基于树状区块链的出租车调度算法设计和系统实现

成佳壮

**** 年 * 月

中图分类号：TQ028.1

UDC分类号：540

基于树状区块链的出租车调度算法设计和系统实现

作 者 姓 名 成佳壮

学 院 名 称 计算机学院

指 导 教 师 陆慧梅副教授

答辩委员会主席 ** 教授

申 请 学 位 工学硕士

学 科 专 业 电子信息

学位授予单位 北京理工大学

论文答辩日期 **** 年 * 月

Algorithm Design and System Implementation of Taxi Scheduling Based on Ethereum

Candidate Name:	Jiazhuang Cheng
School or Department:	Computer Science and Technology
Faculty Mentor:	Prof. Huimei Lu
Chair, Thesis Committee:	Prof. **
Degree Applied:	Master of Engineering
Major:	Digital Information
Degree by:	Beijing Institute of Technology
The Date of Defence:	*, ****

基于树状区块链的出租车调度算法设计和系统实现

北京理工大学

研究成果声明

本人郑重声明：所提交的学位论文是我本人在指导教师的指导下进行的研究工作获得的研究成果。尽我所知，文中除特别标注和致谢的地方外，学位论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京理工大学或其它教育机构的学位或证书所使用过的材料。与我一同工作的合作者对此研究工作所做的任何贡献均已在学位论文中作了明确的说明并表示了谢意。

特此申明。

作者签名：_____　　签字日期：_____

关于学位论文使用权的说明

本人完全了解北京理工大学有关保管、使用学位论文的规定，其中包括：①学校有权保管、并向有关部门送交学位论文的原件与复印件；②学校可以采用影印、缩印或其它复制手段复制并保存学位论文；③学校可允许学位论文被查阅或借阅；④学校可以学术交流为目的，复制赠送和交换学位论文；⑤学校可以公布学位论文的全部或部分内容（保密学位论文在解密后遵守此规定）。

作者签名：_____　　导师签名：_____

签字日期：_____　　签字日期：_____

摘要

车载自组网是在交通环境参与者间构建的开放式网络，可以为用户提供去中心化的数据传输能力。基于车载自组网，可以实现事故预警、辅助驾驶、道路交通信息查询、车间通信和网络接入服务等应用。研发这些应用需要地理信息和交通数据的支持，但信息的垄断会引发不正当牟利和恶性竞争。针对这一问题，本文利用部署在车载自组网上的区块链网络，基于 GeoHash 矢量地图和基于以太坊的树状区块链平台，开发了一套出租车调度和导航系统以完成出租车的去中心化调度。首先，本文选用 GeoHash 作为系统中统一的位置信息表示方法，在系统实现上采用浏览器与智能合约相结合的方式，在智能合约端开发了基于 GeoHash 的路径导航算法和车辆的区域调度算法，解决了车乘分配时的并发冲突问题。在浏览器端实现车辆和乘客的数据采集和乘车业务完整流程的设计。系统充分利用了区块链的性质保证车辆信誉数据的安全性、可溯性和在网络内的同步性。利用 GeoHash 在地理信息上的计算特性对算法速度进行了优化，并进行了优化后的实验验证工作。最后调节系统的关键参数进行性能优化并进行实验验证，通过真实的地图数据验证了此出租车调度系统的可行性。

关键词：区块链；导航；GeoHash

Abstract

VANET is an Adhoc networks between participants in trafic and providing decentralized data transmission service. VANET can be uesd in application like accident warning, drive assist system, trafic information service and InterVehicle Communication. The development of these applications requires the support of geographic information and traffic data, but the monopoly of information will lead to unfair profit-making and vicious competition. In response to this problem, this paper uses the blockchain network deployed on the in-vehicle ad hoc network, based on the GeoHash vector map and the Ethereum platform, to develop a taxi dispatch and navigation system to complete the decentralized dispatch of taxis. First, this thesis use GeoHash for storage and calculation of position. The system in this thesis is made up of browserside programs and smart contract. On the smart contract side, a GeoHash-based route navigation algorithm and a vehicle regional managing algorithm were developed, which solved the problem of concurrency conflicts in the allocation of vehicles and passengers. The browser side realize vehicle and passenger data collection and design the complete process of ride-hailing business. Blockchain makes this system safe, traceable and synchronized. The speed of the algorithm is optimized by using GeoHash's computing characteristics on geographic information, and the optimized experimental verification work is carried out. Finally, the key parameters of the system are adjusted for performance optimization and experimental verification. The feasibility of the taxi dispatching system is verified by real map data.

Key Words: blockchain; navigation; GeoHash

主要符号对照表

BIT	北京理工大学的英文缩写
LaTeX	一个很棒的排版系统
LaTeX 2_ε	一个很棒的排版系统的最新稳定版
XeTeX	LaTeX 的好兄弟，事实上他有很多个兄弟，但是这个兄弟对各种语言的支持能力都很强
ctex	成套的中文 LaTeX 解决方案，由一帮天才们开发
H_2SO_4	硫酸
$e^{\pi i} + 1 = 0$	一个集自然界五大常数一体的炫酷方程
$2\,H_2 + O_2 \longrightarrow 2\,H_2O$	一个昂贵的生成生命之源的方程式

目　录

插　图

表　格

第 1 章　绪论

1.1　本论文研究的目的和意义

随着城市交通的逐渐发展，道路网络的复杂度以及车辆保有量日益增长，交通基础设施的建设无法满足需求，给交通流量的管理带来困难。在智能交通系统中，自动化的出租车调度系统可以有效地满足实时的出行需求，为智能交通系统的完善提供技术支持，有效地满足城市交通需求以及交通流量的管理与诱导，能够有效提高用户出行效率。（研究出租车调度系统的目的和意义）在车辆调度领域，目前已有集中式的自动化出租车调度系统[?]。

信息的去中心化管理是一项关键技术。设计去中心化的出租车调度系统可以保障用户信息不被滥用，避免恶性竞争，在提高交通系统中乘客乘车效率的同时，也兼顾了出租车司机收入的公平。例如已有为缓解机场压力而设计的机场出租车调度模型，从机场管理部门的角度统筹部署机场出租车调度管理系统，在一定程度上避免了私营企业通过集中式后台管理的软件产品进行不正当牟利，提高机场和市政部门的工作效率[?]。

区块链可以用作去中心化的出租车调度系统的开发平台[?]，区块链是一种去中心化的共享账本，它可以安全地将简单，连续和经过验证的数据存储在系统软件中。与传统技术相比，区块链具有以下三个优势：

一是它不能被伪造，所以更安全。每个人都必须验证自己的身份，然后才能根据专用安全通道访问数据库，添加或获取数据，并保留历史访问数据。

第二是具有很强的实用性和可信性。每个节点都将维护一个详细数据账本，并且该数据组仍处于不同对象的控制之下，并且根据共识算法将数据维持在高度一致的水平，增强了系统的健壮性。

第三，它具有智能合约和全自动执行功能。智能合约具有完全透明，可靠，强制执行和全自动执行的优势，可以支持构建去中心化的后台系统。

利用区块链这一去中心化后台开发自动化的出租车调度系统，可以同时拥有信息保密、系统安全健壮、利益分配公平公正这些优点，有助于解决人们对高可信性的出行软件的需求问题，为推动智慧交通系统的建设添砖加瓦。

1.2 国内外研究现状及发展趋势

关于出租车调度系统的研究，目前已经有工作提出基于区块链的拼车系统模型[?]，考虑了车辆和乘客双方的身份验证，但只是提出了原型理论，没有实现车辆调度流程。另外还有在以太坊平台实现的拼车系统，提出了司机和乘客对互相的信任程度的概念[?]，Deng 等提出了基于区块链的车载网安全支付方案[?]，这些工作侧重于安全性的验证和评估，并没有考虑在实际应用中让分布式节点起到车辆调度和导航服务的作用。本文采用基于 GeoHash 的矢量地图作为基础数据，然后基于地理信息在树状区块链上实现导航系统，以实现最低路径成本的导航算法。

随着通信技术和硬件技术的发展，移动用户剧增，随之交互式信息地图服务需求逐渐增多，矢量数据地图正在兴起。在矢量地图数据中，矢量数据可以在所有放缩水平下以不同的颜色正确显示和区分特征，使地图展现更加丰富[?]。矢量地图数据的编码方式是影响其传输性能从而影响其可用性的关键因素[?]。

XML 和 JSON 是 web 应用程序中常用的两种矢量数据编码方法[?]。但 XML 使用重量级语法导致其格式复杂且大小较大，不利于地图数据传输。JSON 是一种易于读写的轻量级数据表示格式，GeoJSON 是其中一种轻量级的数据表示方法，用于编码各类地理数据结构，可用于简单表示地理信息[?]。

传统的矢量地图如 Google Maps[?]，采用二维数据经纬度来表示地理信息。Geohash 是一种新型的地址编码方式，不同的是它使用 Base32 编码成一维的字符串代替二维的经纬度数据，将二维空间查询转换为一维字符串匹配。利用此优势，GeoHash 编码可以实现时间复杂度为 O(1) 快速查询[?]。此外，相比基于 Base4 编码方法的 Bing Maps[?]，GeoHash 编码使用 Base32 编码方法，即同一前缀有 32 个不同子序列，这缩短了一维字符串的长度从而减小了数据存储和传输的成本。根据编码的长短，GeoHash 可以同时表示图块的坐标和索引范围，从而实现两种功能的统一编码。本文所设计系统的地图存储与展现、导航算法的逻辑将统一基于 GeoHash 编码，完全替换传统地图的经纬度数据表示形式，简化了数据传输和算法逻辑，同时方便了区域信息绑定和查询。

传统区块链的特点不能满足车联网的特性。一方面，传统区块链吞吐量较小、出块速度慢，需要改善它的结构来提升可扩展性；另一方面，传统区块链与真实世界的联系较弱，而车联网与地理信息有天然联系并且紧密相关，因此传统区块链结构不能很好适应车联网[?]。就目前来说，主流的方案是将区块链的单链结构改良为多链结

构。根据多链结构的构成方式的不同，可以将相关工作分为三类：并发多链[?]，层次多链[?]，智能合约多链[?]。并发多链存在多条子链，每条子链对应一个独立的功能模块，主链只负责维护多条子链的数据一致性；层次多链以层次化的结构组成区块链；智能合约多链通过智能合约来维护多链数据的一致。树状区块链实现了层次化多链，同时将地理位置融合进区块链的数据结构，对区块链的结构进行修改与优化，使其更能适应车载自组网的特性 [周畅论文]。本文基于树状区块链设计出租车调度系统，能更快速地查询和利用地理位置信息，提高系统性能。

导航系统的核心部分可分为地理基础数据和底层路径规划算法实现，Dijkstra 算法是最短路径算法的鼻祖，但其作为最基本的图搜索算法，无法满足地图路径规划对性能的要求[?]。目前工业界地图产品主流的路径规划算法有 A* 算法，CH(Contraction Hierarchies) 算法[?]。其中 A* 算法是应用最广泛的路径规划算法，其使用了启发式算法，比 Dijkstra 算法速度更快[?]；CH 算法实现了数据预处理，以减少需要搜索的节点，但其不支持地理信息的实时更新，且不能很好地支持多种道路权重类型。本文基于 A* 算法设计了可支持 GeoHash 格式的地理信息的导航算法。

1.3 论文的研究内容、贡献和组织结构

1.3.1 论文的研究内容

（1）为优化区块链对地图数据的处理性能，降低计算量，引入 GeoHash 矢量地理信息存储，并为前端矢量地图数据渲染工具——leaflet 添加支持 GeoHash 格式的放缩和拖动功能。

（2）为解决中心化的打车软件对用户信息进行违规利用、利用信息差进行恶性竞争等问题，本文基于树状区块链设计并实现了去中心化的出租车调度系统。

（3）实现出租车调度系统的核心是实现导航算法和车乘匹配算法，然而，树状区块链缺乏基于矢量地理数据的导航算法支持，为解决这个问题，本文在基于以太坊的树状区块链平台设计并实现了基于 GeoHash 矢量地理数据的导航算法，并在此基础上实现了基于树状区块链的区域调度车乘匹配算法。

（4）针对基于 GeoHash 的距离计算方法，本文通过前缀匹配对算法逻辑进行了优化；同时，本文在对已广泛应用的导航算法进行了修改，使其支持 GeoHash 的距离计算逻辑，在导航算法中加入可调节的参数增强其适配性；对导航算法和区域调度车乘

匹配算法的关键参数进行调优。

1.3.2 论文贡献

（1）完善 leaflet 工具对 GeoHash 格式矢量地图展示的支持。

（2）提升 GeoHash 距离计算算法的速度。

（3）设计出基于 GeoHash 的导航算法并进行参数调优。

（4）设计出基于树状区块链的区域调度车乘匹配算法并进行参数调优。

1.3.3 论文的组织结构

第一章，介绍导航应用研究现状以及去中心化调度系统对于智能交通和反垄断、反恶性竞争的意义。

第二章，对本文涉及的相关工作进行综述，包括基于以太坊的树状区块链、GeoHash 地理信息、leaflet 渲染工具、出租车调度系统、导航算法。

第三章描述系统的框架，包括服务端智能合约的设计结构，和浏览器终端车辆和乘客的设计结构，以及系统的运行流程。

第四章详细介绍系统用到的各种技术，包括基于 GeoHash 的矢量地图存储和展示、基于 GeoHash 的导航算法设计、基于树状区块链的区域调度车乘匹配算法。

第五章，介绍了出租车调度系统的参数调优以及系统在真实数据下的工作状态。

第 2 章　相关工作

本章对本文研究内容的相关工作进行简要介绍。首先是基于以太坊的树状区块链，具体涉及传统区块链的性能瓶颈和树状区块链的结构特性。第二是 GeoHash 地理信息，具体介绍 GeoHash 地理信息便于区域绑定和查询的特点，基于 GeoHash 的几何原理，以及矢量地图渲染框架 leaflet。第三是出租车调度系统的发展现状，并指出其安全性上的不足和解决方法。第四是路径规划算法，介绍路径规划算法的发展种类，对比其性能和特点，在此基础上简要介绍基于 GeoHash 的路径规划算法设计。

2.1　基于以太坊的树状区块链

区块链是一个共享的不可更改的总账，它用于记录交易、跟踪资产以及建立信任[?]。几乎所有有价值的东西都可以在区块链网络上进行跟踪和交易，从而降低了风险并削减了所有相关成本。区块链是传递交易信息的理想选择，因为它可以提供在不可更改的总账中的即时、共享且完全透明的信息，这类信息同时具备一定的安全性，只能由获得许可的网络成员访问。另外，区块链网络也可以跟踪订单、付款、账户、生产等等，因此可以通过此类技术手段减少不必要的交易纠纷。

智能合约是存储在区块链上的程序，可以在满足预定条件的情况下运行，它们通常是自动执行的脚本，以便所有参与者无需任何中介机构的参与就可以立即结果，极大保证安全性。智能合约代码语句十分简单，当预定条件已经得到满足并完成验证时，区块链网络将执行对应动作，比如释放资金，发出凭证等，然后交易完成时更新区块链。由智能合约完成的交易也是无法更改的，只有获得许可的参与者才能看到结果。本文是基于以太坊改进的树状区块链平台来设计和部署智能合约，在此基础上进行出租车调度系统的研究。

2.1.1　传统区块链的特点和不足

传统区块链技术整个网络同时只有一条单链，基于 PoW 共识机制出块无法并发执行，无法满足车载自组网对高并发操作和网络稳定性的需求，传统区块链也不具备移动性，无法与地理位置信息进行绑定，不能有效利用车载网中地理信息的区域化特性。另外，传统区块链的单链结构，要求所有节点必须在同一区块链中，这将会导致

节点数目和数据量过大，不满足车载自组网中车辆节点的移动特性，且一旦出现网络分区，就会对整个区块链产生很大影响。目前采用分片技术更改原始单链的链式结构是解决上述问题的主流方法[?]。当前也有多链结构的相关工作。根据多链结构是否改变底层区块链结构可分为应用多链和结构多链。应用多链，即在应用层面建立不同功能的多个区块链，没有修改底层区块链结构。结构多链，即根据需求对区块链底层结构进行调整的多链结构。

2.1.1.1 应用多链

Shrestha 等[?]研究了区域区块链在车载自组网中的应用。由物理边界区域内的节点共享的区块链，区域内部的传播延迟比较小，可以极大减少消息延迟，但此研究并没有设计跨区域和跨链交易的相关内容。

Hirtan 等[?]提出了一种包含专用链和公用链的医疗保健系统。专用链保存患者的真实身份信息；公用链存储患者的健康信息以及临时 ID 数据，实现隐私数据保护和可用数据公开访问。此研究通过特定节点掌握患者临时 ID 和真实身份的关联，来实现两个区块链数据的传递。

2.1.1.2 结构多链

Youngjune 等人[?]建立多个独立并行的单链共识组。各个共识组地位平等，大部分交易只在组内完成，跨组交易则采用异步方式将中继事务发送到目标区域，而不是整个网络，减轻了网络负载。值得一提的是，为了确保每个区域中的有效采矿能力与整个网络处于同一水平，采用了诸葛弩改进 PoW 的挖矿方式，这也同时保证了分组抵御攻击的能力。但其共识组分区方法没有考虑实际地理位置信息，同时车载自组网中跨区域交易的规模较大，此研究的跨组交易的网络开销较大，不适用于车载自组网。

Zamani 等人[?]提出了基于分片的公共区块链协议，将节点划分为多个较小的称为委员会的节点组，节点组在不相交的交易块上并行操作，并维护不相交的独立账本，也就是分片，分片由每个成员以区块链的形式存储。为了解决节点频繁移入移出对网络造成的影响，将委员会分为活跃和不活跃两个分类，节点创建后，第一次进入委员会需要加入活跃类，再次进入或转移时需要加入不活跃类。但委员会内节点数量

固定，缺少灵活性；委员会构建和重构时不涉及地理因素，增加了更新时间，不适用于车载自组网。

Byung 等人[?] 将物联网与基于区块链的智能合约相结合，用于结构健康监视(SHM)。这个区块链物联网网络分为核心和边缘网络。边缘节点充当查询实时响应的集中式服务器，并提供低延迟和带宽使用率，核心网络由具有高存储容量的 miner 节点组成，负责生成新块、验证工作证明，并包含自主决策的智能合约，这种划分提高了系统的效率和可伸缩性，但核心网络十分庞大是主要问题。核心网络的庞大影响移动节点的交易效率。

Pajooh 等人[?] 提出了一种多层区块链安全模型来保护物联网网络，同时利用群集的概念来简化多层架构，通过使用模拟退火和遗传算法相结合的混合进化算法来定义物联网 K- 未知簇，选择的群集头负责本地身份验证和授权，增强了网络认证机制的安全性，显示出更适合的平衡网络延迟和吞吐量。但上述两类区块链研究没有考虑到地理因素，跟适用于车载自组网的区块链结构还有一定的距离。

Ochôa 等人[?] 提出了侧链结构，由 BlockPRI、BlockSEC、BlockTST 三个不同的区块链，使用了三个区块链来确保系统的隐私性，安全性和信任性。BlockPRI 存储每个用户的隐私首选项。BlockSEC 存储用户的数据。最后，BlockTST 管理并验证有关消费者-生产者与消费者-公司之间的能源贸易的信息。为了保证区块链之间的通信，需要通过智能合约维护多链数据一致性。本文吸收了用智能合约自主决策的思想，基于改进的区块链结构和智能合约实现出租车调度系统。

2.1.2 树状区块链的特性

车载自组网具备低成本、低时延以及地理位置天然联系的特性，在道路安全等应用提供了重要支持[?]。区块链作为一种新型的互联网模式，具备去中心化、不可篡改、可追溯等特点，与车联网的联系愈发紧密[?]。但实际应用上主要包括两个问题，首先，区块链技术通常要求可靠的网络接入，而车载自组网高动态性的接入方式引发的网络不稳定对区块链的应用造成了极大的挑战。其次，车载自组网中的信息通常与地理信息天然绑定，而传统区块链并不支持此特性。因此，研究将地理位置与区块链融合的树状区块链 [周畅论文] 就是为上述灵活性和地理区域问题提供一种可能的解决方案。

由于全局同步与共识速度的限制，传统区块链不能很好的适应车载自组网的高动态性，并且不能保证车辆与路侧节点的稳定连接，这将会极大影响到区块链共识机制

的可用性。车载自组网与地理信息天然相联，并且车辆通常只需要关注特定区域的信息，而传统区块链记录的交易信息不包含位置，无法根据地理区域查询交易，且传统区块链全局同步的设计会造成网络资源的大量浪费，并且传统区块链无法根据地理信息高效地查询账户和交易相关的数据。树状区块链对区块链的结构进行修改与优化，使其更能适应车载自组网特性。树状区块链是一种基于位置的区块链 [周畅论文]，首先，根据 GeoHash 地理编码与地理区域层次关系建立树状结构的区块，同时研究树状结构地理信息相关数据查询速度的优化机制，从而提升区块链在车载自组网中的吞吐量以及与地理信息相关数据的检索速度。

GeoHash 是一个能表示任意精度的高效地理编码系统，它使用二分法将指定区域分为网格，每个网格由唯一的编码表示，网络的大小对应的层次不同，编码越长对应的网格越小，层次越低。树状区块链使用 GeoHash 地理编码作为基本数据编码方式。建立 GeoHash 与区块以及交易的索引，从而依据 GeoHash 编码直接获取区域交易内容，同时，GeoHash 编码可以通过前缀对区域信息进行绑定，因此通过 GeoHash 编码可以很自然地建立区域层次与子链的联系。因此本文选用树状区块链平台来实现分布式的出租车调度系统。

2.2 矢量地图研究

本节回顾和讨论了 Web 终端访问地图数据的应用程序中涉及的矢量地图编码和空间索引方法。然后简单介绍了本文采用的 GeohashTile 系统使用的 Geohash 编码方法和 Leaflet 以及相关工作的比较。

矢量数据的编码是影响其传输性能和可复用性的关键因素。XML 和 JSON 是 Web 应用程序中常用的两种矢量数据编码方法[?]。XML（可扩展标记语言）[?] 用作 Internet 信息交换的标准。由于 XML 使用的语法不够轻量，不利于 Internet 上的数据传输[?]。

2.2.1 矢量地图数据编码

GeoJSON 是一种用于编码各种地理数据结构的开放标准格式，可用于表示简单的地理特征。GeoJSON 可以比 XML 更方便、更快速地被计算机解析，数据结构更轻量易读[?]。GeoJSON 作为一种轻量级的数据编码方法，适用于移动设备之间的数据传输[?]。考虑到编码方式的传输性能、可读性和易分析性，我们选择 GeoJSON 作为移动设备矢量数据的编码方式。

2.2.2 矢量地图空间索引技术

空间索引技术是提高海量空间数据查询效率的关键技术。空间索引对瓦片金字塔的地理数据进行管理和维护具有重要的作用，其性能直接影响地理信息网络服务的整体性能。其中，网格索引和四叉树索引是瓦片金字塔模型中广泛使用的空间索引方法[???]。

网格索引是按照一定的分辨率等级,将地理信息进行划分并按照矩形网格排列[?]。网格索引法要求在查询金字塔中的任何一个瓦片时，只需要查询三个值，即分别代表行、列坐标和缩放级别的 X、Y 和 Z。网格索引是最早的索引方法之一，形式简单。但是，三字段查询也使得它在海量数据的情况下效率低下。

四叉树索引以其每个内部节点都有四个子节点命名，是多分辨率在线地图的常用索引方法。该索引方法具有编码简单、易于实现的优点，已被微软必应地图在内的大多数主要网络地图服务提供商采用。

谷歌地图的索引方式与网格索引相同。(x, y, z) 三个字段用于表示图块索引值。因此存在海量数据查询效率低的问题。Bing Maps 使用（z，quadkeys）两个字段来表示图块索引值，其中 quadkeys 称为四叉树键，通过按位交叉组合将二维块 XY 坐标组合成一维字符串来优化索引和存储。

Geohash 的索引方法是将第 M 层中的一个块划分为第 M+1 层中的 n 个块，所以它也是一种类似四叉树索引方法。与常见的索引结构相比，Geohash 在索引时不需要经过递归计算，因此空间索引只有一层，使得其动态更新更简单[?]。

2.2.3 Geohash 编码

Geohash 编码将纬度和经度分别转换为一组二进制字符串，然后将这两组字符串逐位交叉，生成一组新的二进制字符串。然后将新字符串每五位转成一个十进制数，按顺序转换成 base32 编码，这样就可以使用用一维数组表示二维数组[?]。

与谷歌地图的编码方式相比，Geohash 将二维空间查询转换为一维字符串匹配。凭借这一优势，Geohash 可以实现时间复杂度为 O(1) [?] 的快速查询。与 Bing Maps 的编码方式相比，Geohash 采用 Base32 编码方式，即同一个前缀下有 32 个不同的子序列，而 Bing Maps 编码方式是 Base4，即同一个前缀下只有 4 个不同的子序列，所以 Geohash 查询更方便。文献[?] 还表明，基于 Geohash 的空间索引算法对海量地理数据

具有高性能的查询能力。根据编码长度的不同，Geohash 可以同时表示图块的索引范围和坐标，实现统一编码。

2.2.4 Leaflet 矢量地图渲染框架

Leaflet 是地图的开源 JavaScript 库之一，是 B/S 端 WebGIS 开发项目中广泛使用的开源软件。开发者可以基于库中提供的接口进行开发和扩展，实现地理信息服务的调用和地图数据的基本操作[?]。Leaflet 强大的开源库插件涉及地图应用的方方面面，包括地图服务、数据提供、数据格式、地理编码、地图控制与交互等，也支持自定义控件的实现。这些控件丰富了 Leaflet [?] 的功能。本文的出租车调度系统基于轻量级 WebGIS 库 Leaflet 来呈现终端数据，并拓展了其支持 GeoHash 数据格式的显示功能。

2.3 出租车调度系统

随着互联网通信技术的发展，打车软件和其背后的企业以指数级的速度成长，软件平台的即开即用，让乘客主动广播打车需求，方便了居民出行，同时其灵活性和较高的薪酬水平也在吸引全职和兼职司机的不断加入。一项研究[?] 表明，对灵活工作时间的向往驱使着司机们和打车平台的企业进行合作。但在鼓励此类企业发展以解决就业和城市管理效率的同时，也应当对其行使的权力进行监管，避免出现敏感信息垄断、滥用、泄露，从而造成难以预计的后果。

[?] 提到了一种在用车高峰时进行激增定价的策略，目的是提高司机在用车高峰时的工作积极性，满足乘客在高峰期间的多样化乘车需求，所有利益相关者都可以从具有自我调度能力的平台上通过激增定价策略受益。但实际是让打车平台获得了更多攫取利益的空间，使得平台的使命从方便公共交通转向了使自己利益最大化。

传统中心化调度系统在存储容量和处理性能方面存在不足，当访问服务器的需求规模扩大后，系统就无法及时处理接收到的需求数据，进而导致响应速度降低和服务质量的下降。随着数据规模的逐渐扩大，单机环境下的处理模式已经不适用于海量数据的存储与计算，分布式存储与计算应运而生[?]。但是事实上，由打车软件提供商设计的分布式系统，其收集和存储到服务器中的用户数据很可能会被泄露给第三方，特别在导航系统中，车辆司机和用户都会提交 GPS 位置和个人账户信息来获得交通服务和进行导航服务，服务商会出于自己的目的处理用户隐私。

文献[?] 的研究将车载网和以太坊结合，实现了透明、自管理的去中心化系统，文

章中使用以太坊账户作为车载网中车辆的个人账户，利用智能合约的定制功能，实现个人账户与交通违法行为或税费事物的自动交互，从而实现自管理功能，证明了以太坊平台可以应用于交通领域，但没有实现支持真实交通数据的车辆调度系统。

文献[?]的研究基于 Hyperledger Fabric 平台，提出了一种使用区块链技术的创新方法，避免了使用第三方软件企业的服务，从而在固定地理区域内实现动态导航路由，同时确也能保用户的匿名性。其不足之处在于，一是作为区块链系统要经常跟第三方库打交道，比如动态路径的生成和改变就依赖在线 OSM 库，可能导致信息泄露；二是没有充分考虑性能问题，可以利用特殊的地理数据格式来简化算法和存储。采用基于 GeoHash 的地理信息做存储、导航和调度是不错的改进。

2.4 路径规划算法

路径规划即通过对道路结构的分析找到最短的路径，最短的概念包括很多维度，比如单纯的从起点到终点的路径长度最短，或者路径的通行时间最短，或者是路径上的路口和交通灯最少等。出租车司机可以根据自己的需求来选择不同类型的最短路径。近几年随着自动驾驶汽车和智能交通系统等概念的炒作，最短路径的求解这一问题的研究热度也随之提升，现有的最短路径规划方法主要分为静态路径规划和动态路径规划两种。

静态路径规划算法，即在服务器存储的静态矢量网格地图中寻找从起点到终点的最短路径，可以基于不同的道路权值，计算出该类权值下的最优路径。Dijkstra 算法是此类算法的鼻祖[?]，Dijkstra 算法属于单源最短路径算法，算法规则是从起点开始扩张，向外搜索到达所有节点的最短路径，其时间复杂度是 O(n^2)，文献[?]的研究用 Dijkstra 算法规划了最短路径，设计了基于即时乘车需求和实时路况的路径规划算法。此外，经典的静态路径规划算法还有 Floyd 算法[?]、A* 算法[?]，Floyd 算法使用的是两个二维数组，它们分别代表图中所有的顶点到所有顶点的最短路径值的总合，以及对应顶点的最小路径的上一跳顶点，此算法考虑到了两点间距离是负值的情况，该算法复杂度是 O(n^3)；A* 算法在 Dijkstra 算法的基础上使用了启发函数，它结合了 Dijkstra 和启发式算法的优点，以从起点到中间计算点的距离加上该中间点到终点的估计距离之和作为该中间点在优先队列中的优先级，该优先队列在算法执行过程中用来选择中间点的下一跳，A* 算法与 Dijkstra 相比明显降低了时间复杂度。

动态路径导航规划方法最早由 Cooke and Halsey[?] 提出，动态路径规划根据特定

类型的道路权值求解起点到重点之间的最短路径，同时实时维护道路信息，对预先规划好的最短路径进行合理的调整，如此维护导航结果直到车辆到达目的地，最终的行车路线即为最优的路径。文献[?]将实时交通信息动态地记录为该路段的通行时间，以此数据来支持路径规划。

文献[?]的研究将遗传算法的思想应用到了动态路径规划中，提出了基于动态交通网络模型和遗传算法的最优路径选择算法。但是遗传算法的初始化种群染色体适应度（即连接首尾的路径开销）、“轮盘赌”选择法都增加了不必要的算法负担。且要一直运算到种群中连续 N 代个体的适应度值基本保持不变，才可选择最优路径，算法开销过大，不适合大规模路段。与 Dijkstra 算法相比，遗传算法在大型的网格地图中导航容易导致性能瓶颈[?]。

即使是基于实时路况进行路径计算的动态导航，也存在实时交通信息获取的可靠性和准确性问题，以及数据的高效存储和服务器的实时响应等问题，这些问题导致动态路径规划的算法体系不够成熟完善，路径规划的反馈结果不够及时和准确。理论上，遵照这些最短路径算法的结果，就可以得到最优路径的走法。实际上，驾驶员更愿意根据即时有效的路径规划结果来自己判断更有效的路径。本文在基于以太坊的树状区块链应用中实现了静态矢量地图路网的存储，基于此采用 A* 算法的思路设计路径规划算法，不同于传统的 A* 算法，本文在使用 A* 算法进行路径规划时支持了 GeoHash 的数据格式。A* 的启发函数基于 GeoHash 格式进行运算，简化了相比基于经纬度数据的运算量，优化了服务端的存储性能和算法响应速度，更适合分布式和安全性兼顾的出租车调度系统的设计。

第 3 章　基于树状区块链的出租车调度系统框架

本章介绍了本文所实现系统的框架和流程结构，首先是对系统的架构进行了介绍，包括区块链后台的结构功能，和车辆端和乘客端的终端功能构成；然后对系统的运行流程进行了介绍，包括乘客端的业务运行流程和车辆端的业务运行流程设计。本系统采用本章所描述的结构和流程，结合真实地理数据的矢量地图，模拟了真实地理环境，使系统的运行结果更接近真实环境下的运行情况。

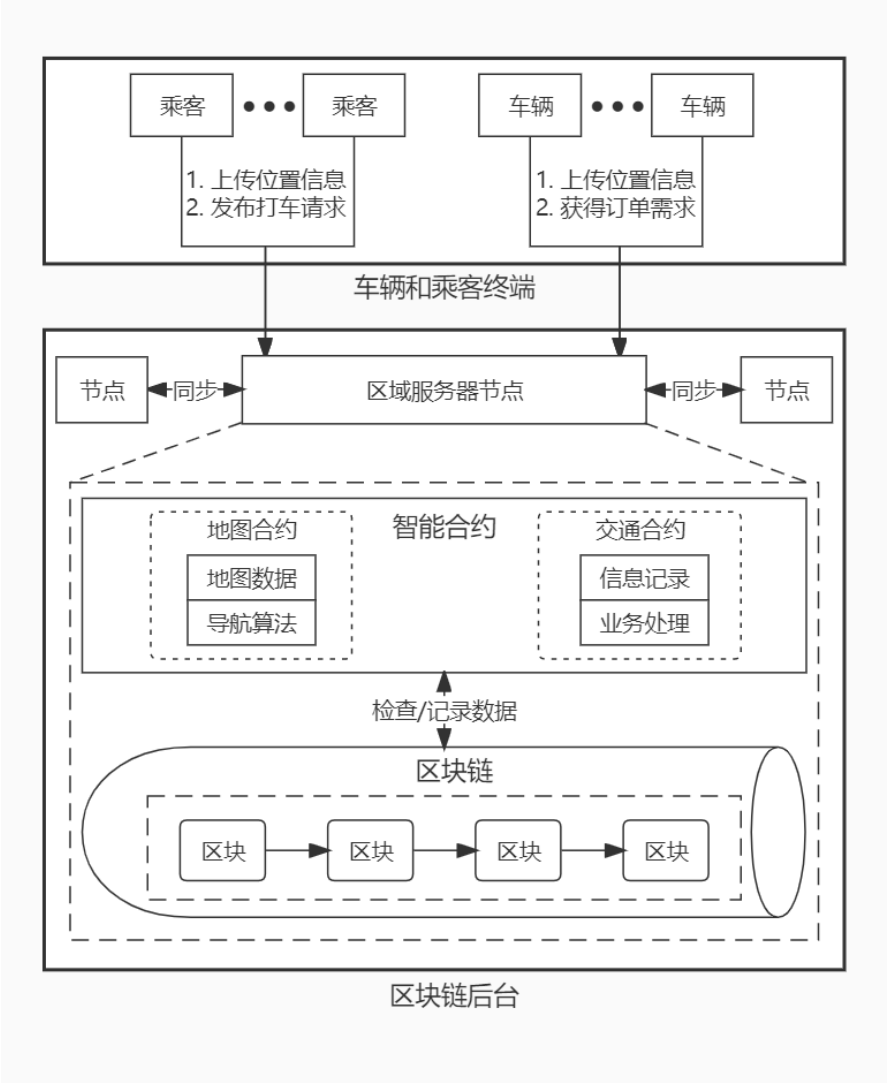

图 3.1　出租车调度系统架构图

3.1 基于树状区块链的出租车调度系统架构设计

区块链后台是指按一定规律分布在城市交通系统中的路边服务节点，其上运行着自动化执行任务的智能合约。因系统后台是基于树状区块链的，故其可以通过地理信息相关的功能更好地支持整个系统。乘客和车辆的终端分别执行发单和接单的功能，以此和区块链后台一同构成完整的调度系统，如图**??**所示。乘客和车辆随身运行的终端与其所属区域内的节点服务器进行交互。

3.1.1 区块链上智能合约端模块

智能合约运行在树状区块链平台上，根据负责的功能不同分为两种合约，第一种是负责记录地理信息和执行路径规划功能的合约，本章中称其为地图合约；第二种是记录乘客和车辆的身份信息和位置信息的合约，本章中称其为交通合约。将智能合约按功能的类型分开，实现了服务端模块之间的解耦，方便了区块链系统的维护，增加了系统的稳定性和可靠性。

地图合约主要由地图存储模块和路径规划算法模块两个模块构成：

（1）地图存储模块：部署地图合约后，地图存储模块接收 JS 脚本上传的 GeoHash 矢量地图，将地图数据与不同前缀长度的 GeoHash 区域进行绑定，同时，将地图的三叉及多叉路口连接的道路进行存储，以方便路径规划算法在执行过程中对路口所连接道路的查找，终端可以根据 GeoHash 区域进行查询，向地图存储合约获取对应的矢量地图并进行展示。

（2）路径规划算法模块：地图合约的路径规划算法模块，是系统进行路径规划的核心部分，其接收一个起点 GeoHash 参数和一个终点 GeoHash 参数，通过改进的 A* 算法进行路径规划运算并返回结果。其中，一对起止点 GeoHash 既可以由空载车辆当时所处的位置（GeoHash 表示）和乘客的上车点位置所组成，也可以由乘客的上车点位置和乘客的目的地位置所组成。路径规划算法返回的结果是一个一维数组，其元素是最短路径上所有路段的起止点。

除去地图存储和路径规划两个模块外，地图合约还提供了修改算法参数的接口。在以道路长度为权值的路径规划算法中，可以通过接口修改参数从而找到计算性能和准确性最优的参数。除此之外，当交通场景需求发生变化时，需要为导航算法引入不同类型的权值进行路径规划，此接口也可以方便参数的修改，以方便路径规划算法计

算不同权值类型下的最短路径结果，拓展了路径规划模块的健壮性。

交通合约主要由信息记录模块和调度处理模块这两个模块组成：

（1）信息记录模块：在树状区块链上部署交通合约后，车辆和乘客登录终端，然后在智能合约的信息记录模块初始化自己的账户信息和位置信息。在初始化时乘客将自己的乘车状态设置为未乘车，车辆将自己的载客状态设置为未载客。信息记录模块同时也能改变乘客的乘车状态和车辆的载客状态，方便调度处理模块确定可调度的车辆。完成订单后，信息记录模块还会通知相关车辆乘客的支付状态，方便车辆司机判断是否可以去接下一单需求。此外，信息记录模块还给司机提供了退出系统的功能，即车辆司机需要休息或者遇到特殊情况时，可以适时退出调度系统，避免被分配到未来的需求订单。

（2）调度处理模块：交通合约的调度处理模块主要是将乘客和距离乘客最近的空车做匹配。乘客提交打车请求后，调度处理模块会根据乘客的位置，遍历乘客所在区域和其周围相邻的区域，寻找距离乘客最近的空车，然后将其分配给乘客。车辆在接到订单并同意后，调度处理模块会将车辆状态改为已载客。总之，调度处理模块主要完成了寻找区域内最近的空车和修改车辆和乘客状态这两个功能。

3.1.2 浏览器终端模块

乘客在浏览器终端通过自己的账户信息进行登录，并初始化自己所处的位置（以GeoHash 表示位置），然后根据自己的位置通过 web3 接口从智能合约获取自己位置周边的矢量地图数据，在前端进行展示，乘客可以在渲染的地图中看到自己所处的位置。浏览器端提供了可以使乘客在智能合约记录出发点和终点的模块，然后乘客可以通过调度车辆的功能来向智能合约申请调度车辆，调度成功的车辆位置会在乘客端进行显示。在完成业务后，乘客通过 web3 接口进行交易确认，同时智能合约会通过监听来确认交易，在车辆确认完成订单后将车辆的状态改为未载客。

车辆在浏览器终端通过自己的账户信息进行登录，并初始化自己所在的位置（以GeoHash 表示位置），车辆根据自己的位置通过 web3 接口从智能合约获取周边的矢量地图数据，在前端进行展示，可以看到自己的位置信息。如果司机需要休息或者遇到意外情况，浏览器还提供了退出按钮供车辆退出调度系统，以免接到新的需求订单。在接到乘客订单时，车主可以看到乘客的位置，判断能否顺利接到乘客，如果有困难无法接客可以选择拒绝接受此订单。在确认接受订单后，车辆可通过 web3 接口向智

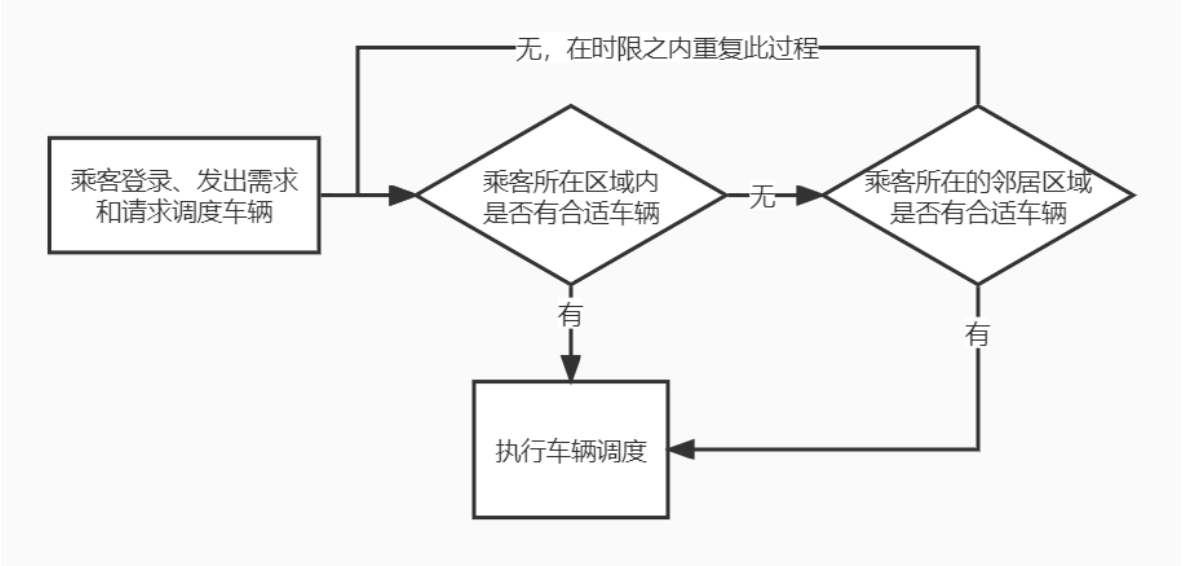

图 3.2　区域调度车辆流程

能合约申请到乘客上车点的路径规划；在确认乘客上车后，车辆可通过 web3 接口向智能合约申请到乘客目的地的路径规划。

3.2　出租车调度系统流程设计

首先乘客通过浏览器终端发出乘车请求，将自己的起点和终点传到智能合约进行记录。服务器端的智能合约会记录乘客的起始点位置，通过起点的 GeoHash 获得乘客所在的 GeoHash 区域，据此进行区域调度（流程如图**??**）来寻找最近的空车，若乘客所在的六位 GeoHash 区域内没有合适的车辆，则通过 GeoHash 邻居算法找到乘客起点相邻的 8 个区域，在这些区域内寻找合适的车辆，然后将最佳匹配的空车的信息通过 web3 回调返回给乘客端。

乘客端在收到车辆信息后，会通知智能合约修改车辆的载客状态和自己的乘车状态。智能合约在接收到请求后会通过 assert 检查该车辆的载客状态（如图**??**），即检查同一车辆是否在短时间内被分配给了不同的乘客。如果状态检查正常，则智能合约认为车辆和乘客匹配成功，顺利修改乘客的乘车状态和车辆的载客状态；如果状态检查不正确，则智能合约认为这次的车辆调度未成功，会通知乘客端重新进行呼车操作，此过程可隐式进行，方便乘客的操作。通过上述流程可以避免同一车辆被同时分配给多个乘客的错误。

车辆和乘客匹配成功后，相关的车辆会从智能合约监听到自身的状态变化，即智

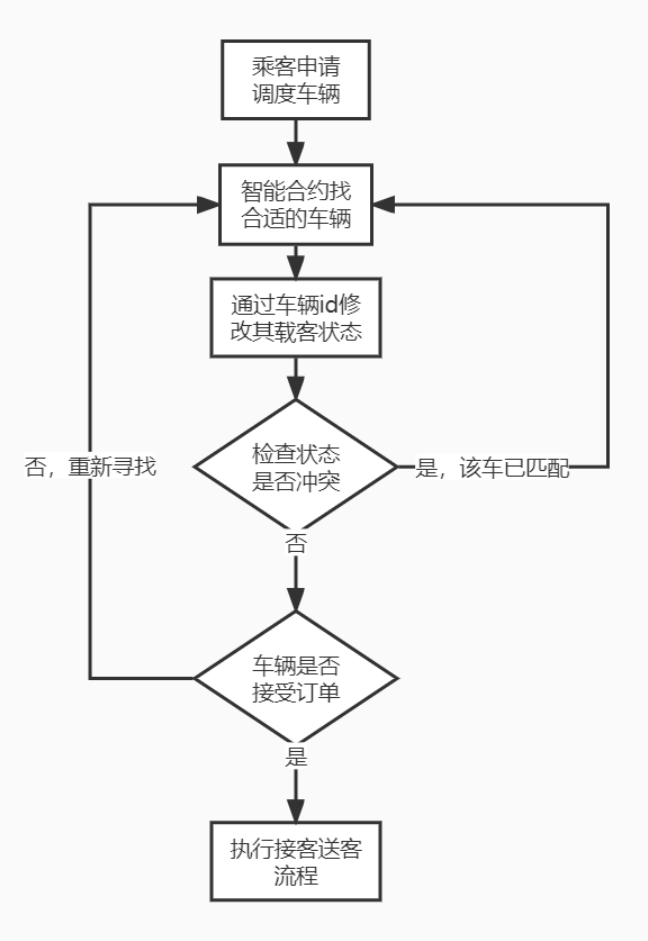

图 3.3　解决调度冲突流程

能合约会把乘客的订单通知到对应的车辆。此时司机可以从终端地图上观察到乘客的位置，同时根据自己的条件和意愿选择是否去接送乘客：如果选择不接送乘客，则会将此选择反馈给智能合约，智能合约会通知乘客订单已经被取消，乘客需重新执行呼车流程，同时智能合约也不会再向原车辆提供乘客相关的信息。

如果车辆确认接送乘客，则如图**??**所示，服务器端的智能合约会根据车辆位置和乘客的上车点执行路径规划，将路径规划的结果进行存储，并将结果通知给车辆端和乘客端，确保双方获取的路径规划结果一致，车乘双方均可在自己的终端地图观察到路径规划结果。车辆会根据路径规划结果去上车点接到乘客。乘客上车后，车辆确认接到乘客，智能合约会再次执行路径规划算法，通过乘客的上车点和终点得到最优路径，将路径规划结果进行存储，并通知车辆端和乘客端，在双方的地图上显示路径规划结果。车辆会根据最优路径将乘客送到终点。

车辆送乘客到达终点后，乘客执行确认到达终点的操作，表示车辆已经按路径规

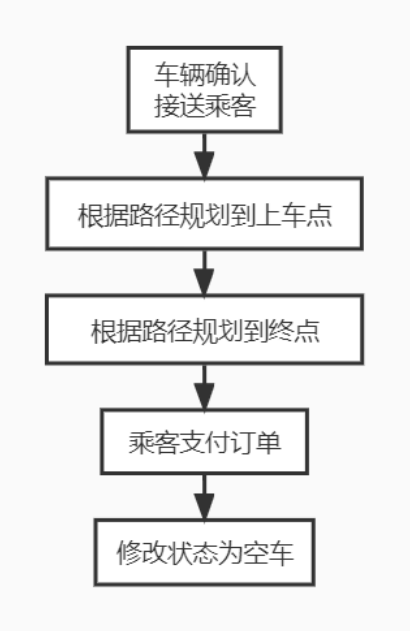

图 3.4　车辆执行业务流程

划结果完成任务。智能合约在接收到乘客的确认到达后，会将乘客的状态从已乘车改为未乘车，将车辆的状态从已载客改为未载客，并更新空车的位置，到此完成了一轮完整的业务流程。车辆在收到智能合约的状态改变通知后，可以根据自己当前的地理位置继续等待下一个订单任务。

3.3　系统自动化运行框架的设计

自动化运行框架基于 python 语法和 selenium 库实现，通过自动化操作浏览器运行来模拟车辆终端和乘客终端的行为（如图**??**）。

乘客端的自动化 python 脚本可以通过设置参数来决定打车乘客的数量。自动化脚本控制每个乘客根据自己的账户信息登录系统，然后按照各自的起止点需求和发布时间来提出打车请求。在智能合约成功记录乘客的起止点需求后，自动化脚本监听到反馈，会控制乘客端开始申请车辆调度。匹配到的车辆在确定接单后，会前来上车点接到乘客，并将乘客送到目的地。自动化脚本监听到乘客到达目的地，会执行确认和支付订单的操作，然后退出浏览器终端，视为乘客完成了一次完整业务。

车辆端的自动化脚本控制每辆车根据自己的账户 id 登录系统并初始化自己车辆的 GeoHash 位置，在智能合约进行记录。自动初始化完成后，车辆即可通过监听智能合约的通知来获得乘客订单并选择是否确认接受。车辆可根据即时的乘客需求来执行

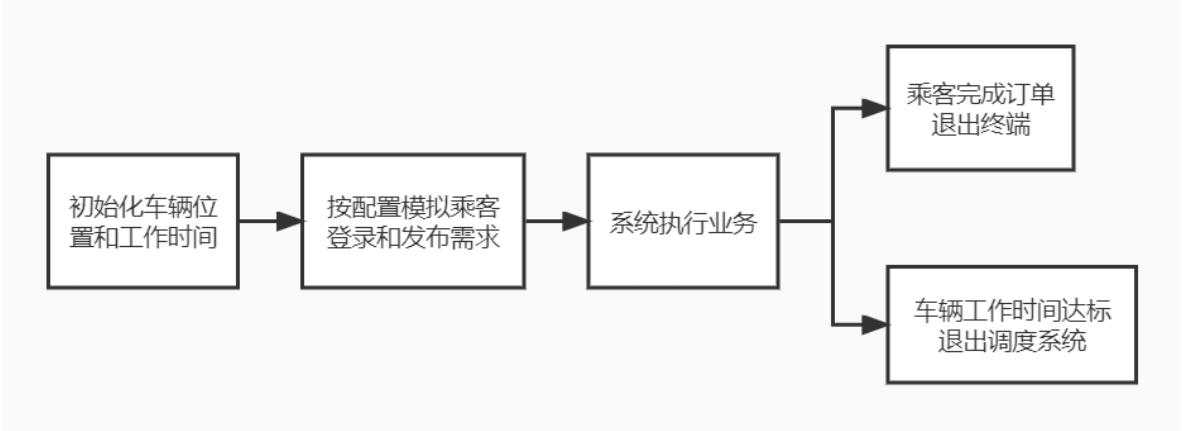

图 3.5　自动化模拟流程

业务流程。此外，自动化脚本可以控制车辆的工作时长，超出工作时长后，相关车辆将会退出系统，智能合约也不再为其提供乘客订单。

第 4 章　系统的工具和算法原理

本章详细介绍系统的工具、功能、算法开发工作和原理。1 段

本章对调度系统的区块链服务端和用户端涉及到的算法和工具进行了详细的介绍。服务器端主要涉及地理信息的存储和基于地理信息的算法工作，包括 GeoHash 静态矢量地图的存储，基于 GeoHash 静态矢量地图的路径规划算法，筛选最近的空车来与乘客进行匹配的车乘匹配算法。在车辆和乘客的用户端，主要是基于 leaflet 工具进行 GeoHash 矢量地图的展示，以及添加相应的功能来优化展示效果。

张禹：14 页 1 段 4.1 低采样率下的服务器端路况挖掘 1 段 4.1.1 近似算法（一图）5-6 段 4.1.2 在线道路匹配算法（三图）6 段 4.1.3 路口模型描述 3 段 4.1.4 路况计算（一图四公式）6 段 4.1.5 缺失路况的补充（一公式）4 段

4.2 3 段 4.2.1 基于 GeoHash 的地图存储 4 段 4.2.2 网页端路况挖掘（两图）6 段 4.2.3 不足与改进方向（一图展示页面）6 段 4.3 小结 1 段

共 47 段

4.1　基于 GeoHash 的矢量地图展示

对系统开发过程中将基于 GeoHash 的矢量地图信息存储在区块链和展示在浏览器端的开发工作和原理做介绍。1 段

由于终端通信设备的普及和迭代，自组网系统的应用规模不断增大，基于中心化服务器的业务处理速度无法满足网络请求响应的实时性，也限制了自组网车辆之间的通信。而利用区块链的特性实现的分布式后台，既可以保证交易记录的安全性，防止恶意攻击者对地理信息和位置信息、交易信息的篡改，还可以方便成员之间的相互通信，同时其分布式的特性也能够满足对大规模网络请求的及时响应。

本节将介绍 GeoHash 矢量地图在区块链中的存储逻辑，同时还会介绍在终端页面实现的对 GeoHash 矢量地图的渲染逻辑，以及为了优化地图展示效果补充的放缩和拖动功能的实现逻辑。

4.1.1 基于 GeoHash 的地图在区块链上的存储

将矢量地图数据以 GeoHash 形式存到区块链上的原理、优点分析和改进空间。6 段

GeoHash 是一种新型的地址编码方式，由 Gustavo Niemeyer 和 G.M. Morton 发明 [董斌 37]，其具体算法是二分法结合编码的一种地理位置信息算法 [董斌 16]，被广泛应用到地理位置表示中。最开始，以本初子午线、赤道为界，地球可以分为 4 个部分，设定西经为负，南纬为负，所以地球上的经度范围就是 [-180,180]，纬度范围就是 [-90,90]。GeoHash 使用了 Peano 空间填充曲线（一张图），在处理地图数据的过程中，要将经纬度点转换成 GeoHash，首先要通过二分的方法把经纬度转化成二进制编码，以 (116.3639092,39.9662639) 为例，将其转化为精度为 8 位的 GeoHash，过程如下表 3-1 所示。将经度和纬度分别转换成一组二进制字符串，然后将经度和纬度的二进制字符串以位置交叉的形式进行组码，生成一组新的更长的二进制字符串，新的二进制字符串中偶数位的数字对应原来的经度二进制串，奇数位的数字对应原来的纬度二进制串，然后将新字符串每五位一组转为十进制数字，并根据数字找到其在 Base32(即数字 0-9 和字母 b-z 不包括 a,i,l,o 的 32 个字符) 数组的映射，以此进行编码 [董斌 16]，最终实现用 GeoHash 一维数据表示原经纬度二维数据的效果。（一张表用来示例）

根据 Geohash 特殊的编码方式，一个 GeoHash 值对应一个近似矩形的覆盖区域，当位数较短时，它可以表示一块区域的坐标，当位数足够长时，它可以表示地球上某个具体点的坐标，重要的一点是，GeoHash 位数短的区域一定包含以这个 GeoHash 为前缀的所有地图数据（一张图示例），这一特点对地图数据进行区域绑定提供了极大的便利。GeoHash 在编码过程中保留了一定的相对地理位置信息，在大部分情况下，GeoHash 编码共同前缀越长的区块物理距离越近 [董斌 37]，一个 6 位 GeoHash 块的覆盖区域大小为 1220m*1220m。传统的地图数据采用经纬度的方法进行存储，不方便实现位置与区域绑定，并且二维数据存储内容较大，对网络环境和存储设备的要求较高。采用 GeoHash 字符串存储地图信息可以使所有信息平面化，更方便区域信息绑定，并且一维字符串降低了存储数据的规模，同时基于 GeoHash 地理信息的运算复杂度较低，也更适用于 Solidity 语言。同时，根据 GeoHash 编码规则，一个 GeoHash 值所代表的区域一定包含以该 GeoHash 值为前缀的所有元素。

车辆和乘客的终端在使用时需要从区块链中读取地图信息。为了降低地图数据对终端设备的内存要求，以及降低对网络资源的占用，在区块链存储地图时可将地图进

行分块处理，这样终端不用一次读入所有的地图信息。同时，分块后的道路会按照其GeoHash 位置的前缀绑定到相应的 GeoHash 区域内，这样终端在获取地图时能够根据用户的 GeoHash 位置读入该 GeoHash 所属的大范围 GeoHash 区域的地图信息，而不用遍历所有的地图数据，在提高获取效率的同时还会减少网页端的内存消耗，也便于终端地图信息的更新。（一张区域绑定示意图）

智能合约中存储的道路信息包括道路 ID、道路起点位置、终点位置、道路路径上的点构成的数组（道路为折线段结构）、标识道路间连接信息的 ID（每条道路只在首尾处与其他道路相接）、道路名、道路长度以及单双向行驶标识。道路信息属于静态数据，在存储到地图合约的过程中，会对道路末尾连接处所能到达的所有道路的信息记录成一个数组映射，以方便路径规划算法在运行时对下一条可达道路的启发式查找。另外，由于开发语言不支持浮点数，因此距离信息需要用 GeoHash 距离运算的整数结果来表示，此处需要调整运算过程中的参数来找到准确度和运算效率双优的参数。（一张表）

4.1.2 基于 GeoHash 的矢量地图的终端数据显示

画一个流程图，按流程图描述（一图）1. 计算当前层级 GeoHashTile 的 size 2. 计算当前屏幕内 GeoHashTile 的个数（一共 5 个公式）3. 计算当前屏幕内所有 GeoHashTile 的编码（简单提及）4. 向服务端请求地图数据（简单提合并的过程）5. 压缩数据，计算 GeoHashTile 的中心 GeoHash（简单提）6. 计算像素距离（简单提及）7. 计算地图对象的屏幕坐标（1 个公式）8. 将 tile pixel distance 和 coordinate pixel distance 做缓存（简单提及）

[周畅] 的 GeoHashTile 工作设计了基于 GeoHash 体系的矢量地理数据结构——GeoHashTile，利用了 GeoHash 对矢量地理数据的高效分区和一维索引，方便对地理数据的查询，并且其支持了对 GeoHash 地理信息的显示，实现了 GeoHash 坐标和屏幕上像素坐标的直接转换，并且验证了 GeoHashTile 在请求数据的速度上比基于经纬度的 GeoTile 系统有明显优势。其 GeoHash 转换成屏幕坐标采用的是相对投影的方法。

终端完成显示 Geohash 矢量地理数据的过程，如图 5 所示，主要包括三个部分：（1）计算 GeohashTile 的数量和编码，包括三个步骤：首先获得当前缩放级别下一块 GeohashTile 的像素大小（每个级别的瓦片像素大小已经通过推理计算设置好 [周畅论文]），然后通过屏幕的长宽像素计算客户端中 GeohashTile 的数量，得到数量后，再通过中心

瓦片的 GeoHash 编码和查找邻居瓦片的算法，计算终端屏幕中所有 GeohashTile 的编码。(2) 地图数据请求发送到服务端之前，为了减少请求报文的规模，使用 GeohashTile 请求合并算法来实现要发送的请求的合并。服务器收到请求后会查找请求的所有区域的数据，并整理成基于 GeoHash 的 GeoJSON 数据返回给终端。(3) 终端收到地图数据后会进行投影显示，相对位置投影过程包括数据压缩、计算像素距离和计算屏幕坐标三个步骤。实现投影的第一步是计算从这些目标点到中心点的相对像素距离列表。有两种相对距离计算（如图 5 中的步骤 6 所示）。第一个是计算从各个 GeohashTile 中心像素点到屏幕中心像素点的相对像素距离。第二个是计算从目标点的像素点到其所属 GeohashTile 中心像素点的相对像素距离。将这二者相加即可得到目标点的在屏幕中的像素位置，通过画点接口进行投影。

GeohashTile 的计算过程是根据给定的中心位置 center$_geohash$□□□□□z□□□□$clientSize$□□□□

4.1.1. 计算一块 GeohashTile 的大小，如步骤 1 所示？如图 5 所示。在瓦片地图系统中，瓦片的索引遵循四叉树原则。在每个缩放级别，等式（2）用于计算在 x 轴和 y 轴方向上的瓦片数量，其中 z 是缩放级别。公式 (2) Geohash 编码的每多一个字符就代表将一块区域进行更深层的子分区，一个区域根据编码的奇偶字节交替划分为 4×8 或 8×4 子区域。每次分区后对应的 x 轴和 y 轴方向上的 GeohashTile 编码数量的计算公式为公式（3），其中 l 是 Geohash 编码的长度。公式 (3) GeoHashTile 的缩放级别 z 与 GeoHash 的字符长度 l 并不是严格相关的。由于 GeoHash 每一层的分区都多于 GeoTile 系统中的四叉树分区，因此，我们结合方程（2）和（3）得到 Geohash 编码长度 l 和缩放级别 z 之间关系的方程（4），在满足公式 (4) 的基础上，使用最短长度的 Geohash 编码来做瓦片，以减少存储。对于每个层级 z，l 的计算结果取最小整数。公式 (4) 无论谷歌地图、必应地图或其他地理信息显示系统如何，每层的分幅都是固定的像素大小（最常见的分幅像素大小为 256×256）。从上面可以看出，Geohash 每一层的瓦片大小都是相同的，并且层与层之间的瓦片大小根据划分规则有规律地变化。由于 x 和 y 方向的分割大小不一致，Geohash 瓦片大多为矩形。为了使 GeohashTile 在每个级别上近似于 256×256 的平方像素大小，并便于计算，GeohashTile 的像素大小在缩放级别 0 设置为 512×512。结合方程式（4）的计算结果，可通过方程式（5）计算相应缩放级别 z 下的 GeohashTile 大小。在等式（5）中，z 是缩放级别，l 是 Geohash 编码长度。由于 Geohash 在 x 和 y 方向上的除法规则不一致，因此需要两个不同的方程来完成计算。公式 (5) 表 3 显示了通过等式（5）计算的 GeohashTile 的像素大小，

其中缩放级别为 1–18。除了能够计算具有不同编码长度的 GeohashTile 的像素大小之外，等式（5）还用于稍后计算 Geohash 坐标点转换屏幕像素坐标。表 3 4.1.2. 计算客户端中 GeohashTile 的数量在获得相应缩放级别中 GeohashTile 的大小后，可以通过组合客户端像素大小来计算屏幕中覆盖的 GeohashTile 的数量，以便为从服务器获取相应的地图数据做准备（如图 5 中的步骤 2 所示）。方程式（6）是计算地砖坐标 x 轴和 Y 轴方向上的 GeohashTile 编码数量的方程式，其中大小为。x、尺寸。y 是客户端屏幕的像素大小。公式 (6) 为了确保 GeohashTile 覆盖屏幕，计算结果将被四舍五入。这也是图 5 所示 GeohashTile 覆盖范围超出屏幕的原因。

4.1.3. 计算客户端中的所有 GeohashTile 编码以从服务器获得相应的地图数据，我们应该计算屏幕覆盖范围中的所有 GeohashTile 编码（如图 5 中的步骤 3 所示）。

4.3 相对位置投影处理所有地图数据都需要从球面数据投影到二维平面数据进行显示。本节描述的 Geohash 地图数据的相对位置投影过程是使用相对位置计算方法将 Geohash 编码的地图数据直接投影到屏幕坐标的过程。具体计算步骤如下：

4.3.1. 编码长度和缩放级别的关系公式 (7) 公式 (8) 这里我们有两个定义来帮助解释。定义 1 定义 2 公式 (9) 4.3.2. 计算相对像素距离算法 3

4.3.3. 客户端屏幕中心点像素 +tile 中心点像素偏移 +coord 相对 tile 中心点的偏移

4.1.3 基于 GeoHash 的矢量地图实现放缩和拖动功能

上一节主要讨论了 GeoHash 矢量地理数据在终端的的投影过程，其核心工作是将 GeoHash 字符串转投影为屏幕上的像素坐标。本节讲述实现放缩和拖动功能的原理工作，其核心部分的实现逻辑与上一节所述内容相反，实现放缩和拖动功能需要重新设置地图视野的中心点 GeoHash，这需要将鼠标在屏幕上的像素点位置转化成相对应地图图层的 GeoHash 地理位置。（逻辑流程图）

实现 GeoHashTile 的拖动功能，即通过监听终端操作获得给定的像素值变化，然后对地图进行平移；实现放缩功能，即通过监听终端操作获得缩放等级的变化，然后将地图展示成新的缩放等级。放缩或者拖动结束之后，需要重新获取视野的中心点 GeoHash 字符串，这个工作是将变化后的地图图层的中心点投影成 GeoHash 字符串，需要实现将屏幕的像素坐标投影成 GeoHash 字符串的功能。

首先遍历屏幕内的所有 GeoHashTile 瓦片，通过（算法）根据每个瓦片的范围判断操作点所在的瓦片，然后获得该瓦片的中心点 GeoHash 和中心点像素位置，通过

操作点的像素位置和操作点所在的 GeoHashTile 的像素位置，可以通过（公式 1）计算出两点之间的相对像素差，根据相对像素差，结合本缩放层级下的瓦片像素大小，可以通过（公式 2）计算出操作点和瓦片中心点在该精度下的 GeoHash 在东西和南北方向上的地理偏移量（以瓦片中心点 GeoHash 的精度下所对应的 GeoHash 格子数表示），根据操作点相对于瓦片中心点的 GeoHash 地理偏移量，可以通过（算法 1）计算出操作点的 GeoHash，此 GeoHash 的精度与瓦片中心点 GeoHash 相同。（放大缩小示意图）

$7086_o nDragEnd- > map.panBy(offset)$□□□□□□□□□□□

4.2 基于 GeoHash 的几何计算优化

3 段由于传统计算两个经纬度所表示坐标点距离时需要使用球面距离公式，若在以 GeoHash 为坐标表示的系统中沿用这套算法，则丧失了 GeoHash 带来的计算简便性。利用 GeoHash 编码的特点进行距离计算，避免了复杂的三角函数和球面计算，并且适用于对小数支持较弱、不提供复杂数学函数计算支持的区块链智能合约编写语言 Solidity。

4.2.1 GeoHash 几何计算原理

具体介绍 GeoHash 数格子进行几何距离计算的原理。

在将经纬度编码为 GeoHash 时，首先要将经纬度分别通过二分法编码为对应长度的二进制编码，然后将经度和纬度的二进制编码按照偶数位放经度的二进制编码、奇数位放维度的二进制码的方式进行组码，最后将新获得的组码串进行 Base32 编码，即可获得 GeoHash 字符串。

在上述编码的过程中，经纬度二分后对应的两个二进制编码，其对应的十进制值也有特殊的地理意义：即在当前要编码的 GeoHash 精度下，这一对经纬度对应的 GeoHash 瓦片，相对于赤道原点处所偏移的相同规模的 GeoHash 瓦片的数量。经度对应的二进制编码的十进制值表示在东西方向上偏移的瓦片数量，纬度对应的二进制编码的十进制值表示在南北方向上偏移的瓦片数量。

根据上述原理，在计算两个 GeoHash 位置之间的距离时，可以先将两个 GeoHash 各自解码为在东西和南北方向上表示偏移的数值，再计算这两者在东西和南北方向上

偏移的数值之差，即可得到两点之间在东西和南北方向上的相对位置差。由于经纬度划分的原因，在一定的 GeoHash 精度下，所有瓦片的南北距离都是相同的，但东西距离会根据 GeoHash 点所在纬度的增加而减小。在 10 位 GeoHash 精度下，每个瓦片的东西宽度，可以按纬度的不同大致划分为 16 个区域。故两个 GeoHash 点之间的南北距离可以根据南北瓦片的偏差值乘以当前 GeoHash 精度下瓦片的南北距离（单位：米）获得；而两个 GeoHash 点之间的东西距离需要根据 GeoHash 所属的纬度区域找到 GeoHash 瓦片的东西距离（单位：米），再乘以两者的东西瓦片偏差值获得。

4.2.2 GeoHash 几何计算方法的优化

在实际的出租车系统应用环境中，两个终端之间的距离计算所涉及到的地区范围，通常情况下不会超过市辖区以上的规模。故在出租车系统的环境下，两个终端的地理位置 GeoHash 会有一定位数的前缀是相同的（通常在 5 位到 6 位，画一个 GeoHash 规模对应格子大小的表格）。据此，这两个 GeoHash 解码后的二进制偏移值的前几位也会相同。事实上二者相同前缀所转化的高位数字是相同的，做减法之后会相互抵消。这些相同前缀转化的的高位数字对二者相对偏移的计算没有影响，这就造成了在 GeoHash 解码时对计算资源的浪费。

在地理视角上，计算两个 GeoHash 之间的距离，首先是对 GeoHash 进行解码计算出二者相对赤道原点的偏移，如果计算时不考虑两个 GeoHash 前缀相同的部分，此处的参照对象就会从赤道原点改为同时包含这两个 GeoHash 瓦片前缀瓦片（即二者相同前缀部分的 GeoHash 所表示的瓦片）原点，以距离二者更近的原点作为参照，可以减少计算瓦片偏移数量时的运算规模，从而加快运算速度，使距离计算的算法性能得到提升。（优化示意图）

本文在进行 GeoHash 距离计算时，首先检查两个 GeoHash 的前缀相同的部分，然后取其不同的部分进行距离计算，以优化距离计算的性能，利用了 GeoHash 字符串的特性，加快系统的响应速度。算法的逻辑如下（算法），具体的性能优化效果在第五章进行描述。

4.3 基于 GeoHash 的路径规划算法

为了完善去中心化的出租车调度系统，需要在智能合约端实现后台的路径规划算法。路径规划算法需要根据智能合约上存储的 GeoHash 矢量地图数据进行静态导航，

参考 GeoHash 地图存储的格式，在进行地图存储时根据路口的连接关系建立路口之间的拓扑连接关系。本节采用优化后的 GeoHash 距离计算方法，结合常规静态路网规划算法 A* 算法，提出了基于 GeoHash 矢量地图的路径规划算法，为车辆提供路径规划服务。本节实现的路径规划算法可以根据不同的道路权值类型进行路径规划，还可以调节算法参数，在保证路径规划准确性的前提下提高效率。同时，为避免不必要的服务纠纷，会对路径规划的结果进行记录。

路径规划算法的提出和发展由来已久，有适合在未知地图环境下运行的启发式路径规划算法，可以应用在智能机器人和无人车等领域，此外，还有可以在已知地图信息的情况下，利用已有的矢量地图数据规划出最短路径的路径规划算法，可以应用在地理信息实时更新的交通系统中。传统的路径规划算法，是在已知地理信息的情况下进行最短路径的规划，可以应用在车载应用的路径规划、公共交通实时路径规划等环境中。启发式路径规划算法，适合在不知道地理信息的情况下进行主观的路径探索，这种算法应用在机器人的自动寻路、游戏中的 AI 角色寻路等场景；本文所描述的系统已有静态矢量地图拓扑信息的支持，故采用静态路网的路径规划算法为车辆提供路径规划的辅助。

4.3.1 矢量地图路径路径规划算法的性能对比

Dijkstra 算法是由 E.W.Dijkstra 于 1959 年提出，属于贪心算法模式，是非常典型的最短路径算法。此算法可以用于求得移动机器人行进路线中的一个节点到其他所有节点的最短路径。算法的主要特点是：以输入的起始点为中心，算法过程中生成无向图一直向外扩展，直到扩展到最终的目标点为止，通过节点和权值边的连接关系来构成整个路径。Dijkstra 算法的缺点是效率低，在无向图扩展的过程中会生成大量的无效路径，占用较多的内存空间。

A* 算法（Dijkstra with a Heuristic）是在 Dijksra 的基础上加入启发式函数实现的，因为加入启发算子的缘故，在拓展无向图的过程中可以根据目标点相对中间点的距离协调选择最好的方向进行搜索。相比 Dijkstra 算法的效率更高，占用内存空间更少。A* 算法的应用范围比较广，可以在电子地图进行路径规划，或者在游戏领域中应用于虚拟角色的行走路径的规划，A* 算法的计算结果正确性较高，能够调整算子来适应不同的路径权值类型，拓展了其健壮性和适用范围。

JPS 算法在 A* 算法框架的基础上加入了跳点搜索（Jump Point Search），JPS 会根

据当前点的方向及当前点周围邻居的特点进行选择，某些特殊的点才能执行加入和删除到可探索点集合的操作。在网格地图中，JPS 算法可以用来提高有障碍物时的寻路效率，在遇到障碍物时需要改变搜索方向，这时只把能够以最短路径避过障碍的跳跃点加入可探索点集合中，可以避免考虑很多不必要的路径点，以此来提高算法的搜索效率。但本文的静态地图在储存时已经生成邻居道路的拓扑结构，不需要考虑避开障碍物。JPS 算法在已有拓扑结构的地图中徒增算法的复杂度，故不在本文应用。

CH 算法在 A* 算法的基础上加入了预处理的过程，算法在预处理阶段创建“shortcuts”来实现提速，然后在最短路径查询中使用这些“shortcuts”来跳过“不重要的”顶点。“shortcuts”可用于保存两个重要路口之间预先计算的距离，从而算法无需在查询时考虑这些路口之间的完整路径，这样可以使查询路径缩短，查询效率得到提高。但 CH 算法的预处理使得查询时的道路网络失去了真实的拓扑结构，在每一条道路信息更新后都需要重新对全局进行预处理；此外，道路权值类型的更改也会重新触发整个预处理过程，不能很好地支持实时的交通信息，系统的拓展性和健壮性不足。

4.3.2 基于 GeoHash 的 A* 路径规划算法的原理

A* 算法是一种直接查询算法，只需要地图的拓扑数据，而不需要做任何预处理。同时 A* 算法是一个启发式算法，利用了启发函数，A* 算法的启发函数算子可以记成 h(n)，代表着算法过程查询到的中间节点到目的地节点的距离估计值。A* 算法在选择路径中第 n 个被检查的节点时，还会参考从起始点到第 n 个中间节点的实际距离，称为耗散函数，记为 g(n)。A* 算法的对中间节点的估计函数（如公式）同时参考了耗散函数和启发函数，将二者的和作为评价该节点处于最优路线上的可能性的量度，这样可以首先搜索可能性大的节点。在算法运行的过程中，A* 算法根据评估函数 f(n) 的值，找到距离终点最近的可能性最大的节点，然后展开搜索该节点，而不用每一步规划都需要搜索所有节点，提高了路径规划的效率。

A* 算法的的核心是设置一个合适的评估函数，评估函数的计算方式对搜索结果有决定性作用。对第 n 个中间节点来说，g(n) 的值基本是固定的，其反映的是从起点到当前节点的实际道路距离。因此，启发函数 h(n) 的选择就显得更重要，它能控制 A* 算法计算过程中的行为。如果把 h(n) 设置为 0，这时候 f(n)=g(n)，A* 算法就会退化为 Dijkstra 算法，也能计算出目标路径。启发函数的影响越小，在 A* 寻路时需要探索的节点越多，寻路过程就会变慢。启发函数 h(n) 的值如果设置的比 g(n) 大非常多，

相当于只有启发函数起作用，A* 算法接近 BFS 算法，路径规划的结果不一定是最短路径。常用的启发函数 h(n) 二维坐标下有以下几个距离度量方法：（1）曼哈顿距离，表示在标准坐标系上的绝对坐标轴距之和，

（2）欧几里得距离，也叫欧式距离，表示两点之间的直线距离。

本文的实验部分基于 GeoHash 矢量地图数据，道路属性里的 cost 字段记录的是道路的实际长度，这个长度即是每条路段的两个端点之间的距离，以米为单位。为了简化启发函数的计算，从而支持智能合约语言的特性，本文采用曼哈顿距离作为度量，结合 GeoHash 计算距离的函数进行启发函数的计算。为了使启发函数 h(n) 的结果与实际情况更接近，本文调整了 GeoHash 计算距离函数的参数，使启发函数的结果更接近真实的道路距离，这一步可以让路径规划结果的正确性和算法的效率都得到保障。

4.4 基于树状区块链的区域调度车乘匹配算法

1 段

在

4.4.1 树状区块链对区域信息的查询

4-6 段介绍树状区块链对区域信息的查询原理。

4.4.2 区域调度车乘匹配算法

3 段介绍在树状区块链对区域信息查询的基础上实现的区域调度车乘匹配算法原理、对并发请求的冲突解决逻辑。

[?] 考虑正常交通对出租车的影响，提出了一种出租车调度系统，该系统使用实时交通状况将空置的出租车与最近的（在旅行时间方面）等候的乘客相匹配。

第 5 章　参数设置和系统测试

出租车调度系统的实验环境、本章的工作整体介绍。张禹:实验环境调节 OHMMM 算法边界值（一个流程图，三个数据图）近似算法参数优化，设置容量上限为 200（一图一表）直接路况精度差错率: 带转向延迟的差错率更低，且不带来计算开销（两图）分析了三种车辆异常行为间接补充的路况精度补充路况能够有效提高路况覆盖率（两图）路况结果分析路况统计比轨迹点统计更平滑（一图）统计道路速度（一图）统计路口转向延迟，和道路速度对比关系，证明正确性（一图）展示矢量地图路况（一图）总结

5.1　参数设置实验

astar 导航算法和区域调度算法的参数调节

5.1.1　astar 导航算法参数

5.1.2　区域调度算法参数

5.2　系统测试实验

5.2.1　模拟双行道正确性测试

模拟双行道的导航结果测试。

5.2.2　真实地图数据测试

真实地图的导航结果测试。

结论

总结工作、进行未来工作的展望（结论作为学位论文正文的最后部分单独排写，但不加章号。结论是对整个论文主要结果的总结。在结论中应明确指出本研究的创新点，对其应用前景和社会、经济价值等加以预测和评价，并指出今后进一步在本研究方向进行研究工作的展望与设想。结论部分的撰写应简明扼要，突出创新性。）

附录 A ***

附录相关内容…

附录 B Maxwell Equations

因为在柱坐标系下，$\overline{\overline{\mu}}$ 是对角的，所以 Maxwell 方程组中电场 **E** 的旋度

所以 **H** 的各个分量可以写为：

$$H_r = \frac{1}{\mathbf{i}\omega\mu_r}\frac{1}{r}\frac{\partial E_z}{\partial\theta} \tag{B–1a}$$

$$H_\theta = -\frac{1}{\mathbf{i}\omega\mu_\theta}\frac{\partial E_z}{\partial r} \tag{B–1b}$$

同样地，在柱坐标系下，$\overline{\overline{\epsilon}}$ 是对角的，所以 Maxwell 方程组中磁场 **H** 的旋度

$$\nabla\times\mathbf{H} = -\mathbf{i}\omega\mathbf{D} \tag{B–2a}$$

$$\left[\frac{1}{r}\frac{\partial}{\partial r}(rH_\theta) - \frac{1}{r}\frac{\partial H_r}{\partial\theta}\right]\hat{\mathbf{z}} = -\mathbf{i}\omega\overline{\overline{\epsilon}}\mathbf{E} = -\mathbf{i}\omega\epsilon_z E_z\hat{\mathbf{z}} \tag{B–2b}$$

$$\frac{1}{r}\frac{\partial}{\partial r}(rH_\theta) - \frac{1}{r}\frac{\partial H_r}{\partial\theta} = -\mathbf{i}\omega\epsilon_z E_z \tag{B–2c}$$

由此我们可以得到关于 E_z 的波函数方程：

$$\frac{1}{\mu_\theta\epsilon_z}\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial E_z}{\partial r}\right) + \frac{1}{\mu_r\epsilon_z}\frac{1}{r^2}\frac{\partial^2 E_z}{\partial\theta^2} + \omega^2 E_z = 0 \tag{B–3}$$

攻读学位期间发表论文与研究成果清单

[1] 高凌. 交联型与线形水性聚氨酯的形状记忆性能比较 [J]. 化工进展, 2006, 532 - 535.（核心期刊）

致谢

本论文的工作是在导师……。

作者简介

本人…。