浙江工业大学

本科毕业设计（论文、创作）

题目：基于多移动机器人的订单分拣系统设计

作者姓名 章　鑫

指导教师 赵云波 教授

专业班级 理科实验班 1502

学　　院 健行学院

提交日期 2019 年 6 月 10 日

Dissertation Submitted to Zhejiang University of Technology for the Degree of Bachelor

Design of Order Sorting System Based on Multi-mobile Robot

Student: Zhang Xin

Advisor: Professor Zhao Yunbo

Jianxing Honors College
Zhejiang University of Technology
June 2019

浙江工业大学

本科生毕业设计(论文、创作)诚信承诺书

本人慎重承诺和声明：

1. 本人在毕业设计（论文、创作）撰写过程中，严格遵守学校有关规定，恪守学术规范，所呈交的毕业设计（论文、创作）是在指导教师指导下独立完成的；

2. 毕业设计（论文、创作）中无抄袭、剽窃或不正当引用他人学术观点、思想和学术成果，无虚构、篡改试验结果、统计资料、伪造数据和运算程序等情况；

3. 若有违反学术纪律的行为，本人愿意承担一切责任，并接受学校按有关规定给予的处理。

学生（签名）：

年　　月　　日

浙 江 工 业 大 学

本科生毕业设计（论文、创作）任务书

专 业 自动化 班 级 健行 1502 学生姓名/学号 章鑫/201524450119

一、设计（论文、创作）题目： 基于多移动机器人的订单分拣系统设计

二、主要任务与目标：

1. 了解掌握订单分拣常规算法；2. 针对订单分拣问题提出新模型与算法；3. 对算法进行分析和实现。

三、主要内容与基本要求：

1. 阅读相关文献，了解该领域基本研究现状；2. 研究多机器人调度相关算法；3. 提出并验证相关的订单分拣算法；4. 撰写毕业论文。

四、计划进度：

2019 开学前 收集相关资料文献，学习相关知识，完成外文翻译、文献综述；熟悉课题，做好开题准备。

第 1-3 周 完成开题报告，参加开题交流

第 4-8 周 提出算法并作验证等工作，接受中期检查

第 9-14 周 进行算法设计和实现等工作。撰写毕业论文初稿

第 15 周 论文修改，毕业答辩，提交相关文档资料

五、主要参考文献：

[1] 白帅福,唐敦兵,顾文斌,郑堃.基于混合区域控制模型的多移动机器人系统调度研究与实现[J].机械与电子,2012(03):8-12.

[2] 王帅安.自动分拣系统订单处理策略研究[D].清华大学,2009

任务书下发日期 2018 年 12 月 27 日

设计（论文、创作）工作自 2018 年 12 月 27 日 至 2019 年 6 月 12 日

设计（论文、创作）指导教师 赵云波

学科（方向）负责人 _____

主管院长 _____

基于多移动机器人的订单分拣系统设计

摘　要

随着网购以及快递物流行业的兴起，大批量订单中的货物分拣已成为电商仓储物流普遍现象，而如何进行高效的分拣也成为了关系到电商整体物流效率的首要难题。相比于传统的人工分拣系统，自动分拣系统的产生虽然使分拣效率有了质的飞跃，但对其订单处理策略效率问题，仍然有很多亟待解决。因此，针对电商仓储物流仓库具有的订单电子化大批量、自动分拣、货物繁多、多货架存放的特点，本文利用移动机器人分拣订单以及对订单进行合理分配以实现上述要求。

本文首先介绍了订单分拣的重要性以及电商物流仓库现有的分拣情况；其次，对订单分拣中的主要问题进行了介绍，并对订单分拣问题的国内外研究现状进行了阐述，分析了国内外研究趋势多元化的原因；然后在此基础上提出订单分拣相关改进算法，并利用 Matlab 仿真平台进行实现，并和传统仓储模型进行仿真对比，最后，结合现有的电商以及人工智能发展情况，对仓储物流的发展提出相应建议，并对其中订单分配和机器人配送环节的合理性和有效性进行展望。

具体工作如下:

1. 基于新型物流仓储模型，提出了分别在两种不同模式下的订单分配算法

2. 将算法在 Matlab 中实现，并与已有的部分 MATLAB 仿真软件平台结合，完善整体平台。

3. 将运用不同算法的新型仓储物流仿真平台与传统的仓储物流模型仿真平台进行对比。

关键词：订单分拣，订单分配，多移动机器人，分配算法

DESIGN OF ORDER SORTING SYSTEM BASED ON MULTI-MOBILE ROBOT

ABSTRACT

With the rise of online shopping and express logistics industry, the sorting of goods in bulk orders has become a common phenomenon of e-commerce warehousing and logistics, and how to carry out efficient sorting has also become the primary problem related to the overall logistics efficiency of e-commerce.Compared with the traditional manual sorting system, the production of automatic sorting system has made a qualitative leap in sorting efficiency, but there are still many problems to be solved in order processing strategy efficiency.Therefore, in view of the characteristics of e-commerce warehousing and logistics warehouses, such as electronic bulk ordering, automatic sorting, large quantity of goods and multi-shelf storage, this paper USES mobile robot to sort orders and make reasonable allocation of orders to achieve the above requirements.

This paper first introduces the importance of order sorting and the existing sorting situation of e-commerce logistics warehouse.Secondly, the main problems in order sorting are introduced, and the research status of order sorting at home and abroad is expounded, and the reasons for the diversification of research trends at home and abroad are analyzed.Then based on this, advances the order sorting algorithm, and use the Matlab simulation platform, implementation and simulation comparison and traditional storage model, finally, combined with the existing electricity as well as the development of artificial intelligence, puts forward corresponding Suggestions on the development of logistics, and the order allocation and distribution link the rationality and validity of the robot.

Specific work is as follows:

1. Based on the new logistics storage model, the order allocation algorithm under two different modes is proposed

2. Realized the algorithm in Matlab and combined with some existing Matlab simulation software platform to improve the overall platform.

3. Compare the new warehouse logistics simulation platform using different algorithms with the traditional warehouse logistics model simulation platform.

Key Words: order sorting, order allocation, multi-mobile robot, allocation algorithm

目　　录

第 1 章　绪　论

1.1　课题研究背景及意义

随着互联网时代的进步发展，电子商务公司也如雨后春笋纷纷成立，消费者通过在网上选择心仪的产品下单，然后卖家在晚上收到订单后用快递的形式寄出商品，完成交易。随着电子商务的快速发展,线上购物逐渐成为许多消费者的主流消费模式，许多知名互联网电商企业在日益强烈的竞争中把改善物流过程的效率视为重中之重。并且随着越来越多的重要节日例如双十一、年终大促等电商活动纷纷兴起，导致该时间节点的物流压力骤然增加，在这样的形势下,如何提高物流仓库的处理效率,减少订单堆积以至“爆仓”现象的产生,成为了物流中十分需要关注的问题。在整个物流仓储作业中，订单分拣是其中十分关键且极有可能最费劳动力的一个环节，整个环节据估计极有可能消耗整个过程近三分之二的劳动力。由此可见，研究并选择合适有效的分拣方法和设定恰当的分拣操作对仓库的整体生产速度的改善会有着至关重要的作用。

订单分拣是在仓库中的存储位置处取出物品以满足用户要求的一项仓库功能。分拣包括 3 个步骤：、拣选、运输、打包。根据近几年发展，在这些步骤中：拣选指令主要由仓库的处理器统筹并下达，而运输过程通常由移动机器人（移动机器人）进行以减少人工成本和提升正确率，而订单打包的物理过程较为简单且方式固定。

不同于传统的由人工分拣订单，现代智能化的仓库中仓储物流机器人在物流配送中已经开始逐渐普及，他通过显著的改善出错率提升了订单的处理效率。而指挥机器人进行货物运送，则需要对订单进行合理的分配并对机器人移动机器人进行合理的规划，现有的订单分拣方式存在的主要问题是是大批量订单“整体分拣”的能力低下，因此，如何合理的建立数学模型并给予智能货架合理分布及其相应的分拣方案，成为了研究的关键和目的。

1.2　问题描述

在仓储物流仓库中，最主要的工作就是：根据客户端传输过来的电子订单，订单中包含种类数量不一的货物，需要在实体仓库中准确找到所需货物所在的货架，并取下货物，统一整合后将一个订单的货物打包在一起。而一个成熟的仓库每天都会实时接收到

很多的客户订单，需要在有限的人力物力资源条件下更快刚准确的完成这些订单。

现有的订单分拣中，移动机器人的运用主要有两种情况。第一种移动机器人安装在小型货架底部，负责移动货架，当需要拣取该货架的货物时，工作人员下单指令，是移动机器人将货架移动到打包点，由人工取下货物进行打包。

第二种情况采用流水线作业，移动机器人将货物拣选后投放到流水线货带上，由流水线的工人进行整理和打包。

但以上两种模式都由很多的局限性，需要的人工以及时间成本较多。而针对智能化的无人仓库，移动机器人也就是无人车应该成为其中更为重要的主力，我们可以将移动机器人订单分拣系统的流程展示如下[1]：

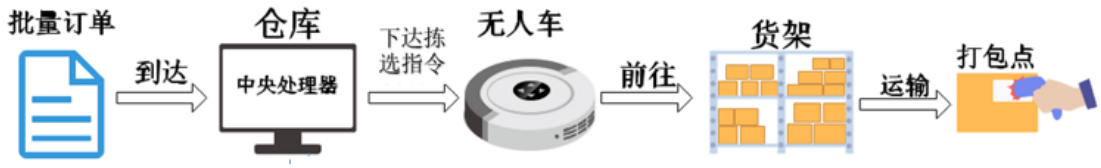

图 1-1　智能化仓库订单分拣系统流程图

如图，仓库中的中央处理器首先接收外部传入的订单，并结合订单信息和仓库中无人车的情况，选择合适的策略并下达相应的拣选指令给对应的无人车，无人车受到指令后，动身前往指令所规定的目的货架，取对应的货物，并把货物运输到打包点，进行打包，至此完成了仓库对于一笔订单的处理。

1.3　研究现状综述

在现有的电商平台，当客户下单之后，如何尽快的完成打包配送并发货，成为消费者衡量电商平台效率的重要因素。而上述所提的无人车仓储物流模式并不适用所有仓储物流的模型，在不同的实际仓储中，会因为模型的不同而产生不同的策略，因此，基于不同的思考角度和模型特点，国内外学者对于提升订单分拣系统效率的问题也有了各种各样的研究和看法。

台湾学者 Chun-Cheng Lin[2]利用曼顿距离的测量来计算仓库中的路径问题，并利用粒子群算法，在一定的实际约束条件下，找到仓库中物品种类繁多的订单的最优批量和最短的路径。较传统的粒子群算法相比，有了一定的提高，但由于其约束条件较为复杂，

货架摆放形式较为单一，在实际应用中，仍然有很多的适用局限性。

Mppani 和 Adil[3]把基于分类的分配问题变成一个非线性整数规划问题，并且用改进的分支定界法解决该问题。同时分析订单的更深层意义，例如，某些货品同时被顾客订购，它们之间有很高的关联性。考虑到类似关联性的存在，再合理的规划订单分配以及路径的规划。

在 Henn 和 Wäscher(2012)[4]关于订单分批问题的论文里，他们认为应当充分考虑订单本身来决定订单分拣的策略，他们主要考虑使移动机器人完成订单的整体运行路径最短。因此，受禁忌搜索算法的启迪，他们使用了 2 种方法来尝试优化上述问题：（1）经典禁忌搜索，（2）基于特征的爬山法。并且在之后，通过验证分析，以节约算法和局部搜索作为评判依据,讨论其有效性。

韩国学者 Soondo Hong 和 Youngjoo Kim[5]提出了一种 S 形路线的并行通道订单拣选系统路线分配模型。宽通道系统中的拣选机更喜欢 S 形路线，在最后一个过道处掉头以缩短行驶距离。在此基础上，提出了一种基于 s 形路径的订货批量的下界算法，并给出了 s 形路线订货批量的最优性差。

Bozer 和 Srinivasan[6]两位学者使用了串联区域控制模型，把整个仓库分为若干小的正方形封闭区域，每个正方形区域内由一辆移动机器人负责，而在相邻的小正方形场地之间需要建立货物转移站，提供移动机器人交换场地的场所，这很大程度简化了移动机器人的运行路径规划，并且避免的多辆移动机器人之间的碰撞，但是同一个货物的拣取过程可能需要经过多个移动机器人，并且在交换站交换要浪费大量的时间，增加了系统的额外的成本，对系统的效率有一定的影响。

在国内的研究中，王帅安（2008）[7]针对单一或多分拣线的自动分拣系统，分别尝试了不同的方法来提升效率。针对单分拣自动上货系统，尝试了两种启发式算法，缩短了订单的处理时间，提高了分拣效率。针对多分拣线同时分拣的自动分拣系统，提出了均匀分步法、遗传算法和而改进分支定界三种求解办法。

白帅福等人（2012）[8]提出了基于任务等待时间最短原则的 SWT 跳读策略，有效优化了分拣效率和整体运行时间，并提出了一种分布式协调的控制机制，解决了多移动机器人在线运行的冲突和思索问题。然而，该方案没能较好的适用于区域较大、移动机器人数量较多时的仓储系统，可能会带来系统实时性问题，有待于进一步的改进。

国内的韦超豪（2016）[9]订单分拣中关于订单优化的问题做了深入研究，提出 k-Means 算法和 Canopy 算法相互结合，先通过 Canopy 进行初步聚类得到初始的聚类中

心点,然后采用 k-Means 算法进行聚类得到最终的分批结果,在这其中，关于距离的定义是多个订单重新整合后的实际移动机器人所需移动总距离。

总的来说，国内外的研究角度和研究方法复杂多样，造成研究方法和方向多元的主要原因还是由于仓储中仓库的摆放情况各异，多种仓库货架的摆放架构使得对应的移动机器人路径会产生较大的不同，因此也需要考虑不同的算法策略去实现订单效率的最大化。换言之，我们在这设计算法策略之前就应该选择好合适的仓储货架摆放模型。

1.4 主要研究内容

依据传统的订单分拣问题以及现有的订单分拣系统，综合而言，一般会从一下三个子问题进行思考和完善：

（1）订单分批问题

已知货物的存储位置、货物的个数等一系列参数，当有外部订单生成发送到仓库，系统可以结合以上的参数信息，利用合适的决策方式，确定合适的订单完成顺序，调整订单的优先级，必要是可以对多个订单进行重新整合，进一步提升分拣效率。

（2）批次分配和定序问题

客户订单的特点是必须以满足期限的前提下，使每个已经被分配到某个批次的客户订单都能顺利完成。因此，当优先级确立，应将批次合理分配给有限数量的拣货移动机器人，并且对于每个拣货移动机器人，确定其对某个订单中货物的拣货顺序。

（3）移动机器人路径问题

当移动机器人收到需要拣货的订单指令，将前往已知货架拾取相应的物品，这些货架分散在不同的位置，那么应该以怎样的次序和行驶速度被访问，使的整体完成效率进一步提升，且在行驶中，如何让设计合适的规则， 减少碰撞和拥堵，成为移动机器人运行策略需要考虑的问题。

以上三个问题，是结合多种分析角度后，综合总结出的一些分类，而实际在不同的系统设计中并非完全独立，在有些算法中会把三个问题放在一起一同优化，而有些则是分别运用相对应的算法解决。需要在具体针对不同的订单分拣系统进行合适的算法设计。其中一二子问题也统称为订单分配问题。

将上述所述的问题细节化后，则可以从一下三个角度设计整体的运行策略，其中，第二三两个子问题是本文主要考虑设计的策略：

（1）移动机器人通行策略：针对移动机器人路径问题，设定合适的运行规则，避

免发生碰撞，减少拥堵。

（2）订单优先级策略：针对订单分批问题，对于接收到的订单，如何合理根据其订单内容和数量，合理排序，使得整体完成效率最大化

（3）订单分配策略：针对批次分配和定序问题，结合订单本身和移动机器人状态，将订单任务如何分配给合适的移动机器人完成，使得效率提升

本研究为面向订单分拣的多移动机器人订单分配研究，具体研究步骤规划如下：

（1）在前期有一定平台基础的情况下，继续完善仿真平台，针对订单和货物尽可能创建更为合理的信息数据集，是的后续不同方法的适用提供更多的可能性。分析研究出针对特定仓储货架摆放下的订单分拣需求新的订单分配算法，

（2）针对特定的仓库场景中订单分拣的特定需求，设计订单分拣中货物分批算法，可以尝试对已有的算法进行改进，也可以针对本研究特定的货物分拣设施，对订单分批和分配优先级设置自定义的算法，提升货物完成效率。

（3）完善软件平台，对不同算法之间进行比较评价。

1.5 本章小结

本章主要是讲述了本项目的研究背景及研究意义，并简单阐述了本文将研究的多移动机器人仓储物流模型的大致分拣流程。之后，针对不同的研究角度和方法，介绍了国内以及国际在现有的智能化订单分拣设计策略方面的研究情况。并依此简单概述了本文索要研究的内容以及侧重点。对研究的预期和具体工作做了简单的陈述和展望。

第 2 章　模型定义及平台介绍

2.1　传统的仓储物流模型

传统的仓储物流货架摆放如下图所示，移动机器人统一从角落起点出发，沿着 S 型的拣选路径方向行走，去拣取沿途中的货架上订单需求的货物，在移动过程中由于所有行进的移动机器人基本沿着大致一致的方向行走，发生碰撞的概率会较低，但也会由于在货架间通道采用依次前进的原则，速度会受到前方行驶车辆的影响，没有办法能很快行驶。且最终所有的移动机器人都要回到统一的起点最为打包点，整体的效率会受较大的影响。具体行走路径实例如下图所示，且当移动机器人完成所有货物的拣取后，会直行至底边沿仓库边缘返回左下角起始点。

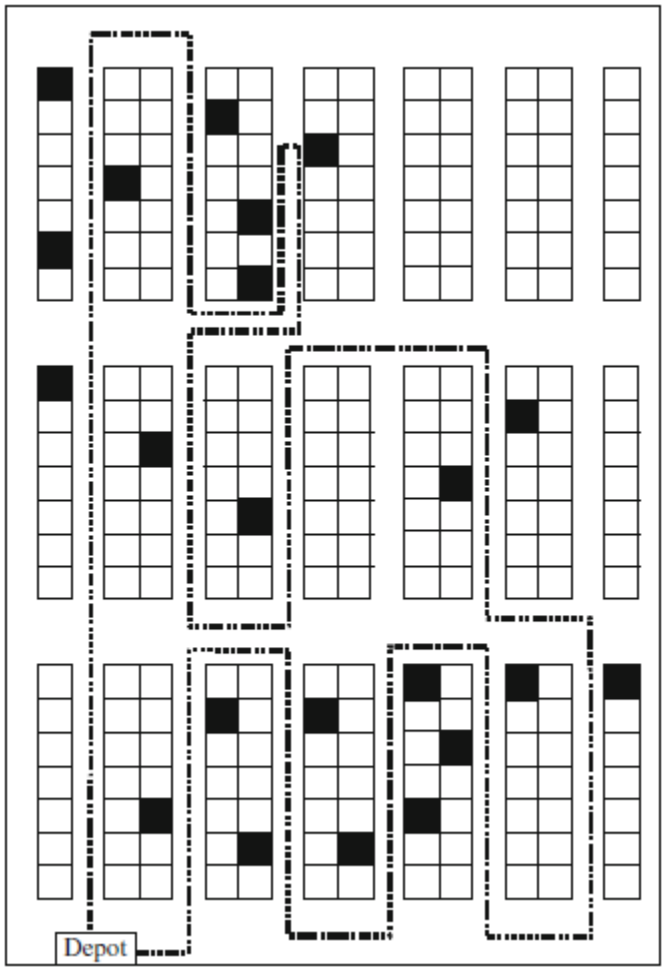

图 2-1　传统仓储物流运行示意图

如下图，将传统仓储物流模型利用 Matlab 平台进行仿真后的展示图如下，图中共 88 个货架，红色的移动机器人依次从菱形的起始点出发，以上下左右直行和直角转弯的形式依次从近到远达到指定的星型取货货架，拣取完所需全部货物后返回取货点。

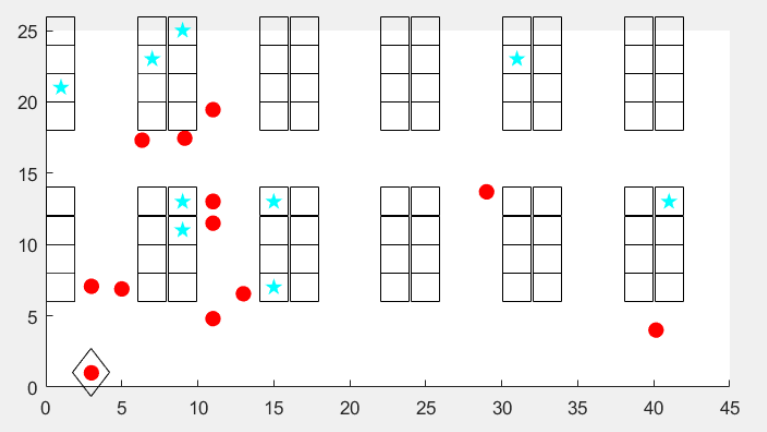

图 2-2　传统仓储物流模式 Matlab 仿真示意图

2.2　新型仓储物流模型

在以上的仓储物流模型中，移动机器人受货架摆放影响了行驶的路径和速度，从而降低了行驶效率，因此，在本研究中我们将采用另一种新提出但在实际运用中并不完全成熟但仍有较大提升空间的货架摆放方式，其摆放能够给移动机器人运输提供更大可能性，，成为分布并行式货架摆放模型，在下文统称为新型仓储物流模型。

在新型的仓储物流摆放方式中，货架连续的排成封闭的方形，移动机器人在其内部运输。

货架：一侧陈列货物，并且设置由机械手臂，当移动机器人到达货架时，可以接受到所需要的货物信息，由机械手臂拾取并将货物放置到移动机器人上。靠外一侧则是打包点，每个货架都可以在收集齐一个完整订单所有货物后，将货物送至货架外侧进行打包。

中央处理机构：获取现有的订单完成信息，以及所有移动机器人的每一时刻的状态，例如位置、速度、加速度，以及空闲情况。中央处理器收集以上信息后，将对所有信息进行整合考虑，规划订单完成的顺序，以及移动机器人的完成策略，并发出对应的指令给移动机器人下达任务。

移动机器人：可以接受指令，并在行进途中根据周围路况调整移动速度防止碰撞，接收指令后将按要求前往货物所在货架拾取物品送至最终打包货架。

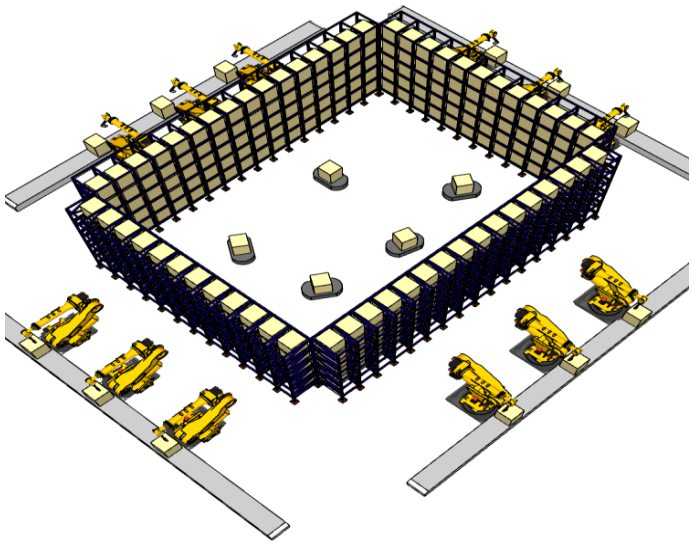

图 2-3　新型仓库模型概念图

本文的主要研究内容主要就是针对中央处理器策略算法，也就是如何协调所有的仓库信息，包括外部传入的订单信息，和仓库内部的移动机器人工作情况，合理的规划和分配工作任务，使整个工作效率最大化。移动机器人的速度方向控制在前期仿真平台中已经实现，而货架打包问题在本文中不做考虑。

针对以上三个定义和新型仓储物流模型，基础仿真平台设计界面如下：

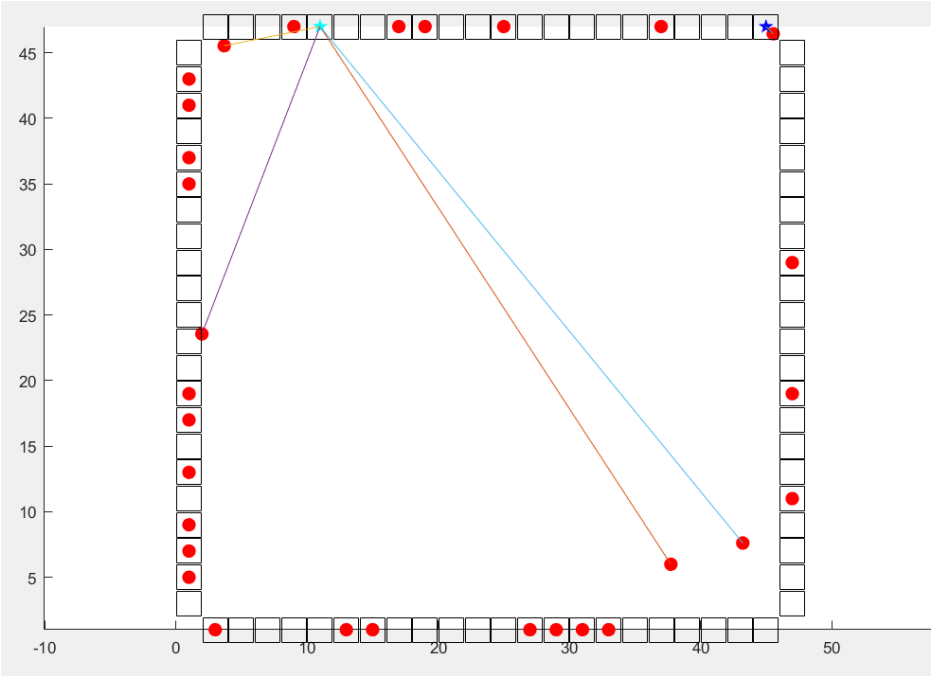

图 2-4　Matlab 可视化仿真平台示意图

如图所示，我们采用 Matlab 作为仿真平台，利用其中自带的图形界面显示功能来

模拟我们设定的封闭方行仓库摆放模式下的移动机器人运行过程，从而体现整个订单分拣的具体实时流程。该仿真平台在前期已经基本达到了可视化的效果，如图，我们选择一个小型仓库作为仿真的模型，可以看出一共 88 个小正方形代表 88 个货架组成了封闭仓库区域，其中的红色源泉部分表示正在运行或空闲的移动机器人，而移动机器人指向的线段则是代表的移动机器人正在行进的路线，五角星则是表示小车正在前往的目的货架。

2.3 模块介绍

对于新型的仓储物流模型，我们将在之后设计其细节部分的算法并完善 Matlab 仿真平台，针对现有的仿真平台，我们将其大致分为下述三个部分，其中，一二部分已经完成，第三部分则是本文研究的主要内容，需要设计相应算法并实现

2.3.1 订单产生

参考实际仓库中的订单产生情况，我们得知，在单位时间的内的订单产生可以近似的看成服从指数分布。根据概率论相关知识，我们可以得知指数分布的分布函数可以表示为：

$$F(x;\lambda)=\begin{cases}1-e^{-\lambda x} & x>0\\ 0 & x\leq 0\end{cases} \tag{2-1}$$

其中，λ 被称为率参数，且应当大于 0。即表示每单位时间内时间发生的次数。假设订单的产生并达到仓库符合λ_0的指数分布，则每一单位时间内有订单来到的概率为：

$$P(X\leq 1)=F(X)=1-e^{-\lambda_0} \tag{2-2}$$

另外，我们设定了每笔订单包含的货物数量的上限为 10 个，且每笔订单产生时包含的货物数量在 0-10 随机分布，且每个货物随机分布在编号 1-88 货架中的任意货架。至此我们实现了在仿真平台中，订单产生的随机性。

2.3.2 移动机器人的运行

设定移动机器人只能在货架围成封闭的区域内移动，空闲的移动机器人可以停在任意货架的底部但不能停在中间的空白区域内。特别说明，移动机器人可以在货架之间穿梭，且设定的货架区域较大，移动机器人不会再货架区域发生碰撞，在仿真界面中将移

动机器人图形相较于货架和仓库有所放大，因此显示时可能有重叠现象。除此之外所有的移动机器人从起点去往任意目的点的时候都沿直线行走，如果遇上相向而行的情况，我们认为移动机器人可以实现左右角度微偏并在行驶而过后较快的回到原始的路径轨道。

而由于仓库的移动机器人数量在多数情况下不会少于一个，则当移动机器人数量逐渐增多且多数移动机器人都处于非空闲状态时，那么其形式过程中极有可能发生运行轨道的交叉且碰撞，因此，仿真平台也设定了一定的防碰撞预测控制，通过每单位时间检测移动机器人的位置、速度、行驶线路，来预测之后可能发生的碰撞，并对将要发生的碰撞通过调整移动机器人加速度的方式进行防范。

在后续的仿真中，我们设定当移动机器人为防止碰撞而导致停车的情况，为“发生拥堵”，并记录每一次仿真中发生拥堵的次数。

2.3.3 订单分配

订单分配主要包含在第一章中所描述的订单分批和批次定序分配问题。

在我们设定的仿真平台中，当订单产生，就已经默认其已经传入了仓库的中央处理器。最简单的方法是采用最简单的先到先得原则，按订单到达的顺序进行分配，若前一订单没有完全分配，则不会分配之后的订单，而这必然没有达到最优设想的仓储运行效率。

在本文中，订单分配算法的设计成为研究的主要任务，我们不再简单采用先到先分配原则，订单产生后，中央处理器需要根据已经累计传入的未完成的所有订单情况，对整体做一个排序或者优化，目的是为了能够以更合理的方式完成这些订单任务，从而提升整个仓库的运行效率。具体的排序优化策略将在之后进行讨论。

除此之外，中央处理器需要将决定处理的订单分配给制定的移动机器人，并将具体的取货送货指令合理的告诉移动机器人。因此，确认打包货架和目的货架并分配给合适的移动机器人完成该订单，则是本文在仿真平台设计中的第二大重点。具体关于该模块的设计会在后序详细阐述。

2.4 本章小结

本章首先对介绍了传统的物流仓储运作模型，并展示了仿真平台的运行特点并提出了其中的不足和劣势，从而提出了本文所要采用的新型仓储物流运系统的优势之处。

接着，介绍了本文模型的相关 Matlab 仿真平台的各部分组成和功能，由于该平台只是有了基本的一些运行模块，但缺少实际的订单分配算法而无法真正的合理运行，因此，本文的研究重点也就是设计出合理的订单分配算法，使得新型仓储物流仿真模型能够有效的运行，并提升其效率。

第 3 章　订单分配算法设计

3.1　算法背景

本文设计的算法主要就是针对在订单分拣中中央处理器操作环节也就是上文提到的订单分配算法，且分为订单分批和批次定序及分配两部分。

订单分批算法的主要目的是为了能够更够对原始传入的订单进行一个再处理，优化订单的完成顺序，从而使整个仓库的效率有所提升。而批次定序和分配算法则是主要是从仓库内所有移动机器人的状态信息角度考虑，分析计算当前订单优先级最高的订单分配给具体哪个移动机器人去完成，并且移动机器人又应当具体以怎样的货架仿真顺序完成订单任务的问题。

而在实际的仓储物流当中，由于移动机器人的种类更加的多元化，有直径约为 60 公分的小型的移动机器人，也有体积较大的可以半直立的移动机器人，前者在仓储物流中一次性取货能力有限，而后者则可同时容纳较多的货物。

因此本文将考虑两种不同大小的移动机器人情况下，对其各自的的订单分批和批次定序和分配算法进行相应的设计和阐述。

3.2 一单多车模式

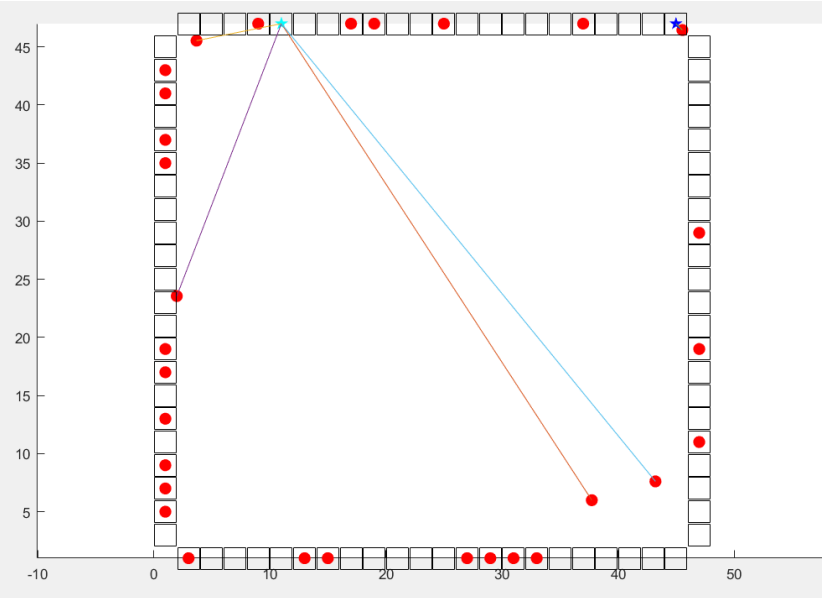

图 4-1　一单多车模式仿真运行示意图

针对体型较小的移动机器人，我们设定一辆移动机器人只能同时容纳一件获取的存

放，因此同一笔订单需要多辆移动机器人共同操作一起完成。本文称为一单多车模式，如图所示，图中所示的移动机器人在取得订单各自的货物后，将其统一送至标星号的打包货架点。

3.2.1 订单分批算法

首先对于已经传入的仓库的所有订单，排除已经完成打包和订单中所有货物都已经处理的订单，其余订单称为未分分配订单，由于此时采用一单多车的模式，而多辆车如果在同一时间段均前往同一打包货架点的话，极容易造成堵塞或者碰撞，因此我们给订单分配优先级时，尽量将点订单中包含剩余未分配货物较少的订单提前，特别说明，订单中剩余未分配货物较少的订单包含（1）本身订单中货物量较少，订单之前未进行分配（2）包含大量货物的订单之前已进行部分货物的分配，还有剩余一些货物未下派给对应的移动机器人。且为保证优先级的排序导致一些大数量订单一直被放置在最后未分配，设定一个延迟上限，当订单产生后一定时间内没有处理的订单无条件提前。

相关集合变量参数说明：

N ：未完成订单的集合 N={1,…,n}

G ：一个订单中的货物集合 G={1,…,g}，g∈ [0,10]

X_n:表示第 n 个订单中未完成分配的货物数量个数　　n∈ N

目标函数：　　$\mathrm{Max}\sum_{n\in N}（X_{n+1}-X_n）$　　(3-1)

3.2.2 批次定序和分配算法

将要分配某一订单时，首先检查该笔订单内的货物数量是否大于一个.若等于 1，则可以直接在唯一货架处直接打包送出，不需要移动机器人参与。之后，先确定仓库内存在空闲移动机器人，若存在，开始计算距离订单中所包含货架点各自最近的移动机器人位置的距离，并求和得D_1，然后分别计算以任一货架作为打包点时其余货架到打包货架的距离，并求和得D_2,求D_1和D_2且减去打包点附近移动机器人的距离后的和最小，从而确定该笔订单所对应的移动机器人拣货以及送货方式。特别说明，当每个货架点对应的最近移动机器人发生重复的时候，则选取距离顺位第二的移动机器人，以此类推，直到没有移动机器人可以分配。

相关集合说明：

C ：移动机器人的集合 C={1,…,c}

S ：所有货架的集合 S={1,…,s} s∈ [0,88]

S_n:第 n 个订单中所有货物对应的货架位置集合

N ：未完成订单的集合 N={1,…,n}

G_n ：订单 n 中的货物集合 G_n={1,…, g_n}，$g_n \in [0,10]$

参数说明：

C_S:距离货架 s 最近的空闲移动机器人 $C_s \in C$，$s \in S$

S_{ng}:第 n 个订单中第 g 个货物所在的货架位置

D(i, j)：i 和 j 位置对应的距离

E_n ：第 n 个订单所确定的最终打包货架 $E_n \in S_n$

T:表示一个订单从产生到完全分配所经历的时间阈值

相关变量说明：

T_n ： 订单 n 产生后至分配所经历的时间

K_{cgn} ：表示订单 n 中的货物 g 被是否被分配到移动机器人 c，若分配，则该值为 1，若没有分配，该值为 0

R_{in} ：表示第 n 份订单中的第 i 个货物的分配情况，当该货物已分配，则该该值等于 1，否则等于 0

Y_c ：当且仅当移动机器人 c 中只有一个货架上的货物时等于 1

目标函数：$\text{Min}\sum_{i\in G,i\neq E_n} D(S_{in},C_i)+\sum_{j\in G} D(S_{jn},E_n) \quad \forall n\in N$ (3-2)

约束条件：

$$\sum_{g_n\in G_n}\sum_{c\in C} K_{cgn}=1,\ \forall n\in N \tag{3-3}$$

$$Y_c=1 \qquad \forall c\in C \tag{3-4}$$

$$\sum_{i\in G} R_{in}=g_n \qquad n\in N \tag{3-5}$$

$$T_n<T \qquad n\in N \tag{3-6}$$

目标函数(3-2)最小化了订单完成所走过的距离最短。约束条件(3-3)是为了约束一个订单的一个货物只分配给一个移动机器人，约束条件(3-4)则是为了保证一个机器人在同一时刻只同时拿一个货物。条件(3-5)是为了防止订单的货物被漏分配，条件(3-6)确保不会有订单被太长时间的滞后未处理。

将上文的订单分配算法整合后用流程图展示如下：

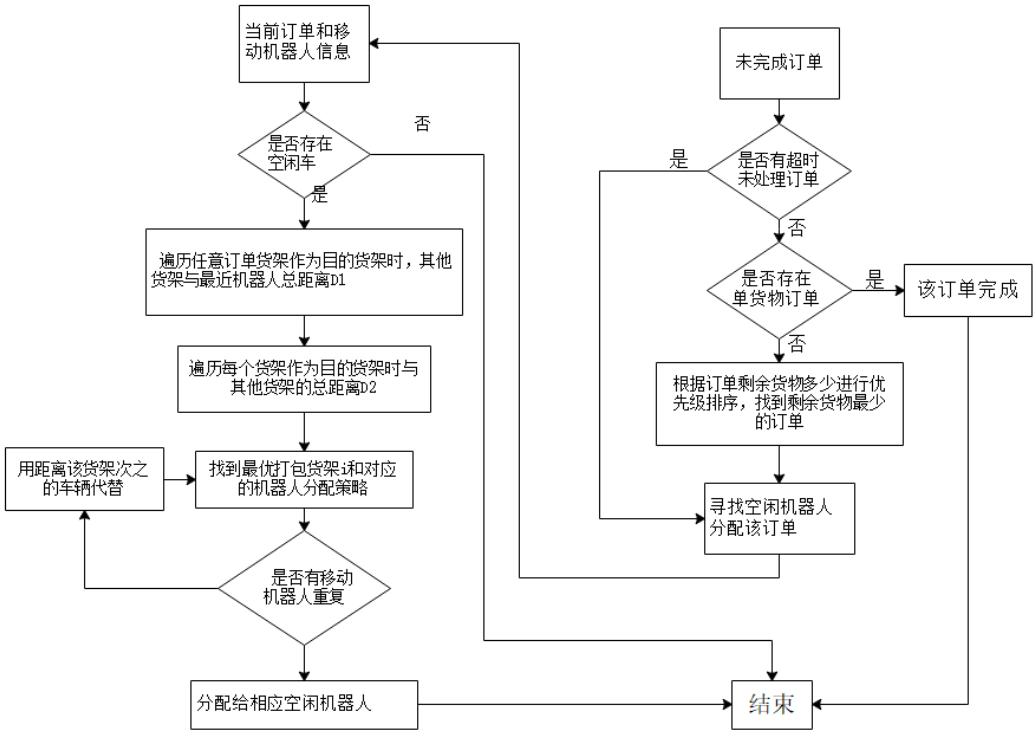

图 4-2　一单多车订单发放和分配流程图

3.3 一单一车模式

针对大型的移动机器人，其在运行途中可以同时容纳多种货物，在本文中，我们设定一辆大型的移动机器人可以一次性取不超过 10 件货物，也就意味着，一笔订单中的所有货物可以由且只需一辆一定机器人就能完成。当然，这种情况下对机器人的硬件也相应的有更加严格的要求。例如图所示，正在进行的订单包含了三个货物，移动机器人按图线所示的顺序依次取完货物，独立完成整个订单。

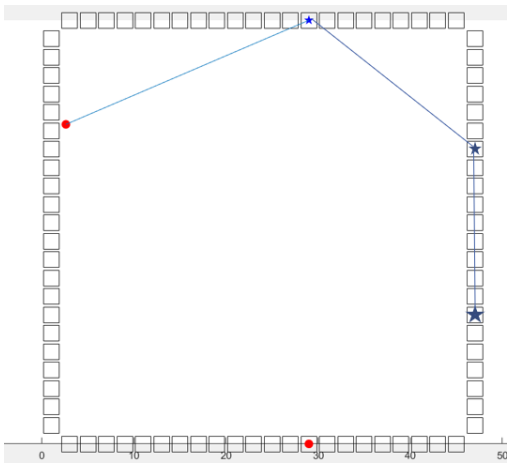

图 4-3　一单一车模式仿真运行示意图

3.3.1 订单分批算法

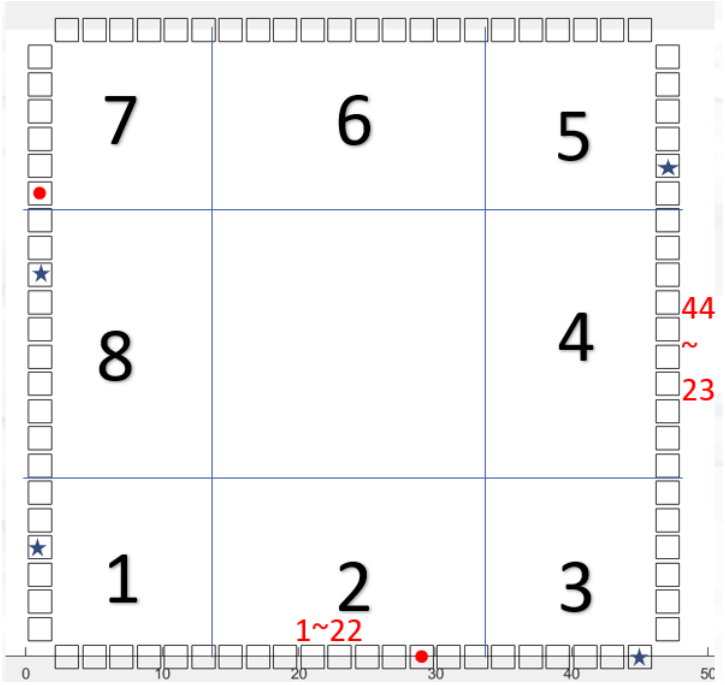

图 4-4　货架区域划分示意图

由于同一笔订单可以由一个移动机器人完成，则我们在处理厨师订单的时候，可以考虑在不超过机器人承载上线的情况下把订单进行合并，而对于合并的原则，我们采用聚类算法的部分思想，如图，我们可以简单的根据货架的位置将订单分为 8 个区域，例如货架 7~16 为第二区域，17~28 为第二区域。对于一个订单，可以设计一个相应的 8 维特征向量，若订单中包含区域内货架的货物，则向量中对应则置为 1，若不存在，则置为 0。举例：举例 如某一订单 i：订单包含货架:[22　40　75　86]，由于 22、40、75、86 分别位于 1 号、3 号、5 号和 8 号区域，提取特征信息后表示为：

$$Dif(i)=[\quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1] \tag{3-7}$$

又假设存在另一订单提取特征信息后表示为：

$$Dif(j)=[\quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1] \tag{3-8}$$

将两个特征向量的每个值依次做差，即订单 ij 相似关联程度为 1，关联度越小，说明订单相似程度越大，则根据设定的相似关联程度，也就是订单合并的阈值.，以及类似如上的订单特征情况，计算订单能否合并，且保证合并后订单包含货物数量不超过 10 个。合并后的订单会前置处理已增大仓库运行效率。同样对于产生后时间太久没有处理的订单也无条件前置。

相关集合变量参数说明：

N ：未完成订单的集合 N={1,…, n_1}

Difference:设定的订单合并阈值

Npro:合并后的订单集合 Npro={1,…, n_2}，$n_2 < n_1$

N_{miss} ： 被合并订单的集合 N_{miss}={1,…, n_3}

G_n ： 订单 n 中的货物集合 G_n={1,…, g_n}，$g_n \in [0,10]$

G_{pro}:合并后的一个订单中货物的集合G_{pro}={1,…, g_{pro}}

Dif：订单特征值的集合

Dif_{ik}:表示第 i 个订单在 k 区域时候有货物需要拣取，若有则为 1 否则为 0 $k \in [1,8]$

目标函数： $$Max(n_1-n_2) \tag{3-9}$$

约束条件：

$$Max(g_{pro}<10) \quad \forall g_{pro} \in G_{pro} \tag{3-10}$$

$$g_i>1, g_j>1 \qquad \forall i \in N_{miss}, \ \forall j \in N_{miss} \tag{3-11}$$

$$\sum_{i \in N, j \in N} Dif_{ik} - Dif_{jk} < Difference \tag{3-12}$$

目标函数(3-9)是为了尽可能进行订单合并，减少订单总个数。约束条件(3-10)是为了约束一个订单的货物最多只能有 10 个，约束条件(3-11)则是为了使订单中货物数量为 1 个的订单不被合并便于直接完成。条件(3-12)为了使合并的订单满足相关程度合并阈值。

3.3.2 批次定序和分配算法

将要分配某一订单时，依旧首先检查该笔订单内的货物数量是否大于 1。若等于 1，则可以直接在唯一货架处直接打包送出，不需要移动机器人参与。之后，先确定仓库内存在空闲移动机器人，若存在，首先计算所有货架按货架号由小至大依次连成的封闭区域的总周长D_1，然后分别计算以某一货架作为起始点，相邻货架作为打包点时距离起始货架最近的移动机器人D_2,从而使得减去起始和打包点距离之后的D_1和 D_2和最小，从而确定该笔订单所对应的移动机器人拣货以及送货方式。

相关集合说明：

C ： 移动机器人的集合 C={1,…,c}

S ： 所有货架的集合 S={1,…,s}

S_n:第 n 个订单中所有货物对应的货架位置集合

N ：未完成订单的集合 N={1,…,n}

G_n ：订单 n 中的货物集合 G_n={1,…, g_n}，$g_n \in [0,10]$

参数说明：

C_S:距离货架 s 最近的空闲移动机器人 $C_s \in C$，$s \in S$

S_{ng}:第 n 个订单中第 g 个货物所在的货架位置

D(i, j)：i 和 j 位置对应的距离

B_n ：第 n 个订单所确定的最终打包货架 $B_n \in S_n$

E_n ：第 n 个订单所确定的最终打包货架 $E_n \in S_n$

T:表示一个订单从产生到完全分配所经历的时间阈值

相关变量说明：

T_n ： 产生后且至今分配的订单 n 所经历的时间

K_{cgn} ：表示订单 n 中的货物 g 被是否被分配到移动机器人 c，若分配，则该值为 1，若没有分配，该值为 0

R_{in} ：表示第 n 份订单中的第 i 个货物的分配情况，当该货物已分配，则该该值等于 1，否则等于 0

Y_c ：当且仅当移动机器人 c 中只有一个订单的货物时等于 1

目标函数：$\mathrm{Min}\sum_{i\in G,j\in G,i>j} D(S_{in},S_{jn}) + D(S_{B_n},C_{B_n}) - D(B_n,E_n)$ (3-13)

约束：

$$\sum_{g_n\in G_n}\sum_{c\in C} K_{cgn} = 1 \qquad \forall n \in N \tag{3-14}$$

$$Y_c = 1 \qquad \forall c \in C \tag{3-15}$$

$$\sum_{i\in G_n} R_{in} = g_n \qquad n \in N \tag{3-16}$$

$$T_n < T \qquad \forall n \in N \tag{3-17}$$

目标函数(3-13)最小化了订单完成所走过的距离最短。约束条件(3-14)是为了约束一个订单的一个货物只分配给一个移动机器人，约束条件(3-15)则是为了保证一个机器人在同一时刻只拿属于一个订单的货物。条件(3-16)是为了防止订单的货物被漏分配，条件(3-17)确保不会有订单被太长时间的滞后未处理。

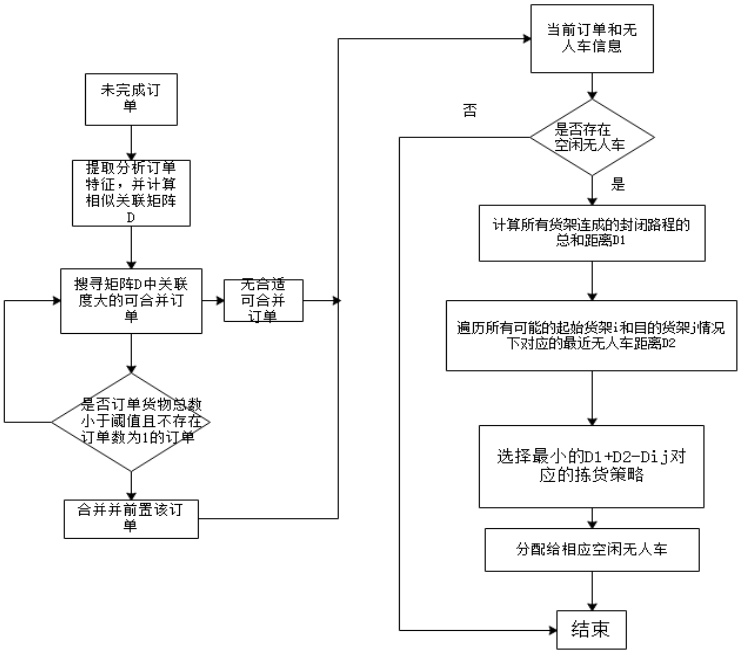

图 4-5　一单一车订单发放和分配流程图

将上文的订单分配算法整合后用流程图展示如上图所示。

3.4 本章小结

在本章中，首先阐明了算法设计所基于的整体仓储物流订单分拣系统的前提。并依据不同的移动机器人模型，提出了两种订单分配算法。基于小型移动机器人提出了一单多车模式，设计了相应的订单分批策略和批次定序分配策略。然后针对大型的移动机器人提出了一单一车模式，基于聚类部分思想对订单进行了整合，并采用旅行商解决策略设计了批次定序和分配算法，针对以上两种算法提出相应的公式描述和流程图展示。

第四章 仿真对比及结果分析

4.1 实验参数设定

本仿真实验在 Windows10 专业版系统，且为 Intel Core 处理器 i5 3.40GHz 和 4.00GB RAM 内存的个人电脑上进行。对了体现不同策略的对比效果，我们进行控制变量，在不同模式下，设定初始如下：订单产生的指数分布λ=0.3，运行循环次数 L=200 步，仓库货架输 N=88 个。

4.2 运行效率对比

为了更好的体现上述两种策略的对现有订单分拣系统效率的提升，我们采用现有的使用最为广泛的传统的仓储物流模型做对比。

仿真结果大致如下表所示，且结果显示在仿真中为探究仓库运行效率的最大化，产生订单个数均远大于完成个数，因此最后均有订单未分配，从而不存在仿真结束末尾由机器人因无订单可做而出现空闲的情况。

表 4-1 多模式下订单完成情况表

仓储物流模式	移动机器人数量/辆	平均完成订单数/个	平均完成货物数/件
传统模式	10	6	35
	15	9	42
	20	9	44
	30	11	68
一单多车模式	10	26	128
	15	33	176
	20	36	203
	30	37	225
一单一车模式	10	40	234
	15	41	239
	20	45	253
	30.	45	259

如上表所示，在不同模式下，随着移动机器人数量的增加，订单完成的个数和货物都有了一定的提高，为更加直观的比较不同模式之间的差异，我们将其转化为如下折线图。且由于每笔订单中的货物数量有较大差异，依据订单数量并不能直观反应仓储物流

效率，因此我们以完成的货物数作为衡量指标，具体图表如下：

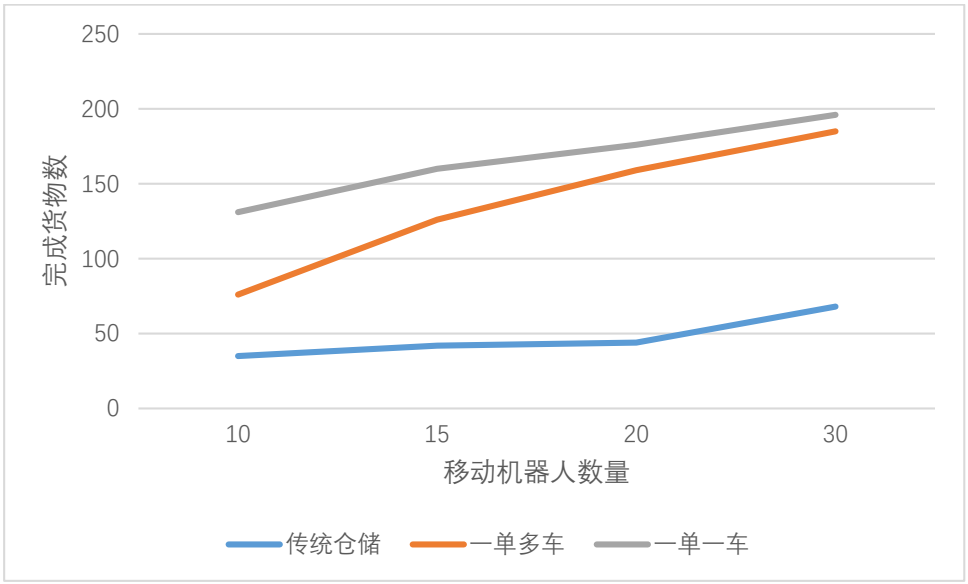

图 4-1　三种模式下订单完成情况折线图

根据以上的折线图显示结果，结合不同模式下订单分拣的情况，我们在一定合理情况下，做出以下的总结和分析：

（1）本文采用的新仓储模型下的一单多车和一单一车模型的仓库运行效率明显优于传统的仓储物流模型，而运行效率的提升有本身新型仓储物流模型的优势，也有所设计的订单分配算法的贡献。而这其中，再同等参数条件下，一单一车模型的效率更高。

（2）随着移动机器人个数的增加，三者的完成情况均有了一定的提升，但效果最为显著的一单多车模式，

经过分析，我们猜测传统模型下下由于机器人的运行路径趋势较为雷同，会受行驶在前方的机器人影响，单纯的增加机器人数量无法很明显的提升整个仓库效率甚至可能造成拥堵。

在一单一车模式下，机器人一次经过所需的目的货架，因此运行轨迹较一单多车模型相比更靠着这个仓库的边界运动，而中心区域则较少经过，因此，若机器人增加到一定程度也无法很好的提升仓库效率，因此对于一单一车的拣货模式来说，移动机器人数量的合理值应当会小于一单多车模式。

4.3 订单效率趋势分析

另外，我们也探究了在新型仓储物流模式下，仓库整体运行效率随时间的变化趋势。我们分别采用的上文提出的两种模式，并受制于硬件设备限制，对仿真运行步数设定为2500 步，并以每 100 步作为间隔，探究每 100 步移动机器人运输订单货物的完成情况。具体结果展示如下，其中移动机器人个数均为 15 辆，订单产生概率λ=0.3.

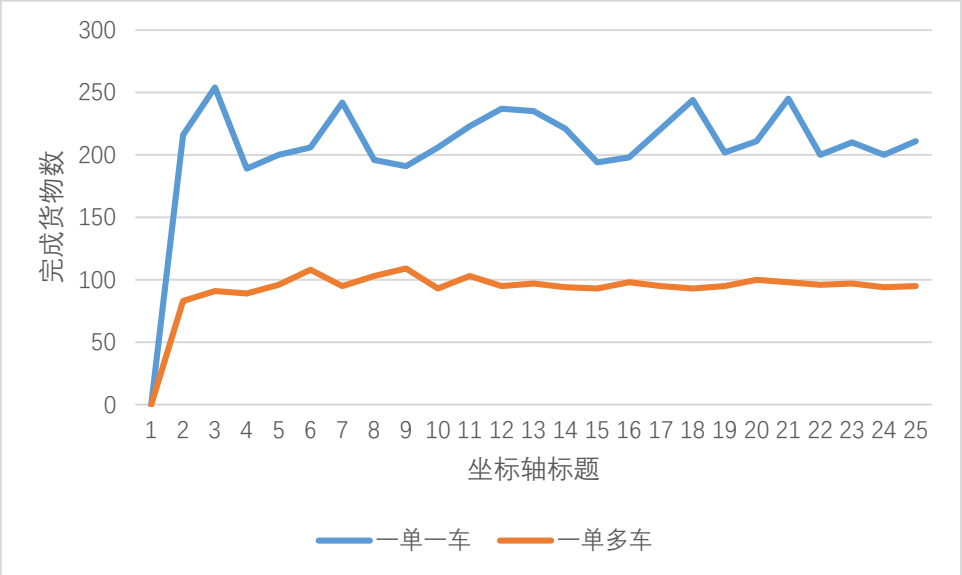

图 4-2　一单多车与一单一车模式下订单完成情况折线图

如图所示，随着仿真中步数的增加，两种模式下的订单完成情况趋势大体一致：起初会随着时间的增加而增长，之后在一定范围内波动，总体趋于平稳，及、即仓库订单每一定时间的完成效率达到较为稳定的状态。同时仍然可以看出，在同等条件下，一单一车模式的订单货物完成效率将远大于一单多车模式。

4.4 模式适用性分析

在上文我们已经初步了解了，本文所设计的仓储物流模式在 Matlab 上仿真平台上运行的时的大体结果和趋势，接下来我们将探究本文着重提出的两种方法，具体在不同仓储物流参数下的适用性。

4.4.1 一单多车模式

针对一单多车模式，由于一辆移动机器人只能同时拣取一个货物，因此理论上增加移动机器人数量能够有效的提升仓库效率，但由于移动机器人间的防碰撞机制，一味的

增加移动机器人数量，会导致发生拥堵从而降低每笔订单的完成效率。因此，我们尝试通过调整订单产生概率和移动机器人数量，分析其合适的适用范围。

具体结果如下表：

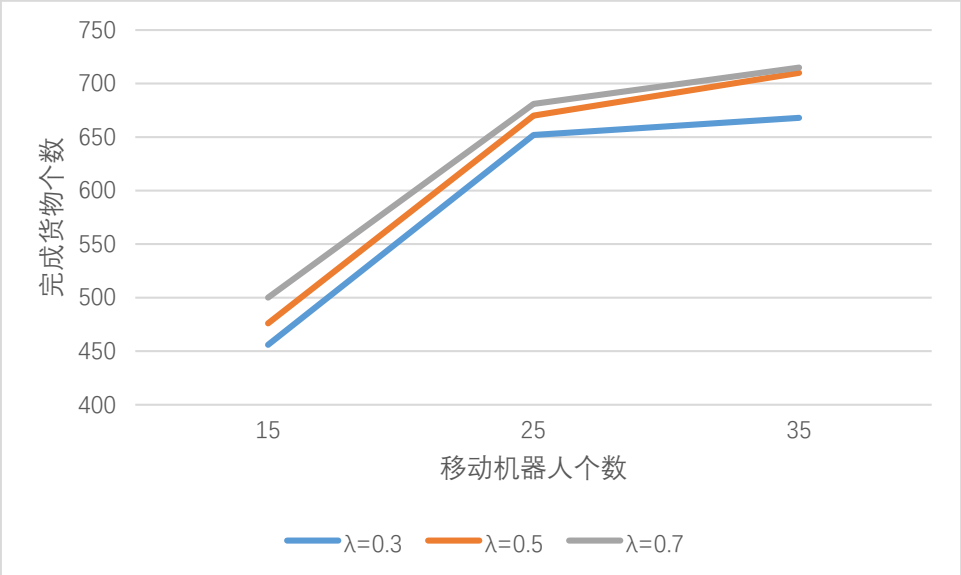

图 4-3 一单多车模式下订单完成情况折线图

如图所示，随着移动机器人数量的增加，仓库中货物完成个数逐步增加， 但同时，由于收到移动机器人数量的影响，在仓库中运行的移动机器人速度收到了极大的限制，并且由于订单产生概率的影响，订单的完成情况提升逐渐变得不在明显，因此根据观察可得，在一单多车模式下，移动机器人个数在 25 左右更为合理，且订单产生概率λ在 0.3~0.7 时，仓库整体运行效率没有很大影响。

4.4.2 一单一车模式

针对一单一车模式，我们同样尝试探究其合适的移动机器人使用数量。并且由于在本文设计的订单分批策略设计中，存在订单合并机制，关于这个机制的触发效果我们也做了相应了仿真分析。我们可以尝试选择合适的订单合并阈值，例如阈值为 0，则表示 i 订单中所有货架区域都被 j 订单中货架所在区域包含，从而可以进行合并。而若阈值为 1，则表示 i 订单中最多只有一个订单区域没有被 j 订单中的所含有的货架区域包含，但仍旧可以进行合并。

具体的实验结果如下，在λ=0.5 情况下，我们首先通过调整和设定订单合并的阈值，并根据完成货物个数和产生以及未完成订单两个值来判断最合理的移动机器人个数，同

时通过展现每 500 步中，订单合并的个数体现该参数设定的效果。同时由于订单产生概率不变，总订单个数在多次实验中没有较大差异，故下图订单相关分配或合并情况换算成百分比形式表示。

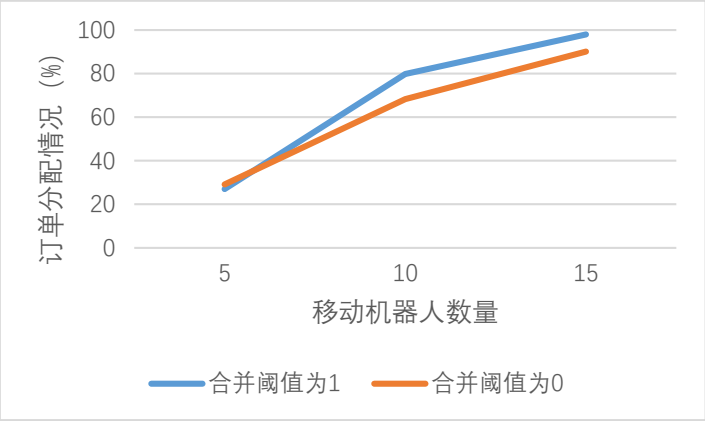

图 4-4　一单一车模式下订单完成情况折线图

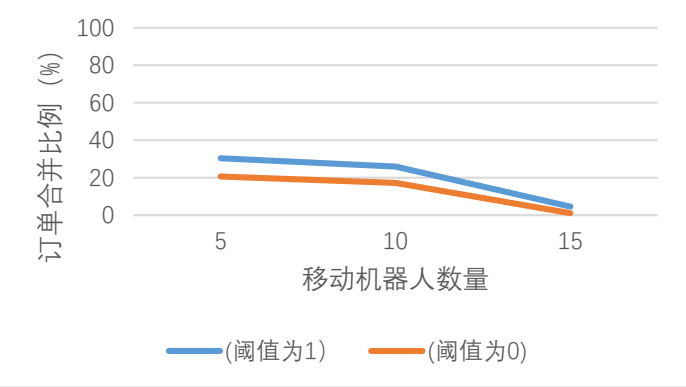

图 4-5　一单一车模式下订单合并情况折线图

如上图所示，随着移动机器人数量从 5 辆增加至 15 时，订单的完成情况情况越来越好，当移动机器人数量达到 15 时，几乎所有的订单都被完成或已被分配，造成了移动机器人的空闲数量过多。

同时由于订单分配并完成的效率过快，导致未完成订单的数量在仿真运行时始终很低，进而合并的数量随着移动机器人的数量增加而降低，从而失去了订单合并所带来的优势。

综合两方面的考虑，我们认为当λ=0.5 时，移动机器人的数量为 10 辆左右较为合理。较一单多车模式相比，移动机器人数量要求较低。且订单合并阈值设定为 1 更能提升整体仓库效率。

4.5 本章小结

本章首先简单探究了三种模式的优劣性，从完成效率上，新型仓储物流模型的效率显著高于传统物流模型，当然这其中包含了模型本身的优势，因此，我们又将一单一车和一单多车模型进行对比，发现同等情况下一单一车的效率较一单多车确实有显著提高，且两种模型下，随着仓库运行时间不断增长，订单完成的效率都能基本达到一个稳定的状态。最后，我们分别探究了两种模式策略下的适用参数，一单一车所适合的移动机器人数量较少，而一单多车策略下的移动机器人数量也不应无限制增加，否则会造成拥堵。

第 5 章　总结和展望

5.1　毕设工作总结

订单分拣系统在近几年越来越收到各大电商平台的关注，而如何进行高效的分拣也成为了关系到电商整体物流效率的首要难题。相比于传统的人工分拣系统，自动分拣系统的产生虽然使分拣效率有了质的飞跃，但对其订单处理策略效率问题，仍然有很多亟待解决。因此，针对电商仓储物流仓库具有的订单电子化大批量、自动分拣、货物繁多、多货架存放的特点，我们就利用移动机器人分拣订单以及对订单进行合理分配以实现上述要求。

本文的研究内容是基于多移动机器人的订单分拣系统设计，具体也就是关于其中订单分配算法的研究。现将本文的工作总结如下：

⑴阅读并掌握一些现有订单分拣系统算法的情况下，介绍了订单分拣在近几年的研究背景和意义，结合国内外的文献对现有的研究情况进行了总结和分析。

⑵讲述了本文所基于的 MATLAB 平台，以及搭建的软件仿真平台中对应的各个仿真模块的组成和运行原理。

⑶基于一种新型仓储物流模式，提出并介绍了本文的的订单分批以及批次定序和分配算法的设计思路和具体实现流程

⑷将本文设计的算法实现在仿真平台实现后与最典型传统的仓储物流模型进行了比较通过比较订单完成情况来得出结论:本文的算法和模型优于传统的仓储物流模型，并且大致展现了仓储物流中订单完成的规律。

5.2　未来展望

随着毕设课题的开展深入，发现针对订单分拣问题，可以优化和考虑的方向还有很多，本文所提出的，只是一种非常理论的订单分拣方法，课题本身也还处于非常初级的阶段，后续还可以有较多的改进和完善指出：

（1）本文对于这种全新的订单分拣系统，只是搭建了较为简单的仿真平台，并没有做真实的实物实验，在实际操作中，对移动机器人与中央处理器之间的订单传输速率有一定要求，也可能会遇到丢包、信息传输存在延迟的问题，这些在仿真平台中都没有体现。除此之外，对移动机器人本身也有一定的硬件要求，加速度的控制，最大行驶路程，载重等等都是实际运行时不可避免需要考虑的问题。

（2）对于机器人的运行规则，在本文研究中没有详细考虑，实际运行时，更为有效的及时预测以及防碰撞策略也能在很大程度上提升完成订单的效率，从而优化整个系统。

（3）在本文中，为探究仓库效率的最大值，因此移动机器人几乎都处于时刻运行的情况下，总而也没有很完整的考虑当移动机器人空闲时该如何和合理停车的问题。

（4）本文在仿真实验时，对于一些变量没有进行更深入的比较研究，比如仓库大小也就是货架数量的多少，或者订单中货物数量的上限，其实都可以进行详细的探究，从而更完整的得到整个模型应用的更优规模和条件。

参 考 文 献

[1] Scholz, André, Schubert D, WäScher G. Order picking with multiple pickers and due dates – simultaneous solution of order batching, batch assignment and sequencing, and picker routing problems[J]. European Journal of Operational Research, 2017, 10(08): 12-44.

[2] Lin C C, Kang J R, Hou C C, et al. Joint order batching and picker Manhattan routing problem[J]. Computers & Industrial Engineering, 2016, 95: 164-174.

[3] Muppani V R, Adil G K. Efficient formation of storage classes for warehouse storage location assignment: A simulated annealing approach[J]. Omega, 2008, 36(4): 609-618.

[4] Henn S, Koch S, Wäscher G. Order batching in order picking warehouses: a survey of solution approaches[J]. Femm Working Papers, 2012: 105-137.

[5] Hong S, Kim Y. A route-selecting order batching model with the S-shape routes in a parallel-aisle order picking system[J]. European Journal of Operational Research, 2017, 257(1): 185-196.

[6] Bozer Y A, Srinivasan M M. Tandem AGV systems; a partitioning algorithm and performance comparison with conventional AUV systems[J]. European Journal of Operational Research, 1992,63(2): 173-191.

[7]王帅安. 自动分拣系统订单处理策略研究[D]. 清华大学, 2009.

[8]白帅福, 唐敦兵, 顾文斌, 郑堃. 基于混合区域控制模型的多移动机器人系统调度研究与实现[J]. 机械与电子, 2012(03): 8-12.

[9] 韦超豪. 面向 B2C 电商平台的订单分拣优化研究[D]. 合肥工业大学, 2016.

[10] 马廷伟. 物流中心订单分拣策略的研究[D]. 北京邮电大学, 2015.

[11] 王宏琴, 贾晓庆. B2C 企业自建物流中心分拣系统研究[J]. 中国市场, 2012(28):54-56.

[12] 刘田. 仓储系统中分拣-存取效率优化模型与策略研究[D]. 华中科技大学, 2016.

[13] 王文蕊. 电子商务配送中心的设计与优化策略研究[D]. 山东大学, 2014.

[14] 肖际伟. 配送中心拣货系统优化[D]. 山东大学, 2010.

[15] 李树平. 快递企业客户关系管理问题研究[J]. 发明与创新(职业教育). 2018(07):53.

[16] 张鹏. 制造商线上分销渠道模式选择研究[J]. 技术经济与管理研究. 2018(10):13-18.

[17] 刘虹, 潘亚宏. 双渠道供应链定价策略研究——基于随机需求下考虑质量改进情

形的分析[J]. 价格理论与实践. 2018(12):165-168.

[18] 王彤. 基于“互联网+”的企业财务管理模式探讨[J]. 中国管理信息化. 2019(02):53-54.

[19] 丁浩延. 电子商务环境下物流管理的优化策略[J]. 现代营销(下旬刊). 2019(01):150.

附录

#主要程序

一单多车：

订单分批策略

```
[orderx,ordery]=size(order);
if orderx~=1
for i=1:orderx-1
      for j=i+1:orderx
       if order(i,2)==0&&order(j,2)==0
       ordersum1=0;
       ordersum2=0;
       ordernum1=0;
       ordernum2=0;
       for k=1:10
            if order(i,2*k-1)~=0
               ordernum1=ordernum1+1    ;
            if order(i,2*k)==1
        ordersum1=order(i,2*k)+ordersum1;
            end
            end
       end
       for k=1:10
            if order(j,2*k-1)~=0
               ordernum2=ordernum2+1 ;
            if order(j,2*k)==1
        ordersum2=order(j,2*k)+ordersum2;
            end
            end
       end

       if                    ordernum1-ordersum1~=0&ordernum2-ordersum2~=0&ordernum1-
ordersum1>ordernum2-ordersum2
            order([i,j],:)=order([j,i],:);
```

```
            norderinum=norder(i).num;
            nordericount=norder(i).count;
            norderibtime=norder(i).btime;
            norder(i).num=norder(j).num;
            norder(i).count=norder(j).count;
            norder(i).btime=norder(j).btime;
            norder(j).num=norderinum;
            norder(j).count=nordericount;
            norder(j).btime=norderibtime;
        end
        end
      end
end
end
for i=1:CARNUM
       carst(1,i)=ncar(i).state;
 end

if(min(carst)==0 && ORDERNUM>0 && norder(m).state~=2)%只有存在空闲车辆且订单中货物多余一件,且剩余订单数大于零时，才进行订单分配
      if(norder(m).state==0)
      [a,b]=orderdistribution(m,order,ncar);
      norder(m).state=1;  %订单正在处理阶段
      norder(m).count=order(m,2*max+2)-1;
      N_g=N_g+1;
      end
      %订单中货物状态：初始 0 ； 已分配 1 ； 未分配 2
      for i=1:2:2*order(m,2*max+2)%步长改为 2
            c=b((i+1)/2,2);
            if(order(m,i+1)==2)
                  [num,ind]=min(carst);
                  c = ind;
                  order(m,i+1)=0;
```

```
        end
            if(ORDERNUM~=0 && order(m,2*max+2)==1)%给空闲的车分配任务，订单里只含有一个商品
              ncar(c).state=0;     %车空闲
              N_g=N_g+1;
              norder(m).state=2;   %订单完成
              norder(m).etime=cputime;%订单完成时间
              m=m+1;   %订单完成
                 N_f=N_f+1;
            elseif(ORDERNUM~=0 && order(m,i)==a(1,2) && order(m,i+1)~= 1)%该货架是最终打包货架，则跳过
                order(m,i+1)= 1;
                n=n+2;
                 if(n==2*order(m,2*max+2)+1)   %若最终打包货架是最后一个货物所在货架
                      m=m+1;   %订单号加一
                      n=1;   %从第一个货物开始
                      break;
                 end
            elseif(ORDERNUM~=0 && ncar(c).state==0 && order(m,i)~=a(1,2)&& order(m,i+1)~=1)   %给空闲的车分配任务，订单里含有多个商品，处理的货物数小于订单内货物的总数
                      ncar(c).state=1;   %取货阶段
                      ncar(c).count=m;     %正在处理第 m 个订单
                      order(m,i+1)= 1;
                      ncar(c).begiX=ncar(c).x;     %车的起点
                      ncar(c).begiY=ncar(c).y;
                      ncar(c).endiX=uuhuojia(a(1,2)).x;   %货物的最终目的地
                      ncar(c).endiY=uuhuojia(a(1,2)).y;
                     fd(ncar(c).count)=plot(ncar(c).endiX,ncar(c).endiY,'pentagram', 'MarkerSize', 8,'MarkerEdgeColor','c','MarkerFaceColor','c'); %确定最优目的货架的位置标
```

记成亮蓝色

```
                    ncar(c).midiX=uuhuojia(order(m,i)).x;   %取货货架的位置
                     ncar(c).midiY=uuhuojia(order(m,i)).y;
                    ms(c)=  plot(ncar(c).midiX,ncar(c).midiY,'pentagram',  'MarkerSize',
8,'MarkerEdgeColor','b','MarkerFaceColor','b');   %确定中间取货货架的位置并标记成蓝
色
                %==========================================
                 lm(c)=plot([ncar(c).x,ncar(c).midiX],[ncar(c).y,ncar(c).midiY]);%画车与
取货货架的线
                    %==========================================
                     n=n+2;
                    if(n==2*order(m,2*max+2)+1)   %当订单内货物分配完成后
                        m=m+1;   %订单号加一
                        n=1;   %从第一个货物开始
                        break;
                    end
            elseif(ORDERNUM~=0 && order(m,i+1)==1)%货物已分配
            elseif(ORDERNUM~=0 && ncar(c).state~=0 && order(m,i+1)==0)%货物未
分配且车辆不空闲
                  order(m,i+1)=2;%该货物优先度标记为 2
            end
    end
end
%--------------------------------------------------------------------------------
```

批次定序和分配：

```
function [a,dcp]=orderdistribution(m,order,ncar)
global CARNUM max;
load huojia;
dcp=zeros(max,2);%用于存放车辆和货物之间的距离的数组
aa=zeros();
fxsum=zeros();
totalgx=zeros();
```

```
finalspot=zeros();
overall=zeros();%用于存储全局车辆距离货架远近的数组
largenum=1000000;
for i=1:2:2*order(m,2*max+2)
        fx=0;
    for j=1:2:2*order(m,2*max+2)
        if(i~=j)
        n1=order(m,i);%第 i 件货物所在货架编号
        n2=order(m,j);%第 j 件货物所在货架编号
        fx=fx+(huojia(n1,2)-huojia(n2,2))^2+(huojia(n1,3)-huojia(n2,3))^2;%以订单中第 i 个物品为打包点的其余物品与该点路程之和的平方
        end
    end
    fxsum(1,(i+1)/2)=fx;%将计算路程放入数组
end
%[minfxsum,index]=min(fxsum);%返回数组中最小元素及它的位置，它的位置即使最佳打包点
%a=[minfxsum,index];%1*2

%找到运货的最优车辆
for i=1:2:2*order(m,2*max+2)
    for j=1:CARNUM
        if(ncar(j).state==0)
        n1=order(m,i);
        gx=(huojia(n1,2)-ncar(j).x)^2+(huojia(n1,3)-ncar(j).y)^2;
        aa(1,j)=gx;

        end
        if(ncar(j).state~=0)
            aa(1,j)=largenum;%货架到无人车的最远距离应该是 46^2+46^2=4232，用以除去不在空闲状态的车辆
            %问题：当所有车均不在空闲状态时，数组中全是 100000，本来这个时候
```

```
应该等待有车空闲时，在分配任务。
                %但是由于车辆都还没有到达目的地，无车空闲，但是订单还在继续分配，此时应停止订单分配
            end
        end
        overall(1:CARNUM,(i+1)/2,1)=aa;%列存第 i 个货物距所有车的距离
        [mingx]=min(aa);%获得距第 i 个货物最近的车辆距离以及其编号
        totalgx(1,(i+1)/2)=mingx;%将每个货物最近的车辆距其距离存入数组
%         dcp(i,:)=[mingx,carindex];%m*2
end

%计算综合最小值
for i=1:2:2*order(m,2*max+2)
        fx=0;
        for j=1:2:2*order(m,2*max+2)
            if(i~=j)
                fx=fx+totalgx(1,(j+1)/2);%除去货架 i 外所有货架与其最近车辆的距离相加
            end
        end
            finalspot(1,(i+1)/2)=fxsum(1,(i+1)/2)+fx;%将第 i 个货物所在货架与其他货架之间的距离和除货架 i 外所有货架与其最近车辆的距离之和相加放入数组
end
[totalmin,huojiaindex]=min(finalspot);%找到总路径和最小值和最终打包点货架

a=[totalmin,order(m,(huojiaindex*2-1))];%1*2
%order(m,huojiaindex)——第 i 个货物所在货架编号

%处理车辆调用冲突
%车辆调用出现冲突时，直接少了一个或多个货物!!!

%将货物 i 与所有车辆的距离和对应车辆编号存入数组，并且升序排列
%overall(:,:,1)按列存放货物 i 与所有车辆的距离，overall(:,:,2)存放对应车辆编号
```

```
for i=1:2:2*order(m,2*max+2)
    temporary(:,1) = overall(:,(i+1)/2,1);
    temporary(:,2) = 1:CARNUM;
    temporary = sortrows(temporary);%将矩阵按第一列从小到大排序
    overall(:,(i+1)/2,1) = temporary(:,1);
    overall(:,(i+1)/2,2) = temporary(:,2);
end
%找出拥有重复最优车辆的订单中货物并将其用次优车代替直至无重复车辆
%找出 overall(:,:,2)中第一行中重复数的位置，并且用下一行相应的数替换，直至第一行没有重复数字，并且将其对应的距离也替换掉
a1 = overall(:,huojiaindex,2);
overall(:,huojiaindex,2) = zeros();
x=overall(1,:,2);
for j=1:CARNUM

x_u = unique(x);
b=[];
for i=1:length(x_u)
    if(length(find(x==x_u(i)))~=1)
        c=find(x==x_u(i));
        b=[b,c(1,2:length(c))];
    end
end

if(isempty(b))
    break;
end
x(1,b)=overall(j,b,2);
overall(1,b,1) = overall(j,b,1);%将车编号对应的距离也覆盖过去
end
overall(1,:,2) = x;
overall(:,huojiaindex,2) = a1;
```

```
for i=1:2:2*order(m,2*max+2)
    dcp((i+1)/2,:)=[overall(1,(i+1)/2,1),overall(1,(i+1)/2,2)];
end

end
```

一单一车：

订单分批：

```
%-----------ORDER 预处理,按货架大小排序--------
for i=1:2:2*order(mm,2*MAX+2)-2
    for j=i+2:2:2*order(mm,2*MAX+2)
        if order(mm,i)>order(mm,j)
            order(mm,[i,j])=order(mm,[j,i]);
        end
    end
end
%------------------------------
mm=mm+1;
else
R=0;
end
%————————————————————————————————
if(R == 1)
    norder(mm-1)=struct('state',0,'num',order(mm-1,2*MAX+2),'count',0,'btime',order(mm-
1,2*MAX+3),'etime',0);
    norder(mm)=struct('state',2,'num',0,'count',0,'btime',0,'etime',0);
end

intng(cout)=intnum;
%--------------------------------------------------------------------------------
if R==1
[orderx,ordery]=size(order);
copyk=1;
ordercopy=zeros();
```

```
for i=1:orderx
    if order(i,2) ==0 && order(i,22)~=0
        for j=1:order(i,2*MAX+2)
            if(order(i,2*j-1)<7||order(i,2*j-1)>82)
                ordercopy(copyk,1)=1;
            end
            if(order(i,2*j-1)<17&&order(i,2*j-1)>6)
                ordercopy(copyk,2)=1;
            end
            if(order(i,2*j-1)<29&&order(i,2*j-1)>16)
                ordercopy(copyk,3)=1;
            end
            if(order(i,2*j-1)<39&&order(i,2*j-1)>28)
                ordercopy(copyk,4)=1;
            end
            if(order(i,2*j-1)<51&&order(i,2*j-1)>38)
                ordercopy(copyk,5)=1;
            end
            if(order(i,2*j-1)<61&&order(i,2*j-1)>50)
                ordercopy(copyk,6)=1;
            end
            if(order(i,2*j-1)<73&&order(i,2*j-1)>60)
                ordercopy(copyk,7)=1;
            end
            if(order(i,2*j-1)<83&&order(i,2*j-1)>72)
                ordercopy(copyk,8)=1;
            end
        end
        ordercopy(copyk,9)=order(i,2*MAX+2);
        ordercopy(copyk,10)=i;
        copyk=copyk+1;
    end
end
[ordercopyx,ordercopyy]=size(ordercopy);
```

```
D=zeros();
for     i=1:ordercopyx
      for j=1:ordercopyx
            D(i,j)=0;
             for kk=1:8
                 if ordercopy(i,kk)-ordercopy(j,kk)==1
                 D(i,j)=D(i,j)+(ordercopy(i,kk)-ordercopy(j,kk));
                 end
           end
      end
end
coutnum=0;
minn=1;

for first=1:ordercopyx-1
      for second=1:ordercopyx
            if D(first,second)==minn&&coutnum==0&&first~=second
                  i=min(first,second);
                  j=max(first,second);
                 if      ordercopy(i,9)+ordercopy(j,9)<11  &&  ordercopy(i,9)~=1  &&
ordercopy(j,9)~=1&&order(ordercopy(i,10),2*MAX+4)==0&&order(ordercopy(j,10),2*MA
X+4)==0
                    for k=1:ordercopy(j,9)
                        order(ordercopy(i,10),2*ordercopy(i,9)+2*k-
1)=order(ordercopy(j,10),2*k-1);
                    end

order(ordercopy(i,10),2*MAX+2)=order(ordercopy(i,10),2*MAX+2)+order(ordercopy(j,10),
2*MAX+2);
%                                                  order(ordercopy(i,10),2*MAX+4)=j+m-
1+order(ordercopy(j,10),2*MAX+1);
                        order(ordercopy(i,10),2*MAX+4)=ordercopy(j,10);

norder(ordercopy(i,10)).num=norder(ordercopy(i,10)).num+norder(ordercopy(j,10)).num
```

```
                    coutnum=coutnum+1;
                    mm=mm-1;
                    for iii=1:2:2*order(ordercopy(i,10),2*MAX+2)-2 %%%货架大小排序
                      for jjj=iii+2:2:2*order(ordercopy(i,10),2*MAX+2)
                        if order(ordercopy(i,10),iii)>order(ordercopy(i,10),jjj)
                        order(ordercopy(i,10),[iii,jjj])=order(ordercopy(i,10),[jjj,iii]);
                        end
                      end
                    end
                     ordercopy(j,10)
                     order([ordercopy(j,10)],:)=[]
                     norder(ordercopy(j,10))=[];
                     mmm=mmm+1;

%----------前置合并订单----------------------
          order([m,ordercopy(i,10)],:)=order([ordercopy(i,10),m],:);
             norderinum=norder(m).num;
             nordericount=norder(m).count;
             norderibtime=norder(m).btime;
             norder(m).num=norder(ordercopy(i,10)).num;
             norder(m).count=norder(ordercopy(i,10)).count;
             norder(m).btime=norder(ordercopy(i,10)).btime;
             norder(ordercopy(i,10)).num=norderinum;
             norder(ordercopy(i,10)).count=nordericount;
             norder(ordercopy(i,10)).btime=norderibtime;
                  end
        end
      end
  end

end
  [orderxxx,orderyxxx]=size(order);
  for i=1:CARNUM
```

```
        carst(1,i)=ncar(i).state;
  end
% %%--------订单规划----------------------------
if(min(carst)==0 && orderxxx>=m)%只有存在空闲车辆且订单中货物多余一件,且剩余订单数大于零时，才进行订单分配
    if order(m,1)~=0&&order(m,2)==0
      if(norder(m).state==0)
      [order,c]=orderdistribution2(m,order,ncar);
       end
       if( order(m,2*MAX+2)==1)%订单里只含有一个商品
                  ncar(c).state=0;     %车空闲
                  order(m,2)=1;
                  N_g=N_g+1;
                  norder(m).state=2;   %订单完成
                  norder(m).etime=cputime;%订单完成时间
                  m=m+1;
                  N_f=N_f+1;
       elseif ( c~=0&&order(m,2*MAX+2)~=1 && ncar(c).state==0)   %给空闲的车分配任务，订单里含有多个商品，处理的货物数小于订单内货物的总数
                  norder(m).state=1;   %订单正在处理阶段
                  norder(m).count=order(m,2*MAX+2)-1; %减去一个打包货架？

                  for i=1:order(m,2*MAX+2)
                        order(m,2*i)=1;
                  end
                        ncar(c).state=1;   %目标是第一个货物取货
                        ncar(c).count=m;     %正在处理第 m 个订单
                        ncar(c).begiX=ncar(c).x;     %车的起点
                        ncar(c).begiY=ncar(c).y;
                        ncar(c).midiX=uuhuojia(order(m,1)).x;   %第 1 个要去的货架
                        ncar(c).midiY=uuhuojia(order(m,1)).y;
                        ncar(c).endiX=uuhuojia(order(m,3)).x;   %第 2 个要去的货架
```

```
                    ncar(c).endiY=uuhuojia(order(m,3)).y;
                    ms(c)= plot(ncar(c).midiX,ncar(c).midiY,'pentagram', 'MarkerSize',
8,'MarkerEdgeColor','b','MarkerFaceColor','b');   %确定中间取货货架的位置并标记成蓝
色
                    lm(c)=plot([ncar(c).x,ncar(c).midiX],[ncar(c).y,ncar(c).midiY]);%
画车与取货货架的线
                    m=m+1 ; %订单号加一
                end
end
end
%% ---------运动规划-----------
for d=1:CARNUM
if(ncar(d).state~=0)%取货阶段

[ncar(d).x,ncar(d).y,ncar(d).v,ncar(d).sing,ncar(d).cosg,lm(d)]=path(h(d),ncar(d).x,ncar(d).y,n
car(d).midiX,ncar(d).midiY,ncar(d).a,ncar(d).v,t,lm(d));
    ncar(d).time=cputime;
    set(lm(d),'visible','off');%取消车与目的货架的线
   lm(d)=plot([ncar(d).x,ncar(d).midiX],[ncar(d).y,ncar(d).midiY]);%画车与取货货架的
线
    if(ncar(d).x==ncar(d).midiX && ncar(d).y==ncar(d).midiY)%到达货点 1
                N_g=N_g+1;
            if ncar(d).state==order(ncar(d).count,2*MAX+2)%%%%达到目的货架
%                 if order(ncar(d).count,2*MAX+1)==0
                norder(ncar(d).count).etime=cputime;%订单完成时间
                norder(ncar(d).count).state=2;%已完成
                ncar(d).a = 0;
                set(lm(d),'visible','off');%取消车与目的货架的线
                set(ms(d),'visible','off');%取消取货货架的星星
                ncar(d).state=0;
                ncar(d).count=0;
                N_f=N_f+1;
```

```
        elseif ncar(d).state==order(ncar(d).count,2*MAX+2)-1
                set(lm(d),'visible','off');%取消车与目的货架的线
                set(ms(d),'visible','off');%取消取货货架的星星
                 ncar(d).begiX=ncar(d).midiX;
                 ncar(d).begiY=ncar(d).midiY;
                 ncar(d).midiX=ncar(d).endiX;
                 ncar(d).midiY=ncar(d).endiY;
                 lm(d)=plot([ncar(d).x,ncar(d).midiX],[ncar(d).y,ncar(d).midiY]);% 画车与取货货架的线
                 ms(d)= plot(ncar(d).midiX,ncar(d).midiY,'pentagram', 'MarkerSize',8,'MarkerEdgeColor','b','MarkerFaceColor','b'); %确定中间取货货架的位置并标记成蓝色
                 ncar(d).state=ncar(d).state+1;


        elseif ncar(d).state<order(ncar(d).count,2*MAX+2)-1%%还没到达
                set(lm(d),'visible','off');%取消车与目的货架的线
                set(ms(d),'visible','off');%取消取货货架的星星
                 ncar(d).begiX=ncar(d).midiX;
                 ncar(d).begiY=ncar(d).midiY;
                 ncar(d).midiX=ncar(d).endiX;
                 ncar(d).midiY=ncar(d).endiY;
                 ncar(d).endiX=uuhuojia(order(ncar(d).count,2*ncar(d).state+1)).x;
                 ncar(d).endiY=uuhuojia(order(ncar(d).count,2*ncar(d).state+1)).y;
                 lm(d)=plot([ncar(d).x,ncar(d).midiX],[ncar(d).y,ncar(d).midiY]);% 画车与取货货架的线
                 ms(d)= plot(ncar(d).midiX,ncar(d).midiY,'pentagram', 'MarkerSize',8,'MarkerEdgeColor','b','MarkerFaceColor','b'); %确定中间取货货架的位置并标记成蓝色
                 ncar(d).state=ncar(d).state+1;
        end
    end
```

```
end
end
```

批次定序：

```
function [order,carnum]=orderdistribution2(m,order,ncar)
%% 未考虑问题：1.多个货物在同一货架  2.车辆有冲突  4.无人车结构体信息包含多个货物
global CARNUM MAX gg;
MAX=10;
load huojia;
largenum=1000000;
fx=0;
gg=zeros();
for i=1:2:2*order(m,2*MAX+2)
    if order(m,i)~=0
        if i~=2*order(m,2*MAX+2)-1
        n1=order(m,i);%第 i 件货物所在货架编号
        n2=order(m,i+2);%第 j 件货物所在货架编号
        distance(1,i/2+1/2)=(huojia(n1,2)-huojia(n2,2))^2+(huojia(n1,3)-huojia(n2,3))^2;
        fx=fx+distance(1,i/2+1/2); %以订单中第 i 个物品为打包点的其余物品与该点路程之和的平方
        end
     if i==2*order(m,2*MAX+2)-1
          n1=order(m,i);%第 i 件货物所在货架编号
          n2=order(m,1);%第 j 件货物所在货架编号
          distance(1,i/2+1/2)=(huojia(n1,2)-huojia(n2,2))^2+(huojia(n1,3)-huojia(n2,3))^2;
          fx=fx+distance(1,i/2+1/2);
        end
    end
end

for i=1:order(m,2*MAX+2)
```

```
    for j=1:CARNUM
        if(ncar(j).state==0)
        n3=order(m,2*i-1);
        gx=(huojia(n3,2)-ncar(j).x)^2+(huojia(n3,3)-ncar(j).y)^2;
        aa(1,j)=gx;
         end
        if(ncar(j).state~=0)
            aa(1,j)=largenum;%货架到无人车的最远距离应该是 46^2+46^2=4232，用以除去不在空闲状态的车辆
            %问题：当所有车均不在空闲状态时，数组中全是 100000，本来这个时候应该等待有车空闲时，在分配任务。
            %但是由于车辆都还没有到达目的地，无车空闲，但是订单还在继续分配，此时应停止订单分配
        end
    end
    overall(1:CARNUM,i,1)=aa;%列存第 i 个货物距所有车的距离
    N(1,i)=min(aa(:));
    if i==1
    gg(1,i)=fx+N(1,i)-distance(1,order(m,2*MAX+2));
    end
    if i~=1
    gg(1,i)=fx+N(1,i)-distance(1,i-1);
    end
    ggper=find(N(1,i)==aa);
    gg(2,i)=min(ggper);
end
    carnum=min(gg(2,find(min(gg(1,:))==gg(1,:)))) ;    %找到最终车辆
    begnum=min(find(min(gg(1,:))==gg(1,:))) ;            %找到车辆对应的起点货架号
    %%
 %-------按订单完成顺序处理-----------------
 order2=order(m,:);
 for i=1:order2(1,2*MAX+2)
```

```
    if begnum+i-1<=order2(1,2*MAX+2)
       order(m,2*i-1)=order2(1,2*(begnum+i-1)-1);
    end
    if begnum+i-1>order2(1,2*MAX+2)
          order(m,2*i-1)=order2(1,2*(begnum+i-1-order2(1,2*MAX+2))-1);
    end
end
%----------------------------
```

致谢

四年的大学时光即将画上句号，这四年，自己收获颇丰，身边遇到了许多值得感谢的人，首先感谢自己的本科导师赵云波教授，除了在课堂上收到了赵云波老师的谆谆教诲之外，在平时的研究和课题方便，也受到了赵老师悉心的指导，让我在毕设期间包括整个大学四年少走了很多弯路。

其次，要感谢在四年指导过自己的其他老师学长学姐，正因为有了他们的帮助，自己才能更加快的进入自己的专业，对自己的专业领域有更多更深的理解和认识。同时，也要感谢和自己的一起参加比赛一起共同进步的同学队友，感谢他们在我压力最大的时候给我的支持帮助，和我一起面对困难。

最后自己也要感谢自己的父母，是他们在各方面做了我最坚强的后盾，让我能在大学四年中毫无顾虑的去追求自己的理想，潜心在学业，并且能在我迷茫的时候给我建议和鼓励，让我有更大的自信继续向前。

时光飞快，大学四年，自己收获了许多也成长的许多，希望在以后的日子，自己能够怀揣这份理想和执着，继续前行，不忘初心！