摘　要

基于移动对象位置的服务（Location Based Service，LBS）是智能交通系统（Intelligence Transportation System，ITS）中的重要组成部分。其中，最优路径选择、实时路况查询、周边目标查询等各类 LBS 服务均要求 LBS 系统的时空数据库系统具备实时快速存取移动对象（主要指车辆）位置数据的能力。但由于车辆位置数据受路网限制分布且时刻变化，不适合采用传统的时空数据索引结构对其进行管理。为解决此问题，本文在 LBS 系统框架下，提出了基于复合索引结构的城市道路网环境下的移动对象索引结构。

复合索引结构是常见的用于管理路网下移动对象位置数据的索引形式，影响其效率的因素包括路网规模、移动对象的更新频率等。其中，路网的划分方式则是最直接、关键的因素，这是因为路网的划分方式直接决定了复合索引的规模以及移动对象的更新模式。为此，本文集中探讨了两种路网划分方式：路段划分及路口区域划分。本文首先提出路段划分的方式，以减少移动对象的更新频率。但由于 LBS 系统被部署在非实时的数据通信环境下，因此本文又提出了一种新的路网划分单元：路口区域（Cross Region，CR），从而能在 LBS 环境下更精确地对移动对象位置进行判断。基于这两种不同的路网划分方式，本文分别对应提出 RR-Tree 与 CR-Tree 两种不同的路网管理结构来作为复合索引的上层结构。

通过实验对上述两种路网划分方式进行比较后发现，尽管 CR-Tree 在索引空间的需求上远超过 RR-Tree。但由于基于路口区域的划分方式解决了 LBS 环境下无法进行实时数据传输的弊病，能够更准确地对车辆位置进行判断，同时在查询效率上的表现也优于路段划分方式，因此更适用于路网下的移动对象数据管理。

关键词：基于位置服务，复合索引结构，路网划分方式，RR-Tree，CR-Tree

Abstract

Location based services (LBS) is an important part of intelligence transportation system (ITS). Many LBS applications such like choosing optimal path, querying real-time traffic information or targets around all need LBS's spatio-temporal database to access moving objects' position data quickly. Since those data are constrained by the road network, and changed quickly, traditional spatio-temporal indexes are hardly to manage them. Therefore, we propose the structures for moving objects in road network based on composite structure underlying the frame of LBS in this paper.

The composite structure is the most common form for managing moving objects' real-time positions underlying road network. The size of the road network or the frequency of objects' updates could all affect structure's performance. But the key factor is the dividing method for road network, because it determines both the size of the structure and the update pattern of objects directly. For that, in this paper, we discuss two different dividing methods for road network: divide by segments and divide by cross region. We first propose the dividing method by segment because it could lower the update frequency of objects. But as the LBS system is disposed in a non-real-time data communication environment, we propose another new basic element called Cross Region (CR) to divide the road network, so that the judgment for objects' positions could be more accurate. Based on these two different dividing methods for the road network, we propose two structures respectively for managing road network to be the upper structure of the composite structure.

Through the comparison between these two different dividing methods, we find that although CR-Tree needs more storage space, it could solve the non-real-time problem caused by LBS, and query quicker than the method with segment, so it is more suitable for the management of the moving objects based on road network.

Key Words: Location Based Service (LBS), composite structure, dividing method for road network, RR-Tree, CR-Tree

学位论文独创性声明：

本人所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果。与我一同工作的同事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。如不实，本人负全部责任。

论文作者（签名）：陆佳民 2008年 6 月 17 日

（注：手写亲笔签名）

学位论文使用授权说明

河海大学、中国科学技术信息研究所、国家图书馆、中国学术期刊（光盘版）电子杂志社有权保留本人所送交学位论文的复印件或电子文档，可以采用影印、缩印或其他复制手段保存论文。本人电子文档的内容和纸质论文的内容相一致。除在保密期内的保密论文外，允许论文被查阅和借阅。论文全部或部分内容的公布(包括刊登)授权河海大学研究生院办理。

论文作者（签名）：陆佳民 2008年 6 月 17 日

（注：手写亲笔签名）

第一章 绪论

在我国，伴随着城市化进程的日益加快，各大中城市的城区面积正在不断扩张；与此同时，社会经济建设的迅速发展也使得城市中私用与公用汽车保有量也在不断增加。面对双重压力，各城市管理部门都在不断加大建设与完善城市交通系统的力度，与此同时，对于智能化交通系统（Intelligence Transportation System，ITS）的研究也逐渐兴起。智能化交通系统最初是在以监控为主体的交通工程（包括交通管理）基础上发展起来的，开始只进行道路和车辆智能化的研究，目前其研究范围已逐渐涉及到铁路、水运及航空等各种交通方式，并逐渐形成一整套为用户及交通管理部门提供道路交通信息的新型交通系统。在ITS中，其“智能化”主要体现在以下3个方面：

(1) 车辆依靠自身智能在道路上安全自由地行驶，在陌生地方不致迷失方向。

(2) 道路依靠自身的智能将交通流调整至最佳状态，缩短行车时间，减少阻塞。

(3) 交通控制管理中心依靠系统的智能对道路和车辆的状态进行实时监控，及时处理事故，保障道路畅通。

这其中，尽管各道路自身的离散式调整与交通控制中心的集中式调整这两种调整方式均能够实现对城市路网中交通流的管理与疏导，但由于交通控制中心处于城市智能化交通系统的核心位置，除了能够从路网整体状态的角度出发设计出更为合理的调整方案之外，也能够全方位地为行驶在路网中的车辆提供各种交通信息的服务，因此备受各级交通管理部门的青睐。

为了使交通控制中心能够更好地进行集中式的路况调整，则需要其对交通道路网中的各项数据进行快速存取，因此一个准确高效的数据库系统就成了交通控制中心中不可获取的部分，这也是本文所要关注的重点问题。

1.1 研究背景

交通控制中心在对当前路网状态进行分析后，依据移动对象的具体位置提供的信息支持被称为基于位置服务（Location Based Service，LBS）。LBS 是建立在定位基础上的服务，其具体含义与工作原理在不同的行业背景下各有不同。总体而言，LBS 是在地理信息平台的支持下，提供给用户各类位置相关服务。在 ITS 的背景下，如最优路径选择、实时路况查询、周边目标查询等均属于 LBS。以下本文对于 LBS 的描述，若无特殊说明，均是指在 ITS 环境下所提供的 LBS 服务。

LBS 的具体工作原理是：移动对象（车辆、行人等所有基于路网行进的物体）通过 GPS 等定位设备获取用户当前位置后，通过移动通信网（WLAN、3G 等）实时地把这一位置信息上传至交通控制管理中心，实现了交通控制管理中心对于全局路网下交通状态的掌握与控制；而面对移动对象所发出的服务请求（路况查询与预测、最优路径选择等），中心服务器可依据当前路网的整体状态，并配合气象、市政交通建设等多种信息进行统一规划，给予用户合理的建议，进而实现对交通状态的疏导与管理。

时空数据库系统是 LBS 中的关键部分。一方面，它所需管理的移动对象数据量十分庞大，据不完全统计，预计至“十一五”末期，南京市的汽车保有量将达到 60 万辆，上海则将超过 150 万辆，而北京目前的汽车保有量就已超过了 300 万辆；而另一方面，车辆的高速移动性又决定了这些移动对象数据只有通过频繁的更新才能使数据库系统准确了解它们的位置变化，根据截止至 2001 的统计数据，在上海市 22,413 个路口中，平均每分钟通过一个路口的车辆数量就高达 33 辆[1]。可见只有采用合适的数据索引对能够对这些数量庞大且更新频率高的数据进行有效地存储与管理。除此之外，位置信息数据由于不同于一般数据的分布特点（如数据的多维分布、动态性等）以及有别于普通数据库的操作（如历史轨迹查询，预测区域查询等），决定了只有研究新的基于时空数据的数据管理结构，包括时空数据索引结构，时空数据查询方法等，才能够为各类 LBS 提供有效的数据支持。

时空数据是指分布在多维空间下，并随时间变化的数据。在 LBS 系统中，时空数据库被部署在系统架构的中心服务器部分，主要用于存储与管理中心服务器所接收到的，路网下各移动对象的实时位置数据。这些数据分布在城市路网中，且随着移动对象的运动而持续更新，因此符合时空数据的特点。同时，由于在路网环境下，移动对象的运动轨迹取决于路网的分布情况，即任何对象的运动范围均不可超出道路范围以外，因此在本文中此类分布在路网中的时空数据被称为受限时空数据。

但是在现有的时空数据库，特别是时空数据索引的研究中，其研究主体并不是路网下的移动对象，而是欧氏空间下的移动对象。在海中航行的轮船、在空中飞行的飞机，以及在城市中任意运动的手机终端等均属于欧氏空间下移动对象的范畴。在欧氏空间中，对象的移动轨迹是线性的、不受约束的，即移动对象在两次更新之间保持匀速直线运动状态，只有当对象的运动速度（方向或速率）发生变化时，才会向数据库提交更新。这些数据在本文中被称为非受限时空数据。

受限时空数据较非受限时空数据的主要不同在于：

(1) 受限时空数据的运动状态受到路网分布的约束，非受限时空数据在速度不变

化的情况下则保持匀速直线运动状态；

(2) 受限时空数据间的距离测算也受到路网分布的约束，非受限时空数据的距离测算即为直线最短距离。

以上这些区别给 LBS 的时空数据库系统带来两点主要问题：

(1) 受限时空数据具有较高的更新密度。这是因为移动对象在路网中行进时，由于道路的约束，导致其前进速度（本文中主要考虑速度方向，速率假设不变）不断发生改变，需要实时地向中心服务器来提交自己的位置变化信息，以使得中心服务器能够准确地了解各移动对象在道路网中的分布状况，为各类位置服务提供有效的数据支持。但这种高密度的数据更新操作不仅会给中心服务器的时空数据库带来巨大的计算压力，也给通信网络带来巨大的数据负载。

(2) 受限时空数据的距离测算信赖于道路网的分布。时空数据索引是依据时空数据间距离的远近对数据进行聚类索引的方法，由于道路网的存在，两个在具有较短直线距离的移动对象可能由于分处不同道路，而导致其间道路距离很长。因此无法采用传统的欧氏空间下时空索引结构来对受限时空数据进行索引管理。

针对第一个问题，LBS 系统特别引入了移动对象追踪模块，以使得系统能够在较低密度的更新水平上，尽可能精确地获取移动对象的位置变化。追踪技术的研究是受限时空数据索引研究的重要前提。丹麦奥尔堡大学的 C.S.Jensen 等人在 2004 年提出了一种基于道路网的移动对象追踪系统[2]，并提出了三种追踪策略：基于点的追踪、基于向量的追踪和基于路段的追踪。其中尤以基于路段的追踪策略能够在保证追踪精度的前提下提供最低密度的更新操作请求，因此在各类 LBS 研究中被广泛使用，也是本文研究的基础。

本文的研究重点是解决上述中的第二个问题，即如何在考虑路网分布的情况下对实时的受限时空数据进行管理，并提出一种新的适用于道路网环境下移动对象实时位置信息管理的受限时空索引，给出其结构和操作描述，并通过实验来证明其正确性和优越性。

1.2 研究现状

数据索引是现代数据库系统中的重要组成部分。在处理大量的数据记录时，通过对索引结构进行查询操作，就可以将属性的值转换为相应记录的地址或地址集，可以达到快速地存取数据的目的。目前就索引结构的研究情况如表 1.1 所示。

在早期的关系数据库系统中，当数据记录的候选键是一个列字段时，可采用

B-tree对此列的记录值进行索引；而当数据记录的候选键是由多个列字段组成时，则采用级联索引的形式，此时组成候选键的列顺序不同，所形成的级联索引也大相径庭。

在处理时空数据时，数据记录的键是由N个空间列字段以及1个时间列字段共同组成的，其中N是指物体运动时所处的空间维度，N>1。由于这N+1个列字段之间彼此没有先后顺序，如果采用传统的级联索引的方式，则数据库系统在处理窗口查询、时间片查询等空间或时空查询时，将会产生查询效率低、占用空间大，甚至完全无法利用索引结构等诸多不良后果。因此只有利用时空索引结构才能有效地解决时空数据的索引问题。

表1.1 已有索引结构研究关系

<table>
<tr><td colspan="5">索引结构</td></tr>
<tr><td>一维</td><td colspan="4">B-Tree</td></tr>
<tr><td rowspan="6">多维</td><td colspan="4">基于B-Tree的级联索引</td></tr>
<tr><td colspan="4">时空索引结构</td></tr>
<tr><td>静态</td><td colspan="3">R-Tree、R*-Tree、R+-Tree、Quad-Tree、BSP-Tree</td></tr>
<tr><td rowspan="4">动态</td><td></td><td>非受限空间</td><td>受限空间</td></tr>
<tr><td>历史位置</td><td>3D R-Tree, TB-Tree, etc</td><td>轨迹索引</td></tr>
<tr><td>当前位置</td><td>2+3 R-Tree, LUR-Tree, etc</td><td rowspan="2">复合索引</td></tr>
<tr><td>当前未来位置</td><td>TPR*-Tree, PMR-Tree, etc</td></tr>
</table>

在现有关于时空数据索引结构的研究中，可依据物体的运动状态分为两个大类，即静态物体的位置索引与动态物体的位置索引。静态物体的位置索引是指对分布在多维空间中的静态物体（如建筑物、道路等）的位置数据进行索引；而动态物体的位置索引则是指对运动在多维空间中的物体（如车辆、飞机等）位置数据进行索引。通常情况下，时空数据索引多指后一种索引结构，而前一种索引结构则被称为空间数据索引。

在空间数据索引的研究中，其原理多是通过自上而下、逐级地划分地理空间的方法，形成树状的索引结构。比较常见的方法包括基于空间划分的网格方式，如Quad Tree[21,22,25]、BSP-Tree[26,28]、K-D-Tree[29,30]、KDB-Tree[27]等，以及基于空间对象区域的层叠外包方式，典型的有R-tree[20]，以及其诸多变体R^+-Tree[23,24]、R^*-Tree[18]等。

在时空数据索引的研究中，又依据物体的运动时域不同划分为三类：对移动对象历史位置信息的索引，对移动对象当前位置信息的索引以及对移动对象当前与未来的位置信息的索引。其中对移动对象历史数据的索引有助于快速查询移动对象的历史运动轨迹，以便对物体运动特性进行统计分析并挖掘出规律信息，此

类中典型的结构有3D R-Tree[17]、MR-Tree[19]、TB-Tree[14]等。对移动对象当前位置的索引则主要用于了解与追踪物体的实时运动状态，以便对现实世界进行模拟，此类中典型的结构有2+3 R-Tree[15]、LUR-Tree[12]等。对移动对象当前及未来位置进行索引的结构根据运动物体的当前位置及运动状态，可以对物体的运动趋势进行预测。由于此类结构独特的预测查询功能（如：查询在未来时刻t区域R中可能存在的物体）可以为许多LBS服务提供基础支持，因而此类索引结构是现在研究的热点，也是本文所要阐述的重点。

按照索引方法的不同，对移动对象当前及未来位置的索引结构也可分为三类：直观地索引物体运动轨迹的方法，如 PMR-Tree[16]；将时空域转换至新空间的转换方法，如 STRIPES[6]的二重转换方法；参数化空间方法，即利用随时间变化的矩形框将运动物体进行外包，再加以索引的方法。参数化空间方法是目前使用最广的针对移动对象当前及未来位置进行索引的方法，其中最为突出的研究成果当属 TPR-Tree[13]，它引入了速度矩形框（Velocity Bounding Rectangle，VBR）的概念可对物体的运动情况加以描述，而其变体 TPR*-Tree[8]由于引入了重插队列，提高了结构插入和分裂算法的效率，更是受到了广泛的使用。

前述的诸多时空索引结构都以非受限时空数据作为研究的主体，由于受限时空数据基于道路网而分布并行进，因此其索引方法与上述方法存在着很多不同。

按研究方法的不同，受限时空数据的索引可分为两类：轨迹索引与复合式索引。轨迹索引针对移动对象在受限空间下的历史移动轨迹进行索引的方法，如 Christian S. Jensen 等人在 2003 曾提出通过对移动对象历史轨迹建模，再利用已有 DBMS 系统管理的方法[9]，此类方法着重分析历史受限时空数据。复合索引结构可用于管理实时数据索引，它由两部分组成：上层的空间索引与下层的时空索引集合。复合索引在管理路网下的移动对象时，其上层部分将路网依据一定的规则划分为不同的单元，并利用空间索引方法对这些单元进行索引；而依据上层的划分规则，移动对象集也被划分为若干个互不相交的子集，每个子集中的移动对象均行进在相同的路网单元中，并被同一个时空索引所管理。当移动对象从一个路网单元中驶出，并驶入另一个路网单元时，则此对象的时空数据就会被从前一个路网单元的时空索引中删除，并插入到当前新的路网单元的时空索引结构中去。由于复合索引结构比较直观地反映了路网中移动对象的运动状态，且不需要对其上层索引结构做太多的更新操作（路网相对静止，变化少），同时，通过将一个大规模的时空数据集划分为若干格小规模的时空数据子集，降低了结构的维护成本，也提高了查询与更新的效率，因此，复合索引结构[3]在受限时空数据的索引中使用广泛，如 FNR-Tree[11]、IMTFN[5]、MON-Tree[7]、IONR-Tree[4]均采用了此种结构。本文的主要工作也是基于复合索引结构的。

1.3 本文的主要工作

本文以受限空间时空数据为研究主体，以复合索引结构为基本方法，针对路网下移动对象的运动特点，对路网的划分方式进行了讨论，分别提出了基于路段（Segment）和基于路口区域（Cross Region，CR）两种基本路网单元，对路网进行划分，并分别建立了 RR-Tree 和 CR-Tree 来作为复合索引的上层静态索引部分。本文所做的主要工作如下：

(1) 对路网进行形式化描述，提出路段的概念，以及路段连接的方法，对 RR-Tree 以及下层时空索引结构进行描述，同时指出其在运行过程中所存在的问题。

(2) 对路口区域进行形式化描述，并指出利用 CR 作为新的路网划分单元有助于改进现有结构的不足。提出以 CR 为基本索引单元的空间索引结构 CR-Tree，给出其结构及基本算法的描述。

1.4 论文的章节安排

本文中各章节安排如下：

第二章介绍 LBS 框架。主要分析其框架结构，以及所涉及的关键技术，阐明受限时空数据索引在 LBS 中的地位。由于追踪系统是受限时空数据库的关键前提，因此对不同的追踪策略进行了分析比较。

第三章对路网及其元素进行了形式化的定义，并提出基于路段的路网划分方法，以及改进的路网连接算法，以减少路段数量。同时还提出了以路段作为基本索引单元构建了路网索引结构 RR-Tree，并利用 RR-Tree 作为上层结构，设计实现了基于路段的复合时空数据索引结构 NCO-Tree。

第四章对基于路段路网划分方式进行分析，指出将路网划分成独立的路段元素并索引后，会存在索引不准确以及更新效率不高的问题。因此本文提出了另一种新的路网划分单元，路口区域，并给出路口区域的形式化定义。同时基于路口区域构建了另一种路网索引结构 CR-Tree，给出其各类元素及操作算法的描述，并对如何构建合适的管理同一路口区域内移动对象数据的时空索引进行了讨论。

第五章对本文所涉及的相关内容进行实验论证。主要对路段划分和路口区域划分这两种不同的路网划分方式进行做索引空间大小、索引准确性以及查询效率上的比较，指出各自的优势及缺陷。

第六章对本文所做的工作进行总结，并指出后续的研究方向。

第二章 基于位置服务系统及其关键技术

在现有的智能交通系统中，其智能性主要体现在车辆自我导航与道路自适应的调节等功能上。就装配有 GIS 导航设备的车辆而言，它可通过设备来指示自身的位置，也可计算出前往目标位置的距离最短路径。

但这种方式存在有两点不足：一是导航设备必须装载有所在城市的电子地图，而且如果城市道路网由于某种原因而被短期或长期的更改，这种更改也无法反映到已经发布的电子地图上；二是导航设备无法反馈道路上的交通状况。在决定如何最为便捷地前往目标地址的计算中，距离上的长短仅仅是考量的因素之一，而道路上交通流状况则更为重要。当导航设备为车辆选择了一条距离最短的路径后，如果该路径上的交通状况十分拥堵，则经过此路径来到达目标地址的时间很可能会比经过更一条距离更长但交通状况良好的路径所要花费的时间要长的多。相同的，在道路的自适应调节上，尽管在当前的某些道路上可以通过摄像头、感应器等设备来监测交通流状况，继而通过调节交通灯等设备来疏导道路。但这种调节是局部性的，而非整体性的。

因此，如果要解决上述现有技术的局限性，就必须要在城市交通系统中设立集中式的交通管理中心，对整体路网中的实时交通状态进行分析，并结合其它一些交通相关信息来动态设置各类交通设施，对移动车辆提供各类 LBS，以达到调整路网状态的目的。

2.1 LBS 系统

2.1.1 LBS 系统框架

LBS 系统由定位系统、移动对象客户端、无线通信网络和中心服务器四部分组成，其体系结构图如图 2.1 所示：

在 LBS 系统中，移动对象客户端通常安装在车辆上，作为 LBS 的终端设备，它需要能够获知自身的位置，并将位置数据与服务请求等数据通过通信网络传输给中心服务器，同时也可从通信网络接收来自于中心服务器的服务反馈数据。因此，它必须要装置有定位终端设备以进行自身定位，无线通信设备以进行数据传输，以及便携式计算设备以进行必要的数据处理。

定位系统可由卫星定位系统或移动电信定位系统来充当。卫星定位系统是利

用多个部署在太空中的定位卫星和若干个地面站组成，通过移动对象所持有的定位终端，可以确定对象在地球上所处的位置及海拔高度等，其定位精度在 10 米左右。由于卫星定位终端代价昂贵，普及度不高，因此也可选择移动电信定位系统来进行定位。移动电信定位系统是利用移动通信网结合用户的移动终端设备（如手机等）进行定位的系统，这种定位的精度在 50-200 米左右。

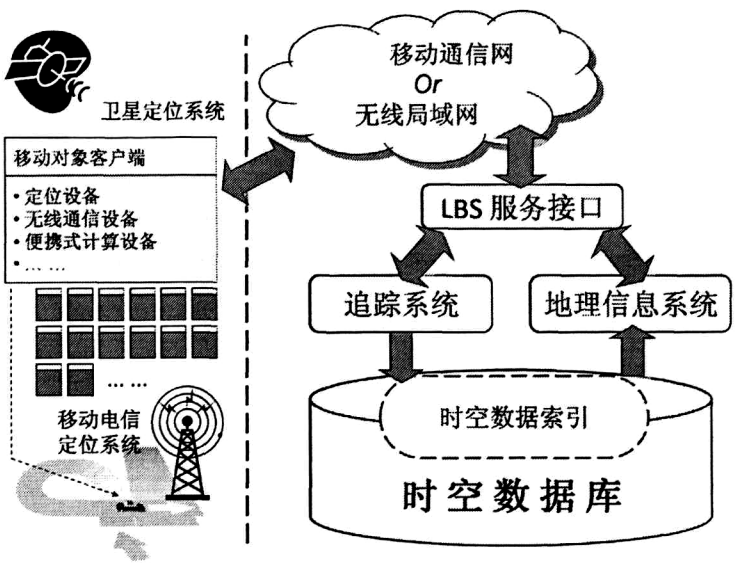


图 2.1 基于位置服务（Location Based Service，LBS）系统框架

无线通信网络是为了便于移动对象客户端与中心服务器进行数据传输的媒体。它可以通过移动通信网，如 GPRS、CDMA、TD-SCDMA、WiMax 等 2G 或 3G 技术来构建，也可通过如 WAPI，WIFI 等无线局域网技术来实现。

中心服务器是提供 LBS 的核心所在。为了能提供准确有效的位置服务，如最优路径查询、周边交况预测、路网调度等，它需要配置有强大的数据计算能力，良好的通讯网络，从而可以实现与众多终端设备的交互或和其它资源的互连，以便能获取各类信息并进行相应的管理与分析。中心服务器由以下四部分组成：LBS 服务接口、追踪系统、地理信息系统与时空数据库。LBS 服务接口主要用于接收来自客户端的服务请求，并将位置信息发往追踪系统，将服务信息发往地理信息系统，并将服务反馈传送给客户端。追踪系统在接收到客户的位置信息后，将相关数据写入时空数据库，并告知客户端当前的追踪范围，以便减少数据更新的次数。地理信息系统则服务具体的服务请求处理，并从时空数据库中获取必要的数据支持。时空数据库用于存储客户端的位置数据等地理信息系统会用到的数据内容，以便提供支持，其中的时空数据索引则主要用于管理客户端的位置数据，以便进行快速的查询与处理。

2.1.2 LBS 关键技术

2.1.2.1 空间定位技术

空间定位技术使得移动对象客户端能够持续获取自己在路网中的位置，这是所有基于位置服务所需要的基础数据。在现有技术下，卫星定位技术与移动电信定位技术是较为常见的。

卫星定位技术，如美国的 GPS 技术，欧州的“伽利略”计划，我国的“北斗”计划都属于此类技术。该技术由三大部分组成：空间定位卫星系统、地面监控系统、用户定位信号接收机。用户只需配备专用的定位信号接收机，即可获知自己所处的位置。这种定位技术的优势在于：全天候，不受任何天气的影响；全球覆盖（高达 98%）；三维定速定时高精度；快速、省时、高效率；应用广泛、多功能；可移动定位。GPS 能提供 20-30 米的定位精度，定位时间在 30 秒至 1 分钟之内。由于这种方案需要有配有专用的信号接收装置，因此价格稍显昂贵，不易普及。

移动电信定位技术提供了一种较为简便的定位手段，它利用 GSM 移动通信网的蜂窝技术来实现位置信息的查询。这是因为 GSM 无线通信网是由许多像蜜蜂蜂窝一样的小区构建而成的，每个小区都有自己的编号，通过手机所在小区的识别号就可以知道手机所在区域。其主要定位手段有：

(1) COO (Cell of Origin) 定位技术。COO 定位技术即基于 Cell-ID (小区识别号) 的定位技术。它通过采集移动台所处的小区识别号来确定用户的位置。这种技术的定位精度取决于所在小区的半径，如在市区，基站密度较高，COO 定位精度可以达到 200 米左右；而在郊区，基站密度较低，COO 定位精度只能达到一两公里。

(2) 增强观测时差技术 (Enhanced Observed Time Difference，E-OTD)。E-OTD 是通过放置位置接收器或参考点实现的，这些参考点分布在较广的区域内的许多站点上，作为位置测量单元 (Location Measurement Unit，LMU) 以覆盖无线网络。每个参考点都有一个精确的定时源，当具有 E-OTD 功能的手机和位置测量单元接收到来自至少 3 个基站信号时，从每个基站到达手机和位置测量单元的时间差将被计算出来，这些差值可以被用来产生几组交叉双曲线，并由此估计出手机的位置。E-OTD 方案可以提供比 COO 高得多的定位精度，约在 50-150 米左右。但这种方案对手机和网络的要求较高，实现也比较复杂。

(3) 此外，类似 TDOA、TOA、AOA、TA 等技术均属于移动电信定位技术的范畴。移动电信技术由于采用普及度很广的移动电话作为定位的终端设备，因此方便推广，但定位精度较低。

2.1.2.2 追踪技术

尽管移动对象客户端借助定位技术能够持继获知自身在路网中的位置，但是作为LBS核心部件的中心服务器却无法实时地更新路网中移动对象的准确位置。其问题首先在于移动对象的数量巨大，如前文 1.1 节所述，北京市目前的汽车保有量就已超过 300 万辆。其次无线通信网络的带宽十分有限，最新的 TD-SCDMA 技术，其最高数据传输率也仅有 2.8Mbps，而类似的 WCDMA、CDMA 2000 等其最高速率也仅有 2Mbps，这种有限的带宽资源是无法支撑大数据量、高频率的数据更新操作的。因此如何能够使中心服务器在降低移动对象更新频率的同时，又能保证其所管理的对象位置是较为精确的呢？追踪技术的出现使得这一问题的解决成为可能。具体的追踪技术，以及不同的追踪策略的比较将在后述做更为详细的说明。

2.1.2.3 地理信息处理技术

地理信息服务技术针对不同区域和城市的基础的地图数据，简称电子地图，提供对电子地图的管理、发布、地理分析、地理编码、路径搜索等诸多功能。除此之外，地理信息处理技术还能利用时空数据库中所管理的移动对象时空数据，模拟当前路网中的交通状态，加以分析，亦可结合其它交通相关信息的分析，来对未来一段时间内的交通状态进行预测，以期为移动对象提供尽可能全面与丰富的 LBS 内容。

系统分为数据库管理、地理信息引擎、地理信息服务 3 个层次。其中数据库管理负责管理运营平台的所有数据。地理信息引擎实现对底层数据库的管理、操作，并支持基于平台的二次开发，一般采用成熟的地理信息平台软件和功能软件。地理信息服务分为地理信息核心服务和地理信息应用服务框架两个部分，其中核心服务是在地理信息引擎的基础上提供的地理信息发布、地理信息分析、路径搜索、地理编码等核心服务功能；应用服务框架是在综合各种地理信息核心服务的基础上提供的应用服务框架，包括城市黄页服务、个人导航服务、公交换乘服务、朋友位置查找服务、地址查找服务等。其核心功能有：

(1) 地图发布引擎：根据用户的需要生成适用于不同终端的地图。所发布的地图

有矢量、栅格或混合模式。

(2) 地理分析引擎：根据经纬度定位地图及相关查询功能。例如可以根据经纬度查出一定范围内地图的基本信息（区、街道、大厦）；可以根据 2 个点的经纬度给出距离信息及乘车信息；可以根据经纬度查询到一些基本的黄页信息（如 ATM 机、餐馆、卡拉 OK、银行等）。

(3) 地理编码引擎：地理编码服务器转换基于字符串的地址（如门牌号码），使之成为地图上准确的经纬度点。要提供上述服务，LBS 空间数据库除了包含电子地图数据库外，还必须提供地理编码数据库。

(4) 路径搜索引擎：根据给定的 2 个点的位置寻找两点间距离最短路径或者时间最短路径，并产生描述推荐路线的街道图形和包括预计路程长度及时间在内的文字描述，该服务可用于个人导航和车载导航以及其它行业的应用。根据用户的请求，服务引擎能够在图形路线介绍和文字指南中重点标示出娱乐场所、路标、交通事故和道路状况（施工、天气影响、路面状况等）。

2.1.2.4 空间数据库技术

空间数据库存储和管理是 LBS 服务平台的关键部分。LBS 空间数据可以在大型商用关系数据库管理系统的基础上构建，以便实现空间实体几何数据和属性数据的一体化存储，充分利用数据库本身的管理能力实现海量空间数据的存储和管理。如中国联通的 LBS 服务平台采用了 Oracle Spatial 来管理和存储空间数据。除了空间数据存储、空间分析、空间查询等基本功能外，空间数据库系统还必须实现以下功能。

(1) 多比例尺图库管理：即建立金字塔型的多级比例尺区域索引和图层索引表。

(2) 元数据管理：通过元数据对分布在各个场所信息资源做有效管理与共享。

(3) 数据安全管理：数据中心能够在数据交换时保障数据安全。

(4) 数据远程协同工作管理与数据同步：重点解决多级服务器之间的动态数据复制问题与数据远程调用问题，实现空间数据操作的主要功能，数据版本控制。

2.2 追踪技术分析

合适的位置追踪系统是 LBS 的重要组成部分。交通控制管理部门的中心服务器只有在尽可能准确地获取路网中所有移动对象的实时位置的前提之下，才能够对当前路网中的具体交通情况做出准确的判断，并进行有效的规划以提供相应的服务。由于中心服务器在 LBS 中处于被动接收的位置，即它不能够主动收集

移动对象位置数据，而只能被动接收移动对象所上传的信息。若移动对象能够在行进过程中，不间断地向中心服务器报告位置自然是保证数据精确的最佳途径。但如 2.2.2 节所述，由于移动对象数量巨大，而无线通信网络的带宽有限，导致中心服务器无法通过持续更新的方式，实时地掌握辖域内移动对象的精确位置。因此需要使用合适的追踪技术来推算移动对象的可能位置，使得在不降低用户位置的精确度的前提下，尽可能地减少更新位置操作的频率。因此只有采用合适的移动对象位置追踪系统，使得中心服务器通过低密度的数据更新方式也能够准确地获知各移动对象的实时位置是保障 LBS 有效性的重要前提。

基于这一目的，丹麦奥尔堡大学的 C.S.Jensen 等人在 2004 年提出了一种基于道路网的移动对象追踪系统。本节将重点对这一追踪系统进行描述，并分析数种不同的追踪策略：基于点的追踪、基于向量的追踪以及基于路段的追踪。在对这几种追踪策略进行比较后，则将引出本文的研究方向。

2.2.1 基于道路网的移动对象追踪系统

图 2.2 给出了 C.S.Jensen 等人所提出的移动对象追踪流程，这也是大多数 LBS 系统中所使用的方法。这个系统的基本思想是，在追踪系统获得某一客户端的实时位置后，可依据对象当前的位置与速度，决定采用某种追踪策略来对对象进行追踪定位，并将选定的追踪策略反馈给客户端，由客户端计算出对应的预测范围。所谓预测范围是指，系统认为该用户的运动轨迹在将来某段时间内不会超出的某个区域。因此当客户端获得预测范围后，它只需在每次获得实时位置信息时，将当前位置与预测范围进行比较，若其依旧在预测范围内运动时，就不需要向中心服务器提交更新，若超出了预测范围，则需要提交更新，以获得新的预测范围。

整个系统按照 C/S 架构部署，分为左右两个部分。图左侧为客户端部分，即安装在 LBS 系统的移动对象客户端部分的软件，而右侧则为服务端部分，即是 LBS 系统的中心服务器的追踪系统部分。客户端首先从定位设备获得其位置信息，然后将其位置数据发送给服务端。服务端得到位置数据后，首先对时空数据库进行更新，继而依据客户端的位置、速度等信息来决定适用的追踪策略，并将此追踪策略反馈给客户端。若选用的是基于路段的追踪策略时，则需计算出对象所在路段的标号。在获取服务端的追踪策略后，客户端会计算出相应的预测范围。在后续过程中，客户端在每次获取实时位置的同时，都要将实时位置与预测范围进行比较。如果这两个位置差超出了给定的精度要求，客户端产生一更新请求给服务端，如果没有，则等待下一个定位时刻的到来。这个过程一直进行直到客户端终止追踪服务为止。

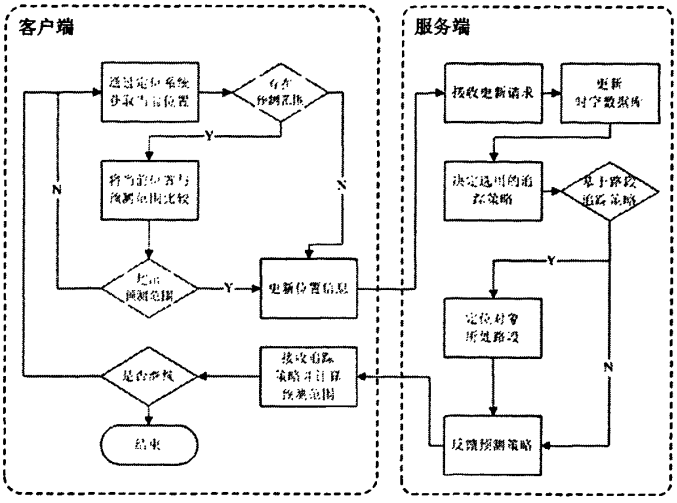


图 2.2 追踪系统框架结构

2.2.2 不同的移动对象追踪策略

常见的移动对象追踪策略有三种：基于点的追踪策略（Point-Based Tracking）[2]；基于向量的追踪策略（Vector-Based Tracking）[10]；基于路段的追踪策略（Segment-Based Tracking）[2]。这些追踪策略在对位置数据的追踪流程上，与图 2.2 描述的追踪流程相同，但他们所给出的预测范围是不同的。

基于点的追踪策略认为移动对象总是处于静止状态。使用这样的策略，服务端对客户端的预测位置总保持其前一次的更新位置。以图 2.2 系统为例：服务端认为客户端始终处于上一更新的位置。这样，由于客户端运动的关系，当其真实位置与服务端预测位置（即上一次更新位置）偏离超过预测精度时，便触发服务端中心数据库更新客户端对象位置信息。用公式表示为：

$$P_{anti}(t)=\begin{cases}P_0 & (|P_0-P_t|<Len_{\lim})\\ P_t & (|P_0-P_t|>Len_{\lim})\end{cases} \qquad \text{（式 2.1）}$$

其中 P_{anti} 为预测位置，P_0 为前一次更新时刻对象的位置 (即服务端数据库存储的对象位置)，P_t 为时刻 t 时追踪对象的实际位置（即其 GPS 位置），$Len_{\lim}$ 为范围精度。其跟踪轨迹如图 2.3（a）所示：圆圈半径表示预测精度，实心点表示当移动对象预测位置与其实际位置的偏差大于预测精度时，对象触发更新时的实际位置，两粗线条表示路网上相连的两条路段。

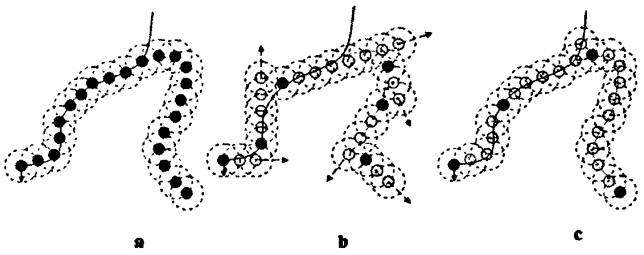


图 2.3 三种移动对象追踪策略的轨迹比较

基于向量的追踪策略认为对象延当前速度方向做匀速直线运动。以图 2.2 系统为例：假设服务端存储的某客户端位置为 $P_0(x_0, y_0)$，速度 $V_0(v_{x0},v_{y0})$。服务端认为客户端以 P_0 为起始点，V_0 为速度做匀速直线运动，即客户端位置由函数 $P_{anti} = P_0+V_0 * (t-t_0)$决定。同样，由于移动对象运动变化的关系（如速度，方向的改变），其当前位置与服务端预测位置的偏离超过预测精度时，就会触发对服务端中心数据库的更新操作，以便获得新的预测范围。用公式表示为：

$$P_{anti}(t)=\begin{cases} P_0+V_0(t-t_0) & (|P_0-P_t|<Len_{\lim}) \\ P_t & (|P_0-P_t|>Len_{\lim}) \end{cases} \qquad \text{（式 2.2）}$$

其追踪轨迹如图 2.3（b）所示：其中实心点表示当移动对象预测位置与其实际位置的偏差大于预测精度时，对象触发更新时的实际位置，空心点表示更新时刻服务端存储的对象位置，即预测位置，两条粗线表示路网上相连的路段。

基于路段的追踪策略认为路网中对象的运动必须遵循路网限制。路段是路网的一种元素，路段经由两端的路口与其它路段相连，而在路段的中间则没有路口。因此当移动对象进入某一路段后，它只能沿着当前路段行进，直至末端后方可离开并转到其它路段中去，或停留在当前路段中不再前行，而不能从路段中间离开。因此基于路段的追踪策略认为对象会沿着当前路段做匀速运动，直至超出该路段的范围为止。以图 2.2 系统为例：假设服务端存储的某客户端位置为 $POS_0(SegID_0, offset_0)$，速度 V_0，时间为 t_0，其中 $SegID_0$ 为所在路段的标号，而 $offset_0$ 则是指物体在该路段中的具体位置。则服务端认为对象以 POS_0 为起点，V_0 为速度在路段 $SegID_0$ 上作匀速运动，即客户端位置由函数 $POS_t = POS_0 + V_0 * (t - t_0)$决定。同样，当对象由于运动变化的关系（如速度、方向的改变），而导致其真实位置与预测范围之间的偏差超出既定精度，或移动对象改变了运动轨迹而不再继续在标号为 $SegID_0$ 的路段中行进时，就会触发对服务端中心数据库的更新操作，并获得新的预测范围。用公式表示为：

$$POS_{anti}(t)=\begin{cases} POS_0+V_0(t-t_0) & (|POS_0-POS_t|<Len_{lim})\cap(SegID_t=SegID_0) \\ POS_t & (|POS_0-POS_t|>Len_{lim})\cap(SegID_t=SegID_0) \\ POS_t & (SegID_t\neq SegID_0) \end{cases} \quad \text{（式 2.3）}$$

其追踪轨迹如图 2.3（c）所示：其中黑点表示客户端对象更新时刻的位置，白点表示更新时刻服务端存储的客户端对象的位置。

在基于路段的追踪策略下，追踪系统认定移动对象的轨迹不会超出当前路段的范围，其位置由更新时刻的位置及速度决定。假设根据追踪系统所得到的数据，对象会在 t'时刻后离开当前路段，但对象在 ＞t'的时刻后一直未提交更新，则系统认为该对象的位置不再发生变化，其追踪策略就会从基于路段的追踪转换为基于点的追踪。

基于路段的追踪策略对路网的精确度要求很高。如果因为某种原因，定位技术没有能够得到运动对象所在的路段（这可能时由于电子地图的精确度不高，或者是地图未能够覆盖运动对象所在区域），预测策略改变为基于向量的预测。但在下一个更新来临时，服务端将继续寻找其所在路段。

客户端对服务端的更新频率随使用不同的追踪策略而各不相同，举例来说：

假设一个出租车在路网上移动，追踪预测精度设定为 100m，即我们需要追踪误差在 100m 内的出租车位置。如果采用基于点的追踪，当出租车运动距离每超过 100m 时，便触发一个服务端的数据库中对此出租车的位置更新，其运动轨迹情况如图 2.3（a）所示。如果使用基于向量的追踪策略，便需要考虑出租车运动方向和运动速度，其运动轨迹情况如图 2.3（b）所示，可以看出采用基于向量的追踪策略减小了更新频率。如果我们有一个相关路网的电子地图（用于定位移动对象所在路段），就可以做基于路段的追踪，如图 2.3（c）所示。与基于点的追踪策略相比，基于路段的追踪策略也减少了追踪的更新频率。

2.2.3 不同追踪策略的实验比较

在对不同追踪策略的更新密度比较上，本文基于真实的道路网：美国加利福尼亚州公共交通网[1]的局部路网，通过路网移动对象的模拟平台，生成了移动对象的位置数据。截取了在精度分别为 20、50、100、200、500、1000 个单位长度内的移动对象更新密度数据，即单位时间内（两个位置数据快照间的时间间隔）在不同追踪精度下，移动对象的更新请求数量占移动对象总定位数量的比例，比较结果如图 2.4 所示。

[1]注：http://libraries.mit.edu/gis/data/findingaids/esridatamaps1999.pdf

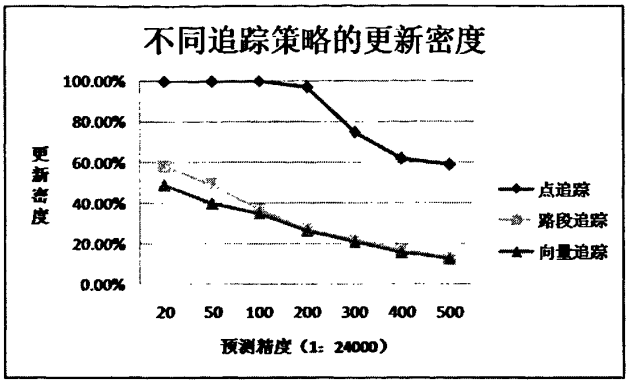


图 2.4 不同追踪策略的更新密度比较

从图 2.4 中可以看出，基于点的追踪策略与其它两种追踪策略在更新密度上差异较大。在 20~200 的精度单位下，其数据更新比例达 100%，而其它两种追踪策略的更新比例在 50%左右。这说明在此精度范围内，路段追踪与向量追踪这两种追踪策略在更新比例上显著减少，提高了追踪效率。

同时，在上述实验中分析得出，影响基于路段追踪策略的数据更新原因主要集中在两个方面：移动对象加减速运动或是移动对象改变其行驶路段。其中，路段的原因所占比例远大于加减速原因，这是由移动对象模拟平台的特点所决定的。移动对象模拟平台所用的路段仅仅为路网中两个交点间部分，其平均长度较短，移动对象频繁的更新路段引起大量的信息更新。因此在实验比较中，向量追踪的更新密度较路段追踪稍低。但由于路段间存在可连接性，可以：

(1) 建筑，停车场，交通堵塞等与路相关的信息可以与移动对象的位置相联系。
(2) 基于路的距离可以用来替代几何距离。
(3) 加速度习惯，拐弯规律，以及其他基于路的信息数据可以用来被开发用于预测运动物体的位置。

因此本文选择基于路段的追踪策略来作为 LBS 追踪系统中的基本策略。

2.3 本章小结

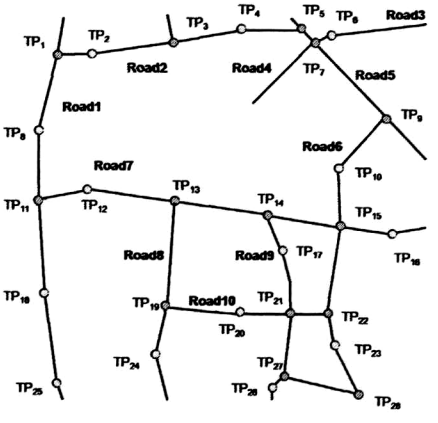
本章主要讨论了 LBS 系统与复合索引之间的重要关系。在阐述 LBS 系统框架与关键技术的同时，更是着重分析了 LBS 中对复合索引结构有着决定性作用的追踪技术，并对不同的追踪策略进行了比较。从比较中可以发现，基于路段的追踪策略可以有效地减少 LBS 的更新操作频率，提高 LBS 在时空数据存取上的效率。因此本文将采用基于路段追踪策略的追踪系统来向所设计的受限时空索引结构提供实时的移动对象位置数据。

第三章 基于路段的时空索引结构

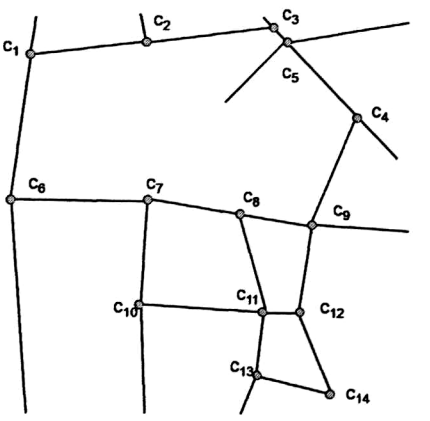
复合索引结构是目前针对受限时空数据，尤其是道路网下时空数据的最有效和使用广泛的形式结构。而通过 1.2 节对复合索引结构的描述可知，当移动对象在相同的路网单元中运动时，移动对象的数据更新代价较低，这是因为移动对象无需通过复合索引的上层路网索引来查询它当前所在的路网单元；但当移动对象从一个路网单元转移到另一个路网单元中时，其更新代价则较高。可见，降低复合索引中的更新代价的关键在于路网的划分方式，通过合理地划分路网，可以使车辆在移动中尽量少地提交不同路网单元间的更新请求，从而可以提高复合索引的整体性能。因此，本章将在对路网构建模型的基础上，选取路段作为复合索引中路网的划分单元，从而达到提高索引效率的目的。

3.1 路网元素的定义及形式化描述

(a) 真实的道路网环境



(b) 细节化道路网模型



(c) 图论化道路网模型

图 3.1 真实路网环境与两种路网模型

城市道路网由不同规格且数量众多的道路构成，不同道路之间纵横交错，存在或不存在命名。为了能够利用信息技术有效地解决 LBS 中所存在的诸多问题，就必须要以一致化的方式来对路网、路网中的各类元素以及元素间的拓扑关系进行形式化的定义与描述。这类问题被称为路网建模（Network Modeling）问题，是一切 LBS 研究的重要前提。

城市道路网环境并不考虑真实交通系统下的交通规则等信息要素，而仅仅对路网的形状、邻接关系做形式化的描述。R.H Güting 等人在 2006 年中对道路网模型化做了深入的分析[31]。依据形式化程度的不同，又可将路网的模型化分为细节化道路网模型与图论化道路网模型两种形式。

细节化道路网模型是指保留路网中的细节元素，以在索引结构中能够尽可能真实地对路网环境加以模拟。在这种模型下，路网元素的定义如下：

定义 3.1：道路网（Road Network）

车辆在城市中行驶时，其轨迹所处的二维受限空间，简称为 RN。

定义 3.2：路网折点（Turning Point）

RN 中可以使得车辆行驶方向发生变化的点被称为折点，简称 TP。

依据车辆在 TP 处可转换方向的数量的不同，而将 TP 又进一步地细分为：

拐点（Connecting Point）：车辆的行驶方向在此 TP 处发生变化，且方向变化有且只有一个，则称这样的折点为拐点，简写为 CNP。

路口（Cross Point）：车辆的行驶方向在此 TP 处可发生变化，且方向变化数量大于 1，则称这样的折点为路口，简写为 CRP。

定义 3.3：路网折线（Polyline Road）

RN 中供车辆行驶的线段称为路网折线，简称 PR。

依据 PR 中 TP 的数量与性质不同，可将 PR 细分为：

边（Edge）：处于两个 TP 之间，且中间没有其它 TP 的 PR，称为边。车辆在边中行驶时不会改变其行驶方向。若边的两端 TP 标号分别为 TP_i 和 TP_j，则该边被标为 E_{i-j}。

路段（Segment）：处于两个 CRP 之间，中间包含若干个 CNP 但没有其它 CRP 的 PR，称为路段。车辆在路段中行驶时不会转换到其它路段中去，但其行驶方向可能会发生变化。若边的两端 CRP 标号分别为 CRP_i 和 CRP_j，则该边被标为 S_{i-j}。

路径（Route）：处于任意两个 TP 之间，中间包含有若干个 CNP 和 CRP，拥有行政命名的 PR，称为路径。

由于道路网与计算机数据结构中的图近似，因此通常采用图论中的定义对路网元素进行描述，其所得到的路网模型就被称为图论化路网模型。图论化路网模

型较细节化路网模型则去除了道路网中的细节元素，而只对路网间的拓扑关系进行描述，其中对路网中的各类元素及关系定义如下：

定义 3.4：图论化道路网（Graphed Road Network）

采用图论中的定义对道路网作形式化描述所得到的路多模型称为图论化道路网，简称为 GRN。它由路口集合与路段集合两部分组成。

定义 3.5：路口（Cross）

GRN 中的点集合，车辆通过路口从一个路段驶向另一路段。

定义 3.6：路段（Segment）

GRN 中的边集合，路段的两端分别是两个不同的路口，车辆通过路段从一个路口驶向另一路口，且中间不包含其它路口。路口 C_i 与 C_j 之间的路段标识为 $C_{i\text{-}j}$。

定义 3.7：路口邻接关系（Cross Neighboring Relationship）

若两个路口间存在路段，则称这两个路口间存在路口邻接关系。

定义 3.8：路段邻接关系（Segment Neighboring Relationship）

若一个路段的某个端点与另一路段的某个端点相同，则称这两条路段间存在路段邻接关系。

定义 3.9：相关关系（Joined Relationship）

若一个路口是一条路段两端路口中的一个，则称路口与此路段相关。

在图 3.1 中，分别演示了真实的路网环境以及其细节化路网模型和图论化路网模型。

定义 3.10：路口的度（Degree of Cross）

与某个路口存在相关关系的所有路段的数量称为该路口的度。

图 3.1(a)是真实的道路网环境，图 3.1(b)则是对图 3.1(a)细节化模型化后所得到的结果，图中的点均是 TP，用白色的点表示 CNP，而且深色点表示 CRP。TP_1 与 TP_2 之间的直线是边 $E_{1\text{-}2}$。由于 TP_1 与 TP_3 均是 CRP，因此 TP_1 与 TP_3 之间的折线是路段 $S_{1\text{-}3}$，其中包含了一个 CNP TP_2。TP_1 与 TP_{25} 之间的折线段，包含有一个 CRP TP_{11} 和两个 CNP TP_8 和 TP_{18}，且此折线段拥有行政命名 Road1，因此此折线为路径。

图 3.1(c)是对图 3.1(a)图论化模型后所得到的结果，其中的黑点表示路口，而边则表示路段。C_1 与 C_2 之间的路段为 $S_{1\text{-}2}$，C_1 与 C_2 之间存在邻接关系，且 C_1 与 C_2 均与 $S_{1\text{-}2}$ 相关。

在上述两个路网模型中，细节化模型对路网的模拟程度高，可真实直观地描述路网中的各类细节，但描述复杂，需使用不同规模的数据结构加以存储；而图论化模型更侧重对路网中各基本元素间的关系描述，对路网的模拟程序较低，但

描述简单，可使用固定大小的数据结构加以存储。而在时空索引结构中，每个索引单元均占据固定大小的磁盘页面，且每个单元中包含有相近数量的空间元素描述，以达到快速存取的目的，因此在时空索引结构中，一般采用图论化的路网模型或图论化与细节化复合的路网模型对路网进行描述。

3.2 基于路段的路网索引结构 RR-Tree

在对路网进行形式化描述与建模后，选取合适的路网元素作为静态路网索引的基本单元，是建立适合路网下移动对象数据索引的首要任务。从 3.1 节中所给出的细节化路网模型中可以看到，由于移动对象大部分处于路网折线中，而小部分处于路口处的移动对象亦可归并到周边的路网折线中去，因此路网折线比较适合作为路网索引的静态单元。

在不同的三种路网折线中，边作为最基本的路网组成单元，具有简单、易操作的特性。但数量多，容易增大索引结构的体积。而且在 LBS 系统中，用户依赖追踪系统来向服务器报告自身的位置。追踪系统采用的是基于路段的追踪策略，车辆在超出原路段范围，并进入路段范围后才会像系统提交更新。边作为路段的一部分，当车辆驶出边的范围时，是不会向系统提交更新。因此当系统需要对移动对象位置进行调整时，就不得不对整个路网索引进行定位查询，以把车辆更新在更为准确的位置上。而实际上，由于车辆在路网中处于受限运动的状态，因此可以简单地把正在道路中行驶的车辆运动方向视为正向行驶，即从当前路段的起始点驶向路段终止点，或逆向行驶，而忽略其具体指向。由此可认为车辆在拐点处的速度变化并不造成其行驶状态的变化。而当车辆在路口处改变了行驶方向，变换了行驶路段的行为，则可视为其行驶状态的变化。因此若采用边作为路网索引的基本单元时，当系统侦侧到车辆超出了当前边的范围时，尽管其行驶状态并未发生改变，但为了准确定位，仍需要对索引结构进行调整，造成了更新次数的增加。路径在路网中具有数量少，易区分的特性。但由于路径的长度较大，覆盖区域大，对于路径的定位查询将会涉及到对多条其它路径的访问，增加了查询成本，降低了查询效率。

路段作为路网的基本单元，其长度较路径短，较边长，车辆在路段中行驶时，其行驶状态不发生变化，只有当车辆超出路段，行驶状态变化时，才会向索引结构提交更新，因此路段较其它两种路网折线元素而言，最适合作为路网索引的基本索引单元。在此基础之上，本文提出了基于路段的路网索引结构 RR-Tree（Road R-Tree）。

3.2.1 RR-Tree 元素和结构

RR-Tree 是在对已有的 R*-Tree 算法进行改进的基础上，以路段作为基本的空间索引单元所提出的，针对静态路网环境进行索引的路网结构。

R*-tree 是通过对多个距离较近点之间迭代外包来构建索引的。当路段作为基本索引单元时，首先要利用最小外包框 MBR（Minimum Bounding Rectangle）对其进行外包，如图 3.2 所示。

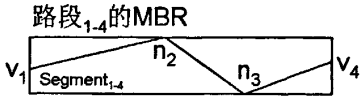


图 3.2 对于路段的 MBR 外包

当各个路段通过各自的 MBR 外包后，基于 MBR 中心点之间的相对位置与重叠关系可以判断不同路段之间的地理关系。因此，在 R*-Tree 算法的基础上，通过对各个单元 MBR 的迭代外包，就可以对路段的 MBR 构建索引树结构。

路段作为最基本的空间元素，是通过连接多个折点而形成的折线段，通过按序记录这些折点数据就可以对路段进行完整描述，因此若仅仅索引其 MBR 的坐标值，则将造成路段本身地理信息的丢失。由于 R*-tree 是将一个基本索引单元的数据封装在称为结点项（Entry）的数据结构内，而将数个结点项封装在固定空间大小的索引结点空间内，若将路段中各折点的坐标值作为路段属性索引时，不仅会增加 Entry 空间的大小，降低结点的空间利用率，且由于路段中折点数量不是固定值，无法确定一个 RR-tree 结点内可以包含结点项的数量，增加了索引的难度。

针对这一问题，RR-Tree 在建立树索引的同时，将当前路段中所包含的折点信息保存在一个单独的折点文件（PolylineFile）中，每一个折点的坐标值数据作为一条单独的记录。以路段中某一个路口点作为起始点，顺序写入，并将起始点位于折点文件的顺序编号与路段中的折点数作为路段的属性特征值。如图 3.3 所示。

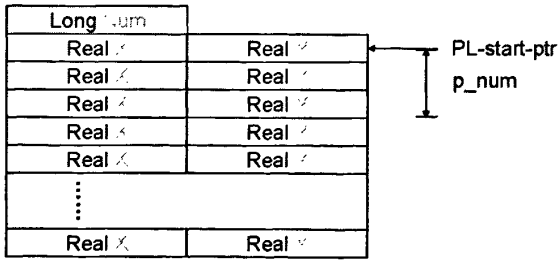


图 3.3 PolylineFile 的文件结构

通过以上设计，得到如图 3.4 所示的 RR-tree 结构。在 RR-tree 结构的下层，由灰色方块所标注的是经由 MBR 外包的路段数据，同时还记录有路段长度 seg-length 以及路段的起始折点在 PolylineFile 中的位置 PL-start-ptr 以及折点数量 p_num，以便在需要的时候可以获取其具体的路段信息。经过封装的路段数据即可同普通的包含有若干点数据的 MBR 一样，利用普通的 R-tree 结构加以索引，因此在 RR-tree 的上层就是普通的 R-tree 结构。通过 RR-tree 利用路网道路间的空间关系对各路段进行索引。不仅可以依据物体的地理位置通过不断减少候选区域的操作达到快速查询的目的，也可以在解决最短路径等问题时，将路网限定在一个较小的范围内，减少了无效的运算量，提高了查询效率。

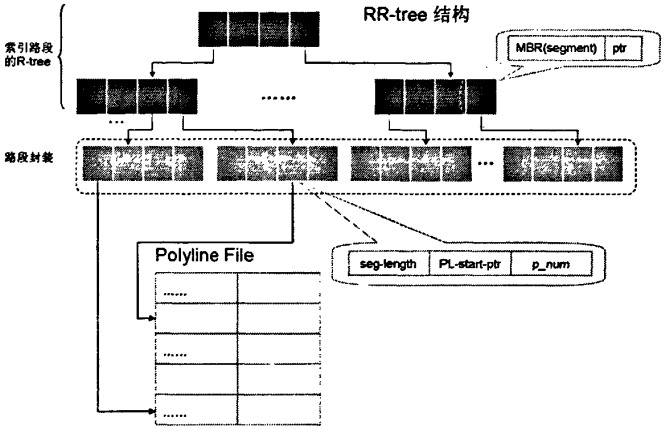
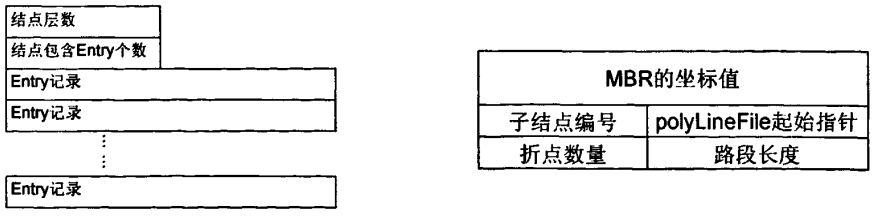


图 3.4 RR-tree 结构示意图

RR-Tree 是对 R-Tree 的改进，以便能够对具有不同数量折点的路段元素的空间信息加以索引管理，因此 RR-Tree 的各类操作，如查询、插入、分裂等算法均与 R-Tree 的操作算法相同。

同时，由于城市中道路数量极大，建构 RR-tree 所需时间也较长。但由于城市路网处于相对静止的状态，其索引结构不易发生改变。因此，若能利用合适的方法对 RR-tree 索引进行存储，就可在较短的时间内重构恢复，提高建构效率。



（a）Node　　（b）Entry

图 3.5 Node 与 Entry 的存储方式

如前所述，R-tree 中的每一个结点都存储在一块固定大小的磁盘空间内，并由此来确定结点中包含结点项的个数。其主要目的是用于提高结点间多组数据的传输速度。这样的数据块被称为索引存储块（Block），其结构如图 3.5（a）所示。其中包含的结点项存储方式如图 3.5（b）所示。由于在结点项中记录中的子结点编号指示了其子结点块的存储位置，而在存储索引存储块的文件 BlockFile 的首部，则存储有对文件信息进行说明的头数据块，在此块数据中指示了 RR-tree 根结点的存储编号。

因此，按照各结点存储编号的指示，递归读取各结点内容，就可以达到快速重构索引的目的。

3.2.2 路段连接算法

路段作为划分路网的基本单位，可以充分地降低索引结构的更新频率，提高索引结构的整体效率。但是在实际路网中，路段间存在一定的连接规律，相似的路段可以被连接以进一步延长数据更新的时间，降低对路网索引的查询频率。

A. Civilis 等人提出了一种用于对道路网改进和优化的算法思想：GSC（General Segment Connection Algorithm）算法[2]。利用 GSC 算法，可以对路网中需要改造的路段集合反复迭代排序，依据不同的相似性准则对路段进行排序，进而选择最合适路段与当前路段合并。在实现过程中，GSC 首先计算出所有与当前路段头尾有相连的路段，依据相似性准则选择最合适的路段延长当前路段，并将延长后的路段放入下次排序集合中，而把原有路段从集合中删除。若当前路段没有被延长，则将其放入结果集合，并不再用于延长。

由于在原有关于 GSC 算法的讨论中，对于路段的定义以及存储方法都与本文有较大的区别，因此本文基于 3.1 节中给出的路段定义以及 3.2.1 节中对于路段的存储方法，对 GSC 算法进行了一定的改造，形成 RGSC（RR-Tree based GSC）算法。对于 RGSC 的算法伪描述如算法 3.1 所述：

对算法所涉及的参数解释如下：

(1) Rn 是指优化前的 Segment 集合

(2) Cn 是指优化后的 Segment 集合

(3) Cand 是指与某个 Segment 的起始、终止点相交的 Segment 集合

(4) PL 是 Rn 排序后，存放 Rn 中第一个 Segment 的变量

(5) CPL 则是 Rn 排序后，存放 Rn 中第一个 Segment 的变量

(6) RnPrioritization 是针对 Rn 的排序算法

(7) CandPrioritization 则是针对 Cn 的排序算法

```
RGSC（Rn; RnPrioritization; CandPrioritization）
1. Cn ← 0
2. While rn != 0 do
3.     PL ←first(Rn; RnPrioritization)
4.     Rn ← Rn \{PL}
5.     Cand←0
6.     CPL ← 0
7.     For each pd ∈{start(PL), end(PL)}
8.         Cand ← {plc|plc∈Rn && (pd = = start(plc) || pd = = end(plc)) }
9.         If Cand != 0 then
10.            CPL←First(Cand; CandPrioritization; PL)
11.            PL ← PL extended with CPL
12.            Rn← (Rn \ {CPL})∪{ PL }
13.        Else
14.            Cn ← Cn∪{PL}
15. End While
16. Return Cn
```

算法 3.1: RGSC 算法伪代码描述

RGSC 算法是路段连接的通用算法，依据不同的连接策调用 RGSC，即按照不同的相似性准则对 Rn 和 Cand 进行排序，由此可以产生不同的路段连接结果。因此本文共实现并分析了两种具体的道路连接算法。

算法 3.1 的流程图如图 3.6 所示：

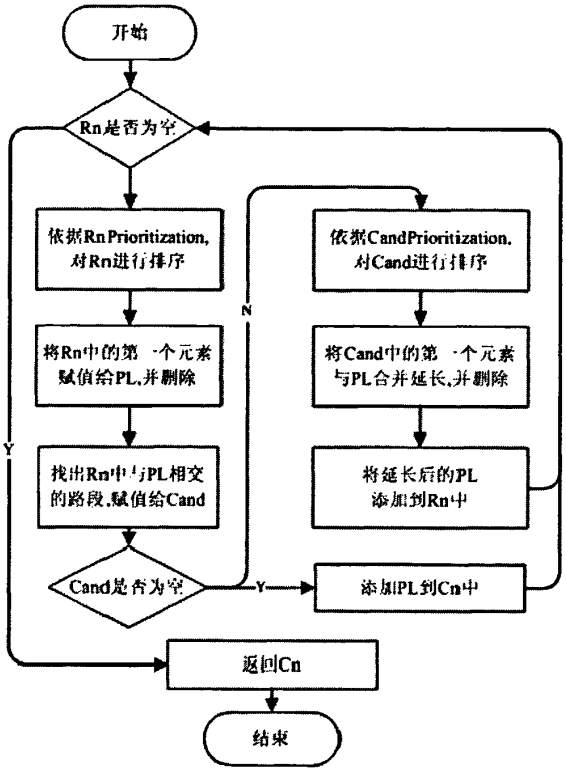


图 3.6 RGSC 的算法流程图

1. 主干道连接算法 LSC

主干道连接（Length and Side-Based Segment Connection，LSC）算法是在 RGSC 算法的基础上，通过对 RnPrioritization 和 CandPrioritization 这两个排序算法具体化后得到的道路网连接方法。

LSC 的主要思想是考虑到移动物体往往在主干道运动，以及从非主干道到主干道运动的可能性更大，并且主干道的 Length 和 Side 往往大于非主干道的，因此设计形成了 LSC 算法。

LSC 算法的表现形式为：

Cn ←RGSC（Cn; [LengthAsc]; [SidesDesc;LengthDesc]）

此处通过 Segment 的两个属性 Length 和 Side 作为排序的依据。Length 指 Segment 的长度，而 Side 则用于判断 Segment 两端与其他 Segment 邻接的状态，当路段两端均与其它路段邻接，其 Side 值为 2，当仅有一端邻接时，Side 值为 1，当两端均不邻接时，Side 值为 0。

在 LSC 中，[LengthAsc]用于表示 RnPrioritization 算法，即依据 Segment 的 Lenth 升序顺序来对 Rn 进行排序。[SidesDesc; LengthDesc]则表示先依据 Side 值

降序排列，再依据 Length 值降序排列。

2. 渐变道路连接算法 ASC

渐变道路连接（Angle_based Segment Connection，ASC）算法也是在 RGSC 算法的基础上，对 RnPrioritization 和 CandPrioritization 做具体化得到的道路网连接方法。

ASC 算法是在 LSC 算法的基础上，增加了新的排序指标，路段间的角度变化。通过分析实际的车辆通行数据可以得到，当车辆移动到道路口时，普遍选择倾角变化小的路段继续移动。在此基础上，本文设计 ASC 算法。

LSC 算法的表现形式为：

Cn←RGSC（rn;[AngleAvgAsc;LengthAsc];[SidesDesc;AngleAsc;LengthDesc]）

在 ASC 算法中，除了以 Length 和 Side 属性作为路段排序的依据外，又引入了两个属性：Angle 和 AngleAvg。

当路段的一个端点和其它路段相关时，此端点处的方向延长线与其相关路段会形成一个倾角，其中最小的倾角称为路段此端点处的 Angle。若路段某一端点与其它路段均不相关，则此端点的 Angle 默认为 180°。路段的 AngleAvg 则是指路段两端 Angle 的平均值。如图 3.7 所示。

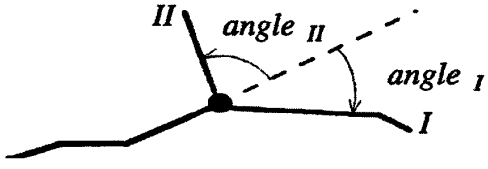


图 3.7 路段间角度

在 ASC 中，[AngleAvgAsc;LengthAsc]用于表示 RnPrioritization 算法，即对 Rn 先依据路段的 AngleAvg 属性升序排列，再依据 Length 属性降序排列。[SidesDesc; AngleAsc; LengthDesc]则表示对 Cand 先依据 Side 属性降序排列，再依据 Angle 降序排列，最后依据 Length 降序排列。

通过实验，可以证实以上三种方案 LSC 与 ASC 算法均可有效地对路段元素进行连接操作，提高了 RR-Tree 的查询效率，具体的实验结果可参见第五章的实验。鉴于实验结果，在后续的研究中，我们通常都用延长后的路段元素作为 RR-Tree 的基本索引单元。

3.3 基于路段的受限时空复合索引结构 NCO-Tree

时空数据库是 LBS 框架中的重要组成部分，包含多种数据资源，如电子地图的辅助数据、市政工程、气象信息等数据。其中最为重要的部分，是行驶在路网中的移动对象所持续提交的位置时空数据。LBS 系统以当前路网中的实时交通状况作为提供各项服务的基础，而掌握移动对象在路网中的分布情况，则是系统得以模拟当前路网状况的重要前提。

如 2.2.2 所分析，路网中移动对象的数量十分庞大，且更新频繁，因此一个合适的时空数据索引结构是时空数据库中不可或缺的部分。但在现有的商用数据库中，一般仅提供一维键值索引，或在面对由多列字段组成的键值时，只能采用级联索引的方式，这显然不能满足 LBS 对于时空索引的需求。当然也有部分商用数据库，如 Oracle 就提供了空间数据插件，能够支持对空间数据的索引需求。但这些插件中所提供的 R-Tree 或 Quad Tree 空间数据索引，仅适用于对静态的空间数据的高效查询。移动对象在路网中是不断移动的，其位置时空数据会随时间不断地发生变化，若利用静态的空间数据索引来管理动态的时空数据，就会带来更新效率低等诸多问题。因此，在时空数据库中建立一个合适的针对受限时空数据的时空索引结构是十分必要的。

通过 1.2 节对现有针对受限时空数据的索引结构做的简要介绍，不难看出复合索引结构是目前较为常见的针对实时受限时空数据的索引方式。如前文所述，复合索引结构由两部分组成：上层静态的空间索引与下层动态的时空索引集合。上层空间索引主要是通过将路网划分成若干个互相独立的子空间，并利用某种空间索引方法加以管理，而在每个子空间中都有一个时空数据索引来管理运动在当前子区域中的所有移动对象提交的时空数据，这些时空索引共同构成了复合索引的下层时空索引集。在复合索引中，当移动对象一直在路网的每个子空间中运动时，其位置数据则一直会由所在子空间的时空索引结构所管理。而当移动对象超出当前子空间的范围，则其数据就会从一个子空间的时空索引中转移到另一个子空间的时空索引中去。如果这种转移在移动对象行进的过程中频繁发生，由此带来的更新操作代价也是十分巨大的。

由此可见，如何对路网进行划分，是影响复合索引结构性能的重要因素。由于基于路段的追踪策略能够给索引结构带来最低密度的更新操作，因此在现有的复合索引结构中，以路段为基本单位对路网进行划分的方法是最为常见的。通过 2.3 节中对数种不同追踪策略的比较可以发现，在将路网划分为以路段为单位的子空间集合后，不论追踪系统采用何种追踪策略，除非移动对象经路口从一条路段中转移至另一条路段中时，才会向索引结构提交一次不同子空间的时空索引之

间的数据转移，否则对象或不提交更新操作，或提交的也只是同一时空索引下的更新操作。此外，由于每个时空索引结构只管理运动在其所在路段中的移动对象，而这些对象在路段中呈一维分布，可以提高时空索引结构的效率。因此，基于3.2节中所提出的RR-Tree结构，就可以构建出以路段为单位的复合索引结构，称为NCO-Tree，其具体结构如图3.8所示：

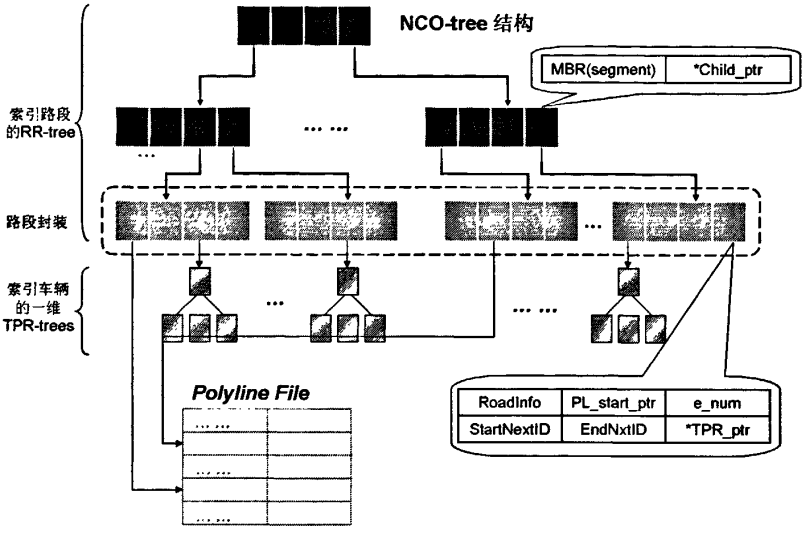


图3.8 NCO-Tree的结构

NCO-tree的上层是RR-tree结构，用于索引利用连接算法延长后的路网路段元素，所有路段元素都用包围RR-Tree的叶结点项中，用虚线矩形框内的灰色矩形块表示。通过对路网按路段单元进行划分后，分布在路网中的移动对象也被划分成若干个互不相交的子集，因此在NCO中，每个路段内都包含有一个用于管理路段中移动对象的索引结构。由于移动对象在单条路段中呈一维运动状态，本文选用一维的TPR-Tree对这些移动对象进行管理。基于此，在每个叶结点项中，都有一个指向一棵TPR-Tree的指针，用*TPR_ptr表示。所有路段中的一维TPR-Tree共同构成了NCO-Tree的下层动态时空索引集。

3.4 本章小结

本章为了能够以一致化的形式对路网数据进行描述，首先利用了细节化道路网模型和图论化道路网模型这两种不同的路网模型，对路网、路网中的各类空间元素以及元素间的拓扑关系做形式化定义。

与此同时，结合基于路段的追踪策略，本章选取路段作为复合索引结构中上层路网管理索引的基本索引单元。在此基础之上，提出了基于路段的路网数据索

引结构 RR-Tree，对其中的各类元素与结构进行描述，并提出路段连接算法对符合要求的路段集进行连接，以降低路段数量，提高 RR-Tree 的查询效率。

同时，本文又提出了以 RR-Tree 为上层结构的用于管理路网下时空数据的复合索引结构 NCO-Tree，并对其元素与结构进行描述。

第四章 基于路口区域的时空索引结构

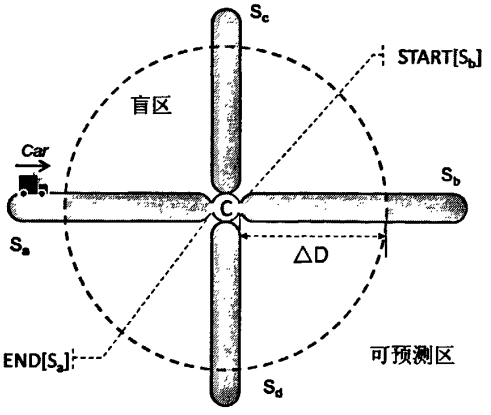
通过前文 3.3 节对基于路段的复合索引结构 NCO-Tree 的介绍，可见在现有结构中，通常是采用上述路网模型中的路段元素作为复合索引上层空间索引的最基本空间单元，并采用普通的空间索引方法加以管理。但这样一种划分方式与索引方法与现有的 LBS 环境并不匹配，其缺陷主要体现在索引不准确与更新效率低这两点上。因此在本节中，将主要对上述两个缺陷做详细的描述与分析。

4.1 基于路段的复合索引结构所存在的缺陷

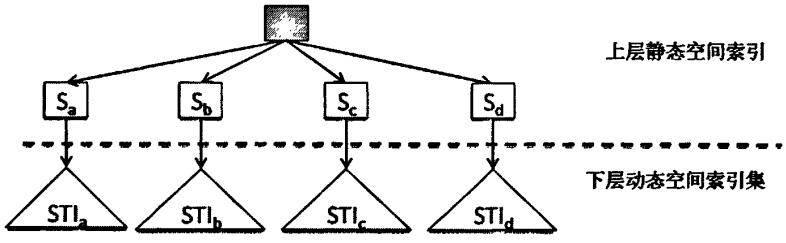
在 LBS 系统中，复合索引结构对移动对象位置的描述取决于追踪系统模块对移动对象的追踪定位。前文 2.2 节对不同的追踪策略进行了分析比较，并得出基于路段的追踪策略可以给复合索引结构带来最低频率的更新操作请求，有助于提高时空数据库的整体效率，因此在一般情况下，LBS 的追踪系统均采用路段追踪策略对移动对象进行追踪。

由于在现有的复合索引结构中通常也采用路段作为划分路网空间的基本单元，因此在移动对象进行更新操作时，对于操作的实时性要求很高。举例来说，图 4.1(a)中给出了一个简单的路网环境，此路网利用图论模型对其进行形式化描述，含有一个路口 c 和四个路段 S_a、S_b、S_c、S_d，c 与 4 四条路段均相关，即此 4 条路段均交汇于路口 c 处。针对此路网环境下的基于路段的复合索引结构如图 4.1(b)所示，路段作为独立的空间单元由上层静态空间索引结构管理，上层索引的每个叶结点均描述了路网中的一条路段，每个叶结点指向一个下层的动态时空数据索引，用于管理行驶在该路段中所有移动对象的时空数据。

现假设有一车辆 Car 从路段 S_a 的起始点出发，匀速始向 S_b 的终点。基于路段的追踪策略可知，当 Car 从 S_a 的起始点出发时，它会将自身的位置告知 LBS 中心服务器，并被告知其当前的预测范围是 S_a，即当车辆驶出 Sa 的范围后就需要向追踪系统提交更新，以便获取新的预测范围。同时中心服务器的复合索引结构会将 Car 的相关数据置于 S_a 的动态时空索引 STI_a 中。



(a) 路网环境



(b) 针对(a)中时空数据的基于路段的复合索引结构

图 4.1 基于路段的复合索引结构的缺陷

因此当 Car 驶过 c，到达 S_b 的起始点 START[S_b]时，就需要进行以下步骤：

(1) Car 进行定位操作，发现当前位置已经超出追踪系统的预测范围

(2) 提交更新请求，告知追踪系统 Car 的新位置。

(3) 追踪系统接收更新请求，更新时空数据索引，将 Car 的相关数据从 STI_a 转移至 STI_b 中。

(4) 追踪系统对 Car 做更新反馈，Car 被告知新的预测范围是 S_b。

以上操作需要被实时地完成，即当车辆一到达 START[S_b]处就需要完成这 4 个步骤，否则当车辆已越过 START[S_b]，但更新尚未完成时，在 LBS 系统对路网进行分析时，就会默认地将 Car 的位置置于 S_a 的终止点 END[S_a]处。这是因为当 Car 在路段 S_a 中时，追踪系统可以基于路段的特性系统确认 Car 的未来行驶方向在 t_p 时间内会被局限在 S_a 内，而不会转移到其它路段中去。其可预测时间 t_p 由以下公式决定：

$$t_p = \frac{Len(S_a) \times (1 - offset_{car})}{v_{car}} \qquad (式\ 4.1)$$

式中 $Len(S_a)$表示路段 S 的长度，$offset_{car}$ 是指 Car 在 S_a 中的偏移比例，v_{car}

是指 Car 的速度。当时间超出 Δt 后，系统根据已有数据可以判定 Car 已到达 END[S_a]，即路口 c 处，但由于 c 与 S_a、S_b、S_c、S_d 均相关，因此 Car 可以在到达 c 后进入 S_b、S_c、S_d 中的任意一条，也可折返再次进入 S_a 中。因此在时间超出 t_p 后，系统无法对 Car 的未来运动方向做出任何预测，因而只能简单地将 Car 的数据保留在 Sa 的动态时空索引中。在这种情况下，如果 LBS 服务器需要对当前路网状态进行分析，则只能认定 Car 当前是处于停留在路口 c 这样一个状态。

LBS 系统的建立信赖于定位技术和无线通讯技术的发展，但在目前科技发展水平的限制下，这两种技术均无法实现实时操作的可能。现有的 GPS 定位时间普遍需要 30 秒至 1 分钟的时间，而在现有的无线通讯技术中，也仅能提供最大 2M 的带宽速度。这样后台服务显然无法在 LBS 系统中实现实时的数据更新操作，而当 LBS 系统能够真实地表现 Car 的位置时，Car 已驶出了 ΔD 的距离，ΔD 由下列公式决定：

$$\Delta D = v_{car} \times \Delta t \qquad (式\ 4.2)$$

其中 Δt 是指 Car 完成了更新操作中的前三个步骤，即系统可以得到车辆提交更新时所处位置的所需时间，Δt 的值与各路口处定位信号的强弱、无线通讯状况的好坏有关。如果定位信号与无线通讯情况良好，则 Δt 较短，否则较长。Δt 越短，空间数据库对移动对象的记录越准确。

由于 ΔD 的存在，会在路网中形成以路口 c 为中心，ΔD 为半径的圆形区域，如图 4.1(a)中虚线圆形区域所表示的，这个区域被称为 Car 的盲区（Blind Region，简称 BR）。因为当 $t_p \leqslant t \leqslant t_p+\Delta t$ 时，Car 可能处于盲区内的任意路段中，因此系统无法正确判定 Car 的准确位置。而在此区域之外的路段范围则被称为 Car 的可预测区，在此区域之内，Car 的位置可依据其所提供的更新数据进行推算。

在实际道路网环境下，同一时刻内通过某一路口的车辆会达到数十辆之多[1]。由于盲区的存在，使得 LBS 系统无法准确地描述这些车辆的准确位置。因此在进行路网分析时，系统会认为有数十辆车均滞留在同一路口时，进而会造成对道路交通状态的错误判断，并提供错误的 LBS，造成严重的后果。

在路网的图论化模型中，各路段之间存在路段邻接关系。如在图 3.1(c)中，$S_{6\text{-}7}$ 与 $S_{7\text{-}8}$、$S_{7\text{-}10}$ 通过路口 C_7 而相连，因此当车辆从 $S_{6\text{-}7}$ 抵达 C_7 路口时，它只能选择 $S_{7\text{-}8}$、$S_{7\text{-}10}$ 中的一条而继续前行。因此在 LBS 系统对车辆进行定位时，只需对 $S_{7\text{-}8}$ 和 $S_{7\text{-}10}$ 进行空间查询即可。但是在现有的复合索引中，其上层的路网索引视所有的路段为独立的空间单元，并依据路段间的空间距离远近加以聚集索引，而没有对路段间的路段邻接关系进行存储。这一缺陷使得车辆在进行定位操作时，就需要对整个路网下的所有路段依据空间距离的关系进行查询，扩大了定位

操作的空间查询范围。

通过 2.1.2.1 节中对于空间定位技术的介绍可知，现有的空间定位技术的最高精确度在 20-30 米左右。而在市区环境下，路网分布密度高，同时由于各类高大建筑物的遮蔽，定位信号较弱，因此采用扩大化的定位查询将导致定位不准确等问题，影响 LBS 的正常功能。

通过以上分析可以得到，由于在现有的复合索引中采用路段作为路网的基本划分单元，使得车辆在驶入盲区后，系统无法对其位置做出准确的判断，造成了索引的不准确。同时由于索引结构没有对路段间的路段邻接关系进行存储，因此需要进行扩大化且不精确的定位操作，降低了路网的更新效率。因此在对复合索引的研究中，需要提出新的路网单元来代替路段单元，并构建一个新的复合索引结构。

4.2 路口区域 Cross Region（CR）

通过上述的分析可以得出，现有的复合索引之所以会存在索引不准确和更新效率低两大不足之处，就是因为采用路段作为路网索引的基本单元。为了解决这些问题，本文提出了一种新的路网单元，称为路口区域（Cross Region），简称 CR。

CR 是路口的扩展区域，它由两部分组成：一个路口和与该路口相关的所有路段的部分。如图 4.2 所示：

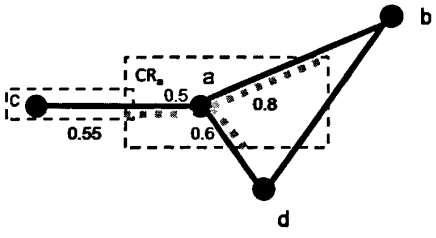


图 4.2 路网中 CR 的覆盖范围及包含的空间元素

图 4.2 中给了一个简单的路网，它由四个路口 a、b、c、d 和四条路段 S_{ab}、S_{ac}、S_{ad}、S_{db} 组成。其中关于 a 的 CR 由以下两部分组成：a 和与 a 相关的所有路段，即 S_{ab}、S_{ac}、S_{ad} 的部分，整个路口区域用 CR_a 表示。CR 所包含的路口称为 CR 的内含路口，所有与 CR 内含路口邻接的路口则称为 CR 的外接路口。如图中，a 是 CR_a 的内含路口，而所有与 a 存在路口邻接关系的路口 b、c、d 则是 CR_a 的外接路口。CR 的内含路口与外接路口间的路段部分称为 CR 的外接路段，

外接路段占对应路段的比例称为外接比例。如图 S_{ab}、S_{ac}、S_{ad} 中用灰色虚线表示的部分即是 CR 的外接路段，而旁边的小数则表示 CR_a 的外接比例。覆盖了所有 CR_a 相关路段的黑色虚线框表示关于 CR_a 的最小外包矩形框（MBR）。需要注意的时，尽管 S_{db} 的部分也被 CR_a 的 MBR 所覆盖，但这部分路分路段并不包含在 CR_a 内，因为 S_{db} 与 a 不相关。同一路段两端的 CR 关于该路段的外接比例和必须大于 1，即这两个 CR 必须可以覆盖整个路段，如图 4.2 中 CR_c 与 CR_a 关于 S_{ac} 的外接比例和为 $1.05 > 1$，因此 CR_c 与 CR_a 可以完整地覆盖路段 S_{ac}。

下面给出路口区域在图论化路网模型中的定义。

定义 4.1：路口区域（Cross Region）

路口区域是路口的扩展区域，它由 GRN 中的一个路口以及该路口所有相关路段的部分组成。邻接路口的路口区域可以覆盖这两个路口间的路段。

定义 4.2：内含路口（Internal Cross）

CR 内部所包含的路口。

定义 4.3：外接路口（External Cross）

所有与 CR 内含路口所邻接的路口。

定义 4.4：外接路段（External Segment）

所有与 CR 内含路口相关的路段部分。

定义 4.5：外接比例（External Proportion）

CR 的外接路段与对应 CR 内含路口相关路段的比例。

定义 4.6：路口区域邻接（CR Neighboring Relationship）

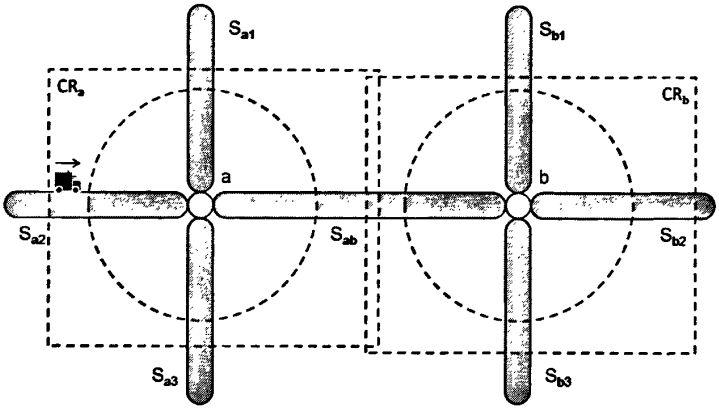
两个路口区域邻接当且仅当它们的内含路口邻接。

定义 4.7：路口区域的度（Degree of CR）

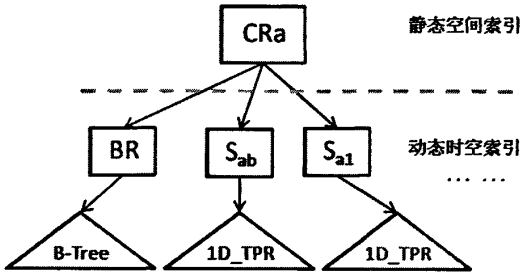
CR 的度是指其所包含的路口的度。

4.3 CR 内部移动对象的索引

采用路口区域来作为划分路网的基本单元，并建立对应的复合结构可以有效地解决现有复合索引结构中的索引不准确的问题。本节将着重描述以 CR 为基本单元的复合索引结构的下层动态索引集部分，并对如何利用 CR 来解决索引不准确性的问题进行分析。如图 4.3：



(a) 路网模型



(b) CR 下的时空索引结构

图 4.3 对于 CR 的实例分析

在图 4.3(a)中给出的简单路网环境中，有两个路口：a 和 b，以及各自的相关路段：S_{ab}、S_{a1}、S_{a2}、S_{a3}、S_{b1}、S_{b2}、S_{b3}。围绕 a、b 形成两个路口区域 CR_a 和 CR_b，其 MBR 分别用虚线矩形框表示。围绕在 a 和 b 周围的虚线圆形区域表示车辆经过这两个路口的盲区，各 CR 的 MBR 以内，盲区以外的区域均是可预测区。

图 4.3(b)所给出的是对所有在 CR_a 中运动的移动对象进行动态时空索引的结构。其中的黑色矩形表示在复合索引上层的空间索引中描述 CR_a 的结点项，该结点项包含了指向 CR_a 中移动对象时空索引结构的指针。CR_a 的内部索引部分包含了 n+1 个索引结构，其中 n 是指 CR 的外接路段数量，此处为 4，这 n 个索引结构均是 1 维的动态 TPR-Tree 结构，用于管理行驶在某一 CR 外接路段中的移动对象，而另一个索引结构则是一个依据车辆编号进行索引的 B-Tree 结构，用于管理所有进入 CR 内含路口盲区的移动对象。

在对 CR 下的动态索引结构进行描述之后，以下将通过实例来分析如何利用 CR 来解决索引不准确问题。假设现有车辆 Car 需要从路段 S_{a2} 出发，穿过路口 a 和 b，驶入 S_{ab}，并到达 S_{b2}。

由于在初始时刻 Car 进入了 CR_a 的范围内，且行驶在路段 S_{a2} 中，因此其位置数据将被置于 CR_a 中 S_{a2} 索引中。在 Car 的行驶过程中，会依次发生以下数据更新操作：

(1) 在系统侦测到 Car 的位置已超出 S_{a2} 的范围，但尚未接收到 Car 的更新请求时，就会自动地将 Car 的数据转移至 CR_a 中 BR 的索引中去。

(2) 当接收到 Car 的更新请求，得知 Car 已经处于 S_{ab} 中时，系统则将其数据从 CR_a 的 BR 的索引中移至 CR_a 中 S_{ab} 的索引内。

(3) 当系统侦测到 Car 的位置已超出 CR_a 的范围时，由于 Car 行驶在 S_{ab} 中，只能前往 CR_b，因此系统可自动地将 Car 的数据转移到 CR_b 中 S_{ab} 的索引中去。

(4) 在系统侦测到 Car 的位置已超出 S_{ab} 的范围，但尚未接收到 Car 的更新请求时，就会自动地将 Car 的数据转移至 CR_b 中 BR 的索引中去。

(5) 当接收到 Car 的更新请求，得知 Car 已经处于 S_{b2} 中时，系统则将其数据从 CR_b 的 BR 的索引中移至 CR_b 中 S_{b2} 的索引内。

在以上 5 步更新操作中可以发现，只有第 3 步的更新操作涉及到 CR 间的数据转移，而其它 4 步更新则均是同一 CR 内部不同索引结构间的数据转移。原复合索引中的索引不准确性是由于移动对象进入盲区，其更新操作请求无法实时送达索引结构造成的。在引入 CR 概念后，当车辆进入路口盲区时，其数据会被转移至该路口 CR 下的盲区索引内。因此在进行路况分析时，系统可以统计到在当前时刻下该 CR 盲区内的车辆数量，并可按照 CR 内不同路段中的车辆密度来对路段中的车辆数量进行估计，而非简单地将进入盲区的车辆数据滞留在路口处，造成路况侦测的不准确。

4.4 基于路口区域的路网下时空索引 CR-Tree

4.4.1 CR-Tree 元素和结构

本文在前述中对路口区域 CR 进行描述分析，指出利用 CR 对路网进行划分后，可以解决现有基于路段的复合索引结构中索引不准确的问题，并对单一 CR 内的时空索引结构做了介绍。

为了能够建立基于 CR 的复合索引结构，就需要建立一个以 CR 为基本索引单元的用于管理路网的静态空间索引结构以作为复合索引的上层结构。由于在现有的静态空间索引中，R*-Tree 是应用最为广泛的索引方法，因此本文将依据 CR 的特性，利用改进的 R*-Tree 索引方法来建立基于 CR 的静态空间索引：CR-Tree。

CR-Tree 与 R*-Tree 在结构上类似，同属于 B-Tree 在多维空间中的自然扩展。

但由于 CR 间存在路口区域邻接关系，若将这种邻接关系存储在 CR-Tree 中，可以缩小移动对象空间定位的查询范围，以有效地解决原有基于路段的复合索引中更新效率低的问题。因此本文对 R*-Tree 的索引结构与索引方法进行改进，使之更适用于存储与管理 CR。

类似于 R*-Tree，CR-Tree 包含两类基本的数据类型：结点项（Entry）与结点（Node）。结点项是对基本空间元素或子结点的描述，结点内包含了多个结点项间，并对它们之间的关系进行描述。下面将分别对这两类元素进行描述。

叶子结点项（Leaf Entry）：当结点的层数为 0，即结点为 CR-Tree 的叶子结点时，其所包含的结点项则被称为叶子结点项，简称为 LE，用于记录基本的空间数据单元 CR。基于上文对 CR 的描述，CR-Tree 中的叶子结点项由下述四元组构成：

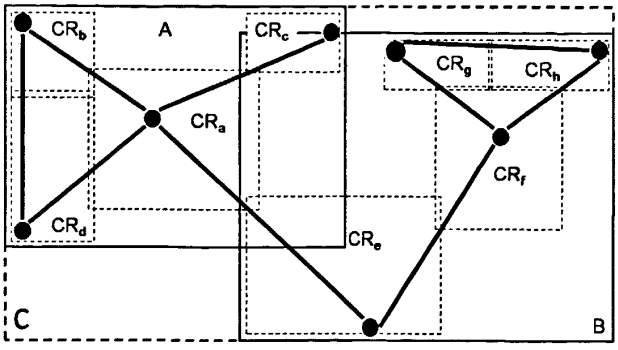
$$< cID, ecInfo\_Array, MBR, *cPtr > \qquad (式\ 4.3)$$

其中，cID 是指该叶子结点项对描述 CR 的编号，由于每个 CR 包含且仅包含一个路口，因此通常情况下也常用 CR 的内含路口编号来代替该 CR 的编号。如对路口 a 的路口区域 CR_a，CR_a.cID = a。ecInfo 是 CR 的外接队列，它包含有（ecID，ep）两元组，其中 ecID 指 CR 外接的某个路口编号，ep 则是 CR 内含路口与该外接路口间的外接比例。CR 的所有外接路口信息共同构成了 ecInfo_Array。MBR 则是指当前 CR 的最小外包矩形。*cPtr 是指向 CR 内部移动对象索引的指针。

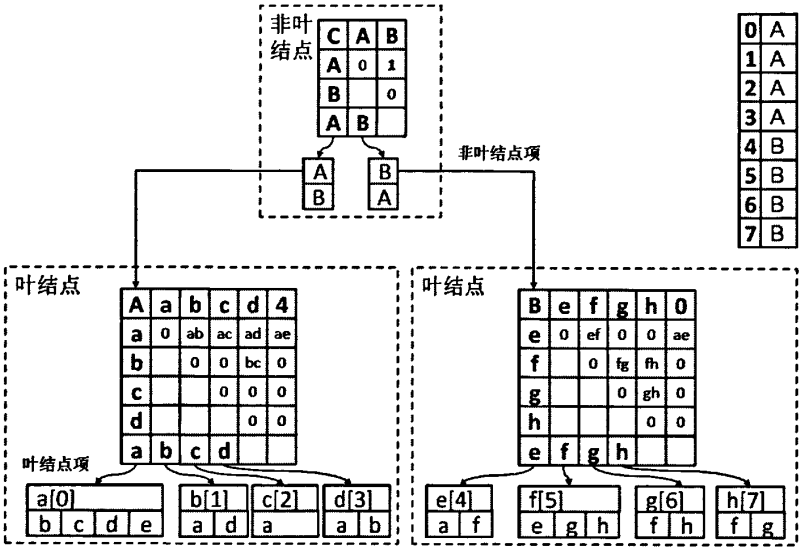
叶子结点（Leaf Node）：叶子结点包含了若干个 LE，即 CR，简称为 LN。所有被包含在当前 LN 内部的 LE 被称为该 LN 的内部结点项（Internal Entry，简称 IE），而所有与 IE 存在邻接关系的但并不存储在当前结点中的结点项则被称为该结点的外接结点项（External Entry，简称 EE）。CR-Tree 采用 R*-Tree 的索引方法来衡量 CR 间的距离远近。存储在同一叶子结点内的 CR 除了在彼此距离上要相对靠近外，由于 CR-Tree 需要记录 CR 间的邻接关系，因此特别要求同一结点的 IE 需要能构成连通图，即任一 IE 都能通过结点内 IE 的相关路段到达其它 IE。叶子结点的由下述三元组构成：

$$< nID, LE\_Array, \text{adjacency\_List}, MBR > \qquad (式\ 4.4)$$

其中，nID 表示该叶子结点的标号，leafEntry_Array 记录了 IE 的数据，MBR 是指能够覆盖结点所有 IE 的 MBR 的最小外包矩形。adjacency_List 用于记录 CR 间的邻接关系，其中不仅记录了 IE 间的邻接关系，也记录了 IE 与 EE 间的邻接关系。



(a) 基于 CR 进行划分的图论化图论模型



(b) 对(a)中路网进行索引后得到的 CR-Tree

图 4.4 CR-Tree 的实例

非叶结点项（NonLeaf Entry）：非叶结点项是对 CR-Tree 中子结点的描述，子结点的层数大于等于零，简称 NLE。NLE 相当于扩展的 CR，其三元组同式 3.3，但其中的 cID 是指其子结点编号，且在其 ecInfo_Array 中，也仅记录与其子结点邻接且层数相同的其它子结点的结点项，*cPtr 是指向子结点的指针。

非叶结点（NonLeaf Node）：非叶结点包含了若干项 NLE，简称 NLN。包含在 NLN 内的 NLE 也被称为内部结点项（Internal Entry，简称 IE），而其它与 IE 存在邻接关系但不存储在该 NLN 内的同层 NLE 则被称为外接项（External Entry，简称 EE）。NLN 的三元组同式 3.4，其 adjacency_List 中也同时记录了 IE 之间以及 IE 与 EE 之间的邻接关系。

关于 CR-Tree 的实例如图 4.4。

图 4.4(a)给出了一个简单的基于 CR 进行划分的道路网模型。它包含 8 个路口：a、b、c、d、e、f、g、h，和 9 条路段：S_{ab}、S_{ac}、S_{ad}、S_{ae}、S_{bc}、S_{ef}、S_{fg}、S_{fh}、S_{gh}。在基于 CR 对路网进行划分后，可以对应得到 8 个 CR，为了便于表述，这些 CR 的编号与其内含路口标识相同，如 CR_a 即用 a 来标识。

在利用 CR-Tree 对图 4.4（a）的 CR 进行索引后，可以得到如图 4.4（b）的 CR-Tree。此结构中包含两个叶结点 A 和 B，一个非叶结点 C。在 A 中包含有 4 个 LE，即 4 个 ICR：a、b、c、d，并用邻接矩阵的形式描述了这 4 个 ICR 与 ECR e 的邻接关系。而在非叶结点 C 中，包含了 2 个 NLE：A 和 B。由于 C 没有与 A、B 对应的 ENLE，因此其邻接矩阵中只描述了 A 和 B 的邻接关系。

在对 CR-Tree 的实验中发现，尽管通过 CR-Tree 中 LN 的 adjacency_List 记录了各 CR 间的邻接关系，当两个 CR 处于相同的叶结点内时，可以有效地缩小定位查询的范围，但当两个 CR 处于不同的叶结点内时，则仍需要以全体 CR 为查询范围进行查询。为了解决这个问题，本文建立了 CR 注册表机制，如图 4.4（b）中右侧所示。每个 CR 均在注册表中记录下包含此 CR 的叶结点指针，并用注册表中的记录偏移量对 CR 进行区分。而 CR-Tree 叶结点的 adjacency_List 中，对于 ECR 则不记录其标号，而记录其在注册表中的偏移量。在这种机制下，系统就可以通过注册表来直接查询 ECR 所在的叶结点。

4.4.2 确定结点内的结点项上限

为了保证时空索引的快速存取，索引中的每个结点均占据相同的磁盘空间，且结点内结点项的数量也是基本固定的。在 R*-Tree 中，通过设置每个结点项内的最小结点项数值 m 和最大结点项数值 M 来确保满足上述条件。若结点内的结点项数量小于 m 时，则该结点会被合并到其它结点中去；当结点项数量大于 M 时，则对当前结点进行分裂操作，由此可以保证整个索引结构的平衡性，达到快速存取的目的。

由上述可见，对于时空索引而言，设置结点内结点项数量的上下限非常重要。一般情况下，下限值 m 的确定可由人为确定，而上限值 M 则由式 3.5 取得：

$$M = \frac{pageSize - nodeAttriSize}{sigEntrySize} \qquad \text{（式 4.5）}$$

其中 pageSize 是指每个结点所需占据的固定磁盘页空间大小，nodeAttriSize 是指对结点进行描述的属性数据所需占据的空间大小，而 sigEntrySize 是指一个单独的结点项所需占据的空间大小。

在 CR-Tree 中，由于结点项内需要记录外接结点项的标识，因此每个

sigEntrySize 的大小都会随着其外接结点项的多少而存在变化，同时结点内也需要对外接结点项与内部结点项之间的邻接关系进行记录，因此 nodeAttriSize 也会随着外接结点项的多少而变化。由此可见，CR-Tree 结点的结点项上限值 M 由其具体所含的结点项所决定，而非常量，因此结点内结点项数量上下限的确定就成了 CR-Tree 中难点问题。

本文中解决此问题的方法是对结点的外接结点项，即 EE 的数量做出限制，要求 EE 的数量不得超出 IE 的数量。假设某个结点内包含有 k 个内部结点项 IE，则此 k 个 IE 的 EE 的总数（允许对同一 EE 作重复计数，因为它会被重复地记录在不同 IE 的外接队列中）不能超过 k。由此得到的上限值 M 的计算公式为：

$$M = \sqrt{PageSize - nodeAttriSize + \frac{(entryAttriSize + sigEcAttriSize)^2}{4}} - \frac{(entryAttriSize + sigEcAttriSize)}{2} \quad \text{（式 4.6）}$$

其中 PageSize 指固定的磁盘页空间的大小，nodeAttriSize 是指结点内属性数据的大小，entryAttriSize 是指结点项内属性数据的大小，sigEcAttriSize 是指每个结点项对其外接结点项的属性描述数据大小。此公式由下列过程推导：

假设某磁盘页中放置了 M 个 IE，则按照要求，整个结点中所含 IE 的 EE 数量至多为 M。假设在结点中所有数据都采用大小一致的数据类型，可得结点中对 M 个 IE 的描述需要数据空间为 M * entryAttriSize，由于至多存在 M 个 EE，因此对 M 个 EE 的描述需要数据空间为 M * sigEcAttriSize，为了存储这 M 个 IE 之间以及 M 个 IE 与 M 个 EE 之间的邻接关系，共需要(2M * M) / 2 = M^2 的数据空间，因此为了存储这个结点共需要磁盘空间为：

$$M^2 + (entryAttri\ Size + sigEcAttri\ Size) \times M + nodeAttriS\ ize$$

$$= (M + \frac{(entryAttriSize + sigEcAttriSize)}{2})^2 - \frac{(entryAttriSize + sigEcAttriSize)^2}{4} + nodeAttriSize$$

$$= PageSize \quad \text{（式 4.7）}$$

从式 4.7 可以推算出同一结点内结点项的数量上限 M，即为式 4.6。

4.4.4 查询操作

查询操作用于对移动对象进行定位，以确定移动对象当前所处的具体 CR 区域。查询操作依据是否存在前置 CR 而分为两类：初始化定位和移动定位。初始化定位发生在车辆对 LBS 系统提交第一次更新请求时，CR-Tree 需要依据车辆所

提交的位置数据对全体进行空间查询定位。但是在后续定位中，由于 LBS 系统已获取车辆在上一时刻提交更新时所处的 CR，即前置 CR 时，因此可以利用 CR-Tree 中叶结点中所存储的 CR 邻接关系表以及 CR 注册表来缩小车辆的定位查询范围，并依据车辆的具体位置来最终确定结果。具体的查询操作算法伪代码如算法 4.1 所述。其中 T 表示被查询的 CR-Tree，tcr 表示目标 CR，而 ln 则返回包含 tcr 的叶结点，并置返回值为 True，若 T 中不包含 tcr，则返回值置为 False。

```
SEARCH CR_Search(T, tcr, *ln)
1.  If (T.level ≠ 0)
2.      For each subNode in T.IE_Array
3.          If (subNode.MBR overlap tcr.MBR)
4.              If CR_Search(subNode, tcr, ln)
5.                  Return True
6.  Else
7.      For each ICR in T.ICR_Array
8.          If tcr.ID = = ICR.ID
9.              ln = &Node
10.             Return True
```

算法 4.1：CR-Tree 查询操作伪代码

4.4.3 插入操作

插入操作是构建 CR-Tree 的基本方法，在对道路网依据 CR 进行划分后，通过向空的 CR-Tree 逐个插入路网中 CR 的方式可以获取关于目标路网的 CR-Tree 索引结构。

```
INSERT ChooseLeafNode(T, ncr, *ln)
1. Find all adjacent CRs of ncr into aCRs
2. For each aCR in aCRs
3.     Invoke CR_Search to fine the leafNode contains aCR pointed by Ln_aCR
4.     Calculate the ratio r_aCR of the Length of the segment between aCR and ncr
       to Ln_aCR.MBR's diagonal
5. Return the Ln_aCR with a minimum r_aCR
```

算法 4.2 CR-Tree 查询可插入新 CR 叶结点操作伪代码

在插入过程中，除了要保证同一结点内各结点项间的距离相近外，还需保证这些结点项能够形成连通图：在 LN 中任一 ICR 都能通过结点内 ICR 的相关路段到达其它 ICR，在 NLN 中任一 INLE 都能通过结点内 ICR 的相关路段到达其它 INLE。因此首先设计了 ChooseLeafNode 算法以查询到可以添加目标 CR 的叶结点。ChooseLeafNode 算法伪代码出算法 4.2 描述，其中 T 代表需要被插入的 CR-Tree，ncr 表示新的 CR，*ln 返回可以添加 ncr 的叶结点。

在选择了合适的叶结点后，算法将新的 CR 置入，若插入操作导致叶结点的溢出，则分裂当前结点。插入算法的伪代码描述如算法 4.3 所述。

```
INSERT CR_Insert(T, ncr)
1. Invoke Choose to select a suitable leaf node L to put ncr in
2. If (L overflow after insert ncr)
3.       Invoke Node_Split to split L into two new leaf node
```

算法 4.3 CR-Tree 插入操作伪代码

4.4.5 分裂操作

在 CR-Tree 的构建过程中，当同一结点内的结点项数量超过上限值 M 时，就需要对当前结点进行分裂，以确保 CR-Tree 的平衡性，达到快速存取的目的。

在对结点内结点项间的空间划分上，CR-Tree 利用与 R*-Tree 相同的划分方法，但受到 CR-Tree 的特性要求，同一结点内的结点项应能形成连通图，因此需要对 R*-Tree 的分裂算法进行改进。改进方法是：首先将一个 CR-Tree 结点内的结点项按照空间距离划分成两个数量相近的子集，然后再对每个子集内的结点项进行检测，若某一结点项无法与同一子集内的其它结点项存在邻接关系，则将此结点项置于另一子集内。分裂操作伪代码描述如算法 4.4 所述，其中 node 代表需要被分裂的 CR-Tree 结点。

```
SPLIT Node_Split(*node)
1. invoke the R*-Tree method to split the entry-Array into two entry sets
2. For each divided entry set
3.   For each entry inside the set
4.       If entry is isolated in present set
5.           move the entry to another set
6. Create a new node NN to contain one entry set
7. Delete N's entries inside NN
```

算法 4.4 CR-Tree 分裂操作伪代码

4.5 本章小结

本章对基于路段的路网划分方法进行分析，指出在 LBS 的非实时环境下，基于路段的受限时空复合索引结构会存在索引不准确以及更新效率低的问题，并着重对这两个缺点进行了描述与分析。基于此原因，本章提出了一种新的路网元素路口区域（CR），并对 CR 以及 CR 间的拓扑关系进行形式化定义。通过对 CR 内部移动对象索引结构的描述与分析，得出基于 CR 的路网划分方式可以解决原有的索引不准确的问题。

在此基础之上，本章提出了基于 CR 的路网索引结构 CR-Tree，并对其中的各类元素以及操作进行描述。通过建立 CR 注册表，可以解决原有的更新效率低的问题。同时给出了固定大小的结点内结点项的数量上限计算公式。

第五章 实验分析

前文主要对LBS框架内的中心服务器部分的时空数据索引技术进行了讨论，其研究对象是行驶在路网中的移动对象所提交的实时受限时空数据，索引方法是复合索引结构。讨论的焦点主要集中在路网的划分方式上，不同的划分方式会形成不同的复合索引结构的上层索引，进而对整体索引结构的效率产生较大的影响。主要的划分方式有两种，一种是基于路段的路网索引方式RR-Tree，另一种是基于路口区域的路网索引方式CR-Tree。在本章中，将主要对这两种不同的路网划分方式的性能进行比较，同时也会对本文中所提到的基本理论进行实验验证。

本文的实验部分是在VC++.NET下开发完成的，采用的实验环境具有Intel P4 1.7G的CPU和640M的内存。

采用的实验数据则分为两部分：即城市路网数据与移动车辆数据。路网数据采用Geographic Data Technology Inc 1999[2]年根据美国加利福尼亚州公共交通网所采集的数据生成，路网如图5.1所例。经测算，加州公共交通网中平均道路密度在2.14 km/km^{2}左右。由于整个路网集非常庞大，为满足实验需要，从中选取了多组数量不同的部分路网供实验比较使用。各组的路段数量从1000至50000不等。移动车辆的行驶位置数据则由移动对象生成平台生成。同样，为满足实验需要，依据不同的路网规模生成不同数量的移动对象，其数量从200到50000不等。另外，为了比较不同路网规模、移动对象密度对索引建立性能的影响，我们另选择了模拟路网作为试验依据，数据内容在具体试验中介绍。

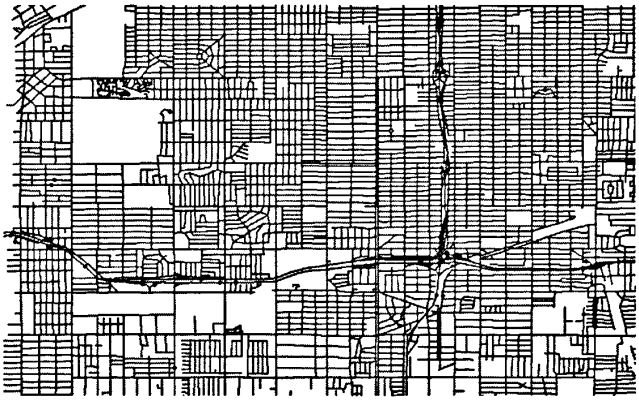
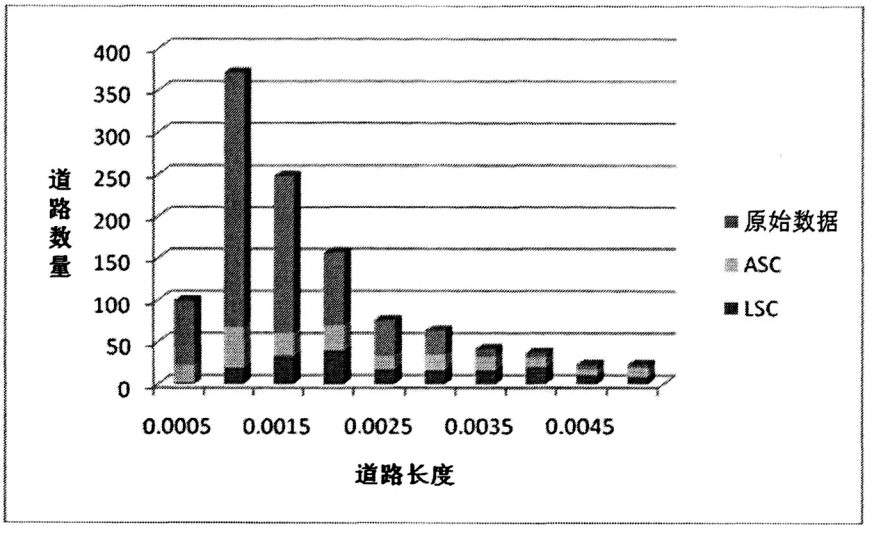


图5.1 实验用道路网环境示例

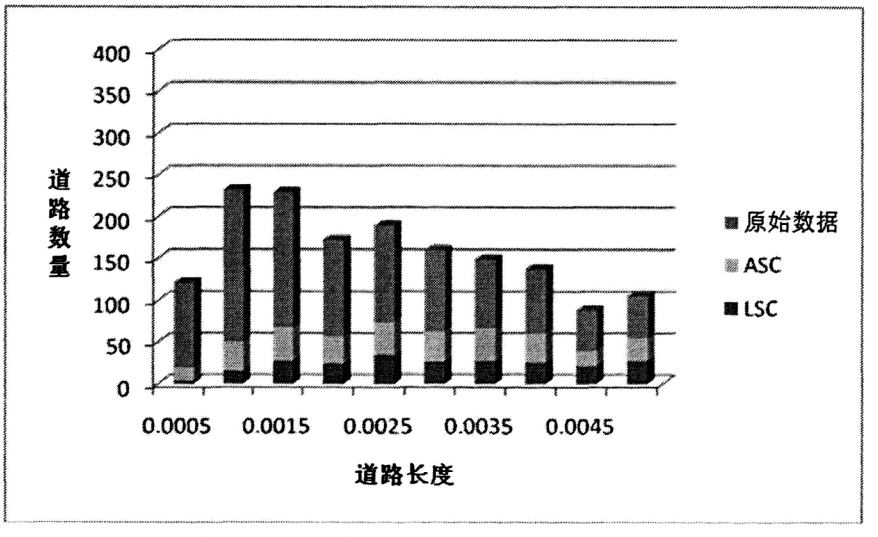
[2]注：http://libraries.mit.edu/gis/data/findingaids/esridatamaps1999.pdf

5.1 路段连接

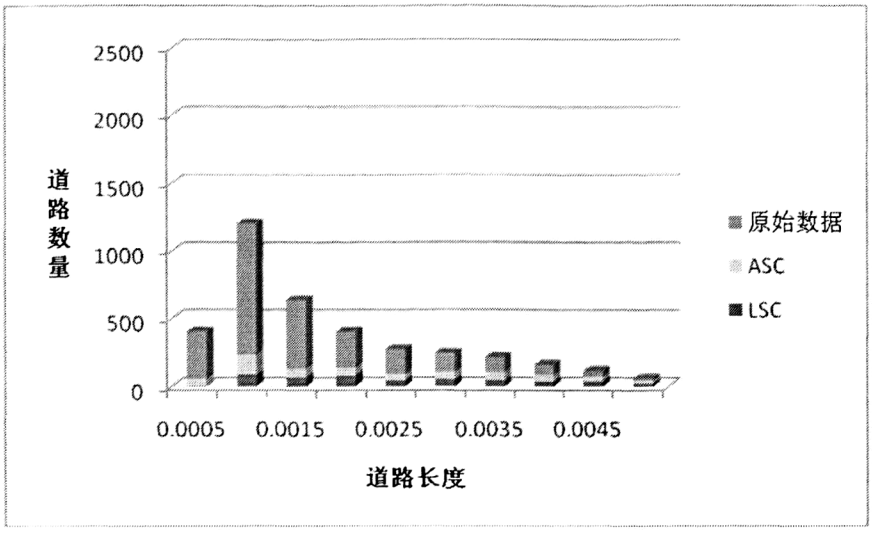
在3.2.2节中，针对路网中的路段单元，基于改进的通用道路连接算法RGSC，本文提出了LSC与ASC这两种道路连接算法，这两个算法可以依据一定规律对路段进行连接操作，以缩小路网索引的规模，提高查询效率。对于LSC与ASC算法对路网的改进程度的考察需要通过实验来进行验证。



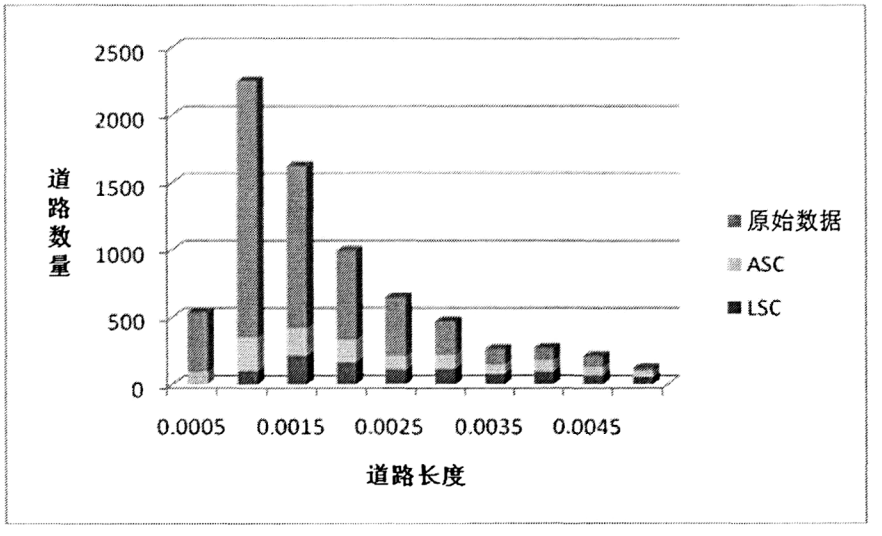
（a）对初始包含749条路段的路网做连接操作后的道路长度分布



（b）对初始包含1237条路段的路网做连接操作后的道路长度分布

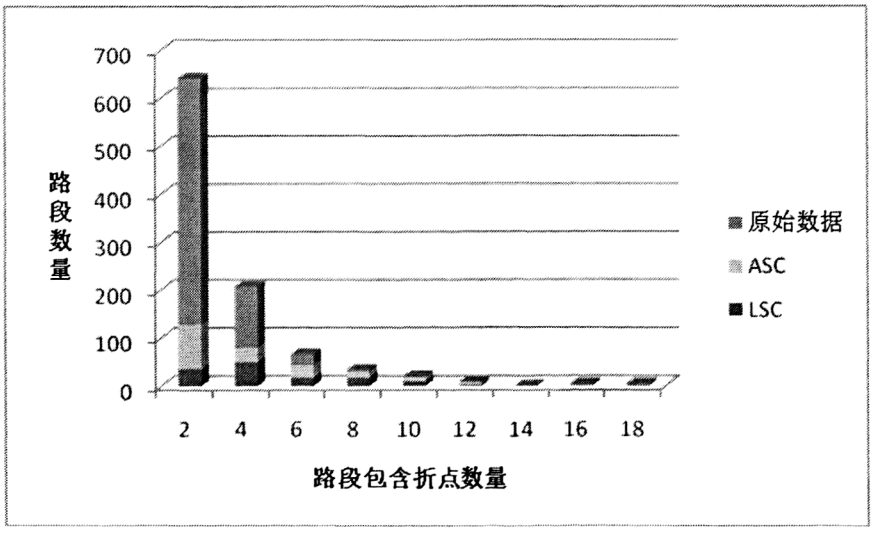


（c）对初始包含 2685 条路段的路网做连接操作后的道路长度分布

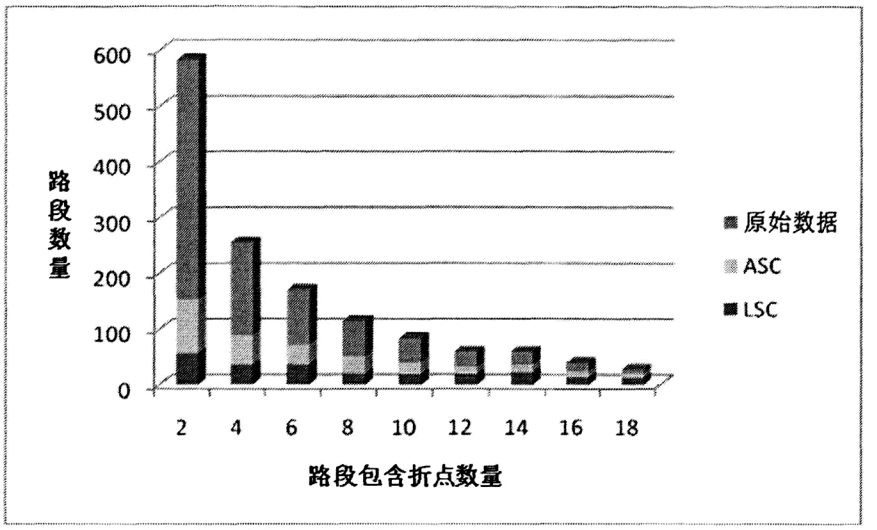


（d）对初始包含 5250 条路段的路网做连接操作后的道路长度分布

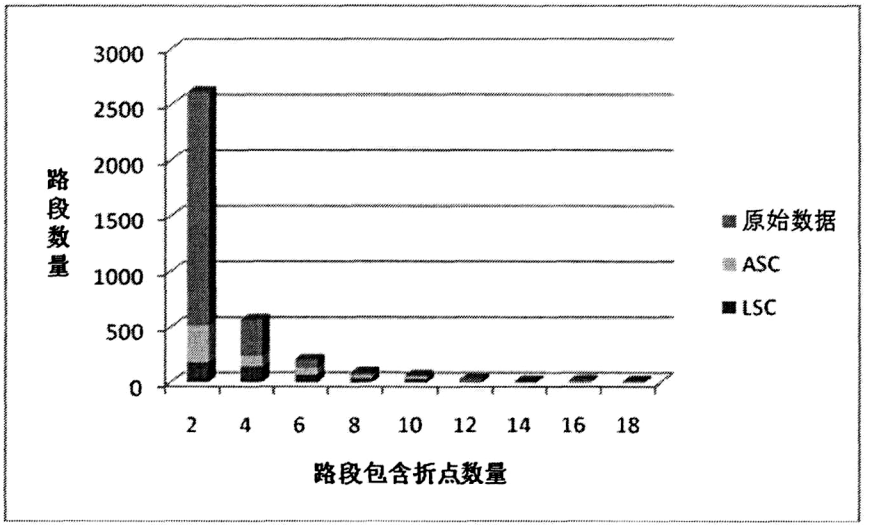
图 5.2 对路网进行连接计算后的路段长度分布情况



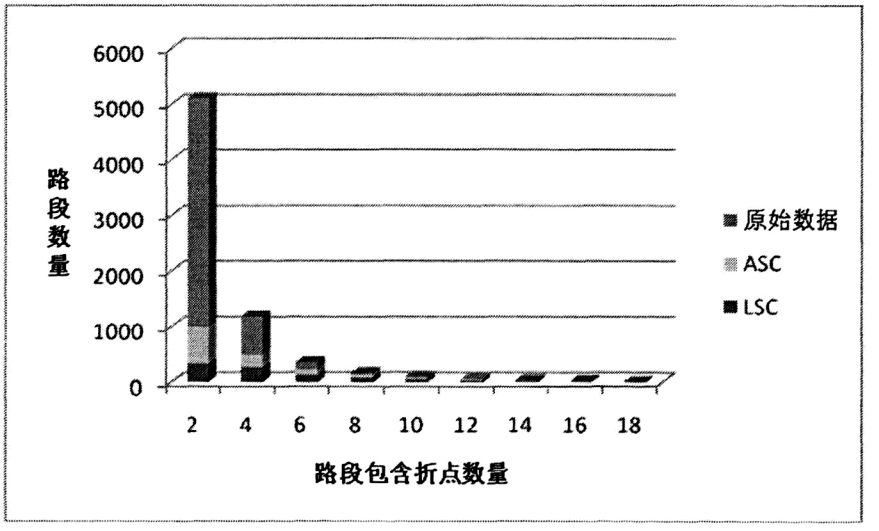
（a）对初始包含 749 条路段的路网做连接操作后的折点数量分布



（b）对初始包含 1237 条路段的路网做连接操作后的折点数量分布



（c）对初始包含 2685 条路段的路网做连接操作后的折点数量分布



（d）对初始包含 5250 条路段的路网做连接操作后的折点数量分布

图 5.3 对路网进行连接计算后的折点数量分布情况

本实验对美国加利福尼亚州公共交通网中四个不同的部分路网进行考察。在原始文件中，每张路网分别包含有 749、1237、2685 和 5250 条路段。通过连接算法计算后，所获得的路网中不同长度的路段分布如图 5.2 所示，而路段中的折点数量分布则如图 5.3 所示。在图 5.2 中，横坐标轴表示的是路段长度，而纵坐标轴则表示了在原始路网中，以及在通过 ASC 或 LSC 算法进行优化后的路网中，含有相应长度的路段数量。而在图 5.3 中，横坐标轴表示的是路网中路段内折点的数量，而纵坐标轴则表示了在原始路网中，以及在通过 ASC 或 LSC 算法进行优化后的路网中，拥有相应数量折点的路段数量。

从实验结果中可以看到，在经过连接运算后，路网中的路段数量明显减少，且不同长度的路段数量差异也显著减小，体现出算法优化的效果。同时在 LSC 与 ASC 算法的比较上，可以看到，ASC 算法中路段的最大长度和含有的折点数均大于 LSC 算法，而分布上，ASC 算法前期分布多于 LSC 算法，后期分布少于 LSC 算法，这里正好体现了算法的思想。由于移动对象在主干道运行的机率更大一些，而主干道的选取上，ASC 算法好于 LSC 算法。同时主干道数量有限，因此在分布上不是主干道越多越好，这点也正是 ASC 算法的优势。两种优化算法都达到了优化的目的，且 ASC 算法较 LSC 算法则更符合移动对象的运动规律。

5.2 索引空间

索引空间的大小，即保存索引结构所需的磁盘空间的大小对索引结构的性能有着重要的影响。索引空间越小，就越能在有限的内存空间中对索引结构的部分甚至全部进行构建，由此可以减少对外存空间的存取要求，进一步提升索引结构的索引效率。

尽管由于数据的内外存交换系统十分复杂，本文中的索引结构均全部构建在内存空间中，不涉及到内外存之间的数据交换，但对于索引结构的索引空间的考量还是十分必要的。

本实验的比较对象是 RR-Tree 与 CR-Tree 两种路网索引结构。这两种结构都实现了对静态路网的索引管理，但由于两者对路网元素的划分方式以及记录内容的区别，导致了其所占用的磁盘空间也是不一样的。在实验中，两种索引结构的结点空间大小均设置为 1KB，索引的路网数据仍是从美国加利福尼亚州公共交通网中所截取的包含不同路段数量的部分路网。

由于 RR-Tree 的基本索引单元是路段，而 CR-Tree 的基本索引单元是路口区域，在同一路网中，路段数量与路口区域的数量是不同的。因此在对索引空间进行比较之前，有必要先对不同路网中路段数量与路口区域的数量进行比较。比较

结果如图 5.4 所示。其中横坐标表示了数个含有不同路段数量的路网数据集，而横坐标中则表示某一路网数据中包含有对应空间元素的数量。

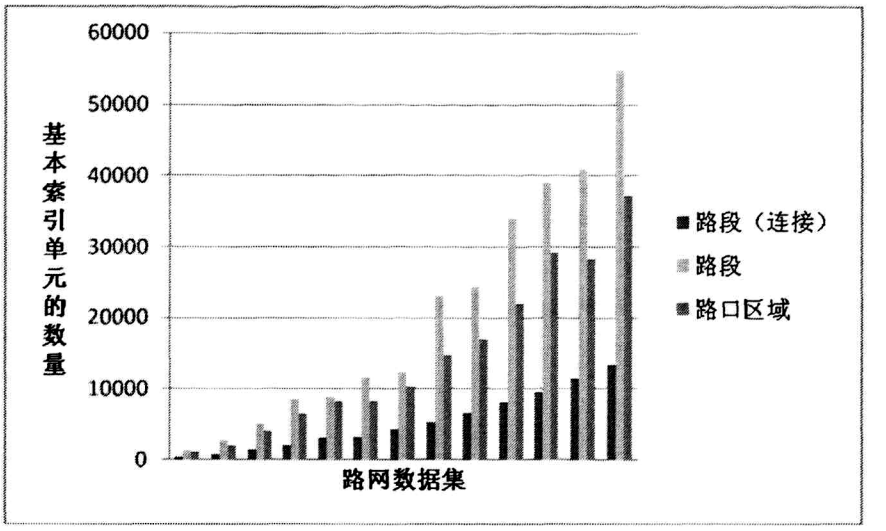


图 5.4 相同路网中索引单元数量的比较

从图 5.3 中可以看出，在相同的路网环境中，路段的数量总是大于路口区域的数量，这是由于同一个路口往往与两到三个以上的路段相关所造成的。随着路网规模的逐渐扩大，即路段数量的上升，路口区域的数量也呈逐渐上升的态势，总体与路段数量保持一定的比例。同时，通过路段连接算法所得到的路段数量较连接之前存在较大的降幅。

对以上路网的索引空间的比较实验结果如图 5.5 所示：

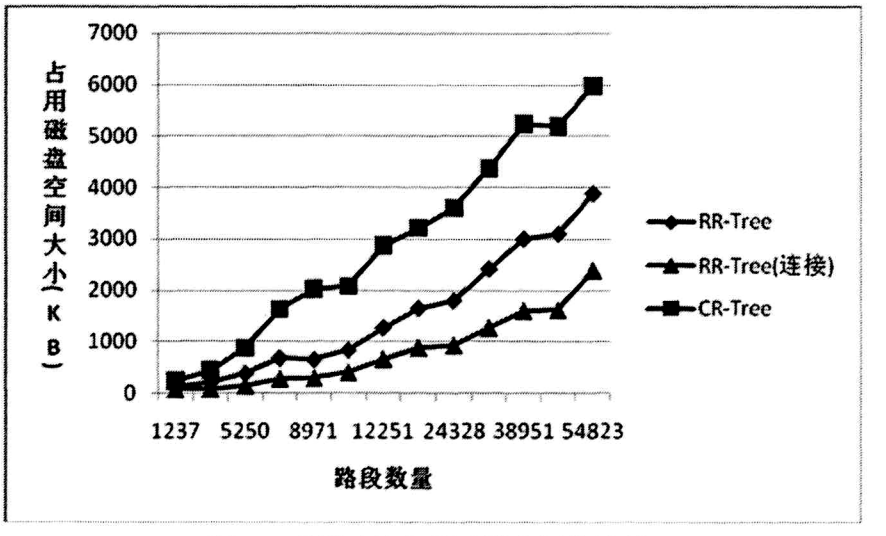


图 5.5 索引结构占用索引空间的比较

从实验中可以看出，随着路网规模的扩大，基本索引单元的数量不断增加，与此同时，对应的索引结构所占用的磁盘空间大小也在不断增大。通过路段连接算法对道路网进行优化后，路段的数量得到了极大的降低，因此占用的索引空间

的大小也减小了。但 CR-Tree 所占用的磁盘空间大小基本上是 RR-Tree 的两倍左右。这是因为 CR-Tree 中不仅记录了路口区域的空间信息，还存储了 CR 间的邻接关系，因此在 CR-Tree 中，每个结点中所能存储的 CR 数据的量降低了，使得索引结构不得不增加更多的结点空间。

5.3 索引准确性

从前文对路网划分方式的讨论中可以看出，以路段为基本空间元素的划分方式可以直观地对路网进行划分，构建类似 NCO-Tree 的复合索引结构以便对路网下的实时受限时空数据加以索引管理。但这种划分方式对于系统环境的实时性要求极高，而 LBS 系统被部署在一个非实时环境中，移动车辆在经过路口时，由于其更新位置无法及时地上传至中心服务器，因此造成了大小不同的盲区，从而对索引结构的索引准确性造成了影响。

为了准确地比较不同的路网划分方式对索引准确性造成的影响，本文构建了一个简单的路网环境，如图 5.5 所示：

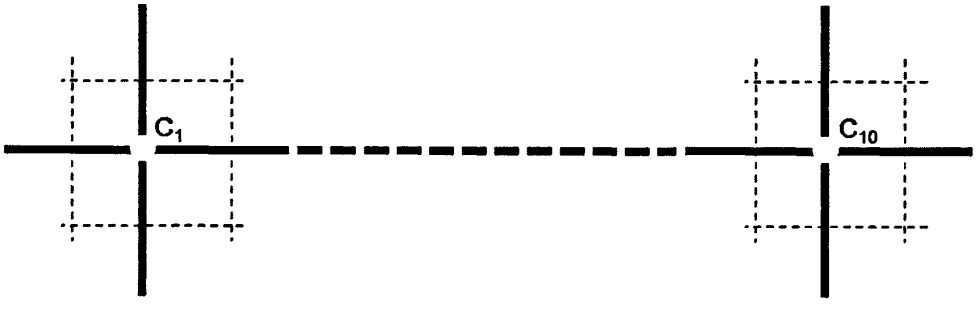


图 5.6 简易路网环境

在如图 5.6 所示的路网环境中，路网由 10 个路口 $C_1, C_2, \ldots, C_{10}$，以及 31 条路段构成，整个路网被划分为 32 个 CR 区域。利用移动对象模拟平台，设置了一批速度相近，轨迹不同的移动对象位置数据快照。

索引结构采用路段追踪策略对移动对象进行追踪索引，但分别利用路段与路口区域对路网进行划分，计算移动对象所在的区域，并与移动对象的真正位置加以比较以得到索引正确率。为了模拟非实时的 LBS 环境，在移动对象通过路口时不会马上提交更新，而是选择一个随机值来延时更新，这样就会对索引的正确率产生影响。

实验结果如图 5.7 所示。图中横坐标轴表示移动对象通过图 5.6 所示路网环境的时间轴，而纵坐标则表示在某一时刻内，索引结构的索引正确率。

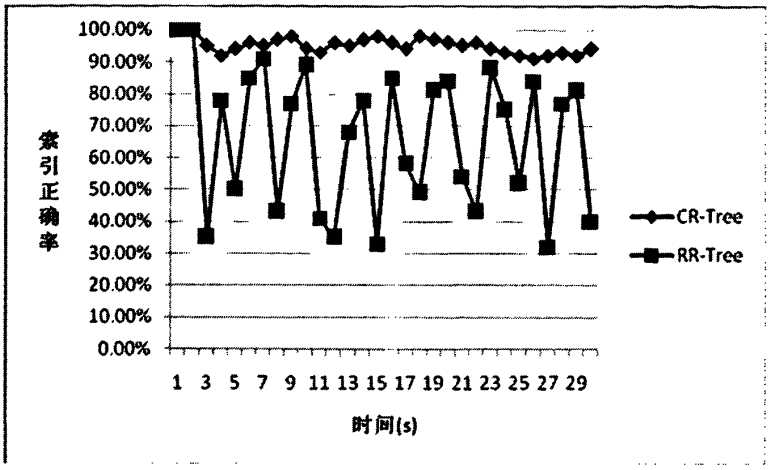


图 5.7 不同路网划分方式对索引正确率的影响

由实验结果可知，基于路段的划分方式对索引正确率存在较大的影响，所得到的结果曲线起伏较大，这主要是因为当车辆通过路口点时，由于盲区的存在，使得系统无法对移动对象的位置做出准确的判断，造成了索引正确率的下降，其平均索引正确率在 66%左右。

CR 的划分方式对索引正确率的影响则较小，由于移动对象的轨迹不同，对象会通过路口离开路网，对索引结构的判断造成了一定的影响，但索引正确率基本稳定在 90%以上。

5.4 查询效率

CR-Tree 与 RR-Tree 除了在对路网的划分方式上存在较大的不同外，两者在存储信息内容上也存在着区别。在 RR-Tree 中，各路段被视为独立的空间元素，原有的邻接关系被忽略了。因此当移动对象从一条路段进行另一条路段中时，RR-Tree 只能依据对象的更新位置，以全体路段元素为查询范围，对索引结构做定位查询操作。CR-Tree 则不同，它利用结点内的邻接矩阵记录了 CR 间的邻接关系，因此当移动对象的前置 CR 已知时，就可以仅查询与前置 CR 存在邻接关系的 CR 即可。同时，通过设置 CR 注册表，可以快速地对存储目标 CR 的叶结点进行访问，进一步提高了查询速度。

本实验着重对 RR-Tree 与 CR-Tree 的查询效率进行比较。测量了在不同规模的路网环境下，RR-Tree 在空间定位查询中所需要访问的索引结点数量，以及 CR-Tree 分别在存在或不存在前置 CR 的条件下做空间定位查询时，所需要访问的索引结点数量。实验结果如图 5.8 所示。图中横坐标表示不同路网数据所包含

路段的数量，而纵坐标则表示在某一路网索引中，对移动对象进行定位查询时，所需访问比较的数据单元数量。

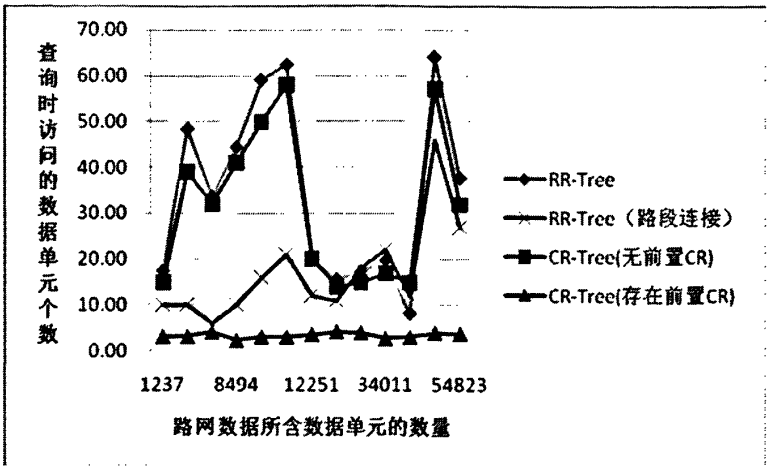


图 5.8 查询效率的比较

由实验结果可知，在不存在前置 CR 信息时，CR-Tree 与 RR-Tree 在查询效率上比较接近。同时，由实验结果也可以看出，两种索引结构的查询效率与路网规模的大小没有直接的联系，这是因为在某些路网中，尽管路段规模大，但道路密度稀疏、定位简单，因此只需访问少量的数据结点就可获得查询结果，而在道路密度高的道路网中，路段间 MBR 存在较多的重叠，因此需对多个结点进行访问比较后才能得出结论。

在利用路段连接算法对路网进行优化后，RR-Tree 的查询效率得到了一定的提升。这主要是因为在对路段进行连接操作后，需管理的路段数量大大降低，因此只需访问较少的索引结点即可获得查询结果。

当存在前置 CR 信息时，访问单元的数量则较少，且较为稳定，不受路网规模或道路密度的影响。这主要是因为在实际路网中，与路口相关的路段数量有限，彼此邻接的 CR 往往存储在相同或邻近的结点中，同时利用 CR 注册表又能快速地对存储邻接 CR 的叶结点进行访问，进一步减少了需要访问的索引结点数量。因此当存在前置 CR 信息时，CR-Tree 则只需访问少量的 CR 单元即可完成查询操作。

5.5 本章小结

本章利用实验对不同的路网划分方式，以及对应构建的两种路网索引结构 RR-Tree 和 CR-Tree 进行了比较。从实验结果可以看出，由于 CR-Tree 不仅存储

了 CR 的空间信息，还存储了 CR 间的邻接信息，因此 CR-Tree 所需占用的磁盘空间大小要远高于 RR-Tree，尤其是当 RR-Tree 的索引单元是连接后的路段元素时，这种空间上的差异则更大。但是采用 CR 的划分方式后，结构对移动对象的索引正确率得到了近 30%的增幅，并能保持稳定。不仅如此，由于 CR-Tree 记录了 CR 间的邻接关系，同时采用了 CR 注册表的方式以达到快速定位 CR 的目的，因此在存在前置 CR 时，其查询效率也较 RR-Tree 有较大的提升。

第六章 总结与展望

为了对 ITS 环境下的基于位置服务提供快速有效的数据支持，本文对城市路网环境下的移动对象索引技术进行了研究，并对其中使用最为广泛的复合索引结构进行了重点分析，指出影响复合索引结构性能的关键因素在于路网的划分方式。基于此，本文提出了两种路网划分方式：基于路段的划分和基于路口区域 CR 的划分，并分别对应建立了两种路网管理结构 RR-Tree 和 CR-Tree 作为索引路网下移动对象位置数据的复合结构的上层结构，且利用实验进行比较。

本文所做的主要工作如下：

(1) 对复合索引结构进行描述，指出其是对路网下移动对象进行管理最有效和使用最广泛的基础结构，并指出影响其结构的关键问题在于路网的划分方式。

(2) 对路网数据进行建模，为后续的研究工作提供了统一的术语空间。

(3) 选择路段作为路网划分的基本单元，以降低复合结构的更新频率。构建了基于路段单元的路网管理结构 RR-Tree，并对以 RR-Tree 为上层结构的受限时空数据索引结构 NCO-Tree 进行了描述。

(4) 指出基于路段的路网划分方式会给索引结构带来索引不准确及更新效率低的问题，提出了基于路口区域 CR 的路网划分方式，以解决原有的索引不准确问题。同时，基于 CR 的空间与拓扑信息，构建 CR-Tree，以降低移动对象查询定位代价，提高以 CR-Tree 为上层结构的复合索引结构的更新效率。

(5) 通过实验对上述两种不同的路网划分方式，以及对应的 RR-Tree 和 CR-Tree 进行比较，对实验结果进行分析，并指出各自的优缺点，

通过理论分析与实验比较，基于 CR 的复合索引结构是一种适合对路网下移动对象进行管理的时空索引结构。但随着研究工作的逐步深入则发现，由于城区路网中的道路密度高、移动对象数量庞大，在更新、查询的效率上仍不能完全满足 LBS 系统的实际需求，因此需要在后续的研究工作中加以改进，具体则拟从以下四方面进行：

(1) 提高 CR 内的移动对象更新效率，以解决基于 CR 的路网划分方式所带来的移动对象更新频率上升的问题。

(2) 车辆在行驶过程中，并不是一直处于行驶状态，而是会因路网中的特例事件而使其运动状态发生改变。如交通事故的发生会导致道路的拥堵，周边车辆的运动或交通信号灯的变化则会导致车辆加速、减速甚至停止。因此如何在索引结构中对此类变化加以考虑，也是后续工作中的重要组成部分。

(3) 由于城市道路网环境复杂，道路密度高，细粒度的划分方式会造成索引结构

的高频更新。本文在 3.2.2 节利用两种路段连接算法 LSC 与 ASC 解决了路段划分的细粒度问题。但由于路口与路段属于路网中两类不同形式的路网单元，且由于时间的限制，无法完整地对路口区域间的连接问题加以研究。因此在后续的工作中，需要对此问题加以更深入的分析研究。

(4) 由于在实际的 LBS 系统中对于交通概要数据的需求，如查询道路中的车辆数量、当前平均时速等往往高于对移动对象个体位置的需求。因此在后续的研究中，需要对如何能够快速有效地查询交通概要数据进行研究，以便更好地为 LBS 系统提供支持。

致　谢

本文是在导师朱跃龙教授和冯钧教授的悉心指导下才得以完成。三年的学习生活中，朱老师和冯老师正直的品质、渊博的学识、严谨的治学态度和务实的工作作风使我受益良多。在学习上，他们给我提供了良好的学习与研究环境，帮助我安心地进行学术研究工作，同时在生活上他们也给予我无微不至的关怀。冯老师在课题的选择范围，学术研究的方法和论文的写作方式上对于我的指导，更是让我终身受用不尽。值此论文完成之际，首先要向朱老师和冯老师表达诚挚的感谢！

感谢王萌同学在本文路段连接算法的研究中所给予我的无私帮助。

感谢任翔、张华、吴林燕、马骏、卢阳、张凡、王琰、王颖，通过与你们的交流和讨论，才有了今天的这篇论文。

感谢所有关心、帮助我的老师、同学和朋友们！

感谢我的父母，感谢我的女友丁娟，是你们的扶持与关爱，才能使我顺利完成学业！

参考文献

[1] Hongzi Zhu, Yanmin Zhu, Minglu Li, Lionel M. Ni: ANTS: Efficient Vehicle Locating Based on Ant Search in ShanghaiGrid. ICPP 2007:34

[2] Alminas Civilis, Christian S. Jensen, Stardas Pakalnis: Techniques for Efficient Road-Network-Based Tracking of Moving Objects. IEEE Trans. Knowl. Data Eng. (TKDE) 17(5):698-712 (2005)

[3] Jun Feng, Jiamin Lu, Yuelong Zhu, Naoto Mukai, Toyohide Watanabe: Indexing of Moving Objects on Road Network Using Composite Structure. KES 2007:1097-1104

[4] Kyoung Soo Bok, Ho Won Yoon, Dong Min Seo, Su Min Jang, Myoung-Ho Kim, Jae Soo Yoo: Indexing the Current Positions of Moving Objects on Road Networks. APWeb/WAIM Workshops 2007:247-252

[5] Li Guohui, Zhong Xiya: Indexing moving objects trajectories on fixed networks. Computer Research and Development 2006 43, 5, pp 828-833

[6] Jignesh M. Patel, Yun Chen, V. Prasad Chakka: STRIPES: An Efficient Index for Predicted Trajectories. SIGMOD 2004:637-646

[7] Victor Teixeira de Almeida, Ralf Hartmut Güting: Indexing the Trajectories of Moving Objects in Networks (Extended Abstract). SSDBM 2004:115-118

[8] Yufei Tao, Dimitris Papadias, Jimeng Sun: The TPR*-Tree: An Optimized Spatio-Temporal Access Method for Predictive Queries. VLDB 2003:790-801

[9] Dieter Pfoser, Christian S. Jensen: Indexing of network constrained moving objects. GIS 2003:25-32

[10] Ouri Wolfson, Huabei Yin: Accuracy and Resource Concumption in Tracking and Location Prediction. SSTD 2003:325-343

[11] Elias Frentzos: Indexing Objects Moving on Fixed Networks. SSTD 2003:289-305

[12] Dongseop Kwon, Sangjun Lee, Sukho Lee: Indexing the Current Positions of

Moving Objects Using the Lazy Update R-tree. Mobile Data Management 2002:113-120

[13] Simonas Saltenis, Christian S. Jensen, Scott T. Leutenegger, Mario A. Lopez: Indexing the Positions of Continuously Moving Objects. SIGMOD 2000:331-342

[14] Dieter Pfoser, Christian S. Jensen, Yannis Theodoridis: Novel Approaches in Query Processing for Moving Object Trajectories. VLDB 2000:395-406

[15] Mario A. Nascimento, Jefferson R. O. Silva, Yannis Theodoridis: Evaluation of Access Structures for Discretely Moving Points. Spatio-Temporal Database Management 1999:171-188

[16] Jamel Tayeb, Özgür Ulusoy, Ouri Wolfson: A Quadtree-Based Dynamic Attribute Indexing Method. Comput. J. (CJ) 41(3):185-200 (1998)

[17] Yannis Theodoridis, Michalis Vazirgiannis, Timos K. Sellis: Spatio-Temporal Indexing for Large Multimedia Applications. ICMCS 1996:441-448

[18] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, Bernhard Seeger: The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. SIGMOD 1990:322-331

[19] X. Xu, J. Han, and W. Lu. RT-Tree: An Improved R-Tree Indexing Structure for Temporal Spatial Databases. In Proc. Intl. Symp. Spatial Data Handling, SDH(1990), pp 1040–1049

[20] Antonin Guttman: R-Trees: A Dynamic Index Structure for Spatial Searching. SIGMOD 1984:47-57

[21] Hanan Samet: The Design and Analysis of Spatial Data Structures. Addison-Wesley 1990

[22] Neil Smith: Serious book for systems writers and students : Applications of spatial data structures: computer graphics, image processing and GIS Hanan Samet Addison-Wesley (1989) 505 pp. Computer-Aided Design (CAD) 22(10):673-674 (1990)

[23]Timos K. Sellis, Nick Roussopoulos, Christos Faloutsos: The R -Tree: A

Dynamic Index for Multi-Dimensional Objects. VLDB 1987:507-518

[24] Michael Stonebraker, Timos K. Sellis, Eric N. Hanson: An Analysis of Rule Indexing Implementations in Data Base Systems. Expert Database Conf. 1986:465-476

[25] Samet H: The quadtree and related hierachical data structure. ACM Comp. Surv. (1984) 16, 2, 187-260

[26] Fuchs H, Abram G D, Grant E D: Near real-time shaded display of rigid objects. Computer Graphics(1983) 17,3,65-72

[27] Robinson J T: The KDB-tree: A search structure for large multidimensional dynamic indexes. In Proc. ACM SIGMOD. Int. Conf. on Management of Data(1981), pp 10-18

[28] Fuchs H, Kedem Z, Naylor B: On visible surface generation by a priori tree structures. Computer Graphics(1980) 14, 3

[29] Bentley J L: Multidimensional binary search in database applications. IEEE Trans. Software Eng(1979) 4, 5, 333-340

[30] Bentley J L: Multidimensional binary search trees used for associative searching. Comm. ACM(1975) 18,9, 505-517

[31] Ralf Hartmut Güting, Victor Teixeira de Almeida, Zhiming Ding: Modeling and querying moving objects in networks. VLDB J. (VLDB) 15(2):165-190 (2006)