

石河子大学

本科毕业论文(设计)



蚁群算法在外卖配送路径优化问题中的 应用研究与实现

学 生 姓 名 :	汪 琰
学 号 :	20161008010
学 院 :	信息科学与技术学院
专 业 :	计算机科学与技术
年 级 :	2016 级
指 导 教 师 :	曹传东

中国·新疆·石河子

2020 年 6 月

摘 要

本课题针对外卖配送的路径规划问题，利用以蚁群算法为核心的算法思想，通过 Android Studio 的 Java IDE 的开发平台，综合利用爬虫知识，Matlab 工具，安卓开发技术，设计实现一个以蚁群算法为核心，用于为单个骑手规划最优外卖配送路径的移动 APP，该软件在路径规划方面能够满足骑手的多种需求，在实际配送中面对多个订单的配送时能够减少配送成本，提高配送效率，和基于非智能算法的同类移动 APP 相比，克服了耗时耗力的缺点，能够在一定程度上减轻外卖配送骑手的出行压力，具有深入开发的研究意义，因此不失为一种可行的选择。

关 键 词：配送路径优化；蚁群算法；TSP；外卖骑手 app

Abstract

In this paper, aiming at the path planning of takeout distribution, using ant colony algorithm as the core algorithm idea, through the development platform of Java IDE based on Android studio, using reptile knowledge, MATLAB tools and Android development technology, we design and implement a mobile app with ant colony algorithm as the core, which is used to plan the optimal delivery path for a single rider, The software can meet the various needs of riders in the aspect of path planning, reduce the distribution cost and improve the distribution efficiency when facing the distribution of multiple orders in the actual distribution. Compared with the similar mobile app based on the non intelligent algorithm, the software overcomes the shortcomings of time-consuming and labor-consuming, and can reduce the travel pressure of riders in the delivery distribution to a certain extent, which has the significance of in-depth development, Therefore, it is a feasible choice.

Key words: Takeaway Delivery Route Optimization Problem; Ant Colony Algorithm; TSP; Takeaway Food Delivery App

目 录

摘 要.....	I
Abstract.....	II
第一章 绪论.....	1
1.1 题目	1
1.2 课题背景	1
1.2.1 国外研究现状	2
1.2.2 国内研究现状	5
1.3 课题研究目的及意义	6
1.3.1 研究目的	6
1.3.2 研究意义	6
1.4 设计时间	7
1.5 课题内容及成果说明	7
1.5.1 课题内容	7
1.5.2 课题成果	7
第二章 相关技术原理及开发环境.....	8
2.1 相关技术原理	8
2.1.1 外卖路径优化问题	8
2.1.2 蚁群算法	10
2.1.3 Matlab	10
2.1.4 网络爬虫	10
2.1.5 Android Studio	11
2.1.6 高德地图 API	11
2.2 开发环境	12
2.2.1 系统硬件环境	12
2.2.2 系统开发工具	12
第三章 需求分析.....	13
3.1 引言	13
3.2 可行性分析	13
3.2.1 技术可行性	13
3.2.2 经济可行性	13
3.2.3 操作可行性	13
3.2.4 法律可行性	13
3.3 功能需求	14
3.3.1 爬虫器模块	14
3.3.2 网页信息抽取模块	14
3.3.3 外卖订单模块	14
3.3.4 地图定位模块	14
3.3.5 路径规划模块	14
3.3.6 优化路径图示模块	14
3.4 系统的 UML 建模	14
3.4.1 用例图	14
3.4.2 类图	16

3.4.3 时序图	18
3.5 系统性能需求	22
3.6 系统数据流图	22
3.6.1 顶层数据流图	22
3.6.2 一层数据流图	23
3.6.3 二层数据流图	23
3.7 加工说明	24
3.8 数据字典	25
第四章 详细设计	26
4.1 概述	26
4.2 系统功能模块图	26
4.2.1 网页爬取模块	27
4.2.2 网页信息抽取模块	28
4.2.3 外卖订单模块	29
4.2.4 地图定位模块	30
4.2.5 路径规划模块	31
4.2.6 优化路径图示模块	32
4.3 系统流程图	33
第五章 外卖路径优化问题建模及算法实现	34
5.1 外卖路径优化问题的数学模型	34
5.2 算法开发环境	34
5.3 算法详细设计	34
5.3.1 参数说明	34
5.3.2 算法流程图	34
5.3.3 算法分析	34
5.4 算法的打包封装	36
第六章 系统测试	39
6.1 编写目的	39
6.2 背景	39
6.3 测试方法	39
6.4 测试概要（或测试过程）	39
6.4.1 网页爬虫模块测试	39
6.4.1 数据抽取模块测试	40
6.4.2 路径规划模块测试	40
6.4.3 优化路径图示模块及定位模块测试	41
6.4.4 外卖订单模块测试	46
6.5 对软件功能的评价	49
第七章 结论与展望	50
7.1 结论	50
7.2 展望	50
参考文献	51
致谢	53

第一章 绪论

1.1 题目

蚁群算法在外卖配送路径优化问题中的应用研究与实现

1.2 课题背景

随着社会的进步与技术的发展，互联网行业迎来了新一轮的高潮，“互联网+”逐渐开始走进人们的生活。新时代的“四大发明”——高铁，移动支付，共享单车，网络购物极大丰富与便利了人们的日常生活。外卖作为一种新形式，近距离的网络购物，在人们的日常生活中承担着不可或缺的任务。然而随着城市化进程的推进，越来越多的人开始享用信息化的成果，其中外卖的配送订单的数量以我们肉眼可见的速度在不断攀升。有研究数据表明^[1]，仅在 2018 年我国外卖交易规模就达到 2480 亿元，相比较 2017 年的 2096 亿元增长 18.3%；同时我国外卖用户规模达 4.06 亿人，比 2017 年的 3.1 亿人增长了 31%。由此预见，将来的外卖使用人次可能会再上一个高峰。规模不断扩大必然会导致社会资源的剧烈消耗，相关产业必然会面临如何降低运营成本以及减少资源消耗问题，以本课题为例，主要解决在外卖骑手选择接收多个订单的情况下，规划出最佳路径，以最低代价将外卖送到顾客手中。

从骑手角度出发，骑手从一个地方出发，需要给他规划一个最优线路，这个骑手应该在最少的时间内配送最多的订单且累积路程最短。如果再简化一下这个问题，例如：如何规划骑手的配送路线，使得骑手的送货距离最短，从而优化单个骑手的配送效率？问题便瞬间变为单个骑手的路径规划问题，而这便是一个典型的 TSP 问题：给定起始点和途经点，求通过所有指定点的最短路径。由于 TSP 是一个 NP-hard 问题，没有多项式时间内的确切解法。为了解决这个问题，需要使用遗传算法、蚁群算法等启发式算法，来快速获取骑手的较短配送路线，并且能够同时得到规划后的路线的最终配送距离。在待规划点超过 10 个以上的情况下，算法的运行时间也能保持在几毫秒，同时保证极高的准确性。有了这个基础性的算法，外卖调度系统后台就可以判断订单间的顺路情况，以及某个订单是否适合由某个骑手来配送。当然，在订单分配的过程中，不仅需要考虑顺路情况、配送效率等，也要考虑骑手的接单偏好、配送能力、活跃状况，以及订单分配的公平性等因素。总而言之，订单指派是一个有限约束的多目标优化问题，不仅需要考虑平台效率，也要考虑骑手的心理因素、运营的稳定性等。因此，为单个骑手设计实现一个能够解决遍历路径优化这种 TSP 问题的算法，便成为了解决复杂订单分配调度问题的基础和出发点。本课题便尝试将蚁群算法这一仿生优化算法应用于解决外卖配送中的路径规划问题。

1.2.1 国外研究现状

蚁群算法^[2]是一种正反馈机制的算法，首先由 M. Dorigo 提出的蚂蚁系统 (Ant System, AS) 是最早的蚁群算法，并以此为基础发展而来，该算法模拟蚂蚁觅食的过程，在没有任何提示下找到从巢穴到食物的最短路径，并且能随着环境的变化，适应性的搜索新的路径，找到新的求解路径。如下图例所示：

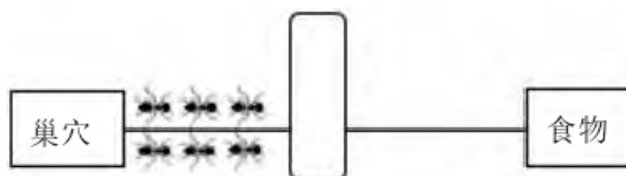


图 1-1 开始觅食

图 1-1 所示，蚂蚁从巢穴出发寻找食物，从两条不同路径出发的蚂蚁的数量是相同的。但是在到达某个时刻，右路出发的蚂蚁到达食物时，左边的蚂蚁明显还未到达。

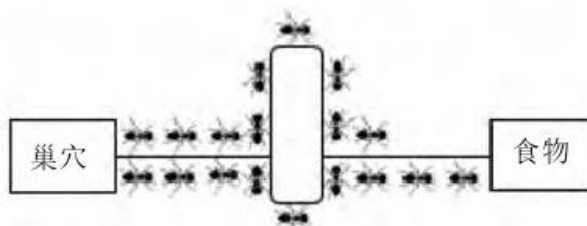


图 1-2 路径寻找过程

图 1-2 所示，当左路蚂蚁到达食物时，右路蚂蚁已经开始返回，由此可见，右路蚂蚁留下的信息素浓度比左路高，通过不断地迭代，随着右边的信息素浓度不断提高，我们可以猜想，所有的蚂蚁基本上都会选择右边的路径，从而达到收敛。如下图所示：

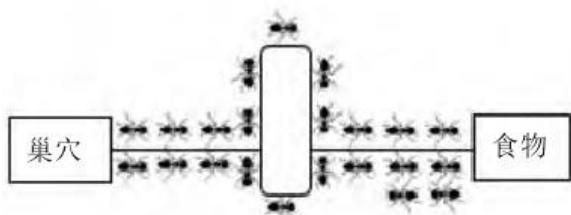


图 1-3 形成趋势

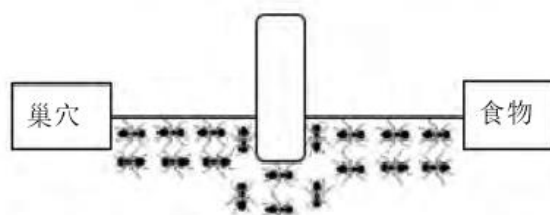


图 1-4 路径收敛

M. Dorigo 提出了最早的关于蚁群算法的数学模型，奠定了蚁群算法的基础，算法指出当某条路径上的蚂蚁越多时，相对应释放的信息素就会越多，信息浓度就会越高，其他蚂蚁选择该路径的可能性就越大，这样个体之间通过信息素交流，最终收敛于最短路径，以下是 M. Dorigo 提出的蚁群算法的基本数学模型：

- (1) 在初始时刻，有 m 个蚂蚁， n 个城市，并且每条路径上的信息素相等；

(2) m 个蚂蚁随机散落在 n 个城市，设为时刻 t ，蚂蚁 k 在第 i 个城市，此时蚂蚁 k 向下一个城市 j 移动的概率为 $P_{ij}^k(t)$ ；其计算公式如下：

$$P_{ij}^k = \begin{cases} \frac{[t_{ij}(t)]^\alpha \cdot [\eta_{ij}(t)]^\beta}{\sum_{s \in allow} [t_{ij}(t)]^\alpha \cdot [\eta_{ij}(t)]^\beta} & s \in allow_k \\ 0 & s \notin allow_k \end{cases}$$

$t_{ij}(t)$ 表示 t 时刻城市 i 与城市 j 之间的信息素浓度；

$allow_k = \{1, 2, 3, \dots, n\} - t_{ij}(t)$ ，其中 $t_{ij}(t)$ 记录了蚂蚁 k 在本次迭代中已经走过的城市，不允许蚂蚁 k 在本次循环中再次去遍历这些城市；当所有的 n 座城市都加入 $t_{ij}(t)$ 中，则蚂蚁 k 完成了一次周游；

$\eta_{ij}(t)$ 表示启发函数，也被称作能见度因子；一般用城市 i 与城市 j 之间的距离的倒数表示，即 $\eta_{ij}(t) = 1/d_{ij}$ ；

α 为信息素的重要程度因子，其值越大，转移中起的作用越大

β 为启发函数的重要程度因子，其值越大，表示启发函数在转移中的作用越大，即蚂蚁以较大的概率转移到距离短的城市。

(3) 蚂蚁释放的信息素会随时间的推进而减少，设参数 ρ ($0 < \rho < 1$) 表示信息素的挥发度，当所有蚂蚁完成一次循环周游后，则各个路径上的信息素需要更新，其相应的公式如下：

$$t_{ij}(t+1) = (1-\rho)t_{ij}(t) + \Delta t_{ij}, \quad \Delta t_{ij} = \sum_{k=1}^n \Delta t_{ij}^k$$

其中根据每次迭代中不同路径上信息素的增量划分，蚁群算法的模型大致可以分为蚁周系统模型 (ant cycle system)，蚁量系统模型 (ant quantity system)，蚁密系统模型 (ant density system)，具体表述如下：

$$\text{蚁周系统模型: } \Delta t_{ij}^k = \begin{cases} Q/L_k & \text{第 } k \text{ 只蚂蚁从城市 } i \text{ 访问城市 } j \\ 0 & \text{其他} \end{cases}$$

Q 为常数，表示蚂蚁循环一次释放的信息素的总量；

d_{ij} 为第 k 只蚂蚁经过路径的长度 (Length)；

$$\text{蚁量系统模型: } \Delta t_{ij}^k = \begin{cases} Q/d_{ij} & \text{第} k \text{只蚂蚁从城市} i \text{访问城市} j \\ 0 & \text{其他} \end{cases}$$

Q 为常数，表示蚂蚁循环一次释放的信息素的总量

d_{ij} 为城市 i 到城市 j 的距离。

$$\text{蚁密系统: } \Delta t_{ij}^k = \begin{cases} Q & \text{第} k \text{只蚂蚁从城市} i \text{访问城市} j \\ 0 & \text{其他} \end{cases}$$

Q 为常数，表示蚂蚁循环一次释放的信息素的总量

其中蚁量系统模型与蚁密系统模型的信息素更新方式均是通过局部更新信息素，只有蚁周系统采用全局更新信息素的方式，因此，蚁周系统模型是标准蚁群算法的基本模型。

随着蚁群算法在实际生活中的应用，一些弊端与不足也逐渐显露出来，传统的蚁群算法的动态因子在调节全局信息素浓度时不具有自适应性，在首次循环时，无法确保蚂蚁经过的路径上所释放的信息素是最优方向，其次，由于全局算法的搜索能力不足，容易导致出现相同解，通过多次迭代后产生局部最优解而不是全局最优解，容易陷入停滞现象；不仅花费时间较长，而且由于需要对所有搜索路径都要更新信息素，因此会造成搜索效率低下。

鉴于上述蚁群算法所出现的问题，很多学者都提出了不同的改进方案^[3]：

(1) 蚁群系统 (Ant System, AS) 是由 Gambardella 等人在 1996 年提出的一种改进的蚁群算法，该算法通过引入新常量来决定蚂蚁每次选择的路径，此外通过对全局信息素和局部信息素的更新策略改进，在一定程度上缩短了搜索时间以及缓解了停滞现象的出现。

(2) 精英蚁群算法 (Elitist Ant System, EAS) 是 M. Dorigo 等人在基于 AS 的蚁群算法上提出的一种改进算法，该算法通过对路径选择概率的改进来增强正反馈效果，即当最优路径在修改信息素轨迹时，通过人工释放额外的信息素。

(3) 最大最小蚁群系统 (Max-Min Ant System, MMAS) 是由德国学者 Thomas Stutzle 提出来的，该算法与 EAS 改进的方向不同，该算法通过对信息素浓度更新的改进，来达到寻找最优解的目的。此外，还为了防止因为信息素浓度影响而出现极端情况，设定了信息素浓度区别。

此外，还有很多不同的改进策略与方法，基本都是通过对路径选择的改进与信息素浓度更新的改进两个角度来实现的，此外还有通过与其他算法有机结合的思路来提高效率，比如典型的排序蚁群算法，就是通过引入遗传算法中的排序概念来达到寻找最优解的目的。

通过国外学者提出的蚁群算法及其改进思路，相对于研究来说，更偏向于侧重一种思路的改进方案，其中包括专门针对启发函数，信息素浓度，信息素的更新或者结合其他智能算法提出的改进想法，这有利于我们在蚁群算法的研究领域中深入交流，提出相同基础不同方向的理论成果。

1.2.2 国内研究现状

随着蚁群算法的改进和发展，国内的许多学者在这方面取得的成绩也非常显著，针对蚁群算法的“瓶颈”，国内学者将目光关注在算法的收敛性与连续空间寻优方面，提出了针对不同问题的蚁群算法：

(1) 针对旅行商问题的蚁群算法，国内学者吴庆洪^[4]受到遗传算法中变异算子的启发，提出了具有新特性的蚁群算法，该算法中提出了变异特征的有关内容，针对旅行商问题中，引入了最小生成树的概念，同时通过提出新的量度构造出动态的候选集，提高了搜索精度，明显改善了收敛性。

(2) 针对避免拥堵的最优路径规划的蚁群算法，胡坤^[5]提出了避免拥堵的蚁群算法 (Avoid Traffic Jam Ant Colony Optimization, 简称 ATJ-ACO)，他在原有的蚁群算法基础上，研究加入了动态的拥堵因素，通过建模和寻找最优解，对原有的蚁群算法加以改进，解决了出行拥堵问题，在对本次课题的路径规划拥有重大的参考意义。在该算法中，作者通过启发函数来改变蚂蚁寻径规则，从而可以达到减小盲目搜索的效果，在削弱最差解的同时，加强了最优解，使得蚁群算法更快收敛于最优解。

(3) 苏晓勤^[6]在基于信息素改进的蚁群算法中针对传统蚁群算法在求解 TSP 问题时容易陷入局部最优，过早收敛，出现停滞等现象，引入了参数 ε 来控制蚂蚁经过路径时留下的信息素的量，如图所示，

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \varepsilon\Delta\tau_{ij}^{best}(t)$$

图(5) 信息素增量表达式

苏晓勤除了通过引入参数使得信息素的增量在每次迭代中均产生相应的变化，同时

还在搜索路径阶段提出了分段迭代思想，在搜索前期将解空间最大化，中期保证了收敛和解空间保持并行，后期有效加速收敛，能够保证改进后的蚁群算法利用较少时间快速收敛最优解。

(4) 针对蚁群算法的应用方面，黄心^[7]在其论文中主要研究蚁群算法在外卖配送路径规划中的应用，其利用 Dorigo.M 提出的人工蚁群算法，针对配送员在接单时间内依次到各个不同的位置进行配送，采用模拟仿真技术，顺利实现了最优路径规划，不仅验证了算法的可行性，更赋予了该算法在指导解决现实问题中的应用价值。

从总的方面来看，无论是从国外研究学者的理论成果，还是国内学者的发表文献来看，针对蚁群算法的改进主要从三个方面着手，即路径选择，信息素浓度以及信息素的更新，最终达到加速收敛最优解的效果，从而可以针对不同问题具体分析，从侧重点不同来选择不同问题的最优解，和国外研究不同的是，国内学者们更倾向于抓住不同控制因素之间的联系，通过控制各个因素的改变达到整体上提高效率的目的，相对比国外学者，具有广泛性。

1.3 课题研究目的及意义

1.3.1 研究目的

本课题针对利用蚁群算法解决外卖配送的路径规划问题，研究基于 Android Studio 的 Java IDE, 综合利用爬虫知识, Matlab 工具, 安卓开发技术, 设计实现一个基于 Android 并且利用蚁群算法用于为单个骑手规划最优外卖配送路径的移动 APP, 在路径规划方面能够满足骑手的多种需求。

通过本课题，深刻理解蚁群算法，并利用其解决外卖配送中的路径优化问题，能够规划出骑手配送的最优路径，即用时最短，配送最多且积累路程最短。确保骑手能够快速获取配送路线，并且能够得到规划后路线的最终配送距离。保证系统运行的高效性和准确性，确保规划点能够迅速反馈结果，保证整个过程在几毫秒之间。保证后台能够就判断骑手顺路情况以及该骑手与相应订单之间是否适合，能够依据多方面因素，例如顺路情况，骑手的接单偏好等做出决策。

1.3.2 研究意义

(1) 学术意义：通过本次课题的研究，能够在理解蚁群蚁群算法的前提下，增加对现实问题的反应，扩大蚁群算法的研究范围，同时为蚁群算法的研究方向填充新的内容。

(2) 应用意义：通过学习蚁群算法，能够将现实环境中的问题抽象成数学模型，采用实际与应用相结合的方式，将问题从现实环境中抽取出来，进行理论化，进而指导现实问题的解决，帮助人们在日常生产消耗中通过优化生产方式达到降低生产成本，提高生产效率的目的。

(3)学习意义：通过对课题的研究与实现，首先能够增强智能算法在现实生活中的应用领域的了解；其次能够体会整个软件开发的过程经历，了解集成式开发的优势；同时能够锻炼开发者对于不同软件工具的使用；在整个过程中开发者可以增强对网络架构的理解与实践，在面向网络编程时能够切身体会不同层次之间的联系与区别。

1.4 设计时间

2019 年 11 月-2020 年 6 月

1.5 课题内容及成果说明

1.5.1 课题内容

- (1) 完成课题开发前的需求分析工作以及系统模型。
- (2) 完成软件开发后期的数据收集工作，利用爬虫工具爬取本课题中需要的订单信息。
- (3) 利用移动 app 开发，Matlab，Java 等相关技术，基于 Android Studio 的 Java IDE，设计实现为单个骑手规划最优外卖配送路径的蚁群算法。
- (4) 利用 Android Studio 或 Eclipse 等相关集成开发工具，结合高级编程语言（例如 Java）开发出以上述算法为核心的应用 app。

1.5.2 课题成果

- (1) 一个以蚁群算法为核心的外卖配送 App。
- (2) 一份用户手册。
- (3) 一份设计论文文档。

第二章 相关技术原理及开发环境

2.1 相关技术原理

2.1.1 外卖路径优化问题

(1) 问题描述：以北京中关村为例：午餐高峰期大约有 200 个左右的骑手，如何在他们之间进行订单分配？最表层的理解是考虑多目标优化。在午高峰的时候每分钟大约有 50 个左右的新订单进来，每个骑手身上可能都有待配送的订单。将订单分配给某个骑手，需要考虑他是不是顺路，他是不是交通比较熟，是不是效率能够保证。还有一些质量要求，例如：用户点了面条，就要及时的送到，否则就会影响口感。此外，骑手的位置是不断在变化着的，可能上一秒他还适合配送这一单，但是在这一秒就不适合了。

(2) 数学建模：上述中的外卖订单分配问题，其实可建模为带有若干复杂约束的 DVRP (Dynamic Vehicle Routing Problem) 问题。这类问题一般可表述为：有一定数量的骑手，(如 1 分钟)内产生了一批新订单，已知骑手的行驶速度、任意两点间的行驶距离、每个订单的出餐时间和交付时间（骑手到达用户所在地之后将订单交付至用户所需的时间），那么如何将这批新订单在正确的时间分配至正确的骑手，使得用户体验得到保证的同时，骑手的配送效率最高。

美团外卖每天产生巨量的订单配送日志，行驶轨迹数据。通过对配送大数据进行分析、挖掘，会得到每个用户、楼宇、商家、骑手、地理区域的个性化信息，以及有关地理区块路径的有效数据，那么，订单智能分配/调度的目标就是基于后台的大数据，根据订单的配送需求、地理环境以及每名骑手的个性化特点，实现订单与骑手的高效动态最优分配，从而为每个用户和商家提供最佳的配送服务，并降低配送成本。

因此，即时配送订单分配问题的优化目标包括：希望用户的单均配送时长尽量短、骑手付出的劳动尽量少、超时率尽量低，等等。一般可表达为如下的目标函数：

$$\begin{aligned} & \text{目标1：最小化超时率} \\ \min g_1(\Omega) &= \frac{\sum_{i=1}^n 1(f_{(i,C)} \geq d_i)}{n} \\ & \text{目标2：最小化单均行驶距离} \\ \min g_2(\Omega) &= \frac{\sum_{j=1}^m \sum_{k1,k2 \in seq_j} l_{k1,k2}}{n} \\ & \text{目标3：最小化单均消耗时间} \\ \min g_3(\Omega) &= \frac{\sum_{j=1}^m \max_{(i,x) \in seq_j} T_{(i,x)}}{n} \end{aligned}$$

图 2-1 数学建模公式

除了要考虑时间、地段等约束外，有时还需要考虑部分订单只能由具备某些特点的骑手来配送(例如火锅订单只能交给携带专门装备的骑手等)、载具的容量限制等。美团的即时配送后台大数据平台需要实现对骑手轨迹数据、配送业务数据、特征数据、指标数据的全面管理和监控，并通过模型平台、特征平台支持相关算法策略的快速迭代和优化。美团的订单分配或者说外卖调度优化问题的模型如图所示：



上述外卖调度系统的机器学习模块，负责从数据中寻求规律和知识，例如对商家的出餐时间、到用户所在楼宇上下楼的时间、未来的订单、骑行速度、红绿灯耗时、骑行导航路径等因素进行准确预估，为调度决策提供准确的基础信息；而运筹优化模块，则在即时配送大数据平台以及机器学习的预测数据基础上，采用最优化理论、强化学习等优化策略进行计算，做出全局最优的分配决策，并和骑手高效互动，处理执行过程中的问题，实现动态最优化。

(3) 简化为 TSP^[8]问题描述：假设骑手从一个地方出发，需要给他规划一个最优线路，这个骑手应该在最少的时间内配送最多的订单且累积路程最短。如果再简化一下这个问题，例如：如何规划骑手的配送路线，使得骑手的送货距离最短，从而优化单个骑手的配送效率？问题便瞬间变为单个骑手的路径规划问题，而这便是一个典型的 TSP 问题：给定起始点和途经点，求通过所有指定点的最短路径。由于 TSP 是一个 NP-hard 问题，没有多项式时间内的确切解法。为了解决这个问题，需要使用遗传算法、蚁群算法等启发式算法，来快速获取骑手的较短配送路线，并且能够同时得到规划后的路线的最终配送距离。在待规划点超过 10 个以上的情况下，算法的运行时间也能保持在几毫秒，同时保证极高的准确性。有了这个基础性的算法，外卖调度系统后台就可以判断订单间的顺路情况，以及某个订单是否适合由某个骑手来配送。当然，在订单分配的过程中，不仅需要考虑顺路情况、配送效率等，也要考虑骑手的接单偏好、配送能力、活跃状况，以及订单分配的公平性等因素。总而言之，订单指派是一个有限约束的多目标优化问题，不仅需要考虑平台效率，也要考虑骑手的心理因素、运营的稳定性等。

因此，为单个骑手设计实现一个能够解决遍历路径优化这种 TSP 问题的算法，便成

为了解决复杂订单分配调度问题的基础和出发点。本课题便尝试将蚁群算法这一仿生优化算法应用于解决外卖配送中的路径规划问题。

(4) 候选算法：蚁周系统(Ant Cycle System)、蚁量系统(Ant Quantity System)，蚁密系统(Ant Density System)。

2.1.2 蚁群算法

(1) 算法介绍：作为一种启发式算法，蚁群算法中每代蚂蚁在其行驶过程中，会在行驶过的路径上留下一定量的信息素，该信息素会随着时间进行挥发，该算法主要通过改变信息素来影响下一代蚂蚁在行驶路线中对不同路线选择的概率，通过多代蚂蚁迭代运行后，信息素浓度会使得所有蚂蚁收敛出一条最佳路线。

(2) 参数说明：

表 2-1 蚁群算法参数表

M	蚂蚁数量
α	信息素重要程度因子
β	启发函数重要程度因子
ρ	信息素挥发系数
NC_max	最大迭代次数
Q	循环一次释放的信息素总量
η	启发函数
Tau	信息素表
Tabu	禁忌表

2.1.3 Matlab

(1) 工具介绍：利用 matlab 工具进行核心算法的实现^[9]，即蚁群算法的实现。由于蚁群算法相比较传统算法实现过程更加复杂，模拟参数的变化更加频繁，matlab 针对数学模型的计算具有较强优势，能够适应整个运算过程中的数据扩张问题。

(2) 编码实现：本课题中需要利用 matlab 开发工具，实现整个应用软件核心部分开发，即蚁群算法的实现，利用 matlab 在处理计算方面的优势能够在相当程度上简化代码，减轻开发人员负担。

2.1.4 网络爬虫

(1) 爬虫介绍^[10-11]：数据集作为整个课题开发的基础部分，想要收集针对课题所需的数据集，利用 python 爬虫来完成，整个爬虫爬取的对象为饿了么官网数据，通过事先预设的爬取内容，主要用于获取该网站中部分地区的商店 ID 名，以及商店的详细地址，经纬度，店铺名称，月销量，起送金额，最低配送以及联系电话等等。以此作为后期中数据基础，用于实现模拟订单进行骑手行驶路线的最优规划。

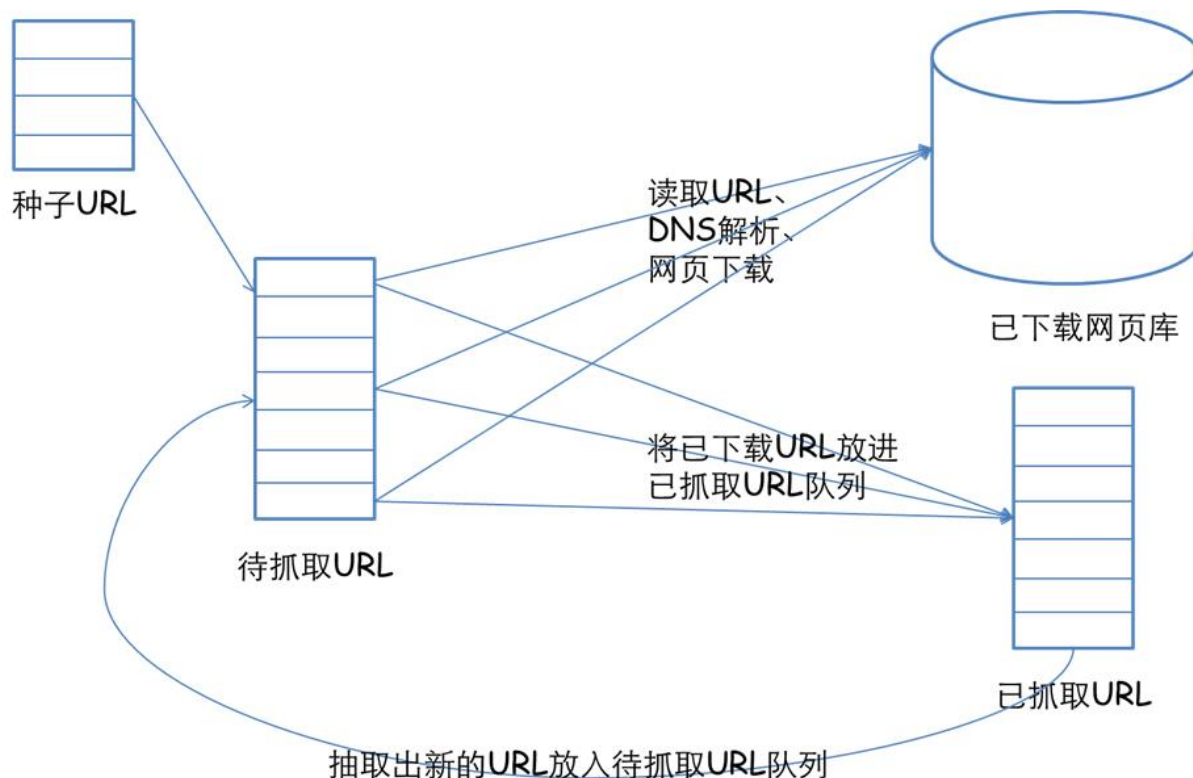


图 2-1 爬虫框架

(2) 外卖数据获取策略简介: 首先利用 Python 代码, 设置好实现待爬取网站的 url, 通过对代码的伪装, 即在头部数据添加浏览器头部信息, 例如“User-Agent”, “Cookies”等伪装成浏览器进行访问, 生成请求对象并对访问的网站发送请求, 之后将接收的数据进行格式化编码, 一般采用“UTF-8”格式, 针对需要数据进行字符匹配, 可利用正则表达式或者第三方库提供的匹配方法进行识别, 最后保存至本地文件, 即我们需要的数据集中。

2.1.5 Android Studio

(1) 任务说明: Android studio 是 Google 公司研发的 Android 集成开发工具^[12], 本次课题中主要利用该开发工具来完成 App 应用的构建, 将蚁群算法为核心的模块搭载于整个 App 之中, 保证课题的实际成果能够顺利在实际平台, 即 Android 手机上运行。

(2) 选择理由: 利用 Android 开发构建整个 app 的运行框架, 使其能够兼容市场上 Android 4.0 及其以上的智能手机, 用户范围广, 其次能够 Android Studio 相比较其他的 Android 开发工具, 具有功能强大, 构建灵活, 支持多种编码等优势, 而且该工具是基于 Gradle 的灵活构建系统上进行开发的, 其中 Instant Run 不仅不需要重新构建 Apk, 还可以将变更推送到正在运行的应用中。

2.1.6 高德地图 API

(1) 基本介绍: 高德开放平台提供 2D 3D 以及卫星等多种地图显示供开发者选择, 开发者可根据自身情况选择平台开放提供的 API 和 SDK 完成地图的构建工作, 除此之外高德地图还提供多种服务以及包括定位导航路线规划等在内的一系列功能, 开发者可根

据该平台提供的接口创建地图，还可以通过与地图交互的方式在地图上进行绘制以及获取地图的数据，含获取出行路线规划，天气情况等信息在内的工具的使用方法。

(2) 调用方式：针对需要实现的不同功能，开发者需要在该平台上下载相应的 SDK 工具包并将相应的 SDK 导入工程中，其次需要在控制台获取高德 Key 以便于调用接口时通过验证，然后在 Android Studio 平台中配置该数据并开启对应服务(例如网络服务)，便可以在代码中通过方法调用来调用该接口。

2.2 开发环境

2.2.1 系统硬件环境

- (1) CUP: Inter(R) Core(TM) i7-9750H CPU @2.60GHz 2.59GHz
- (2) 内存: 16GB
- (3) 系统类型: 64bit
- (4) 操作系统: Windows 10 家庭中文版

2.2.2 系统开发工具

- (1) Matlab R2018b
- (2) Android Studio 3.5.2
- (3) Eclipse 4.5.2
- (4) Python 3.5

第三章 需求分析

3.1 引言

针对本次课题的研究，在开发之前需要进行需求分析工作，本章内容将从以下几个方面来展开，首先从可行性方面来确认该课题是否具备实现条件以及实现价值，在肯定本课题的可行性后，还需要对本课题中的整体功能展开讨论，确定整个项目模块以及各模块的内部功能以及外部联系，之后利用软件工程知识进行 UML 建模，确定本次课题的问题规模以及性能上的需求，最终完成整个项目的建模过程，同时对项目中的具体细节进行记录和备注。

3.2 可行性分析

3.2.1 技术可行性

整个课题在技术层面上具有一定的广泛性，需要综合利用各种技术来完成项目的实行。综合考虑各种技术的应用方向以及难度等一系列问题，通过多个模块分块进行完成，最终将组件部份进行链接合并，整个过程中尽管扩大了技术使用范围，但同时在相当大的程度上降低了难度，大大降低了技术层面上的风险，在实际开发过程中，若遇到了其他不可规避的风险，则可以采用同等类型且不影响课题结论的方法或手段进行替代，相似课题研究方法在该研究领域均存在多种解决方案，目前就技术层面上而言还是具有较为成熟的应对方案。

3.2.2 经济可行性

本课题的研究实现工具，由于仅用于学生研究和个人使用，因此使用的工具均为学习版本或免费版本。课题实现过程均由学生本人完成，不存在额外花销或经济成分，因此经济方面是可行的。

3.2.3 操作可行性

本次课题中通过接受用户的订单进行路径规划，由于是市场化的前期模拟，因此可以通过爬虫数据收集的形式来手动输入模拟订单，在真实的市场运行中是可以直接省略本步骤，通过接口调用由计算机自动进行数据的输入与处理，骑手只需要接收当下多个订单后点击提交按钮，app 将自动向服务端传递数据并接收服务端响应的结果，在地图模块上进行路径规划。整个过程对于骑手基本上是透明的，在操作上是可行的。

3.2.4 法律可行性

本课题的研究与应用开发完全遵循中华人民共和国的有关法律条文，具备法律可行性。

3.3 功能需求

3.3.1 爬虫器模块

课题需要在后期完成开发的基础上利用正确数据进行测试，因此爬虫器模块主要用于向相关网站请求相关数据并接收至当前计算机下，该模块需要相应网站的 URL，模拟浏览器的 user-agent 和 cookie 等头部信息，此外还包含 Python 所支持的一些数据请求接口。

3.3.2 网页信息抽取模块

本模块主要用于对爬虫器爬取的信息进行处理，针对已爬取的信息进行格式转化，以及利用本地库或者第三方支持库进行关键信息匹配，最终将所需要的信息进行分类保存并写入文件。

3.3.3 外卖订单模块

该模块主要用于数据输入以及数据提交，在课题开发过程中，需要对接收的订单进行处理，主要包括订单添加，订单删除以及提交订单等功能，本例中该模块除了完成以上功能，还需要显示外卖骑手接收的订单内容。

3.3.4 地图定位模块

骑手在配送订单过程中，需要能够及时了解到自身所处位置以及距离下一个订单地址的实时距离，模块能够实现骑手在外卖配送过程中实时的动态地理位置变化的显示，使得骑手能够随时观察自身位置情况，根据订单详情做出及时的判断。

3.3.5 路径规划模块

本课题需要在骑手提交订单后能够及时在返回的数据请求中提供相应订单的配送顺序，本模块主要针对骑手提交需要配送的订单信息后，通过调用模块内已封装完成的蚁群算法提供的运算接口，接收骑手提供的订单信息数据，通过运算最终将规划结果返回至骑手。

3.3.6 优化路径图示模块

本课题中，当路径规划模块返回订单配送顺序后，基于用户体验基础上，能够提供可视化方案，给予骑手直观感受，指导骑手根据规划完成的路径进行配送，本模块能够通过接收路径规划模块返回的数据，通过调用高德地图提供的接口方法，在指定的显示界面中创建地图模型，并将待配送的订单依据配送顺序进行路径绘制，将整个过程以图示化的方法直接呈现给骑手。

3.4 系统的 UML 建模

3.4.1 用例图

用例：使用 App 进行订单配送。

主要参与者：骑手。

次要参与者：系统管理员。

情境目标：在当前可使用城市范围内骑手进行外卖配送活动。

前提条件：骑手已经接收外卖订单并准备配送。

起动：骑手打开该 App 并决定开始配送。

场景：①骑手获取定位信息。②骑手接单成功同时提交订单信息。③骑手获取规划的路径数据。④骑手通过交互获取两点最短距离。⑤骑手完成配送后再次准备接单，需要清除历史规划的路径数据。⑥骑手查看当前订单。⑦管理员查看当前骑手待配送订单详情。⑧管理员查看当前骑手订单配送路径以及当前骑手位置。

使用方式：通过基于个人的 Android 4.0 版本以上的智能手机并成功安装该 App 应用。

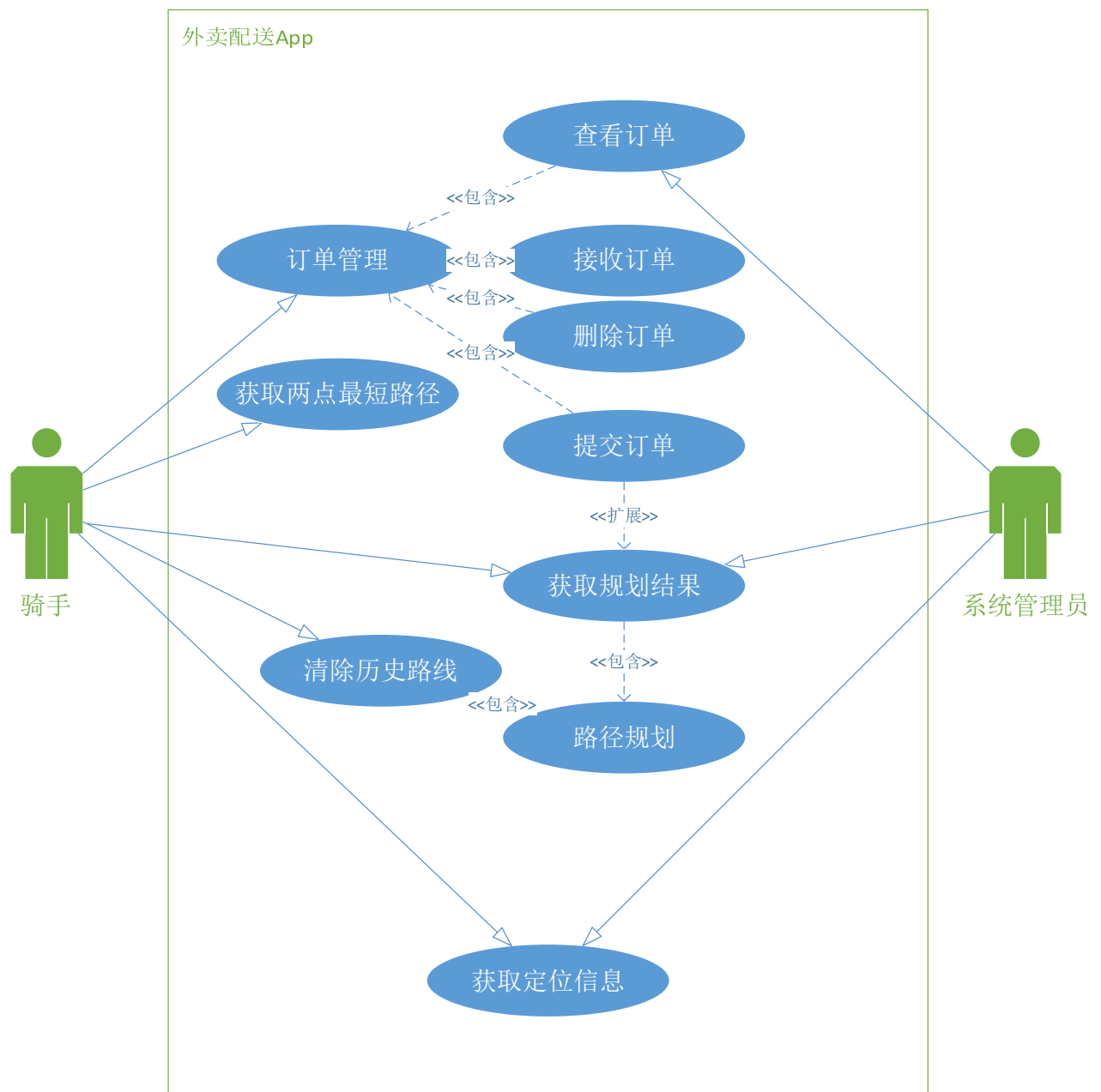


图 3-1 系统用例图

3.4.2 类图

(1) 类建模

- 骑手类 (Rider) :在外卖配送路线规划设计中, 骑手类代表本课题主要面向的服务用户, 课题成果需要识别骑手个人身份 (ID, Name), 考虑其他情况需要联系骑手, 因此系统后台需要能够记录每个骑手的手机号 (Tel)。每个骑手能够查看自身订单 (GetOrder()), 同时能够对自己的订单进行选择修改 (AddOrder(), DelOrder()), 确定自身需要配送订单之后, 骑手可以提交订单 (SubmitOrder()), 获取规划路线 (GetLines()), 再次接单前需要清除历史规划路线 (CleanLines()), 除以上动作以外, 骑手能够获取定位信息 (GetLocation()), 通过直接交互获取两点最短路径 (Interactive())。该类图如下所示:

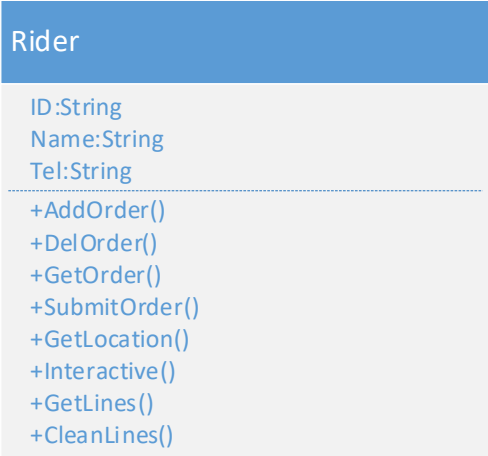


图 3-2 Rider 类

- 管理员类 (Manager) :管理员主要负责系统运维以及日常事务处理, 一个系统可以同时存在多个后台管理员, 系统能够识别每个管理员, 即每个管理员拥有身份识别号 (ID, Name)。管理员在后台能够获取单个骑手的订单详情 (GetOrder()) 以及该骑手的配送路径 (GetLines()), 同时还能够获取到对应骑手的实时位置数据 (GetRiderLocation())。该类图如下所示:

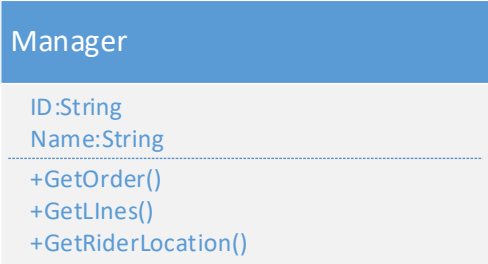


图 3-3 Manager 类

- 订单类 (Order) :该类作为本课题内针对订单管理, 每个订单在系统中都能被识别, 即每个订单都拥有唯一的序列号 (ID), 此外每个订单都拥有店铺取餐坐标 (BussAddress), 以及配送的顾客坐标 (CustAddress), 同时每个订单能

够提供基本操作（Add(), Del(), Sub()），该类图如下所示：



图 3-4 Order 类

- 地图类（Map）：系统中的每个地图视图都具有视图名称（Name），每个类具初始化时能够创建地图视图（Create()），在指定的坐标点上设置 Marker 标识（SetMarker()），通过给定相邻的 Marker 标识找出最短路线（OnRouteResearch()），绘制路线（DrawLine()），以及清除历史路线（Clear()），最后删除该图层（Delete()）。该类图如下：

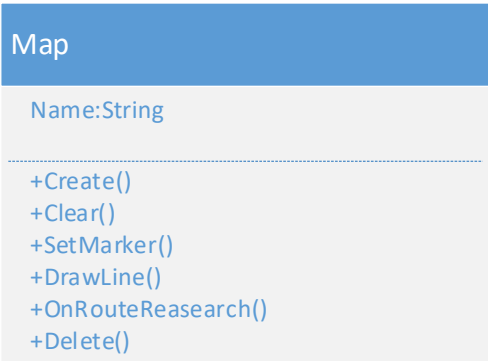


图 3-5 Map 类

- 监听器类（Listener）：该类主要用于监听整个项目中特定控件的变化情况，实时进行处理，每个监听器的用途不同，拥有不同的名称（ID），在监听内容改变时需要调用相应的方法进行处理（OnData()）。该类建模如下：

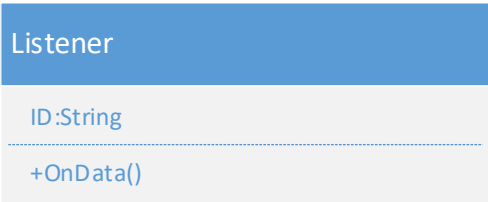


图 3-6 Listener 类

（2）类关系模型图

在本课题中，各类均需要通过监听器进行控件监听，与监听器类构成依赖关系；其次在本课题中，骑手需要根据当前订单进行配送，与订单形成直接关联关系；Matlab 封装的蚁群算法程序包在订单提交时调用接口，订单类为该包提供接口实现；管理员可以查看单个骑手的订单信息，与订单类构成关联关系，此外还可以查看骑手的位置信息，与地图类构成关联关系；订单数据通过直接传送到地图类中，与地图类构成直接关联关

系；具体类模型图如下图所示：

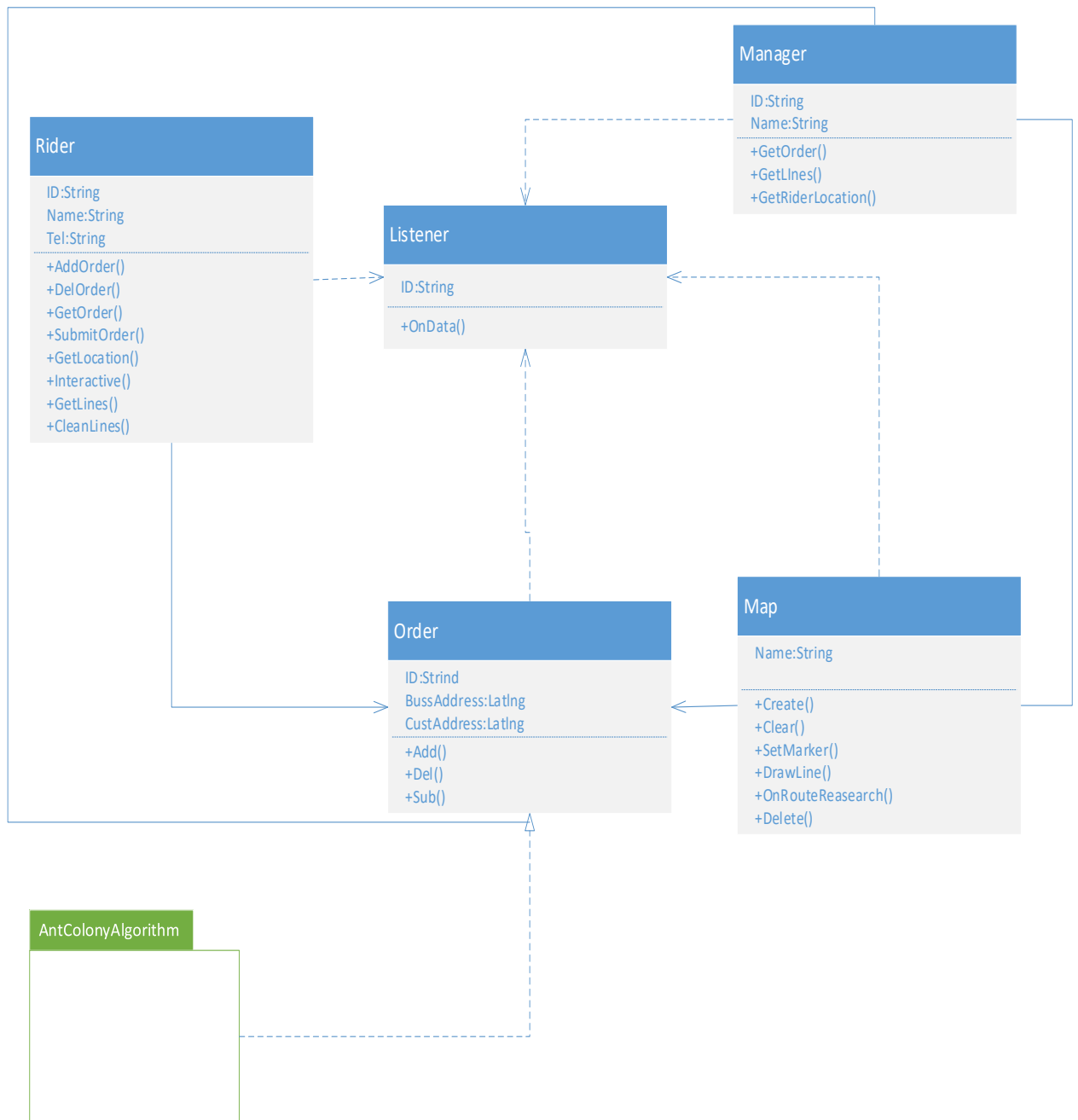


图 3-7 类关系模型图

3.4.3 时序图

以骑手为对象，骑手可以通过添加订单方式，将订单坐标提交至 app 中的数据模块，另外，在多个订单的情况下，骑手通过点击提交订单按钮可以将数据传送至路线模块，再由地图模块传送至服务端处理，服务端经过处理后将结果返客户端中的数据模块，客户端通过数据模块在地图模块上按照服务器返回结果，调用第三方接口（高德地图）进行路线绘制。

(1) 添加订单时序图：

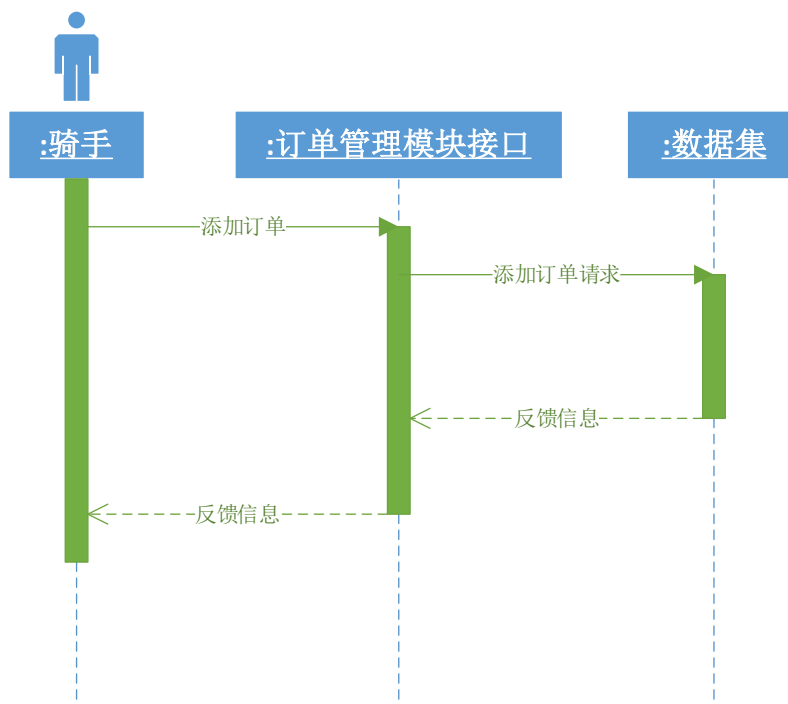


图 3-8 添加订单时序图

(2) 删除订单时序图

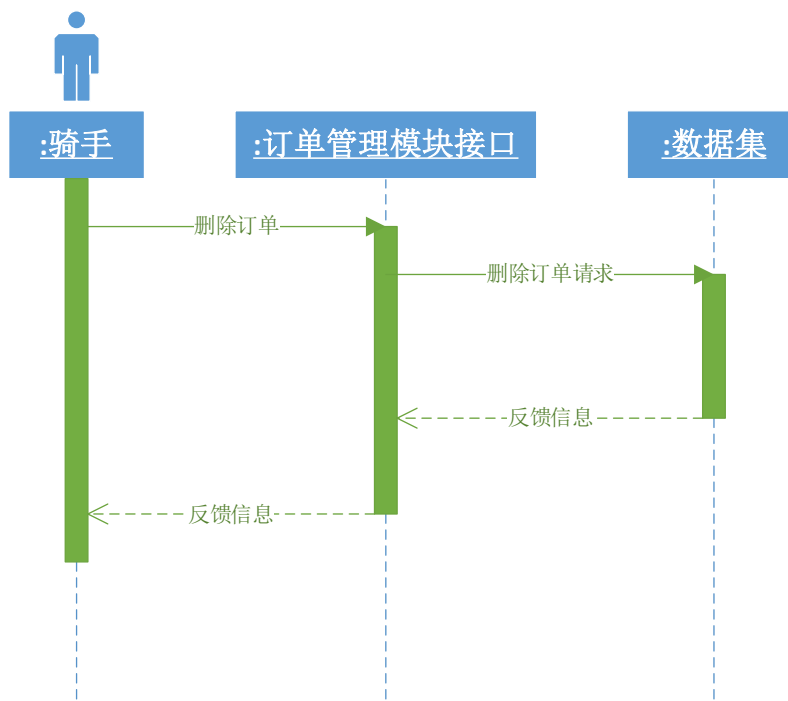


图 3-9 删除订单时序图

(3) 提交订单时序图

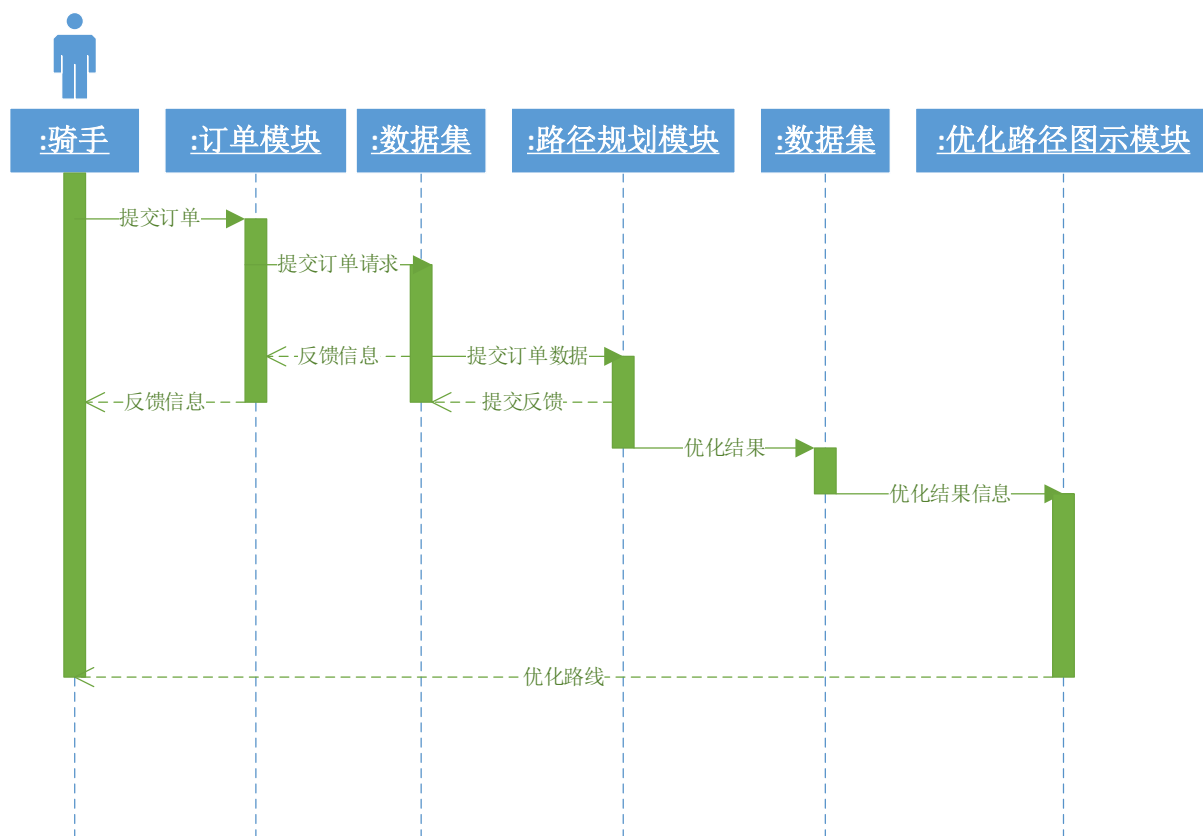


图 3-10 提交订单时序图

(4) 清除路线时序图

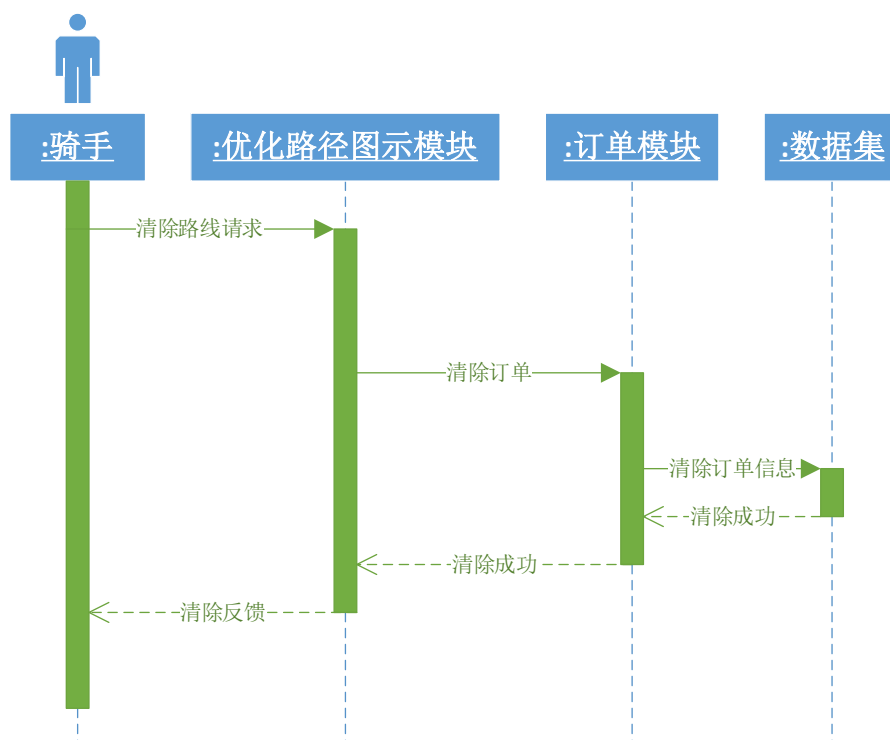


图 3-11 清除路线时序图

(5) 定位时序图

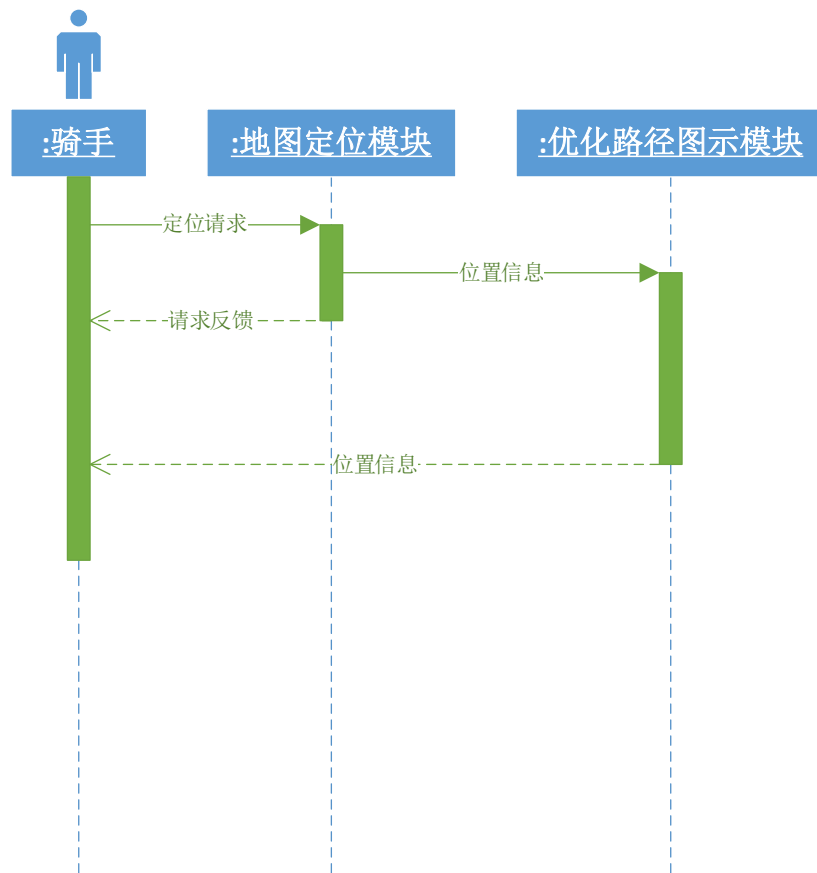


图 3-12 定位时序图

(6) 界面交互时序图

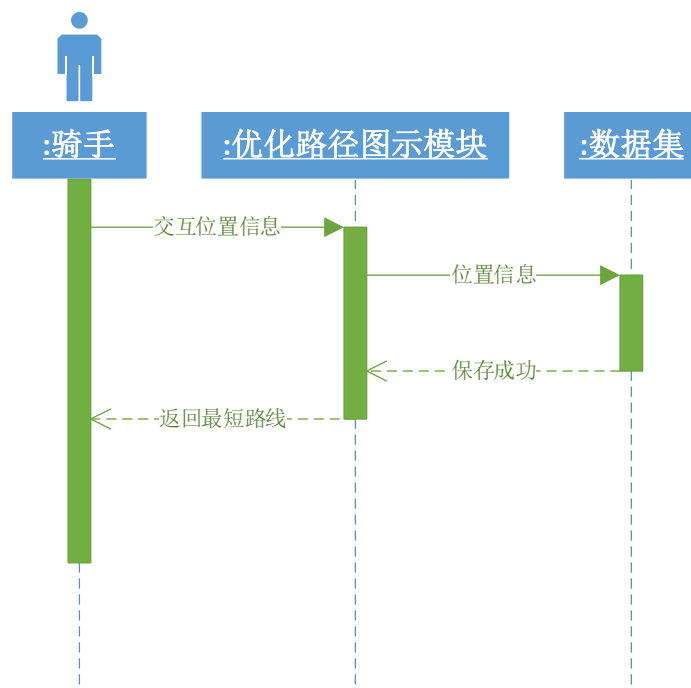


图 3-13 界面交互时序图

3.5 系统性能需求

性能需求分析作为整个设计环节中重要的一个组成部分。它包含了整个项目（课题）一些重要特性。需要从用户使用的角度出发，理解用户的需求和使用习惯，在最大程度上考虑用户的利益，增强用户对产品的信任感，以下几个指标在整个项目开发过程中能够很好地照顾用户感受，即安全性，易用性，可靠性等几大特点是开发过程中需要着重兼顾的方面。

在安全性方面，骑手用户在接收到订单过后，需要根据订单内容进行取餐与送餐，处于对骑手的保护以及对客户隐私的尊重，相应的订单内容不能外泄，否则容易造成不良影响，甚至是财产损失和人身安全，该项指标能够对骑手以及客户的安全有所保障。

考虑易用性方面的需求，由于外卖骑手的文化水平或者理解能力有限，因此在整个设计过程中，外卖骑手通过对订单的提交获得规划后的最佳路径，整个过程不能过于复杂，否则不仅会影响骑手的使用体验，还可能会对骑手的配送时间造成延误，另外考虑到外卖骑手在短距离内的暂时性需求，可以通过直接交互来获取两点之间的最佳路径，比较符合一般使用习惯。

在可靠性方面，最好不需要骑手对重要数据进行操作和干预，数据传输过程会涉及到专业领域知识，应在后台进行，除了个别指示作用控件，整个处理对骑手来说最好完全透明。

3.6 系统数据流图

3.6.1 顶层数据流图

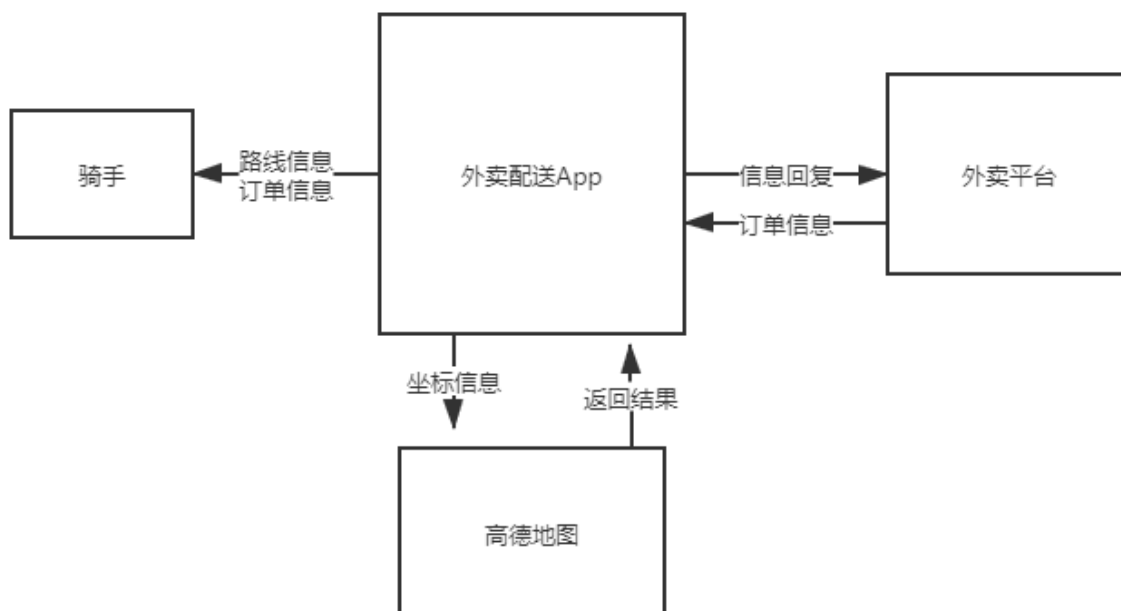


图 3-14 顶层数据流图

3.6.2 一层数据流图

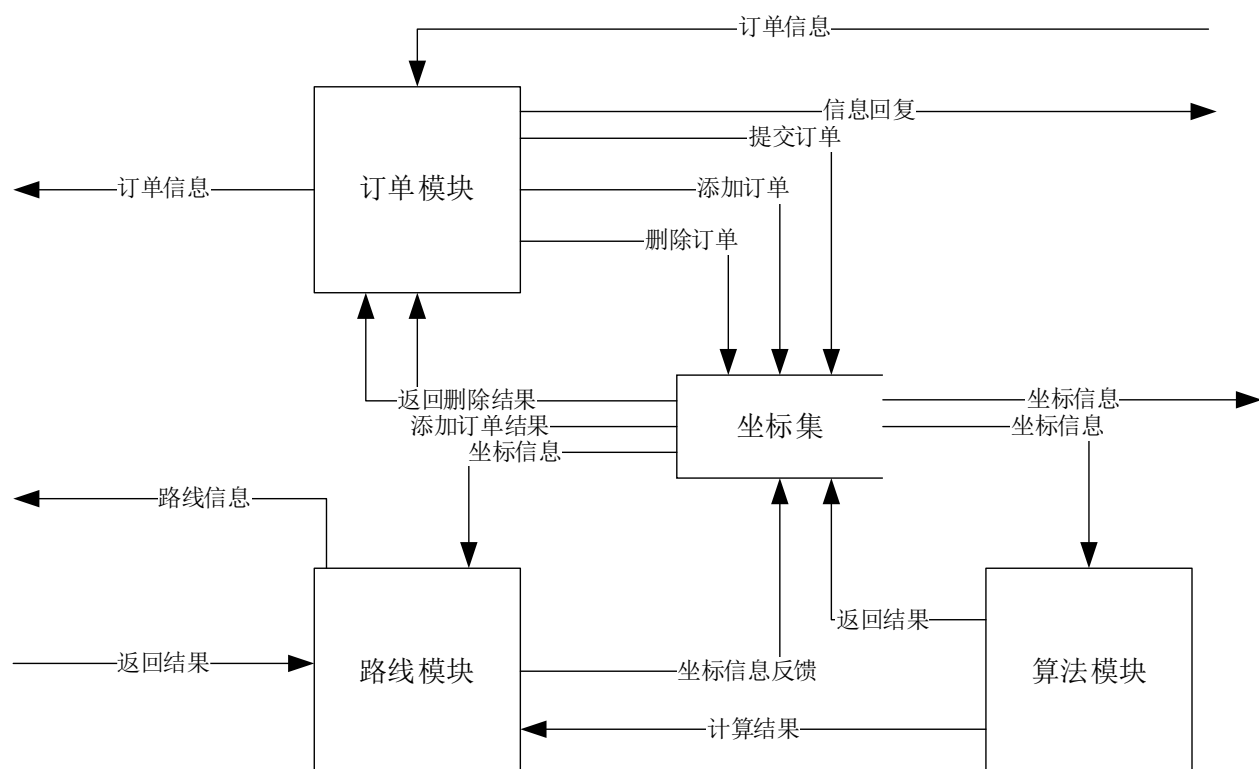


图 3-15 一层数据流图

3.6.3 二层数据流图

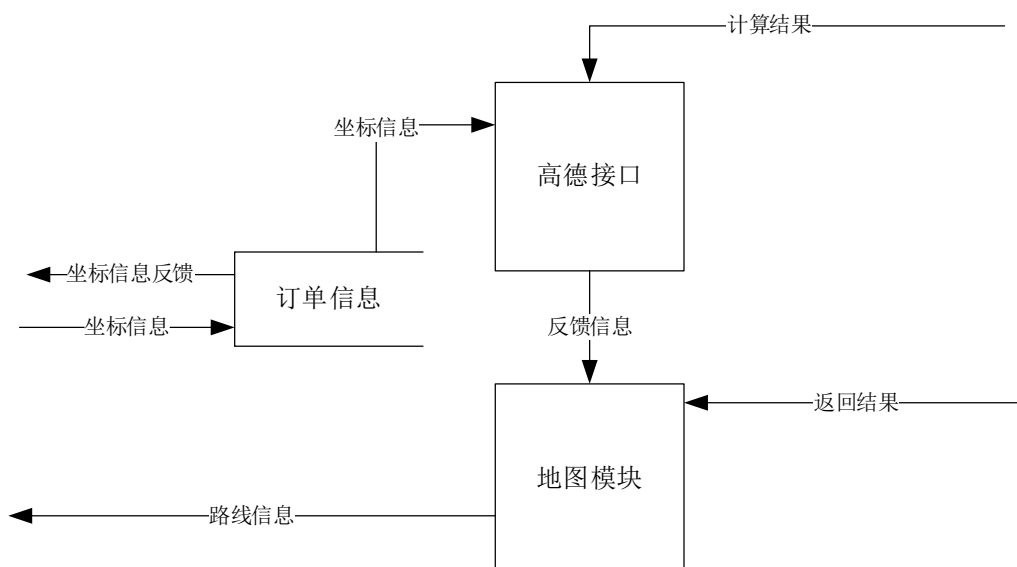


图 3-16 路线模块数据流图

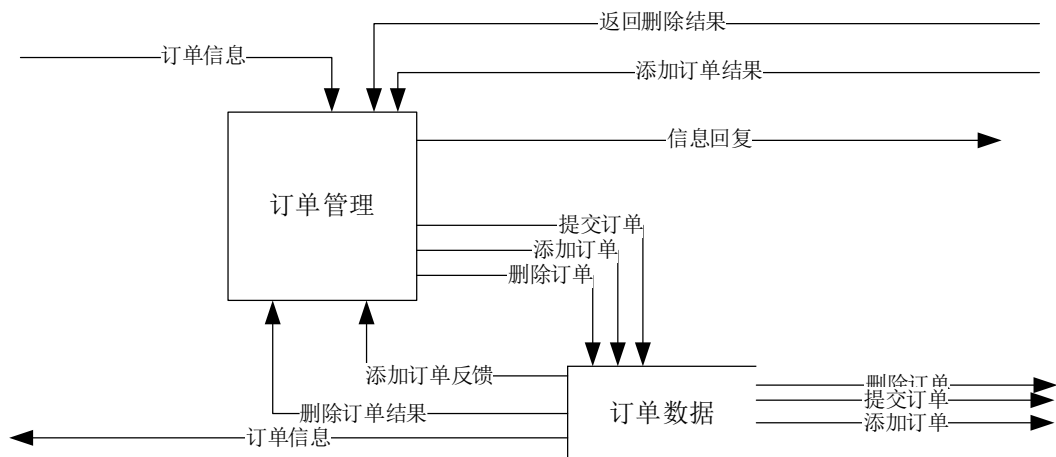


图 3-17 订单模块数据流图

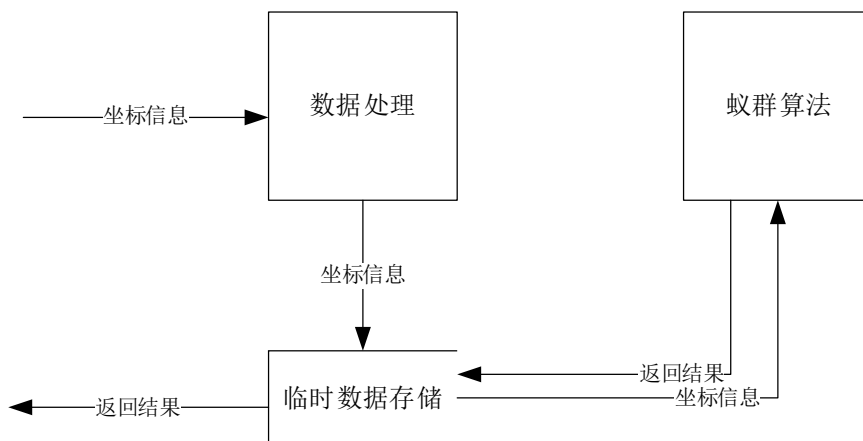


图 3-18 算法模块数据流图

3.7 加工说明

(1) 订单管理：负责接收外界传输的订单数据并针对该数据请求进行分析处理，同时按照相应请求加相应的数据传输至下一个模块。并且接受相应模块返回的信息调用内部逻辑结构对外界进行反映。

(2) 订单数据：在相应模块内负责临时存储订单信息的一块存储区域或者存储文件，接受其他模块传输过来的操作命令，对存储体内的数据进行增删查改相应操作，并将相应的结果通过映射或传输返回至需要模块。

(3) 高德接口：用于负责接收从其他模块传输过来的数据并提交至高德地图服务。

(4) 地图模块：属于地图的高德服务接口中的一个图层显示，根据其他接口返回过来的数据在已建立了的图层上绘制路线或者作出 mark 标记。

(5) 数据处理：接收其他模块传输过来的坐标数据并处理成符合蚁群算法接口的形式。

3.8 数据字典

在本课题中，主要利用订单数据来进行路线规划，仅取目标坐标数据为订单信息，则对应订单的数据集的数据字典如下表所示：

表 3-1 数据字典

数据名称	数据类型	数据宽度
ID	String	16bit
地址	String	16bit
纬度	Double	32bit
经度	Double	32bit
店铺名称	String	16bit
联系电话	String	16bit
品类	String	16bit
备注	String	16bit

第四章 详细设计

4.1 概述

在本次课题的研究中，针对软件的开发，在整体上采用自顶向下，模块化设计。从系统功能图开始，将各功能模块逐步细化在一定程度上做到高内聚低耦合，将整个开发过程实现可追踪化，各模块功能尽可能实现分工明确，责任分明真正做到软件工程思想在实际开发中的学以致用。

4.2 系统功能模块图

本课题的开发设计，主要分为数据集的收集，app 的设计与实现与算法的导入三个部分，其中 APP 负责基础服务部分包括订单的添加删除以及提交，路线模块的清除和定位以及地图路线显示，算法通过事先预定的接口接收从 App 传递过来的数据进行计算处理并返回至 App 显示调用。算法模块的主要功能是调用基于已封装完成的算法提供路径规划，在输入商家与顾客位置信息之后得到最优遍历序列。其中软件主功能模块图如下所示：

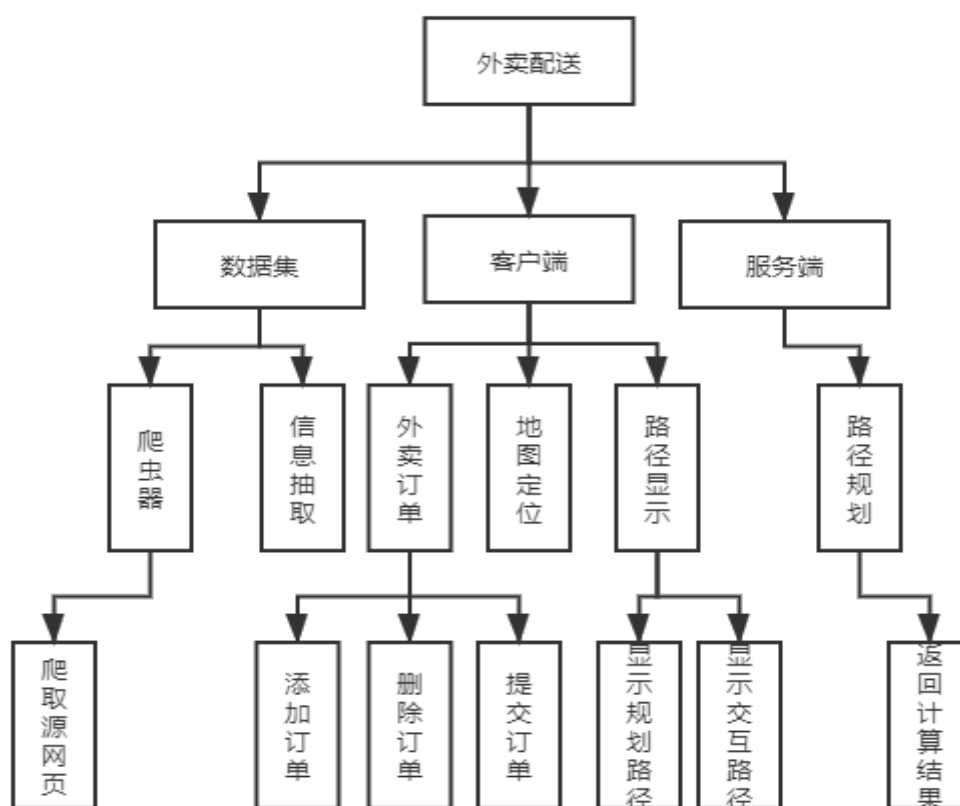


图 4-1 主功能模块图

4.2.1 网页爬取模块

本模块的设计主要用于针对后期软件开发中的数据部分，在利用 Python 3.5 的基础上使用爬虫技术对饿了么官网进行商家坐标和顾客坐标等主要信息的数据爬取，通过代码的形式将程序伪装成浏览器向 <https://h5.ele.me/> 进行发送数据请求，本段代码完成后收到服务器传送过来的 json 数据，利用 json.load() 方法将其转化为字典完成信息爬取工作，以下为该爬虫部份的步骤流程图。

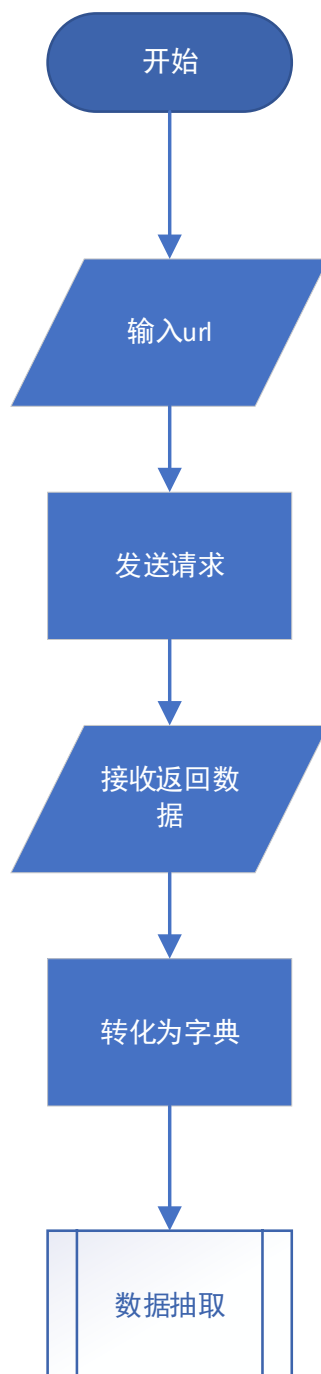


图 4-2 爬虫模块流程图

4.2.2 网页信息抽取模块

在之前爬虫器发出请求的基础上，通过对服务器的响应数据进行接收，调用相关的库提供的方法将该数据进行格式化处理，将其转化成本课题需要的模式，例如调用 `json.loads()` 将其转化成字典，接下来再通过对应的关键字来进行数据匹配，将整个返回信息中的关键数据进行抽取，如 ID，城市，纬度，经度，店铺名称，月销量等信息匹配保存，之后写入到本地文件中。具体流程图如下：

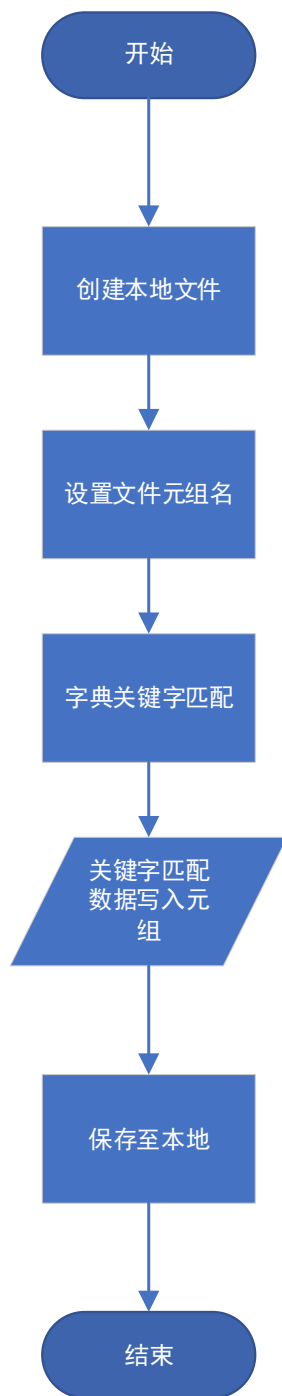


图 4-3 网页信息抽取流程图

4.2.3 外卖订单模块

订单模块作为软件 APP 的一部分，在整个开发过程中从外部获取订单数据，同时将数据传输至数据集中，通过订单栏显示出具体的订单，此外还可以通过对订单的操作进行管理，例如添加订单，删除订单以及提交订单，具体的步骤流程图如下图 4-4 所示：

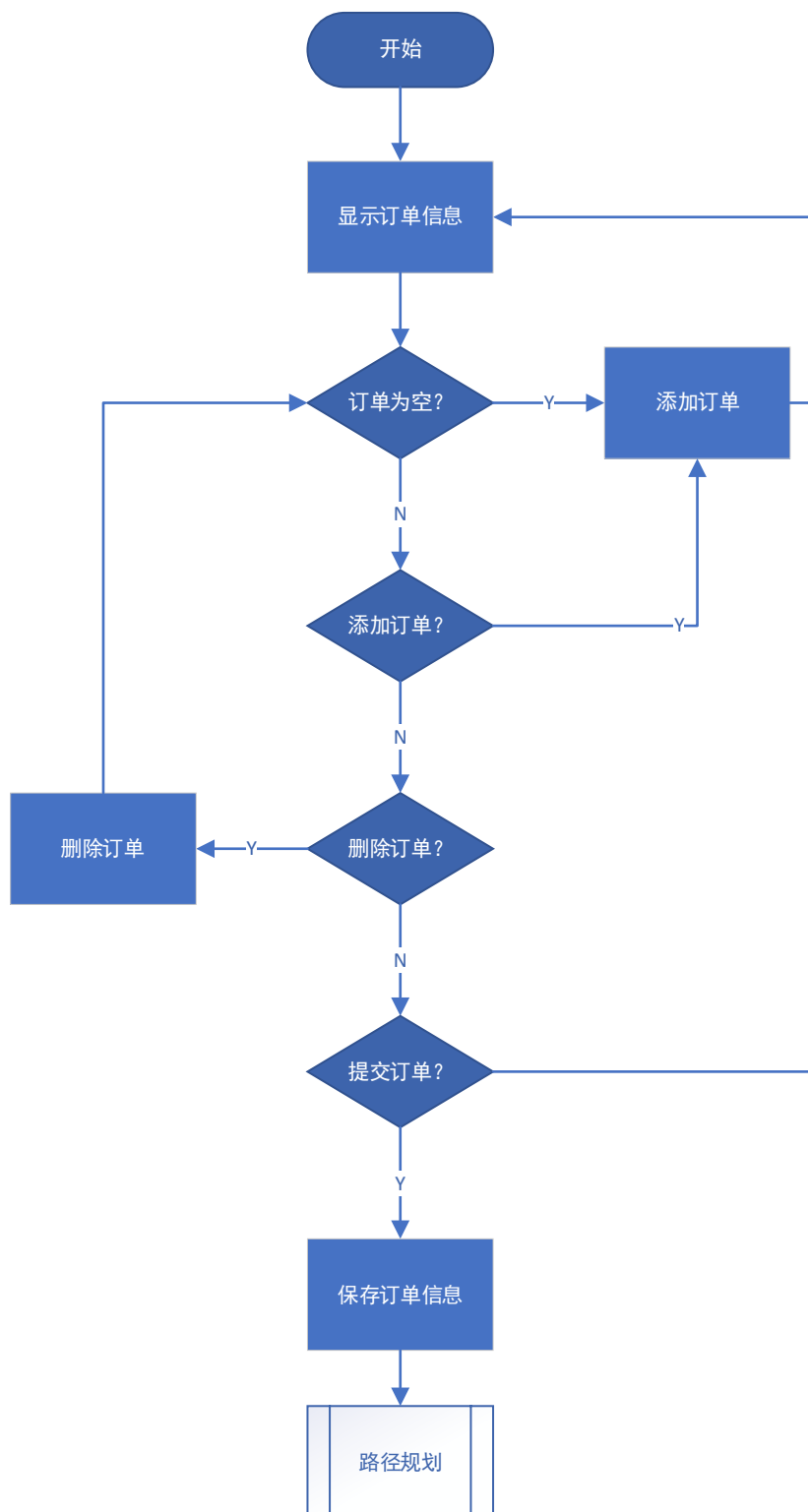


图 4-4 订单模块

4.2.4 地图定位模块

本模块在调用网络服务的前提下需要调用高德地图提供的定位方法，通过创建地图图层，监听自身位置的变化不断触监听器将自身坐标数据传输至本地，并显示在地图上。具体步骤流程图如图 4-5 所示：

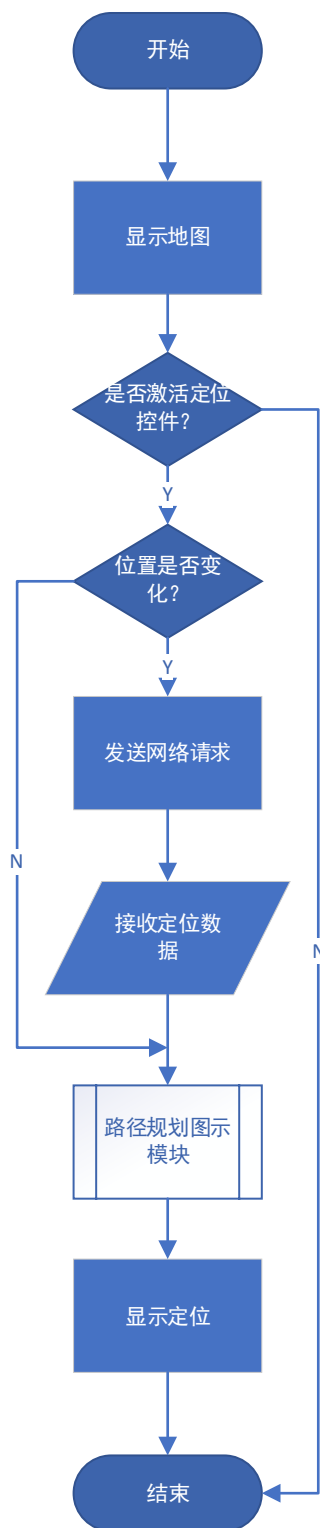


图 4-5 定位模块

4.2.5 路径规划模块

该模块是整个软件进行路径规划的核心，通过接收外部的坐标数据，利用模块内封装好的蚁群算法提供的封装接口参数进行运算，并将得到的返回结果传送至需要的模块，即优化路径图示模块。具体步骤流程如图 4-5 所示：

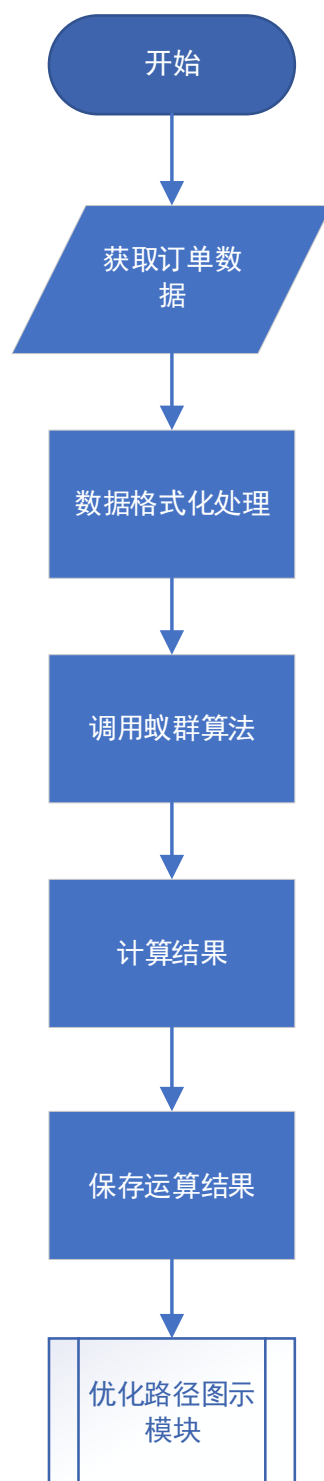


图 4-6 路径规划模块

4.2.6 优化路径图示模块

本模块主要用于显示通过接口调用蚁群算法返回结果坐标，主要将坐标列表中的数据在地图图层中进行显示出来并进行标志，同时还要调用第三方--高德地图提供的接口进行最近路线绘制，将返回结果中的所有坐标点之间连成一条最短路径。具体流程如图4-6所示：

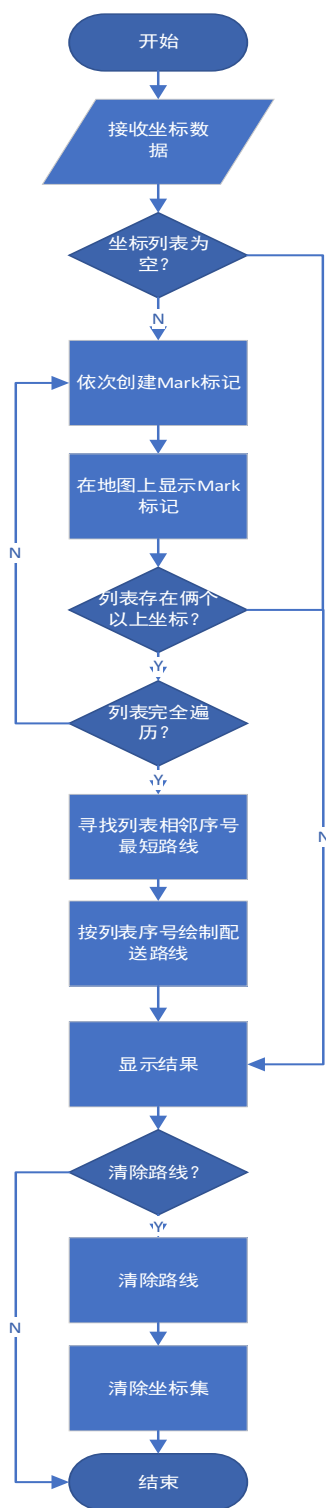


图 4-7 优化路径图示模块

4.3 系统流程图

在整个软件启动后，系统将进行初始化操作，将在路线界面创建地图图层，通过接口调用返回定位数据信息，实现当前情况下骑手的定位显示；然后通过订单界面的初始化，方便骑手基于订单的管理进行灵活操作，同时骑手提交订单数据后，该数据将会被保存至相应的数据集，然后发送至路径规划模块中，通过蚁群算法的接口调用实现最优路径规划，将返回结果发送至对应的数据结构中，该数据将会被优化路径图是模块处理并在地图上绘制出最优路径路线；此外在骑手的配送过程中，定位模块将不断的进行监听以实现骑手的实时位置。

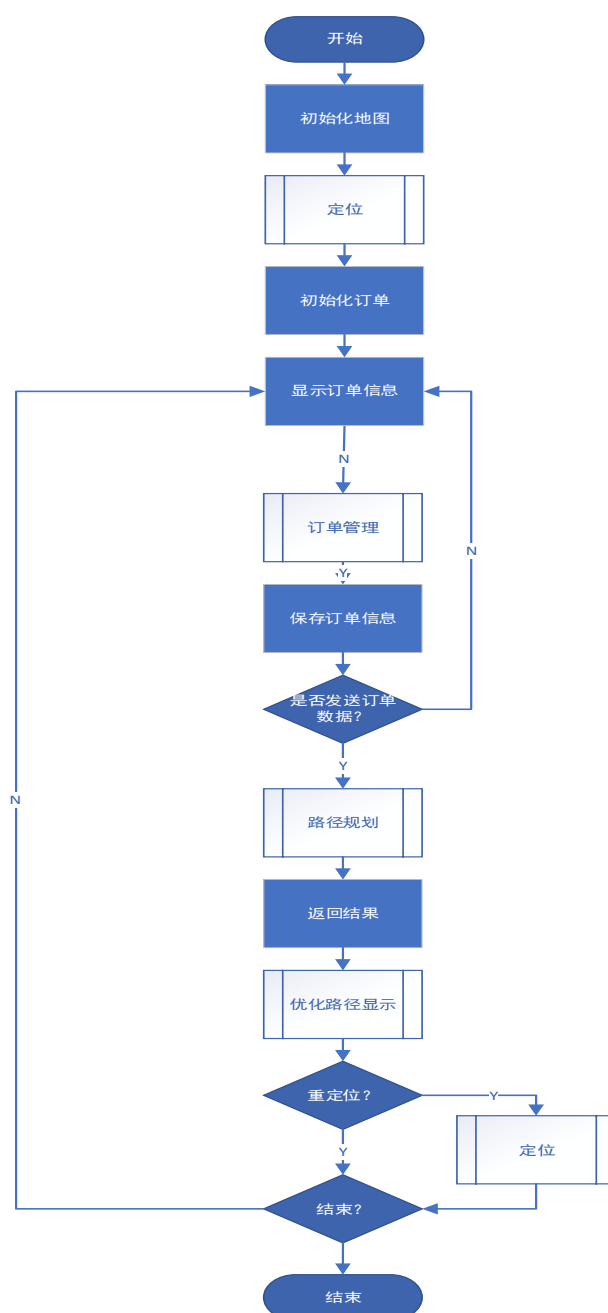


图 4-8 系统流程图

第五章 外卖路径优化问题建模及算法实现

5.1 外卖路径优化问题的数学模型

本节详情参照 2.1.1 小节内容。

5.2 算法开发环境

本课题中利用的算法模型主要建立于 Matlab 工具基础上进行编码，在整个过程中不仅能够观察数据的详细变化。此外，因为算法的初始规模中仅有商家坐标，只有蚂蚁到达对应商家坐标后，未遍历坐标集将会动态添加该商家坐标对应的顾客坐标，该算法动态拓展的特性使得 Matlab 成为首选开发工具，Matlab 还具备代码量少，简便，运行快速等优点，能够在一定程度上降低开发人员工作量。

5.3 算法详细设计

5.3.1 参数说明

参照表 2-1 所示。

5.3.2 算法流程图

如图 5-1 所示。

5.3.3 算法分析

本课题中所采用大的遗传算法是一种启发式算法，该算法能够在具体的环境应用中呈现可观的效果，传统的蚁群算法在实现方面根据相关文献^[13-15]表述其时间复杂度为 $n*(n-1)2*m*T/2$ ，空间复杂度为 $n*n+n*m$ 。

本课题中在面对实际情况时对算法进行适应性的改造，在一定程度上增加了该算法的复杂性，同时由于该算法对于计算结果的收敛性要求过高，因此对计算机的性能方面有一定的要求，改造后的算法在原有的基础上没有改变时间复杂性，但对于空间复杂性做出了一定的妥协，由于涉及到具体的商家和骑手配送的实际情况，每只蚂蚁都具有不同的遍历过程，需要针对每个蚂蚁的遍历路径做出记录，因此空间复杂度为 $n*n*m$ 。

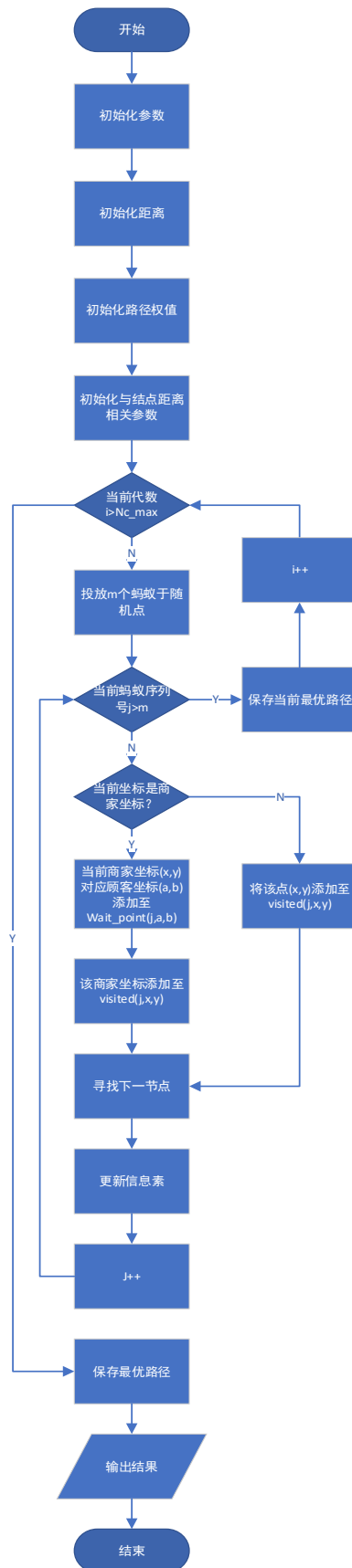
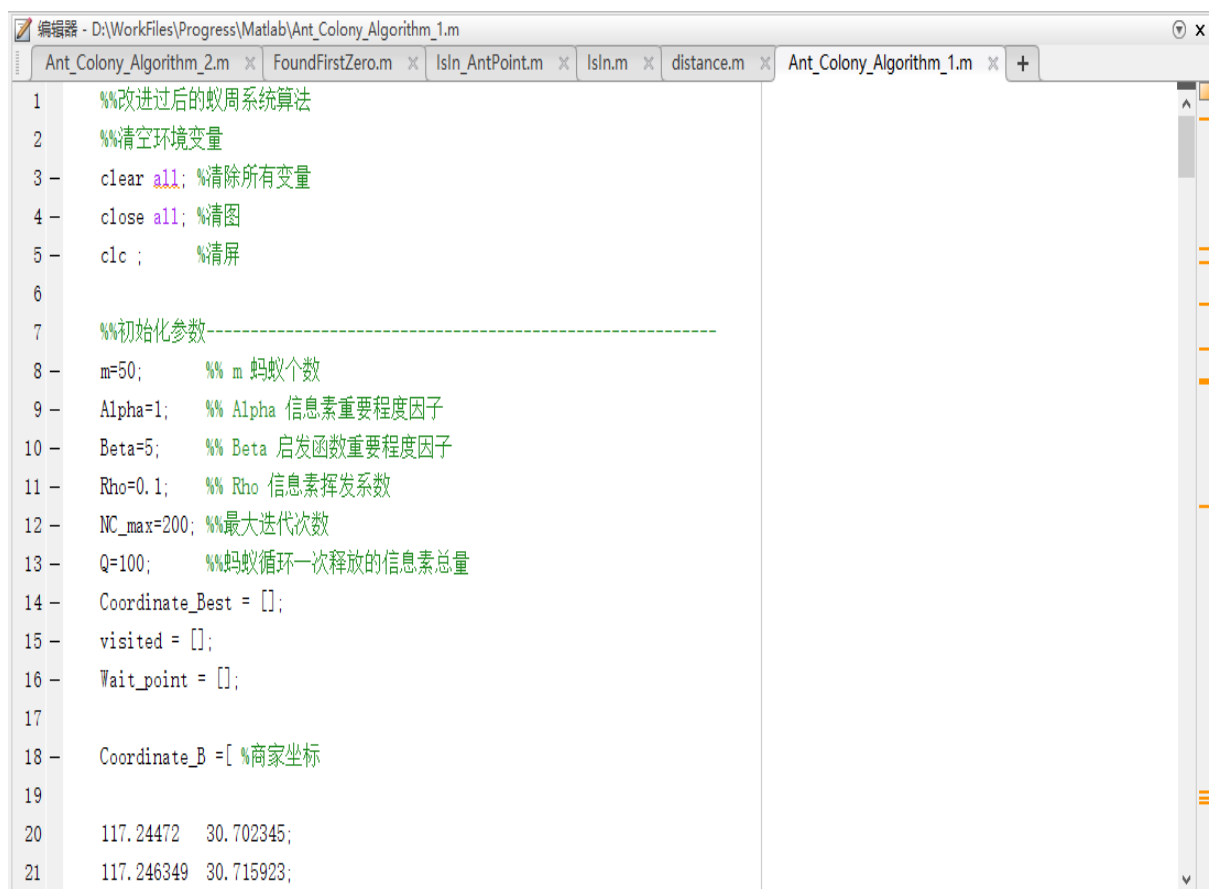


图 5-1 算法流程图

5.4 算法的打包封装

在上述完成算法的前提下按照 MATLAB 在实际编程中的应用将整个过程封装成一个函数，并且确定返回结果以及输入的参数，即商家地址与顾客地址。具体封装截图如下：



```
1 %%改进过后的蚁周系统算法
2 %%清空环境变量
3 clear all; %清除所有变量
4 close all; %清图
5 clc; %清屏
6
7 %%初始化参数-----
8 m=50; %% m 蚂蚁个数
9 Alpha=1; %% Alpha 信息素重要程度因子
10 Beta=5; %% Beta 启发函数重要程度因子
11 Rho=0.1; %% Rho 信息素挥发系数
12 NC_max=200; %%最大迭代次数
13 Q=100; %%蚂蚁循环一次释放的信息素总量
14 Coordinate_Best = [];
15 visited = [];
16 Wait_point = [];
17
18 Coordinate_B=[ %商家坐标
19
20 117.24472 30.702345;
21 117.246349 30.715923;
```

图 5-2 算法测试程序

清除测试程序程序中的部分数据，将整个程序设置成一个功能函数，其中需要的程序功能需要链接到本文件中另外在函数头中设置好输入参数以及输出参数包括整个函数名，如下图所示，第一个参数 Short_Route 为输出参数，代表最短路径的序列，等号后面是函数名，其中括号中的两个参数 Business, Cuserm 分别为商家坐标与顾客坐标。

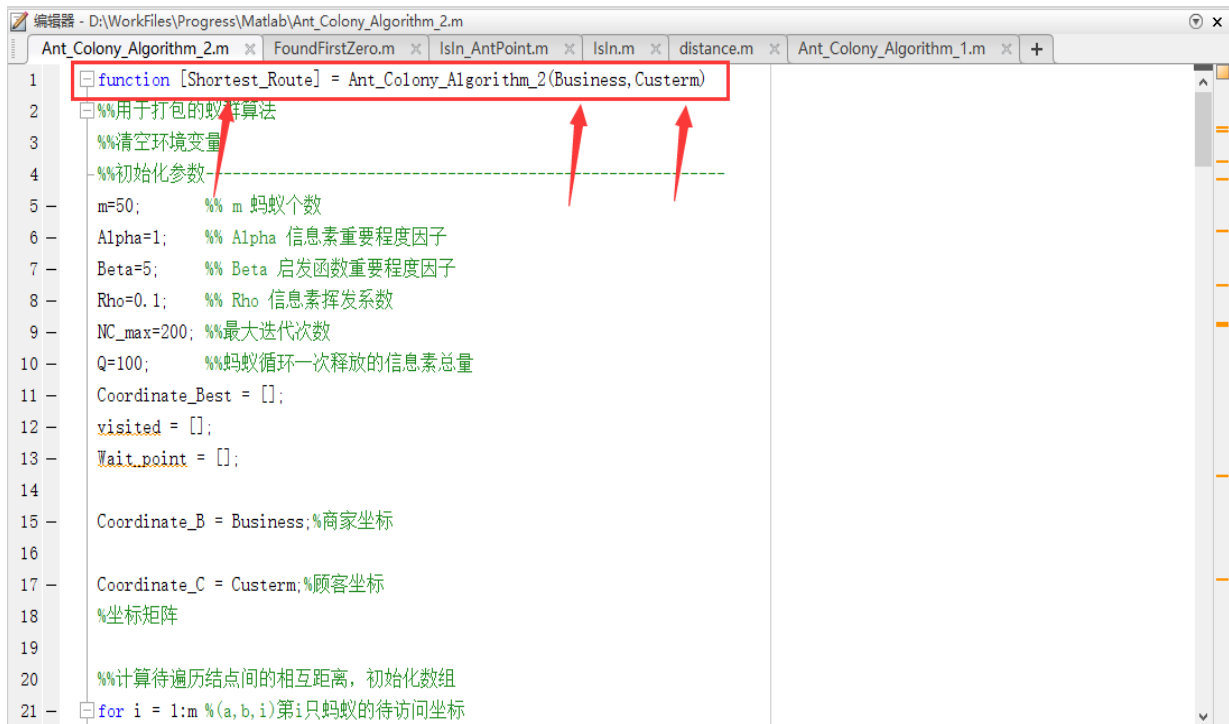
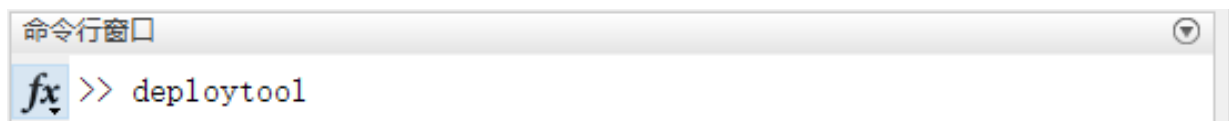


图 5-3 设置接口参数

在命令行窗口输入命令 deploytool 打包成 jar 文件，供 Java 程序调用



打包命令

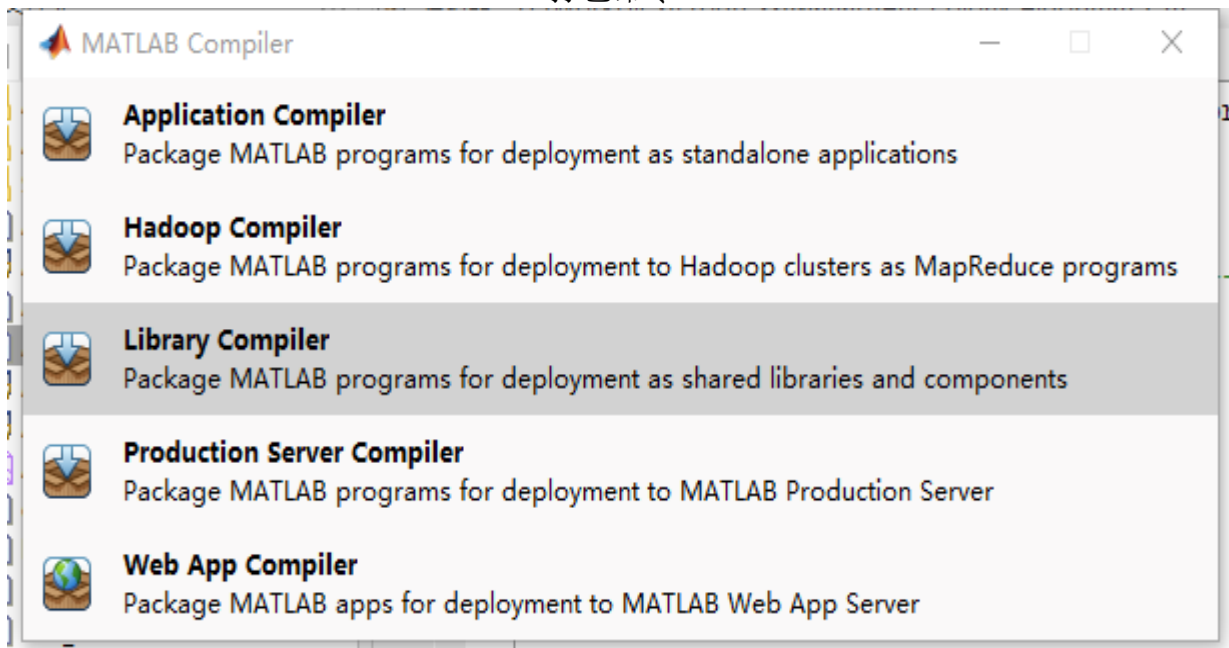


图 5-4 打包类型

根据下图箭头所指示的位置在第一个框中选择打包的类型，此处选择 Java 包，第

二个框中选择的是打包的文件利用旁边的加号进行添加，截图下面的箭头指示的地方表示打包过后该类的名字属性，可利用该类的名字来选择调用。最后点击上方最后一个箭头所指示的勾，完成打包过程。

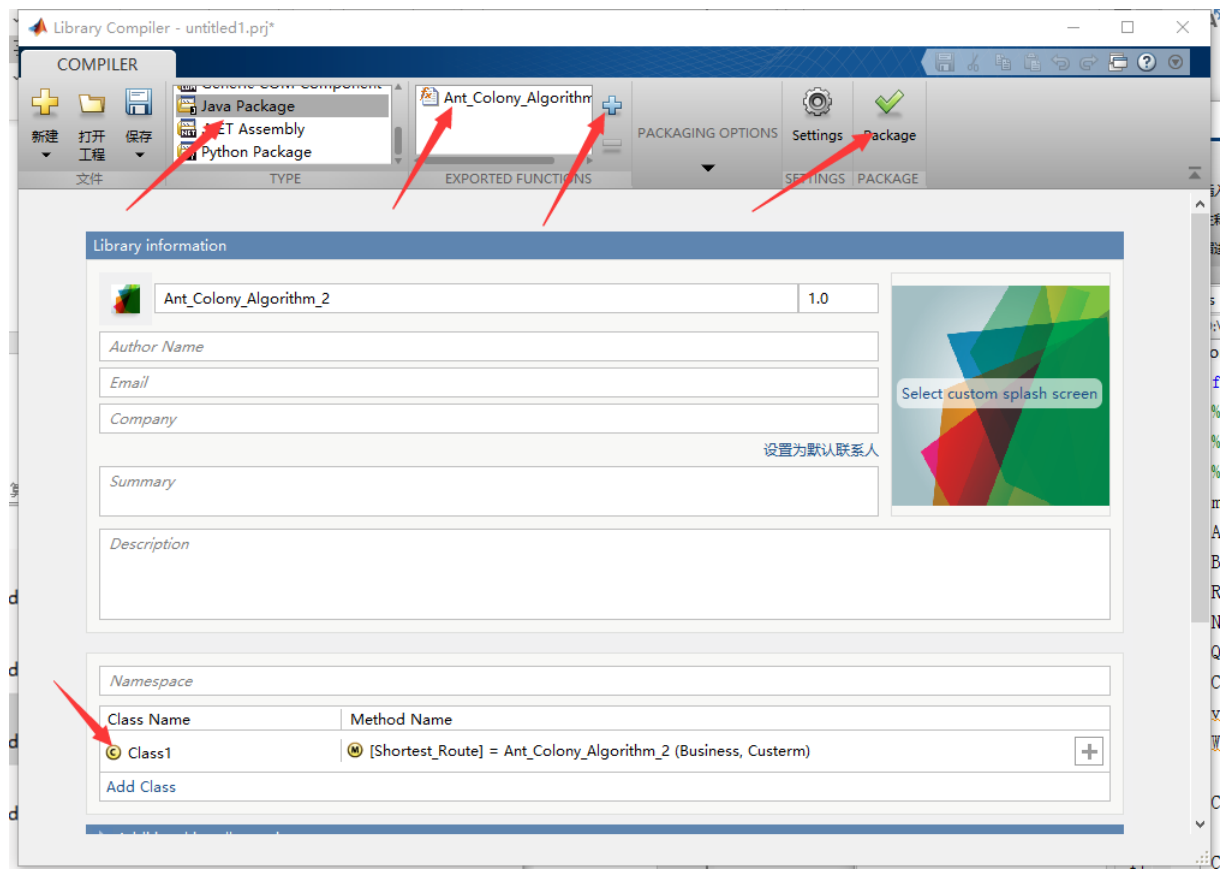


图 5-5 打包界面

第六章 系统测试

6.1 编写目的

通过对整个软件系统进行测试，验证该系统的功能是否正确无误，确保骑手在整个配送过程中行驶路线的正确性。

6.2 背景

(1) 路线规划模块：基于 pc 机中 eclipse 的搭建 Java 环境服务端，接收 app 的数据传输，并能对 App 进行响应，通过 Socket 套接字返回结果。

(2) 其他模块：基于 Android 4.0 以上的操作系统，本次课题中以本人 MI 6 为例，进行 app 的测试。

6.3 测试方法

黑盒测试：将模块打包安装，在面向用户透明的情况下，提示用户进行操作来测试 app 的运行正确性。

6.4 测试概要（或测试过程）

6.4.1 网页爬虫模块测试

测试功能：网页爬取功能。

预期结果：网页数据爬取后在本地副本保存成功

测试过程：打开路径“C:\Users\98763\Desktop\汪琰-数据集”下的爬虫.py 文件，运行后，打开该文件路径，创建生成“data.csv”文件，如下图所示：

```
In [2]: runfile('C:/Users/98763/Desktop/汪琰-数据集/爬虫.py',  
汪琰-数据集')
```

```
按y继续爬取，按任意键结束：y  
正在爬取第1页,请稍后...  
爬取完成
```

图 6-1 爬取截图



图 6-2 本地保存截图

6.4.1 数据抽取模块测试

测试功能：数据抽取

预期结果：本地文件中元组对应数据符合要求，即对应数据成功匹配并写入本地文件中。

测试过程：打开运行生成文件，查看对应数据是否符合要求，截图如下：

ID	城市	纬度	经度	店铺名称	月销	起送金额	最低配送	地址	评价数量	联系电话
E45395066	北京	39.897619	116.33621	俺闺蜜凉皮	2032	5.5	20	北京市海淀	1089	1591038544
E58828796	北京	39.897987	116.3361	四筒炒饭	2680	4	20	北京市海淀	672	1731921154
E22704115	北京	40.036383	116.35529	权金城·烤	1800	1	20	北京市海淀	5709	010-62994
E15654915	北京	39.984408	116.30438	三秦味道	1245	1.5	0	北京市海淀	733	1861274554
E12909233	北京	39.897987	116.3361	安徽板面	353	6	20	北京市海淀	164	1735695854
E24563185	北京	39.897822	116.33606	小谷姐姐席	1227	4	20	北京市海淀	2301	1369928024
E14977891	北京	39.89801	116.33606	鱼你在一起	1828	7.5	20	北京市海淀	930	1831055814
E16227712	北京	39.926853	116.33187	鼓油街(甘	567	7	20	北京市海淀	401	1513659234
E45277245	北京	39.897987	116.3361	丛丛小饼	756	5	20	北京市海淀	354	1761092564
E14855645	北京	39.899918	116.32205	龙门花甲	417	9	20	北京市海淀	268	1773361664

图 6-3 部分数据抽取结果

6.4.2 路径规划模块测试

测试功能：路径规划功能

预期结果：通过接收订单数据进行路径规划并返回结果

测试过程：本例中绑定端口号为 12500，初始化截图如下：

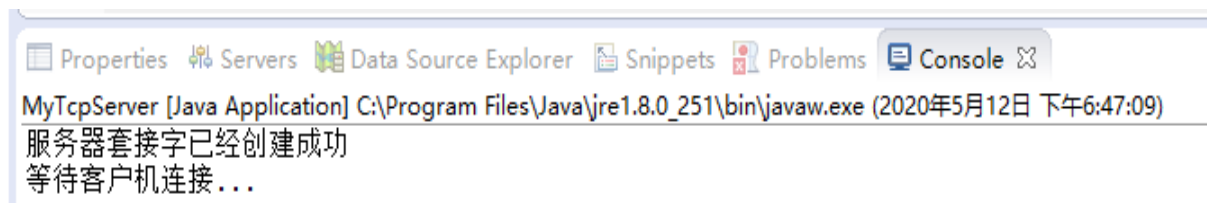


图 5-4 服务端初始化

算法模块通过 Socket 连接客户端并接受来自客户端的数据：

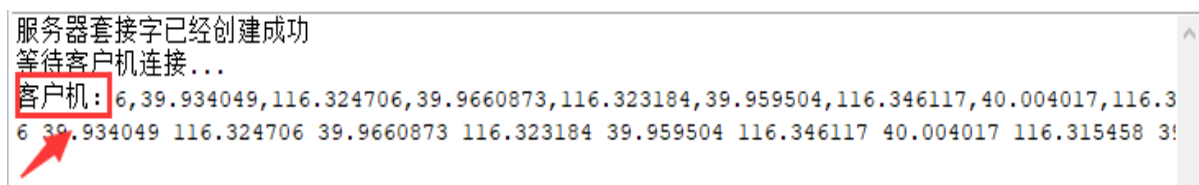


图 6-5 服务端建立连接并接收数据

算法模块通过提交订单发送过来的数据进行处理，并调用已经封装好的 Matlab 程序进行运算，将算法得到的运算结果序列返回至客户端。

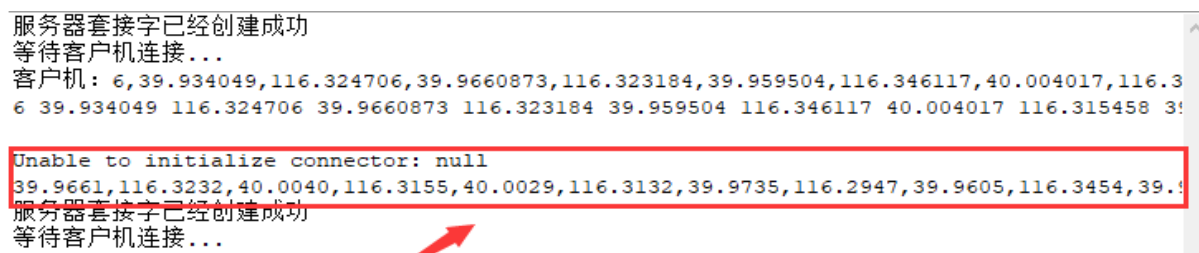


图 6-6 服务端处理数据

服务端在处理完当前结果之后，主动断开 Tcp 连接，并等待客户端新的连接以及请

求。

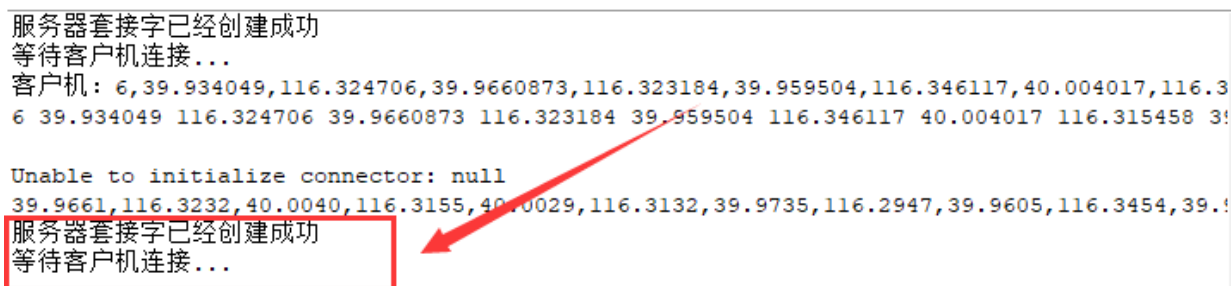


图 6-7 服务端等待新的连接请求

6.4.3 优化路径图示模块及定位模块测试

测试功能：路径图示显示，交互功能，重置功能，界面切换功能，定位功能

预期结果：地图图层在直接交互后能显示两点最短行驶路线，点击重置按钮能够清除图层。

测试过程：打开客户端程序，返回骑手当前坐标截图如下：

```
D/MyLocation: (30.877427,117.451707)
I/le.fooddeliver: ProcessProfilingInfo new_methods=2438
```

图 6-8 骑手坐标

以本人具体所在地点为例，客户端初始化界面截图如下：



图 6-9 初始化界面

交互功能测试：初始化界面通过直接交互绘制两点之间最短路径



图 6-10 两点之间最短路径

后台返回规划结果以及状态截图如下：

```
D/LocationList1: (30.884792564775985, 117.45044522226186)
I/LatLonPoint: (30.884792564775985, 117.45044522226186)
I/Point number: 2
D/LocationList2: (30.85250217892257, 117.45785214232977)
```

图 6-11 后台反馈坐标以及坐标数量

当添加多个点时，交互界面会依次绘制两点直接之间的最短路径测试截图如下：



图 6-12 三个坐标点



图 6-13 四个坐标点

后台返回数据分别为新添加的坐标点，截图如下：


```
I/LatLonPoint: (30.884792564775985,117.45044522226186)
I/LatLonPoint: (30.85250217892257,117.45785214232977)
I/Point number: 3
D/LocationList3: (30.86829245574773,117.47502766746281)
```

图 6-14 第三个坐标点以及坐标集长度

```
I/LatLonPoint: (30.885632745430208,117.45010189951383)
I/LatLonPoint: (30.85225234102106,117.45782800244906)
I/LatLonPoint: (30.86906946916284,117.47445769805691)
I/Point number: 4
D/LocationList4: (30.854584903660562,117.47221537135881)
```

图 6-15 第四个坐标点以及坐标集长度

测试清除当前路线以及重新定位功能，具体截图如下：



图 6-16 清除路线



图 6-17 重新定位

通过点击不同按钮，MainActivity 会切换出不同的界面（Fragment），同时相应的点击按钮会作出相应的颜色改变，其中点击过后并切换至相应的界面时的颜色为浅蓝色背景，白色文字，未点击按钮或未切换的界面则为深蓝色背景，黑色文字，以下给出切换至不同的界面时测试截图（初始化时默认为路线模块界面，颜色未改变，表示该界面

为初始化界面）：



图 6-18 切换至订单模块



图 6-19 切换至路线模块

界面切换测试：两个界面（Fragment）除了数据传输基本保持相互独立，且在切换时不会引起其他界面生命周期变化或者重新生成需要切换的界面，具体测试步骤与截图如下：以当前处于路线模块开始，随机点击多个位置用于记录当前界面是否产生过变化，其路径生成截图如下：



图 6-20 随机点击多个坐标点

I/LatLonPoint: (30. 89215367784281, 117. 47721500887705)
(30. 84683518871518, 117. 46556751642913)
I/LatLonPoint: (30. 882130409462707, 117. 42857449032859)
I/Point number: 6
D/LocationList6: (30. 86477452064656, 117. 49182366002675)

图 5-18 后台数据变化的部分截图

切换至订单模块界面，并通过添加添加订单并在输入框中标明特定字符等操作记录当前界面是否被改变过，具体测试截图如下：



图 6-21 切换至订单并做好记录

```
W/IInputConnectionWrapper: beginBatchEdit on inactive InputConnection  
endBatchEdit on inactive InputConnection  
I/EditTextChangeListener: afterTextChanged---2
```

图 5-20 部分后台反馈信息

再次切换至路线模块以及返回再次返回至订单界面的截图（截图时间证明与上图不属于同一张）：



图 6-22 切换至路线界面



图 6-23 返回订单界面

以上测试表明主模块（MainActivity）切换不同界面时未对其他界面的生命周期产生影响。

6.4.4 外卖订单模块测试

添加订单测试：以下截图为切换至订单模块初始化时，以及添加订单后显示界面截图，订单模块动态生成订单栏，用于提交双方坐标，其中文本框提示中含有说明，后台日志同时给出反馈。

测试功能：添加订单功能，删除订单功能，提交订单功能

预期结果：

（1）添加订单功能：通过点击添加订单按钮，顶部新增订单栏，提示输入目标坐标。

（2）删除订单功能：通过点击删除订单按钮，能够删除最新添加的订单栏，若当前并无订单，则提示没有可删除订单。

（3）提交订单功能：能够通过点击提交订单按钮，将当前界面的订单坐标数据发送至算法模块，返回路径显示模块并绘制出路线。

测试过程：添加订单如图 6-18 所示，以下是测试之后的截图：



图 6-24 添加订单后

以下截图为 Android studio 运行系统日志界面：

```
W/le.fooddeliver: Accessing hidden method Landroid/os/storage/S
D/MyLocation: (30.877041,117.450638)
I/le.fooddeliver: ProcessProfilingInfo new_methods=10 is saved
```

图 6-25 测试前后台信息

```
I/setLinearLayout: setLinearLayout success  
I/Add_Order: Add Order success !
```

图 6-26 测试后后台信息

删除订单测试：以添加订单过后为当前状态，通过点击“删除订单”按钮，删除距离当前时间最近的订单栏，并以视图形式提交给用户。测试前如图 6-24，测试后如图 6-18。

```
D/MyLocation: (30.877041,117.450638)  
I/le.fooddeliver: ProcessProfilingInfo new_methods=10 is saved  
I/setLinearLayout: setLinearLayout success  
I/Add_Order: Add Order success !
```

图 6-27 测试前

```
D/MyLocation: (30.877041,117.450638)  
I/le.fooddeliver: ProcessProfilingInfo new_methods=10 is saved  
I/setLinearLayout: setLinearLayout success  
I/Add_Order: Add Order success !  
D/Del_Order: success
```

图 6-28 测试后

当前无订单状态测试删除订单：系统会自动弹出无订单提示，截图如下：



FoodDelivery: 已经没有订单了呢...



图 6-29 无订单状态下删除订单

I/EmptuList: 订单为空

I/Toast: Show toast from OpPackageName:com. example. fooddelivery,

图 6-30 后台日志反馈

提交订单测试：以北京市海淀区的数据集为测试数据，随机提取客户与商家坐标，添加多个订单栏并填充信息，点击提交按钮，在地图界面得到可视化路径规划。



图 6-31 提交前订单界面



图 6-32 提交前路线界面

提交订单后，app 将坐标序列传递至服务端，服务端调用蚁群算法的过程截图如服务端所示，服务端将运算结果的坐标序列通过 Tcp 连接发送到客户端，客户端接收数据后按照传递序列依次在路线模块绘制出行驶路线。



图 6-33 提交后订单界面



图 6-34 提交后路线界面

```
(39.9119, 116.303)
I/Point number: 12
D/LocationList12: (39.9595, 116.3461)
```

图 6-35 部分日志信息

6.5 对软件功能的评价

在整个基于蚁群算法的外卖配送 app 设计中,本次课题基本实现了单个骑手在单次使用中针对多个订单在基于蚁群算法的处理下绘制出规划路线。根据任务书中的功能要求,主要评价如下:

- (1) 数据采集,数据结果分析设计工作基本完成。
- (2) 针对单个骑手解决遍历路径优化 TSP 问题的蚁群算法基本完成,该蚁群算法成功应用于解决外卖配送中的路径优化问题。
- (3) 面向单个骑手使用的带有基本路径规划功能的移动 app 应用系统基本完成,能够实现当前提交状态下返回路径规划结果并于路径规划图示模块上显示。
- (4) 爬虫器基本成功实现抽取目标网站特定信息项数据,即外卖商家等详细信息。

第七章 结论与展望

7.1 结论

本次课题中针对外卖配送中的路径规划功能基本完成，能够实现单个骑手在接收多个订单配送的前提下，成功规划出配送路径。并实时定位骑手位置。能够在骑手移动过程中随时监听并更新骑手的位置。其中每个坐标点都能够查询到相应的位置信息，都具有明确的顺序编号，可以引导骑手按编号进行遍历送餐。

综上所述，该 app 基本完实现课题中所规定的功能设计，并且调用高德地图接口成功实现路径绘制以及各个坐标点的 Mark 记号。

7.2 展望

在本 app 的开发应用中，尽管能够实现课题中规定的基本功能，但是离市场化运营还具有一定的差距，首先订单界面的设计在面向对象的编程中，不符操作人员的使用习惯，而且订单栏里最好以具体地址显示为佳，通过逆地址编码技术可以定位到相应经纬度。

在下一步的预期中，针对本次研究实现的优化，其中订单部分可设计成对外的接口，通过接收商家发送的订单，以地址的形式出现在订单栏中，操作人员只需要在接收订单以后点击提交按钮。即可获得服务器发送的规划结果。此外，在条件允许的情况下，甚至可以与相应的市场公司进行合作。利用他们提供的接口，骑手在接受系统发来的订单过后，以结果的形式呈现在订单界面上。同时，去除添加订单与删除订单模块，只提供提交订单按钮，简化骑手操作，使软件更容易操作，更符合人们的使用习惯。

由于本次设计更偏向于算法的应用与研究，对于数据存储一块稍有疏忽，在以后的改进中，对于数据的管理这一方面可能需要加大投入力度，新增用户模块，设置访问权限。

参考文献

- [1] 李明俊. 2020-2025 年 中国在线外卖商业模式与投资战略规划分析报告[R]. 深圳: 前瞻产业研究院, 2019.
- [2] Dorigo M, Maniezzo V, Coloni A. The Ant System : Optimization by a Colony of Cooperating Agents[J]. IEEE Transactions on Systems, 1996, 26(1): 29-41
- [3] 张德丰, MATLAB R2017a[M]. 北京: 电子工业出版社, 2018: 208-233.
- [4] 吴庆洪, 张纪会, 徐心. 具有变异特征的蚁群算法[J]. 计算机研究与发展, 1999, 36(10): 328-333.
- [5] 胡坤. 基于改进蚁群算法的避免拥堵最优路径选择[D]. 成都: 西南交通大学, 2018: 32-33.
- [6] 苏晓勤. 改进蚁群算法的 TSP 问题研究[J]. 算法语言, 2019, 29 (12A): 42-43.
- [7] 黄心. 蚁群算法在外卖配送路径规划中的应用[A]. Value Engineering[C]. 昆明: 昆明理工大学国土资源工程学院, 2017: 65-67.
- [8] 峰峰, 王仁明, 伍佳. 求解 TSP 问题的一种改进蚁群算法[J] 自动化技术与应用, 2010 (7) 1-3.
- [9] 刘浩, 韩晶. MATLAB R2014a 完全自学[M]. 北京: 电子工业出版社, 2015.
- [10] 崔庆才. Python3 网络爬虫开发实战[M]. 北京: 人民邮电出版社, 2018.
- [11] 刘宇宙. Python3.5 从零开始学[M]. 北京: 清华大学出版社, 2018.
- [12] 明日科技. Android 开发从入门到精通[M]. 北京: 清华大学出版社, 2017. 10.
- [13] 付宇, 肖健梅. 动态自适应蚁群算法求解 TSP 问题[J]. 计算机辅助工程, 2008. 15(4): 10-13.
- [14] 张永强. 改进蚁群算法及其应用 [D]. 陕西: 西安工程大学, 2017. 3
- [15] 王健, 刘衍珩, 朱建启. 全局自适应蚁群优化算法[J]. 小型微型计算机系统, 2008. 29(6): 1083- 1087.
- [16] 王雷, 李明, 唐敦兵, 等. 基于改进遗传算法的机器人动态路径规划[J]. 南京航空航天大学学报, 2016, 48(6): 841- 846.
- [17] Azimirad V, Shorakaei H. Dual hierarchical control: A new global optimal path planning method genetic-optimal for robots [J]. Journal of Manufacturing Systems, 2014. 33(1): 139-148.
- [18] DORIGO M, MANIEZZO V, COLONI A. The ant system : optimization by a colony cooperating agents[J]. IEEE Transactions on Systems, Man and Cybernetics : Part B, 1996, 26(1): 13-19.
- [19] Marco Dorigo, Thomas Stutzle. 蚁群优化[M]. 北京: 清华大学出版社, 2007.
- [20] 张广林, 胡小梅, 柴剑飞, 赵磊, 俞涛. 路径规划算法及其应用综述[J]. 现代机

械, 2011 (5) :85-90.

[21] 刘崇奇. 改进蚁群算法及结构优化设计应用研究 [D]. 浙江: 浙江工业大学, 2017. 3. 27.

[22] 崔瀛. 关于物流配送路径优化过程中蚁群算法应用的讨论 [J]. 物流工程与管理, 38 (3) : 153-154.

[23] Azimirad V, Shorakaei H. Dual hierarchical control: A new global optimal path planning method genetic-optimal for robots [J]. Journal of Manufacturing Systems, 2014. 33(1):139-148.

[24] 张广林, 胡小梅, 柴剑飞, 赵磊, 俞涛. 路径规划算法及其应用综述 [J]. 现代机械, 2011 (5) :85-90.

[25] 马学森. 动态凸包引导的偏优规划蚁群算法求解 TSP 问题 [J]. 通信学报. 2018. 10, 39 (10) :59-71.

致谢

本次课题的顺利结束，是我本人在大学生活中的又一次重大经历，能够顺利完成大学本科毕业论文，离不开各位同学和老师的帮助，首先我感谢父母在日常生活中对我经济上的支持，以及生活中的关心，如果没有他们的支持，我的求学过程将无法像现在这样顺利。其次，还要感谢我的指导老师，如果没有老师的谆谆教诲，没有老师在我最艰难困苦的时刻给予我技术上的支持，并在整个课题完成过程中对我的细心教导，我可能无法顺利地结束整个课题，最后还要感谢我日常学习生活中的同伴，因为他们，我收获了大学四年中最珍贵的财富——他们的友谊，热情，付出都将成为我大学四年里最珍贵的回忆。如果没有他们与我交流彼此的想法，一起阐述自己的内心，我的思维也无法活动起来，那么我就无法在诸多的难题面前找出一套有效的解决办法。此外还要感谢许许多多在我一路走来对我提供过帮助的人们，没有他们提供的便利，我将寸步难行。最后再次感谢以上大学生活中对我有过诸多帮助的老师，同学以及亲人们，在此向他们表示最诚挚的谢意。