分类号：F274　　密　　级：公开

U D C：　　单位代码：10142

沈阳工业大学

硕士学位论文

电商企业配送中心订单拣选作业优化研究

学　　号：201820611

作　　者：季爱迅

学 位 类 别：工程硕士

领　　域：物流工程

论 文 类 型：研究论文

2021 年 9 月 1 日

沈阳工业大学硕士学位论文

电商企业配送中心订单拣选作业优化研

Research on the Optimization of Order Picking in E-commerce Enterprise Distribution Center

作　　者：季爱迅　单位：化工装备学院

指 导 教 师：魏晓波 副教授　单位：化工装备学院

协助指导教师：　单位：

　　单位：

论文答辩日期：2021 年 8 月 29 日

学位授予单位：沈 阳 工 业 大 学

摘要

随着互联网信息技术的发展，尤其是全球新冠疫情的爆发，电子商务行业得到了快速发展，同时电商行业接收到的小批量、多品种的客户订单明显增多，对电商企业配送中心的要求越来越高。拣选作业作为配送中心的一项核心业务影响着配送中心的运营成本和运营效率。因此，对电商企业配送中心拣选作业进行优化研究具有重要的现实意义。

配送中心的拣选作业涉及很多的环节，可进行优化的环节也很多，本文主要对配送中心拣选作业中的拣选路径策略和订单分批问题进行研究。

首先，结合电商订单特点，对目前电商配送中心常用的 S 型拣选路径策略进行改进，即结合使用 S 型拣货策略和中点型拣货策略；其次，综合考虑当前电商客户对订单时效高要求的特点，将订单完成时效这一要求引入到订单优化分批中，构建订单拣选总作业时间最小和时效要求内订单完成量最大的订单分批优化模型。接着，分析选择蚁群算法求解的原因，其次考虑到电商配送中心订单数量大，分批的可行方案种类众多，考虑到传统蚁群算法可能会产生算法初期收敛速度较慢和陷入局部最优解的情况，所以对蚁群算法进行了改进，最后采用改进的蚁群算法对订单分批优化模型进行求解设计。

为了验证订单优化的有效性，本文选取了 A 电商企业的配送中心进行应用，首先介绍了优化方案应用的配送中心背景和作业流程现状，获得订单基础数据。通过 Matlab 编程对订单分批优化模型进行求解，寻找出订单拣选的最优方案，最后和 A 电商企业配送中心原有的拣选方案进行比较，从而验证本文所提优化方法及模型的可行性和有效性。

研究结果表明，优化后的拣选方案可以有效减少订单拣选作业的总拣选时间和拣货员行走距离，并在订单时效内拣选完成更多的订单。同时，本文对订单拣选作业的优化研究，也可以为其他类似电子商务企业优化订单拣选作业提供参考和借鉴意义。

关键词：电商配送中心，订单拣选作业，订单分批，蚁群算法

Abstract

With the development of Internet information technology, especially the outbreak of global COVID-19, the e-commerce industry has been developing rapidly. At the same time, the number of customer orders received in the small batch and lots of the e-commerce industry has increased significantly, and the demand for distribution centers of e-commerce enterprises has been increasing. As a core business of distribution center, picking affects the operation cost and efficiency of distribution center. Therefore, it is of great practical significance to optimize the picking operation of distribution centers in e-commerce enterprises.

The picking operation of distribution center involves many links, and there are many links that can be optimized. This paper mainly studies the picking path strategy and order batching problem in the picking operation of distribution center.

Firstly, combined with the characteristics of E-Commerce orders, the S-type picking path strategy commonly used in e-commerce distribution centers is improved, that is, the combination of S-type picking strategy and mid point picking strategy; Secondly, considering the characteristics of current e-commerce customers' high demand for order timeliness, the requirement of order completion timeliness is introduced into order optimization batching, and an order batching optimization model with the minimum total order picking time and the maximum order completion within the timeliness requirement is constructed. Secondly, considering the large number of orders in the e-commerce distribution center and the variety of feasible schemes in batches, the traditional ant colony algorithm may lead to slow convergence speed in the initial stage of the algorithm and fall into the local optimal solution, Finally, the improved ant colony algorithm is used to solve the order batch optimization model.

In order to verify the effectiveness of order optimization, this paper selects the distribution center of a e-commerce enterprise for application. Firstly, it introduces the background of the distribution center and the status quo of the operation process of the optimization scheme, and obtains the basic data of the order. The algorithm is simulated in MATLAB, and the order batch optimization model is solved to find out the optimal scheme of order picking. Finally, it is compared with the original picking scheme of a e-commerce enterprise to verify the feasibility and effectiveness of the optimization method and model

proposed in this paper.

The results show that the optimized picking scheme can effectively reduce the total picking time and the walking distance of the order picker, and pick more goods within the order time. At the same time, the research on the optimization of order picking can also provide reference for other similar e-commerce enterprises.

Keywords: E-commerce distribution center, Order picking, Order batching, Ant colony algorithm

目录

第 1 章 绪论

1.1 研究背景和意义

1.1.1 研究背景

随着信息技术和计算机网络的发展，尤其是全球新冠疫情的爆发，电子商务行业得到了快速发展，也给人们的日常生活提供了更多便利。根据中国互联网络信息中心（CNNIC）发布的《中国互联网络发展状况统计报告》，截至 2020 年 6 月，我国网络购物用户规模已经高达 7.49 亿，占网民整体的 79.7%。据《中国电子商务报告 2019》显示，2019 年，中国网民规模已超过 9 亿人，互联网普及率达 64.5%；全国电子商务交易额达 34.81 万亿元，比上年增长 6.7%，如图 1.1 所示，其中网上零售额达 10.63 万亿元，比上年增长 16.5%，实物商品网上零售额 8.52 万亿元，占社会消费品零售总额的比重上升到 20.7%；电子商务从业人员达 5125.65 万人。根据全球领先的金融服务科技企业 FIS 旗下 Worldpay 发布的 The Global Payments Report 2020（《2020 全球支付报告》），研究了 41 个国家当前和未来的支付趋势，报告指出，疫情加快了全球电子商务的发展趋势，使得中国等市场的现金使用率加速下跌。疫情的爆发，使得中国的电子商务行业市场增长迅猛，电子商务行业也迎来了新的发展高潮。FIS 报告还显示，得益于移动端消费的蓬勃发展，中国电商市场预计将在未来四年增长 70%。

图1.1 2011-2019年中国电子商务交易额

Fig. 1.1 China's e-commerce transaction volume from 2011 to 2019

随着社会经济的发展，人们的消费观念和消费习惯也发生了翻天覆地的变化。对于

电商企业来说，用户已经从单一、大批量需求逐渐转变为多品种、小批量的需求，与此同时，消费者对物流配送的时效性也提出了更严格的要求，导致客户订单呈现出：“单笔订单小总体数量大且品种多、配送地址分布广、响应时间短且严格“等特点。为了适应消费者需求的变化，在当下电商发展的情形下，准时和快速配送已经成为电子商务企业提高自身竞争力和提高客户服务水平的重要焦点，电商企业必须提高对混合小订单的响应时间才能保证自身高效运作，并有效应对各种挑战。

配送中心是从传统的仓库演变而来，传统的仓储主要功能为存储，而配送中心则不同，它集货物存储、订单拣选、配送服务等功能于一体，同时也可以有效将企业的商流、物流、资金流、信息流进行整合，成为了电子商务物流系统的重要组成部分，对整个物流系统的高效运作起着举足轻重的作用。据相关统计，在配送中心内，订单拣选作业时间占到配送中心总作业时间的 40%左右，订单拣选作业的成本占据配送中心总成本比例也高达 60%左右，毫无疑问，拣选作业已经成为配送中心的一项核心业务，直接影响了配送中心和电子商务企业的运营效率和客户服务水平。因此，对拣选作业进行优化是配送中心运作过程中最重要的环节之一，也一直是研究的热点。

1.1.2 研究意义

在当下电商企业发展的情形下，准时和快速配送已经成为客户的电子商务企业提高自身竞争力和提高客户服务水平的重要焦点。为了将货物进行准时和快速的配送，订单拣选作业成为配送中心的瓶颈作业环节。本文从订单拣选作业优化入手，目的是提高订单拣选作业效率，确保电商企业能够及时迅速地响应大量客户的需求，提高企业的效益和市场竞争力。本文的研究意义包括以下两个方面：

（1）理论意义

通过梳理国内外相关文献资料，总结归纳电子商务行业订单拣选作业的研究成果，为优化电商配送中心的订单拣选作业研究提供较为系统的理论支撑。目前国内外对订单分批的研究，大多数以最短路径货或者最小化拣选时间为目标建立订单分批的数学模型，本文在此基础上，依据当前电子商务企业发展的现状，综合考虑订单完成时效这一要求，构建更符合当下电商企业实际的订单分批优化模型，并考虑到传统蚁群算法可能会产生算法初期收敛速度较慢和陷入局部最优解的情况，所以对蚁群算法进行了改进并用改进的蚁群算法求解该模型。

（2）实践意义

本文将研究成果应用在 A 电子商务企业配送中心，以验证模型和算法的有效性和其

工程应用价值，优化后的订单分批作业能够有效减少订单拣选作业的时间和行走路程，并且提高订单的准时完成率，提高订单拣选作业效率。同时，本文对订单拣选作业的优化研究，也可以为其他类似电子商务企业优化订单拣选作业提供参考和借鉴意义。

1.2 国内外研究现状

1.2.1 国外研究现状

（1）订单分批研究现状

Gademann（2001）等[1]以最大限度地缩短订单分批的最大准备时间，对平行通道仓库订单分批问题进行了研究，采用分支定界法求解了订单分批问题的数学模型。

Chen（2005）等[2]从订单之间的关联性来测量订单的接近程度，并且使用 0-1 整数规划模型，在此基础上设计了一个聚类模型来使订单间相关性最大。

Hwang（2008）等[3]研究了低层“人到货”的拣选系统中的订单分批问题，基于聚类分析设计了三种路由策略（穿越策略、返回策略和中点策略）下的订单分批算法，最终证明该算法在总作业时间和分批批次数量等方面都比现有算法有优势。

Tsai（2008）等[4]构建了以行走距离成本和延迟完成惩罚成本这两个成本之和最小为目标的订单分批优化模型，并采用了多阶段遗传算法对其进行求解。

Henn（2011）等[5]研究在人工订单拣选系统的批次分拣问题时，为了减少一系列订单的总延迟时间，构造数学模型，并采用了两种元启发式算法进求解，迭代局域搜索算法和基于属性的爬山算法两种，通过使用一些列不同的订单和仓库环境进行了模拟实验，最终证明了其优越性。

Temel（2015）等[6]在处理订单分批问题（OBP）时候，分别考虑了穿越策略、返回策略和中点回转策略，并引入了混合整数线性规划(MILP)公式，同时，还使用了高效的局部搜索算法(ILST)，通过对标准和随机生成的实力进行了实验，最后证明了证明了混合整数线性规划和局部搜索算法对求解订单分批问题都具有很出色的性能。

Chen（2016）等[7]从仓库中多人分批拣选作业可能产生拥堵的角度出发，提出了一种蚁群优化（ACO）算法，然后根据室内定位和信息共享技术，协调 ACO 规划的路线，可以有效解决仓库的拥堵问题。

Scholz（2016）等[8]在研究订单分批时，首次提出了同时考虑所有次级问题的方法。引入了该问题的数学模型，允许解决小问题实例。对于较大的实例，将呈现可变邻里下降算法。通过数值实验，证明该算法提供了质量优良的解决方案。此外，还表明，同时

解决上述次级问题的办法可被视为提高分销仓库业务效率的重要来源。

Menendez(2017)等[9]把订单分批问题看成是仓库运作管理范畴的优化问题。以拣货总时间最少为目标，提出几种基于变邻域搜索的策略。

Xue（2018）等[10]将最小化物流机器人的总拣取和行进时间以及最小化所有拣选站之间的时间为目标，建立订单分批模型，使用改进的聚类算法对模型进行求解。

（2）拣选路径策略

Hwang（2005）等[11]在对拣选路径的研究中发现，在穿越式策略、中点回转策略和返回策略三种策略中，穿越式策略在订单中品项数量较多时表现效果较好，中点回转以及返回策略在品项较少的情况下能够减少行走路径，取得较为理想的结果。

Pratik（2010）等[12]以拣选路径最短为目标，同时考虑了作业时间，研究了多目标的掠选路径优化问题。

Chan（2011）等[13]在对拣选路径的研究中，将存储策略、路径策略和拣选密度三个因素进行组合，比较了不同组合因素对拣选路径的影响。

Ncan（2015）等[14]将订单批次问题与 S 型策略、中点返回型策略和返回型策略三种启发式路径策略相结合，构建出混合整数线性规划模型。

Lu（2016）等[15]提出了一种动态拣选路径优化算法，结合算例实验验证其有效性，结果表明在一定条件下该算法优化性能强于静态拣选路径算法和启发式路径优化算法。

Dijkstra（2017）等[16]利用精准的数学模型描述了穿越式策略、中点回转策略、返回策略，同时联合优化了储位分配问题以提高订单拣选效率。

Yener（2019）等[17]通过数据挖掘了解到目前仓库内订单拣选作业时间和距离的平均数据，将商品存储位置之间的距离和商品的关联度作为影响因素，并借鉴了 VRP 模型中的车辆路径方法来进行了拣选路径的优化研究。

1.2.2 国内研究现状

马士华（2004）[18]在研究配送中心的订单分批问题时候，引入了一种“延迟制造”理念并提出了基于时间延迟的动态时窗分批策，并用计算机仿真实现了这一策略。研究结果表明，该策略能有有效地消除现有拣选系统中的等待时间，提高配送中心的拣选作业效率。

郝彤彤（2014）[19]在对京东商城 3C 仓储中心的拣货体系中采用种子算法对订单分批模型进行求解，她提出了 6 个依据来进行种子订单的原始筛选和 6 种原则来建立新的订单集，最后用京东商城的数据与先到先服务进行对比，结果表明该算法能减少订单的

行走距离。

王旭坪（2014）等[20]从联合优化的角度出发，对传统的拣选模型进行优化和改进，不再是单一的优化订单分批，而是与配送系统一起进行联合优化，最终求解结果表明这种联合优化可以可以显著提高拣选作业的作业效率，同时也对提高配送资源的利用率产生了一定的积极作用。

陈方宇（2014）[21]采取了新的求解思路优化拣选作业，创新地将实时订单也输入到待拣选的系统中，并设计算法同时优化了订单分批和路径设计，最终结果表明，对于电商物流来说，这种优化效果显著，且订单的完成时效性也大大提高。

邹霞（2018）[22]参考国内外现有的文献后，综合考虑订单密度等相关影响分批结果的因素,在改进传统分批时窗模型时，创新地引入“订单行”这一概念，考虑了订单行约束的限制，最终求解结果使得批次间效率差异降低，稳定性增强。

郜振华（2019）[23]针对双区型多设备配送中心的订单分批作业，以拣选距离最短为目标,建立了订单分批问题的数学模型。对该问题进行萤火虫算法设计，并通过算例进行萤火虫算法分批。将萤火虫算法分批与按订单分批、先到先分批进行对比分析，验证萤火虫算法分批的优越性。

徐鹏（2019）[24]在对装配式建筑建造过程中关于 PC 构件分批进行配送研究过程中，建立了最小化总配送时间的分批模型，并采用蚁群算法对 PC 构件分批模型进行求，达到了定量优化的目的解，为装配式企业的分批问题提供了参考。

黄敏芳（2020）等[25]以提高大型网上超市订单分拣方法的科学性为目标，针对订单拣选、集货复核、包装这一作业流程，基于 JIT 装配流水线思想，建立大型网上超市订单成组分拣的优化模型与求解方法。在构建订单分拣流程中各工序作业时间计算模型的基础上，建立了订单分批与排序的联合优化模型。围绕缩减求解空间的思想提出两阶段启发式优化方法。通过应用实例分析和灵敏度分析证明了算法的有效性。

冯辰吉（2020）等[26]将订单寻仓和分批拣选两个问题进行了联合研究，针对电商订单的特点设计了三阶段启发式算法，将多商品订单和单商品订单分开处理，通过交换关联订单的处理顺序，降低订单履行成本和包裹数量；生成初始订单分批解后，以一定规则使得解在领域内搜索，改进订单分批问题的解。实验结果表明，提出的算法能够提升订单履行的整体效率。

秦馨（2021）等[27]在研究订单分批优化时，以最小化货物搬运次数为目标构建了数学模型，并采用遗传算法对该模型进行求解，通过设定初始种群、选择、交叉、变异等一系列遗传操作设计得到了分批结果，其次将分批优化的结果与不分批和先到先分批进

行了对比，结果证明该订单分批优化模型可以有效减少拣选作业中货物的搬运次数。

（2）拣选路径策略现状

华红艳（2010）等[28]将蚁群算法用来优化拣选路径，在算法设计的时候，通过改进扰动分量，有效地提高了算法的运行效率。

朱杰（2011）等[29]在物品随机存储情况下，分别构建返回型与 S 型拣选方式下拣选距离的随机模型，研究结果表明，S 型拣选方式在拣选数量较多的订单时可以有效减少行走距离，这一结论可为实际物流配送中心提供一定的决策支持。

朱文真（2011）[30]在研究拣选路径的优化问题时候，将遗传算法和其他禁忌搜索算法进行有机结合，并有效得出了立体仓库拣货的最优路径。

庞龙（2012）等[31]使用蚁群算法和遗传算法相结合的方法对仓库拣选作业进行优化，首先利用蚁群算法生成初始种群，然后使用遗传算法进行优化求解，并通过仿真来表明该模型是可行的。

卢子甲（2013）等[32]针对配送中心订单拣选的行走路径进行了优化研究，以优化行走距离为目标，通过遗传算法来求解中点型和穿越型路径拣选方法的行走距离，并通过实验进行了比较。

李建斌（2014）等[33]基于 TSP 对双区型仓库的拣货路径进行建模，利用蚁群算法、模拟退火和禁忌搜索对模型求解，最终得出结果，当品项数量较少时应采用模拟退火算法求解，而当品项数量大时采用蚁群算法效果更好，且只需迭代一次，所得结果已应用于某大型电子商务企业，优化效果明显。

李栋栋（2015）[34]在研究双区仓库拣货路径优化时，考虑到拣货车的载量有限，以拣货车的行驶距离最小为目标，建立了路径优化问题的数学模型，以有效减少拣货车的行驶里程，这对于提高双区型仓库的作业效率，降低配送的总作业成本起到了很好的效果。

刘建胜（2017）等[35]采用蚁群算法、混合蚁群算法求解最短拣货路径，结果表明适合于大型仓储拣货路径优化。

纵观国内外参考文献，对订单分批和路径优化一直是研究的热点，且成果颇丰，对学习订单分批和路径优化问题有着非常重要的参考意义，但仍存在一些不足之处：比如订单分批大多数以拣选距离最短或者拣选时间最小为单一目标；其次是，许多研究对拣选路径进行优化时，往往使用各种元启发式算法进行优化，然而现在很大一部分电商配送中心仍采用“人到货”的拣选系统，在这种系统下，实行标准化是非常重要的，智能算法对路径优化往往得出的路径复杂且多编，不便于拣选人员记忆，还会增加拣选作业

的出错率，影响拣选作业效率。

1.3 研究的主要内容和方法

1.3.1 研究的主要内容

（1）第 1 章 绪论

介绍本论文的选题背景和研究意义，并对国内外文献进行梳理和总结，其次提出本论文的主要研究内容，确定本论文所涉及的研究方法，并绘制本论文的框架图。

（2）第 2 章 相关理论基础

阐述本论文分析研究所需的相关概念和基础理论，主要包拣选作业概述、拣选路径策略概述和订单分批概述，为后续电商企业配送中心的订单拣选作业优化提供相应的理论基础。

（3）第 3 章 电商企业配送中心订单拣选作业优化

基于当下电商行业的发展现状，首先，对目前电商配送中心常用的 S 型拣货拣选路径策略进行改进，其次，综合考虑订单完成时效这一要求，构建订单拣选总作业时间最小和时效要求内订单完成量最大的订单分批优化模型。接着分析选择蚁群算法求解的原因，其次考虑到电商配送中心订单数量大，分批的可行方案种类众多，采用传统蚁群算法可能会产生算法初期收敛速度较慢和陷入局部最优解的情况进行改进，尝试对蚁群算法进行改进，最后采用改进的蚁群算法对订单分批优化模型进行求解。

（4）第 4 章 案例应用

本文选取了 A 电商企业的配送中心进行应用，首先介绍了优化方案应用的配送中心背景和作业流程现状，获得订单基础数据。在使用蚁群算法时，本文将用到 Matlab 编程语言来实现多目标函数的求解，使最后得到的订单拣选的方案较为理想，最后和 A 电商企业配送中心原有的拣选方案进行比较，从而验证本文所提优化方法及模型的可行性和有效性。

（5）第 5 章 结论和展望

对论文的研究内容进行归纳总结，其次指出本论文的不足之处，并对未来所需继续进行的研究内容和方向进行展望。

以下是论文框架图，如图 1.2 所示。

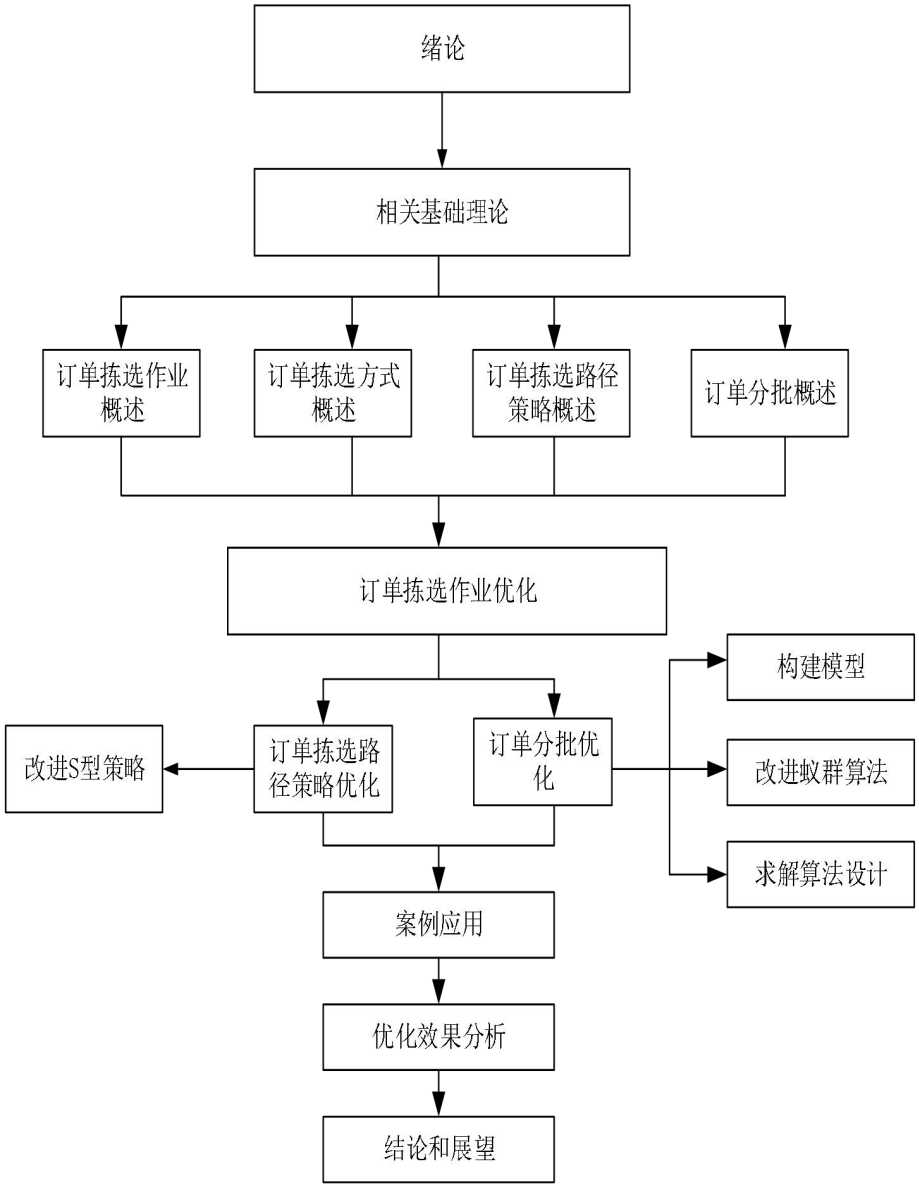

图1.2 论文框架图

Fig. 1.2 The framework of the paper

1.3.2 研究方法

（1）文献研究法：本文主要通过查阅订单拣选作业领域的各种专著和期刊杂志以及中国知网数据库的优秀硕博士论文等，对研究所需的理论进行系统性的总结和归纳，从而帮助更加深刻的了解所研究内容，也为本文的创新提供了理论基础。

（2）案例研究法：本文选取了 A 电子商务企业配送中心系统进行应用，以验证模型和算法的有效性和其工程应用价值，优化后的订单分批作业能够有效提高 A 公司配送的订单拣选作业效率。

（3）数学建模法：本文依据当前电子商务企业发展的现状，综合考虑订单按时完成拣选量最大和拣选总作业时间最小这两个目标，构建更符合当下电商企业实际的订单分批优化模型，并设计改进蚁群算法求解该模型。

（4）编程求解法：本文设计的优化数学模型，需要使用改进蚁群算法求解，在使用改进蚁群算法时，使用计算机编程软件上是较为高效的一种求解方法。在使用改进蚁群算法时，本文将用到 MATLAB 编程语言来实现多目标函数的求解，使最后得到的结果较为理想。

第 2 章 相关理论基础

2.1 拣选作业概述

2006 年，我国发布的国家标准《物流术语》（GB/T 18354-2006）将“拣选”定义为：接收到订单需要或者出库需求，从物品储存场所拣出物品的作业。拣选作业的目的是快速从仓库中拣出订单中的物品，完成客户订单。一般仓库或者配送中心的拣选作业的流程如图 2.1 所示。

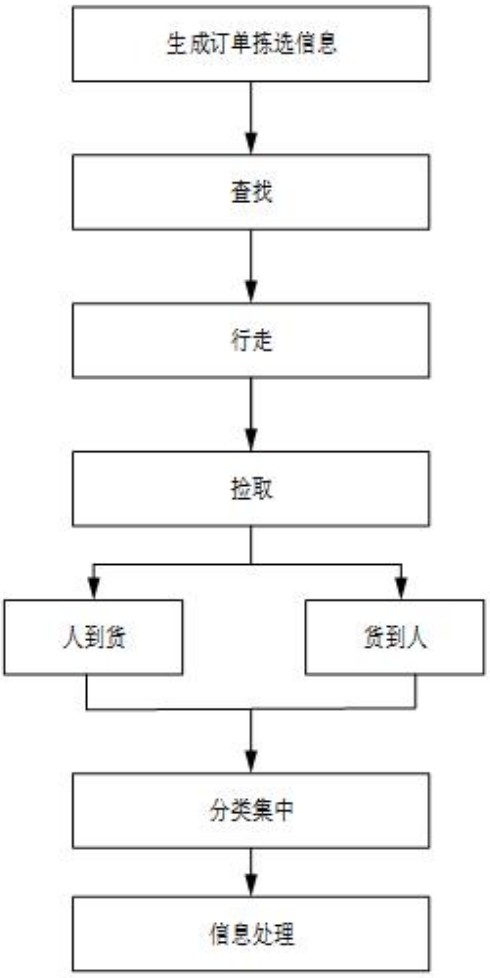

图 2.1 拣选作业流程图

Fig. 2.1 picking operation flow chart

订单拣选作业是一直以来都是电子商务企业非常重视的物流环节之一，企业在客户提交订单信息或者需求配送计划后，了解订单货物的基本信息，包括品名、数量，然后

准备的对应上该货物所在的货位位置，准确而高效地完成货物的捡取，并按照企业事先制定的规则，对货物进行分类和集中，最终将货物拼装然后发货。由于拣选作业的高准确性和高准时性要求，使得拣选作业一直以来都是配送中心花费人力、财力和时间最多的作业环节。拣选作业的根本目的就是准确而高效地集合客户订单的所有品项，主要的目标就是降低拣选作业的成本并减少拣选的时间。要实现这一目标，配送中心必须依据客户的订单信息，选择适合拣选的设备，并依照自身的运营状况，采用一定的拣货策略或者方式，只有这样，才可以实现降低成本和减少时间。不仅如此，还可以更快的响应客户的需求，提高物流服务和企业服务的满意度。

从自动化角度，订单拣选系统可以被分为两种，人工拣选系统和自动分拣系统。从目前国内电商发展现状来看，客户需求品项多且繁杂，有很大一部分企业仍然采用人拣选系统。从移动方向角度来看，拣选系统也可以被分为 2 中，一种是“人到货”，一种是“货到人”。就目前的电商配送中心而言，“人到货”的拣选系统实践程度更高。“人到货”顾名思义就是人去找货物，往往是拣货员按照订单信息，带着相应的拣选设备到制定的货位进行货物的捡取，当拣选设备容量达到最大时再到制定地点卸下拣选的货物。所以，本文研究“人到货”的电商企业配送中心的订单拣选作业。

整个订单拣选作业流程大致可以划分成三个阶段：订单拣选信息的收集、移动和拣选以及分拣和集中。第一个阶段是指有关每个批次中要处理的订单，以及每个批次要拣选和货物信息的收集。第二阶段是指拣货员如何根据拣选的货物来完成订单，在货位前确认待拣选的商品的品项和数量的所有任务，并完成货物的提取，然后将拣选完成的货物移动到货物的集中收集点，订单拣选作业流程中花费的时间往往是这一个阶段。第三阶段是指，将完成拣选的货物可能还需要按订单类别进行分类的集中的过程。整个拣选作业所需的时间主要集中在，拣货员的行走时间、寻找品项时间和提取货物时间。

据统计，在配送中心内，订单拣选作业时间预计占到配送中心总作业时间比例的 30%~40%左右，订单拣选作业的成本约占据到配送中心总运营成本的比例也高达 60%。也存在部分研究表明，订单拣选作业占据配送中心运营成本甚至可以高达 75%[36]。从这些数据可以得出结论，拣选作业已经成为配送中心的核心业务，它对决定了配送中心和电子商务企业的运营效率和客户服务水平的高低至关重要。因此，对拣选作业进行优化一直以来都是配送中心运营过程中非常重要的环节之一。

2.2 订单拣选方式概述

配送中心每天接收大量客户的订单，每个订单都有各自把不同的特点和要求，目前

所有拣选作业优化的目标就是，如何将这些不同特点和要求订单以更快更准备的处理完毕，根据拣选数量的多少可以将拣选方式分为两种，单订单和多订单拣选。根据拣选的形式的不同，又可以将其划分为波次拣选和区域拣选。

单订单拣选：是当配送中心接收到一个客户的订单后，就立即开始拣选，拣货人员只需要完成该订单所有品项的拣选，完成拣选后放到货物的待配送点即可。单订单拣选的主要比较明显，由于订单前置期相对较短，所以很能快响应每个订单并完成拣选，当出现紧急订单进行插单的情况，也可以快速响应。其次，因为是按照每个订单单独进行拣选的，因此拣选完成后就无需在进行后续的二次分拣，所以人工操作起来比较简单和方便。但单订单拣选的缺点也很明显，如果订单包含的品项数比价多且比较繁杂的时候，且品项数重复率出现很低的时候，单订单拣选会大幅度增加拣货员的行走距离，使得订单拣选作业时间更长，并且影响订单拣选的完成效率所以，单对单拣选的方式局限性较强，适合 SKU 数量比较少的配送中心或者客户订单商品种类单一的情况。

多订单拣选方式：是指在拣货人员开始拣选货物之前，按照适合企业自身的规则对货物进行合并，然后再安排拣货人员捡取货物，这种方法在货物合理合并的基础上可以大大缩小拣货人员的拣选距离以及拣选时间。多订单拣选方式的优点也很明显，可以通过合并相同的货物或者有一定相似度的商品，减少重复捡取所需行走的距离和时间，达到提高拣选作业效率的目的。其次，多订单拣选比单对单拣选多一个货物的在分拣过程，可以在二次分拣时检查所拣选货物是否正确并及时发现错误的货物进行调整。可以有效降低货品的错发率，也有利于提高企业的物流服务水平。多订单拣选还有一个优点就是，可以提高拣选设备的利用率。使用多订单拣选策略进行拣选的主要缺点是，往往是等订单达到一定的数量才会进行订单的合并操作，越往前的订单前置等待时间越长[37]。如果遇到“紧急插单”的特殊情况，配送中心或许没办法第一时间响应。其次，由于合并方式的多样性和不确定性导致多订单拣选的策略在实际操作时候较为复杂。

波次拣选方式：又称波段拣选方式，该方法需要将到达时间不同的订单固定在同一个时间段内一起进行拣选作业，具体如何划分这个合理的时间段，主要的影响因素就是这个时间段到达订单数量和多少以及不同订单的拣选行走路线。配送中心在使用波次拣选方式时，往往会依据订单是否紧急，以及订单响应的长短进行合并成，有可能合并成一个批次有可能合并成多个批次，然后在安排拣选人员进行拣选，在实际运用过程中，该方法往往是和单订单或者多订单拣选方式搭配着来运用，从而提高订单拣选作业的效率。

区域拣选方式：又称订单分区拣选方式。他主要依靠仓库的布局决定，将一整个拣

选区域划分成每个单独的小的拣选区，往往一个拣货员只需要负责一个拣选区域。当这个拣选区的和货物全部捡取完毕后，再进入下一个拣选区进行拣选，如此反复，一直到订单货物全部拣选完毕。这种分区拣选方式方式的优点很明显，每个拣货员负责拣选的区域较小，能够提高拣货员对所负责区域内的商品熟悉程度，提高拣货效率，降低拣货的差错率[38-42]。分区的规则依据不同的情况来划分，表 2.1 总结了目前较为常见的分区拣选规则。

表2.1 分区拣选规则

Tab. 2.1 partition picking rules

名称	规则
货品性质分区	根据货品原有的性质划分，如一般存储品和特殊存储品
拣货单位分区	根据拣货的单位进行划分，如单品货品区和箱式货品区
拣选形式分区	根据不同的拣货形式进行划分，如 ABC 拣货模式
工作分区	根据不同的拣货场地划分，如串行分区和并行分区

2.3 订单拣选路径策略概述

1959 年，美国学者 Dantzig G B 和 Ramser J H 在研究油库与该油库加油站之间的油车运输队的最优路径时，首次提出“拣货路径策略”这一概念。随着二战后物流产业的快速发展，学者们对于这一领域的研究也逐渐增多。目前配送中心常用的拣选路径策略主要有 S 型策略（穿越策略）、中点策略、返回策略和最大间隙策略[43]。

（1）S 型策略（穿越策略）

如果某条巷道上有待拣选的货物时，拣货员在拣选完该货物后需要继续前行，穿越整条巷道之后，才能进入下一条巷道，等拣货员拣选完需要拣选的货物后，返回到出入口，在拣选过程中拣货员会走遍每一条需要拣选货物的巷道，如图 2.2 所示。由于行走路径像一个大写的“S“，所以也叫 S 型策略。由于 S 型策略行走的路径较为简单，且在订单拣选作业中容易被拣货员执行，所以目前“人到货”的配送中心采用 S 型策略较多。由于 S 型策略的特点，所以它比较适合订单货物品项较多、货架分布散且广等情况，这时使用 S 型策略可以兼顾到每一个货位[44]。

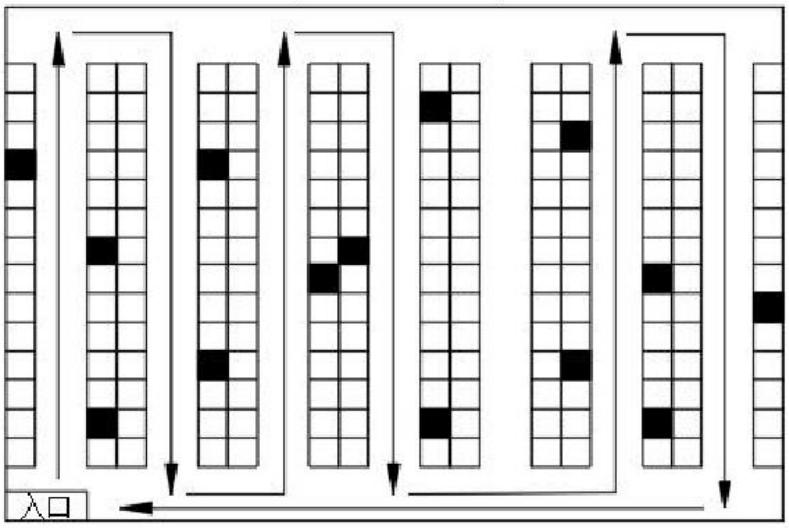

图2.2 S型策略

Fig. 2.2 s-strategy

（2）中点策略

配送中心在使用中点策略时，首先要按垂直于巷道的方向将拣选区分为对称的两部分。拣货员从入口开始拣选货物，先拣选完其中一个区域需要拣选的货物，然后再进入另一半区域进行拣选，拣选完所有待拣选货物后回到入口，如图 2.3 所示。当待拣选货物品项分布在各个巷道的两端时，可以在一定程度上减少拣货员的拣选行走总路程，所以中点策略较适合把出入库频次较高的货物品项放在距离起始点（I/O）较近的货位上。

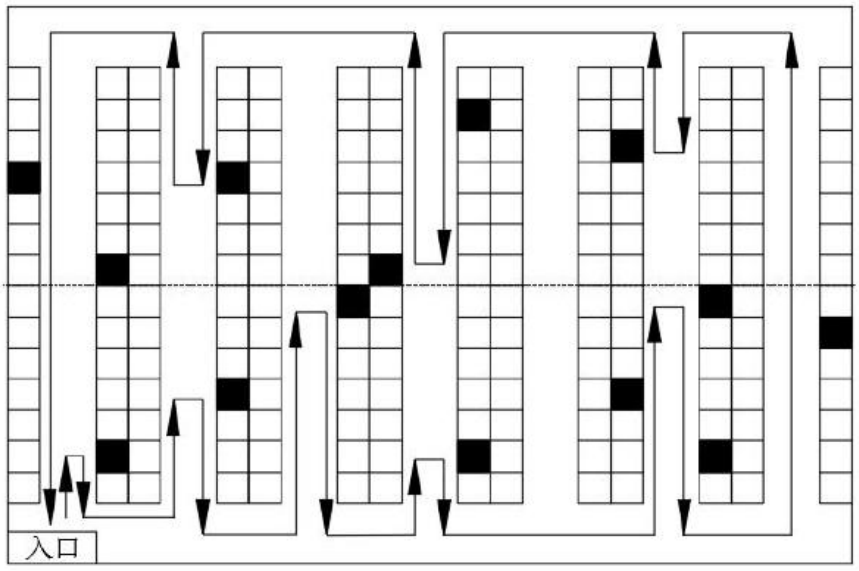

图2.3 中点策略

Fig. 2.3 midpoint strategy

（3）返回策略

拣选人员从入口出发，首先从靠近入出口的巷道的一端进去，把该巷道中所有需要捡取货物拣选完了之后，沿原路返回进入巷道的位置进入，然后，再进入下一个有待拣选货物的巷道，当拣货员拣选完所有货物品项后，再次回到入口[45]。如图 2.4 所示。

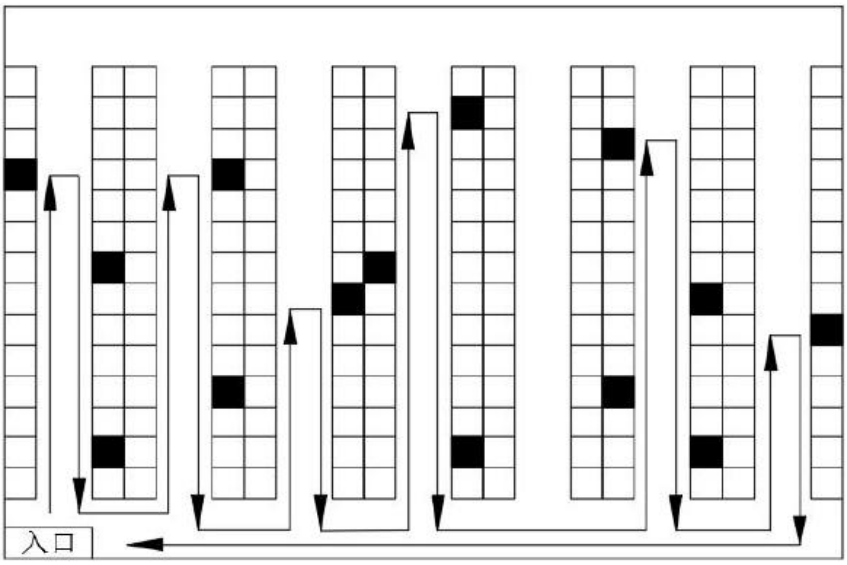

图2.4 返回策略

Fig. 2.4 return strategy

（4）最大间隙策略

拣选人员必须完全两穿越两侧的两条巷道，其他的巷道并不完全需要，这就是最大间隙原则。拣选人员将该巷道的所有货物都捡取后从巷道出来，并且使得未达到的巷道之间间隙最大化即可。例如，在某个巷道内，从当前巷道的当前货位完成拣货后，当等待拣货的下一个货位与巷道中的当前货位之间的距离小于从当前货位到巷道一端的位置时，将从该巷道中的下一个货位进行拣货，否则就折返进到下一个需要拣选货物的巷道进行拣选。如图 2.5 所示。

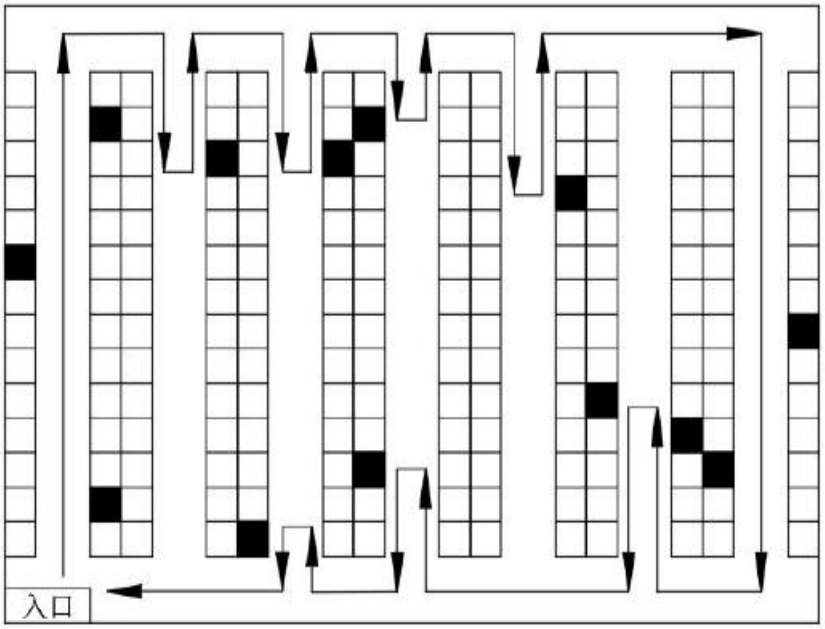

图2.5 最大间隙策略

Fig. 2.5 maximum clearance strategy

2.4 订单分批概述

目前大多数配送中心在拣选货物时，采取较多的是先到先服务方式，这种方式容易被理解且操作难度不高所以在实际应用中也比较广泛。但是当今客户对订单处理时效的要求越来越高，以及配送中心对订单响应速度要求也越来越严格，再加上电商订单量的快速增加，这些原因都给配送中心的作业效率带来了很大考验和挑战。大批量的客户订单源源不断地下达给配送中心，传统的先到先服务当时的已经明显不能满足当下电商企业配送中心的需求。目前，大多数电商企业的配送中心的订单都有订单时效的限制，无法承受因为过长的拣选距离和过多的拣选时间，导致的订单延误和订单延误所带来的各种风险和损失。因此，“订单分批”便产生了。简单来说订单分批就是将多个订单集合起来，再将其划分为多个批次进行拣选作业，每个批次包含一定数量的订单，这个数量可以是“一”也可以是“多”，然后分在同一批次的订单就可以一起进行拣选。

订单分批的根本目标就是缩短拣货过程中的总行走距离和总拣选时间。将含有相同货物的订单划分到同一批次后集中拣选作业，可以避免重复走相同的路线和减少寻找相同货物所浪费的时间，有效提高配送中心的拣选作业效率。因此，如何将多个订单进行合理的分配到一个批次中，显得尤为重要。订单分批适用于订单量较大的配送中心，订单的特点是货物种类多且单品数量较少。研究表明，通过订单分批不仅可以有效地降低

拣选作业的时间，还可以有效减少拣货员捡取货物的行走路程，减轻拣货员的工作量，同时还可以降低配送中心的整体物流成本。因此，订单分批的优化一直以来都是各个学者和专家研究的热点。

2.4.1 传统的订单分批策略

根据订单规律的不同，订单分批的策略也有所不同，目前传统的订单分批策略主要有以下几种：

（1）总和计量分批策略

总和计量分批是先需要将所有订单中的每件商品按照品类进行汇总，然后根据总和的数量进行拣选，以缩短拣选作业中人员的的行走距离。由于订单间不同货物进行合并时候，合并种类和方案非常多，这就需要一个更强大的分类系统来进行分类合并。就当下该策略的应用情境而言，固定配送点相互之间的配送业务往往采用该策略。在操作中，首先需要花费时间对订单货物进行收集然后汇总，最后安排拣货员一起进行拣取，全部拣选完成后，再对拣选完成的货物按照订单进行分类分拣[46]，最后安排运输车进行装货配送。

（2）时间窗分批策略

时间窗分批策略是以一个时间段为时间节点，将该时间段内的订单进行集中汇总，然后组合成一个批订单，再安排拣选人员进行拣选作业。具体批次数是由总作业时间和时间窗长短来决定的。该策略适用于拣选订单比较紧急的情况下。如何确定时间窗长短的需要考虑的因素较多，比如客户能接受的最大等待时间，订单分批的预处理时间，等等。通常来说，时间窗的划分不可以太长，否则会导致订单前置时间过长，影响订单的完成时效。有学者证明，时间窗分批策略配合分区拣选策略使用，可以取得较好的效果。

时间窗分批又可以细分为两种：静态时间窗分批和动态时间窗分批两种，动态时间窗分批的应用比静态时间窗分批的应用更加灵活，尤其是对订单时效要求严格的配送中心来说，动态时间窗的分批结果往往比静态时间窗的分批结果更有效。

（3）固定订单数量分批策略

固定订单数量分批策略又称先到先服务分批（FCFS）策略，这种策略是根据哪个订单先到达，哪个订单先分批的原则，然后按照订单下达的货物进行数量的集合和排序[47]。当下达的订单数量到达一个固定的数量（一般配送中心都是以拣货车的最大载重量为限制）时再进行货物的分批，具体分批的次数是由订单的总数量和进行订单分批的数值共同决定的。当订单总数量不多，或者订单中货物的种类比价少时，可以采取该分批策略，

如图 2.6 中所示。

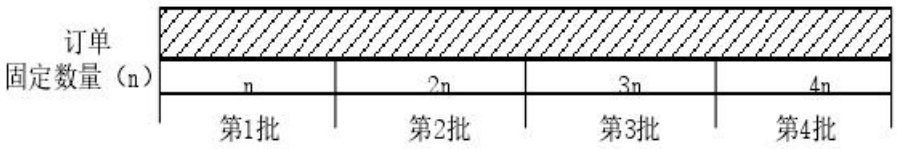

图2.6 固定订单量分批方式

Fig.2.6: Fixed order quantity method of order batching

为了更直观的了解传统分批策略的适用情况和优缺点，传统分批策略比较如表 2.2 所示：

表2.2 传统分批策略比较

Tab. 2.2 comparison of traditional batching strategies

分批策略	适用情况	优点	缺点
总和计量分批策略	固定配送点之间的配送任务	分批方案较为简单和操作	需要提前汇总订单信息，无法实时对订单进行分批
静态时间窗分批策略	订单到达时间有规律性	批次划分简单	在订单随机到达情况下效果较差
动态时间窗分批策略	订单随机到达	等待时间较短，订单批次划分准备	划分依据较为复杂
固定订单数量分批策略	订单数量以及货品数量较少	分批操作简单	大批量订单分批效果较差

2.4.2 新型的订单分批策略

随着国内外专家学者对订单分批问题的投入研究，订单分批的策略已经不再局限于传统方法进行分批了。由于配送中心订单量较大，计算分批问题的过程往往非常繁琐和复杂，这就意味着传统的分批方法无法用于配送中心同时处理大批量到达的订单量，尤其是当遇到“紧急插单”的情况时，配送中心往往无法第一时间进行响应。所以国内外学者也开始利用启发式算法来对订单进行分批研究，如何更加合理的对订单进行分批，也逐渐成为拣选作业新的研究方向。接下来将会对常用于订单分批策略的启发式算法进行分类论述，主要包括种子算法、节约算法、数据挖掘分批策略和元启发式算法。

（1）种子算法

种子订单的实现步骤是根据客户订单的相关特点，选择几个订单作为种子订单，然后依据到达的订单数量、订单中待拣选货物的货位分布情况、订单的拣选距离等因素制定相关的分批策略，最后，其他剩余的订单也按照这个分批策略一一的添加到每个种子订单的批次中，直到所有的订单都完成之后才算处理完毕。这种分批策略的特点是订单

的批次是依次生成的，这种种子算法适合于配送中心接受的订单数量比较庞大的时候使用，且效果较好。

（2）节约算法

节约算法最早是在研究车辆路径问题时提出的。该算法的主要目标就是节约作业流程中的各种时间和路程，这样就达到了拣选作业优化的效果。节约算法是通过优化局部的效果而达到整体优化的效果，即缩短每个订单批次拣选路径或者每个订单批次内拣选的时间，最终实现订单整体优化的结果。订单的相关特征，比如订单内货品的数量、订单所在货位位置、以及订单达到的时间间隔长短等等因素，都会对节约算法的分批切入点产生不同程度的影响，所带来的分批的结果也会有所不同。从国内外参考文献来看，节约算法往往和时间窗分批策略共同使用，可以得到较好的分批结果。

（3）数据挖掘分批策略

首先依据数据挖掘的方法分析所有订单之间的信息，然后依据分析结果制定订单之间的，接着利用所制定的关联规则确定客户订单之间的相似性，其次，根据相似度确定一个用来做参考标准的订单，然后找到如何使剩余订单和这个参考标准订单相互之间的关联程度，从最大化关联度角度出发将剩余的其他订单挨个分配到这些参考订单中，直到所有的订单完成分批处理。如何合理地确定各个订单之间的关联性和相似性，一直以来都是研究的难点和重点，一个更可行的想法就是利用数据挖掘来确定不同订单相互之间的相关性和相似性。

（4）元启发式算法

元启发式算法是在传统的启发式算法基础上改进而来的。元启发式算法包括遗传算法、蚁群算法、模拟退火算法和变领域搜索算法等群体智能算法。利用这些智能算法进行订单分批灵活性较高，并且相对于传统的启发式算法，元启发式算法可以处理大批量订单、和紧急插单情况，或者订单品项较为复杂的情况用元启发式算法优化订单分批问题也一直是目前订单分批问题的研究重点和热点。

2.5 本章小结

阐述本论文分析研究所需的相关概念和基础理论，主要包拣选作业概述、拣选路径策略概述和订单分批概述，为后续电商企业配送中心的订单拣选作业优化提供相应的理论基础。

第 3 章 电商企业配送中心订单拣选作业优化

3.1 订单拣选作业优化思路

据了解，电商配送中心的成本占其整个物流环节成本的很大比重，并且对于电商配送中心来说，订单的拣选作业是其物流系统中最为复杂的一个环节，且搬运成本、拣选作业劳动力和时间占比都较大，如图 3.1 所示，同时是出错率较高的一项作业。

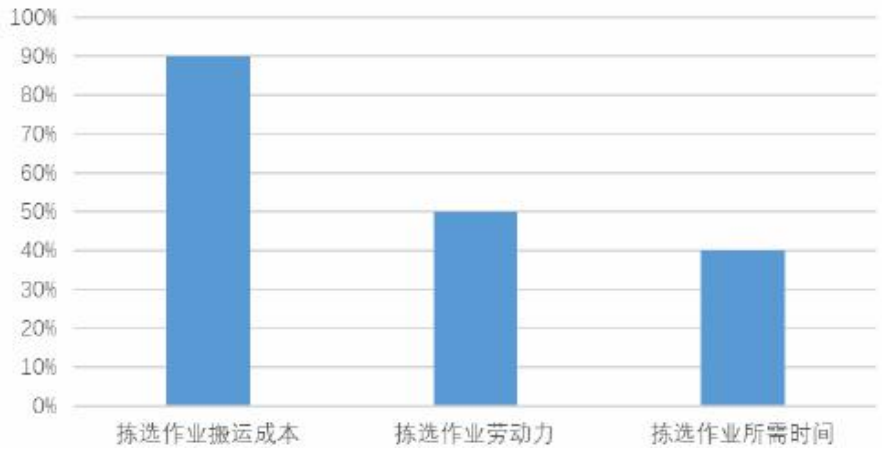

图 3.1 电商配送中心拣选作业搬运成本、劳动力和时间占比

Fig. 3.1: The proportion of picking operation cost, labor and time in e-commerce warehouse

据研究，拣选作业的搬运成本占整个电商配送中心的搬运成本的比列近乎高达 90%，拣选作业的劳动力占整个配送中心劳动力的 1/2 左右，拣选作业时间也占据了整个电商配送中心的 40%左右。由于拣选作业的高搬运成本、劳动力和时间占比，拣选作业一直以来都是配送中心作业流程中的重点环节，拣选作业效率的高低直接影响着对电商配送中心的订单完成率。所以，如果想要提高电商配送中心的运营效率，必须从订单的拣选作业入手，通过提高拣选作业的效率，才能有效提高订单商品出库作业的速度，满足客户对订单物流高时效和高准确率的要求。

合理的订单分批方式可以有效减少订单拣选作业的总时间和总行走路程，以及从接收订单到订单出库配送的总时间，也有助于平衡拣选人员的工作量，减少拣选作业中的总行走距离。而订单拣选路径的设计，也与拣选人员的总行走距离和总行走时间密切相关，综上所述，优化订单分批方式、合理优化拣选路径策略方式可以帮助减少订单拣选作业的时间和拣选行走距离，提高拣选作业效率。因此，本文中将主要从订单拣选路径策略优化和订单分批优化两个方面进行探讨。

3.2 订单拣选路径策略优化

由于当下电商环境下客户订单总体数量大且差异性较大，需要拣选的商品分布没有很强的规律性，导致拣货时品项分布比较分散且密度较大，不同订单和批次的货物分布往往差异较大，单独采用任何拣货路径策略都是不合理、不合适的。综合考虑当下电商企业的发展现状和各拣选路径的特点，采用组合拣货路径侧策略。由于 S 型策略行走的路径较为简单，且在订单拣选作业中容易被拣货员执行，所以目前“人到货”的配送中心采用 S 型策略较多。由于 S 型策略的特点，所以它比较适合订单货物品项较多、货架分布散且广等情况，这时使用 S 型策略可以兼顾到每一个货位。当待拣选货物品项分布在各个巷道的两端时，可以在一定程度上减少拣货员的拣选行走总路程，所以中点策略较适合把出入库频次较高的货物品项放在距离起始点（I/O）较近的货位上。综合考虑 S 型策略和中点策略的特点，本文在对订单拣选路径进行优化时，结合使用 S 型策略和中点策略这两种策略，即引入中点策略对 S 型策略进行部分环节优化，如图 3.2 所示。

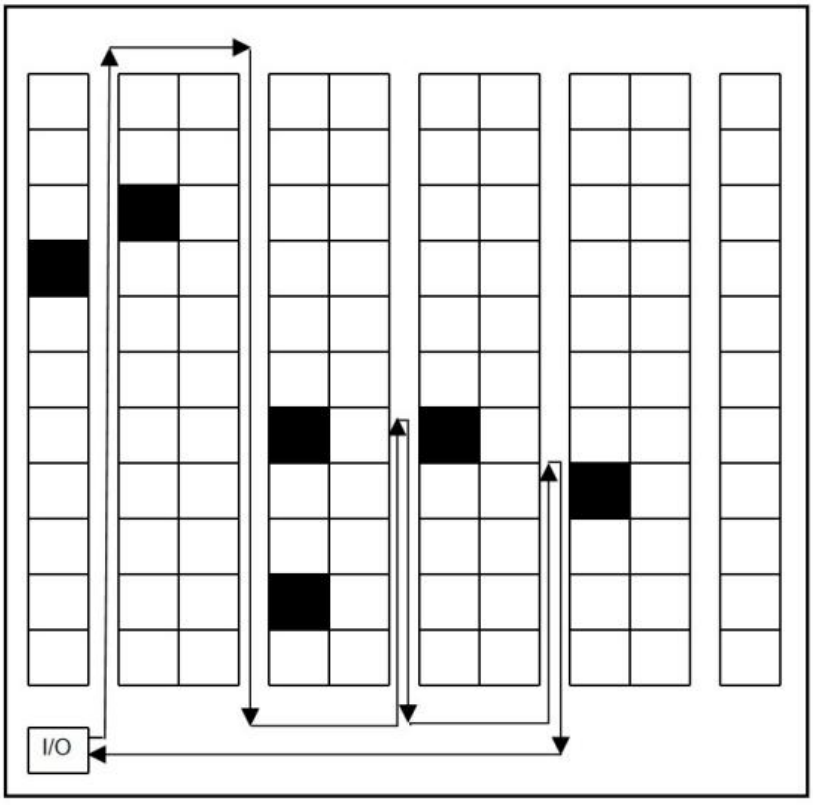

图3.2 改进S型策略拣选路径

Fig. 3.2 Improved s-strategy picking path

改进 S 型策略具体步骤如下：

步骤 1：拣选时，如果该待拣选货物货位的位置距离该巷大于该巷道长度的 1/2 时，拣选人员拣选完货物后无需返回，直接穿过该巷道今年入下一个货物所在巷道即可而

步骤 2：如果该待拣选货物货位所在位置小于该巷道总长度的 1/2 时，拣选人员拣选完货物后拣选后无需再穿过一整条巷道，直接返回到进入该巷道的位置就可以，然后接着对下一个货物进行拣选，然后去到到下一个待拣选货物的所在巷道。

步骤 3：当某个巷道内没有需要被拣选的货物，直接跳过该巷道，进入下一个拣选巷道。

对于同一个批次的拣选订单，重复循环以上三个步骤，直到取完该分批批次的所有订单为止。改进后 S 型策略作业的流程，如图 3.3 所示：

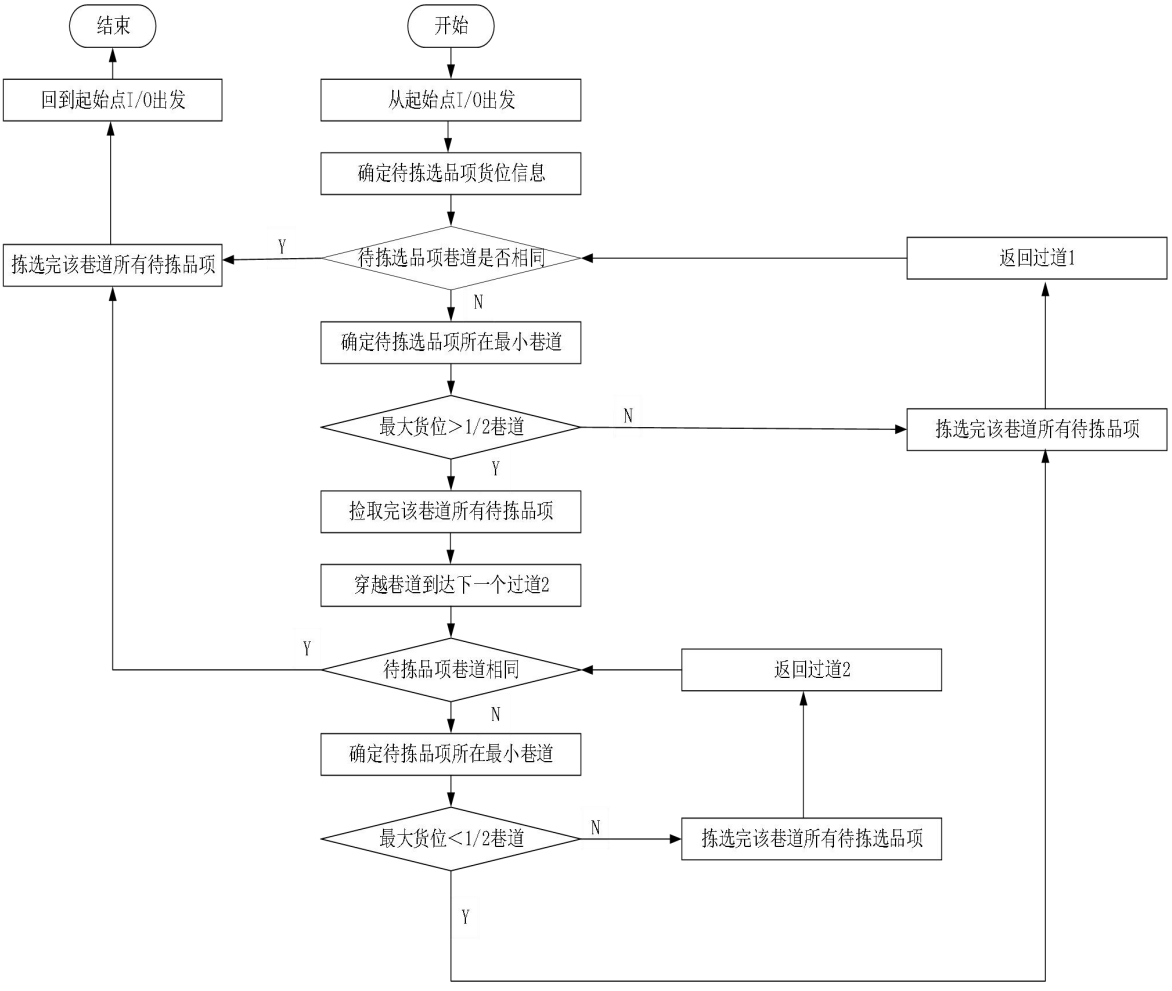

图3.3 改进S型策略作业流程如图

Fig. 3.3 flow chart of improving s-strategy

本文采用 Matlab 编程进行拣选距离的计算，主要代码展示如下：

（1）S 型策略 ：

```
 if rem(length(c1),2)==0    &&    length(c1) ~= 0    %  是偶数
      distance = length(c1) * (2*road+road_L);      %拣货走的巷道长度
      distance = distance + 2*((c1(1,length(c1))*2 - 1) * shelf_R + (c1(1,length(c1))-1) *
road_R);
 else if rem(length(c1),2)~=0    &&    length(c1) ~= 0
      distance = (1 + length(c1)) * (2*road+road_L);      %拣货走的巷道长度
      distance = distance + 2*((c1(1,length(c1))*2 - 1) * shelf_R + (c1(1,length(c1))-1) *
road_R);
      else
          distance = 0;
      end
```

（2）改进 S 型策略

从靠近出入口（I/O）一段进入巷道时：

```
c5 = abs(c4-0);
 if length(c4) == 0
      w = 0;
 else
      w = max(c5);
 end
 if w > half     %过半  则穿越巷道
      Y_S = -1 *    Y_S;
      distance = distance +    road_L + 2 * road;
 else
      if w <  货架数量  *  货架货位数
          distance = distance +    2 * w * shelf_L;
      else
          distance = distance +    2 * w * shelf_L + 2 * road;
      end
 end
```

从远离出入口（I/O）一段进入巷道时：

```
c5 = abs(c4-Lmax);
 if length(c4) == 0
      w = 0;
 else
      w = max(c5);
 end
 if w > half     %过半 则穿越巷道
      Y_S = -1 *    Y_S;
      distance = distance +    road_L + 2 * road;
 else
      if w <  货架数量  *  货架货位数
           distance = distance +    2 * w * shelf_L;
      else
           distance = distance +    2 * w * shelf_L + 2 * road;
      end
 end
```

3.3 订单分批优化模型构建

3.3.1 问题描述

本文主要优化人工拣选系统，“人到货”拣选方式的订单分批，首先，拣选员工需要根据待拣选单的信息，从出入口进入货物的拣选区，按照拣选路径策略确定的顺序依次来完成拣货作业，捡取完所有需要拣选的货物之后，拣选员工再次回到出入口，防止货物并完成接下来二次分拣和打包配送。拣选员工金进行拣选作业时，往往使用拣货小车进行货物的放置，考虑到拣货小车是有体积和质量限制的，因此，分批后的每一个订单批的总体积和总质量都不能超过拣货车的最大容量。

随着电商快速发展和普及人群广泛，顾客的年龄的差异性和个人喜好差距都较大，所以，对商品的偏好和喜好是多元化和多样化的，这使得顾客订单中的商品呈现多种类、小规模的特征。在竞争激烈的电商环境下，这种订单模式的变化对配送中心订单处理效率提出了更高的要求。顾客对订单响应时间也更加敏感，能否快速和准确的收到自己下单的货物已经成为顾客判断该电商企业服务水平高低的重要影响因素。因此，通过合理

分批可以不仅可以减少订单拣选时间和企业响应时间，还可以提高订单完成率，从而到达一种更高的服务水平。图 3.4 为具有完成期限的订单拣选系统示意图。

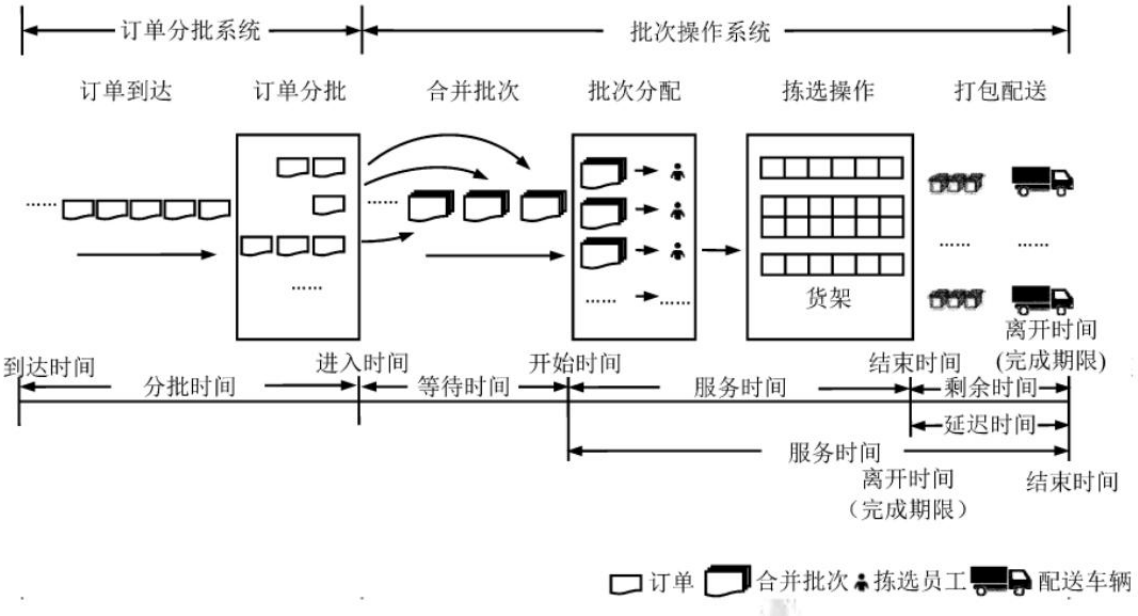

图3.4 具有完成期限的订单拣选系统示意图

Fig. 3.4 schematic diagram of order picking system with deadline

通过上述分析，可以用数学语言来描述具有订单时效要求的分批优化问题：在一个电商配送中心内，有 a 名拣选员工，在一段时间内有 b 张有时效要求的订单需要进行拣选作业，且每个订单上都包含有一个或者多个品项，且存放每种品项的货位位置是固定和已知的。按照一定的规则或者要求，将 b 张订单分成若干批次，然后将分批后的订单批次分配给这 a 名拣选员工，再安排每位拣选员工的拣选批次顺序，使得所有订单拣选的总作业时间最小、订单时效要求内完成的订单数量最大。

3.3.2 模型假设

本文研究的配送中心是人工进行拣选作业的，是劳动密集型的非自动化配送中心。考虑到计算的复杂性，未简化的模型是极为复杂、非常难以求解的，在不影响本质的情况下，必要的假设可以减少不必要的因素对模型的影响，并使问题简单、清晰，易于理解和突出重点。因此，本研究将对订单分批问题模型做出如下假设：

（1）同一批次的订单需要在一次拣选作业中完成

（2）拣货过程中不存在缺货问题

（3）所有拣选设备的规格型号都相同

（4）拣货通道够宽敞，多个拣选员可以同时通过，不会出现拥堵情况

（5）储位够大，多个拣货员可以同时提取同一储位的 SKU，不需要等待

（6）不考虑紧急插单的情况

3.3.3 数学模型构建

目标函数：

（1）拣选总作业时间最小

由前文可知，拣选作业时间占整个配送中心总作业时间的比例约为 40%左右，是目前配送中最耗时的作业，因此，在对订单进行分批的时候其中一个目标就是使得拣选总作业时间最小，同时本文研究的订单分批优化问题是在一段时间内需要拣选固定数量的订单，所以，当拣选作业总时间减少，就意味着拣选作业效率的提升，因此构建总拣选作业时间最小目标函数如式（3.1）所示：

$$f_1 = min\sum_{h\in H}\sum_{k\in K}\sum_{s\in S} T_k y_{khs} \tag{3.1}$$

其中，

$$T_k = T_{kw} + T_{kp} \tag{3.2}$$

$$T_{kw} = L_k / V_w \tag{3.3}$$

$$T_{kp} = Z_k * V_p \tag{3.4}$$

（2）订单时效要求内订单完成量最大

当前电商发展情形下，顾客对订单响应时间也更加敏感，能否及时收到自己下单的货物已经成为顾客判断该电商企业服务水平高低的重要影响因素，所以本文将订单完成时效这一要求引入到订单优化分批中，即订单时效要求内订单完成量最大，其目标函数如式（3.5）所示。

$$f_2 = max\sum_{i\in I} O_i \tag{3.5}$$

其中，

$$O_i=\begin{cases}1, \sum_{k\in K}\sum_{h\in H}\sum_{s\in S}x_{ik}y_{khs}(D_i-DT_{hs})\geq 0\\ 0, \text{其}\end{cases} \tag{3.6}$$

约束条件：

$$\sum_{k\in K}x_{ik}=1,\ \forall i\in I \tag{3.7}$$

$$\sum_{h\in H}\sum_{s\in S}y_{khs}=1,\ \forall k\in K \tag{3.8}$$

$$\sum_{k\in K}y_{khs}\leq 1,\ \forall h\in H,\ \forall s\in S \tag{3.9}$$

$$x_{ik}=\begin{cases}1, \text{订单i是否在第k批次中拣选}\\ 0, \text{否则}\end{cases} \tag{3.10}$$

$$y_{khs}=\begin{cases}1, \text{批次k分配给拣选员h第s批拣拣}\\ 0, \text{否则}\end{cases} \tag{3.11}$$

$$\sum_{i\in I}q_ix_{ik}\leq Q,\ \forall k\in K \tag{3.12}$$

$$\sum_{i\in I}m_ix_{ik}\leq M,\forall \mathrm{k}\in K \tag{3.13}$$

$$DT_{hs\text{-}1}+\sum_{k\in K}T_ky_{khs}\leq DT_{hs},\ \forall \mathrm{h}\in H,\forall k\in K \tag{3.14}$$

$$\sum_{k\in K}T_ky_{kh1}\leq DT_{h1},\ \forall \mathrm{h}\in H \tag{3.15}$$

$$DT_{hk}\geq 0,\ \forall \mathrm{h}\in H,\forall k\in K \tag{3.16}$$

式中，各字母代表的含义如下：

I:顾客订单集合

K:可行批次集合

H:拣选员集合

S :拣选员拣选订单批次顺序集合

Z_k:批次 k 包含的品项数量

T_k ：拣选批次 k 总的作业时间

T_{kw}：拣选批次 k 的行走时间

T_{kp}：寻找和提取批次 k 所有品项所需的时间

L_k ：完成批次 k 所需行走的距离

Vw ：拣选员的行走速度

Vp ：拣选员寻找和提取品项速度

O_i：按时完成拣选的订单量

D_i：订单 i 的完成期限

DT_i：订单 i 的完成时间

DT_{hs}：拣选员 h 第 s 批次的拣选完成时间

q_i：订单 i 所有品项体积

m_i：订单 i 所有品项质量

Q：拣货车的最大体积限制

M :拣货车的最大质量限制

x_{ik}：决策变量，订单 i 否在第 k 批次中拣选

y_{khs}:，决策变量，批次 k 是否分配给拣货员 h 第 s 批拣选

其中，各公式代表的含义如下：

（3.1）式：使得拣选总作业时间最小的目标函数，

（3.2）式：拣选批次 k 总的作业时间，包括拣选批次 k 的行走时间以及寻找和捡取批次 k 中所有品项所需的时间，

（3.3）式：拣选批次 k 行走时间的计算公式

（3.4）式：寻找和提取批次 k 所有品项所需时间的计算公式

（3.5）式：使得订单时效要求内订单完成量最大的目标函数

（3.6）式：订单在时效要求内完成，O_i 取值为 1，否则为 0

（3.7）式：每个订单只能分配给一个批次

（3.8）式：每个批次只能分配给一个拣选员的一个拣选顺序位置

（3.9）式：拣选员的每个拣选批次顺序最多只能安排一个订单批

（3.10）式：决策变量 x_{ik} 取值范围，取值为 1 或 0

（3.11）式：决策变量 y_{khs} 取值范围，取值为 1 或 0

（3.12）式：批次 i 的总体积不能超过拣货车的体积限制

（3.13）式：批次 i 的总质量不能超过拣货车的质量限制

（3.14）式：任意两个批次的拣选作业不重合

（3.15）式：计算批次完成时间

（3.16）式：完成时间的非负性

3.4 订单分批模型求解

3.4.1 求解算法分析和选择

订单分批问题已被证明是典型的 NP-hard 题，目前学者对于此问题的求解提出了很多算法方案，大致可以分为三类：精确算法、传统启发式算法以及元启发式算法。

在求解小规模的订单分批问题时，可以使用精确算法，当模型中有很多约束条件和变量时，使用精确算法不能保证在合理的时间得到最优解，所以该方法并不适合求解电商配送中心的订单分批模型问题。

传统启发式算法包括种子算法、节约算法和两阶段算法，但使用传统启发式算法求解分批模型时，容易陷入局部最优解，无法得到整体最优解。在传统启发式算法上，又产生了一些改进，于是便有了元启发式算法，元启发式算法是多种算法的总称，包括模拟退火算法、遗传算法、蚁群算法、禁忌搜索算法等等。元启发式算法灵活度较高，适用于处理订单数量庞大的情况，使用全局搜索技术可以大大降低陷入局部最优的可能性。可见，该算法也可以用于订单分批问题。多数学者基于该算法涉及求解，如采用模拟退火算法、禁忌搜索算法、蚁群算法和遗传算法等算法。

表3.1 常见求解分批问题算法对比

Tab. 3.1 comparison of common algorithms for solving batch problems

算法	优点	缺点	适用范围
模拟退火算法	收敛速度快	不一定趋于全局最优解	优化已知路径
蚁群算法	不依赖初始解，收敛速度先快后慢	初代计算速度较慢	多目标优化问题
遗传算法	全局搜索能力强	对初始种群依赖大	复杂的优化问题
禁忌搜索算法	最接近最优解	计算量大，求解速度慢	软时间窗的 VRP

蚁群算法鲁棒性较强，且与其他的元启发式算法相比而言，对初始方案要求并不高，即蚁群算法的求解结果不依赖于初始方案的选择，也不需要在搜索过程中人工进行调整。其次，蚁群算法的参数数目少，设置简单。同时，蚁群算法在组合优化问题上优于粒子

群算法、模拟退火算法与遗传算法，适合解决本文研究的订单分批的多目标模型，并且蚁群算法应用到电商配送中心的人工订单分批问题上的文献还比较少，为此，本文将使用蚁群算法求解订单分批模型。

3.4.2 蚁群算法（ACO）介绍

蚁群算法（ACO）算法通过模拟蚂蚁的觅食行为来寻求最优路径，是由 M.Dorigo 提出的一种仿生学群体智能算法。

蚂蚁在寻找食物源的过程中，会在行走路径上释放一种叫“信息素”的激素，以便其他蚂蚁可以在一定的时间和距离范围内发现。当越来越多的蚂蚁沿着某些路径行走时，这些路径上积累的信息素也会增加。蚂蚁在选择路径时，选择这些路径的概率也就变得更高，最后，导致选择走这些路径的蚂蚁越来越多，积累的信息素也就越来越多，所以这是一种正反馈机制，这种会自主选择最短路径的过程就被称蚂蚁的自催化行为。从某一只单独的蚂蚁角度来看，它并不是自己去寻找到最短路径，只是根据路径上的信息素浓度进行概率选择，但从整个蚁群系统角度来看，它们最终在宏观上达到了寻找最优路径的效果，这就是群体智能的一种客观表现形式。这也是蚁群算法求解问题的基本思想，其中每只蚂蚁行走的路径代表一个可行解，整个蚁群即构成解空间，随着迭代过程的进行，蚂蚁会集中在最短路径上，最短路径所对应的就是优化解就是这个问题的全局最优解。

数学模型如下：

蚂蚁 k （ $\mathrm{k}=1,2,3,\ldots,\ \mathrm{m}$ ）在运动过程中，运动的方向主要由每条路径上的信息素浓度决定。$tabu_k$（禁忌表）可以用来记录第 k 只蚂蚁当前经过的所有节点；这个存放节点的集合会随着蚂蚁的运动动态调整。在算法的搜索过程中，蚂蚁会只能地选择下一步所要走的路径。

设 m 表示蚂蚁总数量，用 d_{ij}（ $\mathrm{i,j}=0,1,\ldots,\ \mathrm{n}-1$ ）表示节点 i 和节点 j 之间的距离，$\tau_{ij}(t)$ 表示在 t 时刻 ij 连线上的信息素浓度。在初始时刻，随机放置这 m 只蚂蚁，每条路径上的初始信息素浓度是相同的。在 t 时刻，蚂蚁 k 从节点 i 转移到节点 j 的状态转移率如式（3.17）。

$$p_{ij}^{k}=\begin{cases}\dfrac{\tau_{ij}^{\partial}(t)\eta_{ij}^{\beta}(t)}{\sum_{k\in allowed_k}\tau_{ij}^{\partial}(t)\eta_{ij}^{\beta}(t)}, & j\in allowed_k\\ 0, other\end{cases} \tag{3.17}$$

其中，$allowed_k=\{c-tabu_k\}$ 表示蚂蚁 k 接下来可以选择的所有节点，c 为全集节点集合；α 为信息启发因子，代表轨迹在算法中的相对重要程度，反应路径上的信息量对蚂蚁路径所起的影响程度，该值越大，蚂蚁之间的协作性就越强；β 可称为期望启发因子，代表能见度在算法中的相对重要性。η_{ij} 是启发函数，在算法中表示蚂蚁由节点 i 转移到节点 j 的期望程度，通常可取 $\eta_{ij}=1/d_{ij}$。算法运行的时候，每只蚂蚁将根据式（3.17）进行搜索前进。

在蚂蚁运动过程中，为了避免在路上残留过多的信息素而使启发信息被淹没，在每只蚂蚁遍历完成后，要对残留信息进行更新处理。由此，在 $t+n$ 时刻，路径 ij 的信息素。

由于信息素更新策略有所不同，学者 M.Dorigo 研究发现了三种不同的蚁群算法模型：蚁周系统（Ant-cycle system，ACS）、蚁量系统（Ant-quantity system，AQS）和蚁密系统（Ant-density system，ADS），这三种模型的增量 $\Delta\tau_{ij}$ 不同

（1）ACS 模型中

$$\Delta\tau_{ij}^{k}=\begin{cases}\dfrac{Q}{L_k}, & \text{第k只蚂蚁在本次循环中经过ij}\\ 0, & \text{其它}\end{cases} \tag{3.19}$$

（2）AQS 模型

$$\Delta\tau_{ij}^{k}=\begin{cases}\dfrac{Q}{d_{ij}}, & \text{若第k只蚂蚁在}t\text{与}t+1\text{之间经过}ij\\ 0, & \text{其它}\end{cases} \tag{3.20}$$

（3）ADS 模型

$$\Delta\tau_{ij}^{k}=\begin{cases}Q, & \text{若第}k\text{只蚂蚁在时刻 t和t+1之间经过 ij}\\ 0, & \text{其它}\end{cases} \tag{3.21}$$

Q 是一个常数，反应蚂蚁留下的轨迹数量，L_k 表示 k 只蚂蚁在本次循环中所走路径的长度。这三种模型的信息素更新方法略有不同。在ACS模型中，释放的信息素量与路径距离有关，当所有蚂蚁完成一次遍历后，他们将更新所有路径上的信息素，这是一种全局更新方法。而在AQS模型和ADS模型中，蚂蚁从一个节点移动到另一个节点后，会更新它刚经过的路径上的信息素，然后搜索下一个路径地址，即利用的是局部信息。AQS模型中信息素更新的计算方法与两个地址之间的路径距离有关，而ADS模型更新的信息素数量是固定的，与路径距离无关。在一系列标准测试问题上运行试验表明，“ ACS ”算法的性能于AQS和ADS这两种算法。因此，对蚂蚁系统的研究正朝着更好地了解“ ACS ”特征方向发展[48-56]。

以经典 TSP 问题为例，蚁群算法流程如图 3.5 所示：

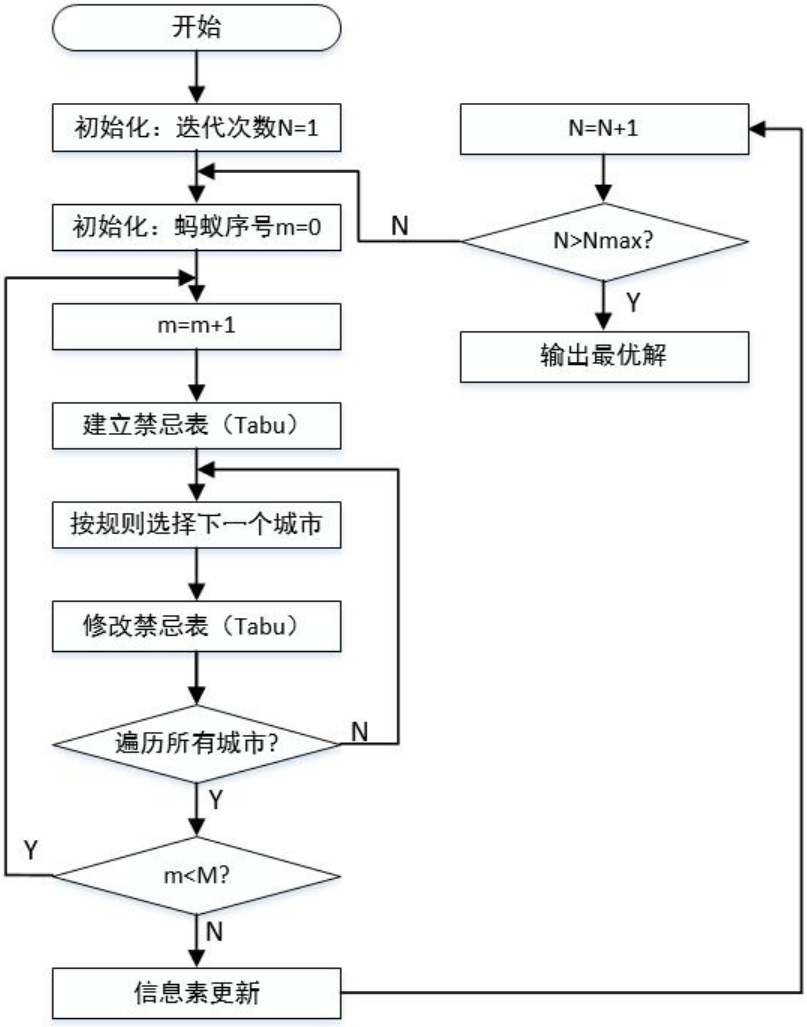

图3.5 传统蚁群算法流程图

Fig. 3.5 flow chart of traditional ant colony algorithm

求解该问题的蚁群算法的基本步骤如下：

第一步：对蚁群中蚂蚁数量m、启发函数因子α和β、信息素挥发率ρ、信息素总量Q、最大迭代次数NC_{max}等有关参数进行预处理，同时还需要根据数据的规模和格式进行相关程序的编写。

第二步：随机生成各蚂蚁的初始所在城市，再通过信息素浓度或者随机策略逐个选择下一个城市，直至各蚂蚁遍历所有城市。

第三步：通过各蚂蚁的路径记录，计算各蚂蚁所选路径的总长度，再找出本次迭代中所选路径最短的蚂蚁及其路径，并将其标记为本次迭代过程中出现的最优解，并根据本次迭代中最优解的总长度和路径对各城市节点的信息素浓度进行更新。

第四步：判断当前迭代次数是否达到最大迭代次数，若未达到最大迭代次数，返回步骤 2，重复步骤 2、3、4；若达到最大迭代次数，则程序终止运行，寻优结束。

第五步：根据各次迭代中的最优路径及其长度，找出整个迭代过程中的最优解，并输出相应的路径和总长度。

3.4.3 改进蚁群算法

就电商配送中心的拣选过程作业而言，往往每次接收到的订单数量都较大，所以需要的分批总数也较大，也就意味着分批的可行方案种类众多，但由于传统蚁群算法具有算法初期收敛速度较慢，易陷入局部最优解等缺点所以本文首先对传统的蚁群算法进行改进。

在订单分批问题中，每次迭代过程中各蚂蚁通过对应的动作选择策略，选择动作，生成一种分批方案。在设计动作选择策略时，考虑到分批问题的可行解数量众多，如果采用传统蚁群算法的启发函数来选择分批方案，由于初始迭代时生成的一组可行解具有随机性，之后迭代若采用启发函数来选择动作，生成分批方案，则算法可能会提前收敛，得到目标函数的某个局部最优解（极小值），而不是全局最优值（最小值）。为了增大蚁群的搜索范围，对每次迭代中的蚂蚁随机分成两批，一批采用类似启发函数的方式，使用信息素进行动作选择，生成新的方案，剩下的部分蚂蚁采用随机策略在全局搜索，生成新的分批方案。

由于每次迭代中有部分蚂蚁的路径（即分批方案）是采用随机策略生成的，所以在信息素更新时，传统蚁群算法的一个蚂蚁更新一次信息素的信息素局部更新策略将不再适用。这里改成对每次迭代中的所有蚂蚁得到的分批方案的目标函数值排序，取得目标

函数值最小的蚂蚁的匹配方案，对该只蚂蚁的路径进行信息素的增加，即每次迭代只对本次迭代中方案最好的蚂蚁路径进行信息素更新的信息素全局更新策略。

3.4.4 基于改进蚁群算法的模型求解算法设计

本文构建的订单拣选优化模型有两个优化目标，不同的配送中心可以对不同的目标的给予不同的重视度，可视具体运营情况来决定或者需要改善的方向来确定，因此本文采用权重系数法将多目标函数转换成单目标函数进行求解，如式（3.22）：

$$minf = \xi_1 \times f_1^{'} + \xi_2 \times f_2^{'} \tag{3.22}$$

其中，权重系数ξ_1、ξ_2的值根据个目标的重要度来权衡，$\xi_1 + \xi_2 = 1$。$f_1^{'}$、$f_2^{'}$是将f_1、f_2转换为最小值进行[0，1]标准化处理，其转换如式（3.23）、式（3.24）：

$$f_1^{'} = \frac{f_1 - f_1^{min}}{f_1^{max} - f_1^{min}} \tag{3.23}$$

$$f_2^{'} = \frac{f_2^{-1} - (f_2^{max})^{-1}}{(f_2^{min})^{-1} - (f_2^{max})^{-1}} \tag{3.24}$$

式中，$f_1^{\min}$、$f_1^{\max}$、$f_2^{\min}$、$f_2^{\max}$分别是目标函数f_1和f_2的最小值和最大值。

蚁群算法中设计到一系列的参数，且这些参数均对于程序的最终结果优劣有一定的影响，因此，有必要根据实际问题仔细考虑释放的参数设置。蚁群算法在寻找最优解的过程中具有一定的随机性，有可能出现后续迭代结果可能劣于当前迭代结果的情况，以经典 TSP 问题为例，出发点的选择上就具有一定的随机性。蚁群算法通过这个初始点的选择将全局寻找最优解转化为寻找局部最优解。如何设置合理的参数，就是在“全局”和“局部”之间建立一个相对平衡点，使算法尽可能的搜索全局最优解，而不是陷入局部最优解，以此保证解的最优性。同时，算法要具有良好的收敛性，尽快完成收敛，节省寻优时间。

改进蚁群算法中需要进行设定的主要参数如下：

（1）蚂蚁数量m

如何确定蚂蚁数量m对结果很重要，如果m过大时，会导致搜索过的路径上信息素变化区域平均，这样就很难找到最优解，且收敛的速度也会比较慢慢；如果m过小时，可能会导致部分分批方案未被搜索到，最终其信息素浓度减小到 0，也就意味着可能会陷入局部最优，过早收敛。据对以往文献的学习和实验参考，最终确定蚂蚁数量m=25。

（2）信息素挥发率ρ

当蚂蚁完成对所有订单的遍历后，需要更新一次残留信息，也就是所谓的信息素更新，即弱化原来的旧信息，避免因残留信息过多而淹没启发信息。因此，信息挥发率 ρ 的设置也会影响算法的全局搜索能力和收敛速度。本研究从优化结果及收敛速度两个维度对信息素保留率 ρ 取值进行对比实验，得出参数信息素挥发率 ρ 取值为 0.5 时得出的最优结果和迭代次数都较为合理，因此本文最终选取信息素挥发率 ρ =0.5 进行模型求解。

（3）信息素总量 Q 和初始信息速浓度设置

Q 为蚂蚁遍历解空间内所有点后最终遗留的信息素总量。Q 的取值越大，每条路径上累积的信息素浓度就越大，且在较短路径上遗留的浓度将更大，由此吸引更多蚂蚁向此路径聚集，并继续分泌信息素，影响其他蚂蚁，算法收敛速度将因此加快。根据对以往文献的学习和实验参考，本研究选取选取信息素总量 Q =100，初始浓度为 3，进行后续模型求解。

（4）最大迭代次数 NC_{max}

一般来说，优化算法的终止条件有最终优化结果满足一定的阈值或者达到一定的循环次数。蚁群算法的最优解一般是未知的，所以蚁群算法一般采用迭代次数来控制算法的终止。如果最大迭代次数过下，可能导致算法收敛未完成，陷入局部最优，所得结果并不是全局最优解；如果最大迭代次数设置过大，会导致计算机资源的浪费。一般来说，通过先设定一个试验值，通过查看程序的收敛轨迹来判断算法是否完成收敛，来更改最大迭代次数。最终本文选取的最大迭代次数 NC_{max} =100。

基于改进蚁群算法的模型求解流程如图 3.6 所示：

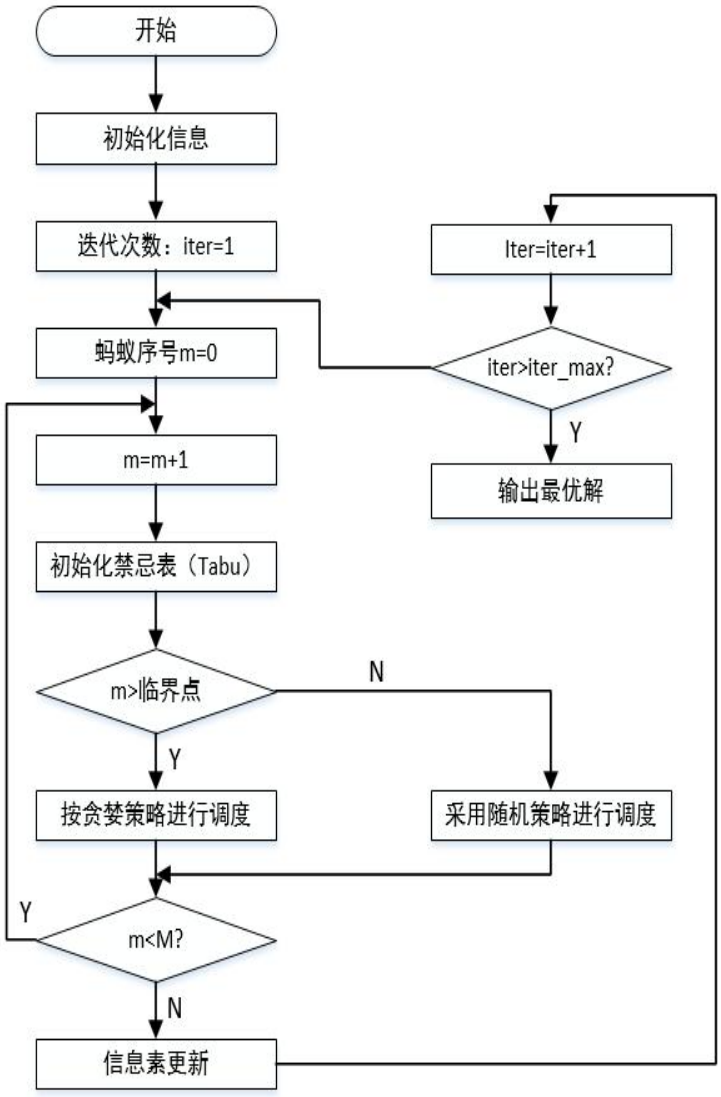

图 3.6 改进蚁群算法流程图

Fig. 3.6 flow chart of improved ant colony algorithm

基于改进蚁群算法的模型求解步骤如下：

步骤 1：初始化参数。根据配送中心实际情况对订单数 n_1，工人数 n_2，批次数量 n_3 进行赋值，并根据不同参数值得情况下的算法结果调试参数，确定蚁群中蚂蚁数量 m、信息素挥发率 ρ、信息素总量 Q、初始信息素浓度、最大迭代次数 NC_{max} 等有关参数取值。令时间 t=0 时，信息素矩阵中各匹配方案信息素值相同。最后对订单数据格式进行预处理。

步骤 2：每次迭代中随机生成采用不同分配策略的蚂蚁的临界编号 criticalPointMatrix。每次迭代中都有 antNum 只蚂蚁完成所有任务的分配工作。并且分配过程是按照蚂蚁编

号 i 从小到大的顺序进行的(蚂蚁从 1 开始编号)。在分配任务时，编号是 i <= criticalPointMatrix 的蚂蚁根据信息素浓度进行任务分配(即：采用贪婪策略将任务分配给本行中信息素浓度最高的批次)，其余蚂蚁则采用随机分配的方式(即采用随机策略将任务随机分配给任意一个批次)。

步骤 3：蚂蚁完成所有遍历后，计算各蚂蚁对应得分配方案的目标函数值；记录并更新当前最优解（即：目标函数值最小的分配方案）。

步骤 4：按更新方程更新信息素浓度。首先对信息素矩阵中各信息素值进行一定比例的减少，模拟现实环境中的信息素逸散。对上次迭代中最优解对应的信息素矩阵位置，增加相应的信息素浓度（增加量=每次迭代信息素总量/订单数量），以此达到更新信息素浓度的作用。

步骤 5：$Nc = Nc + 1$判断是否小于 NC_{max} 。若小于则转步骤 2，若已达最大迭代数则转至步骤 6。

步骤 6：输出最优解。

算法实现如表 3.2 所示：

表 3.2 算法实现
Tab. 3.2 algorithm implementation

改进蚁群算法
Input: antNum,p,D,NCmax, pheromoneMatrix,n1,n2,n3
Output: X
1. while iter < iter_max do
2. while i < antNum do
3. while j < n1 do
4. while i < pheromoneMatrix do
5. pathMatrix_oneAnt(j, maxPheromoneMatrix(j,1)) = 1
6. else
7. pathMatrix_oneAnt(j, randperm(n2*n3,1)) = 1
8. end while
9. end while
10. pathMatrix_allAnt(:,:,i) = pathMatrix_oneAnt
11. end while
12. Xone = best(pathMatrix_allAnt(:,:,i))
13. if Xone(j,k) == 1
14. $\Delta\tau(i,j) = D/n1$
15. $\tau = \tau \times p + \Delta\tau$
16. end while

3.5 本章小结

基于当下电商行业的发展现状，首先，对目前电商配送中心常用的 S 型拣货拣选路径策略进行改进，其次，综合考虑订单完成时效这一要求，构建订单拣选总作业时间最小和时效要求内订单完成量最大的订单分批优化模型。接着分析选择蚁群算法求解的原因，其次考虑到电商配送中心订单数量大，分批的可行方案种类众多，采用传统蚁群算法可能会产生算法初期收敛速度较慢和陷入局部最优解的情况进行改进，对蚁群算法进行改进，最后采用改进的蚁群算法对订单分批优化模型进行求解。

第 4 章 案例应用

4.1 应用背景

A 电商是一家较为综合的电子商务企业，目前已经涵盖家居厨具、服饰、箱包、个护清洁、家电、食品、户外运动等多个品类。A 配送中心是 A 电商企业其中的一个物流基地。物流一直以来都是 A 电商企业的核心竞争力之一，而使得物流成为 A 电商企业核心竞争力的一个重要原因就是 A 电商的仓储一体化物流体系，通过建设专业的物流配送中心，以仓库存储功能作为基础，再加上车辆配送服务，通过建设经济、安全、透明、便捷的仓储一体化系统，以便提高物流在运作过程中的效率。A 电商配送中心的运作流程图如图 4.1 所示。

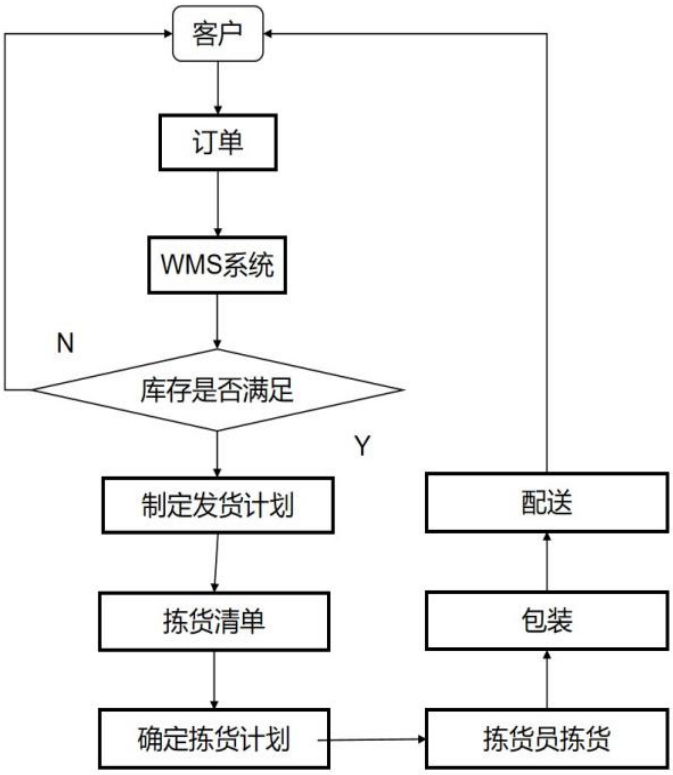

图4.1 A电商配送中心的运作流程图

Fig. 4.1 operation flow chart of a e-commerce distribution center

目前采用的拣选系统式为“人到货”，现行的拣选方式主要是先到先服务（FCFS）拣选，拣选员工利用手持无线设备和拣货车进行拣选作业，拣选作业流程如图 4.2 所示：

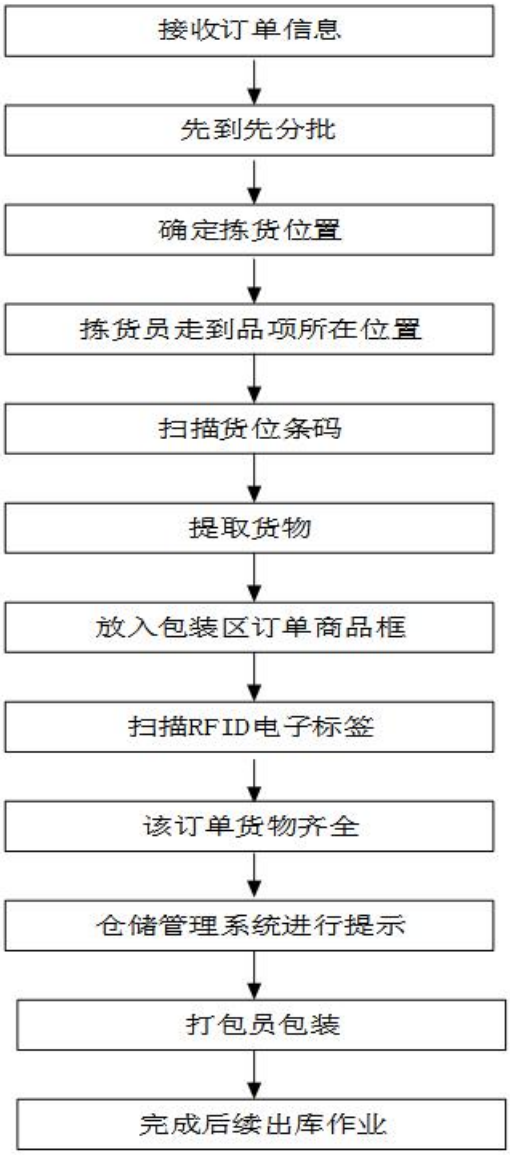

图4.2 A电商配送中心拣选作业流程图

Fig. 4.2 a e-commerce distribution center picking operation flow chart

在本文中，主要对订单拣选作业这一环节进行研究和优化，因为 A 电商配送中心是按照分区作业进行订单拣选作业的，每个存储区的作业流程基本相似，所以本文选取了 a 库一楼的某户外运动区的订单拣选作业作为主要的研究对象。该存储区的布局是示意图如图 4.3 所示。

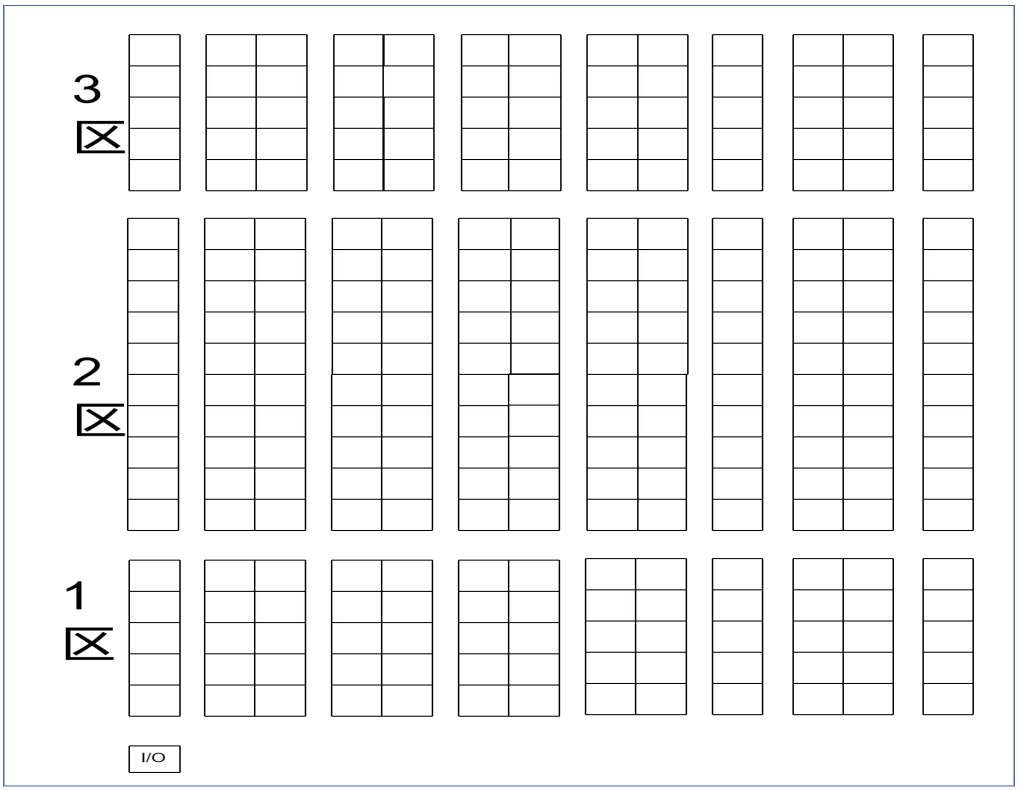

图4.3 存储区的布局示意图

Fig. 4.3 Layout of storage area

对订单拣选作业的研究需要计算不同货架之间的距离，所以需要先对该存储区货位的编码规则进行说明。该存储区共有 3 条通道和 25 条巷道，3 条通道将该存储区域分成了三个子区域，靠近起始点的 1 区，中间为 2 区，远离起始点的为 3 区。由于本文的订单拣选作业优化研究中，我们将垂直货位的拣选操作距离忽略不计，所以不考虑垂直货位。每条巷道的每侧有 36 个货架，而一个货架被分为 4 列，因此共有 144 个货位，从下往上货位编号为 1，2，3，...，144。每条巷道内两侧均可进行拣选作业，其中每条巷道内左侧的货位编号为 1，右侧为 2。综上所述，每个货位的编码可以表示为（区号，巷道号，左右号，货位号）。其中，通道的宽度为 1.2 米，巷道的宽度为 0.8 米，货架的长为 2.4 米，货架的宽为 0.5 米。

本文选取该分区某时间段内其中 118 张订单进行研究，订单信息节选表如表 4.1 所示，完整订单信息表见附录 A：

表4.1 订单信息表

Tab.4.1 Order information table

订单号	货位编号				数量(个)	总和（个）	订单时效要求(min)
1	1	8	2	19	2	2	39
2	1	7	2	34	1		
2	3	14	2	137	1		
2	2	14	1	50	1	5	51
2	1	18	1	1	1		
2	1	1	2	19	1		
3	1	1	2	19	1	2	49
3	3	4	2	132	1		
4	2	5	2	71	1	2	45
4	3	12	1	111	1		
...	...	...	...	...	...		...
60	2	16	2	76	5	5	31
61	2	4	1	74	1	1	42
62	2	13	1	96	6	6	45
63	3	10	2	138	1	1	26
64	1	10	2	18	4	4	26
65	1	16	1	15	2	2	26
66	1	1	2	9	1	1	26
67	2	11	2	88	1	2	42
67	1	10	1	3	1		
68	2	14	1	50	2	2	26
69	1	14	1	10	1	1	45
...	...	...	...	...	...		...
113	1	1	1	6	2	2	37
114	2	3	2	63	1	1	24
115	1	7	1	10	1	2	40
115	2	23	1	82	1		
116	1	1	2	9	1	1	45
117	3	17	2	122	1	2	34
117	1	3	2	29	1		
118	2	18	2	106	1		
118	3	13	2	137	1	3	67
118	2	13	1	104	1		

4.2 方案应用

将ξ_1设置为 0.4，ξ_2设置为 0.6。拟采用拣选员工数量为 8 名，行走速度为 60m/min，拣选速度为 5 秒/个，最大拣选容量为 20 个（不考虑重量和体积）。

4.2.1 先到先服务（FCFS）应用

首先，分析配送中心目前采用的先到先服务（FCFS）策略。先到先分批策略是按照到达订单的先后顺序，在满足不超出拣选设备最大装载能力（20 个）的情况下，依次将订单分批形成拣选批次。将 118 个订单按照先到先分批策略进行分批，得到各订单的批次，如表 4.2 所示，按照配送中心现有的分批方式进行分批，共有 14 个批次。

表4.2 先到先分批策略分批结果

Tab. 4.2 Results of first come first batch strategy

批次/订单号	订单号											
批次1	1	2	3	4	5	6	7	8				
批次2	9	10	11	12	13	14	15					
批次3	16	17	18	19	20	21						
批次4	22	23	24	25	26	27	28	29	30	31	32	33
批次5	34	35	36	37	38	39	40	41				
批次6	42	43	44	45	46	47						
批次7	48	49	50	51	52							
批次8	53	54	55	56	57	58	59	60	61			
批次9	62	63	64	65	66	67	68	69	70			
批次10	71	72	73	74	75	76	77	78	79	80		
批次11	81	82	83	84	85	86	87	88	89	90		
批次12	91	92	93	94	95	96	97					
批次13	98	99	100	101	102	103	104	105	106	107	108	
批次14	109	110	111	112	113	114	115	116	117	118		

首先，将拣货策略设置为原来的 S 型拣货策略，拣选完成该 118 个订单共需要 360 分钟，总拣选路程为 20317 米，时效要求内完成订单数为 90 个

其次，将拣货策略设置为改进的 S 型拣货策略，拣选完成该订单共需要 344 分钟，总拣选路程为 19406 米，时效要求内完成订单数为 93 个。

4.2.2 基于改进蚁群算法的订单分批优化应用

其次，由前文参数设置可知，针对本文采用的改进蚁群算法，最终确定最好的组合参数设置为蚂蚁数量 m=25、迭代次数 NC_{max} =100、初始信息素浓度=3、信息素挥发率 ρ =0.85、信息素总量 Q =60，每次迭代信息素增加量进行初始化定义。Matlab 编程语言来实现求解（Matlab 主程序代码见附录 B），运行在 MATLAB 2020a 平台，该平台安装在处理器为 AMD Ryzen 7 4800H with Radeon Graphics @ 2.9GHZ 和 Win10 家庭版软件系统的 PC 上。

首先，将拣货策略设置为改进的 S 型拣货策略，在 Matlab 软件中运行，程序运行过程中，目标函数值的变化过程如图 5.2 所示。从图 5.2 可以看出，算法从约 89 次开始收敛，找到最优解，程序总运行时间为 56.712 秒，算法收敛性和执行效率较高。

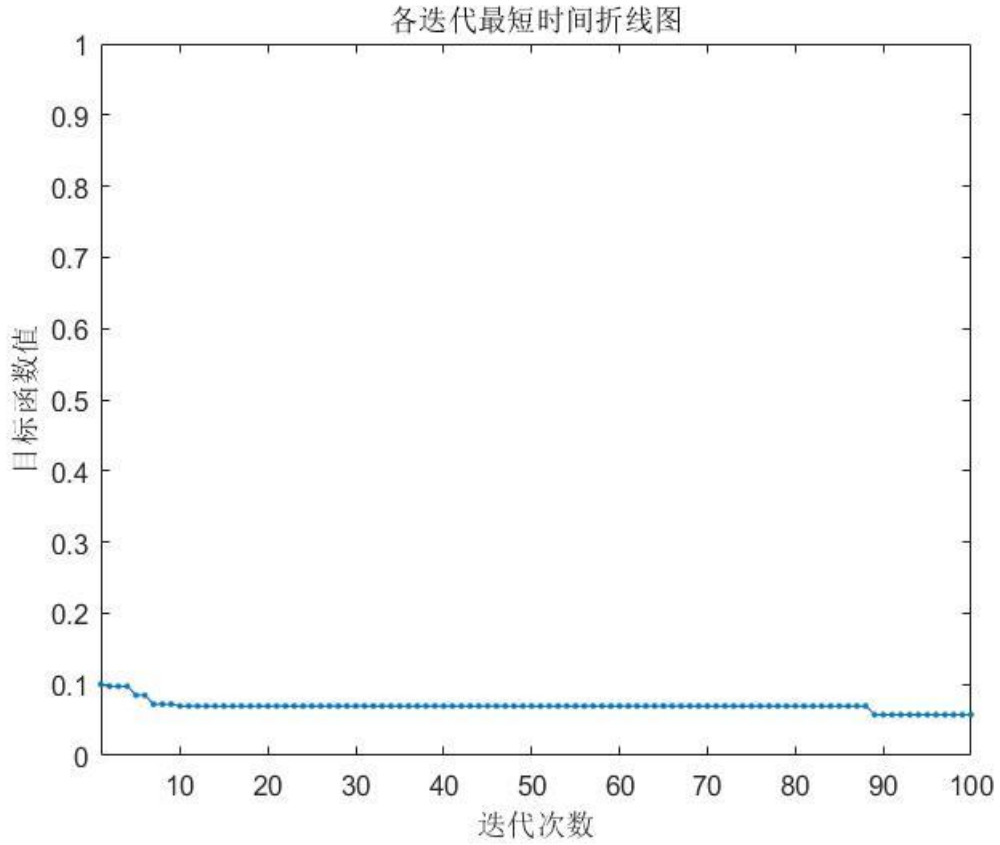

图4.4　目标函数值变化图

Fig. 4.4 Variation of objective function value

运行算法后，优化分批后总拣选时间为 270min，总拣选路程为 14944m，时效要求内完成订单数为 113 个，程序总运行时间为 29.8628s，最终订单分批运行结果如表 4.3 所示。

表4.3 订单分批优化运行结果表

Ta. 4.3 Results of order batch optimization

74	0	0	0	0	0
11	37	39	81	118	0
52	73	0	0	0	0
27	33	46	97		0
36	84	87	0	0	0
13	44	0	0	0	0
5	32	0	0	0	0
17	60	63	65	101	103
0	0	0	0	0	0
22	31	61	78	0	0
0	0	0	0	0	0
9	23	77	0	0	0
8	58	62	0	0	0
47	53	71	98	0	0
109	115	0	0	0	0
3	0	0	0	0	0
107	0	0	0	0	0
64	70	0	0	0	0
0	0	0	0	0	0
6	89	0	0	0	0
18	0	0	0	0	0
0	0	0	0	0	0
29	112	113	0	0	0
1	75	95	0	0	0
10	41	111	0	0	0
0	0	0	0	0	0
16	0	0	0	0	0
24	38	102	0	0	0
59	79		0	0	0
0	0	0	0	0	0
50	66	86	105	114	
92	96	106	0	0	0
26	48	0	0	0	0
14	30	42	67	108	116
20	34	0	0	0	0
80	85	117	0	0	0
45	104	110	0	0	0
54	55	0	0	0	0

续表 4.3

7	15	49	57	82	94
51	100	0	0	0	0
12	21	35	88	91	
0	0	0	0	0	0
43	68	93	99	0	0
2	4	40		0	0
28	0	0	0	0	0
0	0	0	0	0	0
19	69	76	90	0	0
25	56	72	83	0	0

将运行结果的最优解以订单分批的形式展示，具体结果如表 4.4 所示：

表4.4 基于改进蚁群算法的订单分批结果

Tab. 4.4 Order batching result based on improved ant colony algorithm

批次	订单号
1	74，5，32，8，58，62，10，41，111，50，66，86，105，114，45，104，110，43，68，93，99
2	11，37，39，81，118，17，60，63，65，101，103，47，53，71，98，6，89，92，96，106，54，55，2，4，40
3	52，73，109，115，18，16，26，48，7，15，49，57，82，94，28
4	27，33，46，97，22，31，61，78，3，24，38，102，14，30，42，67，108，116，51，100
5	36，84，87，107，29，112，113，59，79，20，34，12，21，35，88，91，19，69，76，90
6	13，44，9，23，77，64，70，1，75，95，80，85，117，25，56，72，83

其次，将拣货策略设置为原来的 S 型拣货策略，在 Matlab 软件中运行，运行算法后，拣选完成该 118 个订单共需要 325 分钟，总拣选路程为 18265 米，时效要求内完成订单数为 103 个。

4.2.3 订单拣选策略对比分析

为了验证改进 S 型拣选策略的效果，首先，在现行先到先分批 FCFS 情况下，采用

不同拣选策略进行对比。其次，在改进蚁群算法的蚁群算法的订单分批优化情况下，采用不同拣选策略进行对比。

（1）FCFS+不同拣选策略

使用 S 型拣选策略进行拣选作业，拣选完成该 118 个订单共需要 360 分钟，总拣选路程为 20317 米，时效要求内完成订单数为 90 个；使用改进 S 型拣选策略进行拣选作业，拣选完成该订单共需要 344 分钟，总拣选路程为 19406 米，时效要求内完成订单数为 93 个，具体对比如表 4.5 所示。

表4.5 先到先服务分批下订单拣选策略对比

Tab. 4.5 Comparison of first come first serve Batch Order Picking Strategies

方案	总拣选时间（min）	总拣选路程(m)	时效要求内完成订单数
FCFS和S型策略	360	20317	90
FCFS和改进S型策略	344	19406	93

从表中可以看出，不分批的情况下，改进 S 型拣选策略，可以减少 16 分钟的的总拣选时间，优化程度为 4.4%；减少了 914 米总拣选路程减少了 911 米，优化程度为 4.5%；提高 3 个的订单完成数量，优化后程度为 3.3%。

（2）分批优化+不同拣货策略

使用 S 型拣选策略进行拣选作业，拣选完成该 118 个订单共需要 325 分钟，总拣选路程为 18265 米，时效要求内完成订单数为 103 个；使用改进 S 型拣选策略进行拣选作业，拣选完成该订单共需要 270 分钟，总拣选路程为 14944 米，时效要求内完成订单数为 111 个，具体对比如表 4.6 所示。

表4.6 优化分批下订单拣选策略对比

Tab. 4.6 Comparison of optimized Batch Order Picking Strategie

方案	总拣选时间（min）	总拣选路程(m)	时效要求内完成订单数
优化分批和S型策略	325	18265	103
优化分批和改进S型策略	270	14944	111

从表中可以看出，在改进蚁群算法的订单分批优化情况下，改进 S 型拣选策略，可以减少 55 分钟的总拣选时间，优化程度为 16.9%；减少了 3321 米总拣选路程，优化程度为 18.2%；提高 8 个的订单完成数量，优化程度为 7.8%。

通过对比表 4.5 和 4.6 可知，不管是分批优化后还是原来的在现行先到先分批 FCFS 情况，改进 S 型策略都能起到减少总拣选时间和总拣选路程，并且提高订单的完成数量的效果。且在分批优化后，改进 S 型策略的优化程度也更高。

4.2.4 订单分批方案对比分析

为了验证改进蚁群算法的蚁群算法的订单分批效果，首先，在 S 型拣选策略下，采用不同分批方案进行对比。其次，在改进 S 型拣选策略下，采用不同分批方案进行对比。

（1）S 型拣选策略+不同分批方案

在现行的先到先服务即不分批方案进行拣选作业，拣选完成该 118 个订单共需要 360 分钟，总拣选路程为 20317 米，时效要求内完成订单数为 90 个；使用改进蚁群算法的订单分批优化后，拣选完成该订单共需要 325 分钟，总拣选路程为 18265 米，时效要求内完成订单数为 103 个，具体对比如表 4.7 所示。

表4.7 S型策略下订单分批方案对比

Tab. 4.7 Comparison of order batching schemes under s-strategy

方案	总拣选时间（min）	总拣选路程(m)	时效要求内完成订单数
S型策略和FCFS	360	20317	90
S型策略和优化分批	325	18265	103

从表中可以看出，S 型拣选策略下，改进蚁群算法的订单分批可以减少 35 分钟的总拣选时间，优化程度为 9.7%；减少 2052 米的总拣选路程，优化程度为 10.1%；提高 13 个的订单完成数量，优化程度为 14.4%。

（2）改进 S 型拣选策略+不同分批方案

在现行的先到先服务即不分批方案进行拣选作业，拣选完成该 118 个订单共需要 344 分钟，总拣选路程为 19406 米，时效要求内完成订单数为 93 个；使用改进蚁群算法的订单优化分批后，拣选完成该订单共需要 270 分钟，总拣选路程为 14944 米，时效要求内完成订单数为 111 个，具体对比如表 4.8 所示。

表4.8 改进S型策略下订单分批方案对比

Tab. 4.8 Comparison of order batching schemes under improved s-strategy

方案	总拣选时间（min）	总拣选路程(m)	时效要求内完成订单数
改进S型策略和FCFS	344	19406	93
改进S型策略和优化分批	270	14944	111

从表中可以看出，改进 S 型拣选策略下，改进蚁群算法的订单分批可以减少 74 分钟的总拣选时间，优化程度为 21.5%；减少的总拣选路程 4462 米，优化程度为 23.0%；提高的订单完成数量，优化程度为 19.4%。

通过表 4.7 和 4.8 可知，不管是原来的 S 型策略还是改进后的 S 型策略，优化分批

都能起到减少总拣选时间和总拣选路程，并且提高订单的完成数量。且在改进拣选路径策略后，订单分批优化程度也更高。

4.2.5 订单拣选方案对比分析

为了验证本文订单拣选方案的优化效果，将该配送中心原来的拣选方案（S 型拣选策略和先到先服务即不分批）与优化后的拣选方案（改进 S 型拣选策略和改进蚁群算法的订单分批优化）进行对比。采用原来的拣选方案选完成该 118 个订单共需要 360 分钟，总拣选路程为 20317 米，时效要求内完成订单数为 90 个；使用优化后的拣选方案后，拣选完成该订单共需要 270 分钟，总拣选路程为 14944 米，时效要求内完成订单数为 111 个，具体对比如表 4.9 所示。

表4.9 订单拣选方案对比

Tab. 4.9 Warehousing management module function

方案	总拣选时间（min）	总拣选路程(m)	时效要求内完成订单数
S型策略和不分批	360	20317	90
改进S型策略和优化分批	270	14944	111

从表中可以看出，优化后的拣选方案可以减少 90 分钟的总拣选时间，优化程度为 25.0%；减少 5357 米的总拣选路程，优化后程度为 26.4%；提高 21 个订单完成数量，优化程度为 23.3%。

由此可见，本文提出的订单拣选作业的优化方案取得不错的成效，总拣选作业和总拣选路程都显著减少，时效要求内完成的订单数量显著增加。

4.3 本章小结

本文选取了 A 电商企业的配送中心进行应用，首先介绍了优化方案应用的配送中心背景和作业流程现状，获得订单基础数据。本文将用到 MATLAB 编程语言来实现多目标函数的求解，使最后得到的订单拣选的方案较为理想，最后和 A 电商企业原有的拣选方案进行比较，从而验证本文所提优化方法及模型的可行性和有效性。

第5章 结论和展望

5.1 结论

订单拣选作业优化对提高配送中心的作业效率，降低配送中心的运营成本成本，保证订单及时准确出库，提高企业服务水平有着重要的意义。当前电商环境下，订单呈现出“单笔订单商品数量少、订单总体数量多且差异性大、响应时间短且严格”等特点，使得订单拣选作业变得更为复杂。为此，本文面向电商配送中心订单拣选作业优化问题进行了研究，通过对拣选路径策略优化和订单分批优化对订单拣选作业进行优化，并将优化结果进行了应用。本文的主要工作和结论如下。

（1）阐述了对电商配送中的订单拣选作业进行优化的必要性，总结并分析了国内外学者对订单拣选作业问题的研究现状和取得的相关理论成果。

（2）针对拣选路径策略优化问题，本文结合电商订单特点，对目前电商配送中心常用的S型拣选路径策略进行改进，即结合使用S型拣选策略和中点型拣货策略。

（3）针对订单分批优化问题，将订单完成时效这一要求引入到订单优化分批中，构建订单拣选总作业时间最小和时效要求内订单完成量最大的多目标订单分批优化模型。接着，对蚁群算法进行改进，最后采用改进的蚁群算法对订单分批优化模型进行求解。

（4）为了验证订单优化的有效性，本文选取了A电商企业的配送中心进行应用。并使用改进蚁群算法对订单分批优化模型进行求解。在使用改进蚁群算法时，本文将用到Matlab编程语言来实现多目标函数的求解，使最后得到的结果较为理想在最后和A电商企业配送中心原有的拣选方案进行比较，从而验证本文所提优化方法及模型的可行性和有效性。

（5）研究结果表明，优化后的拣选方案订单总拣选时间减少了25.0%，拣货员行走距离减少了26.4%，在订单时效内拣选货物的数量提高了23.3%。

同时，本文对订单拣选作业的优化研究，也可以为其他类似电子商务企业优化订单拣选作业提供参考和借鉴意义。

5.2 展望

本文是对电商配送中心的订单拣选作业进行拣选作业的优化研究，从拣选路径策略和订单分批两个方面进行了优化设计，并且得出了有效的优化结果。但是在实际的电商配送中心订单拣选作业优化问题上，本文中也存在一些研究的不足之处：

比如，在实际操作中，拣选作业的效率受到很多因素的影响，例如仓库的布局、存储策略的选择、订单分批拣选、拣选路径规划、分拣方式等等，它是一个系统的过程，并且各个因素之间相互影响，例如存储策略的不同会导致订单拣选路径的不同，前期使用订单分批拣选方式会带来后面的分拣问题。而在本文中，对于拣选作业的优化只针对订单分批和拣选路径策略优化来进行研究，并没有从全局的角度进行分析。如何将各个影响因素相互联系进行分析评价，并且从整体角度对拣选作业进行优化将会是未来的一个研究方向。

另外，本文中进行订单拣选作业优化设计时，没有考虑多个拣选人员一起拣选时可能会造成的拥堵情况。在实际的仓库拣选作业中，往往是每个拣选人员负责一批订单中全部商品的拣选，有的仓库中也有每个拣选人员负责一部分拣选区域商品的拣选。无论是何种情况，在订单比较多时，多张订单一起进行拣选作业，很可能产生拥堵的情况，在这种情况下，如何对拣选作业进行优化，也是未来值得研究的问题。

以上为对未来研究的展望，本篇文章存在的不足之处，将会在以后的学术研究中进行探究。

参考文献

[1] Gademann A,Van D, Van D. An order batching algorithm for wave picking in a parallel-aisle warehouse[J]. AIIE Transactions, 2001, 33(5):385-398.

[2] Chen M C, Wu H P. An association-based clustering approach to order batching considering customer demand patterns[J]. Omega, 2005, 33(4):333-343.

[3] Hwang H, Kim D G. Order-batching heuristics based on cluster analysis in a low-level picker-to-part warehousing system[J]. International Journal of Production Research, 2005, 43(17):3657-3670.

[4] Tsai C Y, Liou J H, Huang T M. Using a multiple-GA method to solve the batch picking problem: considering travel distance and order due time[J]. International Journal of Production Research, 2008, 46(22):6533-6555.

[5] Henn S, Schmid V. Metaheuristics for Order Batching and Sequencing in Manual Order Picking Systems[J]. Computers & Industrial Engineering, 2011, 66(2):338-351.

[6] Henn,S.,Schmid,V.Metaheuristics for order batching and sequencinginmanual order picking systems[J]. Computerts & Industrial Engineering, 2013, 66(2):338-351.

[7] Ncan T. MILP formulations and an Iterated Local Search Algorithm with Tabu Thresholding for the Order Batching Problem[J]. European Journal of Operational Research, 2015, 243(1):142-155.

[8] F Chen, Wang H, Yong X, et al. An ACO-based online routing method for multiple order pickers with congestion consideration in warehouse[J]. Journal of Intelligent Manufacturing, 2016, 27(2):389-408.

[9] Scholz A, Schubert D , Scher 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, 2016:461-478.

[10] Menendez, Soda, Molina, et al. Variable Neighborhood Search strategies for the Order Batching Problem[J]. Computers & Operations Research, 2017,78:500-512.

[11] Xue F, Dong T, Qi Z . An improving clustering algorithm for order batching of e-commerce warehouse system based on logistics robots[J]. International journal of wireless and mobile computing, 2018,

15(1):10-15.

[12] Hwang H, Kim D G. Order-batching heuristics based on cluster analysis in a low-level picker-to-part warehousing system[J]. International Journal of Production Research, 2005, 43(17):3657-3670.

[13] Parikh P J, Meller R D . A travel-time model for a person-onboard order picking system[J]. European Journal of Operational Research, 2010, 200(2):385-394.

[14] Chan F, Chan H K . Improving the productivity of order picking of a manual-pick and multi-level rack distribution warehouse through the implementation of class-based storage[J]. Expert Systems with Applications, 2011, 38(3):2686-2700.

[15] Ncan T. MILP formulations and an Iterated Local Search Algorithm with Tabu Thresholding for the Order Batching Problem[J]. European Journal of Operational Research, 2015, 243(1):142-155.

[16] Lu W, Mcfarlane D, Giannikas V, et al. An algorithm for dynamic order-picking in warehouse operations[J]. European Journal of Operational Research, 2016,80(80):9-20..

[17] Dijkstra A S, Roodbergen K J. Exact route-length formulas and a storage location assignment heuristic for picker-to-parts warehouses[J]. Transportation Research Part E: Logistics and Transportation Review, 2017, 102(7):38-59.

[18] Yener F, Yazgan H R. Optimal Warehouse Design: Literature Review and Case Study Application[J]. Computers & Industrial Engineering, 2019,129:1-33.

[19] 马士华，文坚．基于时间延迟的订单分批策略研究[J]．工业工程与管理，2004（6）：1-4．

[20] 郝彤彤．GD 商城 3C 仓储中心拣货体系选择分析[D]．青岛：中国海洋大学，2014.

[21]]王旭坪，张珺，马骏．考虑完成期限的电子商务在线订单分批模型及算法[J]．管理科学，2014，27（6）：103-113.

[22] 陈方宇．多区块仓库环境下订单掠选路线规划研究[D]．武汉：华中科技大学，2014．

[23] 邹霞．面向 B2C 电商配送中心的分散式自动存取及拣选系统效率优化研究[D]．济南：山东大学，2018．

[24] 郜振华，陈卓．基于萤火虫算法的订单分批问题研究[J]．物流科技，2019，42（7）：10-15．

[25] 徐鹏．基于蚁群算法的 PC 构件分批配送模型研究[J]．大众标准化，2019（14），106-107.

[26] 黄敏芳，张源凯，王颜新，胡祥培．基于 JIT 装配模式的网上超市订单分拣优化模型[J].中国管

理科学．2020，28（05）.

[27] 冯辰吉，苌道方，宋鑫，范雨涵．订单寻仓与分批拣选联合调度问题研究[J]．制造业自动化，2020，42（11）：75-81.

[28] 秦馨，赵剑道，任楠，李佳顺，马洪泽．基于遗传算法的订单分批策略研究[J]．制造业自动化，2021，43(5)：108-112.

[29] 华红艳，张丹.基于蚁群算法的自动化立体仓库路径优化.计算机技术与自动化，2010，29（1）：51-54.

[30] 朱杰，郭键，周丽．随机存储下返回型与 S 型拣选路径随机模型的比较研究[J]. 系统仿真学报，2011，23（2）：223-227.

[31] 朱文真，唐敦兵，王雷．基于遗传禁忌搜索算法的自动化立体仓库出入库路径优化研究[J]．机械科学与技术，2011，30（007）：1202-1206.

[32] 庞龙，陆金桂．基于蚁群遗传算法的自动化立体仓库拣选路径优化[J]．计算机工程与科学，2012，34（3）：148-151.

[33] 卢子甲，韩义民，张少卿．基于遗传算法的配送中心订单拣选路径优化案例研究[J]物流技术，32（9）：3.

[34] 李建斌，周玮，陈峰．B2C 电子商务仓库拣货路径优化策略应用研究[J].. 运筹与管理，2014（1）：11-18.

[35] 李栋栋．双区型仓库拣货路径优化研究[D]．青岛：青岛大学，2015.

[36] 刘建胜，熊峰，陈景坤等.基于蚁群算法的双分区仓库拣货路径的优化[J]，高技术通讯，2017，27（1）：72-79.

[37] Quader S, Castillo-Villar K K. Design of an enhanced multi-aisle order-picking system considering storage assignments and routing heuristics[J]. Robotics and Computer-Integrated Manufacturing, 2018(50):13-29..

[38] Ackerman K B. Practical Handbook of Warehousing[M]. 1990.

[39] 王转，裴泽平．启发式路径下节约里程的订单分批算法[J]．计算机工程与应用，2018，54（23）：209-215+228.

[40] 王聪，金淳，马琳等．考虑排产约束的汽车零部件出库订单排序优化[J].工业工程与管理，2016，

21（4）：106-113.

[41] 胡小建，韦超豪．基于 Canopy 和 k-means 算法的订单分批优化[J]．合肥工业大学学报(自然科学版)．2017．40（3）：414-419.

[42] 唐思园. 基于订单分批的配送中心拣货作业流程研究[D]．浙江海洋大学，2020.

[43] 丁连红，时鹏，刘丙.仓储中心拣选作业研究综述.物流技术[J]．2008（2）：131-135.

[44] 孙辉．A 电商企业配送中心拣选作业优化研究[D]．厦门：厦门大学,2017.

[45] 王艳丽．配送中心拣选作业优化研究[D]．吉林：吉林大学．2017.

[46] 马飞．电商配送中心订单分批及拣选路径优化研究[D]．昆明：昆明理工大学，2016.

[47] 赵兰．基于拣选和分拣时间的订单分批优化方法研究与应用[D]．武汉：华中科技大学，2015.

[48] Hall, Randolph W. Distance approximation for routing manual picker in a warehouse[J].IIE Transaction，1993，25（4）：76-87．.

[49] 赵邦磊.基于改进多目标蚁群算法的冷链物流路径优化研究[D].淮南：安徽理工大学，2020.

[50] 殷玲玲, 苏剑锋. 基于蚁群算法的多配送中心车辆调度问题的探讨[J]．九江学院学报(自然科学版)：2020，35（3）：40-42.

[51] 曹如月，李世超，季宇寒等．基于蚁群算法的多机协同作业任务规划[J]．农业机械学报，2019（S1）：34-39.

[52] 乔东平，裴杰，肖艳秋等．蚁群算法及其应用综述［J］．软件导刊，2017，16(12)：217-221。

附录 A 订单信息表

订单号	货位编号				数量	总和	订单时效要求（min）
1	1	8	2	19	2	2	39
2	1	7	2	34	1		
2	3	14	2	137	1		
2	2	9	1	106	1	5	51
2	2	14	1	50	1		
2	1	18	1	1	1		
3	1	1	2	19	1	2	49
3	3	4	2	132	1		
4	2	5	2	71	1	2	45
4	3	12	1	111	1		
5	3	7	1	134	1	2	57
6	1	4	2	1	1	2	45
7	1	2	1	3	1	2	37
8	3	14	2	129	3	3	94
9	1	22	1	32	1	5	86
10	2	3	2	95	1	2	48
10	2	21	1	81	1		
11	3	9	2	121	2		
11	1	9	1	23	1	5	71
11	2	18	2	87	1		
11	1	10	2	31	1		
12	1	10	1	4	1	1	41
13	2	18	2	87	1	1	32
14	1	13	1	1	3	3	61
15	1	1	2	6	2	2	67
16	2	14	2	50	1	2	43
16	1	3	2	31	1		
17	2	6	2	103	1	2	51
17	1	7	2	34	1		
18	3	3	2	141	1		
18	1	17	1	8	1	3	29
18	2	12	2	58	1		
19	2	21	2	81	1		
19	3	15	1	139	1	9	49
19	2	20	1	104	1		

续表

19	2	19	2	76	3		
19	2	13	1	96	1		
19	3	24	1	143	1		
19	1	2	1	5	1		
20	2	18	1	49	2	3	70
20	1	6	1	24	1		
21	2	11	2	49	1	1	31
22	1	21	1	27	1	3	51
22	3	13	2	138	2		
23	2	3	2	87	1	1	21
24	1	5	2	32	1	2	70
24	1	24	1	1	1		
25	2	23	2	87	1	1	34
26	2	8	1	89	1	1	21
27	1	14	1	16	1	1	29
28	3	2	2	114	1	1	49
29	1	6	1	23	1	1	21
30	2	11	1	42	1	1	21
31	2	8	1	96	1	4	51
31	2	9	2	95	3		
32	3	15	1	139	1	2	51
32	1	17	2	8	1		
33	1	1	1	6	1	2	51
33	1	15	1	29	1		
34	2	9	1	49	1		
34	2	24	2	44	1		
34	2	20	2	105	1		
34	1	17	2	12	1	7	70
34	2	4	2	51	1		
34	2	2	1	88	1		
34	2	21	2	103	1		
35	1	15	2	7	2	2	34
36	1	1	1	27	1	1	
37	3	17	1	122	1		
37	1	19	2	3	1	3	41
37	3	13	2	119	1		
38	1	15	1	16	1	1	21
39	1	10	1	27	1	2	28
39	2	2	1	72	1		

续表

40	3	10	2	114	1	1	17
41	3	16	1	118	1	1	20
42	1	8	2	23	1		
42	2	6	1	88	1		
42	1	2	1	2	1	6	57
42	2	7	1	103	1		
42	2	13	1	96	2		
43	2	3	1	103	1	2	69
43	2	2	1	103	1		
44	2	2	1	40	1	1	24
45	3	1	2	136	3	3	45
46	2	4	1	78	1	1	54
47	3	4	1	123	3	3	43
48	1	15	1	31	2	6	40
48	1	4	1	1	4		
49	2	21	1	103	2		
49	1	19	2	8	1	5	67
49	3	19	2	127	1		
49	2	13	1	88	1		
50	1	4	2	1	2	4	67
50	1	1	1	19	2		
51	1	4	2	1	2	2	26
52	2	21	1	72	1	2	28
52	1	1	2	9	1		28
53	2	19	1	89	2	2	51
54	3	7	1	121	1	1	19
55	2	12	1	97	1	1	26
56	1	3	1	5	1	1	12
57	2	4	2	96	1	1	23
58	2	4	1	78	2	2	34
59	2	17	2	96	1	1	42
60	2	16	2	76	5	5	31
61	2	4	1	74	1	1	42
62	2	13	1	96	6	6	45
63	3	10	2	138	1	1	26
64	1	10	2	18	4	4	26
65	1	16	1	15	2	2	26
66	1	1	2	9	1	1	26
67	2	11	2	88	1	2	42

续表

67	1	10	1	3	1		
68	2	14	1	50	2	2	26
69	1	14	1	10	1	1	45
70	3	1	2	121	1	1	17
71	2	23	2	95	1	1	49
72	1	6	1	5	1	2	84
72	2	3	1	82	1		
73	3	17	2	129	2	2	62
74	3	8	1	133	3	3	32
75	2	24	1	49	1	2	81
75	1	6	1	2	1		
76	2	13	2	96	3	3	21
77	2	19	1	71	2	1	26
78	3	2	1	127	1	2	81
78	1	8	2	19	1		
79	1	17	1	8	1	2	41
79	1	3	1	2	1		
80	1	4	2	1	1	2	96
80	1	15	1	29	1		
81	2	3	2	87	1	1	52
82	2	12	1	97	1	2	76
82	1	8	2	20	1		
83	3	17	1	134	1	1	12
84	3	4	1	140	1	1	21
85	1	1	2	9	1	1	41
86	2	4	1	78	1	1	51
87	3	19	2	134	2	2	41
88	1	6	2	2	2	2	33
89	1	17	2	8	1	1	23
90	2	6	2	58	1	2	78
90	2	22	2	69	1		
91	3	16	1	129	1		
91	2	21	1	51	1		
91	2	19	1	88	2	7	45
91	1	14	2	8	1		
91	2	17	2	76	1		
91	1	8	2	20	1		
92	2	10	2	57	1	1	21
93	2	10	1	57	1	1	31

续表

94	3	14	2	133	1	3	76
94	2	2	2	102	2		
95	1	6	1	7	5	5	52
96	3	19	1	114	2	2	26
97	2	10	2	57	1	1	46
98	1	1	2	9	3	3	49
99	3	12	2	144	1	2	76
99	2	9	1	89	1		
100	3	3	2	134	1		
100	1	19	2	16	1	3	67
100	3	22	1	141	1		
101	2	3	1	87	1	1	29
102	2	15	1	49	1	1	35
103	2	2	2	83	1	1	27
104	1	15	2	29	3	3	31
105	2	19	2	83	1	1	24
106	1	1	1	6	1	2	37
106	1	15	1	31	1		
107	2	11	2	96	1	1	18
108	2	8	1	41	1	1	45
109	2	6	2	75	2	2	21
110	1	21	2	7	1		
110	2	25	1	82	1	3	81
110	2	18	2	87	1		
111	3	15	2	129	1	1	38
112	3	12	1	140	1	1	42
113	1	1	1	6	2	2	37
114	2	3	2	63	1	1	24
115	1	7	1	10	1	2	40
115	2	23	1	82	1		
116	1	1	2	9	1	1	45
117	3	17	2	122	1	2	34
117	1	3	2	29	1		
118	2	18	2	106	1		
118	3	13	2	137	1	3	67
118	2	13	1	104	1		

附录 B MATLAB 主程序代码

```
%% I. 清空环境变量
clear
clc
tic    % 程序计时开始
global n1 n2 n3  a B

a =  xlsread('a.xlsx','Sheet1','A2:F189'); % 订单总表
B = xlsread('a1.xlsx','Sheet1','A2:C119');  % 读取每个订单的货物数量和时间限制
a(:,4) = [];

n1 = 118; % 订单数量
n2 = 8; % 工人数量
n3 = 6; %批次
number = 20; % 每批的数量限制
%%
antNum =25;  %蚂蚁数量
pheromoneMatrix =ones(n1,n2*n3);   %信息素矩阵
pheromoneMatrix = pheromoneMatrix.*3;
maxPheromoneMatrix = zeros(n1,1); %每行最大信息素节点
criticalPointMatrix = zeros(n1,1);  %蚂蚁临界编号 及每行信息素平均值计算得到的值
p = 0.85;  %信息素衰减比例
D = 100;  %信息素增加   根据 total_time() 平均值更改 适应
iter = 1;       % 迭代次数初值
iter_max = 100;  %迭代次数
Time_best = zeros(iter_max,1);     % 各代最佳分配

best_oneiter = zeros(n1,n2*n3);  % 每次迭代中的最优解保存
```

```
    best_objv = Inf;    %针对不收敛问题加入精英保留策略
    best_node = 0;  % 信息素更新标志位

    %% 定义数据整理矩阵
    All_iter_min_t = zeros(iter_max,1);
    All_iter_match = zeros(iter_max,n1,n2*n3);

    %%
    resultData = zeros(iter_max,antNum);    % 每次迭代目标结果记录
    while iter <= iter_max
        pathMatrix_allAnt = zeros(n1,n2*n3,antNum); %本次迭代中分配方案（路径）记录
        for i = 1:antNum
            pathMatrix_oneAnt = zeros(n1,n2*n3); % 单个蚂蚁的路径记录
            count = zeros(n2*n3,1);
            for j = 1:n1
                if i <= criticalPointMatrix(j,1)        %防止局部最优
                    staCount = maxPheromoneMatrix(j,1);    %% 第 J 个订单的最大信息素的节点
                    while (count(staCount) + B(j,2)) > number
                        staCount = randperm(n2*n3,1); %最大信息素节点 不可选随机选择批次
                    end

                else
                    staCount = randperm(n2*n3,1);    % 随机选取批次
                    while (count(staCount) + B(j,2)) > number
                        staCount = randperm(n2*n3,1); %最大信息素节点 不可选随机选择批次
                    end
                end
```

```
                count(staCount,1) = count(staCount,1) + B(j,2);
                pathMatrix_oneAnt(j,staCount) = 1;  %标记当前蚂蚁的对应路径
            end
            pathMatrix_allAnt(:,:,i) = pathMatrix_oneAnt;    % 将当前蚂蚁路径加入总的路径表
        end
    %% 计算本次迭代中所有蚂蚁的任务处理时间
        time_allAnt = zeros(1,antNum);  %计算 本次迭代中 所有蚂蚁 的任务处理时间（所有蚂蚁路径对应方案的目标函数值）
        for i = 1 : antNum
            pathMatrix = pathMatrix_allAnt(:,:,i);    %获取第 i 只蚂蚁的行走路径
            time_oneant = fun(pathMatrix);
            time_allAnt(i) = time_oneant;
        end


        resultData(iter,:) = time_allAnt;    % 将本地迭代中 所有蚂蚁的目标函数值加入总结果集
        %%更新信息素
        %% 找出本次迭代中目标函数最小的蚂蚁
        time = time_allAnt(1,1);
        min_index = 1;
        for i = 2 : antNum
            if time_allAnt(1,i) < time
             time =  time_allAnt(1,i);
             min_index = i;
            end
        end
        % 目标函数最小的蚂蚁的路径匹配矩阵
        X = pathMatrix_allAnt(:,:,min_index);
```

```
%输出目标函数最小最短蚂蚁的目标函数
min_tim_oneiter = time_allAnt(min_index);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% 精英保留策略
if min_tim_oneiter > best_objv
    All_iter_min_t(iter,1) = best_objv;
    All_iter_match(iter,:,:) = best_oneiter;
else
    best_objv = min_tim_oneiter;
    best_oneiter = X;
    All_iter_min_t(iter,1) = min_tim_oneiter;
    All_iter_match(iter,:,:) = X;
    best_node = 1;
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            % 每只蚂蚁的信息素更新
    if best_node ==1
        add_t = zeros(n1,n2*n3);
        for j = 1 : n2*n3
            for k = 1 : n1
                if X(k,j) == 1
%                     add_t(k,j) = min_tim_oneiter/n1;          %%%这
里用常值来更新增加量更好一点
                    add_t(k,j) = D/n1;      %% 常值更新   用常值的话要
考虑订单数量和批次数量改变之后的变化
                end
            end
```

```
                end
                best_node = 0;    %信息素更新标志位恢复
            else
                add_t = zeros(n1,n2*n3);
                for j = 1 : n2*n3
                    for k = 1 : n1
                            add_t(k,j) = D/n1;      %% 常值更新  用常值的话要
考虑订单数量和批次数量改变之后的变化
                    end
                end
            end

            for k = 1 : n1
                for j = 1 : n2*n3
                    pheromoneMatrix(k,j) = pheromoneMatrix(k,j) * p +
add_t(k,j); %信息素更新
                end
            end


        %
        for k = 1 : n1
            maxPheromone = pheromoneMatrix(k,1);
            maxIndex = 1;
            sumPheromone = 0;
            isAllSame = 1;

            for j = 1 : n2*n3

                if pheromoneMatrix(k,j) > maxPheromone
                    maxPheromone = pheromoneMatrix(k,j);
```

```
                maxIndex = j;

            end
            if j >= 2
                if pheromoneMatrix(k,j) ~= pheromoneMatrix(k,j - 1)
                    isAllSame = 0;
                end
            else
                if pheromoneMatrix(k,j) ~= pheromoneMatrix(k,j + 1)
                    isAllSame = 0;
                end

            end
            sumPheromone = sumPheromone + pheromoneMatrix(k,j);
        end
        % 若本行信息素全都相等，则随机选择一个作为最大信息素
        if isAllSame==1
            maxIndex = randperm(n2*n3,1);
            maxPheromone = pheromoneMatrix(k,maxIndex);
        end
        % 将本行最大信息素的下标加入 maxPheromoneMatrix
        maxPheromoneMatrix(k,1) = maxIndex;
        % 将本次迭代的蚂蚁临界编号加入 criticalPointMatrix(该临界点之前的蚂蚁的任务分配根据最大信息素原则，而该临界点之后的蚂蚁采用随机分配策略)
        criticalPointMatrix(k,1) = round(antNum * (maxPheromone/sumPheromone)); %四舍五入 求临界
    end
     % 迭代次数加 1，清空路径记录表
    iter = iter + 1;
end
%% 结果显示
```

```
%% 查找所有迭代中的最短时间和对应匹配矩阵
time = All_iter_min_t(1,1);
min_index = 1;    %%这里的索引指的是迭代次数索引
for i = 2 : iter_max
    if All_iter_min_t(i,1) < time
        time = All_iter_min_t(i,1);
        min_index = i;
    end
end
% 目标函数最小的蚂蚁的路径 匹配矩阵
X  = zeros(n1,n2*n3);
X(:,:) = All_iter_match(min_index,:,:);
%输出目标函数最小蚂蚁的用时
BestResult = All_iter_min_t(min_index)
writematrix(X,'Best_match_.xlsx');



%% 每次迭代的最优结果和迭代次数图
figure(1)
F = zeros(iter_max,2);
for i = 1 : iter_max
    F(i,1) = i;
    F(i,2) = All_iter_min_t(i);
end
plot(F(:,1),F(:,2),'.-');
axis([-inf,inf,0,1])
title('各迭代最短时间折线图')
xlabel('迭代次数')
ylabel('目标函数值')
```

```
result(X);


toc      %  程序计时终止
disp(['程序运行时间: ',num2str(toc)]);
```

在学研究成果

一、发表论文

[1] ***，***. 电商企业仓储 ABC 差异化库存管理策略研究[J]. 中国物流与采购，2020(17)：33-34.

致谢

行文至此，落笔为终，三载青春转眼而过，三千往事浮现眼前。始于2018年金秋，终于2021年盛夏，目之所及，皆是回忆与感动，心怀感激！

本论文是在***导师的悉心指导下完成的，从论文选题、课题调研、试验指导、理论分析到论文撰写，无不倾注了导师的心血和汗水。他渊博的学识、严谨的治学态度、高度的责任心和精益求精的科研精神深深的影响着我。在读研期间，他在学习上总是毫不吝啬的将知识传授给我，带领我参与科研工作，锻炼我撰写科研论文的能力；在生活上也给予了很多帮助，对我的教导受益匪浅。在此，我要向您表示最真诚的感谢！在此，谨向***导师致以衷心的感谢和深深的敬意！

岁月清浅，时光潋滟。感谢朝夕相处的216的室友、老师和同学们，给予三年朴实无华的陪伴，知我悲喜，解我困顿。

再者，感谢给我提供调查和调研机会的A公司，能够让我了解到配送中心拣选作业流程及相关业务数据，感谢各位一线操作人员的配合，感谢管理人员在管理经验方面的分享。

此外，我还要感谢我的家人，尤其是我的父母，感谢他们对我的无私付出、大力支持和关心照顾。

最后，谨向在百忙之中审阅我论文以及答辩委员会的各位专家、教授表示由衷的感谢，对于您提出的宝贵意见，我会认真思考虚心接受。