2，q_3}，并在这三个位置上分别带有关键字集合{bar，shop}，{bank，parking}，{movie}。如果只考虑轨迹的空间特性，T_1 可以作为与 Q 最为相似的轨迹，这是因为 $p_{1,1}$，$p_{1,3}$，$p_{1,5}$ 分别是查询位置 q_1，q_2，q_3 的最近邻节点。但是从另一角度来看，即使 T_2 与查询位置在地理上较远，但是 T_2 在查询位置可以满足出行者的偏好，因此 T_2 可以为用户作为更好的参考。T_2 在每一个查询位置上都有匹配的关

键字（其中 q_1 匹配 $p_{2,1}$ 和 $p_{2,2}$，q_2 匹配 $p_{2,3}$，q_3 匹配 $p_{2,4}$）。

在实际的应用场景中，可能历史轨迹数据并不能严格经过用户指定的地点，因而返回的结果通常是 k 条最“接近”用户指定地点集合的轨迹。为了更好的满足用户的要求，查询结果应该同时满足以下两个条件。由于考虑空间特性和文本特性，因此应该从这两个方面来解决问题。空间特性方面，返回的轨迹应该尽可能的经过用户指定的位置或者在地理空间上尽量与这些位置相近。文本特性方面，用户在每一个地理位置指定的关键字都包含在返回的轨迹上对应的点的关键字集合中。

有的时候，轨迹上包含查询中所有的关键字同时又与查询“相近”，有时，轨迹上包含查询中所有的关键字但与查询相隔“较远”，因此定义一个相似性度量规则来将空间特性和文本特性结合起来是一项很有挑战性的工作。为解决这个问题，度量规则应满足以下要求：

(1) 如果有两条轨迹在地理空间上完全匹配，但是文本这一特性上，一条轨迹 A 比另一条轨迹 B 拥有更多的查询活动数量，则轨迹 A 对于查询 Q 来说比其他轨迹具有更好的相似性。

(2) 如果两条轨迹在文本这一特性上完全匹配，但是在空间上并不“相近”，那么在地理空间上与查询点集更为相近的那条轨迹与查询具有更好的相似性。

本章针对目前轨迹相似性存在的问题，除了考虑轨迹的空间特性之外，还将用户偏好考虑在内，给出了路网环境下的轨迹之间的相似性度量方法，并设计出了一种新的查询处理算法，最后通过实验验证算法的性能。

4.2 相关定义

本章对移动对象轨迹空间-文本相似性查询的研究是基于路网环境下的，因而首先对路网进行建模。路网仍然被模型化一张图 $G = (V, E)$，其中 V 代表一系列路网中相应路段相交的节点集合，E 是一系列 V 中节点之间的边的集合，与第三章的路网定义相同，这里就不再过多介绍。

定义 4-1 （轨迹） 轨迹数据库 D 中的每一条轨迹 $T \in D$ 定义为一系列附有相关关键字的位置集合，即 $T = p_1, p_2, \ldots, p_i, \ldots, p_n$ ，其中 $p_i = <loc_i, keys_i>$, loc_i 表示 p_i 的地理空间位置，$keys_i$ 表示在地理位置 loc_i 所从事的活动的关键字集合。

从本质上来讲，此处轨迹表示描述移动对象在多个位置的行为的历史记录。为了描述方便，采用形式 $p=<loc, keys>$，$p.loc$ 表示的 p 的空间位置，$p.keys$ 描述在位置 loc 的关键字集合。注意，在某些点的关键字集合可能为空。

定义 4-2 （子轨迹） 将轨迹 T 的子轨迹定义为从轨迹 T 的位置 s 到 e，其中 $s,e \in [1,n]$, $s \le e$，子轨迹表示为 $T.p_s^e$。

定义 4-3 （轨迹包含关系） 给定两条子轨迹 $T.p_{s1}^{e1}$ 与 $T.p_{s2}^{e2}$，如果 $s1 \leqslant s2$ 同时 $e1 \geqslant e2$，则称 $T.p_{s1}^{e1}$ 包含 $T.p_{s2}^{e2}$。

定义 4-4 （点匹配与点匹配距离） 给定一个查询点 $q = <loc,keys>$和一条轨迹 T，查询点 q 对于轨迹 T 的点匹配，记为 $T.pmatch(q)$，代表的是 T 的子轨迹 $T.p_s^e$，子轨迹中

所有的点集合满足：$q.keys \subseteq \cup_{i=s}^{e} p_i.keys$，其中点匹配距离为

$$Dist_{pm}(q, T.pmatch(q))= \min(d_N(q, T.p_s), d_N(q, T.p_e)) + \sum_{i=s}^{e-1} d_N(T.p_i , T.p_{i+1}) \quad (4\text{-}1)$$

定义 4-5（点集匹配与点集匹配距离） 给定一个查询点集 $Q=<q_1, q_2, \ldots, q_m>$，$q_i=<loc_i,keys_i>$和一条轨迹 T，如果对于查询点集中的每一个查询点 $q_i \in Q$，都存在 q_i 到 T 的点匹配，那么所有查询点 q_i 的点匹配的集合就是 Q 与 T 是点集匹配，记为 $T.psmatch(Q)$。其中点集匹配距离为

$$Dist_{psm}(Q, T.psmatch(Q)) = \sum_{q_i \in Q} Dist_{pm}(q_i, T.pmatch(q_i)) \quad (4\text{-}2)$$

定义 4-6 （最小点匹配与最小点匹配距离） 给定一个查询点 $q=<loc,keys>$和一条轨迹 T，若称一个点匹配被称为最小点匹配，则对于其他任何 q 关于 T 的点匹配 $T.pmatch(q)'$，都有 $Dist_{pm}(q, T.pmatch(q)) \leqslant Dist_{pm}(q, T.pmatch(q)')$，最小点匹配距离为

$$Dist_{mpm}(q, T) = Dist_{pm}(q, T.psmatch(q)) \quad (4\text{-}3)$$

定义 4-7 （最小点集匹配与最小点集匹配距离） 给定一个查询点集 $Q = <q_1, q_2, \ldots, q_m >$，$q_i = <loc_i, keys_i>$和一条轨迹 T，如果一个点集匹配被称为最小点集匹配，则对于其他任何 Q 关于 T 的点集匹配 $T. psmatch(Q)'$，都有 $Dist_{psm}(Q, T.psmatch(Q)) \leqslant Dist_{psm}(Q, T.psmatch(Q)')$。那么最小点集匹配距离为

$$Dist_{mpsm}(Q, T) = Dist_{psm}(Q, T.psmatch(Q)) \quad (4\text{-}4)$$

基于上面的定义可以得知，Q 与 T 的最小点集匹配是由 Q 中的每一个查询点 $q_i \in Q$ 的最小点匹配点组成的，那么最小点集匹配距离是各个查询点 q_i 的最小点匹配距离的加和，即

$$Dist_{mpsm}(Q, T)= \sum_{q \in Q} Dist_{mpm}(q, T) \quad (4\text{-}5)$$

下面对本章的移动对象轨迹空间-文本相似性查询问题进行定义：给定一个轨迹数据库 D，一个查询 Q，一个正整数 k，移动对象轨迹空间-文本相似性查询从轨迹数据库 D 中返回 k 条与 Q 拥有最小点集匹配距离的轨迹。

4.3 轨迹空间-文本相似性查询方法

在实际情况下，地理空间上距离相近的轨迹之间往往具有更好的相似性。因此在解决轨迹空间-文本相似性查询问题的时候可以这样考虑：以用户指定的每一个查询位置

点为中心，逐步进行范围扩张，如图 4-2 所示。在扩张的过程中判断扩张范围内的轨迹是否与查询点是点匹配的。若是，并计算出最小点匹配距离，并存入一个匹配队列 Q 当中，这个序列是以最小点匹配距离进行升序排列的。这样每一个查询点对应一个有序队列，即有 m 个这样的队列。这些队列用于以后计算最小点集匹配距离，详细内容见后面介绍。

扩张过程中需要完成三个阶段的工作：最小点匹配计算阶段，候选轨迹生成阶段和候选轨迹验证阶段。最小点匹配阶段的工作是计算每一个查询点与扩张过程中碰到的轨迹的最小点匹配距离，详细介绍见 4.3.1 部分。候选轨迹生成阶段根据从上一阶段计算出的最小点匹配距离，生成候选轨迹集合，详细介绍见 4.3.2 部分。候选轨迹验证阶段的工作是将候选轨迹进行验证，根据最小点集匹配距离的大小对轨迹进行排序，最终得到 top-k 轨迹结果，详细介绍见 4.3.3 部分。

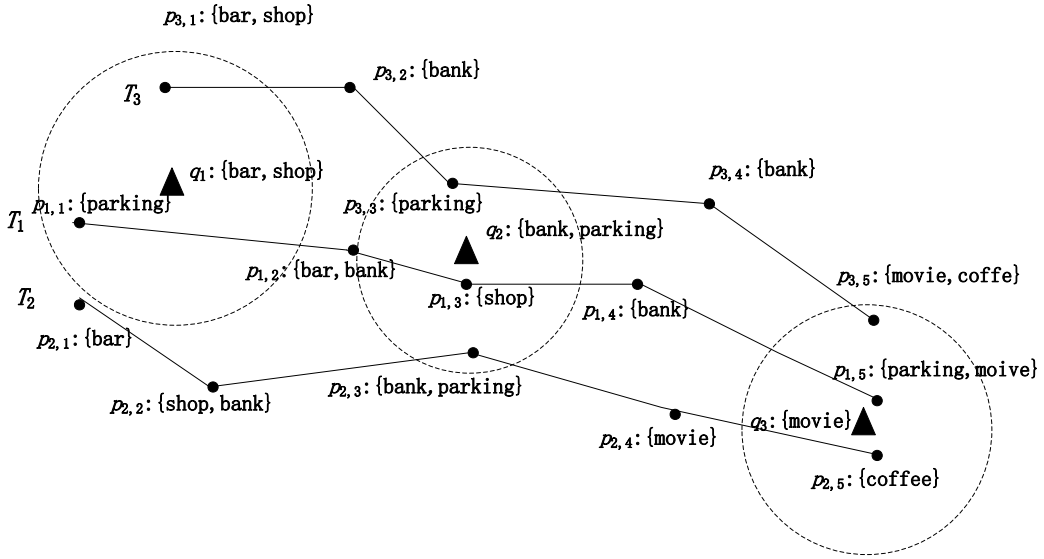


图 4-2 逐步范围扩张过程

4.3.1 最小点匹配计算阶段

给定一条轨迹 T 和一个查询点 q = <*loc*, *keys*>，计算最小点匹配距离首先需要找到最小点匹配，而寻找查询点 q 的最小点匹配最先想到的方法是：判断 q 与 T 的所有子轨迹是否是点匹配的，然后找出最小点匹配。如果采用这种方法，时间复杂度是 $O(|T|^2)$。因此这里采用分而治之和动态编程的方法将这个问题划分为多个子问题，每一个子问题都从轨迹 T 的一个位置上开始查找最小点匹配。在 T 上的每一个位置都需要检查从这个位置开始往后的子轨迹是否与查询点 q 是最小点匹配的。这里最为关键的一点是：在从当前位置开始查找最小点匹配这个子问题时会重用从先前位置开始查找最小点匹配子问题的计算。在完成所有的子问题后，如果存在最小点匹配的，将会计算出最小点匹配距离。

基于定义 4-4，可以得到以下推论：

推论 1：若子轨迹 $T.p_s^e$ 与查询 q 是点匹配的，任何包含子轨迹的子轨迹 $T.p_s^e$ 都与 q

是点匹配的。若子轨迹 $T.p_s^e$ 与查询 q 不是点匹配的，则 $T.p_s^e$ 的任何子轨迹与 q 都不是点匹配的。

推论 2:如果 $T.p_s^e$ 是查询 q 的最小点匹配，$T.p_{s1}^{e1}$ 也是 q 的点匹配且满足 $s1 \le s$ 和 $e1 \ge e$，则 $Dist_{pm}(q, T.p_s^e) \le Dist_{pm}(q, T.p_{s1}^{e1})$。

推论 3：如果子轨迹 T_2 包含子轨迹 T_1，则查询点 q 与 T_1 的点匹配距离小于 q 与 T_2 的点匹配距离。

推论 4：假设子轨迹 $T.p_s^e$ 与查询点 q 是点匹配的，则 $Dist_{pm}(q, T.p_s^e) \ge \max_{i\in[s,e]} d_N(q,T.p_i)$。

基于上面的分析，最小点匹配算法描述如下：

算法PMatch(query q, trajectory T , distance ξ)

输入：查询点q，轨迹T，已经查找到的第k大的点匹配距离ξ

输出：最小点匹配距离

```
1  minDist←∞;  tstart←∞;  tend←∞
2  C ←an array of |q.keys| elements of 0
3  for each key k in p1.keys do C[k] ←C[k]+1
4  end ←1
5  start ←1
6  while end ≤ n do
7     (bool, minDist, tstart, tend)←IsMatch (q, C, start, end, minDist)
8     if bool then
9        for each key k in pstart.keys do C[k] ←C[k]−1
10       start ←start+1; continue
11    end ←end +1
12    if Dist(q, pend ) > ξ then
13       start ←end +1
14       C ←0
15       continue
16    for each key k in pend.keys do
17       C[k] ←C[k]+1
18    if min(dN (q,pstart), dN(q, pend )) + ∑_{j=start}^{end-1} dN (pj ,pj+1) > ξ then
19       for each key k in pstart.keys do C[k] ←C[k]−1
20       start ←start+1; continue
21    (bool, minDist, tstart, tend)←IsMatch (q, C, minDist, start, end)
22    if bool then
23       for each key k in pstart.keys do C[k] ←C[k]−1
24       start ←start+1
25    if end = n and not( ∀ k ∈q.keys, C[k] > 0) then
```

26 **break**
27 **end while**
28 **return** (minDist, tstart, tend)

算法 IsMatch(*q, C*, tstart, tend, minDist)
输入：查询点*q*, 计数器数组*C*, 起始位置tstart, 终止位置tend, 匹配距离minDist
输出：*bool*, minDist, tstart , tend
1 **if** ∀ $k \in q.keys\ C[k] > 0$ **then**
2 $md = \min(d_N(q,p_{start}), d_N(q,p_{tend})) + \sum_{j=tstart}^{tend-1} d_N(p_j,p_{j+1})$
3 **if** minDist > *md* **then** minDist←*md*
4 return (*true*, minDist, tstart, tend)
5 return (false)

在算法 PMatch 中，ξ 表示的是已经查找到的第 k 大的点匹配距离，minDist 用来记录当前最小点匹配距离。tstart 和 tend 分别用来标识相应最小点匹配的起始位置和终止位置（行 1）。*C* 用来记录子轨迹中查询关键字出现的次数（行 2）。变量 start 代表子轨迹的起始位置，end 用来表示子轨迹的终止位置。算法首先用位置 p_1 处的查询关键字对数组 *C* 进行初始化（行 3）。对于每一个位置 p_{end}，算法 PMatch 从位置 p_{start} 到 p_{end} 查找是否点匹配（行 7-24）。算法 PMatch 扫描到 p_{start} 的右端来检查从 p_{start} 开始的子轨迹是否与 q 是否是点匹配（行 7-17）。在扫描的过程中，当遇到一个新的位置 p_{end} 时，算法 PMatch 会更新每一个关键词的计数器（行 9-10）。另外算法中选用了 4 个剪枝策略，用于避免不必要的计算。

(1) 根据推论 4 如果遇到一个位置 p_{end}, $d_N(q, p_{end})$大于当前第 k 个结果的最小匹配距离，那么任何包含位置 p_{end} 的子轨迹不可能作为返回结果（行 12）。

(2) 如果 $\min(d_N(q,p_{start}), d_N(q, p_{end})) + \sum_{j=start}^{end-1} d_N(p_j ,p_{j+1})$大于第 k 个结果的最小匹配距离，那么从位置 start 开始往右的子轨迹不可能作为返回结果，因此将起始位置挪至 p_{end+1}（行 20）。

(3) 如果找到一个点匹配，那么就停止从当前位置向右进行查找。根据推论 2，包含点匹配子轨迹的轨迹将会有更大的匹配距离。

(4) 根据推论 1，如果从位置 loc_{end} 到位置 loc_n 到子轨迹都与 *q* 不是点匹配的，那么算法终止执行（行 25-26）。

IsMatch 算法，如果对于查询点 *q* 的每一个关键字都包含在从位置 tstart 到 tend 的子轨迹中，那么子轨迹与 *q* 是匹配的，那么 IsMatch 将计算出匹配距离 md，并用 md 更新 minDist。

4.3.2 候选轨迹生成阶段

查询 *Q* 中的每一个查询点 *q* 在范围逐步扩张的过程中都会计算出扩张范围内轨迹与 *q* 匹配距离，同时将点匹配对按照最小点匹配距离从小到大存入到一个队列 Qe_i 中。算

法引入一个全局队列 G，它有两个基本操作 *pop* 和 *push*，*pop* 操作就是简单得将 G 中用于最小点匹配距离的最小点匹配对输出到候选轨迹集合中去。*push* 操作发生在 *pop* 操作之后，它是从刚才 *pop* 出的查询点所对应的队列中找到一个最小的点匹配对，并把它放到 G 中。因此 G 中总会存在 m 个最小点匹配对。候选轨迹生成过程的一个示例如图 4-3 所示。

全局队列 G 的 *pop* 操作之后，匹配对将保存到候选轨迹集合中去。刚开始的时候，候选轨迹只有很少几个匹配对，称这样的轨迹为部分匹配的候选轨迹。当全局队列 G 中 *pop* 出越来越多的点匹配对，一些轨迹就会最终包含查询 Q 所有的匹配对，称这样的轨迹为完全匹配的候选轨迹。如图 4-3 所示。轨迹 T_1 是完全匹配的候选轨迹，T_2 和 T_4 是部分匹配的候选轨迹。

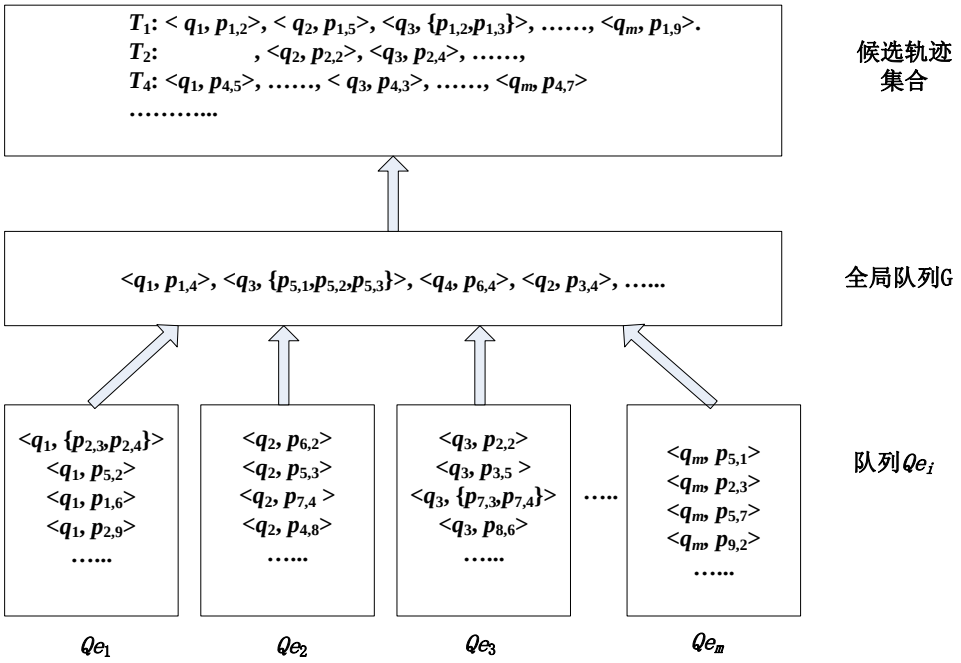


图 4-3 候选集合生成举例

基于上面的分析，候选轨迹生成的算法如下：

算法 Generation(*D*, *Q*, *k*)
输入： 轨迹数据集 *D*，查询 *Q*，常数 *k*
输出： 候选轨迹集合 *C*

```
1   for each qi∈Q do
2     construct the individual Queue Qei for qi
3     initialize the global Queue G and a candidate set C
4   for each individual Queue Qei do
5     pop a matching pair and push it to G
6   repeat
```

7 *pop the shortest matching pair* $<p_j, \{q_j\}>$ *from* G
8 **if** $<p_j, \{q_j\}>$ *is a shortest matching pair* **then**
9 *add* $<p_j, q_j>$ to C
10 *pop a matching pair from* Qe_j and *push it to* G
11 **until** C *contains* k *full-matching candidates trajectories*
12 **return** C

算法 Generation 首先对每一个查询点 q 构建一个有序队列 Qe_i，并初始化全局队列 G（行 1-3），每一个 Qe_i 都会 *pop* 出一个最小点匹配对到全局队列 G 中，这样 G 中始终就含有 m 个匹配对（行 4-5）。之后候选轨迹生成过程开始，一旦 G 中含有 m 个匹配对，它就从中 *pop* 出一个点匹配距离最小的到候选轨迹集合中（行 7-12）。算法的终止条件是候选集合中存在 k 条完全匹配的轨迹。

4.3.3 候选轨迹验证阶段

生成的候选轨迹仍然需要进行验证，这样最终才能得到 Top-k 的轨迹。对于部分匹配的轨迹，算法开始填补缺失的最小点匹配对。为了更好地对候选轨迹进行验证，这里引入部分匹配轨迹点集匹配距离下界的概念。

给定一个查询 Q，一个全局队列 G，一条部分匹配的候选轨迹 T，一个包含候选轨迹集合中的轨迹 T 最小点匹配对的查询点子集 Q_T，那么轨迹 T 与查询 Q 的点集匹配距离的下界为：

$$\mathrm{lb}(Q, T) = \sum_{q_i \in Q_T} Dist_{pm}(q_i, T.pmatch(q_i)) + \sum_{q_j \in Q-Q_T} Dist_{pm}(q_j, T.pmatch(q_j)) \qquad (4\text{-}6)$$

其中$<q_i, T.pmatch(q_i)>$是 T 中的最小点匹配对，$<q_j, T.pmatch(q_j)>$是 G 中的最小点匹配对。

证明：对任意的 $q_j \in Q-Q_T$，记 q_j 到轨迹 T 的最小点匹配对为$<q_j, \{p_T\}>$, $<q_j, \{p_T\}>$为全局队列 G 中未 *pop* 的匹配对。由于 G 中是一个按照最小点匹配距离升序排列的队列，那么有 $Dist_{pm}(q_j, \{p_T\}) \geq Dist_{pm}(q_j, \{p_G\})$，所以有：

$$\begin{aligned} Dist_{pm}(Q, T) &= \sum_{q_i \in Q} Dist_{pm}(q_i, \{p_T\}) = \sum_{q_i \in Q_T} Dist_{pm}(q_i, \{p_T\}) + \sum_{q_j \in Q-Q_T} Dist_{pm}(q_j, \{p_T\}) \\ &\geq \sum_{q_i \in Q_T} Dist_{pm}(q_i, \{p_T\}) + \sum_{q_j \in Q-Q_T} Dist_{pm}(q_j, \{p_G\}) \\ &= \sum_{q_i \in Q_T} Dist_{pm}(q_i, T.pmatch(q_i)) + \sum_{q_j \in Q-Q_T} Dist_{pm}(q_j, T.pmatch(q_j)) \\ &= \mathrm{lb}(Q, T) \end{aligned}$$

因此，$\mathrm{lb}(Q, T)$是 Q 与 T 的点集匹配距离的下界。

基于上面的分析，候选轨迹验证阶段的算法 Validation 如下：

算法 Validation(*C*, *G*, *Q*)

输入：候选轨迹集合 *C*，全局队列 *G*，查询 *Q*

输出：Top-k 轨迹

```
1   initialize result set TK
2   add all full-matching candidate of C to TK
3   sort TK in the order of increasing minimum point set match distance
4   min←k-th trajectory's minimum point set match distance in TK
5   for each partial-matching candidate T in C
6     compute lb(T, Q)
7     if lb(T, Q) ≥ min then
8       continue;
8     else
9       compute Dist_mpsm(Q, T)
10      if Dist_mpsm(Q, T) < min then
11      add T to TK
12      sort TK and update min
13  return the top-k trajectories in TK
```

算法 Validation 是基于部分匹配轨迹点集匹配距离下界来验证候选轨迹的。该算法首先将候选集合中所有的完全匹配候选轨迹加入到结果集 *TK* 中，然后按照最小点集匹配距离升序进行排列，并把第 *k* 大的最小点集匹配距离赋值给 min（行 1-4）。然后开始部分匹配的候选轨迹计算匹配距离的下界 lb(*T*,*Q*)。如果 lb(*T*,*Q*)大于 min，这条部分匹配的候选轨迹就可以排除在结果集之外了（行 6-7）；否则的话，对应查询点继续扩张，并计算出最小点集匹配距离（行 8-9），如果该值小于 min，就将这条部分匹配的候选轨迹加入到 *TK* 中（行 9-12）。在处理完所有的部分匹配的候选轨迹之后，算法终止并将 *TK* 输出。

4.4 实验与分析

4.4.1 实验环境与设计

在这一部分，通过实验来验证本章所提出的算法的性能。实验中使用的轨迹数据集是从 Foursquare 网站上抓取的真实数据，该数据集包含 31557 条轨迹和 215614 个位置点。实验环境是在 Windows 平台下配置如下 Intel(R) Core(TM) Duo CPU E7400 处理器和 2GB 内存，实验程序完全由 Java 语言实现。

在设计实验的时候，选用查询响应时间来衡量算法性能的依据，另外通过返回轨迹数量 k、查询点数量、轨迹数量这三个变量来验证算法的性能。在进行实验的时候，将

选择 Shang[18]和 Zheng[19]的方法来作为一个基准方法来与本章提出的方法进行比较。为了以示区别，本章的查询处理方法标示为 ST。对于查询中的关键词，本实验是从轨迹数据集中随机选择的。

4.4.2 实验结果与分析

针对上一小节的实验设置，对本章提出的算法进行实验分析，并与其它方法做出性能的比较。实验结果及分析如下。

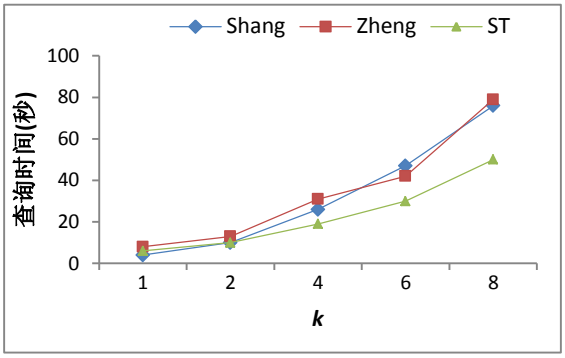


图 4-4 k 值对查询时间的影响

图 4-4 展示了三种方法 k 值对查询响应时间的性能影响。在这个实验当中，使用的是整个数据集，查询点的数量为 4。由图 4-4 可以发现随着 k 值的增大，查询响应时间也基本是呈线性增长的。这是因为随着 k 值的增大，查询点的扩张范围需要再次扩张以保证有更多的候选轨迹，这样也就导致计算量增大，增加查询相应时间。另外从图可以得知 SF 方法的性能是要优于 Shang 和 Zheng 的方法的。

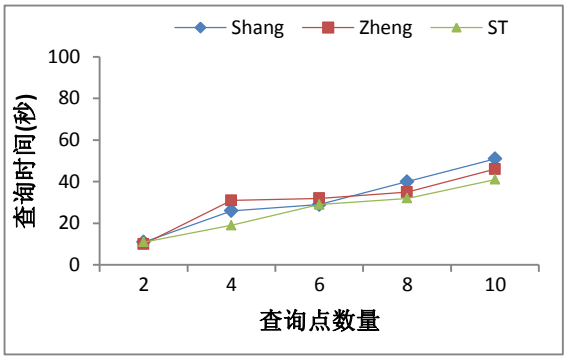


图 4-5 查询点数量对查询时间的影响

图 4-5 展示了三种方法查询点数量对查询响应时间的性能影响。在这个实验当中，使用的也是整个数据集，k 值为 4。由图 4-5 可以发现随着查询点数量的增大，查询响应时间也有所增加。这是因为随着查询点数量的增大，多个查询点扩张的范围也会增大，这样会消耗更多的计算资源，因而导致查询响应时间增加。

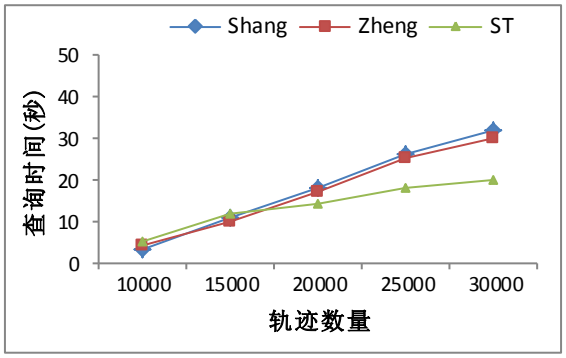


图 4-6 轨迹数量对查询时间的影响

图 4-6 展示了三种方法轨迹数量对查询响应时间的性能影响。在这个实验当中，使用的轨迹是整个数据集随机选取的，选取的数量变化如图，k 值为 4，查询点数量也为 4。由图 4-6 可以发现随着轨迹数量的增大，查询响应时间也有所增加。这是因为随着轨迹数量的增大，单位地理空间的轨迹密度会增加，那么查询点在扩张的过程中会遇到等多的轨迹，因而增加计算量，导致查询响应时间增加。

4.5 本章小结

本章首先讨论了路网环境下的移动对象轨迹空间-文本相似性查询的问题背景，给出了轨迹空间-文本相似性查询的几个相关定义，包括空间网络的定义、轨迹的定义等，并在此基础之上提出轨迹之间的空间-文本相似性度量方法，同时对路网环境下移动对象的轨迹空间-文本相似性查询问题进行了形式化的定义。然后在明确问题定义后，简要介绍了该问题解决方法的基本思想，并提出了解决轨迹空间-文本相似性查询问题高效算法，对算法的三个阶段进行了重点阐述。最后是对提出算法的实验验证，包括实验环境、实验数据集的介绍、实验结果与分析。

第五章 总结与展望

5.1 论文总结

近年来，位置采集设备（如 GPS、智能手机等）的技术发展积累了海量的轨迹数据。由此产生了对轨迹数据多种多样的查询类型，同时也对轨迹数据的存储、运算、处理和管理带来了巨大的挑战。其中移动对象轨迹相似性查询问题近年来备受关注。轨迹相似性查询问题可以用于旅行规划，交通规划以及好友推荐等领域，正是由于这种查询的广泛的应用场景，如何高效的解决轨迹相似性查询问题成为数据库领域的一个热点。本文将这个问题置于路网空间中，更加具有实际意义。

本文完成的主要工作如下：

(1) 系统阐释了路网环境下移动对象轨迹相似性查询的研究背景和现状，明确了这种查询的研究意义和实用价值。

(2) 提出了一种移动对象轨迹时空相似性查询算法。定义出轨迹之间的时间相似性度量、空间相似性度量以及时空相似性度量，采用空间过滤-时间提纯的策略来设计查询处理算法，减少计算量。由于网络 Voronoi 图在处理空间最近邻查询和范围查询有着高效的性能，所以将网络 Voronoi 图引入到空间过滤当中来。最后通过实验验证算法的性能。

(3) 提出一种移动对象轨迹空间-文本相似性查询算法。给出了轨迹空间-文本相似性查询的几个相关定义，包括空间网络的定义、轨迹的定义等，并在此基础之上提出轨迹之间的空间-文本相似性度量方法，同时对路网环境下移动对象的轨迹空间-文本相似性查询问题进行了形式化的定义。然后对查询处理方法的三个阶段—最小点匹配计算阶段，候选轨迹生成阶段和候选轨迹验证阶段进行了详细的描述。最后是对提出算法的实验验证。

5.2 未来展望

由于个人能力有限，本文的工作还存在很多不足之处，以下方面仍然需要进一步改进：

虽然本文提出的算法已经基本实现，并进行了相关的实验验证。但是，采用的数据集并不是采用真实环境下的数据集来进行实验验证的，而是选用生成器生成的数据集。在以后的工作中还需要在真实的数据集上进行算法效率的验证，这样也更加具有说服力。

本文所提出的算法的效率上还有待提高，多个查询点在范围扩张的过程中可能会存在范围重叠，这将导致对同一区域的的重复查找，降低效率。因此未来算法的改进主要

应该将重点放在减少范围的重叠上。

在现实情况下，可能并不能存在一条轨迹能够完全满足用户需求，但是轨迹的某一段却能完全匹配用户需求，这样的话，可以考虑将轨迹进行分段，将满足条件的轨迹段重新组合成一条新的轨迹推荐给用户，这样可以更好的满足用户的偏好。

另外在完善算法的同时可以考虑将算法用于原型系统以更好的服务用户，并考虑进一步建立一个在路网环境下针对移动对象轨迹数据的综合系统查询框架。

参考文献

[1] 袁晶.大规模轨迹数据的检索、挖掘及应用[D].中国科学技术大学, 2012.

[2] Xu Jianqiu, Victor Almeida, Qin Xiaolin. An Efficient Technique for Distance Computation in Road Networks[C].Information Technology: New Generations International Conference,2008,304–109.

[3] Peiquan Jin, Xu Zhang. A New Approach to Modeling City Road Network[C]. Computer Application and System Modeling (ICCASM), 2010 International Conference.

[4] Ralf Hartmut Güting, Victor Teixeira de Almeida, Zhiming Ding. Modeling and querying moving objects in networks[J]. The VLDB Journal,2006,165–190.

[5] Nico Van de Weghe, Anthony G. Cohn, Peter Bogaert et al. Representation of moving objects along a road network[C]. Proc. 12th Int. Conf. on Geoinformatics - Geospatial Information Research: Bridging the Pacific and AtlanticUniversity of Gävle, 2004,187–194.

[6] Yi, B-K., Jagadish, H., Faloutsos. Efficient Retrieval of Similar Time Sequences under Time Warping[C]. 14th IEEE International Conference on Data Engineering, 1998,23–27.

[7] Vlachos, M., Kollios, G., Gunopulos. Discovering Similar Multidimensional Trajectories[C]. 18th IEEE International Conference on Data Engineering, 2002, 673–684.

[8] Chen, L., Ng, R. On the Marriage of Lp-norms and Edit Distance[C]. 30th International Conference on Very Large Data Bases, Morgan Kaufmann, Toronto,2004, 792–803.

[9] Chen, L., Özsu, M.T., Oria, V. Robust and Fast Similarity Search for Moving Object Trajec-tories[C]. ACM SIGMOD International Conference on Management of Data, ACM, Baltimore 2005, 491–502.

[10] Lin, B., Su, J.W. Shapes Based Trajectory Queries for Moving Objects[C]. 13th Annual ACM International Workshop on Geographic Information Systems, ACM, New York 2005, 21–30.

[11] Yanagisawa, Y., Akahani, J., Satoch, T. Shape-based Similarity Query for Trajectory of Mobile Objects[C]. Mobile Data Management, Springer LNCS, 2003, 63–77.

[12] Haibo Wang,Kuien Liu. User Oriented Trajectory Similarity Search[C]. Proceeding UrbComp '12 Proceedings of the ACM SIGKDD International Workshop on Urban Computing, 2012,103–110.

[13] Hwang, J.-R., Kang, H.-Y., Li, K.-J. Searching for Similar Trajectories on Road Networks Using Spatio-temporal Similarity[C]. Manolopoulos, Y., Pokorn′y, J., Sellis, T.K. (eds.) Advances in Databases and Information Systems 2006. Springer, Heidelberg LNCS, 2006, 282–295.

[14] Tiakas, E., Papadopoulos, A.N., Nanopoulos, A., Manolopoulos, Y.,et al, S.: Searching for Similar Trajectories in Spatial Networks[J]. Journal of Systems and Software,2009, 772–788.

[15] Chen, Z., Shen, H., Zhou, X.,el al. Searching Trajectories by Locations: An Efficiency Study[C]. ACM SIGMOD International Conference on Management of Data, ACM, Indianapolis 2010, 255–266.

[16] Jung-Im Won, Sang-Wook Kim, Ji-Haeng Baek et al. Trajectory Clustering in Road Network Environment[C]. Computational Intelligence and Data Mining, CIDM '09 IEEE Symposium. 2009,299–305.

[17] Jae-Gil Lee, Jiawei Han, Kyu-Young Whang.Trajectory clustering: a partition-and-group framework[C]. ACM SIGMOD, 2007, 593–604.

[18] Shang, S., Ding, R., Yuan, B., Xie, K., et al. User oriented trajectory search for trip recommendation[C]. EDBT, 2012, 156–167.

[19] Kai Zheng, Shuo Shang, Nicholas Jing Yuan et al. Towards Efficient Search for Activity Trajectories[C]. Proceedings of the 29th IEEE International Conference on Data Engineering ICDE 2013, 230–241.

[20] Hye-Young Kang, Joon-Seok Kim,Ki-Joune Li. Similarity measures for trajectory of moving objects in cellular space[C]. Proceeding SAC '09 Proceedings of the 2009 ACM symposium on Applied Computing,2009, 1325–1330.

[21] Sultan Alamri, David Taniar, Maytham Safar.Indexing Moving Objects in Indoor Cellular Space[C]. NBiS 2012, 38–44.

[22] H. Alt, A. Efrat, G. Rote, and C. Wenk. Matching planar maps[C]. In SODA, 2003, 589-598.

[23] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data[C]. In VLDB, 2005, 853–864.

[24] J. Greenfeld. Matching gps observations to locations on a digital map[C]. In 81th Annual Meeting of the Transportation Research Board, 2002.

[25] C. Wenk, R. Salas, D. Pfoser. Addressing the need for map-matching speed: Localizing globalb curve-matching algorithms[C]. In SSDBM, 2006,379–388.

[26] Y. Gao, B. Zheng, G. Chen, Q. Li. Algorithms for Constrained K-nearest Neighbor Queries over Moving Mbject Trajectories [J]. Geoinformatica. 2010,, 241–276.

[27] F. Boem, F. A. Pellegrino, G. Fenu, T. Parisini. Trajectory clustering by means of Earth Mover's Distance [C]. 2011 International Federation of Automatic Control World Congress, 2011,4741–4746.

[28] A. Guttman. R-trees: A dynamic index structure for spatial searching[C]. In SIGMOD, ACM,1984, 47–57.

[29] Timos Sellis,Nick Roussopoulos and Christos Faloutsos. The R+Tree: A Dynamic index for multi-dimensional objects[C]. In Proceedings of the 13th VLDB conference,Brighton, 1987, 507–518.

[30] Norbert Beckmann,Hans-Peter Knegel, Ralf Schneider, Bernhard Seeger,The R*-tree: an efficient and robust access method for points and rectangles[C]. Proceeding SIGMOD '90 Proceedings of the ACM SIGMOD international conference on Management of data, 1990, 322–331.

[31] Yufei Tao and Dimitris Papadias. Mv3r-tree: A spatiotemporal access method for timestamp and interval queries[C]. In Proceedings of the 27th International Conference on Very Large Data Bases, San Francisco, 2001, 431–440.

[32] Pfoser, D., Jensen, C., Theodoridis, Y.: Novel approaches to the indexing of moving object trajectories[C]. VLDB,2000, 395–406.

[33] Okabe, A., Boots, B., Sugihara, K., Chiu, S. N. Spatial Tessellations, Concepts and Applica-tions of Voronoi Diagrams[M]. John Wiley and Sons Ltd, New York,2000.

[34] Kolahdouzan, M., Shahabi, C.: Voronoi-Based K Nearest Neighbor Search for Spatial Net-work Databases[C]. 30th International Conference on Very Large Data Bases, Morgan Kaufmann, Toronto 2004, 840–851.

[35] Road Network Datasets, http://www.cs.utah.edu/lifeifei/SpatialDataset.htm[Z].

[36] Brinkhoff, T. Generating Network-Based Moving Objects[C]. 12th International Conference on Scientific and Statistical Database Management, IEEE Press, Berlin,2000, 253–255.

发表论文和科研情况说明

发表的论文：

[1] Wenqiang Sha, Yingyuan Xiao, Hongya Wang, Yukun Li, Xiaoye Wang, 《Searching for Similar Trajectories on Road Networks using Network Voronoi Diagram》, Communications in Computer and Information Science, 2014 年 11 月

参与的科研项目：

本人参与了移动位置服务中时空聚集查询处理方法研究课题，2013 年-至今，天津市自然科学基金

致　　谢

本文的工作是在我的导师肖迎元教授的耐心指导下完成的。肖迎元教授以其深厚的理论功底、严谨的治学态度和科学的工作方法，给予我极大的帮助和影响，使我受益终生。从他的身上，我不仅学到了如何做科研、做学问，更让我学习到了踏踏实实、认认真真的做人态度。在此衷心感谢三年来肖迎元老师对我的关心和指导，同时衷心祝愿肖老师工作顺利，身体健康。

感谢李玉坤老师、王晓晔老师，感谢我的同学赵欣荣、任标、郝亚培等人对我论文中的研究工作给予了热情帮助，在此向他们表达我的感激之情。两年半的时间，大家相互关心、相互帮助，度过了愉快的研究生生活。

感谢我的家人，他们的理解和支持使我能够在学校专心完成我的学业，他们的鼓励与关怀让我在学习上更进一步。

感谢天津理工大学 2012 级计算机与通信工程学院的所有同学，能够在一起求学是我们永远值得珍惜的缘分，祝愿他们在以后的生活和工作中顺利。

最后感谢各位评阅老师审阅本文并提出合理而宝贵的意见。