上海交通大学硕士学位论文

# 全自动交通系统车辆调度与人机交互系统研究


硕 士 研 究 生：李爽

学　　号：1130209287

导　　师：王春香副教授


申 请 学 位：工程硕士

学　　科：机械工程

所 在 单 位：机械与动力工程学院

答 辩 日 期：2016 年 1 月

授予学位单位：上海交通大学

Dissertation Submitted to Shanghai Jiao Tong University
for the Degree of Master

# RESEARCH ON VEHICLES' SCHEDULING AND MAN-MACHINE INTERACTIVE SYSTEM IN CYBERNETIC TRANSPORTATION SYSTEM


| | |
|---|---|
| **Candidate：** | Shuang Li |
| **Student ID:** | 1130209287 |
| **Supervisor：** | Aassociate Prof. Chunxiang Wang |
| **Academic Degree Applied for：** | Master of Engineering |
| **Speciality：** | Mechanical Engineering |
| **Affiliation：** | School of Mechanical Engineering |
| **Date of Defence：** | Jan, 2016 |
| **Degree-Conferring-Institution：** | Shanghai Jiao Tong University |

# 上海交通大学

# 学位论文原创性声明

本人郑重声明：所呈交的学位论文《全自动交通系统车辆调度与人机交互系统研究》，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

日期：　　年　　月　　日

# 上海交通大学

# 学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权上海交通大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于

**保密**□，在___年解密后适用本授权书。

**不保密**□。

（请在以上方框内打“√”）

学位论文作者签名：　　指导教师签名：

日期：　　年　　月　　日　　日期：　　年　　月　　日

# 全自动交通系统车辆调度与人机交互系统研究

## 摘 要

全自动交通系统是由一系列无人车组成的城市或园区交通系统，该系统中的无人车会根据乘客的叫车请求快速做出响应，然后将乘客安全高效地送到目的地，该系统可以有效地解决汽车带来的安全、能源和环境等方面的问题，因此研究全自动交通系统具有重要意义，而其中的车辆调度算法又是保证其最优化的关键所在。目前，针对传统车辆调度问题的研究较为成熟，相对而言，全自动交通系统发展时间极短，对该系统中车辆调度问题的研究尚不成熟。目前已有的车辆调度算法应用于该系统中存在无人车响应效率不高，处理请求速度较慢等现象，本文针对上述问题，对车辆调度算法展开了较为深入的研究。

首先，本文通过分析全自动交通系统运行特点，提出了多个优化目标。对全自动交通系统中的站点、用户叫车请求以及无人车分别做了相应的数学实体建模，并且对优化目标进行高度抽象，得到一组优化目标函数。

其次，本文研究了用于车辆调度问题的传统算法，分析了遗传算法和贪婪算法的本质，遗传算法无法对乘客的叫车请求快速处理，贪婪算法无法对乘客的叫车请求迅速响应并且可能导致部分叫车请求饿死的问题，针对以上问题，本文研究了站点中聚类的特点，提出了一种基于站点关联度的聚类方法，进而提出了聚类贪婪遗传算法。本文设计并开发了一个全自动交通系统的仿真系统，并面向等待时间最优化和处理时间最优化分别做了仿真实验，实验证明聚类贪婪遗传算法用于全自动交通系统车辆调度可使系统在不同指标下达到最优化。

最后，本文在已有的研究成果基础上，设计并开发了一套用于全自动交通系统的车辆呼叫系统，该呼叫系统采用了 JavaEE 的技术架

构，使用了微信公众平台和百度地图 API，能够满足高负载高并发等技术要求，具有界面美观，使用简单等特点，目前该车辆呼叫系统已成功在东莞、上海等地的全自动交通系统中运行。

**关键词：**全自动交通系统，遗传算法，贪婪算法，聚类贪婪遗传算法，车辆呼叫系统

# RESEARCH FOR VEHICLES' SCHEDULING ALGORITHM AND MAN-MACHINE INTERACTIVE SYSTEM IN CYBERNETIC TRANSPORTATION SYSTEM

## ABSTRACT

Cybernetic Transportation System (CTS) is a city transportation system based on a fleet of driverless vehicles. The driverless vehicles can response passengers' calling requests quickly and then escort passengers to their destinations. CTS can effectively solve the problems of safety, energy and environment caused by traditional vehicles, so the research on CTS in which vehicles' scheduling algorithm plays a key role makes a big difference. Currently the research for classic Vehicle Scheduling Problem (VSP) matures while CTS only has a very short history, so the research on CTS needs further study. To solve the problem of traditional algorithm's inefficiency, this paper has carried out deep study on VSP in CTS.

Firstly, this paper proposed four optimization targets by deeply analyzing the characteristics of CTS, and then build mathematical models for driverless vehicles, passengers' calling requests and stations. By using the models we get the functions of optimization targets.

Secondly, the genetic algorithm has the disadvantage of long average handling time while the greedy algorithm has a slow response and may lead some requests to starve. To solve the problems above, this paper comes up with a clustering method based on stations' relevance and then comes up with Proximity Cluster Greedy and Genetic Algorithm (PCGGA). This paper also designs a simulation system to test all the three algorithms based on the waiting time's optimization and the handling time's optimization. The results show that PCGGA has a steady and fine performance in many optimization targets.

Finally, based on the results of research above, this paper designs and

develops a vehicle calling system for CTS. The calling system takes JavaEE as technology structure, and uses Wechat and Baidu Map API. The system has can bear both high loads and high concurrency, it also has a beautiful interface and friendly interaction. Currently, the calling system has run in many CTSs in Dongguan and Shanghai.

**KEY WORDS:** Cybernetic Transportation System, Genetic Algorithm, Greedy Algorithm, Proximity Cluster Greedy and Genetic algorithm，Vehicle Calling System

# 目 录

# 图 录

# 表 录

# 第一章 绪论

## 1.1 研究背景与意义

进入 21 世纪后，汽车工业逐渐成为我国国民经济的支柱产业，也是“中国制造 2025”必争产业。我国 21 世纪每年的汽车产量如图 1 - 1 所示，中国汽车产销量已连续六年蝉联世界第一，汽车相关产业的从业人员占全国城镇就业人数 12%以上，汽车工业总产值占我国 GDP 的 10%以上。汽车提供舒适便利，同时也带来安全、交通、能源、环境等全球性问题。2014 年我国发生了 16 万多起交通事故，其中有 4 万起造成了人员死亡，交通事故数量分别较 2012 年和 2013 年上升了 8.6%和 13.3%，中国的万车死亡率为 7.95，高居世界之首；2013 年交通拥堵造成国民经济损失占 GDP 的 1%；中国石油对外依存度超过 58.1%，每年新增石油消费量的 70%以上被新增汽车所消耗。2012 年机动车 NOx 排放量 640.0 万吨，占 NOx 排放总量的 27.4%，机动车 PM 排放量 62.1 万吨，占 PM 排放总量的 5.0%；CO 排放量 3467.1 万吨[1][2]。如何高效地解决交通拥堵，能源危机与环境污染等问题已迫在眉睫。

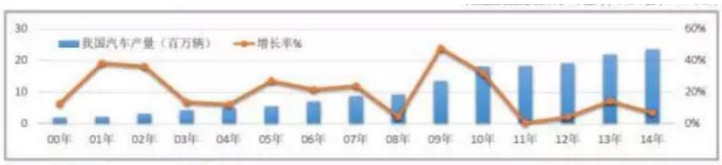



图 1 - 1 21 世纪我国每年汽车产量

Fig.1-1 The annual car production in China in the 21st century

信息化与智能化是解决能源、安全、拥堵、环境等全球性问题的重要手段，目前已成为世界汽车工业的技术前沿和发展方向。世界各国纷纷开始制定汽车智能化研究计划，日本正在着手 SmartWay 计划，欧洲正在着手 CVIS 计划，美国交通部正推进 IntelliDrive 项目，我国的《中国制造 2025》提出装备辅助驾驶系统的汽车减少交通事故 30%，减少交通死亡人数 10%，装备自动驾驶系统的汽车，综合能耗较常规汽车降低 10%，减少排放 20%，减少交通事故数 80%，基本消除交通死亡的汽车智能化技术目标[3][4]。

全自动交通系统是由一系列无人车组成的城市或园区交通系统。全自动交通系统如同在平面上展开的升降式电梯，在全自动交通系统中有很多供无人车停靠的站点，用户可以在这些站点上下车，用户通过叫车终端、手机短信或者微信给

控制中心发送叫车请求，控制中心接收到叫车请求后经过调度算法的运算，给对应的无人车发送接送用户的任务指令，然后无人车根据任务指令完成到对应站点装载用户并将用户送至相应目的地。

全自动交通系统是汽车信息化与智能化的一种高级表现形式，全自动交通系统可以有效地解决汽车带来的安全、交通、能源和环境等方面的问题，并且随着无人车的飞速发展，越来越多的全自动交通系统将投入到实际生产使用中来，甚至全自动交通系统在将来有可能取代公交系统。

因此如何使全自动交通系统的运转达到最优化是一个前沿且具有重要研究意义的课题，而车辆调度算法正是这一课题的关键所在。因此本文立足于此全自动交通系统中各项最优化目标，对车辆调度算法进行了广泛探索和深入研究，并且设计和开发了对与之匹配的车辆呼叫系统。

## 1.2 国内外研究现状

全自动交通系统是在无人车驾驶技术逐步成熟的基础上发展起来的，全自动交通系统的发展历史很短。在国外研究全自动交通系统的主要有欧盟的 Cybercars, Cybercars2, CyberMove 以及 CityMobil 项目[5]。2011 年欧盟 CityMobil 项目展示了全球第一个全自动交通系统[6]。图 2 展示的是英国希斯罗机场的全自动交通系统


图 1 - 2 英国希斯罗机场的全自动交通系统
Fig.1-2 Cybernetic Transportation System in Heathrow Airport, England

在国内研究全自动交通系统的有上海交通大学的 CyberC3 项目，2015 年万科与上海交通大学合作开发了被称为“大白”的无人车，并且在东莞、上海的园区中建立了全自动交通系统。现有的全自动交通系统具有无人车数量少，行车路线单一等特点，因此并未车辆调度系统也并未针对大规模数量无人车，复杂行车路线做

优化，而是通常采用常规的车辆调度算法。

### 1.2.1 国外研究现状

从 1959 年起车辆调度问题开始得到广泛研究，该问题由 Dantzig 和 Ramer[7] 首次提出。车辆调度问题是一个经典的 NP-Hard 问题[8]，在多项表达式的时间复杂度内无法求出该问题的准确解。

求解车辆调度问题的一大类方法是精确算法，该方法具有里程碑意义的是：1981 年 Christofieds 等采用动态规划方法放宽空间变量，通过对状态优化来求解车辆调度问题[9]；1991 年 Padberg 等利用分枝剪枝法对问题进行了求解[10]；1995 年 Fisher 采用了 K-度中心树松弛对车辆数目恒定的车辆调度问题进行了求解[11]；因为该问题本身是一个 NP-Hard 问题，采用精确的求解算法只针对较小规模的车辆调度问题有效，这种算法在面对大规模问题是具有相当大的局限性，因此近年来也鲜有学者在精确算法求解车辆调度问题领域取得突破性进展。

另一类比较重要的方法是启发式智能算法。

1981 年 Glover 提出禁忌搜索算法[12]，这是一种从局部领域到全局逐步搜索扩展的寻优算法。2012 年 Petersen H L 在使用禁忌搜索算法对车辆调度问题求解时，首先建立了一张禁忌表，然后对问题的全局逐步搜索扩展寻优[13]。

1953 年 N. Metropolis 等人最早提出了模拟退火算法[14]，该算法模仿钢铁退火的整个过程，通过全局搜索来求得最优化问题的最优解或者次优解。2013 年 Banos R 研究了使用模拟退火算法求解多目标带有时间窗的车辆调度问题[15]，同年 Mousavi 基于混合模拟退火算法对供应链中的车辆调度问题进行了研究[16]。

1975 年 J.Holland 根据生物进化的思想提出了遗传算法[17][21]，该算法是基于遗传变异与自然选择的全局性搜索算法，其具有并行性高、扩充性好、鲁棒性好、操作灵活等特点，因此可用其求解大规模的车辆调度问题，2003 年 Berger J 基于遗传算法提出一种用于解决车辆调度问题的混合遗传算法[18]，2005 年 Ombuki B 使用遗传算法对多目标带有时间窗的车辆调度问题进行了研究[19]。

1992 年 Dorigo M 首次在其博士论文中提出蚁群算法[20]，该算法是通过模拟蚂蚁寻找食物时发现路径的方式进行的，这是一种在图中寻找优化路径的概率型算法，它具有自组织、并行、正反馈以及鲁棒性等特点。蚁群算法能够针对车辆调度问题求得较优解，因此有部分学者对基于蚁群算法的车辆调度问题进行了研究，如 2004 年 JE Bell 基于蚁群算法对车辆调度问题进行了研究[21]。

直至目前，针对不同约束条件下的车辆调度问题国外许多学者进行了大量的

研究。

### 1.2.2 国内研究现状

国内对车辆调度问题的研究相较于国外起步比较晚，由于在多项表达式的时间复杂度内车辆调度问题无法求解，因此精确求解无法处理较大规模的车辆调度问题，因此对于车辆调度问题的研究主要是基于启发式搜索算法。

在禁忌搜索算法方面，2005 年钟石泉针对多车场有时间窗的多车型车辆调度及禁忌算法进行了研究，并根据车辆调度问题的具体约束情况设计了禁忌算法，进行了算例实验，最后得出禁忌算法针对有时间窗的多车型车辆调度问题具有有效性[22]。在模拟退火算法方面，郎茂祥针对装卸混合车辆路径问题利用模拟退火算法进行了研究，得出使用模拟退火算法求解混合装卸车辆路径问题具有效率高、收敛速度快、计算结果稳定等优点[23]。2010 年杨善林研究了时变条件下针对带时间窗的车辆调度问题的模拟退火算法，在研究时将车辆行驶速度考虑成时变分段函数，最后加以实验验证[24]。在遗传算法方面，卫田对使用遗传算法对物流配送中车辆路径问题进行了相应研究[25]。

启发式算法一般都具有易与其他启发式算法相结合的特点,因此很多学者采用启发式算法想结合的方式来改进算法。李高扬对蚁群算法在车辆调度问题中的应用进行了研究，他在求解过程把蚁群算法与遗传算法进行了结合和复用，从而有效回避了二者的缺点，保留了二者的优点，从而有效地对算法进行了改进[26][27][28]。柳武生将遗传算法和禁忌搜索算法相结合来对实时订货的车辆调度问题进行了求解，最终通过实例分析得出两种算法结合后不仅可以加快求解过程而且可以得到更可靠的解[29]；周兴田将遗传算法和模拟退火算法结合对车辆调度问题进行了研究，从而改进了遗传算法和模拟退火算法的缺点，有效地提高了算法的效率[30]；郎茂祥将遗传算法和爬山算法相结合，构造了求解物流配送路径优化的混合遗传算法，并使用该算法对车辆调度问题进行了求解，最终在一定程度上克服了遗传算法在局部搜索能力不足而爬山算法在全局搜索方面能力不足的缺点，从而得到质量较高的解[31]。2013 年邢鹏结合大数据互联网，对基于云平台对多配送中心车辆调度问题进行了研究，从而得出了一种新模式下的车辆调度算法，最后根据得出的算法设计了多配送中心车辆调度系统[32]。

随着车辆调度问题的需求日益变更，国内对于这方面的研究也在不停发展。截止目前尚无针对全自动交通系统比较成熟的车辆调度算法。

## 1.3 关键技术

全自动交通系统中的车辆调度问题不同于普通车辆调度问题，它是一种新型的交通方式，在全自动交通系统中有很多供无人车停靠的站点，用户可以在这些站点上下车，用户通过叫车终端、手机短信或者微信给控制中心发送叫车请求，控制中心接收到叫车请求后经过调度算法的运算，给对应的无人车发送接送用户的任务指令，然后无人车根据任务指令完成到对应站点装载用户并将用户送至相应目的地。在研究全自动交通系统中的车辆调度算法主要关注以下几方面。

### 1.3.1 数学模型建立

建立数学模型首先需要对问题背景有初步的估计和直观的认识，其次需要对问题中的一些情况作假设，接着需要分析问题中各元素之间的关系从假设推导数学模型，最后还需要对数学模型进行验证。

全自动交通系统中包含多种元素，主要有无人车，用户和站点，每个元素又由多个属性构成。只有对全自动交通系统中每个元素有准确的评估和认识，才能数学表达式来描述全自动交通系统中的每个元素，近而根据优化目标建立目标函数。而优化目标函数又是各种算法的基础，因此准确建立数学模型本文难点，同时也是研究全自动交通系统车辆调度算法的先决条件。

### 1.3.2 高频呼叫站点聚类

传统解决车辆调度问题的算法中都是从规划最短的车辆行驶路径的角度研究，本文同样采用了该思想，使用遗传算法在全自动交通系统中所有站点中规划了一条较短的无人车行驶路径。但本文最重要的算法是基于遗传算法和贪婪算法改进后的聚类贪婪遗传算法，该算法的一个关键技术点是根据叫车请求的历史纪录，从所有的站点中划分出高频呼叫站点和低频站点。而另一个更重要的关键技术是找出已有的高频呼叫站点中的聚类，寻找离散点的聚类中心可以使用 K-Means 算法[33]，然而全自动交通系统中的站点不是独立的，各个站点之间有通过叫车请求有不同程度的联系，因此在全自动交通系统下寻找高频呼叫站点的聚类是一个重要且困难的技术点，因此准确有效地寻找高频呼叫站点的聚类对于本文的核心算法——聚类贪婪遗传算法具有重要意义。

### 1.3.3 车辆呼叫系统中的高负载和高并发

本文将在全自动交通系统车辆调度算法研究成果上设计并开发一个车辆呼叫系统，该系统将在上海、东莞等地的全自动交通系统中运行，因此整个系统的设计与开发均应从企业级需求出发。企业级系统要求系统具有易用性，可靠性，易维护性，易拓展性等特点，更重要的是要求系统必须能够同时处理大量并发访问，同时处理较高的数据负载。因此在开发全自动交通系统中的车辆呼叫系统应着重关注其高负载和高并发的能力，这两项技术是整个车辆呼叫系统的关键所在。

## 1.4 本文主要研究内容

本文主要研究全自动交通系统中的车辆调度算法，针对全自动交通系统的具体情况，建立了全自动交通系统中各元素的数学实体模型，从全自动交通系统中乘客和运营者角度出发提炼了四大优化目标，并根据数学实体模型得出优化目标函数。之后本文基于乘客平均等待时间最优化分别对遗传算法、贪婪算法和新提出的聚类贪婪遗传算法做了研究，并通过仿真实验验证了聚类贪婪遗传算法的有效性。接着本文转换优化目标，基于叫车请求平均处理时间最优化分别再对遗传算法、贪婪算法和聚类贪婪遗传算法做相应研究，同样通过仿真实验验证了聚类贪婪遗传算法的高效性。然后本文将前两中不同优化目标下的聚类贪婪遗传算法结果进行了对比，并得出结论。最后在前文车辆调度算法研究成果的基础上，结合了全自动交通系统的实际情况，进行了深入的需求分析，基于 JavaEE[34]开发了一套面向企业级的车辆呼叫系统。文章的各章节内容安排如下：

第一章介绍了本文的研究背景及意义，详细阐述了国内外全自动交通系统的发展和现状，分析了车辆调度算法在其中的重要性，结合全自动交通系统的实际情况，分析得出其中的关键技术分别是数学模型建立、高频呼叫站点聚类、车辆呼叫系统中的高负载和高并发。

第二章用具体的数学问题描述定义了全自动交通系统中的“最优化”，并且提出了相应的最优化目标；对全自动交通系统中的无人车站点、用户叫车请求以及无人车分别做了相应的数学实体建模，得出了一组方便用于数学描述和算法描述的符号；然后本章利用上文中的数学实体模型，对优化目标重新用数学公式进行高度抽象以方便下文的算法研究。

第三章从解决传统车辆调度问题的思路出发，对遗传算法的产生和发展，基本思路以及其特点做了相应的概述；接着针对本文实际情况研究了面向最优等待

时间的遗传算法在全自动交通系统中的具体方法和步骤。然后本章采用面向最优等待时间的贪婪算法来求解全自动交通系统中的车辆调度问题，给出了相应的流程图和伪代码。本章针对遗传算法和贪婪算法的缺点，并保留二者优点，提出聚类贪婪遗传算法，在求解高频呼叫站点聚类群时提出基于高频呼叫站点关联度的聚类算法，进而得出聚类贪婪遗传算法的流程图。最后本章根据全自动交通系统的实际情况，分析了具体需求，设计并开发了一个全自动交通系统的仿真系统。在该仿真系统中验证了聚类贪婪遗传算法的高效性和稳定性。

第四章在第三章的基础上以缩短处理时间为优化目标，调整了遗传算法的适应度函数，重新选择了贪婪算法的贪婪条件，调整了聚类贪婪遗传算法的无人车分配策略，并且进行了仿真实验与分析，验证了聚类贪婪遗传算法的高效性和稳定性。最后本章针对不同优化目标下聚类贪婪遗传算法的实验结果做了对比分析。

第五章通过分析系统中的多个角色及其在全自动交通系统中的行为得出了详细的功能需求，然后结合需求分析和技术需要得出车辆呼叫系统的技术架构图。接着本章对框架中的消息模块的架构，存储和代理等部分做了重点研究，从而得到了一种分布式，易于向外扩展，能够同时为订阅和发布消息提供高吞吐量，支持多个订阅者，当任务失败时能够自动平衡消费者，能够将消息持久化到磁盘上，可以用于批量消费的消息模块。其次本文对系统中 Web 模块进行了相应研究，使用 MVC 设计可以使 Web 模块具有高解耦性和高鲁棒性的特点；采用 RBAC[35]的权限设计模式，可以将权限控制精细到每个方法和每个资源的粒度，采用微信公众平台和百度地图 API 可以更方便的和用户进行交互，增强用户体验。最后，本章根据前文中车辆调度算法的研究以及对于车辆呼叫系统的研究，设计和开发了一套用于全自动交通系统的车辆呼叫系统，并且目前已投入生产实践中了。

第六章对全文进行了全面的总结和展望，综合性地分析了本文的主要研究过程，总结了本文研究问题的核心难点，简要介绍了文本的创新点和不足之处。进而针对文中所提方法进一步分析了其改进的方向，同时对全自动交通系统中的车辆调度算法的研究进行了展望。

# 第二章 数学模型建立

## 2.1 引言

全自动交通系统是一种基于一定数量的电动无人驾驶车的城市交通系统，这些无人驾驶车通常采用了传感器技术、定位技术、避障技术、识别技术、控制技术等，在中央控制系统的调度下能够在园区环境中自动将用户送到目的地。在本文中，全自动交通系统有一定数量的无人车以及一些无人车停靠站点，用户可以通过微信或者叫车终端发送叫车请求，中央控制系统接收到用户的叫车请求后，根据调度算法安排无人车前往指定的站点接送用户到达目的站点。如何调度无人驾驶车从而使得用户的叫车请求响应达到最优化是本文的研究重点。最优化问题一般具有多维度的目标，本文拟从以下角度进行研究：

1) 如何使叫车请求平均等待时间尽量短。
2) 如何使叫车请求平均处理时间尽量短。
3) 如何使叫车请求最大等待时间尽量短。
4) 如何使所有车辆在一定时间内处理叫车请求数量尽量多。

用户通过微信或者叫车终端向中央控制系统发送叫车请求后，控制终端会立即安排无人车前往用户设置的起始站点接用户，在这个过程中用户发送叫车指令、中央控制系统根据叫车指令调度无人车的时间一般很短，可以忽略不计。但是无人车到用户起始站点的时间则和无人车接受到调度任务时的位置有关，若此时无人车距离叫车指令中的起始站点太远，则用户必然要等待较长时间才会有无人车前来接送。因此叫车请求等待时间是影响用户体验的一个非常重要因素，本文采用全自动交通系统中这一指标的平均值来作为本文的优化目标之一。

同时，中央控制系统在调度车辆的时候可能出现叫车请求平均等待时间较短，但是个别叫车请求的等待时间特别长，甚至可能出现个别叫车请求没有无人车前来响应，对于这种情况若只单纯用平均等待时间则无法完整地描述最优化，因此本文将叫车请求最大等待时间也作为优化目标之一。

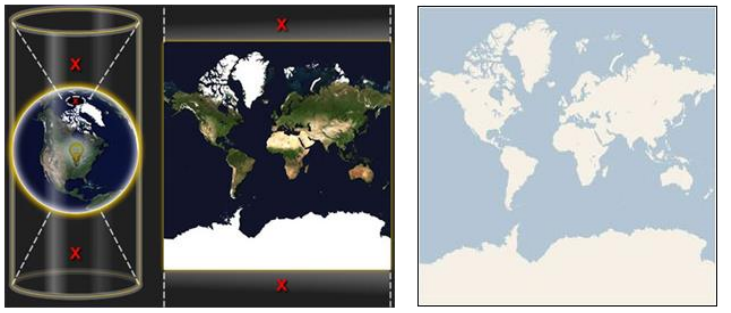
用户乘坐无人车从起始站点到目的站点的时间和站点之间的距离有关，同时也和无人车的行车速度和行驶路径有关，而无人车的行驶路径则主要由调度算法决定的，该指标是决定最优化的一个关键性指标，因此本文将叫车请求平均等待时间作为另一优化目标。

从用户体验的最优化，前三个优化目标已经完全覆盖了，但是如果从全自动

交通系统运行的整体效率看，还应关心单位时间内处理的叫车请求数量，因此本文将所有无人车在一定时间内处理叫车请求数量作为优化目标之一。

## 2.2 墨卡托投影坐标系

本文中无人车的定位，距离计算，路径规划均依赖于车辆地理位置，墨卡托投影是目前使用最广泛的地图坐标系。墨卡托投影基本是由荷兰地图学家墨卡托在 1569 年 (Mercator)提出[36]。墨卡托投影是基于这样一种假设，把地球放在在一个刚好可以与地球赤道相切的空心圆柱体中，地球中心有一个光源，从光源出发地球上的每一点都被投影到圆柱体柱面上，把圆柱体展开即得到墨卡托投影，原理如如图 2 - 1 a)所示。图 2 - 1 b)展示的是一幅标准纬线为零度(即赤道)的“墨卡托投影”绘制出的世界地图。



a) 墨卡托投影原理　　b) 墨卡托投影下的世界地图

图 2 - 1 墨卡托投影
Fig.2-1 Mercator Projection

墨卡托投影只根据赤道长度保留中间正方形的区域，赤道半径 r=6378137 米，赤道周长为 c=2*π*r=20037508.3427892 米，因此墨卡托投影的 X 轴取值范围是[-20037508.3427892,20037508.3427892]，Y 值的取值范围和 X 相同，如此便将地球投影成了一张正方形的地图。

在本文中无人车向中央控制系统汇报位置时采用的是经纬度的方式，中央控制系统会将该经纬度坐标转换成墨卡托投影坐标用于计算无人车在墨卡托投影坐标系中的位置，同时可以将无人车在地图中实时地显示出来。

## 2.3 实体模型建立

全自动交通系统由以下元素组成：

(1) 一组坐标已知的无人车站点 $S =< s_1, s_2, \dots, s_n >$

(2) 一组由用户发出的叫车请求 R $=< r_1, r_2, \ldots, r_n >$

(3) 一组无人车 M $=< m_1, m_2, \ldots, m_n >$

因为叫车请求的属性能够完全表达用户的行为，因此下文以叫车请求作为研究就像，不再考虑用户实体。本文中，每辆无人车都有一份详尽的叫车请求清单(在全自动交通系统中通过中央控制系统和无人车的通信来保证的)，这份清单中记录了所有的叫车请求，包括叫车请求的所有者(用户)，发起时间，起始站点，目的站点等信息。全自动交通系统中各个元素的数学模型抽象如下：

**站点**$s_i$采用元组$\{x_i, y_i, d_i[\mathrm{N}]\}$描述,其中

(1) $x_i$是站点$s_i$的墨卡托投影横坐标。

(2) $y_i$是站点$s_i$的墨卡托投影纵坐标。

(3) $N$是全自动交通系统中站点总数。

(4) $d_i[\mathrm{N}]$是一个数组，它代表了站点$s_i$和其它站点的距离。

墨卡托投影横纵坐标可以和经纬度进行转化，所以站点的位置同样可以用经纬度来描述。每个站点均有一个长度为N的数组来表示和其它站点的距离，和自己本身的距离是 0，和不联通的站点之间的距离是无穷大，我们可以用一个非常大的数来表示无穷大，例如在程序中用一个最大的 int 来表示。通过这样 N 个数组就可以用矩阵法描述全自动交通系统中站点形成的图，在本文中以无向图表示，无向图指两个顶点之间的路径不带有方向。如果在全自动交通系统实际的道路中只允许无人车在某些道路上单方向运行，那么就应该使用有向图，在二维矩阵中用类似的方法描述。

**叫车请求**$r_i$采用元组$\{f_i, t_i, I_i, J_i, m_i\}$描述,其中

(1) $f_i$是叫车请求$r_i$的起始站点。

(2) $t_i$,是叫车请求$r_i$的目的站点。

(3) $I_i$是叫车请求等待时间。

(4) $J_i$是叫车请求处理时间，即从用户上车到用户抵达目的站点下车的这段时间。

(5) $m_i$是处理该请求的车辆。

在全自动交通系统中无人车只能在站点停靠，因此用户并不是可以在任何地方上下车，$f_i$是指叫车请求的起始站点，即乘车希望的上车站点，一般情况下，用户希望在开始叫车时会选择一个离自己最近的站点作为自己的起始站点。$I_i$是从用户发出叫车请求$r_i$到用户上车的这段时间。

**无人驾驶车**$m_i$采用元组$\{L_i, R_i[L_i], x_i, y_i, v_i, s_i\}$描述，其中

(1) $L_i$是车辆$m_i$的准载数。

(2) $R_i[L_i]$是一个长度为$L_i$的数组，数组中存放的是车辆$m_i$当前正在处理的请求，换言之，$R_i[L_i]$表示了车辆$m_i$上所有用户。

(3) $x_i$是车辆$m_i$的墨卡托投影横坐标。

(4) $y_i$是车辆$m_i$的墨卡托投影纵坐标。

(5) $v_i$是车辆$m_i$的行驶速度。

(6) $s_i$是车辆$m_i$即将前往的下一个站点。

## 2.4 目标函数建立

在全自动交通系统中首先要考虑的是用户的乘车体验，因此从用户的角度考虑有：

**优化目标 1**：叫车请求平均等待时间尽量短。

叫车请求的等待时间是一个典型的巡逻问题[38]优化目标,因此本文使用标准巡逻评估方程[39][40]作为目标函数，即 t 时刻的平均等待时间$\bar{I}_t$为:

$$\bar{I}_t = \frac{1}{|R_t|}\sum_{i=1}^{|R_t|} I_{i(t)} \tag{2-1}$$

其中，$|R_t|$表示在 t 时刻所有的叫车请求总数，$I_{i(t)}$表示 t 时刻叫车请求$r_i$的等待时间。

**优化目标 2**：叫车请求平均处理时间尽量短。

平均处理时间同样采用标准评估方程：

$$\bar{J}_t = \frac{1}{|R_t|}\sum_{i=1}^{|R_t|} J_{i(t)} \tag{2-2}$$

其中，$|R_t|$表示在 t 时刻所有的叫车请求总数，$J_{i(t)}$表示 t 时刻叫车请求$r_i$的处理时间。

**优化目标 3**：叫车请求最大等待时间尽量短。

在设计车辆调度算法以及规划车辆行驶路径的时候，有可能会忽视了某些站点，从而使得有的叫车请求一直没有车辆响应，或者车辆响应了某个叫车请求，但是一直不行驶到目的站点，从而造成请求饿死状态。请求的最大等待时间是尚未处理的请求和正在处理中的请求以及处理完毕的请求中的最大等待时间：

$$I_{\max(t)} = \max(\forall r_i \in R, I_{i(t)}) \tag{2-3}$$

其中，R 表示所有叫车请求的集合，$I_{i(t)}$表示 t 时刻叫车请求$r_i$的等待时间。

在以用户乘车体验作为优化目标的同时，也考虑全自动交通系统中车辆运营的最优化，因此从车辆运营最优化考虑有：

**优化目标 4**：单位时间内处理叫车请求尽量多。

本文用$C_t$表示时刻 t 总共处理的叫车请求：

$$C_{\mathrm{t}} = |R_f(t)| \tag{2-4}$$

其中，$R_f(t)$为 t 时刻已经处理完毕的叫车请求集合。

本文接下来将结合传统求解车辆调度问题的算法以及巡逻问题的算法使得目标函数(2-1), (2-2), (2-3), (2-4)达到最优化。

## 2.5 本章小结

本章用具体的数学问题描述定义了全自动交通系统中的“最优化”，并且提出了相应的最优化目标，对实体模型中采用的墨卡托投影坐标系做了相应的介绍；接着本章对全自动交通系统中的无人车站点、用户叫车请求以及无人车分别做了相应的数学实体建模，得出了一组方便用于数学描述和算法描述的符号；然后本章利用上文中的数学实体模型，对优化目标重新用数学公式进行高度抽象以方便下文的算法。

# 第三章 面向最优等待时间的算法研究

## 3.1 引言

全自动交通系统中的车辆调度算法直接影响到用户叫车的平均等待时间，最大等待时间，平均乘车时间以及无人车有效利用率，从而直接决定了全自动交通系统整体的效率，因此车辆调度算法在整个全自动交通系统中具有十分重要的作用。

因此在上文中提到的四个优化目标中，有可能存在不同优化目标之间产生互斥，即可能存在当系统中叫车请求平均等待时间较短时，系统中的叫车请求平均处理时间变长的情况。因此本章拟基于用户叫车请求等待时间最优化进行相应的算法研究。

遗传算法是一种求解经典车辆调度问题的重要算法，它具有自组织，自适应以及自学习等优点，因此本章首先采用遗传算法来对全自动交通系统中的车辆调度问题进行数学建模、算法流程设计。贪婪算法是一种用于求解多智能体巡逻问题的重要算法，它具有快捷性、直观性等特点，因此本章接着采用贪婪算法对全自动交通系统中的车辆调度问题进行贪婪条件研究、算法流程设计。然后本章根据全自动交通系统的实际情况，结合遗传算法和贪婪算法的特点，尝试保留二者优点，规避二者缺点，提出一种专门用于解决全自动交通系统车辆调度问题的算法——聚类贪婪遗传算法。最后本章使用上述三种算法进行仿真实验，并对实验结果进行分析。

## 3.2 遗传算法

1967 年 Bagley J.D 最早提出了遗传算法(Genetic Algorithm, GA)的概念[41]，1975 年密西根大学的 J.H. Holland 教授开始了遗传算法理论和方法的系统性研究[42]。至此以后遗传算法在图像处理、模式识别、工业优化控制、神经网络、交通物流等多个科学领域中得到了广泛的使用和快速的发展。

遗传算法的基本思想是基于 Darwin 的进化论[43]和 Mendel 的遗传学说[44]的。优胜劣汰，适者生存，不适者淘汰是 Darwin 进化论中最重要的思想。遗传算法是一种模拟自然界中生物进化过程的一种搜索算法。它最初随机产生一组初始解，这组初始解称为种群。种群中每个个体是搜索问题的一个解，这些个体称为染色体，在遗传算法中染色体通常用一串数据或数组来表示，这个表示的过程叫做编

码。不同染色体之间进行交叉，染色体自身也会发生变异，从而产生后代，该过程称为遗传。遗传算法中采用适应度来评价一个染色体的好坏，在筛选个体时也是根据适应度的高低随机进行筛选的。通常经过若干次迭代之后，算法会收敛，从而得到问题的较优解。

全自动交通系统中的车辆调度问题本质上是调度问题，因此本文可以借鉴求解经典车辆调度问题的思路，使用遗传算法对问题求解。全自动交通系统中的车辆调度问题和经典车辆调度问题不同之处在于，经典车辆调度问题中车辆只需要前往特定目的站点一次，因此针对目的站点使用遗传算法规划处一条较短路径即可作为车辆的行驶路径。而全自动交通系统中不仅有目的站点，而且还有起始站点，同一个起始站点对应着不同的目的站点，仅仅采用一条简单的路径无法满足全自动交通系统中车辆调度问题的实际需要。因此本文基于用户叫车请求的等待时间最优化采用遗传算法为全自动交通系统生成无人车行驶策略。

遗传算法通过模拟生物进化的自然选择和遗传法则来实现的，面向最优等待时间的遗传算法主要包括以下要点：

**编码：**遗传算法不能够直接处理问题空间的数据，因此我们必须首先将问题空间的数据映射成遗传空间的基因型结构数据，这个过程叫做编码，一组编码便对应了一组解。本文将采用值编码，如站点 A, 站点 B，站点 C，站点 D 就编码为 1-2-3-4。

**适应度：**物种的适应度体现为对环境的适应能力，该能力也是生物进化的驱动力。遗传算法中适应度函数是用来评价解好坏的标准，解的质量和适应度函数值成正相关关系。在使用遗传算法求解经典车辆调度问题时，通常采用规划路径长度的倒数作为适应度函数，即：

$$\mathrm{F} = \frac{1}{\sum_{i=1}^{n} |S_i S_{i+1}|} \tag{3-1}$$

其中，$S_{n+1} = S_1$，$|S_i S_{i+1}|$表示站点$S_i$和站点$S_{i+1}$之间的距离。

当用户请求的等待时间作为最优化目标时，本文选取规划路径下的等待时间总和的倒数作为适应度函数，即：

$$\mathrm{F} = \frac{1}{\sum_{i=1}^{n} \left( \frac{\sum_{j=1}^{i} |S_j S_{j+1}|}{v_k} |r|_{f=s_{i+1}} \right)} \tag{3-2}$$

其中，$S_{n+1} = S_1$，$|S_i S_{i+1}|$表示站点$S_i$和站点$S_{i+1}$之间的距离，$v_k$表示无人车$m_k$的行驶速度，$|r|_{f=s_{i+1}}$表示以$s_{i+1}$为起始站点的所有未处理的叫车请求总数。

**选择：**遗传算法模拟自然选择过程，让适应度函数值大的个体具有较高的概率保留下来，而让适应度函数值较低的个体具有较高的概率被淘汰。常见的选择

方法有轮盘赌选择法，锦标赛选择法，随机遍历选择法，排序选择法。本文采用轮盘赌方法选择算子，操作方法是想象有一个转动的轮盘，轮盘上的扇区大小是根据个体的适应度按比例分配的，适应度越大则扇区面积越大。每次转动轮盘的时候轮盘中心指针指向某一个固定的地方，当轮盘停下来时，指针所指的扇区对应的个体便是选择出来的个体。

**交叉：**生物进化是通过基因重组的方式让父代的基因遗传给子代，遗传算法的则是采用交叉算子的方式从父代中产生子代，交叉算子操作就是让两个或两个以上的父代个体的部分结构加以替换重组的操作。本文采用两交换启发交叉方式。以 8 个站点为例，将其顺序编码为 1-2-3-4-5-6-7-8.

随机选出两条路径α, β：

$$\alpha = 3\ 7\ 2\ 8\ 6\ 4\ 1\ 5$$
$$\beta = 4\ 8\ 3\ 2\ 1\ 7\ 5\ 6$$

随机选出初始站点 j=1(位置号)，$S_j = 1$(站点号),向右调整使得 1 成为两双亲的第一位置站点如下:

$$\alpha1 = 1\ 5\ 3\ 7\ 2\ 8\ 6\ 4$$
$$\beta1 = 1\ 7\ 5\ 8\ 6\ 4\ 3\ 2$$
$$\gamma1 = 1\ *\ *\ *\ *\ *\ *\ *$$

判断因为d(1,5)＞d(1,7)，则 A1 以 7 为中心向右调整得如下中间结果:

$$\alpha2 = *\ 7\ 2\ 8\ 6\ 4\ 5\ 3$$
$$\beta2 = *\ 7\ 5\ 8\ 6\ 4\ 3\ 2$$
$$\gamma2 = 1\ 7\ *\ *\ *\ *\ *\ *$$

接着从第 2 个位置开始继续重复以上操作过程

判断因为$d(7,5) = d(7,2)$则可任选 5 或 2， β2以 5 为中心向右调整得如下中间结果:

$$\alpha3 = *\ 7\ 5\ 3\ 2\ 8\ 6\ 4$$
$$\beta3 = *\ 7\ 5\ 8\ 6\ 4\ 3\ 2$$
$$\gamma3 = 1\ 7\ 5\ *\ *\ *\ *\ *$$

如此反复得出

$$\gamma = 1\ 7\ 5\ 3\ 2\ 8\ 6\ 4$$

**变异：**变异是指在产生子代时，其基因上的部分信息会有一定概率发生突变，从而产生一些不同于父代的特征。遗传算法中的变异操作模拟了生物进化中的基因突变，反应到遗传算法中就是求得的子代有一定概率跳出局部最优解，从而可能搜索到全局最优解。变异概率$P_m$决定了遗传算法全局搜索的能力，太大则算法

不容易收敛，太小则又容易收敛于局部最优解，从而寻找不到全局最优解。通常将该概率设置为$P_m = 0.5$。本文采用逆转变异算子进行变异操作：

在个体中按照变异概率随机挑选两个站点，然后再交换两个站点，操作如下：

变异前：

$$\alpha = 1\ 3\ 8\ 7\ 6\ 4\ 2\ 5$$

变异后：

$$\alpha' = 1\ 2\ 8\ 7\ 6\ 4\ 3\ 5$$

遗传算法的流程图如图 3 - 1 所示：

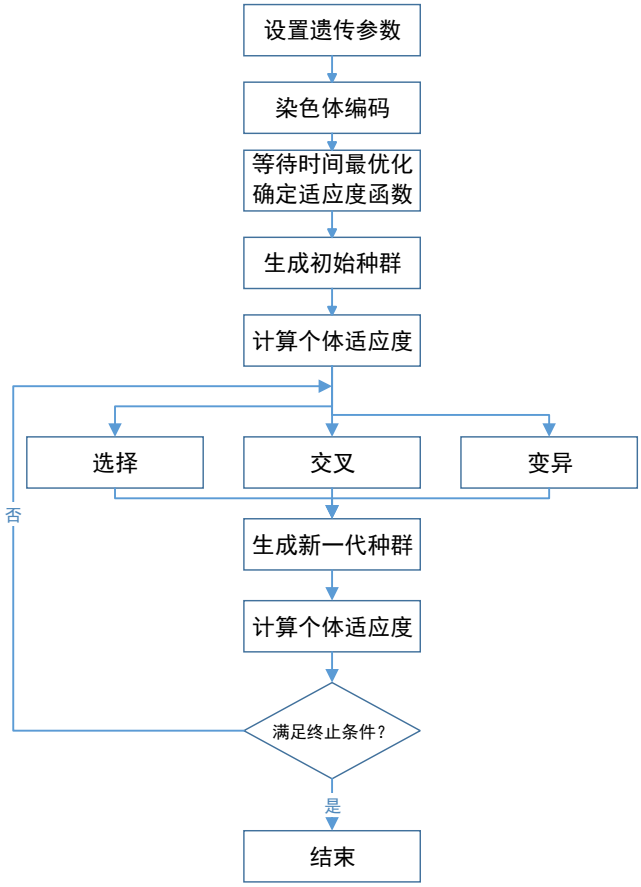



图 3 - 1 遗传算法流程图

Fig.3-1 Flow chart of genetic algorithm

**Algorithm 1. Genetic Algorithm**

**t** ← 0
**initialize Pop(t)with N chromosomes** $Pop_i(t)$
**while (not terminate condition)** ***do***
  **for i** ← **to N do**
    $f_i \leftarrow f(Pop_i(t), I_i)$
  **for i** ← **to N do**
    $\mathbf{NewPop}_i(t+1) \leftarrow$ **roulette choose**

$\boldsymbol{Pop_i(t)} \in \mathbf{Pop(t)}$ **with** $\boldsymbol{p_j} = \frac{f(j)}{\sum_k f(k)}$
**CrossPop(t + 1)** ← **recombine**
( **NewPop(t + 1)**) **with** $\boldsymbol{P_c}$
**MutPop(t + 1)** ← **mutate(CrossPop(t + 1)**
**with** $\boldsymbol{P_m}$
**Pop(t + 1)** ← **MutPop(t + 1)**
**t** ← **t + 1**
**while(not terminate condition) do**
**car moves by the path of Pop(T).**

遗传算法的伪代码如 Algorithm 1，其中

(1) 初始化 t 代表设置进化的代数。

(2) $f_i$代表第 i 个个体的适应度。

(3) $p_j$ 代表了个体 j 根据其适应度被选择遗传的概率。

(4) $P_c$ 为交叉概率。

(5) $P_m$ 为变异概率。

在全自动交通系统中，用户叫车请求的分布是随时间变化的，上文中是使用某一时刻的叫车请求分布来计算适应度函数。面向最优等待时间的遗传算法是一种动态的方法，系统会在某辆无人车出发前为其生成一条独有的路径，该路径是采用遗传算法生成，其中的适应度函数是当前所有未处理的叫车请求的总等待时间的倒数，每当无人车按照该路径运行一周时，系统便又按照上述方法给无人车生成一条新的路径。

## 3.3 贪婪算法

贪婪算法是一种不从全局考虑，而只关注当前局部最优解的算法，因此即使贪婪算法得不到问题的全局最优解，它也可以得到问题的局部最优解，从而得到的解也是近似的全局最优解[45]。因为可以通过对一系列贪婪算法获得的局部最优做选择，从而获得整体最优解，所以对于范围十分宽广的最优化问题来讲，贪婪算法是一种最直接最实用的算法设计思想[46]。通常说来，求解最优化问题的全局最优解时，都是从贪婪选择开始的，即在做每一次贪婪选择之后，原有的较大的优化问题就能够简化为一个规模较小的最优化子问题，然后通过多次贪婪选择后，最终求得一个整体最优解。

全自动交通系统中众多的无人车接受中央控制系统的调度，控制系统根据用户的叫车请求为无人车生成特定的行车路线，从另一个角度看可以将无人车当作

是在全自动交通系统中为用户一对一服务，当用户有乘车需求时，无人车便响应用户的需求，因此使用贪婪算法求解此问题便成为了一种较为自然的思路。

本章首要考虑用户的叫车请求等待时间最优化，因此使用贪婪算法时首要以降低用户叫车请求的等待时间为目的，因此主要有以下两个贪婪原则：

1) 对于空闲无人车（即当前没有乘客且尚无下一步任务的无人车），中央控制系统总是让其优先处理等待时间最长的叫车请求。

2) 对于非空闲无人车（即载有乘客且乘客尚未到达目的站点的无人车），中央控制系统总是让其优先处理起始站点离自己最近的叫车请求。

根据以上贪婪原则，得到面向最优等待时间的贪婪算法流程图如图 3 - 2 所示：

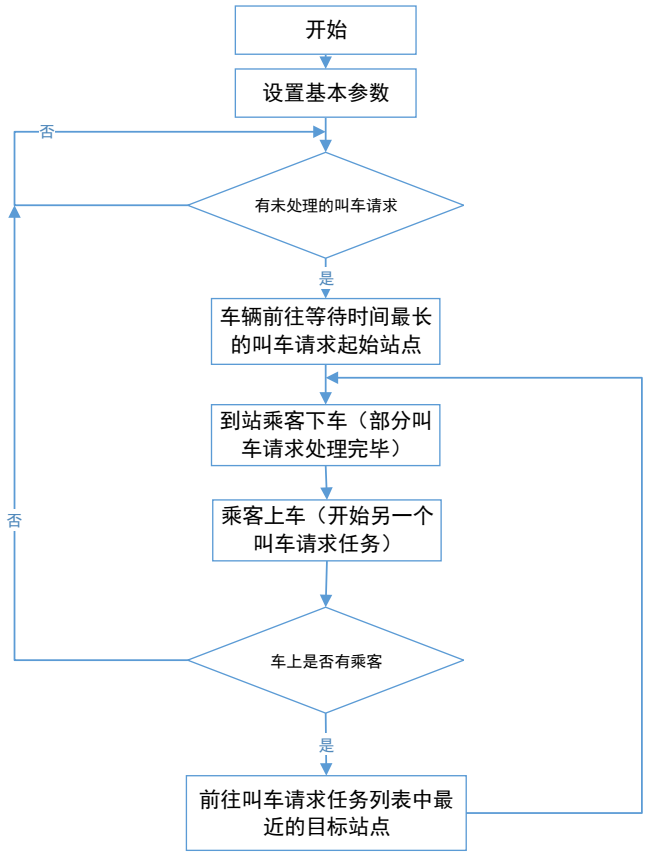



图 3 - 2 面向最优等待时间的贪婪算法流程图

Fig.3-2 Flow chart of greedy algorithm based on the optimization of waiting time

该算法的伪代码如 Algorithm 2 所示，Algorithm 2 采用 2.3 中建立的数学模型描述。

**Algorithm 2. Greedy Algorithm**

**t** ← 0
**initialize car m**
**while** (**not terminate condition**) ***do***

**if** (**exists any waiting requests**) ***then***
  $\boldsymbol{find\ r_m\ where\ r_m.I_m = \max(r_i.I_i, r_i \in R_t)}$
  Add $\boldsymbol{r_m}$ in $\mathbf{m}.\boldsymbol{R}[\boldsymbol{L_i}]$
  while ($\mathbf{m}.\boldsymbol{R}[\boldsymbol{L_i}]$ not empty) then
    **Add all** $\mathbf{r}_j$ **with** $\mathbf{r}_j.\mathbf{f}_j = \boldsymbol{r_m.f_m}$ **in** $\mathbf{m}.\boldsymbol{R}[\boldsymbol{L_i}]$
    $\mathbf{m.s} \leftarrow \boldsymbol{r_m.I_m}$
    **Remove** $\mathbf{r}_k$ **with** $\mathbf{r}_k.\mathbf{f}_k = \boldsymbol{r_m.f_m}$ **in** $\mathbf{m}.\boldsymbol{R}[\boldsymbol{L_i}]$
    $\boldsymbol{find\ r_m,\ where\ |m.s - r_m.f_m| = \min(|m.s - r_i.I_i|, r_i \in \mathbf{m}.R[L_i])}$
    $\mathbf{m.s} \leftarrow \boldsymbol{r_m.I_m}$
$\mathbf{t} \leftarrow \mathbf{t+1}$

## 3.4 聚类贪婪遗传算法

### 3.4.1 聚类贪婪遗传算法简介

上文中讨论了面向最优等待时间的遗传算法和贪婪算法用于求解本文的车辆调度问题，虽然遗传算法具有算法简单，生成的车辆路径能够覆盖每个站点同时保证单一路径是总多路径中的较优解的特点，但是该方法在对待每个站点均同等对待，因此容易出现个别站点叫车请求聚集而另外的某些站点叫车请求稀少但无人车经常路过的现象，即容易出现叫车请求平均处理时间较长的情况。采用贪婪算法求解本文车辆调度问题虽然处理请求的平均时间较短，但是极有可能出现部分叫车请求没有无人车响应，即叫车请求饿死的状况。

针对以上两种算法的优点和缺点，本文提出一种解决全自动交通系统车辆调度问题的算法——聚类贪婪遗传算法。

通过叫车请求记录的分析知道，各个站点的请求数量不是一样的，换言之，有的站点是请求聚集的，本文称这种站点为高频呼叫站点，有的站点是请求稀疏的，本文称这种站点为低频站点。本文通过站点$s_i$在 0~t 时间段出现在$r_i$中（$s_i = f_i$ 或 $s_i = t_i$）的频率$P_{s_i}(t)$来确定$s_i$是否为高频呼叫站点：

$$P_{s_i}(t) = \frac{\sum_{i=0}^{N(t)} g(s_i,f_i)+\sum_{i=0}^{N(t)} g(s_i,t_i)}{2*\sum_{i=0}^{N(t)}|r_i(t)|} \tag{3-3}$$

其中：

$$\mathrm{g(x,y)} = \begin{cases} 1, \text{当}x\text{和}y\text{为同一站点} \\ 0, \text{当}x\text{和}y\text{为不同站点} \end{cases} \tag{3-4}$$

如果$P_{s_i}(t) > P_{hot}$则$s_i$在时刻 t 为高频呼叫站点，$P_{hot}$是设定的高频呼叫站点标准频率。

聚类贪婪遗传算法结合了贪婪算法和遗传算法，该算法让部分无人车在高频呼叫站点聚类中按照贪婪算法决定的路径行驶，让另外的无人车在全部站点中按

照遗传算法决定的路径行驶。求解离散点的聚类时通常有 K-means[33]等聚类方法。

### 3.4.2 K-Means 聚类算法

在聚类贪婪遗传算法中首先就是需要确定高频呼叫站点的聚类，如图 6 所示，在图的左边分布着一些离散点，用肉眼可以看出来左图有四个高频呼叫站点的聚类，但是如何通过计算机程序找出这几个聚类来呢？于是就出现了 K-Means 算法。

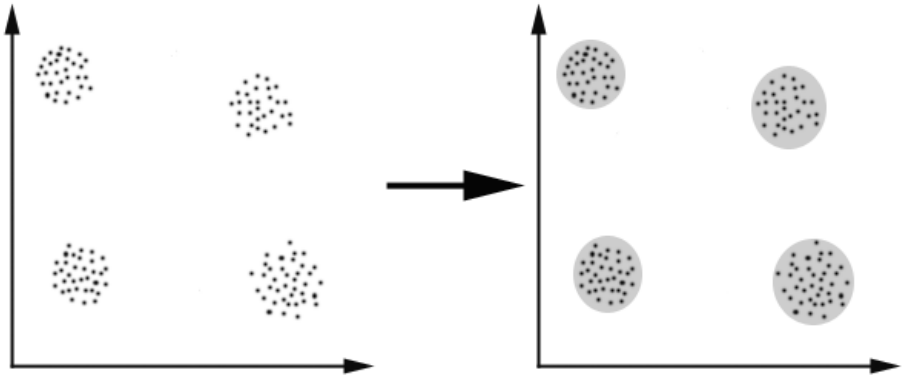


图 3 - 3 K-means 要解决的问题
Fig.3-3 Problems to be solved by K-means.

在使用 K-Means 算法时一个重要点就是计算聚类的中心点，通常聚类中心点的算法可以简单地使用个点的 X,Y 坐标的平均值，这里给出另外三种求解中心点的方法。

(1) Minkowski Distance 公式

$$d_{ij} = \sqrt[\lambda]{\sum_{k=1}^{n}\left|x_{ik} - x_{jk}\right|^{\lambda}} \tag{3-5}$$

其中 λ 可以随意取值，可以是负数，也可以是正数，或是无穷大。

(2) Euclidean Distance 公式

$$d_{ij} = \sqrt{\sum_{k=1}^{n}\left|x_{ik} - x_{jk}\right|^{2}} \tag{3-6}$$

Euclidean Distance 公式是 Minkowski Distance 公式中取$\lambda = 2$。

(3) CityBlock Distance 公式

$$d_{ij} = \sum_{k=1}^{n}\left|x_{ik} - x_{jk}\right| \tag{3-7}$$

CityBlock Distance 公式是 Minkowski Distance 公式中取$\lambda = 1$。

这三个公式的求中心点有些不一样的地方，如图 3 - 4 所示：

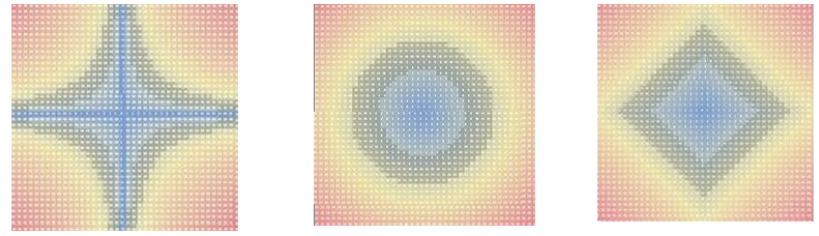



图 3 - 4 三个公式的求中心点逼近方法

Fig.3-4 Three methods of center approaching.

第一个图 Minkowski Distance 以星形的方式，第二个图 Euclidean Distance 以同心圆的方式，第三个图 CityBlock Distance 以菱形的方式。

K-Means 算法是通过逐步迭代的方式求解聚类中心点的。从图 3 - 5 中，可以看到，A，B，C，D，E 是五个高频呼叫站点。而灰色的点是种子点，也就是用来寻找聚类的点。有两个种子点，所以 K=2。

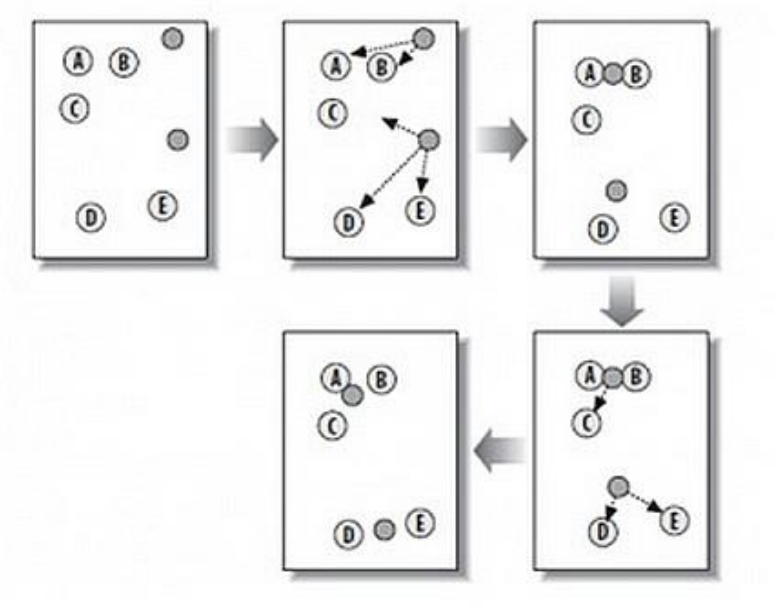



图 3 - 5 K-means 算法过程

Fig.3-5 Procedure of K-means

然后，K-Means 的算法如下：

(1) 随机地在图中取 K（这里 K=2）个种子点。

(2) 然后求图中的所有高频呼叫站点到这 K 个种子点的距离，假如点$P_i$离种子点$S_i$最近，那么$P_i$属于$S_i$聚类。（上图中，可以看到 A，B 属于上面的种子点，C，D，E 属于下面中部的种子点）。

(3) 接下来，移动种子点到属于他的“聚类”的中心。（见图上的第三步）

(4) 然后重复第（2）和第（3）步，直到种子点没有移动（可以看到图中的第四步上面的种子点聚合了 A，B，C，下面的种子点聚合了 D，E）。

使用 K-Means 聚类算法虽然可以比较方便地求得高频呼叫站点的聚类，但是高频呼叫站点之间的关联性却无法保证，即一个叫车请求的起始站点属于聚类α，而该叫车请求的目的站点属于聚类β，在这种情况下如果无人车在聚类中仍然按照贪婪算法的策略行驶就会出现跨聚类的情况，那么这时便退化为了普通的贪婪算法，并没有发挥其聚类的作用。因此本文针对这种特殊情况，提出了一种基于高频呼叫站点关联度的聚类算法。

### 3.4.3 基于高频呼叫站点关联度的聚类算法

本文通过高频呼叫站点$s_i$与$s_j$出现在同一个请求$r_i$的频率来定义高频呼叫站点间的关联度：

$$Q_{s_{ij}}(t)=\frac{\sum_{i=0}^{N(t)} h(s_i,s_j,r_i)}{\sum_{i=0}^{N(t)}|r_i(t)|} \tag{3-8}$$

其中：

$$h(s_i,s_j,r_i)=\begin{cases}1, s_i\in r_i 且 s_j\in r_i\\ 0, 其他\end{cases} \tag{3-9}$$

当$Q_{s_{ij}}(t)>Q_{cor}$则表示站点$s_i$与$s_j$是强关联，$Q_{cor}$为设定的强关联标准频率。

基于站点强关联的聚类算法操作如图 3 - 6 所示：

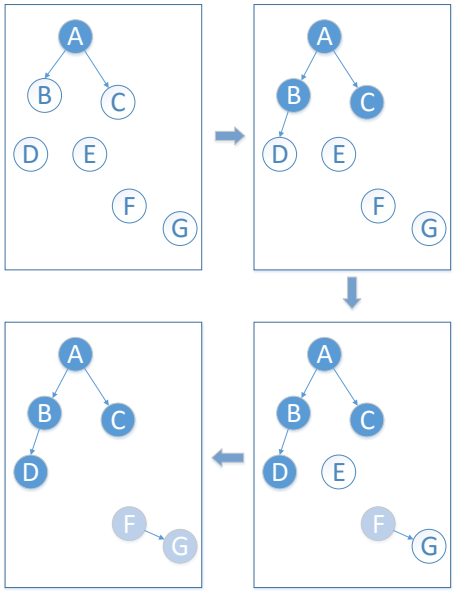


图 3 - 6 基于高频呼叫站点关联度的聚类操作方法
Fig.3-6 Clustering method based on hot stations' relationship

(1) 从某一个高频呼叫站点，比如 A，开始遍历高频呼叫站点，并找出与 A 强关联的高频呼叫站点，如果有与之强关联的站点，B 和 C，则将 A 放入聚类 1（图中深蓝色圆的集合）中。

(2) 将 B，C 放入聚类 1 中，并找出和 B,C 强关联的站点，直到没有强关联站点为止。

(3) 若某个站点已经在某聚类中了，如 B，那么便跳过该站点的遍历，进入到下一个站点的遍历。若某个站点，如 E，没有与之强关联的站点，那么便剔除该高频呼叫站点。

(4) 新聚类，如 F 遍历的与之强关联的站点放入聚类 2（图中灰色圆的集合）。

重复(1)-(4)直到所有高频呼叫站点都遍历了，那么便得到多个高频呼叫站点的聚类。聚类数量的多少和强关联标准频率$Q_{cor}$成负相关关系。

### 3.4.4 面向最优等待时间的聚类贪婪遗传算法

聚类贪婪遗传算法结合了遗传算法和贪婪算法，该算法让部分无人车在高频呼叫站点聚类中按照贪婪算法决定的路径行驶，让另外的无人车在全部站点中按照遗传算法决定的路径行驶，因此如何动态分配用于两种算法的无人车数量是聚类贪婪遗传算法的关键。

聚类贪婪遗传算法从三个维度对无人车进行动态分配：

第一是分配聚类高频呼叫站点和其他站点的无人车，即分配在聚类高频呼叫站点的无人车会按照贪婪算法接受调度，而分配至其他站点的无人车则按照遗传算法接受调度。本文按照比例加权的方式分配无人车数量，聚类高频呼叫站点的数量为：

$$M_{Greedy} = \left\lceil \mathrm{W}|M|\frac{|S_C|}{|S|} \right\rceil \tag{3-10}$$

其中，W 为设定的权重值，$|M|$表示无人车总数，$|S_c|$表示聚类的高频呼叫站点总数，$|S|$表示站点总数，$\lceil\ \rceil$是向上取整符号。那么，

$$M_{Genetic} = |M| - M_{Greedy} \tag{3-11}$$

第二是分配各高频呼叫站点聚类的无人车具体数量。因为当分配的无人车越多时，请求的平均等待时间和处理时间均下降，本章主要对叫车请求的等待时间进行优化，因此本文采用$\bar{I}_t$作为动态调节各高频呼叫站点聚类中无人车数量的依据，即当某聚类 A 中$\bar{I}_t$达到最小，那么动态分配出一辆无人车给$\bar{I}_t$最大的聚类，每

隔一段时间$T_2$进行一次这样的动态分配。

第三是随着时间的增长，高频呼叫站点会发生改变，因此每隔一段时间$T_3$（$T_3 \gg T_2$）,需要重新生成高频呼叫站点，再次按照上文的方式得出高频呼叫站点聚类，接着进行第一二中的动态分配。

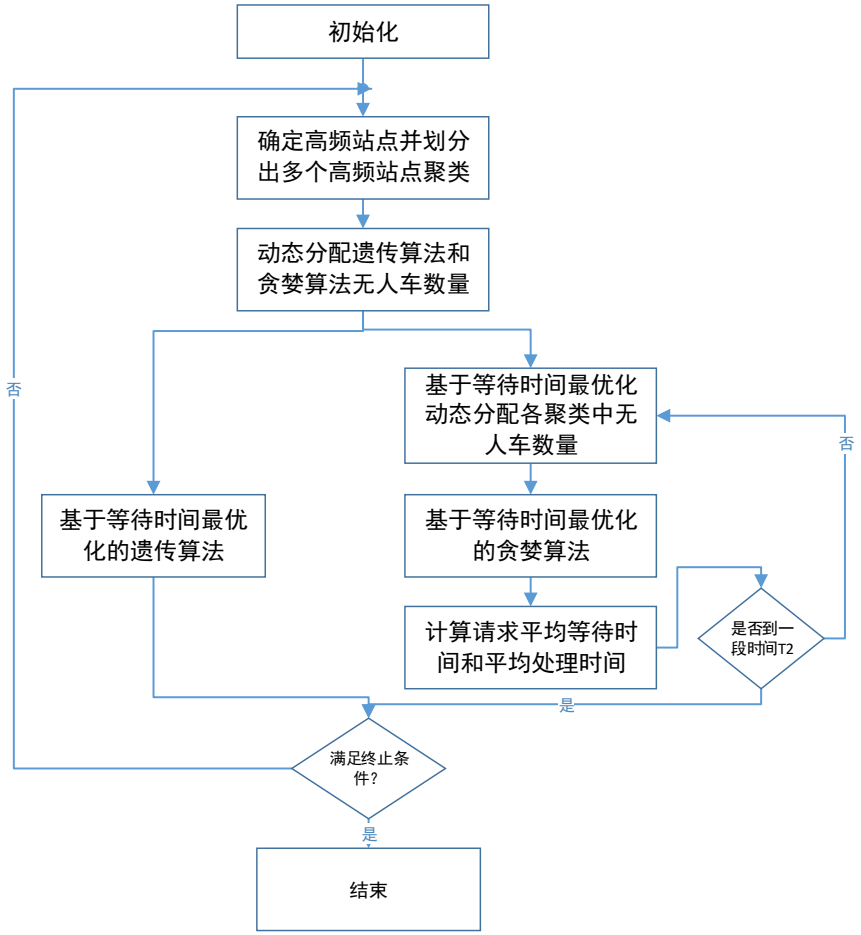



图 3 - 7 面向最优等待时间的聚类贪婪遗传算法流程图

Fig.3-7 Flow chart of cluster greedy genetic algorithm based on the optimization of waiting time

面向最优等待时间的聚类贪婪遗传算法流程图如图 3 - 7 所示，该算法结合了贪婪算法和遗传算法，它让部分无人车在高频呼叫站点聚类中按照贪婪算法决定的路径行驶，让另外的无人车在全部站点中按照遗传算法决定的路径行驶。算法中采用动态分配无人车，充分地利用了遗传算法和贪婪算法的优点。

## 3.5 实验与分析

### 3.5.1 仿真系统设计

为了模拟全自动交通系统中无人车运行、用户叫车的场景，同时为了测试上文三种不同算法的效率，本章专门设计一个全自动交通系统仿真系统用于演示全自动交通系统运行状况，同时对比实验三种不同算法，从而得出结论。

自动交通系统仿真系统可以仿真在不同调度算法下全自动交通系统中无人车运行，叫车请求处理等情况，提供给操作者修改站点常量、无人车常量、数据采集环境等的修改功能，同时允许操作者配置不同的叫车请求生成策略，无人车调度算法。全自动交通系统仿真系统需求分析如表 3 - 1 所示：

表 3 - 1 全自动交通系统仿真系统需求分析

Table.3-1 Requirements analysis for Cybernetic Transportation System

| 属性 / 需求 | 功能点 | 功能点描述 |
|---|---|---|
| 常量配置需求 | 站点常量配置 | 在配置文件中可以修改站点总数，站点之间距离矩阵 |
| | 无人车常量配置 | 无人车总数量，速度，准载人数 |
| | 数据采集环境配置 | 采集数据周期，总时长，数据输出文件 |
| 策略配置需求 | 叫车请求生成策略配置 | 配置叫车请求的生成频率，数量上下限，按实际情况的分布模型 |
| | 无人车调度算法配置 | 能够在遗传算法、贪婪算法以及聚类遗传贪婪算法中切换 |

根据上述需求分析，全自动交通系统仿真系统采用如图 3 - 8 所示的架构图。该仿真系统采用模块化设计，主要分为配置参数、叫车请求生成器、叫车请求管理器、车辆、计算中心、数据输入输出等模块，各个模块通过生命周期管理容器和事件总线有序的组织起来。

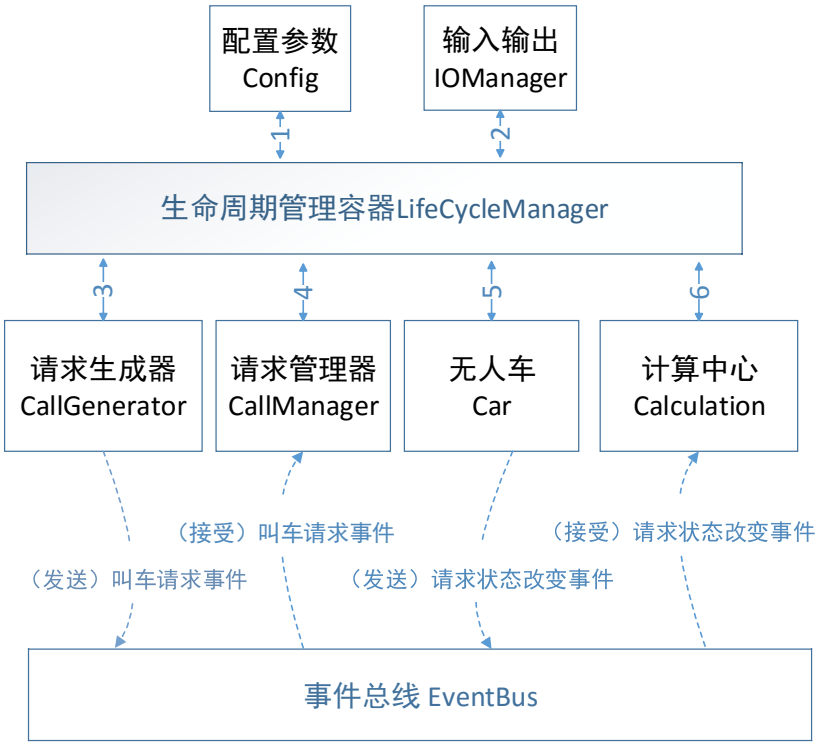



图 3 - 8 全自动交通系统仿真系统架构图

Fig.3-8 Structure of the simulation system of Cybernetic Transportation System

叫车请求生成器会每隔一段时间加权随机生成一个叫车请求，请求生成好后叫车请求生成器会把该消息发送给事件总线；事件总线会实时把该消息分发给叫车请求管理器。叫车请求管理器管理了所有的叫车请求，主要会和无人车进行交互，二者之间的交互也是在事件总线中通过消息分发和接受的方式进行的。

当车辆开始处理或者已经处理完毕一个叫车请求后会给事件总线发送请求状态改变的消息，用以告知叫车请求状态的关心者叫车请求发生了变化，计算中心会从事件总线中收到订阅的该类消息，然后进行统计计算。计算中心会统计计算出请求实时的平均等待时间、平均处理时间、最大等待时间以及处理数量，并且通过 IO 管理器把统计后的数据输出到指定文件中。

根据上述系统架构图，本文采用如图 3 - 9 所示的技术路线图进行仿真系统的开发。

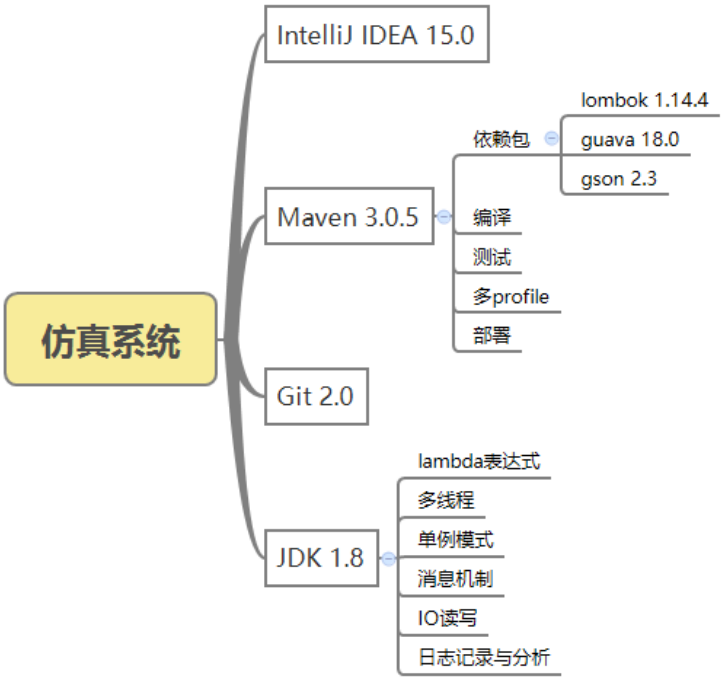



图 3 - 9 全自动交通系统仿真系统技术路线图

Fig.3-9 Technology roadmap of the simulation system of Cybernetic Transportation System

### 3.5.2 实验参数配置

本文以上海交通大学校园作为仿真实验对象，共选取了校园中 36 个具有典型代表的站点，如图 3 - 10 所示。

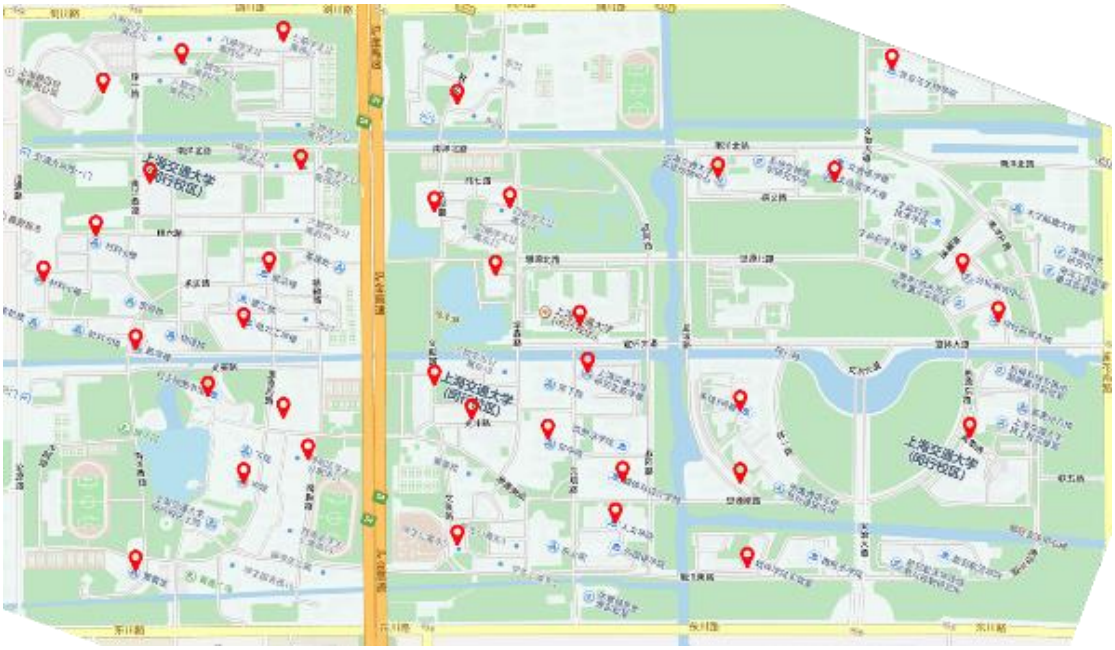


图 3 - 10 上海交通大学地图
Fig.3-10 Map of Shanghai Jiao Tong University

本实验总尽可能地针对较大规模的无人车、站点以及叫车请求，从而较准确地对本章中的三种算法进行验证。本实验中参数设置如表 3 - 2 所示。

表 3 - 2 面向最优等待时间的仿真实验参数设置
Table.3-2 Setting of experiments' parameters based on the optimization of waiting time

| **参数** | **值** |
|---|---|
| 站点总数 | 36 个 |
| 无人车总数 | 16 辆 |
| 无人车速度 | 5m/s |
| 每辆无人车准载人数 | 10 人 |
| 叫车请求生成周期 | 1~4 s |
| 数据采集周期 | 10min |
| 数据采集总时长 | 200min |
| 交叉概率$P_c$ | 1 |
| 变异概率$P_m$ | 0.015 |
| 强关联标准频率$Q_{cor}$ | 0.002 |
| 高频呼叫站点标准频率$P_{hot}$ | 0.28 |

### 3.5.3 结果与分析

在上文实验参数的条件下，面向最优等待时间的叫车请求平均等待时间的三种不同算法实验数据如附录 1 所示。

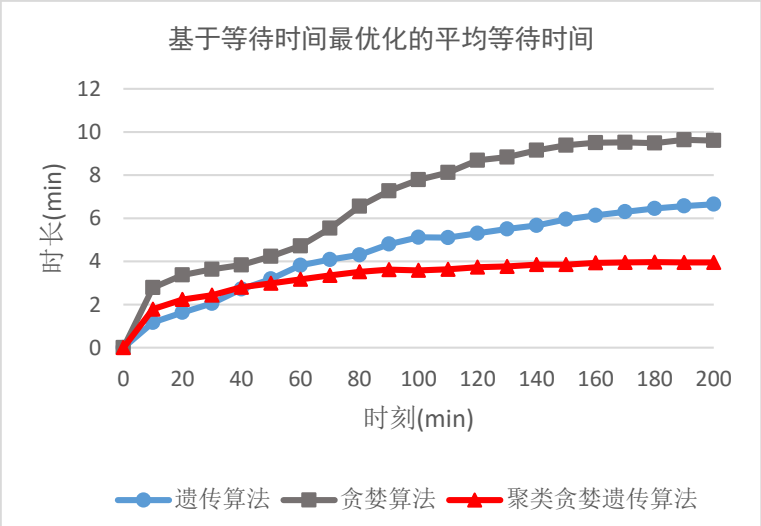



图 3 - 11 面向最优等待时间的平均等待时间
Fig.3-11 Average waiting time based on the optimization of waiting time

根据平均等待时间数据绘制出平均等待时间折线图如图 3 - 11 所示。面向最优等待时间的三种不同算法在 200 分钟时间内的叫车请求平均等待时间一开始是逐渐增高的，并且会随着时间的推移而逐渐趋于稳定。从图中可观察到在整个过程中，贪婪算法的平均等待时间均明显高于其他二者，在系统刚启动时遗传算法的平均等待时间是低于聚类贪婪遗传算法的，而当系统进入稳态时聚类贪婪遗传算法迅速进入平稳期，并且平均等待时间一直维持在较低的值；因此就平均等待时间而言，遗传算法在短时间内是三者中最优的算法，而当时系统随着时间推移趋于稳定时聚类贪婪遗传算法优势愈发明显。

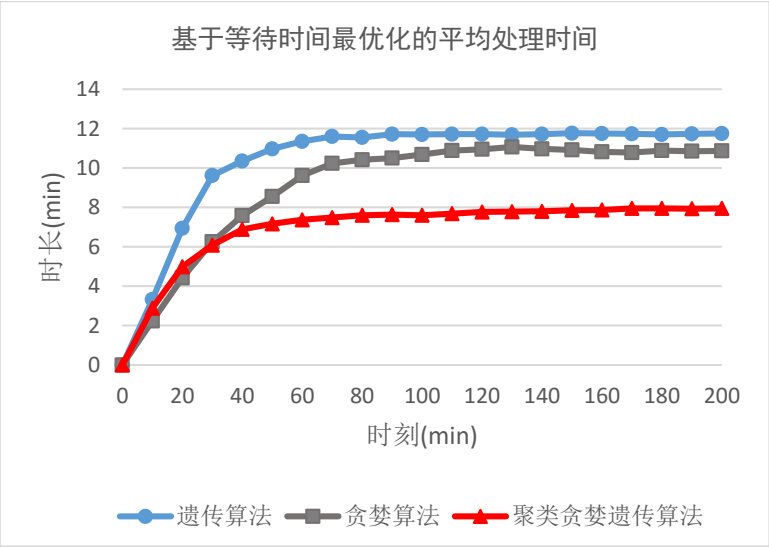



图 3 - 12 面向最优等待时间的平均处理时间

Fig.3-12 Average handling time based on the optimization of waiting time

根据平均处理时间数据绘制出平均处理时间折线图如图 3 - 12 所示。面向最优等待时间的三种不同算法在 200 分钟时间内的叫车请求平均处理时间一开始是逐渐增高的，并且会随着时间的推移而逐渐趋于稳定。从图中可观察到在系统刚启动时遗传算法和贪婪算法的平均处理时间是低于聚类贪婪遗传算法的，而当系统进入稳态时聚类贪婪遗传算法迅速进入平稳期，并且平均处理时间一直维持在较低的值，而贪婪算法趋于平稳时平均处理时间高于聚类贪婪遗传算法而低于遗传算法；因此就平均处理时间而言，贪婪算法在短时间内有较低的平均处理时间，当系统趋于稳定时仍有不错的平均处理时间表现，因此贪婪算法相比于遗传算法在平均处理时间方面具有明显的优势。而聚类贪婪遗传算法虽然在一开始表现不如贪婪算法，但是当系统越趋于稳定，其优势就愈明显。

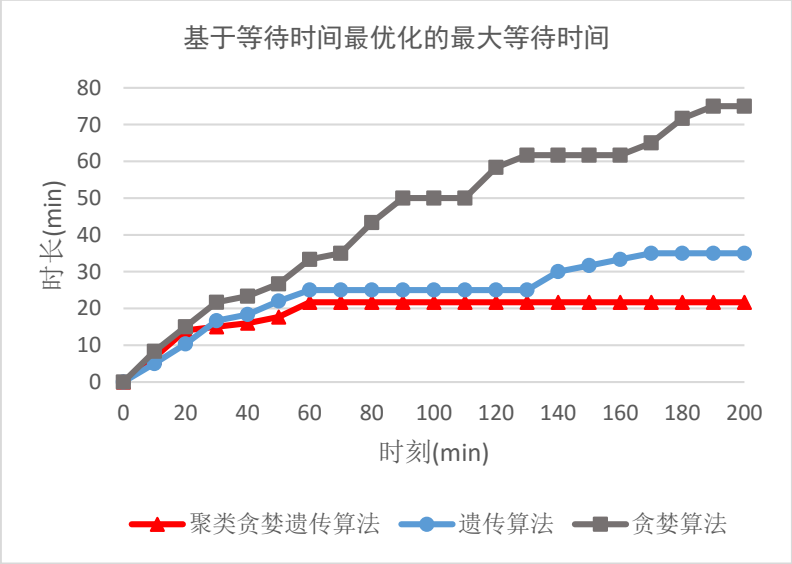



图 3 - 13 面向最优等待时间的最大等待时间

Fig.3-13 Maximum waiting time based on the optimization of waiting time

图 3 - 13 展示了面向最优等待时间的三种不同算法在 200 分钟时间内的叫车请求最大等待时间的情况，采用遗传算法和贪婪算法时最大等待时间不停增大，贪婪算法中甚至可能出现某些叫车请求等待时间过长而导致饿死的状态，而聚类贪婪遗传算法最大等待时间基本趋于稳定，结果明显优于前两种算法。

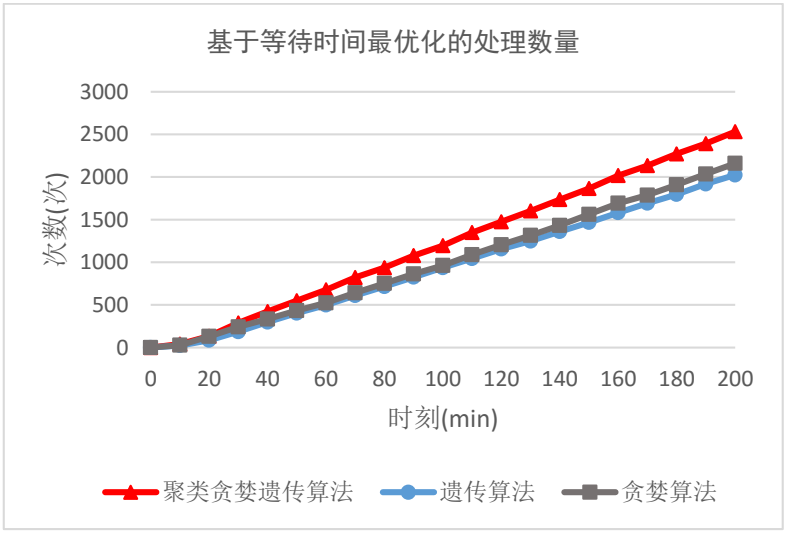



图 3 - 14 面向最优等待时间的处理数量

Fig.3-14 No. of handled calling requests based on the optimization of waiting time

图 3 - 14 展示了采用三种不同算法在 200 分钟时间内系统处理的叫车请求数量的情况，结果显示三种算法在该项目标最优化上比较接近，贪婪算法略优于遗传算法，而聚类遗传算法略优于贪婪算法。

面向最优等待时间的贪婪算法和遗传算法两种传统的算法中，贪婪算法时叫车请求的平均处理时间更短，处理叫车请求数量更多，但是贪婪算法容易造成部分叫车请求饿死，这和贪婪算法总是优先处理等待时间最长的叫车请求，处理时优先处理站点最近的请求有关。而采用遗传算法在平均等待时间和最大等待时间上相较于贪婪算法具有明显的优势。

由于聚类贪婪遗传算法中部分车辆使用贪婪算法只在高频呼叫站点的聚类中规划行车路线，这样保证了在高频呼叫站点中的叫车请求可以快速的响应执行，同时剩余车辆又在所有站点中规划路径，这样保证了不会有叫车请求饿死。并且实验结果也显示出了聚类贪婪遗传算法结合了前两种算法的优点，有效地规避了其缺点。

## 3.6 本章小结

本章从解决传统车辆调度问题的思路出发，对遗传算法的产生和发展，基本思路以及其特点做了相应的概述，接着针对本文实际情况研究了面向最优等待时间的遗传算法在全自动交通系统中的具体方法和步骤。然后本章采用面向最优等待时间的贪婪算法来求解全自动交通系统中的车辆调度问题，给出了相应的流程图和伪代码。本章针对遗传算法和贪婪算法的缺点，并保留二者优点，提出聚类贪婪遗传算法，在求解高频呼叫站点聚类群时提出基于高频呼叫站点关联度的聚类算法，进而得出聚类贪婪遗传算法的流程图。最后本章根据全自动交通系统的实际情况，分析了具体需求，设计并开发了一个全自动交通系统的仿真系统，并且在仿真系统中使用上文中研究的三种算法进行对比实验，经过对比实验得出面向最优等待时间的贪婪算法在平均处理时间，叫车请求数量上比遗传算法有优势，而遗传算法则在平均等待时间，最大等待时间方面比贪婪算法有优势。聚类贪婪遗传算法虽然在系统刚启动时在多项优化目标不及前二者，但是当系统趋于稳定时在四项优化目标中均具有明显优势。

# 第四章 面向最优处理时间的算法研究

## 4.1 引言

全自动交通系统中的车辆调度算法直接影响到第 2 章中的四个优化目标，但有可能存在不同优化目标之间产生互斥，即可能存在当系统中叫车请求平均等待时间较短时，而系统中的叫车请求平均处理时间变长的情况。第 3 章面向最优等待时间进行了三种算法的研究，本章将从平均处理时间这一优化目标出发，进行基于平均处理时间最优化的算法研究。

本章仍主要针对遗传算法、贪婪算法和聚类贪婪遗传算法在第 2 章的基础上继续进行研究。面向最优处理时间的遗传算法需要重新选择适应度函数，而贪婪算法则需要重新确定贪婪条件以及重新设计算法流程图和伪代码。聚类贪婪遗传算法同样需要根据遗传算法和贪婪算法的调整做出相应的调整。然后本章使用上述三种算法进行仿真实验，并对实验结果进行分析。最后再将面向最优等待时间的算法结果和面向最优处理时间的算法结果对比分析。

## 4.2 遗传算法

遗传算法通过模拟生物进化的自然选择和遗传法则来实现的，面向最优处理时间的遗传算法同样包括编码、适应度、选择、交叉、变异等要点，和面向最优等待时间的遗传算法不同的是适应度函数的选择：

**适应度**：物种的适应度体现为对环境的适应能力，该能力也是生物进化的驱动力。遗传算法中适应度函数是用来评价解好坏的标准，解的质量和适应度函数值成正相关关系。而当用户请求的处理时间作为最优化目标时，应采用在规划路径下的处理时间总和的倒数作为适应度函数，即：

$$\mathrm{F} = \frac{v_k}{\sum_{i=1}^{n} |f_i t_i|_p} \tag{4-1}$$

其中，p 表示某一个个体解，$|f_i t_i|_p$表示叫车请求$r_i$在个体解 p 下的路径长度。

面向最优处理时间的遗传算法的流程图如图 4 - 1 所示：

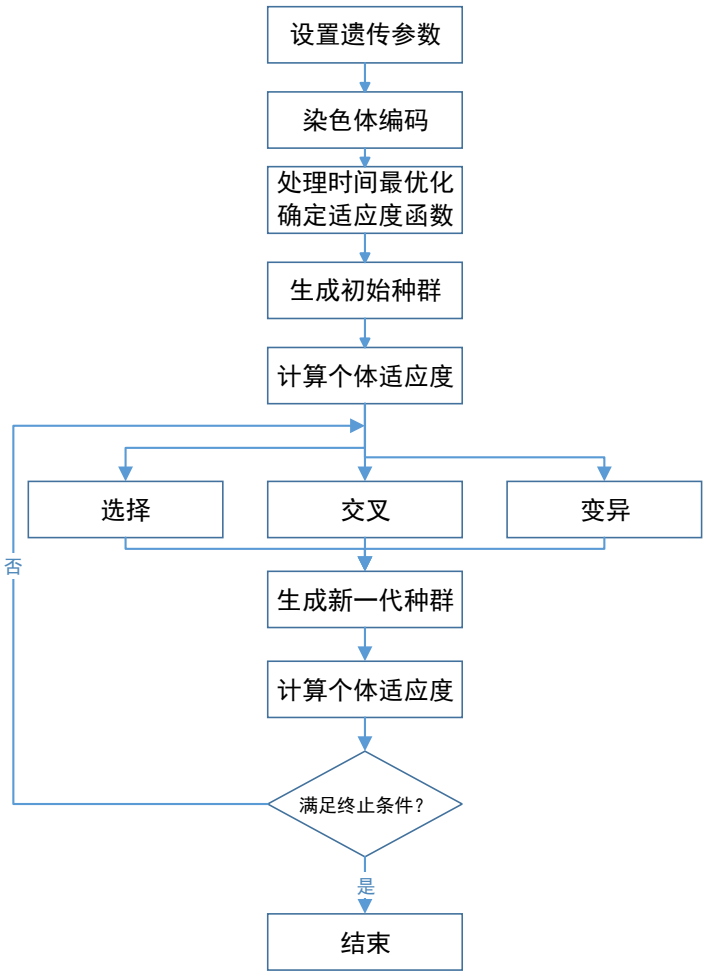



图 4 - 1 面向最优处理时间的遗传算法流程图

Fig.4-1 Flow chart of genetic algorithm based on the optimization of handling time

**Algorithm 3. Genetic Algorithm**

$\mathbf{t} \leftarrow 0$

**initialize** $\mathbf{Pop(t)}$ **with N chromosomes** $\boldsymbol{Pop_i(t)}$

**while** (**not terminate condition**) ***do***

  **for i** $\leftarrow$ **to N do**

    $\boldsymbol{f_i} \leftarrow \boldsymbol{f(Pop_i(t),\ J_i)}$

  **for i** $\leftarrow$ **to N do**

    $\mathbf{NewPop_i(t+1)} \leftarrow$ **roulette choose** $\boldsymbol{Pop_i(t)} \in \mathbf{Pop(t)}$ **with** $\boldsymbol{p_j} = \frac{f(j)}{\sum_k f(k)}$

  $\mathbf{CrossPop(t+1)} \leftarrow$ **recombine** $\big(\mathbf{NewPop(t+1)}\big)$ **with** $\boldsymbol{P_c}$

  $\mathbf{MutPop(t+1)} \leftarrow \mathbf{mutate(CrossPop(t+1))}$ **with** $\boldsymbol{P_m}$

  $\mathbf{Pop(t+1)} \leftarrow \mathbf{MutPop(t+1)}$

  $\mathbf{t} \leftarrow \mathbf{t+1}$

**while**(**not terminate condition**) **do**

**car moves by the path of Pop(T).**

面向最优处理时间的遗传算法伪代码如 Algorithm 3，其中:

(1) 初始化 t 代表设置进化的代数。

(2) $f_i$代表第 i 个个体的适应度。

(3) $p_j$ 代表了个体 j 根据其适应度被选择遗传的概率。

(4) $P_c$ 为交叉概率。

(5) $P_m$ 为变异概率。

在全自动交通系统中，用户叫车请求的分布是随时间变化的，上文中是使用某一时刻的叫车请求分布来计算适应度函数。面向最优处理时间的遗传算法是一种动态的方法，系统会在某辆无人车出发前为其生成一条特有的路径，该路径是采用遗传算法生成，其中的适应度函数是当前所有未处理的叫车请求的总处理时间的倒数，每当无人车按照该路径运行一周时，系统便又按照上述方法给无人车生成一条新的路径。

## 4.3 贪婪算法

本章首要考虑用户的叫车请求处理时间最优化，因此使用贪婪算法时首要以降低用户叫车请求的处理时间为目的，因此主要有以下两个贪婪原则：

(1) 对于空闲无人车（即当前没有乘客且尚无下一步任务的无人车），中央控制系统总是让其优先处理离自己最近站点的叫车请求。

(2) 对于非空闲无人车（即载有乘客且乘客尚未到达目的站点的无人车），中央控制系统总是让其优先处理最先上车的用户的叫车请求。

根据以上贪婪原则，得到面向最优处理时间的贪婪算法流程图如图 4 - 2 所示：

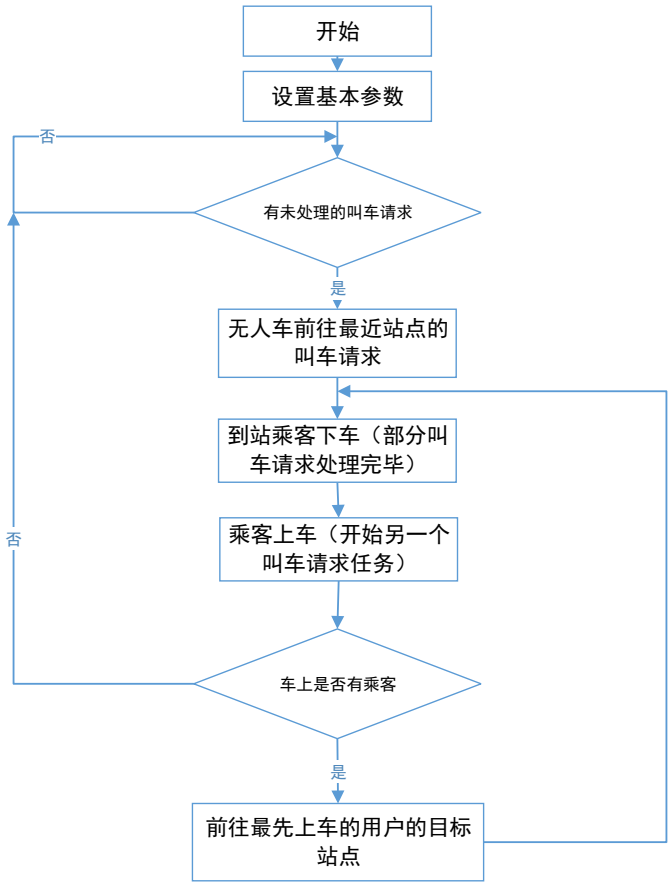



图 4 - 2 面向最优处理时间的贪婪算法流程图

Fig.4-2 Flow chart of greedy algorithm based on the optimization of handling time

该算法的伪代码如 Algorithm 4 所示，伪代码中数学符号均采用 2.3 节中的数学实体模型。

**Algorithm 4. Greedy Algorithm**

$\mathbf{t} \leftarrow 0$
**initialize car m**
**while** (**not terminate condition**) ***do***
  **if** (**exists any waiting requests**) ***then***
    $\boldsymbol{find}\ r_m\ \boldsymbol{where}\ r_m.I_m = \max(r_i.I_i, r_i \in R_t)$
    Add $r_m$ in $\mathbf{m}.R[L_i]$
    while ($\mathbf{m}.R[L_i]$ not empty) then
      **Add all** $\mathbf{r}_j$ **with** $\mathbf{r}_j.\mathbf{f}_j = r_m.f_m$ **in** $\mathbf{m}.R[L_i]$
      $\mathbf{m}.\mathbf{s} \leftarrow r_m.I_m$
      **Remove** $\mathbf{r}_k$ **with** $\mathbf{r}_k.\mathbf{f}_k = r_m.f_m$ **in** $\mathbf{m}.R[L_i]$
      $\boldsymbol{find}\ r_m,\ \boldsymbol{where}\ |m.s - r_m.f_m| = \min(|m.s - r_i.I_i|, r_i \in \mathbf{m}.R[L_i])$
      $\mathbf{m}.\mathbf{s} \leftarrow r_m.I_m$
  $\mathbf{t} \leftarrow \mathbf{t} + \mathbf{1}$

## 4.4 聚类贪婪遗传算法

面向最优处理时间的聚类贪婪遗传算法与面向最优等待时间的聚类贪婪遗传算法的思想相同，都是让部分无人车在高频呼叫站点聚类中按照贪婪算法决定的路径行驶，让另外的无人车在全部站点中按照遗传算法决定的路径行驶。

面向最优处理时间的聚类贪婪遗传算法同样从三个维度对无人车进行动态分配：

第一是分配聚类高频呼叫站点和其他站点的无人车，此维度的分配方式和面向最优等待时间的聚类贪婪遗传算法相同，据采用按照加权比例的方式分配。

第二是分配各高频呼叫站点聚类的无人车具体数量。当某个高频聚类中分配的无人车越多，该聚类中的请求平均等待时间和处理时间均下降，本章主要对叫车请求的处理时间进行优化，因此本章采用$\bar{J}_t$作为动态调节各高频呼叫站点聚类中无人车数量的依据，即当某聚类 A 中$\bar{J}_t$达到最小，那么动态分配出一辆无人车给$\bar{J}_t$最大的聚类，每隔一段时间$T_2$进行一次这样的动态分配。

第三是随着时间的增长，高频呼叫站点会发生改变，因此每隔一段时间$T_3$（$T_3 \gg T_2$）,需要重新生成高频呼叫站点，再次按照上文的方式得出高频呼叫站点聚类，接着进行第一二中的动态分配。

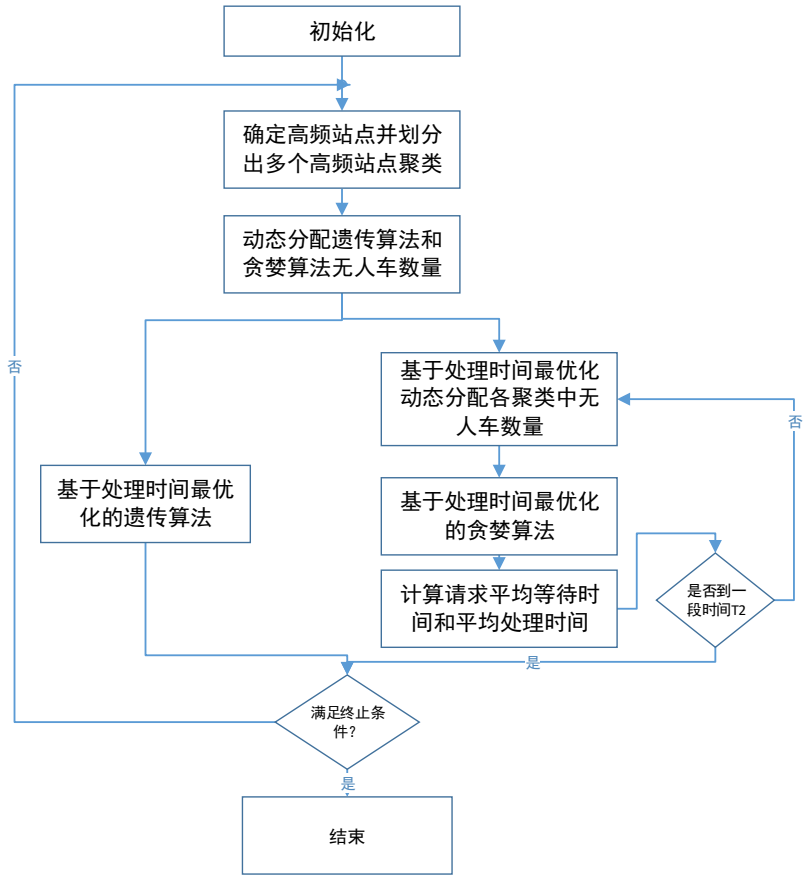



图 4 - 3 面向最优处理时间的聚类贪婪遗传算法流程图

Fig.4-3 Flow chart of cluster greedy genetic algorithm based on the optimization of handling time

面向最优处理时间的聚类贪婪遗传算法流程图如图 4 - 3 所示，该算法结合了贪婪算法和遗传算法，它让部分无人车在高频呼叫站点聚类中按照贪婪算法决定的路径行驶，让另外的无人车在全部站点中按照遗传算法决定的路径行驶。算法中采用动态分配无人车，充分地利用了遗传算法和贪婪算法的优点。

## 4.5 实验与分析

### 4.5.1 实验参数配置

本章采用第三章中所开发的全自动交通系统仿真系统作为实验平台进行仿真实验，同样以上海交通大学校园作为仿真实验对象，共选取了校园中 36 个具有典型代表的站点，如图 3 - 10 所示。

本实验总尽可能地针对较大规模的无人车、站点以及叫车请求，从而较准确地对本章中的三种算法进行验证。本实验中参数设置如表 4 - 1 所示。

表 4 - 1 面向最优处理时间的仿真实验参数设置

Table.4-1 Setting of experiments' parameters based on the optimization of handling time

| 参数 | 值 |
|---|---|
| 站点总数 | 36 个 |
| 无人车总数 | 16 辆 |
| 无人车速度 | 5m/s |
| 每辆无人车准载人数 | 10 人 |
| 叫车请求生成周期 | 1~4 s |
| 数据采集周期 | 10min |
| 数据采集总时长 | 200min |
| 交叉概率$P_c$ | 1 |
| 变异概率$P_m$ | 0.015 |
| 强关联标准频率$Q_{cor}$ | 0.002 |
| 高频呼叫站点标准频率$P_{hot}$ | 0.28 |

### 4.5.2 结果与分析

在上文实验参数的条件下，面向最优处理时间的叫车请求平均等待时间的三种不同算法实验数据如附录 2 所示

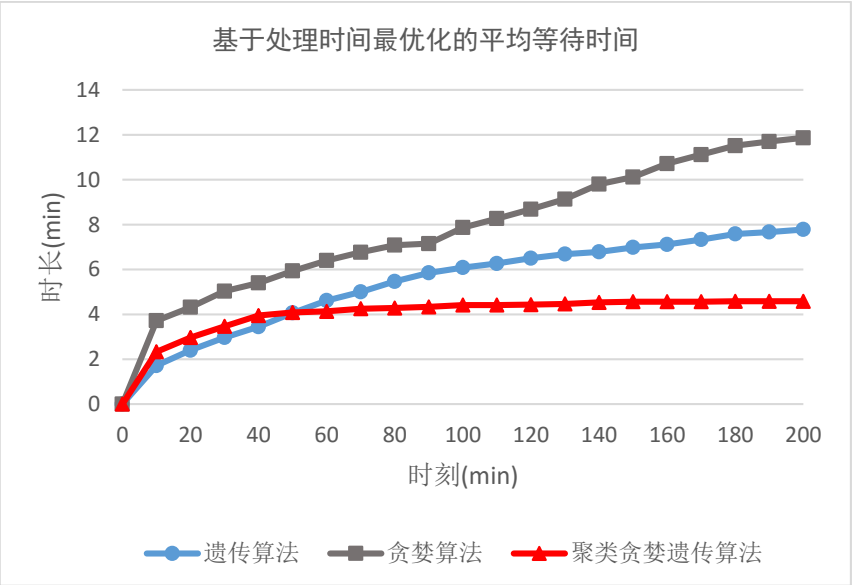



图 4 - 4 面向最优处理时间的平均等待时间

Fig.4-4 Average waiting time based on the optimization of handling time

根据平均等待时间数据绘制出平均等待时间折线图如图 4 - 4 所示。面向最优处理时间的三种不同算法在 200 分钟时间内的叫车请求平均等待时间一开始是逐渐增高的，并且会随着时间的推移而逐渐趋于稳定。从图中可观察到在整个过程中，贪婪算法的平均等待时间均明显高于其他二者，在系统刚启动时遗传算法的

平均等待时间是低于聚类贪婪遗传算法的，而当系统进入稳态时聚类贪婪遗传算法迅速进入平稳期，并且平均等待时间一直维持在较低的值；因此就平均等待时间而言，遗传算法在短时间内是三者中最优的算法，而当时系统随着时间推移趋于稳定时聚类贪婪遗传算法优势愈发明显。

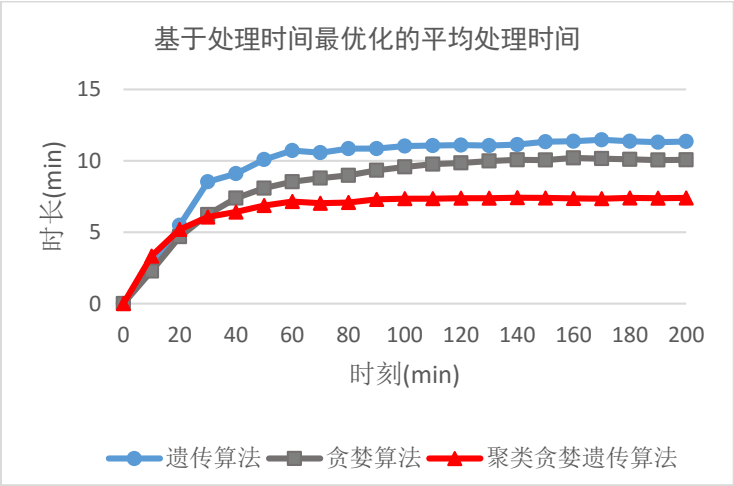



图 4-5 面向最优处理时间的平均处理时间

Fig.4-5 Average handling time based on the optimization of handling time

根据平均处理时间数据绘制出平均处理时间折线如图 4-5 所示。面向最优处理时间的三种不同算法在 200 分钟时间内的叫车请求平均处理时间一开始是逐渐增高的，并且会随着时间的推移而逐渐趋于稳定。从图中可观察到在系统刚启动时遗传算法和贪婪算法的平均处理时间是低于聚类贪婪遗传算法的，而当系统进入稳态时聚类贪婪遗传算法迅速进入平稳期，并且平均处理时间一直维持在较低的值，而贪婪算法趋于平稳时平均处理时间高于聚类贪婪遗传算法而低于遗传算法；因此就平均处理时间而言，贪婪算法在短时间内有较低的平均处理时间，当系统趋于稳定时仍有不错的平均处理时间表现，因此贪婪算法相比于遗传算法在平均处理时间方面具有明显的优势。而聚类贪婪遗传算法虽然在一开始表现不如贪婪算法，但是当系统越趋于稳定，其优势就愈明显。

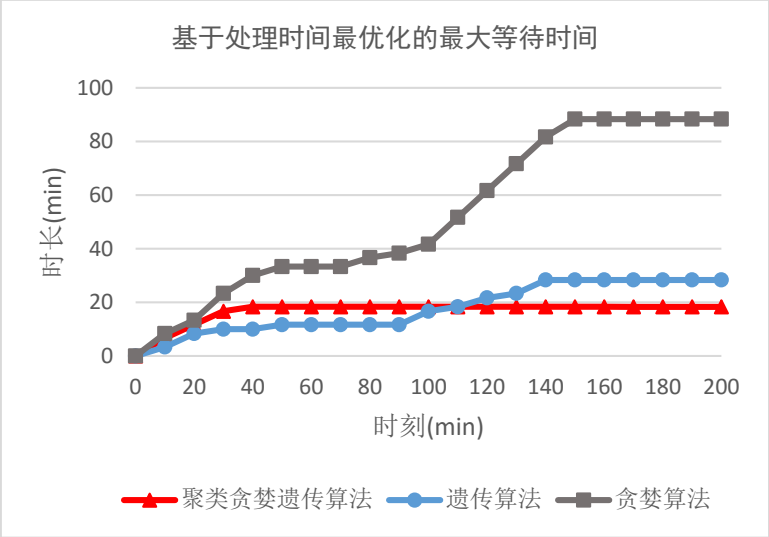



图 4 - 6 面向最优处理时间的最大等待时间

Fig.4-6 Maximum waiting time based on the optimization of handling time

图 4 - 6 展示了面向最优处理时间的三种不同算法在 200 分钟时间内的叫车请求最大等待时间的情况，采用遗传算法和贪婪算法时最大等待时间不停增大，贪婪算法中甚至可能出现某些叫车请求等待时间过长而导致饿死的状态，而聚类贪婪遗传算法最大等待时间基本趋于稳定，结果明显优于前两种算法。

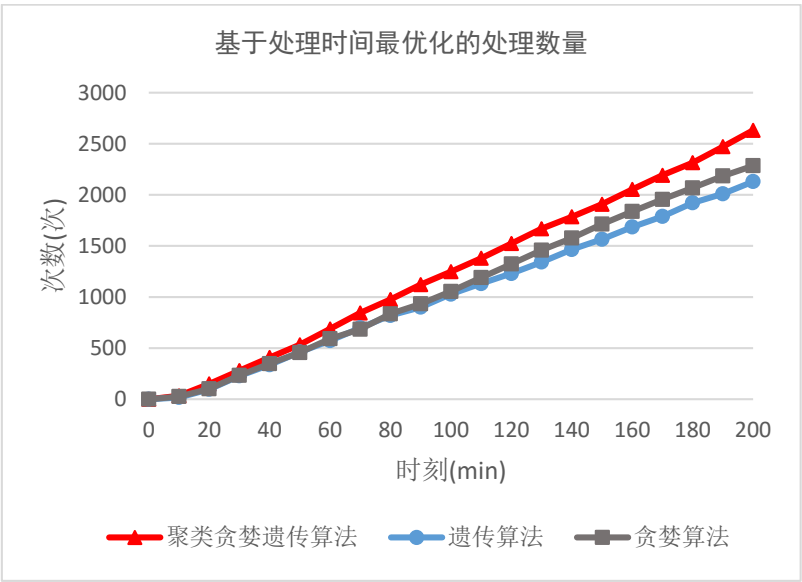



图 4 - 7 面向最优处理时间的处理数量

Fig.4-7 No. of handled calling requests based on the optimization of handling time

图 4 - 7 展示了采用三种不同算法在 200 分钟时间内系统处理的叫车请求数

量的情况，结果显示三种算法在该项目标最优化上比较接近，贪婪算法略优于遗传算法，而聚类遗传算法略优于贪婪算法。

面向最优处理时间的贪婪算法和遗传算法两种传统的算法中，贪婪算法时叫车请求的平均处理时间更短，处理叫车请求数量更多，但是贪婪算法容易造成部分叫车请求饿死，这和贪婪算法总是优先处理等待时间最长的叫车请求，处理时优先处理站点最近的请求有关。而采用遗传算法在平均等待时间和最大等待时间上相较于贪婪算法具有明显的优势。

由于聚类贪婪遗传算法中部分车辆使用贪婪算法只在高频呼叫站点的聚类中规划行车路线，这样保证了在高频呼叫站点中的叫车请求可以快速的响应执行，同时剩余车辆又在所有站点中规划路径，这样保证了不会有叫车请求饿死。并且实验结果也显示出了聚类贪婪遗传算法结合了前两种算法的优点，有效地规避了其缺点。

## 4.6 不同优化目标的对比实验与分析

全自动交通系统是一个具有多维度多优化目标的复杂系统，各优化目标对应了不同的优化策略。第三章研究了面向最优等待时间的遗传算法，贪婪算法以及聚类贪婪遗传算法，同时针对三种算法各自的特点提出相应的优化策略，并且采用三种算法进行了仿真实验与分析。本章在第三章的基础上以缩短处理时间作为优化目标，对遗传算法，贪婪算法以及聚类贪婪遗传算法均做了相应优化调整，并且进行了仿真实验与分析。由上文的实验结果可知，在系统趋于稳定时聚类贪婪遗传算法在各个优化目标函数下均表现最好，因此本节拟针对不同优化目标下聚类贪婪遗传算法的实验结果做对比分析。

第三章和本章的仿真实验参数设置均一致，因此本节使用第三章与本章第 5 节的实验数据作为对比实验的实验数据。

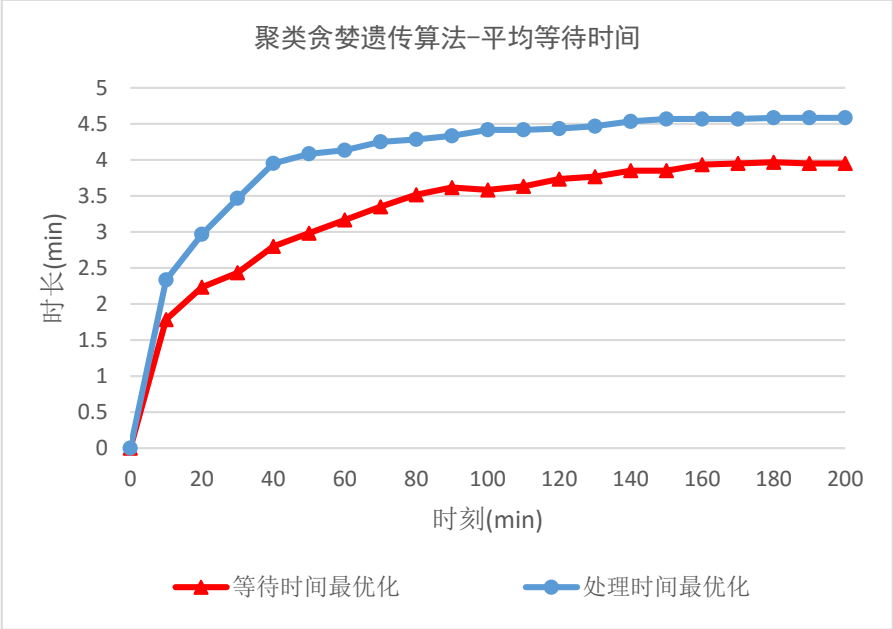



图 4-8 聚类贪婪遗传算法平均等待时间对比图

Fig.4-8 Comparison of waiting time in clustering greedy genetic algorithm

面向最优等待时间和面向最优处理时间的聚类贪婪遗传算法的平均等待时间对比图如图 4-8 所示，从图中可以观察到两组数据均随着时间的推移而逐渐趋于稳定，并且以等待时间为优化目标的聚类贪婪算法的平均等待时间明显优于以处理时间为优化目标的结果。

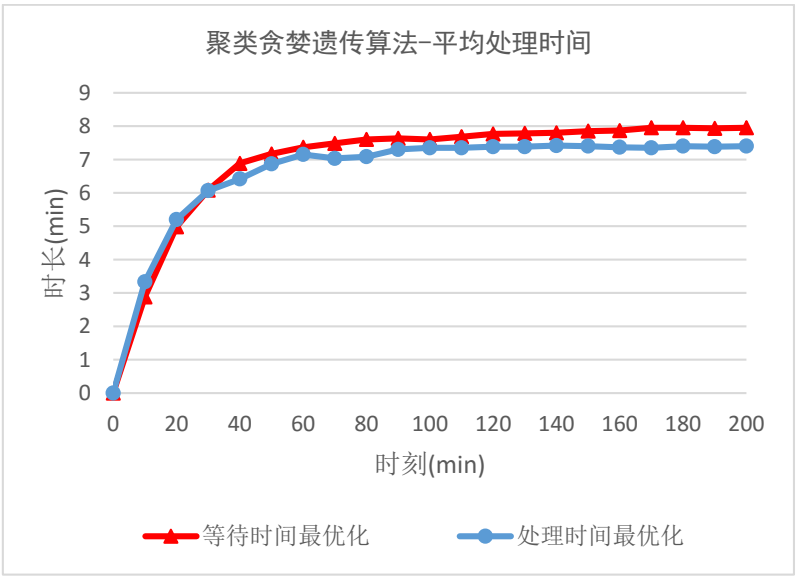



图 4-9 聚类贪婪遗传算法平均处理时间对比图

Fig.4-9 Comparison of handling time in clustering greedy genetic algorithm

面向最优等待时间和面向最优处理时间的聚类贪婪遗传算法的平均等待时间

对比图如图 4 - 9 所示，从图中可以观察到两组数据均随着时间的推移而逐渐趋于稳定，并且以处理时间为优化目标的聚类贪婪算法的平均处理时间明显优于以等待时间为优化目标的结果。

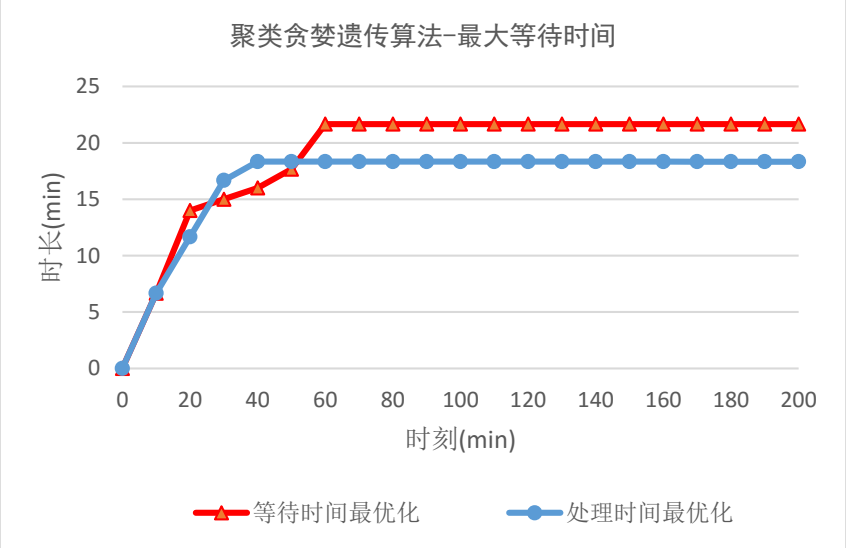



图 4 - 10 聚类贪婪遗传算法最大等待时间对比图

Fig.4-10 Comparison of maximum waiting time in clustering greedy genetic algorithm

面向最优等待时间和面向最优处理时间的聚类贪婪遗传算法的最大等待时间对比图如图 4 - 10 所示，从图中可以观察到两组数据均随着时间的推移而逐渐趋于稳定，并且以处理时间为优化目标的聚类贪婪算法的最大等待时间明显优于以等待时间为优化目标的结果。因此从该组对比实验中可以得出基于平均等待时间最优化的聚类贪婪遗传算法并不能够让该算法的最大等待时间这一优化指标达到最优。

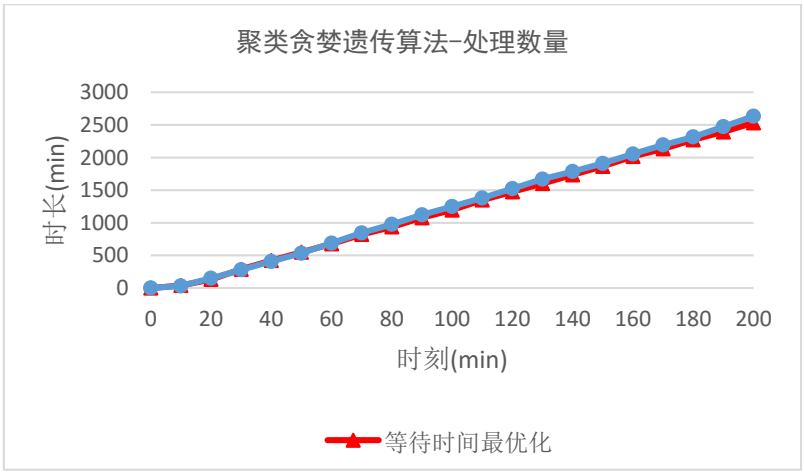



图 4 - 11 聚类贪婪遗传算法叫车请求处理数量对比图

Fig.4-11 Comparison of number of calling requests in clustering greedy genetic algorithm

面向最优等待时间和面向最优处理时间的聚类贪婪遗传算法的叫车请求处理数量对比图如图 4 - 11 所示，从图中可以观察到两组数据均随着时间的推移均成线性增加，并且二者在叫车请求数量上结果非常接近，因此以等待时间和处理时间作为优化条件的聚类贪婪遗传算法对叫车请求数量的影响较小。

从上文四组对比实验中得出以下结论，面向最优等待时间的聚类贪婪遗传算法能够有效降低系统中叫车请求的平均等待时间，而面向最优处理时间的聚类贪婪遗传算法能够有效降低系统中叫车请求的平均处理时间和叫车请求的最大等待时间。

## 4.7 本章小结

本章在第三章的基础上以缩短处理时间为优化目标，调整了遗传算法的适应度函数，重新选择了贪婪算法的贪婪条件，调整了聚类贪婪遗传算法的无人车分配策略，并且进行了仿真实验与分析，面向最优处理时间的贪婪算法在平均处理时间，叫车请求数量上比遗传算法有优势，而遗传算法则在平均等待时间，最大等待时间方面比贪婪算法有优势。聚类贪婪遗传算法虽然在系统刚启动时在多项优化目标不及前二者，但是当系统趋于稳定时在四项优化目标中均具有明显优势。最后本章针对不同优化目标下聚类贪婪遗传算法的实验结果做对比分析，得出面向最优等待时间的聚类贪婪遗传算法能够有效降低系统中叫车请求的平均等待时间，而面向最优处理时间的聚类贪婪遗传算法能够有效降低系统中叫车请求的平均处理时间和叫车请求的最大等待时间的结论。

# 第五章 基于 JavaEE 的车辆呼叫系统研究

## 5.1 需求分析

2015 年万科与上海交通大学合作开发了被称为“大白”的无人车（图 5 - 1），“大白”装有磁通传感器，它可以获取道路上的磁场信号，从而实现车辆的控制及导航。目前在上海、东莞等地的试用园区配有全自动交通系统，其中“大白”担任全自动交通系统中无人车的角色，本章将基于 JavaEE 对全自动交通系统中的车辆呼叫系统进行研究。首先针对全自动交通系统中各角色进行需求分析，从不同的角色角度出发提出需求；然后对提出车辆呼叫系统的技术架构图；接着针对技术架构中重要的模块做相应研究；最后根据研究成果开发一套功能完善的车辆呼叫系统，并将该系统用于上海、东莞等地的全自动交通系统中。


图 5 - 1 万科无人车”大白”
Fig.5-1 Vanke’s driverless vehicle naming Baymax

全自动交通系统中的车辆呼叫系统主要有无人车、用户、管理员等角色，其中管理员按照权限不同，又分为超级管理员、总管理员、园区管理员等三个角色。按照角色功能得出的系统需求分析如表 5 - 1 所示。

表 5 - 1 车辆呼叫系统需求分析
Table.5-1 Requirement analysis for vehicle calling system.

| 属性 / 角色 | 功能点 | 功能点描述 |
|---|---|---|
| 无人车 | 登录 | 无人车启动时需要向呼叫系统登录，登录后呼叫系统方能控制每辆无人车。 |
| | 汇报地理位置 | 无人车按照一定频率向系统汇报自己 |

| | | |
|---|---|---|
| | | 当前所在位置。 |
| | 到达站点汇报 | 当无人车新到达一个站点，需要向系统汇报到达站点的事件。 |
| | 离开站点汇报 | 当无人车离开一个站点，需要向系统汇报离开站点的事件。 |
| | 日志 | 日志分为三个普通、警告和错误三个级别，无人车按照车况给系统汇报日志。 |
| | 退出 | 无人车停止运行时，退出登录，此后该无人车不再接受系统调度。 |
| 用户 | 关注微信公众号 | 使用微信公众号作为用户呼叫无人车的客户端，通过扫描或搜索的方式添加。 |
| | 登录 | 关注公众号后系统自动拉取用户基本资料，使用基本资料为用户登录，用户无需再输入用户名和密码。 |
| | 修改地理位置 | 系统自动根据微信定位功能获取用户地理位置，同时用户也可以手动设置。 |
| | 呼叫无人车 | 用户可以选择乘车的起始站点和目的站点，提交后系统生成一条叫车请求。 |
| | 叫车信息查询 | 用户可以查询到自己当前的叫车情况以及叫车请求的基本状态。 |
| | 取消叫车请求 | 用户可以取消当前的叫车请求，系统便不再对该叫车请求继续响应。 |
| | 查看无人车地理位置 | 在微信对话聊天窗口点击“无人车当前位置”，系统回复无人车所在站点或者位置。 |
| | 查看地图中无人车动态 | 显示一张园区中的地图，无人车的位置在地图中标示出，并且动态呈现。 |
| 管理员 | 登录 | 通过用户名，密码登录到呼叫系统后台 |
| | 忘记密码 | 通过邮箱找回密码 |
| | 纵览概况 | 系统中无人车、站点、用户、叫车历史纪录等信息统计 |
| | 切换园区 | 切换当前管理的园区，园区管理员无此权限 |
| | 无人车实况 | 在地图中查看无人车的实时动态 |
| | 无人车视角锁定 | 锁定视角中心点为无人车，方便查看无人车的实时运行位置 |
| | 站点定位 | 将视角平滑定位到某个站点 |
| | 园区运行状态监控 | 监控某个园区中所有的叫车请求，所有无人车运行状态 |
| | 用户管理 | 查看、修改、删除园区中的用户资料 |

| | | |
|---|---|---|
| | 叫车请求记录查询 | 根据用户或无人车或园区查询历史叫车请求 |
| | 车辆日志管理 | 管理车辆在运行中产生的日志 |
| | 园区管理 | 对系统中的园区可以做增删改查操作 |
| | 管理员管理 | 超级管理员可以管理其它管理员，包括查看、添加、删除和修改其他管理员 |
| | 系统角色管理 | 超级管理员可以提升或者降低某种管理员角色的权限 |
| | 权限结点管理 | 对系统中所有的接口做录入，方便权限控制 |
| | 微信菜单管理 | 配置微信公众号中的菜单 |
| | 百度地图管理 | 配置百度地图 API 的相关参数 |

## 5.2 技术架构

根据全自动交通系统中的车辆呼叫系统的需求分析，本文采用面向企业级，能够满足高并发，高负载需求的 JavaEE 作为技术，整个系统的主要技术架构如图 5 - 2 所示。

无人车和中央控制系统采用 GPRS 方式[47]通信，由于无人车和中央控制系统之间的通信具有高实时性，高安全性要求等特点，因此常规的 Http 协议[48]将不能满足需求，本文采用 TCP 协议[49]，实现了无人车和中央控制系统的实时通信，同时消息采用双向加密，保证了消息的安全性。

用户叫车需要借助于叫车客户端，在本系统中用户使用微信作为叫车客户端。在客户端用户可以生成一条叫车指令发送给中央控制系统，然后中央控制系统经过匹配计算，安排无人车前来接送用户，同时用户在客户端中可以查看无人车当前的实时状况以及在地图中看到车辆行驶的路线。

中央控制系统是整个车辆呼叫系统的核心，所有的信息以及任务指令都是在这里汇总和分发，它起到中间协调用户和无人车的作用，同时中央控制系统还负责统计和分析历史数据，并且根据历史数据调整车辆路径规划的算法，以使全自动交通系统达到最优化。下文将着重对中央控制系统各个模块进行研究，以使得整车辆呼叫系统性能达到较优状态。

中央控制系统主要由消息模块、Web 模块和算法模块等 9 个模块组成，本文就每个模块的核心技术做相应的研究。

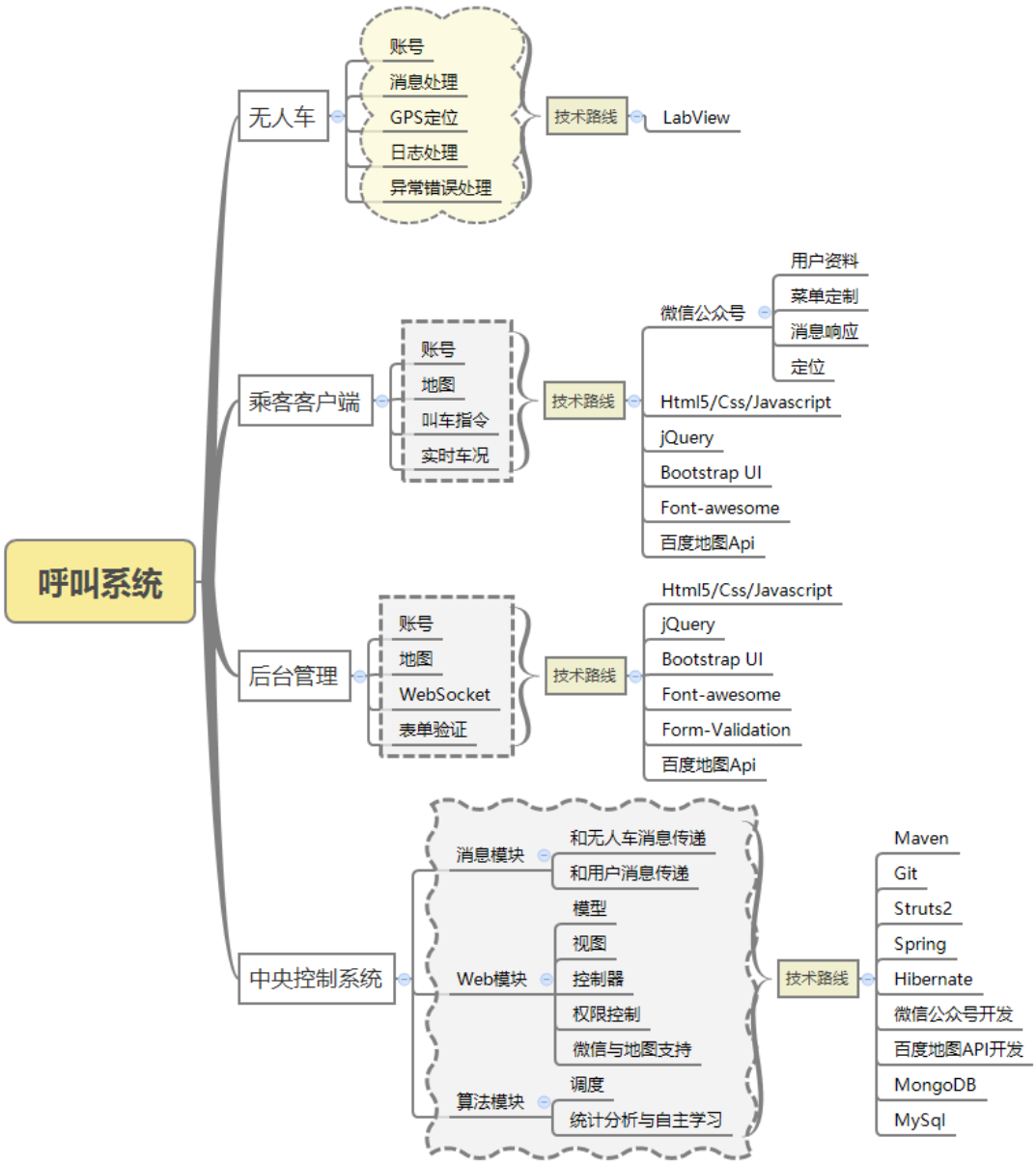



图 5 - 2 呼叫系统技术架构图

Fig.5-2 Technology structure of calling system

## 5.3 消息模块

消息模块是车辆调度系统中非常关键的一个模块，无人车与中央控制系统，中央控制系统与用户之间的信息传递均依赖于消息模块。本文所研究的车辆呼叫系统需要处理上百数量级的无人车以及上千数量级的用户，同时系统要能承受高并发，高负载的支持，因此系统中的消息模块就必须满足以下特点[50]：

(1) 消息模块易于向外扩展，易于各个客户端的接入。

(2) 消息模块具备给订阅消息和发布消息提供高吞吐量的能力

(3) 消息模块支持多个订阅者的模式，当任务失败时能够自动平衡消费者

(4) 消息模块能够将消息持久化到磁盘上，从而可以用于批量消费。

消息模块包括消息，话题，生产者等组件，其中消息是字节的有效负载，话题是特定类型的消息流，它也是消息的种子名或分类名，生产者是具有发布消息到话题能力的对象。在全自动交通系统中，叫车请求便是生产者；车辆呼叫，日志记录是不同的话题；叫车请求，无人车行驶路线指令等是消息。

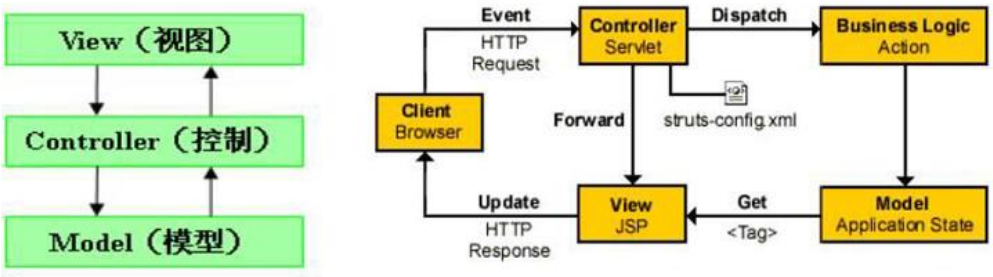
在消息模块中，采用生产者，订阅者，消费者，中间代理的方式进行。如无人车对于车辆呼叫的消息感兴趣，于是向中间代理订阅车辆呼叫这一话题，当用户发出一条叫车请求时，系统便产生一条话题为车辆呼叫的消息，产生之后中间代理便将消息发送给订阅者，也就是无人车，当无人车收到之后便向系统确认已经收到，消息便设置为已经处理，至此该消息消费完毕。倘若系统产生一条话题为站点统计的消息，系统便不把该消息发送给无人车了，因为无人车并未订阅站点统计这一话题。

本文采用这种消息模块策略保证了车辆呼叫系统中消息的不重不漏，从而使得信息传递准确而高效。

## 5.4 Web 模块

本文所研究的车辆呼叫系统是面向企业级进行设计的，因此对其处理高并发，高负载能力有较高的要求。而 Web 模块是整个系统中的核心，因此本文采用面向企业级的 Struts[52]作为 Web 技术框架，Struts 采用模型-视图-控制器（Model-View-Controller，MVC）模式，它是面向对象设计，能利用 MVC 模式分离显示逻辑和业务逻辑。

Struts 框架的核心是一个弹性的控制层，基于如 Java Servlets，JavaBeans，ResourceBundles 与 XML 等标准技术，以及 Jakarta Commons 的一些类库。Struts 由一组相互协作的组件、Serlvet 以及 jsp tag lib 组成。图 5 - 3a)是 MVC 的结构，Controller 是连接 View 和 Model 的枢纽。图 5 - 3 b)展示的是 Struts 的体系结构和工作原理。




a)Model-Controller-View结构　　b)Struts的体系结构与工作原理

图 5 - 3 Struts 中的 MVC 及结构原理
Fig.5-3 Principle of MVC in Struts

同时在本文系统中还包含了管理员操作的权限控制，以及中央控制系统对于微信基本服务所做的支持和对百度地图所做的支持。

### 5.4.1 模型(Model)

模型是本文系统中用于处理应用程序数据逻辑的部分，它响应控制器获取数据的请求，同时也接受控制器的更新请求，模型同时要负责数据的持久化，本文系统中是将数据存储在数据库，数据持久化采用 Hibernate[53]。Hibernate 是对象关系映射框架，它对 JDBC 进行了轻量级的对象封装，使得 Java 程序员可以随心所欲地使用对象编程思维来操纵数据库。Hibernate 可以应用在任何使用 JDBC 的场合，既可以在 Java 的客户端程序使用，也可以在 Servlet/JSP 的 Web 应用中使用，在本文中央控制系统中主要使用在 DAO 中。

### 5.4.2 视图(View)

视图是应用程序中处理数据显示的部分，它主要展示控制器提供的数据，并且按照一定规范呈现出来。本文中央控制系统的视图主要使用 JSP(Java Server Pages)，JSP 是在传统的 HTML(Hyper Text Markup Language)页面中添加了特殊的标签，使得其具有解析 Java 程序段的功能，在 JSP 页面中通过引用 Java 的 jar 包后便可以在符号<%%>中编写 Java 代码。虽然有直接编写 Java 的功能，但是本文不倡导如此使用，因为本文的车辆呼叫系统是严格按照 MVC 的模式设计的，在 JSP 页面中编写 Java 代码会使得 MVC 逻辑混乱，视图层和业务逻辑层耦合过强的。通常采用 JSP 内置的标签或者自定义的标签来完成和控制器的交互。

在车辆呼叫系统的后台系统页面均使用 JSP 页面呈现，一般页面中包含了引

用的 CSS（Cascading Style Sheets），JS(Javascript)，图标，字体等文件，通过 CSS 样式的控制使得页面呈现出美观的界面，通过 JS 使得页面可以进行动态交互。

### 5.4.3 控制器(Controller)

控制器是 MVC 体系中的核心，它负责处理浏览器传送过来的所有请求，根据请求参数从模型获取数据并通过视图返回给浏览器，因此控制器是连接模型和视图的中枢。

在 Struts 中，控制器的设计主要是通过配置文件依靠 Java 的反射技术来完成的。对于一个浏览器发过来的请求，步骤如图 5 - 4 所示：

可以看出控制器在整个流程中起到承上启下的中枢作用，图 5 - 4 中的 Action Invocation 就主要是控制器的作用范围。

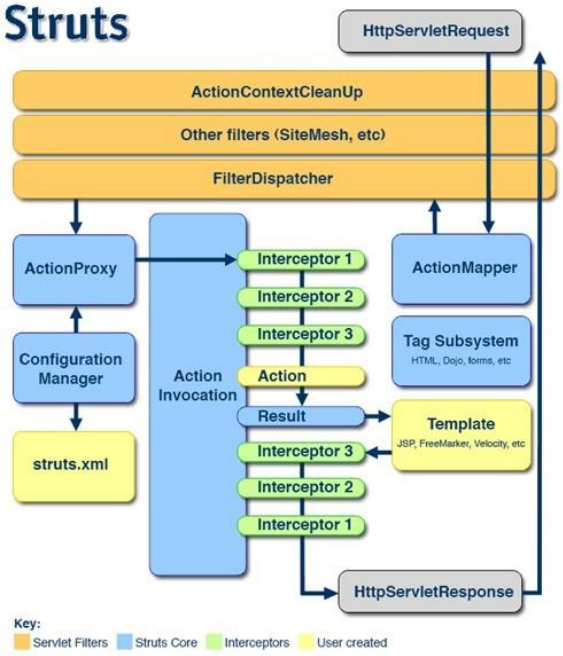



图 5 - 4 Struts 请求原理图

Fig.5-4 Schematic diagram of struts

### 5.4.4 权限控制

在一个系统中，权限控制是一个非常重要的模块，本文采用基于角色的访问控制(Role-Based Access Control，RBAC)[35]， RBAC 权限模型如图 5 - 5 所示。

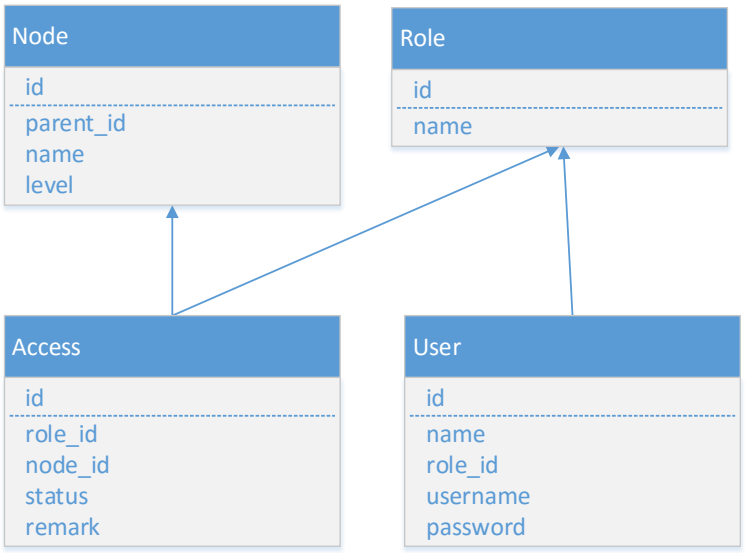



图 5 - 5 RBAC 权限模型
Fig.5-5 Permission model of RBAC

在车辆呼叫系统中使用四张数据库表共同完成权限的控制，Node 记录的是系统中的权限结点，如添加园区，删除园区，编辑园区，查看无人车实况等。User 记录的是系统中的用户，Role 记录的是系统中的权限。在系统中每个 User 均拥有一个 Role，系统中共有游客，普通用户，园区管理员，总管理员以及程序员等 5 中角色。Access 中记录了某一个 Role 对应的 Node，即反映了某个角色对于某一个权限结点的操作权，比如在 Access 中有园区管理员对应编辑园区的记录，而没有普通用户对应删除园区的记录，就代表园区管理员有权编辑园区，普通用户无权删除园区。通过这种方式，使得系统中任何一个操作都能够达到非常精细粒度的控制，从而保证了系统的安全性。

### 5.4.5 微信公众平台

腾讯公司在 2011 年推出了一款主要面向移动端的即时通讯软件——微信，它可以通过网络发送语音、文字、图片和视频，同时支持语音聊天和视频聊天。微信公众平台是腾讯公司在微信的基础上，推出的一点对多点的信息推送平台。随着微信覆盖率越来越高，本系统采用微信客户端作为用户主要的叫车终端。

在本系统中主要用到微信开发者接口有如下：权限请求基本支持，接受消息，发送消息，用户管理，自定义菜单，素材管理，多客户功能服务，网页账号，网页图片及音视频服务，地理位置等功能。中央控制系统对每个功能模块均做了相应

的接口支持，充分地利用微信了提供的丰富 API，实现了用户和中央控制系统的强交互，进而使整个系统达到了易用性和高效性。

### 5.4.6 百度地图 API

百度地图 API（Application Programming Interface,应用程序编程接口）是由百度退出的一套由 JavaScript 编写的应用程序接口。百度将复杂的地理信息数据进行了高度的封装，使得开发者只需要懂的基本的网站前端开发知识即可编写出界面美观，交互能力强的地图应用。

在本系统中主要用到百度地图 API 提供的地图基本功能（2D，3D，卫星视图，平移，缩放，拖拽），地图控件展示功能，覆盖物功能，工具类功能，定位功能，右键菜单功能，鼠标交互功能，图层功能，本地搜索功能等，系统中不仅在微信客户端用到百度地图以展示无人车当前位置及自己当前位置，还在后台管理页面中用来设置站点地理位置，设置无人车地理位置，实时查看各个无人车的运行情况。

## 5.5 程序实现

结合上文对于算法和系统的研究，本文开发了一套完整的用于全自动交通系统的车辆呼叫系统，在本系统中无人车被命名为 CyberOne。

在环境平台方面，该车辆呼叫系统采用 JavaEE 作为核心开发技术，使用 java8 作为开发语言；使用 IntelliJ14.0 作为集成开发环境，并且采用 Markdown，Lombok, Shell, File Watcher 等多个插件共同工作；使用 Git 作为版本控制软件，并且在服务器上自建了 Gitlab 用于在线代码托管，方便在实验室与寝室中协同开发；使用 Maven 作为项目管理工具，便于控制项目的第三方库依赖，管理系统的运行以及部署；使用 Mysql 作为逻辑数据存储的数据库，利用 Mysql 关系型的特点，系统中带有业务逻辑的数据均使用 Mysql 存储；使用 MongDB 作为图片、文档等二进制资源存储的数据库，利用 MongDB 优秀的文件管理功能，系统将所有的二进制资源交由 MongDB 统一管理；使用 Tomcat 作为服务器容器，利于系统的快速部署高效运行。

在系统架构方面，该车辆呼叫系统服务器后台采用 Struts, Spring, Hibernate 技术体系，从而保证了整个项目具有层次结构清晰，异常处理机制完善，业务逻辑与系统无侵入，组件之间松耦合，可移植性好，可定制性强等优点；客户端前端采用 Html, Css, Javascript, jQuery, Bootstrap, font-awesome, 微信公众号，百度

地图 API 等基于浏览器的技术体系，从而使得车辆呼叫系统访问更加友好，使用更加快捷。

在业务逻辑方面，该车辆呼叫系统使用上文研究的分布式消息系统，用以保证中央控制系统和无人车以及和用户之间消息准确的传递；使用 RBAC 的权限控制方式，保证了整个系统中的权限细粒度的控制，方便在整个系统最高级别层面上的管理；

在用户界面方面，该车辆呼叫系统采用用户友好型的 Bootstrap 作为主要 UI 框架，界面设计以简单清新实用为主。如图 5-6 展示的是无人车实况页面，在该页面中管理员可以实时监控无人车在全自动交通系统中的运行状况。如图 5-7 展示的是微信叫车交互页面，用户在图示的三个页面中分别可以完成呼叫无人车，查看当前叫车状态，查看无人车实时动态等功能。

该车辆呼叫系统从最初提出需求，到研究各个技术要点，再到后来开发，测试，再到最后上线运行历时 2 年时间，整个项目共涉及三门编程语言，横跨了两大数据库，包含了 61 个第三方库，6 万行独立编写的代码，接入了两个第三方平台服务。以上所有工作均由笔者一人完成，截止到目前，该车辆呼叫系统已经成功在东莞、上海等地的全自动交通系统中运行了。

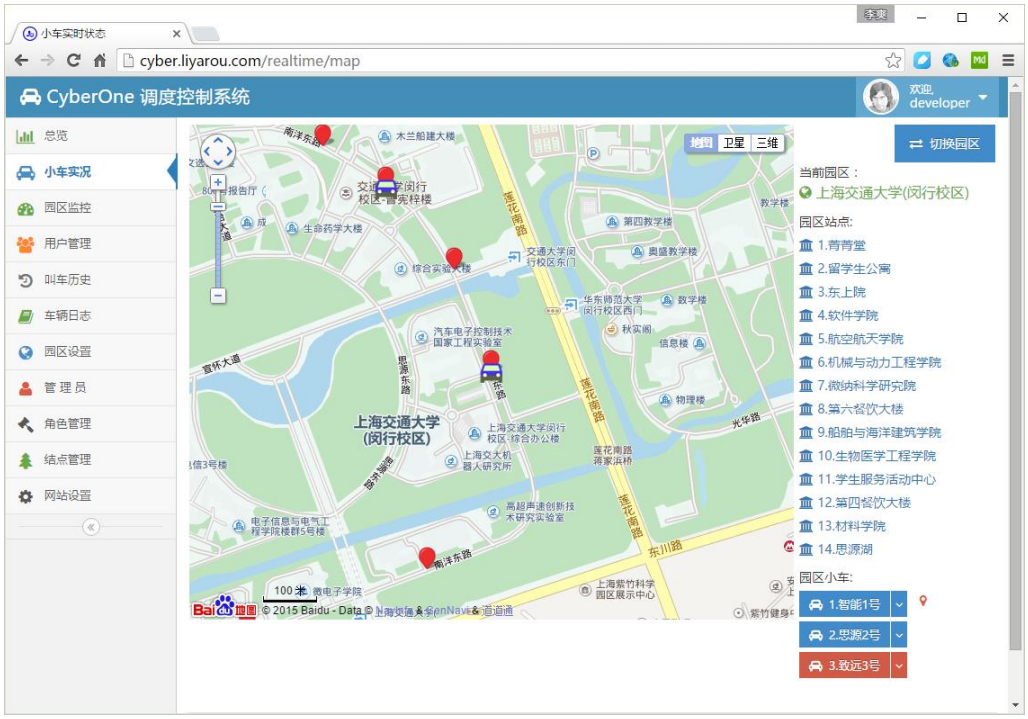



图 5-6 无人车实况页面

Fig.5-6 Page of driverless vehicle's realtime


a）我要叫车页面　　b) 叫车信息详情　　c) 小车实时位置

图 5 - 7 微信叫车交互页面

Fig.5-7 Interactive page of calling in Wechat

## 5.6 本章小结

本章车辆呼叫系统以全自动交通系统为基础，通过分析系统中的多个角色及其在全自动交通系统中的行为得出了详细的功能需求，然后通过需求分析得出车辆呼叫系统的技术架构图。接着本章对框架中的消息模块进行了研究，该系统使用基于订阅者，发布者，消费者模式的消息架构。其次本文对系统中 Web 模块进行了相应研究，使用 MVC 设计可以使 Web 模块具有高解耦性和高鲁棒性的特点；采用 RBAC 的权限设计模式，可以将权限控制精细到每个方法和每个资源的粒度，采用微信公众平台和百度地图 API 可以更方便的和用户进行交互，增强用户体验。最后，本文在已有的研究成果基础上，设计并开发了一套用于全自动交通系统的车辆呼叫系统，该呼叫系统采用了 JavaEE 的技术架构，使用了微信公众平台和百度地图 API，能够满足高负载高并发等技术要求，具有界面美观，使用简单等特点，目前该车辆呼叫系统已成功在东莞、上海等地的全自动交通系统中运行。

# 第六章 总结与展望

## 6.1 本文总结

全自动交通系统是汽车信息化与智能化的一种高级表现形式，全自动交通系统可以有效地解决汽车带来的安全、交通、能源和环境等方面的问题，并且随着无人车的飞速发展，越来越多的全自动交通系统将投入到实际生产使用中来，甚至全自动交通系统在将来有可能取代公交系统，因此全自动交通系统具有十分重要的研究价值。结合全自动交通系统的特点及其车辆调度的关键技术，本文展开了对全自动交通系统车辆调度算法和人机交互系统的研究。

本文首先用具体的数学问题描述定义了全自动交通系统中的“最优化”，并且提出了相应的最优化目标；对全自动交通系统中的无人车站点、用户叫车请求以及无人车分别做了相应的数学实体建模，得出了一组方便用于数学描述和算法描述的符号；对优化目标进行高度抽象，得到一组优化目标函数。

其次本文从解决传统车辆调度问题的思路出发，对遗传算法的产生和发展，基本思路以及其特点做了相应的概述；本文面向最优等待时间和面向最优处理时间，分别研究了遗传算法求解全自动交通系统中车辆调度问题的具体方法和步骤，研究了贪婪算法求解全自动交通系统中车辆调度问题的具体方法和步骤，并给出了相应的流程图和伪代码。针对遗传算法和贪婪算法的缺点，本文尝试并保留二者优点，提出了聚类贪婪遗传算法，在求解高频呼叫站点聚类群时提出基于高频呼叫站点关联度的聚类算法，进而得出聚类贪婪遗传算法的流程图。本文根据全自动交通系统的实际情况，分析了具体需求，设计并开发了一个全自动交通系统的仿真系统。在该仿真系统中验证了面向最优等待时间和面向最优处理时间的聚类贪婪遗传算法的高效性和稳定性。

最后本文通过分析系统中的多个角色及其在全自动交通系统中的行为，得出了详细的功能需求，然后通过需求分析得出车辆呼叫系统的技术架构图接着本章对框架中的消息模块进行了研究，该系统使用基于订阅者，发布者，消费者模式的消息架构。本文对系统中 Web 模块进行了相应研究，使用 MVC 设计可以使 Web 模块具有高解耦性和高鲁棒性的特点；采用 RBAC 的权限设计模式，可以将权限控制精细到每个方法和每个资源的粒度，采用微信公众平台和百度地图 API 可以更方便的和用户进行交互，增强用户体验。该车辆呼叫系统从最初提出需求，到研究各个技术要点，再到后来开发，测试，再到最后上线运行历时 2 年时间，整

个项目共涉及三门编程语言，横跨了两大数据库，包含了 61 个第三方库，6 万行独立编写的代码，接入了两个第三方平台服务。以上所有工作均由笔者一人完成，截止到目前，该车辆呼叫系统已经成功在东莞、上海等地的全自动交通系统中运行了。

本文主要针对全自动交通系统的车辆调度算法进行了研究，其中提出的某些方法和解决问题的思路具有一定的可参考意义，具体的贡献表现在以下几个方面：

(1) 针对全自动交通系统中各个站点的流量不均匀，分析了出现站点不同流量的特点和实质，提出了一种用户划分高频呼叫站点和低频站点的方法，准确而高效地将全自动交通系统中的站点进行了分类。

(2) 针对全自动交通系统中车辆调度问题，分析了遗传算法和贪婪算法的特点与实质，结合遗传算法具有叫车请求平均等待时间较优而贪婪算法具有叫车请求平均处理时间较优的特点，提出了用于解决全自动交通系统中车辆调度问题的聚类贪婪遗传算法，并且通过仿真实验验证了该算法的高效性和稳定性。

(3) 针对高频呼叫站点的聚类，分析了现有聚类方法如 K-Means 算法的特点和实质，剖析了其用于全自动交通系统中高频呼叫站点聚类的缺陷，提出了一种基于高频呼叫站点关联度的聚类算法，该算法既保证了高频呼叫站点的聚集性，同时又保证了高频呼叫站点之间的关联性。

## 6.2 工作展望

本文对全自动交通系统车辆调度算法进行了广泛地调研和深入的研究，以及根据研究成果成功地开发了一套适用于全自动交通系统的车辆呼叫系统。虽然文中提出的基于高频呼叫站点关联度的聚类算法能够寻找到关联度较高的高频呼叫站点聚类，采用本文提出的聚类贪婪遗传算法进行车辆调度能够保证全自动交通系统中各优化目标在稳定状态最优，但从一定程度而言，本文中仍存在较多的问题，需要进一步深入和探讨。比如在高频呼叫站点聚类时没有考虑某个聚类中只包含较少高频呼叫站点的情况，而是将其和其他聚类统一对待，在采用基于高频呼叫站点关联度的聚类算法时有必要对特殊的聚类进行排除或者归并研究。再比如本文仅针对四种优化目标中的两个优化目标作为最优化条件进行了研究，对于另外两个优化目标作为最优化的条件的算法研究本文尚未涉及，有必要在此方向做进一步研究。

上述存在的问题，均需要在未来的工作中进一步展开和深入。车辆调度算法

作为全自动交通系统中的重要组成部分，随着无人车技术的不断发展，随着全自动交通系统的不断普及，作为全自动交通系统中的重要组成部分，车辆调度算法的研究必然会迎来新的机遇和挑战。

# 参 考 文 献

[1] 国家统计局. 中国汽车市场年鉴.2013.

[2] 中国自动化学会. 汽车智能化进程及其关键技术 http://mp.weixin.qq.com/s?__biz=MzA3MDU5ODMzMg==&mid=402588460&idx=1&sn=34637f0b1b1f64e22d9a7324c91f9856&scene=1&srcid=1205KmZLQhyd6LHhUIs9Xnlf#wechat_redirect

[3] 中国交通事故赔偿网. 2014 年全国道路交通事故数据统计[EB/OL]. http://www.peichang.cn/detail/id2737.html,2015-3-13.

[4] 工信部详解新能源汽车和智能汽车 2025 发展目标. 节能与新能源汽车网. http://www.chinaev.org/DisplayView/Normal/News/Detail.aspx?id=20710

[5] Luis R, Rastelli J P, Bautista D G, et al. Description and Technical specifications of Cybernetic Transportation Systems: an urban transportation concept[C]//2015 IEEE International Conference on Vehicular Electronics and Safety. 2015.

[6] Flores C, Milanés V, Pérez J, et al. An Energy-Saving Speed Profile Algorithm for Cybernetic Transport Systems[C]//ICVES 2015-IEEE international conference on vehicular electronics and safety. 2015.

[7] Dantzig G, Ramer J. The trunk dispatching probem[J]. Management Science, 1959, 10(6): 80-91.

[8] Tarantilis C D, Ioannou G, Kiranoudis C T, et al. Solving the open vehicle routing problem via a single parameter metaheuristic algorithm[J]. Journal of the Operational Research Society, 2005, 56(5): 588-596.

[9] Christofides N, Mingozzi A, Toth P. State Space Relaxation Procedures for the Computation of Bounds to Routing Pmblems [J]. Networks, 1981, 11 (2):145-164.

[10] Padlberg M W, Rinaldi G A Branch-and-Cut Algorithm for the Resolution of Large—Scale Symmetric Traveling Salcman Problem [J]. SLAM Review, 1991, 33 (1):60-100.

[11] M. Fisher. Vehicle Routing In: M. O. Ball, T. L. Magnanti, Handbooks in Operations Research and Management Science, Volume 8: Network RoutingAmsterdam：Elsevier Science, 1995,1,13.

[12] Glover F. Tabu search-part I[J]. ORSA Journal on computing, 1989, 1(3): 190-206.

[13] Petersen H L, Larsen A, Madsen O B G, et al. The simultaneous vehicle scheduling and passenger service problem[J]. Transportation Science, 2012, 47(4): 603-616.

[14] Dowsland K A, Thompson J M. Simulated annealing[M]//Handbook of Natural

Computing. Springer Berlin Heidelberg, 2012: 1623-1655.
[15] Banos R, Ortega J, Gil C, et al. A simulated annealing-based parallel multi-objective approach to vehicle routing problems with time windows[J]. Expert Systems with Applications, 2013, 40(5): 1696-1707.
[16] Mousavi S M, Tavakkoli-Moghaddam R. A hybrid simulated annealing algorithm for location and routing scheduling problems with cross-docking in the supply chain[J]. Journal of Manufacturing Systems, 2013, 32(2): 335-347.
[17] John Henry Holland. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence[M]. MIT press, 1992.
[18] Berger J，Barkaoui M．A new hybrid genetic algorithm for the capacitated vehicle routing problem，Journal ofthe Operational Research Society,2003,54:1254-1263.
[19] Ombuki B, Ross B J, Hanshar F. Multi-objective genetic algorithms for vehicle routing problem with time windows[J]. Applied Intelligence, 2006, 24(1): 17-30.
[20] Dorigo M, Maniezzo V, Colorni A. The ant system: An autocalytic optimizing process[R]. Technical Report, 1991.
[21] Bell J E, McMullen P R. Ant colony optimization techniques for the vehicle routing problem[J]. Advanced Engineering Informatics, 2004, 18(1): 41-48.
[22] 钟石泉, 贺国光. 多车场有时间窗的多车型车辆调度及其禁忌算法研究[J]. 运筹学学报, 2005(4):67-73.
[23] 郎茂祥. 装卸混合车辆路径问题的模拟退火算法研究[J]. 系统工程学报, 2005, 20(5):485-491.
[24] 杨善林, 马华伟, 顾铁军. 时变条件下带时间窗车辆调度问题的模拟退火算法[J]. 运筹学学报, 2010, 14(03):83-90.
[25] 卫田. 物流配送中车辆路径问题的多目标优化算法研究[D]. 清华大学, 2007.
[26] 李高扬. 蚁群算法在车辆调度问题中的应用研究[D]. 东北大学, 2008.
[27] 吴珂. 基于蚁群算法的单配送中心车辆调度问题研究[D]. 大连海事大学, 2013.
[28] 黄兆年, 李海山, 赵君. 多目标蚁群算法城市车辆调度站部署的研究[J]. 舰船电子工程, 2015(8):41-44.
[29] 柳武生,谭倩. 基于混合算法的实时订货信息下车辆调度优化[J]. 应用数学与计算数学学报，2012,26(1):53-65.
[30] 周兴田. 基于模拟退火遗传算法的车辆调度问题研究[D]. 辽宁:大连海事大学硕士学位文,2007.
[31] 郎茂祥，胡思继. 用混合遗传算法求解物流配送路径优化问题的研究[J]. 中国

科学管理,2002，1010(15):51-56.

[32]邢鹏．基于云平台的多配送中心车辆调度问题研究[D]．北京：北京交通大学硕士学位论文, 2013.

[33]Parmar M P, Pandi M G S. Performance Analysis and Augmentation of K-means Clustering, based approach for Human Detection in Videos[C]//International Journal of Engineering Development and Research. IJEDR, 2015, 3(2 (May 2015)).

[34]Heffelfinger D R. Java EE 7 Development with NetBeans 8[M]. Packt Publishing Ltd, 2015.

[35]Hamann L, Sohr K, Gogolla M. Monitoring Database Access Constraints with an RBAC Metamodel: A Feasibility Study[M]//Engineering Secure Software and Systems. Springer International Publishing, 2015: 211-226.

[36]Allan Heydon, Marc Najork. Mercator: A scalable, extensible Web crawler, 2(1999) 219-229.

[37]GIS 理论 http://blog.sina.com.cn/s/blog_4e45b8430100j65l.html 2015

[38]Lauri F, Créput J C, Koukam A. The Multi-agent Patrolling Problem Theoretical Results about Cyclic Strategies[M]//Advances in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection. Springer International Publishing, 2014: 171-182.

[39]Fabio Pasqualetti, Antonio Franchi, and Francesco Bullo. On optimal cooperative patrolling. In 49th IEEE Conference on Decision and Control, pages 7153–7158, Atlanta, GA, USA, 12/2010 2010.

[40]C. Poulet, V. Corruble, and A.E.F. Seghrouchni. Working as a team: Using social criteria in the timed patrolling problem. In Tools with Artificial Intelligence (ICTAI), 2012 IEEE 24th International Conference on, volume 1, pages 933–938, Nov 2012.

[41]Bagley J D. The behavior of adaptive systems which employ genetic and correlation algorithms[J]. 1967.

[42]Goldberg D E, Holland J H. Genetic algorithms and machine learning[J]. Machine learning, 1988, 3(2): 95-99.

[43]Skinner M K. Environmental Epigenetics and a Unified Theory of the Molecular Aspects of Evolution: A Neo-Lamarckian Concept that Facilitates Neo-Darwinian Evolution[J]. Genome biology and evolution, 2015, 7(5): 1296-1302.

[44]Schwarzbach E, Smýkal P, Dostál O, et al. Gregor J. Mendel–genetics founding father[J]. Czech J. Genet. Plant Breed, 2014, 50(2): 43-51.

[45]Fernandez-Viagas V, Framinan J M. A bounded-search iterated greedy algorithm for the distributed permutation flowshop scheduling problem[J]. International Journal of Production Research, 2015, 53(4): 1111-1123.

[46] Chang F S, Wu J S, Lee C N, et al. Greedy-search-based multi-objective genetic algorithm for emergency logistics scheduling[J]. Expert Systems with Applications, 2014, 41(6): 2947-2956.

[47] GSM, GPRS and EDGE performance: evolution towards 3G/UMTS[M]. John Wiley & Sons, 2004.

[48] Fielding R, Reschke J. Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing[J]. 2014.

[49] Forouzan B A. TCP/IP protocol suite[M]. McGraw-Hill, Inc., 2002.

[50] Apache Kafka：下一代分布式消息系统 http://chuansong.me/n/479250 2015.12.10

[51] Sendfile API http://man7.org/linux/man-pages/man2/sendfile.2.html 2015.12.10

[52] Brown D, Davis C M, Stanlick S. Struts 2 in action[M]. Dreamtech Press, 2008.

[53] Bauer C, King G. Java Persistance with Hibernate[M]. Dreamtech Press, 2006.

# 附录 1. 面向最优等待时间算法实验数据

表 附 1 - 1 面向最优等待时间的平均等待时间

Table.Appendix 1-1 Waiting time based on the optimization of waiting time

| 时刻 min | 遗传算法 min | 贪婪算法 min | 聚类贪婪遗传算法 min |
|---|---|---|---|
| **0** | 0 | 0 | 0 |
| **10** | 1.166666667 | 2.783333333 | 1.783333333 |
| **20** | 1.633333333 | 3.366666667 | 2.233333333 |
| **30** | 2.05 | 3.633333333 | 2.433333333 |
| **40** | 2.716666667 | 3.833333333 | 2.8 |
| **50** | 3.183333333 | 4.233333333 | 2.983333333 |
| **60** | 3.816666667 | 4.716666667 | 3.166666667 |
| **70** | 4.083333333 | 5.533333333 | 3.35 |
| **80** | 4.3 | 6.55 | 3.516666667 |
| **90** | 4.8 | 7.266666667 | 3.616666667 |
| **100** | 5.116666667 | 7.783333333 | 3.583333333 |
| **110** | 5.1 | 8.116666667 | 3.633333333 |
| **120** | 5.3 | 8.683333333 | 3.733333333 |
| **130** | 5.5 | 8.833333333 | 3.766666667 |
| **140** | 5.666666667 | 9.15 | 3.85 |
| **150** | 5.95 | 9.383333333 | 3.85 |
| **160** | 6.133333333 | 9.5 | 3.933333333 |
| **170** | 6.3 | 9.516666667 | 3.95 |
| **180** | 6.45 | 9.483333333 | 3.966666667 |
| **190** | 6.566666667 | 9.633333333 | 3.95 |
| **200** | 6.65 | 9.6 | 3.95 |

表 附 1 - 2 面向最优等待时间的平均处理时间

Table.Appendix 1-2 Handling time based on the optimization of waiting time

| 时刻 min | 遗传算法 min | 贪婪算法 min | 聚类贪婪遗传算法 min |
|---|---|---|---|
| **0** | 0 | 0 | 0 |
| **10** | 3.316667 | 2.233333 | 2.883333 |
| **20** | 6.933333 | 4.416667 | 4.983333 |
| **30** | 9.616667 | 6.25 | 6.083333 |
| **40** | 10.35 | 7.583333 | 6.883333 |
| **50** | 10.96667 | 8.55 | 7.166667 |
| **60** | 11.35 | 9.616667 | 7.366667 |

| | | | |
|---|---|---|---|
| **70** | 11.6 | 10.23333 | 7.483333 |
| **80** | 11.55 | 10.41667 | 7.6 |
| **90** | 11.71667 | 10.5 | 7.633333 |
| **100** | 11.7 | 10.68333 | 7.6 |
| **110** | 11.71667 | 10.88333 | 7.683333 |
| **120** | 11.71667 | 10.95 | 7.766667 |
| **130** | 11.68333 | 11.06667 | 7.783333 |
| **140** | 11.71667 | 10.96667 | 7.8 |
| **150** | 11.76667 | 10.91667 | 7.85 |
| **160** | 11.75 | 10.81667 | 7.866667 |
| **170** | 11.73333 | 10.78333 | 7.95 |
| **180** | 11.7 | 10.88333 | 7.95 |
| **190** | 11.73333 | 10.85 | 7.933333 |
| **200** | 11.75 | 10.86667 | 7.95 |

表 附 1 - 3 面向最优等待时间的最大等待时间

Table.Appendix 1-3 Maximum waiting time based on the optimization of waiting time

| **时刻 min** | **遗传算法 min** | **贪婪算法 min** | **聚类贪婪遗传算法 min** |
|---|---|---|---|
| **0** | 0 | 0 | 0 |
| **10** | 5 | 8.333333 | 6.666666667 |
| **20** | 10.33333 | 15 | 14 |
| **30** | 16.66667 | 21.66667 | 15 |
| **40** | 18.33333 | 23.33333 | 16 |
| **50** | 22 | 26.66667 | 17.66666667 |
| **60** | 25 | 33.33333 | 21.66666667 |
| **70** | 25 | 35 | 21.66666667 |
| **80** | 25 | 43.33333 | 21.66666667 |
| **90** | 25 | 50 | 21.66666667 |
| **100** | 25 | 50 | 21.66666667 |
| **110** | 25 | 50 | 21.66666667 |
| **120** | 25 | 58.33333 | 21.66666667 |
| **130** | 25 | 61.66667 | 21.66666667 |
| **140** | 30 | 61.66667 | 21.66666667 |
| **150** | 31.66667 | 61.66667 | 21.66666667 |
| **160** | 33.33333 | 61.66667 | 21.66666667 |
| **170** | 35 | 65 | 21.66666667 |
| **180** | 35 | 71.66667 | 21.66666667 |
| **190** | 35 | 75 | 21.66666667 |
| **200** | 35 | 75 | 21.66666667 |

表 附 1 - 4 面向最优等待时间的处理数量

Table.Appendix 1-4 Number of handled requests based on the optimization of waiting time

| 时刻 min | 遗传算法（次） | 贪婪算法（次） | 聚类贪婪遗传算法（次） |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 10 | 23.33333333 | 26.6666667 | 40 |
| 20 | 85 | 130 | 133.3333333 |
| 30 | 185 | 241.666667 | 288.3333333 |
| 40 | 298.3333333 | 333.333333 | 421.6666667 |
| 50 | 403.3333333 | 433.333333 | 550 |
| 60 | 498.3333333 | 523.333333 | 676.6666667 |
| 70 | 610 | 641.666667 | 821.6666667 |
| 80 | 715 | 751.666667 | 936.6666667 |
| 90 | 825 | 863.333333 | 1078.333333 |
| 100 | 936.6666667 | 961.666667 | 1195 |
| 110 | 1043.333333 | 1088.33333 | 1348.333333 |
| 120 | 1155 | 1205 | 1475 |
| 130 | 1248.333333 | 1315 | 1601.666667 |
| 140 | 1360 | 1431.66667 | 1735 |
| 150 | 1466.666667 | 1561.66667 | 1865 |
| 160 | 1581.666667 | 1691.66667 | 2016.666667 |
| 170 | 1691.666667 | 1786.66667 | 2133.333333 |
| 180 | 1795 | 1908.33333 | 2271.666667 |
| 190 | 1918.333333 | 2035 | 2391.666667 |
| 200 | 2023.333333 | 2160 | 2531.666667 |

# 附录 2. 面向最优处理时间算法实验数据

表 附 2 - 1 面向最优处理时间的等待时间

Table.Appendix 2-1 Waiting time based on the optimization of handling time

| 时刻 min | 遗传算法 min | 贪婪算法 min | 聚类贪婪遗传算法 min |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 10 | 1.716666667 | 3.716666667 | 2.333333333 |
| 20 | 2.4 | 4.316666667 | 2.966666667 |
| 30 | 2.966666667 | 5.033333333 | 3.466666667 |
| 40 | 3.45 | 5.4 | 3.95 |
| 50 | 4.066666667 | 5.933333333 | 4.083333333 |
| 60 | 4.616666667 | 6.4 | 4.133333333 |
| 70 | 5 | 6.766666667 | 4.25 |
| 80 | 5.466666667 | 7.083333333 | 4.283333333 |
| 90 | 5.85 | 7.15 | 4.333333333 |
| 100 | 6.083333333 | 7.866666667 | 4.416666667 |
| 110 | 6.266666667 | 8.266666667 | 4.416666667 |
| 120 | 6.5 | 8.683333333 | 4.433333333 |
| 130 | 6.683333333 | 9.133333333 | 4.466666667 |
| 140 | 6.783333333 | 9.8 | 4.533333333 |
| 150 | 6.983333333 | 10.11666667 | 4.566666667 |
| 160 | 7.116666667 | 10.71666667 | 4.566666667 |
| 170 | 7.333333333 | 11.11666667 | 4.566666667 |
| 180 | 7.583333333 | 11.51666667 | 4.583333333 |
| 190 | 7.666666667 | 11.7 | 4.583333333 |
| 200 | 7.783333333 | 11.86666667 | 4.583333333 |

表 附 2 - 2 面向最优处理时间的处理时间

Table.Appendix 2-2 Handling time based on the optimization of handling time

| 时刻 min | 遗传算法 min | 贪婪算法 min | 聚类贪婪遗传算法 min |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 10 | 2.466667 | 2.283333 | 3.333333 |
| 20 | 5.483333 | 4.683333 | 5.2 |
| 30 | 8.533333 | 6.233333 | 6.066667 |
| 40 | 9.1 | 7.383333 | 6.416667 |
| 50 | 10.08333 | 8.083333 | 6.866667 |
| 60 | 10.71667 | 8.533333 | 7.15 |
| 70 | 10.56667 | 8.783333 | 7.033333 |

| | | | |
|---|---|---|---|
| 80 | 10.85 | 8.983333 | 7.083333 |
| 90 | 10.85 | 9.333333 | 7.3 |
| 100 | 11.03333 | 9.566667 | 7.35 |
| 110 | 11.06667 | 9.766667 | 7.35 |
| 120 | 11.1 | 9.85 | 7.383333 |
| 130 | 11.06667 | 9.983333 | 7.383333 |
| 140 | 11.13333 | 10.06667 | 7.416667 |
| 150 | 11.33333 | 10.05 | 7.4 |
| 160 | 11.36667 | 10.2 | 7.366667 |
| 170 | 11.46667 | 10.15 | 7.35 |
| 180 | 11.36667 | 10.1 | 7.4 |
| 190 | 11.3 | 10.05 | 7.383333 |
| 200 | 11.35 | 10.06667 | 7.4 |

表 附 2 - 3 面向最优处理时间的最大等待时间

Table.Appendix 2-3 Maximum waiting time based on the optimization of handling time

| 时刻 min | 遗传算法 min | 贪婪算法 min | 聚类贪婪遗传算法 min |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 10 | 3.333333 | 8.333333 | 6.666666667 |
| 20 | 8.333333 | 13.33333 | 11.66666667 |
| 30 | 10 | 23.33333 | 16.66666667 |
| 40 | 10 | 30 | 18.33333333 |
| 50 | 11.66667 | 33.33333 | 18.33333333 |
| 60 | 11.66667 | 33.33333 | 18.33333333 |
| 70 | 11.66667 | 33.33333 | 18.33333333 |
| 80 | 11.66667 | 36.66667 | 18.33333333 |
| 90 | 11.66667 | 38.33333 | 18.33333333 |
| 100 | 16.66667 | 41.66667 | 18.33333333 |
| 110 | 18.33333 | 51.66667 | 18.33333333 |
| 120 | 21.66667 | 61.66667 | 18.33333333 |
| 130 | 23.33333 | 71.66667 | 18.33333333 |
| 140 | 28.33333 | 81.66667 | 18.33333333 |
| 150 | 28.33333 | 88.33333 | 18.33333333 |
| 160 | 28.33333 | 88.33333 | 18.33333333 |
| 170 | 28.33333 | 88.33333 | 18.33 |
| 180 | 28.33333 | 88.33333 | 18.33 |
| 190 | 28.33333 | 88.33333 | 18.33 |
| 200 | 28.33333 | 88.33333 | 18.33 |

表 附 2 - 4 面向最优处理时间的处理数量

Table.Appendix 2-4 Number of handling requests based on the optimization of handling time

| 时刻 min | 遗传算法（次） | 贪婪算法（次） | 聚类贪婪遗传算法（次） |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 10 | 16.66666667 | 26.6666667 | 33.33333333 |
| 20 | 96.66666667 | 101.666667 | 150 |
| 30 | 228.3333333 | 233.333333 | 280 |
| 40 | 336.6666667 | 348.333333 | 408.3333333 |
| 50 | 461.6666667 | 456.666667 | 533.3333333 |
| 60 | 573.3333333 | 591.666667 | 688.3333333 |
| 70 | 698.3333333 | 685 | 845 |
| 80 | 820 | 835 | 976.6666667 |
| 90 | 900 | 933.333333 | 1121.666667 |
| 100 | 1028.333333 | 1055 | 1248.333333 |
| 110 | 1131.666667 | 1190 | 1380 |
| 120 | 1230 | 1323.33333 | 1523.333333 |
| 130 | 1341.666667 | 1458.33333 | 1668.333333 |
| 140 | 1465 | 1576.66667 | 1785 |
| 150 | 1565 | 1713.33333 | 1908.333333 |
| 160 | 1685 | 1836.66667 | 2053.333333 |
| 170 | 1788.333333 | 1955 | 2191.666667 |
| 180 | 1921.666667 | 2068.33333 | 2315 |
| 190 | 2010 | 2185 | 2471.666667 |
| 200 | 2130 | 2285 | 2631.666667 |

# 致　谢

本论文是在王春香老师悉心指导下完成的。王老师对我的研究生生涯乃至整个人生都产生了积极而深远的影响。王老师在本科阶段就是我的班主任，7 年以来一直对我照顾关心，她总是让我以自己最舒服最能发挥自己特长的方式成长，同时在学业中又不乏建设性地指导，她对我的关心就像母亲关心自己的孩子一样。在此，要感谢王老师在论文写作中提供的指导和帮助，王老师的经验和意见对于本文的最终形成有非常重要的指导作用。

感谢研究生实验室，实验室中的师兄师姐师弟师妹们就像一家人，大家在研究生阶段一同经历了欢乐的真人 CS，别墅大轰趴，上海夜骑行，也经历了紧张激烈的无人车挑战赛，挑灯夜战的系统开发，实验室让我在研究生阶段收获了欢乐，感动同时也体会了团队的力量，合作的魅力。并且实验室的师兄弟们对于本文也有很多具有建设性的指导作用。

感谢我的家人，在做课题的时间里，父母经常关心我的身体，关心我的生活饮食，关心我的课题进度，并且经常鼓励我，让我更加有信心做好课题，完成本论文。

感谢我的挚友段必义同学，胡睢宁同学，蒋志鹏同学。三年来，他们给我带来了研究生阶段不少的快乐时光，在做论文阶段经常帮助我，给了我很多指导性的建议。

最后感谢所有帮助过我的老师，亲人和同学，因为有你们大家我的生活变得阳光，向上。

# 攻读硕士学位期间已发表或录用的论文

[1] 李爽，杨明，王春香，王冰. 面向全自动控制交通系统的车辆调度算法研究[J]. 上海交通大学学报. 2015.