硕士学位论文

带时间窗取送货车辆路径规划与系统实现

VEHICLE ROUTING AND SYSTEM IMPLEMENTATION FOR PICKUP AND DELIVERY PROBLEM WITH TIME WINDOWS

郭宋湖

哈尔滨工业大学

2019 年 12 月

国内图书分类号：TP399　　学校代码：10213
国际图书分类号：621.3　　密级：公开

工程硕士学位论文

带时间窗取送货车辆路径规划与系统实现

硕士研究生：郭宋湖
导　　师：黄荷姣教授
申请学位：工程硕士
学　　科：计算机技术
所在单位：哈尔滨工业大学 (深圳)
答辩日期：2019 年 12 月
授予学位单位：哈尔滨工业大学

Classified Index: TP399
U.D.C: 621.3

A dissertation submitted in partial fulfillment of the requirements for the professional degree of Master of Engineering

VEHICLE ROUTING AND SYSTEM IMPLEMENTATION FOR PICKUP AND DELIVERY PROBLEM WITH TIME WINDOWS

Candidate: Songhu Guo
Supervisor: Prof. Hejiao Huang
Academic Degree Applied for: Master of Engineering
Specialty: Computer Technology
Affiliation: Harbin Institute of Technology,Shenzhen
Date of Defence: December, 2019
Degree-Conferring-Institution: Harbin Institute of Technology

摘 要

随着互联网和经济全球化的发展，物流在制造相关的企业中占据越来越重要的作用，如何提高生产制造速度，降低物流成本是企业关注的重要方向之一。Milk-Run(循环取货) 作为一种新的物流模式，在小批量、多频次的物料运输中能有效降低库存，提高车辆利用率，减少运输成本。本课题从车辆路径规划角度研究 Milk-Run 模式下的带时间窗取送货的车辆路径规划问题 (Pickup and Delivery with Time Windows, PDPTW)，该问题是一个 NP-hard 问题，没有多项式时间复杂度的求解算法，现实中在零部件运输、快递配送等行业普遍存在该问题。目前国内外对 PDPTW 的研究的不多而结合 Milk-Run 模式的更少，因此研究 Milk-Run 模式下的带时间窗取送货车辆路径规划问题，具有一定的现实意义。

针对带时间窗取送货车辆路径规划问题，本课题设计了大邻域搜索的改进算法，外部加入多次重启策略，使得结果更稳定。算法内部提出操作算子间自适应选择策略以及操作算子内的参数向量自适应策略以提高搜索效率和邻域多样性。本算法在 Li_100 标准数据集下取得车辆数为目标的最优解，并在以平均装载率为第二目标下，与模拟退火嵌套禁忌搜索算法做对比，本算法在规则分布、随机分布以及混合分布的数据集上均能取得更优的结果。

根据实际应用场景，本课题进一步拓展了上述改进大邻域搜索算法的使用场景，创新性地引入休息时间窗约束和白、夜班换班的约束。针对休息时间窗约束，通过分段讨论并设定线性惩罚函数的方式来优化算法在该时间段的路径安排。针对白/夜班换班约束，本算法采取基于时间聚类的方式，先将订单按照各天的白班和夜班各自进行聚类，然后对各自班次的订单运用上述改进大邻域搜索算法进行求解，最后对所得各个白班和夜班的路径进行联合优化。通过与遗传算法的方案对比，本算法在所得车辆数目相同的情况下，能达到更高的车辆装载率。

结合上述算法，本课题设计带时间窗取送货的车辆路径规划系统。该系统后端采用 SSM（SpringMVC 、Spring 、MyBatis）框架，浏览器端基于 WebGIS 技术并使用百度地图 API 以及 Echarts 库进行路径和数据的展示。系统主要包含三大功能，客户的订单信息管理、路径规划安排、地图展示。系统执行过程为：后台获取有效订单在前端展示并提供增加、查询等操作，服务器对所有订单运行车辆路径规划模块进行路径规划，最后在前端将各个车辆的路线规划信息进行展示。

关键词：车辆路径规划问题；路径优化；时间窗；大邻域搜索

Abstract

With the development of Internet and economic globalization, logistics plays an increasingly important role in manufacturing-related enterprises. How to improve the manufacturing speed and reduce the logistics cost is an important direction for enterprises to focus on. As a new logistics mode, milk-run can effectively reduce inventory, improve vehicle utilization rate and reduce transportation cost in small batch and multi-frequency material transportation. This topic studies the Pickup and Delivery Problem with Time Windows (PDPTW) in the milk-run mode from the perspective of vehicle routing. This Problem is a NP-hard problem with no polynomial time complexity solution. In reality, this problem is common in parts transportation, express delivery and other industries. At present, there are few researches on PDPTW at home and abroad, and even fewer combined with the milk-run mode. Therefore, it is of certain practical significance to study the vehicle routing of pickup and delivery problem with time windows under the milk-run mode.

Aiming at the vehicle routing problem with pickup-delivery and time Windows, the improved algorithm of large neighborhood search is designed in this project. Multiple restart strategies are added externally to make the results more stable. The adaptive selection strategy between operators and the parameter vector adaptive strategy within operators are proposed to improve the search efficiency and neighborhood diversity. This algorithm obtains the optimal solution with the number of vehicles as the target under the Li_100 standard data set, and compared with the simulated annealing nested tabu search algorithm under the average loading rate as the second target, this algorithm can obtain better results on the regular distribution, random distribution and mixed distribution data sets.

According to the actual scene, this paper further expands the use scene of the above improved large neighborhood search algorithm, and innovatively introduces the constraint of rest time window and white and night shift. For the constraint of the break time window, the path arrangement of the algorithm in this time period is optimized by means of piecewise discussion and setting up the linear penalty function. In view of the constraints of white shift and night shift, this algorithm adopts a time-based clustering method, first clustering orders according to the day shift and night shift, then solving

the orders of each shift using the improved large neighborhood search algorithm, and finally optimizing the paths of each day shift and night shift. Compared with the genetic algorithm, this algorithm can achieve higher vehicle loading rate with the same number of vehicles.

Combined with the above algorithms, the vehicle routing system for pickup and delivery problem with time windows is designed. The back-end of the system adopts SSM (SpringMVC, Spring, MyBatis) framework, and the browser is based on WebGIS technology and uses baidu map API and Echarts library for path display and data display. The system mainly contains three functions, customer order information management, vehicle routing arrangement, map display. The execution process of the system is as follows: obtain valid orders in the background and display them in the front end to provide operations such as increase and query, etc. The server performs path planning on the vehicle routing module for all orders, and finally displays the route planning information of each vehicle in the front end.

Keywords: vehicle routing problem, path optimization, time windows, large neighborhood search

目 录

第 1 章 绪论

1.1 课题研究背景

随着经济全球化和互联网的高速发展，商品的制造和流通过程逐步增速，一个产品在当今环境下的生产制造，演变成本土组装制造、全球零部件采购以及全球销售的模式，以降低产品管理和制造成本，提高盈利空间。在此期间，企业普遍面临着高额的物流成本，特别在复杂制造业如汽车制造业和飞机制造业。在复杂制造中，面对着种类庞杂各异的机械部件、每个零部件具有不同的结构特点和技术指标要求，该情况大大增加了供应链的管理难度。制造业亟需改进供应链管理、优化物流运输以适应全球化背景下的生产制造需求。据公开数据显示，在 2018 年中国社会物流总费用占 GDP 的 14.4%，日本社会物流总费用与 GDP 占比为 6.9%，美国只有 8.5%，不仅高于发达国家，在经济发展水平基本相当的金砖国家相比也偏高，例如印度为 13.0%，巴西为 11.6%。我国物流产业发展还有较大空间，优化物流成本是提高企业竞争力的关键之一。

Milk-Run 最早源自于乳制品行业的运输问题，在该问题中，送奶员按照规划好的路线将装满牛奶的瓶子运往各个用户家里并在送完后回收空瓶返回出发地。该过程对应到企业中则表示为货车沿着设计好的路线从生产中心出发到一个个供应商去装载原材料，最后把空箱带回公司，如图 1-1 所示，该模式所面对的问题为带时间窗取送货的车辆路径规划问题。Milk-Run 模式适合小批量、多频次的货物运输，已被应用于各行各业中，例如零部件运输、生鲜运输、快递运输，最著名的应用样例是丰田采用 Milk-Run 模式对零部件进行管理，在使用 Milk-Run 模式后优化了车辆的配送安排，使得在相同条件下需要更少的车辆、花费更少时间以及行驶更少的距离，从而实现节约物流成本的目的。因此针对该模式下带时间窗取送货车辆路径规划问题的不断研究和优化，在进一步降低企业物流成本中起着重要作用，具有一定的研究价值和应用前景。

1.2 国内外研究现状

Milk-Run 是在 JIT（Just In Time）供应和精益物流思想下产生的有效物流管理模式，是整合精益物流战略的重要元素[1]，零售业公司 7- Eleven 采用了 Milk-Run 模式[2]，丰田也使用 Milk-Run 的模式支持其在日本和美国的 JIT 生产系统。按照生产流程上划分，Milk-Run 的研究集中于优化零部件的入厂物流、时间耗费、库

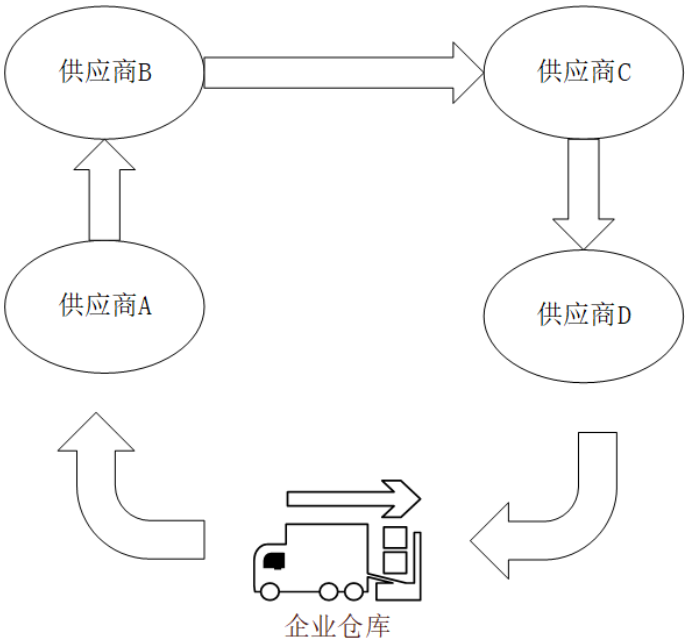

图 1-1 Milk-Run 过程图

存耗费、成本耗费等方面来降低生产制造成本，提高竞争力。在有关 Milk-Run 的主题上，可以分为信息系统方面的研究、车辆路径规划方面的研究、数值模拟方面的研究、Milk-Run 应用方面的研究。本文集中于基于 Milk-Run 的带时间窗车辆取送货问题（Pickup and Delivery Problem with Time Windows，PDPTW）的路径规划研究。PDPTW 是车辆路径规划问题（Vehicle Routing Problem，VRP）的一个子方向，在实际应用中 VRP 延伸出许多变种。根据附加约束的不同可分为带装载能力约束的车辆路径问题（Capacitated Vehicle Routing Problem, CVRP），带时间窗的车辆路径问题（Vehicle Routing Problem with Time Windows, VRPTW），带回程的车辆路径问题（Vehicle Routing Problem with Backhauls, VRPB）, 带取送货的车辆路径问题（Vehicle Routing Problem with Pickup and Delivery, VRPPD）。

如果以车辆路径规划问题的解决方法来进行分类则又可分为精确算法、启发式算法两个类别。

（1）精确算法

精确算法是指对问题通过严谨的数学公式推导从而建立该问题的数学模型，并依据数学原理提出问题最优解策略的算法。因为精确算法以数学为理论基础来寻找问题优解，且是从所有可能存在的可行解集合中依据一定原理去寻找最优解，所以随着客户点规模的逐步增加，算法的搜索空间和运算量将呈指数规模增长。因此，目前的精确算法主要以求解小规模问题为主。

已知用于求解 PDPTW 的精确算法有列生成法[3]、分支定界法[4,5]、分支定价法[6][7]、精确算法与其他算法的混合算法[8] 等等。

（2）启发式算法

传统启发式算法包括两阶段改进法[9]、C-W 节约法[10]、最邻近法[11]、扫描算法[12] 和最近插入法等。这些算法策略丰富多样，结构简单。但是当问题的规模逐渐变大时，则越来越难以得到优解。而新型元启发式算法如遗传算法、蚁群算法、模拟退火算法等依据相应的原理对所得初始解进行优化，最终能求得次优或者最优的结果，因此是目前该问题的主要研究和求解方法。元启发式算法可分为三类：

a) 单一元启发式算法求解 PDPTW 问题。例如遗传算法[13]、禁忌搜索算法[14]、蚁群算法[15]、粒子群算法[16]、大邻域搜索[17] 等。上述方法，利用其元启发算法本身原理，并针对 PDPTW 问题作出适应性设计和改造，在推广该算法于其他领域问题的研究中具有一定借鉴意义。

b）元启发式算法间的嵌套使用来求解 PDPTW 问题。例如 Li 和 Lim[18] 的禁忌搜索嵌入模拟退火算法，Alaia 和 Ben[19] 的粒子群算法结合遗传算法应用于解决多车场的 PDPTW 问题，Ben[9] 的两段式策略同时结合模拟退火和大邻域搜索算法。上述方法结合不同算法各自的优点，一定程度上弥补了自身的不足，在数据集中计算的结果值表现优异。

c）新型元启发式算法求解 PDPTW 问题。如 Créput J C 和 Koukam A 的进化算法[20]，Velasco 和 Castagliola 等的模因算法[21]，Cherkesly 和 Desaulniers 等的基于人口模型的启发式算法[22]，王超和刘超等应用于带时间窗同时取送货的车辆路径规划问题的布谷鸟算法[23]。上述算法为新近提出的算法理论并结合到该问题中，具有一定创新和借鉴意义。同时多数新型算法在设计之初就避免了其他算法的缺点，例如相对于遗传算法，布谷鸟算法具有参数少、求解速度快的优点。

启发式算法因其可以与现代计算机技术相结合所以在求解大规模问题时相比于精确算法有更好的表现，也因此成为目前车辆路径规划主流的方法。目前启发式算法结构多为：利用传统启发式算法如 C-W 节约法、Solomon 插入法[24] 等生成次优的初始解，然后再利用元启发式算法对构造的解进行再优化。

1.2.1 国外研究现状

带时间窗取送货的车辆路径规划问题（PDPTW），需要在满足客户点服务时间窗和车辆容量要求外，还需满足不同客户取货和送货顺序和成对要求，以及所有车辆从车场出发并最终都回到车场的要求。该问题下，车辆可以利用中间路途，顺路完成其他订单的送货或取货，有效的提高了车辆的装载率。如果只单纯的取货或者送货则装载率会不断降低或者增加，车辆的行驶距离因该限制而被大大延长。PDPTW 问题最早由 Wilson[25,26] 提出和解决，之后在 2000 年 Nanry and Barnes[27]

针对 PDPTW 问题提出了第一个启发式算法。近年来带时间窗取送货的车辆路径规划问题取得研究整理如下。

Li 和 Lim 于 2003 年提出禁忌嵌入模拟退火的启发式算法[18]，该算法结合了禁忌表和模拟退火跳出局部循环的特点，高效的求解了 PDPTW 问题，同时该论文依据 Solomon 数据集设计成适合 PDPTW 的 Li 数据集，是第一个 PDPTW 的公开数据集，并公布了该算法的求解结果，算法能求得大部分数据的最优结果。

Bent[9] 于 2006 年提出了第一个两阶段混合算法用于解决 PDPTW 问题，在第一阶段使用简单的模拟退火算法来减少路线数量和优化解质量，而第二阶段使用大邻域搜索（LNS）来减少总旅行成本。该算法在 Li 数据集中求得许多比 Li 和 Lim 算法更好的解。同时验证了两阶段策略的有效性以及 LNS 在解决 PDPTW 问题的优越性，为后续 PDPTW 算法的研究提供了不同的策略。

Ropke 和 Pisinger 于 2006 年提出自适应大邻域搜索算法[28]，通过大邻域搜索算法的删除重建操作产生多样化新解，在最小化路线和距离上均显示出良好的效果，该算法还处理了多车场 PDPTW 的情况。

Hezer 和 Seda 等于 2011 年首次采用基于细菌觅食的新型优化算法（BFOA）[29]来解决同时取送货问题，该算法以总距离最小化为目标并与插入算法作对比，结果显示该算法优于插入法。

Cherkesly[22] 于 2015 年提出后进先出（last in first out ，LIFO）装载模式的 PDPTW 算法，通过 LIFO 模式可以减少车辆在卸下和装上物品时的时间耗费，并提出一种基于人口的元启发式方法，能够更快地处理大型实例。同年，Nalepa 和 Bocho 在 PDPTW 的并行加速计算方面提出了在引导弹射搜索算法（P-GES）中划分搜索空间的并行策略[30] 来求解 PDPTW 问题，同时也在 2017 年提出了基于岛模型的并行模因算法[31]。

Ghilas 和 Demir 等[32] 在 2016 年为 PDPTW 问题增加预定时间线路的约束条件，并使用自适应大邻域搜索算法求得在已知若干条必经路线下的路径规划安排，使得算法求解方案更贴合实际情况。

近几年来，针对 PDPTW 问题的研究在原有基础上拓展了多种方向。例如多种新约束如固定路线下、固定取送货对信息、多周期等情况下该问题的求解，还包括不同背景下如打车、飞机运输等背景下该问题的求解等。具体有 Al Chami 和 Zaher 等[33] 于 2018 年采用遗传算法结合若干贪婪随机自适应搜索（GRASP）策略的方法解决了带多周期的 PDPTW 问题，算法能在较短时间内求出该问题的高质量解。Peng 和 Al Chami[34] 针对带有时间窗和成对需求的车辆规划问题（Selective Pickup and Delivery Problem with Time Windows and Paired Demands ，SPDPTWPD）

提出粒子群与局部搜索相结合的算法用于解决该类问题，结果显示该算法对该问题的求解效率优于遗传算法。Bertsimas 和 Chang 等[35] 将空运中乘客和货物的运输问题抽象成 PDPTW 问题，并采用初始化启发式、局部启发式算法以及列生成算法相结合的形式求解该问题，将飞机总延误时间和飞行时间减少了 8%-12%。

1.2.2 国内研究现状

国内在 PDPTW 问题方向起步虽然比较晚，但在初始路径方案的快速求解、多种路径的启发式优化策略、新型优化算子的构造等方面均有成果。具体内容如下。

贾永基于 2004 年采用禁忌搜索算法[36] 对于带时间窗取送货车辆路径规划问题进行快速求解。算法使用两阶段的思想，第一阶段采用插入法求得初始解，第二阶段使用禁忌搜索对解的质量进行优化。

潘立军和符卓[37] 于 2012 年提出时差的插入策略，设计了快速时差插入和最优时差插入方法，并结合遗传算法求解带时间窗的取送货问题，实验结果显示，该算法的求解质量高于已有同类文献算法。

王超和穆东等[38] 于 2015 年采用通用的混合整数规划模型来最小化路线成本，并开发出一种并行模拟退火（p-SA）算法，该算法包含剩余容量和径向附加（RCRS）插入的启发式算法。算法与遗传算法的结果作比较，展现出更好的精度结果。

黄务兰和张涛等[39] 与 2016 年提出一种改进全局人工鱼群的算法应用于带时间窗取送货的车辆路径规划问题求解，并将时间窗与车容量加入适应度函数中，在以最小化发车数和路程距离的目标下，该改进算法实验表现优于并行模拟退火（p-SA）算法。

王超和刘超等[23] 于 2018 年创新性提出使用离散布谷鸟算法来求解 PDPTW 问题，并设计在莱维飞行位置更新期间使用 2-opt 算子进行路径内部搜索，用 swap、shift 等算子对路径间进行优化，在寄生巢穴的位置更新期间使用 relocate 算子和 exchange 算子进行路径内搜索，使用广义交换法（Generalized Exchange，GENE）法来进行路径间搜索。在标准数据集中，离散布谷鸟算法求得了多数数据集的最优解。王旭坪和李新宇[40] 针对不同车型的同时取送货车辆路径问题，引入时空距离的概念，并设计两阶段启发式算法，第一阶段采用聚类簇划分和就近原则配送的方式求得初始解，第二阶段采用变邻域搜索算法对初始路径优化。采用时空距离的变邻域算法在求解质量和算法收敛速度方面均得到大幅度提高。

张庆华和吴光谱[41] 于 2019 年将模因算法应用于带时间窗同时取送货问题，使用引导弹射搜索（GES）[42] 方法来获得初始种群，使用边界组合交叉（EAX）方法来产生下一代，并通过邻域搜索算子不断优化后代，该算法能取得小规模数据集

的全部最优结果。

1.3 本文主要研究内容和组织结构

1.3.1 主要研究内容

本课题的研究目的是针对带时间窗的取送货车辆路径规划问题探讨进一步的优化算法，并结合实际场景引入新的约束条件，并讨论结合新约束下该问题的求解方案，最后结合上述算法，设计和实现带时间窗取送货车辆路径规划系统。具体研究内容包括：

（1）带时间窗取送货车辆路径规划问题的大邻域搜索算法设计。分析当前问题的约束条件，针对已有算法邻域探索深度不够、各算子无法协调搭配求解的问题，设计算子内参数更新策略以及算子间协调选择策略。算子间通过参数记录下各个算子对解产生的影响，以此辅助算法选择操作算子。算子内通过执行该算子产生的新解是否更差，来判断算子的参数是否需要改变，以此更新算子内的参数向量。算法设计了新的操作算子，以增加邻域搜索多样性和搜索效率。实验采用 Li_100 数据集，分别测试了规则性地点分布、随机性地点分布、混合型地点分布下的路径规划效果，在车辆数目为第一目标下，选取该数据集的当前最优解与本算法做对比，来研究算法在优化车辆数方面得效果，结果显示，本算法均达到数据集的最优车辆数目。以车辆装载率为第二目标时与 Li 和 Lim[18] 的模拟退火嵌套禁忌搜索算法做对比，结果显示本算法在大部分数据集下均能达到更优的平均装载率。

（2）引入新约束条件的 PDPTW 问题算法设计。根据现实场景需求，加入休息时间窗约束、白/夜班换班约束情况。对于休息时间窗约束，算法分段考虑车辆到达客户点时间与休息时间窗的各个情况，并设置线性惩罚函数。对于白/夜班换班约束，算法按照各天的白班和夜班进行聚类划分，针对各个班次内的订单，先使用算法求得初始车辆路径规划，之后将各班次规划的路径进行联合优化以得到更优结果。本算法与基于遗传算法的带时间窗取送货车辆路径规划算法做对比，结果显示本算法能达到更高的车辆装载率。

（3）设计 Milk-Run 模式下的带时间窗取送货车辆路径规划系统。本课题结合企业应用实际，利用真实需求数据，结合 SSM（SpringMVC 、Spring 、MyBatis）框架完成了带时间窗取送货的车辆路径规划系统，该系统包括 PC 端模块和数据处理模块两部分。PC 端模块主要负责于浏览器端展示系统的相关信息以及路径规划的结果，数据处理模块主要负责记录系统各个部分的数据表结构信息以及在服务器进行车辆路径规划的计算。

1.3.2 组织结构

第一章，绪论。本章节阐述了论文研究的理论、应用背景、国内外的研究现状以及本论文研究的问题和内容。

第二章，简要介绍 Milk-Run 模式，针对带时间窗取送货车辆路径规划问题描述了传统启发式求解算法和元启发算法为第三章算法的提供预备知识。内容包括 PDPTW 问题的初始解构造算法例如节约法、Solomon 插入法、扫描算法以及元启发式求解算法，如模拟退火算法、遗传算法以及大邻域搜索算法。

第三章，详细介绍了本课题的改进大邻域搜索算法。包括算法在操作算子间、操作算子内的改进策略，并使用 Li_100 数据集进行对比实验分析，讨论了车辆数目标以及装载率目标下本算法的表现。之后讨论引入供应商休息时间窗约束以及白夜班划分约束下 PDPTW 问题的求解方案，提出带休息时间窗与白/夜班划分的车辆路径规划算法，并与遗传算法作对比，实验结果显示本算法在车辆装载率方面表现更优。

第四章，设计了带时间窗取送货车辆路径规划系统。介绍了系统的两个功能模块，包括 PC 端功能模块，数据处理功能模块，以及各个模块内的详细功能内容。还介绍了系统使用的技术，以及该系统在现实中的应用价值。

第 2 章 Milk-Run 模式介绍与相关算法

2.1 Milk-Run 模式简介

Milk-Run 的概念最初起源于乳制品行业，该背景可抽象为一个运输网络图，以其中一个站点为配送中心来访问其他所有站点，并根据每个站点预先定义的时间窗约束进行访问，从而实现所有站点取货和送货的物料需求。当每个单站的装载量小于卡车装载量时，Milk-Run 模式就非常经济。区别于供应直销模式的送货入场，Milk-Run 模式按照预先计划的路线前往供应商和生产企业进行取货和送货，能有效利用供应商和厂家间的信息并进行协同配合，从而达到优化车辆路径的效果。目前 Milk-Run 的研究主要为如何最大程度的减少空运行情况、降低库存水平和减少运输里程等方面。

对于优化库存方面，提前制定计划，对流程的调度进行优化并且尽量准时交货是取得库存成本减少的关键同时也是达到 JIT（Just In Time) 效果的关键。采用 Milk-Run 的模式能尽可能地消除由于等待而产生的的库存储备问题和零部件因为长时间存储导致缺陷而造成的浪费情况。现实中，如果将必要的零件批量生产并准时运输到装配线，则能减少装配线附近物料的囤积问题。通过该方式在准时制生产开始前就提供材料，使产品的开始生产和原料运输到达的时间接近同步，促进生产制造按照计划好的节拍进行即可实现准时生产。在实际中，JIT 在丰田生产系统中扮演着重要角色，能够在必要的时间按必需的数量交付必要的零件，使得现有库存一直保持在最低水平。JIT 的生产过程能够缩短从材料输入到完成工作的交付周期。

现实中，Rachman 和 Dhini 等[43] 用 Milk-Run 系统成功的解决印度尼西亚某一汽车公司的车辆路径规划问题，并最大程度地降低成本。该 Milk-Run 系统成功的应用于该公司与印度尼西亚所有供应商间的物流管理。该系统采用差分演化（DE）算法进行路径优化求解，实验对该模式下的系统与原始系统在车辆利用率和总行驶距离方面进行比较，新采用的 Milk-Run 系统在一天之内的总里程为 6847.10 公里，而公司原始系统规划的路线总里程是 8077.5 公里。总行驶距离减少量等于 1230.4 公里占原来路程的 15.23%。在车辆的平均利用率方面，新采用的路径规划系统提高了 22.88%的车辆利用率。原始使用卡车的租金为 41450000 卢布，经过优化后的卡车租金 31000000 卢布。采取 Milk-Run 规划的路线每天可以节省 10450000 卢布的卡车租金，达到 25.21%。因此该系统的研究具有一定的实际价值。

车辆路径规划是 Milk-Run 的主要研究方向之一，本课题研究 Milk-Run 模式下带时间窗取送货车辆路径问题的算法以及优化策略，以减少车辆数目、提高装载率，求得该目标下最优车辆路径安排。

2.2 带时间窗取送货车辆问题典型算法

2.2.1 节约法

由于 VRP 是 NP-hard 问题，所以在算法优化开始前产生一个有效的初始解非常重要。C-W 节约法是 1964 年由 Clarke 和 Wright 提出的，该算法的思想是：假定配送中心派车辆为两个客户单独配送货物，如图 2-1 所示。那么该车辆的行驶距离为 $2(L_{01}+L_{02})$，若采用混合配送的方式即车辆采用不返回配送中心的方式，如图 2-2 所示，则行驶距离为 $L_{01}+L_{12}+L_{02}$，因此可得从单独配送转换成混合配送的行驶过程中路线的节约值为

$$\triangle L = L_{01} + L_{02} - L_{12} \tag{2-1}$$

由公式 2-1 可得，计算得到节约值越多的路线所达到优化的效果越好。因此只需对计算所得到的各个点节约值从大到小进行排序，即可得到优化后的路线。

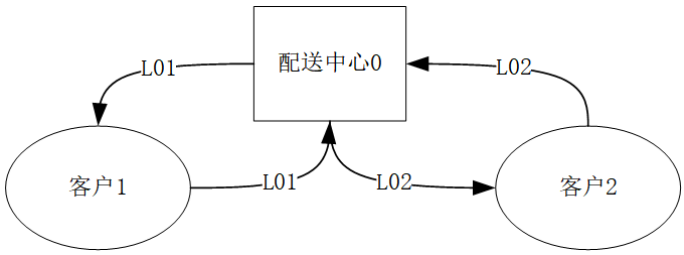

图 2-1 单独配送示意图

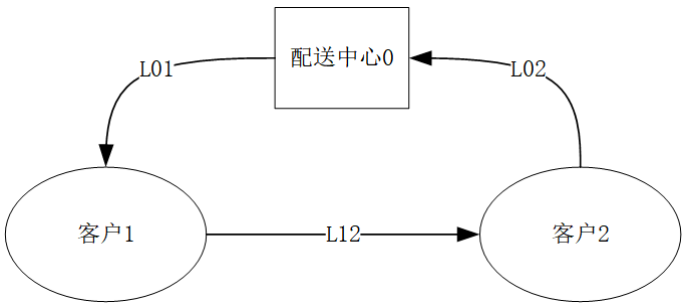

图 2-2 混合配送示意图

2.2.2 Solomon 插入法

在 C-W 节约法中，节约值只考虑距离的节约量对所规划路径的影响，并未考虑其他因素对路径解的影响，而 Solomon[24] 插入法在节约法基础上增加时间因素，并提出了四种插入启发式算法：即节约启发式算法、基于时间的最近邻启发式算

法、插入启发式算法和基于时间的扫描启发式算法。实验结果表示插入算所得的解最优。上述四种启发策略的具体思想如下

记 $m-1$ 个客户所组成的初始路径表示为 $(i_0,i_1,...,i_m)$，其中 i_0=i_m 表示配送中心，其他点为客户点。若在新路线中插入新客户点 u，则该点在路线的最佳插入位置可由下列公式表示

$$c_1(i(u),u,j(u)) = min[c_1(i_{p-1},u,i_p)], p = 1,2,...,m. \tag{2-2}$$

上述公式 2-2 理解为，单条路径中若要加入新客户点 u，则只需计算 u 点插入到路径中各个空余位置点后所产生的额外费用，选取产生最小费用的那个位置即为最佳插入位置，其中 c_1 表示评价函数。

若需计算新客户点 u^* 在多条路径的最佳插入位置，则该最佳插入位置的计算公式如下

$$c_2(i(u^*),u^*,j(u^*)) = optimum[c_2(i(u),u,j(u))] \tag{2-3}$$

上述公式 2-3 中，u^* 表示未在路线中的新插入点，$i(u^*)$ 表示在路线 i 插入未在路径的客户点 u^*，$j(u^*)$ 表示在路线 j 插入未在路径的新客户点 u^*，如果该点无法插入到目前所有路线中，则新建一条路径，该过程直至所有点均插入到路径中为止，其中 c_2 表示评价函数。

上述公式为路径插入点的选取策略，包括单条路径和多条路径的插入点选取策略，其中 c_1、c_2 为评价函数，依据不同参考因素可产生不同的评价函数，Solomon 插入法共有三种评价函数如下所示：

（1）方式一

$$c_{11}(i,u,j) = d_{iu} + d_{uj} - \mu d_{ij}, \mu > 0 \tag{2-4}$$

$$c_{12}(i,u,j) = b_{ju} - b_j \tag{2-5}$$

上述公式 2-4 中，b_{ju} 表示插入客户 u 后，车辆在客户 j 处的服务开始时间，b_j 表示未插入客户 u 时的服务开始时间。

$$c_1(i,u,j) = \alpha_1 c_{11}(i,u,j) + \alpha_2 c_{12}(i,u,j), \quad \alpha_1 + \alpha_2 = 1, \alpha_1 \geq 0, \alpha_2 \geq 0 \tag{2-6}$$

$$c_2(i,u,j) = \lambda d_{0u} - c_1(i,u,j), \lambda \geq 0 \tag{2-7}$$

上述公式 2-6、2-7 表示路径的最佳可行的插入位置是最小化距离与插入时间的加权组合所对应的位置，即最小化访问客户所需额外距离和额外时间的所对应的位置。

（2）方式二

$$c_2(i,u,j) = \beta_1 R_d(u) + \beta_2 R_t(u), \beta_1 + \beta_2 = 1, \beta_1 \geq 0, \beta_2 > 0 \tag{2-8}$$

其中，$R_d(u)$ 表示将新客户点 u 插入到当前线路之后所需要的路线距离，$R_t(u)$ 表示将新客户点 u 插入到当前线路之后所需要的路线时间。第二种启发式插入法旨在选取最小化总路线距离与时间加权和为目标的新客户来插入。

（3）方式三

$$c_{13}(i,u,j) = l_u - b_u \tag{2-9}$$

$$c_l(i,u,j) = \alpha_1 c_{11}(i,u,j) + \alpha_2 c_{12}(i,u,j) + \alpha_3 c_{13}(i,u,j) \tag{2-10}$$

其中，$\alpha_1 + \alpha_2 + \alpha_3 = 1, \alpha_1 \geq 0, \alpha_2 \geq 0, \alpha_3 \geq 0, c_2(i,u,j) = c_1(i,u,j)$，上述公式表示寻找最小化 $c_2(i,u,j)$ 目标的新客户来插入路线中。

将上述插入算法在数据集中测试可以得，方式一所得的结果好于方式二与方式三的结果。本课题采用方式一所对应的 Solomon 插入法。

2.2.3 扫描算法

扫描法[12] 是由 Gillett 和 Miller 提出。该方法首先确定配送中心位置，以此为旋转中心向客户节点释放一条射线，按照节点区域进行划分组别，然后对各自区域进行求解以生成行车路线。该算法具有一定的鲁棒性，如图 2-3 所示。

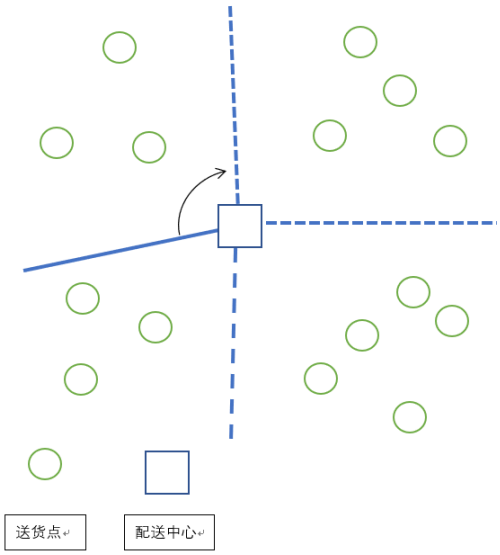

图 2-3 扫描算法示意图

2.2.4 模拟退火算法

模拟退化算法[44] 是一种通用概率演算法，最早源自冶金学，借鉴金属退火的

原理为高温加热金属后其固体内部处于无序状态，等待其慢慢退火冷却后，粒子逐渐恢复稳定有序状态，直至达到基态下内能减为最少。该过程的流程图如图 2-4 所示。由图可得，初始阶段模拟退火算法依据问题本身获得初始解并初始化温度、

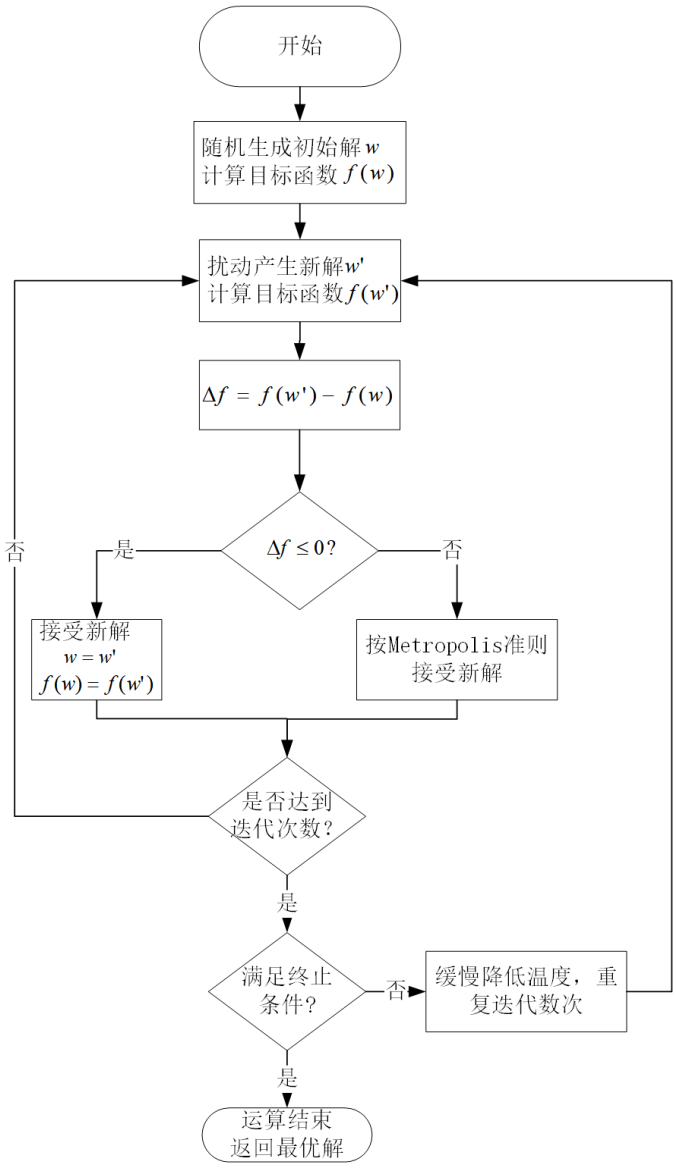

图 2-4 模拟退火算法流程图

迭代次数等参数，然后根据实际问题约束产生扰动制造临近新解，依据评价函数计算当前解与历史解的差值。如果差值小于等于 0，则接受当前解，否则以概率 $e^{(-\triangle f/T)}$ 与随机数做比较，若计算所得概率大于随机值，则邻近解替换当前解。如果概率计算值等于随机值则算法不做处理。若概率值小于随机值则接受 $f(w')$ 为新的当前解同时减去迭代次数并对初始温度降温。算法最后如果满足终止条件则输出当前解作为最优解，然后结束程序。本课题算法借鉴模拟退火的概率接受解策略，来促进算法解向最优解方向探索。

2.2.5 遗传算法

遗传算法是模拟自然界生物进化论的自然选择和遗传学中生物进化过程的计算模型。自然界生物的进化对应着优胜劣汰、适者生存，遗传学中则表示着染色体的选择、交叉、变异的过程。染色体通过不断的交叉变异，最终最适应环境的个体染色体保留下来，相反，劣势的个体则淘汰。

在遗传算法中，一个种群也代表待优化集合，可表示成 $f(t)$，其中用 t 表示种群的代数。因此下一代种群可表示为 $f(t+1)$，种群的个数用 N 表示，也代表种群的规模，一般情况为固定值。种群中每个个体的染色体可表示成 $x_i(t)$，其中 i 表示第 i 个个体，t 表示第 t 代，x 所代表的染色体内容则由待解决问题通过一定策略映射成染色体 x 的形式，将其转化成遗传算法的迭代求解过程。染色体的编码形式有二进制码、自然数编码、格雷码、浮点数码等，其中 VRP 问题大多使用自然数编码来表示染色体。初始解转化为染色体编码型后即可开始使用遗传算法来解决问题，一般遗传算法流程如图 2-5 所示

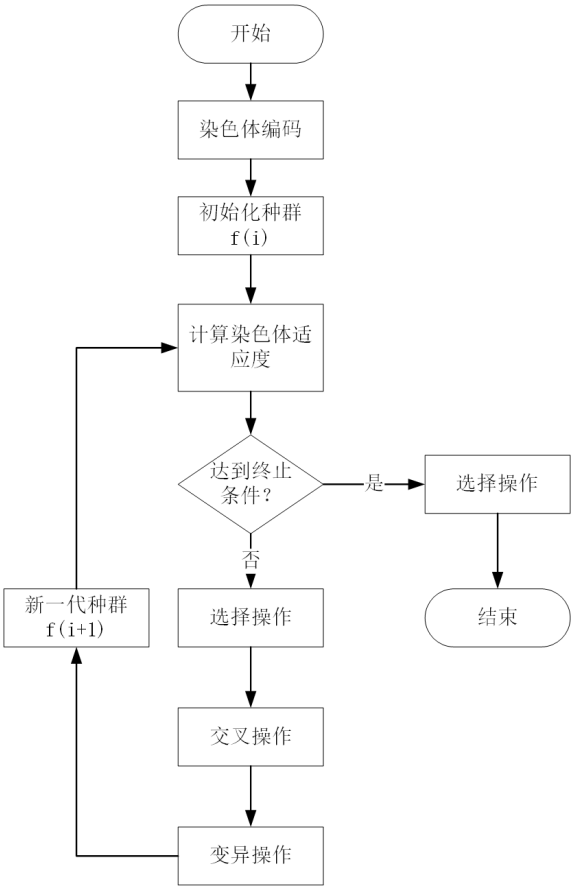

图 2-5 遗传算法流程图

以 VRP 问题为参考例子，假设有 m 辆车，初始化种群时每个个体对应一个解，每个解包含该 m 辆车的路径规划安排。可以采用不重复自然数的编码方式将所有

点用自然数进行编码，例如染色体为（1,2,3,4,5,6,7,8），以车容量为约束的条件按顺序如果分为（1,2,3）、（4,5）、（6,7,8），则该条染色体代表的解需要三辆车来完成。获得这些初始解之后，记录各自原始的适应度，留作做后续对比用。之后通过选择、交叉、变异等行为产生新解，并对照原始解更新下一代种群，不断进行该过程，直到满足终止条件获得次优解或者最优解。遗传算法是本课题的对比实验算法之一。

2.2.6 大规模邻域搜索算法

大规模邻域搜索 (Large Neighborhood Search，LNS) 算法由 Shaw[45] 提出。其搜索过程主要包括毁坏-重建两个部分。毁坏步骤即从已有路径通过若干策略和规则来从路径中删除一定量节点。重构步骤则把删除的节点重新插入到路径中，针对不同的插入思想对应着不同插入算子。这其中，如果毁坏策略选择删除一条完整的路径，且重建策略将该条路径上的所有点完整的重新插入到其他路径中。若全部插入成功，则该过程达到节约一辆车花费的作用，因此毁坏重建策略在降低车辆数的目标上具有优异效果。该算法的伪代码如算法 2-1 所示。

```
Input: InitialSolution
Output: BestSolution
BestSolution = InitialSolution
CurrentSolution = InitialSolution
while the termination criterion is not satisfied do
    S = CurrentSolution
    S = Destroy(S)
    S = Re-insert(S)
    if c(S) < c(BestSolution) then
        BestSolution =S
        CurrentSolution =S
    end
    else if Accepted(S, CurrentSolutiont) then
        CurrentSolution = S
    end
end
```

算法 2-1 大邻域搜索算法 LNS

该算法的传入参数为一个初始路径解，之后在循环过程中对解不断进行删除、

重新插入操作以获得优化解，当陷入局部最优时对次优解进行一定概率的接受，使得该算法避免陷入局部最优的情况。达到终止条件后即可得到最终解。该算法所得的解质量较高，同时因该算法框架简单且拓展性好而被广泛研究和采用。

2.3 本章小结

本章简要介绍了 Milk-Run 的概念、优点、面临的各种约束条件、现实中的主要优化的目标以及现有企业应用该模式所取的成果。同时以带时间窗取送货的车辆路径规划问题为本课题研究方向，针对性描述了带时间窗取送货的车辆路径规划问题的解决策略。包括初始解的若干生成策略，本章描述了节约算法、Solomon 插入法、扫描算法等思想策略。以此为基础探讨后续章节涉及到的启发式算法以及各自算法思想与策略，包介绍了以一定概率接受次优解从而跳出局部最优的模拟退火算法，描述了遗传算法在 VRP 问题的解决策略，同时讲解了大邻域搜索算法的结构、思想以及优缺点，并为后续算法做预备知识。

第 3 章 改进大邻域搜索的 PDPTW 算法

3.1 带时间窗取送货车辆路径问题描述

本课题基于 Milk-Run 模式主要有两个研究方向，一个为研究在传统约束下的 PDPTW 算法，约束包括车辆容量约束、供应商时间窗约束、每个订单取货和送货的顺序约束以及所有车辆从车场出发最终全部返回车场的约束，思考基于这些约束的带时间窗取送货车辆路径规划问题的优化算法和改进策略，另一方面为结合现实场景创新性引入休息时间窗约束以及白/夜班换班约束。针对以上约束情况，本课题提出改进大邻域搜索算法，设计算子间的选择策略以及算子内的参数调整策略。算子间通过衡量算子对解的优化情况来评价算子的贡献能力，依此选择算子。算子内参数则通过该算子是否无法使解变好，以此来改变参数的值，从而拓展邻域解，最终求得算法的最优解。在现实场景中则增加了车辆存在白班和晚班的情况，为司机安排路线时需考虑换班的条件，防止司机疲劳驾驶。同时，对于供应商的开放时间窗为较长时间的类型，供应商中必然存在休息时间，因此本文提出带休息时间窗与白/夜班划分的车辆路径规划算法来解决该类现实场景。

PDPTW 问题所对应路径解的取送货流程如图 3-1 所示所示。图中每个点对应一个订单，数字表示取货或者送货的订单数量，实心圆表示取货，空心圆表示送货，方块表示物流中心。每条路径上需满足车辆容量约束、地点时间窗约束、从车库出发和回到车库的约束以及车辆取送货偏序关系的约束，以及在该访问过程中，每个点只能访问一次的约束。

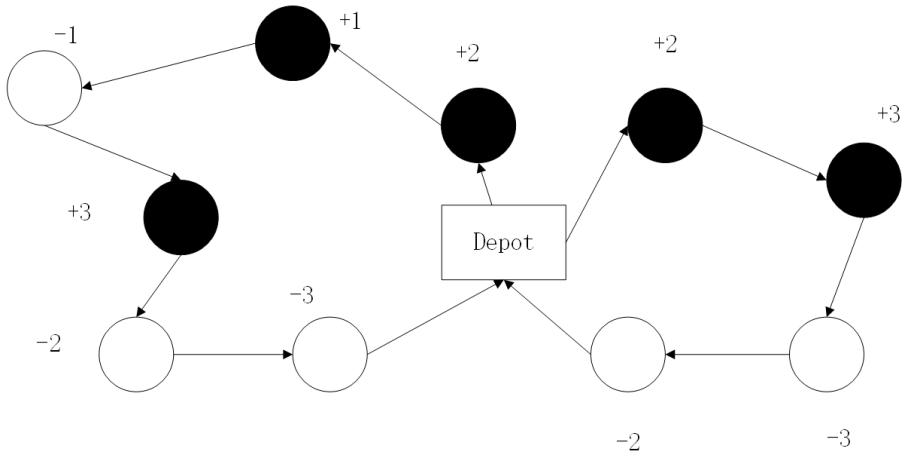

图 3-1 PDPTW 问题取送货路径示意图

3.2 带时间窗取送货车辆路径问题的数学模型

本课题的主要研究内容为 Milk-Run 模式下的带时间窗取送货路径规划问题（PDPTW），具体为单车库的带时间窗取送货车辆路径规划问题。该问题可以描述为: 车库拥有 V 辆车，车容量为 Q，总共有 n 个订单需要完成，其中有 $P^+ = \{1,2,3...,n\}$ 个点为取货点，$P^- = \{n+1, n+2, ..., 2n\}$ 个点为卸货点，车库为点 0 和点 $2n+1$。因此所有点可用 $N = \{0,1,2,...,n,...,2n,2n+1\}$ 表示，除了车库点之外的点用 $P = P^+ \cup P^-$ 表示。针对每个订单请求 i，其取货点为 P_i^+ 位置，送货点为 P_{n+i}^- 位置。每个订单的需求量用 l_i 表示。每个订单的时间窗对应为 $[a_i, b_i]$，其中 a_i 表示 i 点的最早开始时间，i 点的服务时间为 s_i，b_i 表示 i 点的最晚开始时间，定义车辆 v 从点 i 到点 j 所花费的时间为 t_{ij}^v，车辆 v 行驶的路程为 d_{ij}^v。假如一辆车 v 从车库出发到达 i 点，记此处时间为 T_i^v，那么可得出该车的离开时间为 $T_i^v + s_i$。

因此，该问题的决策变量如下所示：

$$X_{ij}^v = \begin{cases} 1 & \text{如果车辆 } v \text{ 从点 } i \text{ 到达点 } j \\ 0 & \text{否则} \end{cases} \tag{3-1}$$

因此该 PDPTW 问题涉及的约束[36] 具体描述如下。

由于一个请求只能被一辆车处理，故每个点只能被访问一次，且每个点进去和出来的数目必须一致。可得约束如下

$$\sum_{v \in V} \sum_{j \in N} X_{ij}^v = 1, i \in P \tag{3-2}$$

$$\sum_{j \in N} X_{i,j}^v - \sum_{j \in N} X_{ji}^v = 0, i \in P, v \in V \tag{3-3}$$

由于所有车辆从车库出发，沿着安排的路线运送货物并且最后返回车库，所以需满足从车库出发的车辆数目与最后返回车库的车辆数目相等，因此可得约束表示如下

$$\sum_{j \in P^+} X_{0,j}^v = 1, v \in V \tag{3-4}$$

$$\sum_{j \in P^-} X_{j,2n+1}^v = 1, v \in V \tag{3-5}$$

由于完成一个订单包含取货和送货两种操作，且该订单只能由同一辆车来完成，所以可得约束表示如下

$$\sum_{j \in N} X_{i,j}^v - \sum_{j \in N} X_{j,n+i}^v = 0, i \in P^+, v \in V \tag{3-6}$$

由于订单存在取货和送货的操作，所以在 i 点取货的时间要早于在 $n+i$ 点送货的时间，可得约束表示如下

$$T_i^v + t_{i,n+i}^v \le T_{n+i}^v, i \in P^+, v \in V \tag{3-7}$$

其中，T_i^v 表示车辆 v 到达 i 点的时间，$t_{i,n+i}^v$ 表示车辆 v 从取货点 i 到送货点 $n+i$ 之间所需的时间。在满足上述取货和送货间的先序关系后，还需满足所规划的路线上，车辆先访问点的时间要早于后访问点的时间，约束表示如下

$$X_{i,j}^v = 1 \Rightarrow T_i^v + t_{i,j}^v \le T_j^v, i, j \in P, v \in V \tag{3-8}$$

$$X_{0,j}^v = 1 \Rightarrow T_i^v + t_{i,j}^v \le T_j^v, j \in P^+, v \in V \tag{3-9}$$

$$X_{i,2n+1}^v = 1 \Rightarrow T_i^v + t_{i,2n+1}^v \le T_{2n+1}^v, j \in P^-, v \in V \tag{3-10}$$

除了上述约束，还需明确车辆在各个客户点需满足的该地点开放时间窗约束，约束表示如下

$$a_i \le T_i^v \le b_i, i \in P, v \in V \tag{3-11}$$

$$a_0 \le T_0^v \le b_0, v \in V \tag{3-12}$$

$$a_{2n+1} \le T_{2n+1}^v \le b_{2n+1}, v \in V \tag{3-13}$$

其中 a_i 表示 i 点的最早开始时间，b_i 表示 i 点的最晚开始时间。在此过程中车辆需满足容量一致性要求，即在离开上一点与到达下一点之间容量的一致性，同时不能超过车的最大容量，约束表示如下

$$X_{i,j}^v = 1 \Rightarrow L_i^v + l_j = L_j^v, i \in P, j \in P^+, v \in V \tag{3-14}$$

$$X_{i,j}^v = 1 \Rightarrow L_i^v - l_{j-n} = L_j^v, i \in P, j \in P^-, v \in V \tag{3-15}$$

$$X_{0,j}^v = 1 \Rightarrow L_0^v + l_j = L_j^v, j \in P^+, v \in V \tag{3-16}$$

$$L_0^v = 0, v \in V \tag{3-17}$$

$$l_i \le L_i^v \le Q^v, i \in P^+, v \in V \tag{3-18}$$

其中，L_i^v 表示车辆 v 行驶到 i 点后的装载量。Q^v 车辆 v 本身的最大装载量。

该带时间窗取送货问题的目标函数为公式 3-19 与公式 3-20。该目标函数包含车辆数目、车辆平均装载率。

$$\min \quad \sum_{v \in V} \sum_{j \in N} X_{i,j}^v \tag{3-19}$$

$$\max \quad \frac{\sum_{v \in V} \max_{i \in N} L_i^v}{V} \tag{3-20}$$

公式 3-19 表示车辆数目标。公式 3-20 记录每条路径下车辆的最大装载量，并求出所有车辆的平均装载量，用以表示该方案的平均装载率目标。

3.3 改进大规模邻域搜索算法

3.3.1 算法内容

大邻域搜索最早由 Shaw 提出。算法核心主要包括毁坏-重建两步骤，毁坏步骤代表删除一定数量的节点，通过删除节点来调整路径间线路。重建步骤则代表着把删除的节点通过一定策略进行恢复。算法对初始解通过操作算子不断迭代优化来产生改进，该过程与局部搜索算法相似但局部搜索算法在不断迭代优化的过程中搜索空间逐步减少，进而容易陷入局部最优解的问题。大邻域搜索算法则把该过程分解为两部分，最终的解由两部分组合产生，大大的拓展解空间搜索范围，克服了局部搜索算法陷入局部最优解后解空间搜索范围减少的缺点。当大邻域搜索算法的毁坏步骤是将一整条路径删除时，重建的步骤若能完整把所有点插入到其他路径中，则该过程实现了降低车辆数的效果，因此大邻域搜索算法在最小化车辆数上具有先天优势。原始的大邻域搜索算法伪代码执行过程在第二章有详细介绍。

本算法在原有大邻域搜索的基础上，在外部增加 K-Restarts 策略，通过对大邻域搜索结果的多次重启运算使得最终得到更稳定的结果，内部则为各个优化算子间设计辅助参数，通过判断该优化算子是否产生更优解，依此来提高算子的选中概率，从而使算法能自动选择优化效果好的算子。算法同样依据新解是否更好，来更新算子内向量参数，以此拓展搜索空间。为防止算法在进入局部最优时出现多次选举的所得算子都为同一个的情况，算法引入随机选举策略，辅助跳出局部解。同时为大邻域搜索算法增加模拟退火的概率接受解机制促使获得最优解。本算法的伪代码如算法 3-1 所示。

算法 3-1 详细描述了改进大邻域搜索算法的各部分思想，在算法第 2 行包含两个新变量：$\rho^- \in \mathbb{R}^{|\Omega^-|}$ 和 $\rho^+ \in \mathbb{R}^{|\Omega^+|}$，分别记录着每种删除算子和重建算子的权重。初始化所有算子权重均为 1。在第 5、6 行中的权重向量 ρ^- 和 ρ^+ 采用轮盘赌博方法来选择使用哪种删除算子和重建算子。其中第 j 个删除算子的选择概率为

$$\varphi_j^- = \frac{\rho_j^-}{\sum_{k=1}^{|\Omega^-|} \rho_k^-} \tag{3-21}$$

Input: a feasible solution x

Output: x_{best}

$x_{best} = x_{current} = x$;

$\rho^- = (1, ..., 1); \rho^+ = (1, ..., 1); rate^- = (0.1, ..., 0.1); m = 0$;

while *not stop criteria* **do**

  select destroy operator based on ρ^- and repair operator based on ρ^+

  if *current* d_{op} = *before* d_{op} **then**

    m=m+1;

  end

  if *m*>*threshold* **then**

    random select destroy operator d_{op}

  end

  $x_{new} = r_{op}(d_{op}(x, rate^-))$

  if $c(x_{new}) < c(x_{best})$ **then**

    $x_{best} = x_{current} = x_{new}; \Psi = \omega_1$

  end

  else if $c(x_{new}) < c(x_{current})$ **then**

    $x_{current} = x_{new}; \Psi = \omega_2$

  end

  else if *accept*(x_{new}) **then**

    $x_{current} = x_{new}; \Psi = \omega_3$

    update destroy operator d_{op} parameter $rate^-$

  end

  else

    $\Psi = \omega_4$

    update destroy operator d_{op} parameter $rate^-$

  end

  update ρ^- and ρ^+

end

算法 3-1 改进大邻域搜索算法 (PLNS)

公式 3-20 依据每种删除算子和重建算子对解的贡献程度动态调整选择概率。下述公式 3-21 针对当前解与历史最优解的关系划分成四种情况，并设置相应调整

权重，四种情况分别对应算法 13 行内容、算法 16 行内容、算法 19 行内容、算法 23 行内容。

$$\psi = \max \begin{cases} \omega_1 & \text{如果新解是新的全局最优解} \\ \omega_2 & \text{如果新解比当前解好} \\ \omega_3 & \text{如果新解被接受} \\ \omega_4 & \text{如果新解被拒绝} \end{cases} \tag{3-22}$$

其中 ω_1、ω_2、ω_3 和 ω_4 是参数。ψ 值越高表示该算子优化效果越好。一般有 $\omega_1 \geq \omega_2 \geq \omega_3 \geq \omega_4 \geq 0$。

算法第 24 行用于更新参数 ρ^- 和 ρ^+，该更新思想是：令 a 和 b 分别是算法最后一次迭代中使用的删除和重建算子的索引，公式 3-22 应用于参数 ρ^- 和 ρ^+ 的更新。

$$\rho_a^- = \lambda\rho_a^- + (1-\lambda)\psi, \rho_a^+ = \lambda\rho_a^+ + (1-\lambda)\psi \tag{3-23}$$

其中 $\lambda \in [0,1]$ 是衰减参数，用于控制权重对删除算子和修复算子性能变化的敏感程度，迭代中未使用到的算子权重保持不变。自适应权重调整的目的是选择产生优解的算子来增加权重，同时减少生成劣解的可能，因为该迭代浪费了时间和空间，该情况可以通过为 ω_4 分配一个低值来实现。

通过公式 3-22 来选择优解算子的过程中，算法在收敛到局部最小值时存在优化时间长或者陷入局部最优解的情况，针对该情况，若由公式 3-22 计算而得的删除算子 d_{op} 在 m 次循环下均选择同一个算子，那么判断本算法陷入局部最优情况。此时引用随机筛选的方式，在所有删除算子中随机选择一个删除算子作为当前执行的算子，通过该方式可以跳出原有搜索方向，算法 4-10 行对此过程进行了描述。

原始情况，利用 ρ 参数来记录各个操作算子对结果是否产生更好的影响，以此来达到自动选择算子的作用，但对于各个算子内部的参数则还没有算法讨论过自适应调整策略，故本算法依据相同原理，增加删除算子的算子内参数向量 $rate^-$，这里选择删除比例参数来作为参数向量的值，初始化向量比例为 0.1。在对解的影响中，如果产生更好的解，则更新 ρ，如果产生更差的解，则更新 $rate^-$，算法通过改变算子参数值，来增加搜索可行解空间，促进寻找到更优解，下列公式为参数向量的更新策略公式包含更新步长。

$$rate^- = \max \begin{cases} \varepsilon_1 & \text{如果新解被接受} \\ \varepsilon_2 & \text{如果新解被拒绝} \end{cases} \tag{3-24}$$

为提升结果的稳定性本算法引入 K-Restarts 机制。对算法运行结果强制进行 K

次重启运算。因为在单次求解过程中有一定概率出现质量较差的次优解，因此记录下 K 次重启仍然保持稳定的算法值，并从中选出最优的解。该方法使得算法结果更稳定，策略如算法 3-2 所示。

```
Input: a solution S, restart times K
Output: improve solution S'
1  while t=0 < K do
2      S'=PLNS(S)
3      if Cost(S')<Cost(S) then
4          S=S'
5          t=0
6      end
7      else
8          t=t+1
9      end
10 end
11 S=S'
```

算法 3-2 K-Restarts 策略

算法第 1 行定义了重启次数 K，在第 2 行执行改进大邻域搜索算法。在第 3-6 行用于判断当前解是否更优，如果得到更优解则更新全局最优解，若不是则计数加一，如果计数达到 K 则说明当前的解已经达到稳定，然后输出该稳定解。

3.3.2 构造初始解算法

初始解的好坏对算法后续优化的工作量有很大影响，一个好的初始解可以减少算法的优化时间，提高求解速度。目前大多数启发式算法的初始解均采用插入法，插入法的思想如下：

（1）初始化一条空路径以及所有待插入客户点集合 U。

（2）对每个 U 中的客户点，计算该点插入当前构造解的最小代价位置并将该客户插入对应位置的路径中。若当前解的路径不存在可以插入的位置，则新建一条路径并将该客户点插入该路径中。

（3）将成功插入当前路径的客户点从待插入集合 U 中移除。

（4）不断执行上述步骤，直至待插入集合 U 为空时算法停止。

本问题的初始解求解流程基本类似，常用的求解算法有节约法、扫描法、Solomon 插入法，这几种算法在第二章均进行了详细介绍。本课题采用 Solomon 插

入法对 PDPTW 问题进行初始解求解。

3.3.3 插入删除启发式策略

大邻域搜索算法通过毁坏-重建策略不断产生新的邻域解，从而获得更优解，但是单一的删除或者插入策略容易使算法陷入局部最优解，为了获得多样化的邻域解，本算法采用的删除算子内容如下。

（1）Random removal heuristic

该算子从全部客户点集 U 中随机选取 ϕ 个客户点，从所得路径中随机删除 ϕ 个客户点。本算法删除点的数量通过该算子的参数向量值，即比例系数 $rate^-$ 与全部客户数 U 的乘积计算所得，该数量与该算子的删除比例 $rate^-$ 有关，而 $rate^-$ 与执行该算子后是否对解进行优化有关。通过该算子能产生一定量的删除客户从而改变路径的结构，为之后行程的重新插入提供更多样化的邻域路径。

（2）Worst removal heuristic

该算子以降低路线的总距离为目标，策略为：从全部客户 U 中选取一个客户点将其从路径中删除，计算并保存删除该点前后的路径节约值，然后对所有客户点重复该过程，即可得到所有客户在解中删除自身所得的节约值，然后对值从大到小排序，选取前 ϕ 大的节约值所对应客户点即为本算法想要删除的客户点。将这些点从路径找到并删除后，所得路径即是本执行本算子后的输出路径。本算法的删除数目 ϕ 是由大邻域搜索的删除比例 $rate^-$ 来控制。算法 3-3 描述了该最坏移除算法的步骤。

Input: a solution S, remove number ϕ

Output: solution S'

1 v is empty
2 **for** *customer i in U* **do**
3 $\quad v_i$ = cost(current solution, solution after remove this customer)
4 **end**
5 sort v from big to small and select the first ϕ numbers
6 S' is the result after remove ϕ customers from S

算法 3-3 最坏移除算法

（3）Related removal heuristic

该算子从客户点相关度角度来选择删除的点。原始 Pisinger 和 Ropke[46] 在解决 PDPTW 问题所采用的相关度指标只考虑客户点间的距离间隔，本算法引入客户点间的开始时间间隔、各自的需求量差异。评价指标如下.

$$r(i,j) = \lambda(d_{P_i,P_j} + d_{D_i,D_j}) + \mu(|T_{P_i} - T_{P_j}| + |T_{D_i} - T_{D_j}|) + \xi(|q_i - q_j|) \tag{3-25}$$

$$R(i,\Gamma) = \frac{\sum_{j\in\Gamma} r(i,j)}{|\Gamma|} \tag{3-26}$$

其中，P_i 表示取货点，D_i 表示送货点，T_i 表示 i 点最早开始访问时间，q_i 表示 i 取货点的需求量，Γ 表示移除点集合。该算子的执行策略为：初始阶段从当前解 S 中随机选取一个客户点将其加入集合 Γ 中，之后对 S 中非 Γ 客户点计算该点与集合 Γ 所有客户点的相似度 $r(i,j)$，并累加取平均求得该点与集合 Γ 的相似度。选取相似度值最小的客户点加入集合 Γ 中，不断重复该过程直至达到要求的删除个数即可。

（4）Random Schedule remove heuristic

为提高路径删除算子的鲁棒性和多样性，设计路径的随机删除算子，该算子随机选取一条路线，删除该路线上所有点。

（5）Node distance removal heuristic

该算子计算每条路径的平均距离占用值，选取占用值最大的路径将其移除。设该条路径的总距离值为 $D(i)$，该条路径上的节点数目为 $N(i)$，则平均距离占用值表示如下

$$f(i) = \frac{D(i)}{N(i)} \tag{3-27}$$

本算法采用的插入操作算子如下：

（1）Greedy insertion heuristic

该算子遍历路径中所有空闲时间段，并计算插入节点前后的代价增量。在路线上的客户 i 与客户 j 中间插入客户 k 所产生的代价增量如下

$$c_i = c_{i,j} + c_{i,k} - c_{j,k} \tag{3-28}$$

其中 $j = 1,...,|\Gamma|$，并且 $i = 1,...,|U-\Gamma|$，$|\Gamma|$ 表示待插入节点集合大小，$|U-\Gamma|$ 表示非 Γ 节点数目。

（2）Regret insertion heuristic

当路径只有很少可行移动的节点时，贪心插入算子经常需要迭代到后期才能满足条件插入。为了解决这个问题，遗憾插入算子使用了 2-regret 准则。令 $\triangle c_i$ 表示将节点 i 寻找最优插入位置后所节约的代价值，计算公式如下

$$i^* = \arg\max_{i\in\Gamma} (\triangle c_i^2 - \triangle c_i^1) \tag{3-29}$$

其中 $\triangle c_i^1$ 表示第一最优插入点所对应的代价节约值，$\triangle c_i^2$ 表示第二最优插入点所对应代价节约值。每轮迭代时依据公式 3-29 选取最佳插入位置，直至将 Γ 中所有客户点均插入路径中。

3.3.4 实验结果

本实验采用 Li_100 标准数据集，数据集包含规则地点分布的 LC 数据集，随机地点分布的 LR 数据集，以及混合分布的 LRC 数据集。由于本数据集 Li_100 有最优的路径规划所得车辆数的结果，因此实验以本文算法计算结果与最优解的值之间做对比，比较车辆数目的优化效果。之后将本算法即改进大邻域搜索算法(PLNS) 与同为启发式算法的模拟退火嵌套禁忌搜索算法做对比，来比较在启发式算法中各自在车辆数的求解效果，目标为车辆数目，通过该实验验证 PLNS 算法在该问题的求解效果，之后以平均装载率为第二目标进行实验，验证 PLNS 算法在提高平均装载率上的效果。

如下表 3-1 所示为 Li 数据集中 lc101 的部分数据。数据格式包含订单任务号、X 坐标、Y 坐标、订单需求量、客户点的最早开始时间和最晚开始时间、客户点的服务时间以及若该点为取货点，则其送货 NO 为该订单送货记录所在位置，同理可得取货 NO 的意思，其中物流中心位置的任务 NO 为 0。

表 3-1 lc101 部分数据展示表

NO	X	Y	需求量	最早开始时间	最晚开始时间	服务时间	取货 NO	送货 NO
0	40	50	0	0	1236	0	0	0
1	45	68	-10	912	967	90	11	0
2	45	70	-20	825	870	90	6	0
3	42	66	10	65	146	90	0	75
4	42	68	-10	727	782	90	9	0
5	42	65	10	15	67	90	0	7
6	40	69	20	621	702	90	0	2
7	40	66	-10	170	225	90	5	0
8	38	68	20	255	324	90	0	10
9	38	70	10	534	605	90	0	4
10	35	66	-20	357	410	90	8	0
11	35	69	10	448	505	90	0	1
12	25	85	-20	652	721	90	18	0
13	22	75	30	30	92	90	0	17
14	22	85	-40	567	620	90	16	0
15	20	80	-10	384	429	90	19	0
16	20	85	40	475	528	90	0	14
17	18	75	-30	99	148	90	13	0
18	15	75	20	179	254	90	0	12
19	15	80	10	278	345	90	0	15
20	30	50	10	10	73	90	0	24

在 Li 数据集上经过多次实验和调优之后得到该算法的相关参数，兼具了算法收敛精度和求解速度。参数如下表 3-2 所示。

表 3-2 参数设置

参数	定义	取值
K	外循环重启次数	5
ω_1	全局最优解的得分	1
ω_2	局部最优解的得分	0.5
ω_3	接受不好解的得分	0.25
T	模拟退火初始温度	100
c	冷却系数	0.9
m_{thr}	相同删算子出现次数阈值	20
ε_1	新解被拒绝时的删除比例	0.04
ε_2	新解被接受时的删除比例	0.02

对比实验的结果以车辆数为目标进行比较，次要比较总行驶距离，采用 $\frac{d_{PLNS}-d_{Li}}{d_{PLNS}}$ 来表示距离差距，其中 d_{PLNS} 表示本文 PLNS 算法求得的最优路径总距离，d_{Li} 表示 Li 和 Lim 的算法求得的最优路径总距离。实验结果如表 3-3 所示。

表 3-3 实验一 LC 型数据的实验结果

Instances	Optimal Solution		Li&Lim		PLNS		
	车辆数	总距离	车辆数	总距离	车辆数	总距离	gap
lc101	10	828.94	10	828.94	10	828.94	0
lc102	10	828.94	10	828.94	10	828.94	0
lc103	9	1035.35	10	827.86	9	1035.35	0.2004
lc104	9	860.01	9	861.95	9	860.01	-0.0023
lc105	10	828.94	10	828.94	10	828.94	0
lc106	10	828.94	10	828.94	10	828.94	0
lc107	10	828.94	10	828.94	10	828.94	0
lc108	10	826.44	10	826.44	10	826.44	0
lc109	9	1000.6	10	827.82	9	1000.6	0.1727
lc201	3	591.56	3	591.56	3	591.56	0
lc202	3	591.56	3	591.56	3	591.56	0
lc203	3	591.17	3	585.56	3	591.17	0.0095
lc204	3	590.6	3	591.17	3	590.6	-0.0010
lc205	3	588.88	3	588.88	3	588.88	0
lc206	3	588.49	3	588.49	3	588.49	0
lc207	3	588.29	3	588.29	3	588.29	0
lc208	3	588.32	3	588.32	3	588.32	0

由表 3-3 可以得出，PLNS 算法所规划路径在所有规则分布的数据集中均达到计算所需车辆的最优值，实现了最优的路径规划求解效果。在与同类启发式算法的做对比的过程中，Li 和 Lim 的算法在有规则的地点分布数据集中整体上大部分

能求得最优结果，其他少部分中，在 lc103 数据集中本文设计的 PLNS 算法所得车辆需求数目为 9 辆，优于 Li 和 Lim 算法的 10 辆车。在 lc104 数据集中，PLNS 算法求得最优的总路程为 860.01，Li 和 Lim 算法求得所需总路程为 861.95，路程耗费降低了 0.23%。在 lc109 数据集中，PLNS 算法计算需要花费 9 辆车可完成任务，优于 Li 和 Lim 算法的 10 辆车。在 lc204 数据集上，PLNS 算法求得的总路程为 590.6 要低于 Li 和 Lim 的总路程 591.17。上述数据集中的 lc104 计算所得的最优路线如图 3-2 所示。

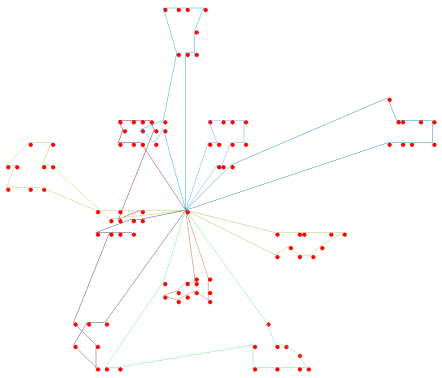

图 3-2 lc104 数据集路径规划展示图

如表 3-4 所示，本文设计的 PLNS 算法同样求得了随机地点分布下的车辆数目标的所有数据集最优解。在启发式算法中，与 Li 和 Lim 的模拟退火嵌套禁忌搜索算法作对比，在 lr109 数据集上，Li 和 Lim 算法计算所需总路程为 1239.96，PLNS 算法规划所需的最优行驶总距离为 1208.96，本算法节约了 2.56% 的路程。在 lr201 数据集上，Li 和 Lim 的算法需花费 1263.84 的总路程，PLNS 算法则为 1253.23 的总路程花销，优化了 0.85% 的路程。在数据集 lr209 上，Li 和 Lim 算法求得所需总路程为 937.05，PLNS 算法规划所需最优总路程为 930.59，路程节约了 0.69%。在数据集 lr211 上，Li 和 Lim 算法需行驶总距离 927.8，PLNS 算法则需要 911.52，路程节约了 1.79%。其中 lr201 规划的最优车辆行驶路线如图 3-3 所示。

表 3-5 为本文设计的 PLNS 算法在混合地点分布的数据集中与 Li 和 Lim 算法对比实验得出的结果。从表中可以得出，在混合地点分布下，PLNS 算法同样求得了所有数据集车辆数的最优解。在与 Li 和 Lim 的算法对比中，PLNS 算法在 lrc102、lrc106、lrc108、lrc201、lrc204、lrc206、lrc207 上均取得优于 Li 和 Lim 算法的结果。由此可以看出，本算法 PLNS 在混合分布点集中求解的效果好于 Li 和 Lim 的算法。在上述路数据集中，选取 lrc206 数据集作为路径规划，算法规划所得最优路径结果展示如图 3-4。

表 3-4 实验一 LR 型数据的实验结果

Instances	Optimal Solution		Li&Lim		PLNS		
	车辆数	总距离	车辆数	总距离	车辆数	总距离	gap
lr101	19	1650.8	19	1650.8	19	1650.8	0
lr102	17	1487.57	17	1487.57	17	1487.57	0
lr103	13	1292.68	13	1292.68	13	1292.68	0
lr104	9	1013.39	9	1013.39	9	1013.39	0
lr105	14	1377.11	14	1377.11	14	1377.11	0
lr106	12	1252.62	12	1252.62	12	1252.62	0
lr107	10	1111.31	10	1111.31	10	1111.31	0
lr108	9	968.97	9	968.97	9	968.97	0
lr109	11	1208.96	11	1239.96	11	1208.96	-0.0256
lr110	10	1159.35	10	1159.35	10	1159.35	0
lr111	10	1108.9	10	1108.9	10	1108.9	0
lr112	9	1003.77	9	1003.77	9	1003.77	0
lr201	4	1253.23	4	1263.84	4	1253.23	-0.0085
lr202	3	1197.67	3	1197.67	3	1197.67	0
lr203	3	949.4	3	949.4	3	949.4	0
lr204	2	849.05	2	849.05	2	849.05	0
lr205	3	1054.02	3	1054.02	3	1054.02	0
lr206	3	931.63	3	931.63	3	931.63	0
lr207	2	903.06	2	903.06	2	903.06	0
lr208	2	734.85	2	734.85	2	734.85	0
lr209	3	930.59	3	937.05	3	930.59	-0.0069
lr210	3	964.22	3	964.22	3	964.22	0
lr211	2	911.52	2	927.8	2	911.52	-0.0179

表 3-5 实验一 LRC 型数据的实验结果

Instances	Optimal Solution		Li&Lim		PLNS		
	车辆数	总距离	车辆数	总距离	车辆数	总距离	gap
lrc101	14	1708.8	14	1708.8	14	1708.8	0
lrc102	12	1558.07	13	1563.55	12	1558.07	-0.0035
lrc103	11	1258.74	11	1258.74	11	1258.74	0
lrc104	10	1128.4	10	1128.4	10	1128.4	0
lrc105	13	1637.62	13	1637.62	13	1637.62	0
lrc106	11	1424.73	11	1425.53	11	1424.73	-0.0006
lrc107	11	1230.14	11	1230.14	11	1230.14	0
lrc108	10	1147.43	10	1147.97	10	1147.43	-0.0005
lrc201	4	1406.94	4	1468.96	4	1406.94	-0.0441
lrc202	3	1374.27	3	1374.27	3	1374.27	0
lrc203	3	1089.07	3	1089.07	3	1089.07	0
lrc204	3	818.66	3	827.78	3	818.66	-0.0111
lrc205	4	1302.2	4	1302.2	4	1302.2	0
lrc206	3	1159.03	3	1162.91	3	1159.03	-0.0033
lrc207	3	1062.05	3	1424.6	3	1062.05	-0.3414
lrc208	3	852.76	3	852.76	3	852.76	0

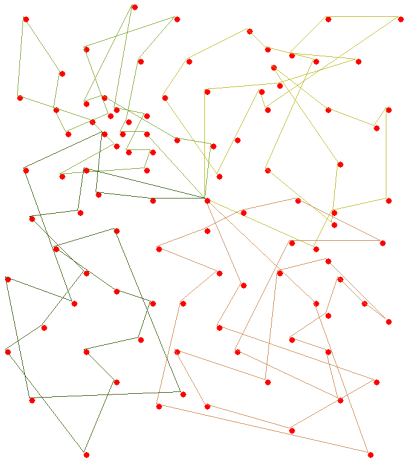

图 3-3 lr201 数据集路径规划展示图

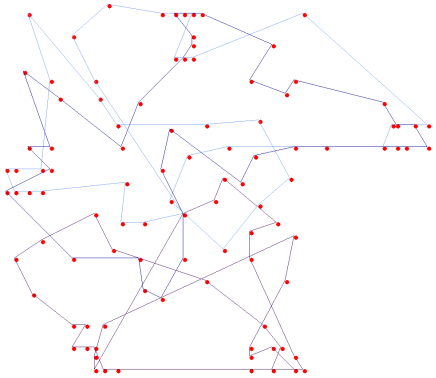

图 3-4 lrc206 数据集路径规划展示图

表 3-6 实验二 LC 型数据的实验结果

Instances	最优车辆数	Li&Lim		PLNS		
		车辆数	平均装载率	车辆数	平均装载率	gap
lc101	**10**	**10**	0.29	**10**	0.29	0
lc102	**10**	**10**	0.375	**10**	**0.385**	0.0260
lc103	**9**	10	0.295	**9**	**0.35**	0.1571
lc104	**9**	**9**	0.25	**9**	**0.322**	0.2236
lc105	**10**	**10**	0.34	**10**	**0.35**	0.0286
lc106	**10**	**10**	0.305	**10**	**0.31**	0.0161
lc107	**10**	**10**	0.34	**10**	**0.345**	0.0145
lc108	**10**	**10**	0.37	**10**	**0.39**	0.0513
lc109	**9**	10	0.28	**9**	**0.316**	0.1139
lc201	**3**	**3**	0.228	**3**	0.228	0
lc202	**3**	**3**	0.247	**3**	**0.257**	0.0389
lc203	**3**	**3**	0.209	**3**	**0.219**	0.0457
lc204	**3**	**3**	0.219	**3**	**0.247**	0.1134
lc205	**3**	**3**	0.176	**3**	**0.1809**	0.0271
lc206	**3**	**3**	0.219	**3**	**0.2238**	0.0214
lc207	**3**	**3**	0.18	**3**	**0.1809**	0.0050
lc208	**3**	**3**	0.18	**3**	**0.19**	0.0526

上述算法充分体现了 PLNS 算法在求解 PDPTW 问题的精度情况，本文设计的 PLNS 算法能求得该数据集下以车辆数为第一目标，总行驶距离为第二目标的问题的最优解，下列实验采用车辆数目为第一目标，平均装载率为第二目标检验 PLNS 算法在提高车辆平均装载率方面的效果。评价指标为 $\frac{d_{PLNS}-d_{Li}}{d_{PLNS}}$，实验同样与模拟退火嵌套禁忌搜索算法作对比。

如表 3-6 所示，实验以车辆数为第一目标，平均装载率为第二目标，在 lc103 数据下 PLNS 算法计算所需车辆为 9 辆，优于模拟退火嵌套禁忌搜索算法的 10 辆车且平均装载率提高了 15.71%，最高为 lc104 数据在使用 PLNS 算法后求得平均装载率提高了 22.36%，在 lc109 数据下，PLNS 算法能求得最优需求车辆数目为 9 辆，而对比算法则需要 10 辆车，同时本文 PLNS 算法所得装载率比模拟退火嵌套禁忌搜索算法那提高了 11.39%，PLNS 算法在 lc102、lc105、lc106、lc107、lc108、lc202、lc203、lc205、lc206、lc208 数据集下平均装载率提高了 1%-5%，lc207 数据在 PLNS 算法下平均装载率提升了 0.5%。由上述可得在规则分布的 LC 所有数据集下，本文 PLNS 算法所得方案的平均装载率均优于模拟退火嵌套禁忌搜索的算法。

如表 3-7 所示，在 LR 随机分布数据集下，PLNS 算法对所有数据所求规划路线方案的装载率均优于模拟退火嵌套禁忌搜索的方案，其中在 lr209 的数据下提高效果最好，达到 42.95% 的平均装载率提高。其他数据均有实现平均装载率 3.68% 到 42.95% 的优化程度。因此可以得出，本文设计的 PLNS 算法在随机分布的数据集下求解方案比模拟退火嵌套禁忌搜索的方案优，能达到更高的车辆平均装载率。在数据集中，lr1xx 的数据集为注重车辆数目优化的情况，lr2xx 的数据集为少车辆但注重长单条路径优化的情况。从表 3-7 可以得出，lr2xx 数据集出现多个优化效果超过 20% 的情况，由此可得本算法 PLNS 在少车辆长路径的数据集中能达到更好点的优化效果。

由表 3-8 可得，在混合型分布的数据集下，PLNS 算法所得路线的车辆平均装载率均高于模拟退火嵌套禁忌搜索算法，其中，在 lrc208 数据下 PLNS 算法平均装载率提高了 40.8%，提高程度最大。其他数据在 PLNS 算法下均使得车辆平均装载率从 1.29% 提高到 40.8%，在表 3-8 同样可以看出，在混合分布的数据集下，所需车辆比较多的数据能取得更大的装载率优化效果。

表 3-7 实验二 LR 型数据的实验结果

Instances	最优车辆数	Li&Lim		PLNS		
		车辆数	平均装载率	车辆数	平均装载率	gap
lr101	**19**	**19**	0.166	**19**	**0.173**	0.0405
lr102	**17**	**17**	0.182	**17**	**0.2047**	0.1109
lr103	**13**	**13**	0.2	**13**	**0.225**	0.1111
lr104	**9**	**9**	0.202	**9**	**0.2405**	0.1601
lr105	**14**	**14**	0.206	**14**	**0.215**	0.0419
lr106	**12**	**12**	0.228	**12**	**0.2475**	0.0788
lr107	**10**	**10**	0.209	**10**	**0.2235**	0.0649
lr108	**9**	**9**	0.2705	**9**	**0.295**	0.0831
lr109	**11**	**11**	0.209	**11**	**0.24**	0.1292
lr110	**10**	**10**	0.2125	**10**	**0.2585**	0.1779
lr111	**10**	**10**	0.21	**10**	**0.233**	0.0987
lr112	**9**	**9**	0.2485	**9**	**0.258**	0.0368
lr201	**4**	**4**	0.0918	**4**	**0.11925**	0.2302
lr202	**3**	**3**	0.10375	**3**	**0.13775**	0.2468
lr203	**3**	**3**	0.114	**3**	**0.1746**	0.3471
lr204	**2**	**2**	0.159	**2**	**0.1725**	0.0783
lr205	**3**	**3**	0.108	**3**	**0.156**	0.3077
lr206	**3**	**3**	0.12175	**3**	**0.191**	0.3626
lr207	**2**	**2**	0.128	**2**	**0.1556**	0.1774
lr208	**2**	**2**	0.145	**2**	**0.2155**	0.3271
lr209	**3**	**3**	0.085	**3**	**0.149**	0.4295
lr210	**3**	**3**	0.123	**3**	**0.166**	0.2590
lr211	**2**	**2**	0.112	**2**	**0.149**	0.2483

表 3-8 实验二 LRC 型数据的实验结果

Instances	最优车辆数	Li&Lim		PLNS		
		车辆数	平均装载率	车辆数	平均装载率	gap
lrc101	**14**	**14**	0.216	**14**	**0.22**	0.0182
lrc102	**12**	13	0.224	**12**	**0.244**	0.0820
lrc103	**11**	**11**	0.25	**11**	**0.265**	0.0566
lrc104	**10**	**10**	0.2615	**10**	**0.2945**	0.1121
lrc105	**13**	**13**	0.2369	**13**	**0.24**	0.0129
lrc106	**11**	**11**	0.218	**11**	**0.255**	0.1451
lrc107	**11**	**11**	0.28	**11**	**0.3**	0.0667
lrc108	**10**	**10**	0.26	**10**	**0.292**	0.1096
lrc201	**4**	**4**	0.113	**4**	**0.13325**	0.1520
lrc202	**3**	**3**	0.135	**3**	0.135	0
lrc203	**3**	**3**	0.136	**3**	**0.153**	0.1111
lrc204	**3**	**3**	0.135	**3**	**0.179**	0.2458
lrc205	**4**	**4**	0.093	**4**	**0.12375**	0.2485
lrc206	**3**	**3**	0.11925	**3**	**0.143**	0.1661
lrc207	**3**	**3**	0.10125	**3**	**0.1663**	0.3912
lrc208	**3**	**3**	0.1326	**3**	**0.224**	0.4080

3.4 带休息时间窗与白/夜班划分的路径规划算法

3.4.1 带休息时间窗约束处理策略

目前 PDPTW 问题考虑的约束为车辆容量约束、取送货匹配约束和供应商的开放时间窗约束。但实际场景中供应商给出的开放时间窗可能为一个较长时期，例如某日一整天或者连续几天均为开放时间窗。此时如果车辆在中午 12 点到 13 点（午休）期间或者晚上 1 点到 3 点（晚休）期间去供应商取货，则供应商可能因为休息而无法正常提供服务。因此在考虑供应商开放时间窗约束的同时引入供应商休息时间窗约束对于 PDPTW 问题应用于实际场景具有一定的现实意义。

当供应商开放时间窗跨越若干天时，中间必然存在无法取货的时间段，我们将其定义为供应商的休息时间窗。当到达供应商的车辆正好处于供应商休息时间窗时，则需等待至休息时间结束。根据车辆到达供应商的时间与供应商休息时间前后关系，我们分情况讨论了车辆可能的离开时间情况，内容如下：

（1）当休息时间 $[re,rl]$ 不在车辆的取/送货期间 $[t_i,l_i]$ 时，则车辆正常离开。其中，re 表示开始休息时间，rl 表示休息结束时间，t_i 表示车辆取/送该订单时所在点的开始时间，l_i 表示预计离开时间且 ST 表示该订单的服务时间，此时车辆的离开时间为

$$l_i = t_i + ST \tag{3-30}$$

（2）当休息时间窗与车辆在该订单地点的取/送货时间窗 $[t_i,t_i+ST]$ 相交时，此时，车辆的离开时间需考虑上供应商休息时间产生的影响，需把因休息而推迟的那部分任务继续完成，故离开时间为

$$l_i = t_i + ST - re + rl \tag{3-31}$$

（3）当车辆到达供应商的时间位于休息时间中间时，需等休息结束后供应商才正式开始给车辆装卸货，此时离开时间

$$l_i = rl + ST \tag{3-32}$$

3.4.2 白/夜班处理约束处理策略

在带时间窗取送货车辆路径规划问题中，当所得规划路径为多车辆长路程的路径时，每条路径需要花费若干天才可以完成该任务，那么可以认为该算法得出的路线安排并不具有实际执行意义，其背后是默认司机不管被安排多长的行驶时间和行驶距离下均可以不眠不休完成。基于上述情况，本课题提出车辆的白班和夜班的约束要求。在该约束下，需考虑司机的生理需求而进行白班与夜班安排，长

时间长距离的路线安排是不合理的。

针对该约束，本课题提出基于时间片聚类的策略，将订单按照各自所在白班或者夜班进行聚类，以订单记录的最早开始时间为划分对象，将订单聚类于各自日期和班次内。之后针对本班次内的订单运用改进大邻域搜索算法求得路线安排，由于所安排订单均为同一班次，故降低了跨周期运输的情况。求得班次内排班后再对所有班次的路线进一次大邻域搜索的联合优化，该过程由于订单本身的开放时间较长，因而可以灵活出现和被安排在多个班次中，通过对所有班次间进行大邻域搜索，最终可得满足现实需求的优解。

传统方案在规划得到最终路线后，并未考虑部分长路线现实行驶的不可行性。当路线行驶时间过长或者跨天时，所安排的路线则存在不合理的行驶情况。针对该种问题，本方案将车辆划分为白班和夜班，要求车辆回到仓库的时间要在该白/夜班的最晚时间之前。基于此约束，本课题将订单按照时间和所属的白班或夜班进行划分聚类，并针对各个白/夜班的订单集合做路径规划，最后将所得各路径通过大邻域搜索进行联合优化，算法各步骤展示如图 3-5。

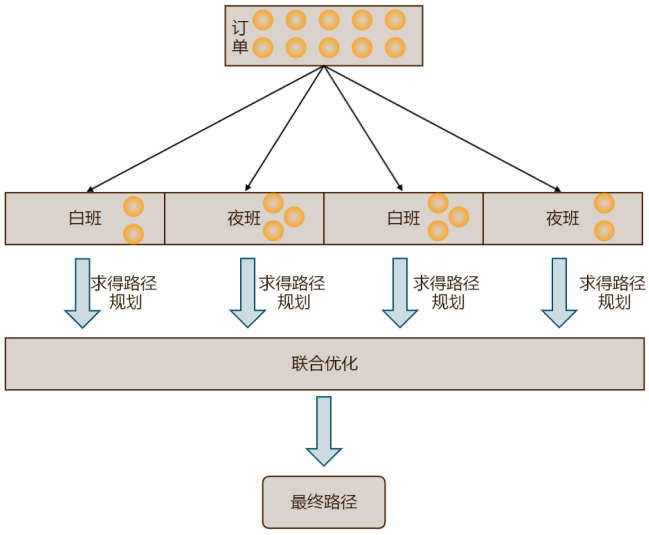

图 3-5 算法各步骤展示图

3.4.3 实验结果

本实验环境为 Intel（R) Core（TM）i7-6700，CPU@3.40GHz，8G RAM 内存，在 eclipse 环境，使用 Java 编写。

算法在不同数量的数据集运算得到对应所需车辆数目如图 3-6 所示。

由图 3-6 可得，随着数据量的增多所需的车辆数目也不断增多，但增长趋势相对稳定，因此经算法求得的解也是相对稳定。但现实场景中，企业往往按一段时间来租用车辆，此时企业关注的更多是车辆是否一直在不停的运行，并且车辆每

次运送货物时是否都尽可能地装满再出发。因此，平均装载率在此情况下具有更大的意义。

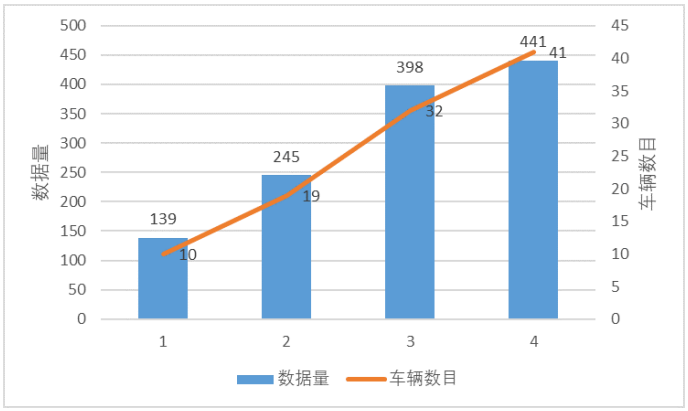

图 3-6 订单数目与所需车辆关系图

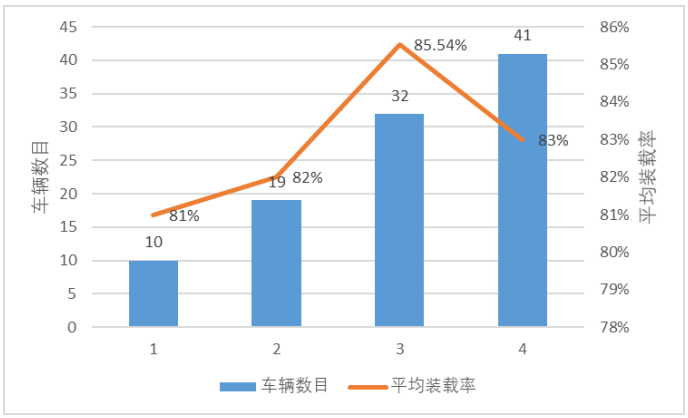

图 3-7 车辆数目与装载率关系图

本算法分别在不同大小数据集下运行得到各自的平均装载率结果如图 3-7，在不同数量数据下，本算法所安排路线的车辆平均装载率维持在 80% 左右，该情况也体现了本算法结果具有一定的稳定性。

如图 3-8 为本算法与遗传算法在现实企业数据下，所规划路径在平均装载率上的对比，由图可得，本算法相较于遗传算法，在平均装载率方面有明显提高。根据本算法原理，判断为本算法考虑多日期班次间的联合调优，相较于未联合调优的遗传算法，本算法应用了该策略后大大提高了车辆的装载率。

综上所述，本算法针对现实场景的休息时间窗与白/夜班约束提出了相应的解决方案，同时采用改进大邻域搜索结合嵌套模拟退火的算法以及算子的优化选择策略使得最终结果达到令人满意的效果，并且相比遗传算法效果有较好的提升。

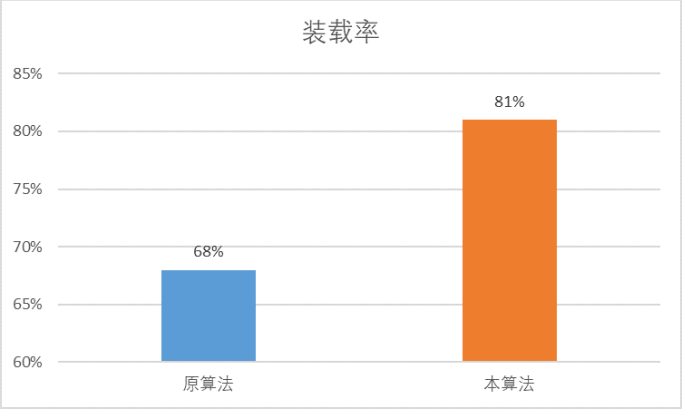

图 3-8 结果对比图

3.5 本章小结

本章研究了 Milk-Run 模式下的带时间窗取送货车辆路径规划问题，包括该问题的数学模型，对各个约束条件的数学公式化描述。然后针对该问题提出改进大规模领域搜索算法，算法采用毁坏重建原则，在满足约束的情况下能寻找更低车辆的安排方式，结合模拟退火的概率接受解策略来取得全局最优，算法外部增加 K 重启策略来促进结果的稳定，内部设计算子间自适应选择策略和算子内参数向量更新策略，来促进算法更快更深的搜索。本文设计的 PLNS 算法在 Li_100 的数据集下与 Li 和 Lim 算法作比较，以车辆数为第一目标，行驶总距离为第二目标的实验一结果显示，PLNS 算法能求得该数据集车辆目标的最优解，结果优于 Li 和 Lim 的模拟退火嵌套禁忌搜索算法。以平均装载率为第二目标的实验二结果显示，PLNS 算法在 Li_100 数据集下求出的路线装载率均优于模拟退火嵌套禁忌搜索的算法。

针对实际场景的特殊化需求，本课题创新性引入休息时间约束以及白夜班换班约束条件。现实中厂家提供的开放时间多数不止一天，这期间就存在午休、午夜等非开放时间窗的存在，因此引入了休息时间窗约束。本课题对车辆到达供应商的时间分段讨论，加入线性惩罚因子，以减少车辆在休息时间窗而等待的情况。针对白夜班换班需求，本课题采用基于时间聚类的方式，将订单按照每天的白/夜班情况进行聚类，再针对各自班次的订单进行路径规划求解，最后把各班次安排的路线进行联合调优，即可得到最终路线安排，将本算法与遗传算法作对比，结果显示本算法能取得更高的装载率。

第 4 章 带时间窗的取送货车辆路径规划系统实现

4.1 系统概述及架构

随着近几年科技的迅猛发展，与计算机相关的技术发展尤其突出，包括硬件方面在运算能力和存储能力的逐年提升，通信方面随着 5G 通信技术的推出给车辆零误差导航和智能驾驶带来了无限可能。这些均给人们生活带来了更优品质的体验，同时也为更快速、精确的车辆路径规划提供了坚实的技术基础。而智能设备用户数量也是逐年上升，使用出行软件的比例也在增加，路径规划的需求也越来越频繁，用户在出行和导航越来越依赖地图软件。国内出行地图软件以百度地图、高德地图为主，国外则是谷歌地图。这里面支撑用户在线浏览电子地图的技术是名为 Web-GIS（网络地理信息系统）的技术，该技术通过浏览器对地理位置信息进行显示和管理，是一种基于浏览器/服务器（B/S）架构的技术。用户通过浏览器进行地理信息的获取和其他处理操作，浏览器将对应操作发送至地图服务器，地图服务器执行操作后快速返回至用户终端。解决了客户端/服务器（C/S）架构下只能在电脑访问和处理地理信息以及不便于推广的缺点。用户在进行路径规划过程中能在浏览器上实时观察位置变化，这在物流快速发展的今天具有重要的作用，即用户可以实时查看快递包裹位置。在此期间地图还可以通过不同图层展示不同信息。本课题采用百度地图对规划路径进行显示，并在地图上展示了各个供应商和厂商的相关信息以及规划的路径装载率、最早开始时间与最晚开始时间等信息。

本系统为基于百度地图的 Web-GIS 路径规划系统，用于解决制造业中供应商的带时间窗取送货车辆路径规划问题，采用本文设计的改进大邻域搜索算法应用于某企业的零部件订单运输。系统包括三个部分：客户 C 端、企业终端、司机移动终端。客户 C 端将订单信息以表单的形式发送至企业端，企业端将所得的订单存入数据库中并调用服务器对订单进行路径规划，之后企业端通知客户 C 端订单已经被规划完成，然后企业将所得规划的路径信息发送至司机移动终端，司机在移动终端查看运送货物的路径、各个地点的时间以及取货或者送货的零部件类别并完成运输过程。企业端可以在浏览器实时查看车辆运输的整个过程。企业端系统执行流程为：系统在每天固定时间访问后台获得当前需要安排的零部件订单，后台服务器将新增订单信息、供应商位置和时间窗信息、车辆信息等数据代入算法中执行车辆路径规划算法并获得经过优化计算得出的最优路径安排，将所得路径发送至司机终端和浏览器端，并在百度地图上显示各个车辆的路径规划情况，企

业 C 端的浏览器上展示当前车辆的装载率和车辆的开始与结束时间等信息以及各个车辆在各个时间点所处的状态（取货、送货）的甘特图信息。由于某企业安全要求，而地图上包含了该企业和所有供应商的自身信息以及行程信息，本章只展示车辆装载率信息和甘特图信息，车辆在地图上显示的与各个供应商间的路径规划展示图不进行显示。

4.2 系统的流程逻辑

物流的配送广泛存在于制造业中，其运输费用占据企业供应链管理成本的一大部分。物流管理系统通过管理供应商和厂商间的订单信息，对订单进行合理分配，并对车辆进行高效有序的安排，使供应链各流程有条不紊的进行，在生产制造中具有重要作用。因此本课题以物流配送为核心，对供应商和厂商的订单进行有效管理并合理安排车辆进行配送。基于场景现实约束条件，设计带时间窗取送货的车辆路径规划系统，采用大邻域搜索算法进行车辆和订单的合理安排，以达到车辆数目的最少化和装载率的最大化效果。

本带时间窗车辆路径规划系统对于配送中心的管理功能主要分为三个部分：数据的采集与预处理部分、路径规划求解部分、配送信息部分。数据的采集与预处理部分是订单信息管理和上传至服务器并进行预处理的过程，对于不符合现实情况的订单进行标记和归类，整理出满足要求订单进入下一个部分。路径规划求解部分是后台服务器在获得有效订单后结合车辆信息、供应商与厂商信息构建车辆带时间窗取送货的车辆路径规划模型，并应用本课题研究的改进大邻域搜索算法进行车辆路径的最优安排，实现订单与车辆的合理配送。配送信息部分是将车辆规划的方案传达给司机并实时更新司机位置，在浏览器端进行实时显示。

系统的整体执行流程为：每日固定时刻企业工作人员登录进系统，在浏览器端查看当天需要完成的订单信息，然后执行订单预处理过程，得到正常订单和异常订单，在异常订单备注字段记录下异常原因。之后判断正常订单是否存在新增供应商信息，若存在则依据百度地图计算新的地点与其他地点间的距离同时对关联矩阵进行更新。路径规划模型所需的信息都准备充分后执行路径规划算法，生成尽可能最优的配送路径，并在浏览器端调用百度地图进行显示。当司机通过移动设备要获取行驶路径时，对后台服务器发送请求，后台服务器将规划的配送路径返回至司机终端。浏览器实时获取和更新司机的配送过程。

4.3 系统的功能模块

基于零部件在供应商与厂商间的运输情况，本带时间窗取送货车辆路径规划

系统分为两个功能模块，分别是 PC 端的功能模块以及所有数据的处理模块。各个模块相应的子模块如图 4-1 所示。

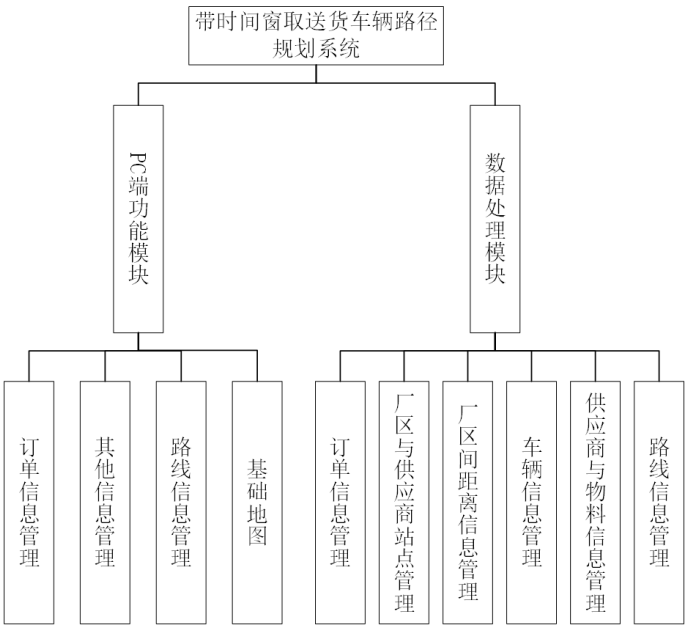

图 4-1 系统功能模块图

本系统根据不同性能需求设计有两大模块，PC 端模块主要为物流中心的工作人员对系统平台进行管理，查看订单数据、已求得的路线信息以及包含距离矩阵和供应商等的其他信息，能对行驶中司机的位置和行驶路线在百度地图上实时显示。数据处理模块包含了该系统的所需的各类基本信息，存储于数据服务器中，同时依据服务器的强大计算能力，执行订单的路径规划运算，并把计算结果在 PC 端进行展示。本节将对各个模块进行详细介绍。

4.3.1 PC 端功能模块

PC 端为物流中心工作人员提供一个浏览器监测整个配送过程的方式。在 PC 端模块包含对数据的基本操作，例如订单数据、供应商数据、车辆数据的增加、删除与导出。包含规划路线的信息管理例如路线上车辆各个时间点状态的甘特图显示以及各个路线的装载率信息和出发以及返回的时间显示，还包含在百度地图的上展示所有车辆规划的路线以及实时显示车辆的行驶位置的功能。综上所述，PC 端功能模块包含四个部分：订单信息管理模块、其他信息管理模块、规划路线信息模块、基础地图展示模块。

（1）订单信息管理模块　　该模块展示有订单的详细信息，数据格式兼具 Li 标准数据集的格式，包含配送的地点、时间窗、物料量、取货点和送货点等信息，该模块具有订单的增加、删除、更新、查询以及导出功能。

（2）其他信息管理模块　　该模块包含厂区与供应商站点的关联信息、车辆信息、供应商与物料关联信息以及厂区间距离矩阵信息。是订单信息的补充。其中，算法从订单中获得供应商/厂商的取货点和送货点 ID，之后可以通过距离矩阵查询该点到其他供应商的距离信息，也可以在供应与厂区信息中获得该供应商的详细信息例如地理经纬度或者供应商详细地址。通过该模块，物流中心的工作人员可以对非订单信息进行增加、删除、查询和更新管理。

（3）规划路线信息模块　　该模块包含配送路径规划、距离矩阵更新、规划路线指标展示三个功能。在该配送路径规划功能下，物流中心工作人员执行路径规划计算，系统调用服务器对所有正常订单使用改进大邻域搜索算法计算路径规划，并将得到的所有车辆路径返回至浏览器端。距离矩阵更新功能则是若正常订单中出现未出现的供应商地点时，则调用百度地图求解该地点与其他供应商点的距离，并更新距离矩阵。规划路线指标展示，该功能下对获得规划路径计算车辆装载量，使用 Echarts 在浏览器绘制每条规划路径的装载率数据图如图 4-2 所示。还提供了规划车辆在行驶过程中各个时间点状态的甘特图，使用 JavaScript 绘制甘特图的各个部分并填充相应的内容和颜色，如图 4-3 所示，将车辆的状态分为发车、装货、在途、卸货、休息状态。

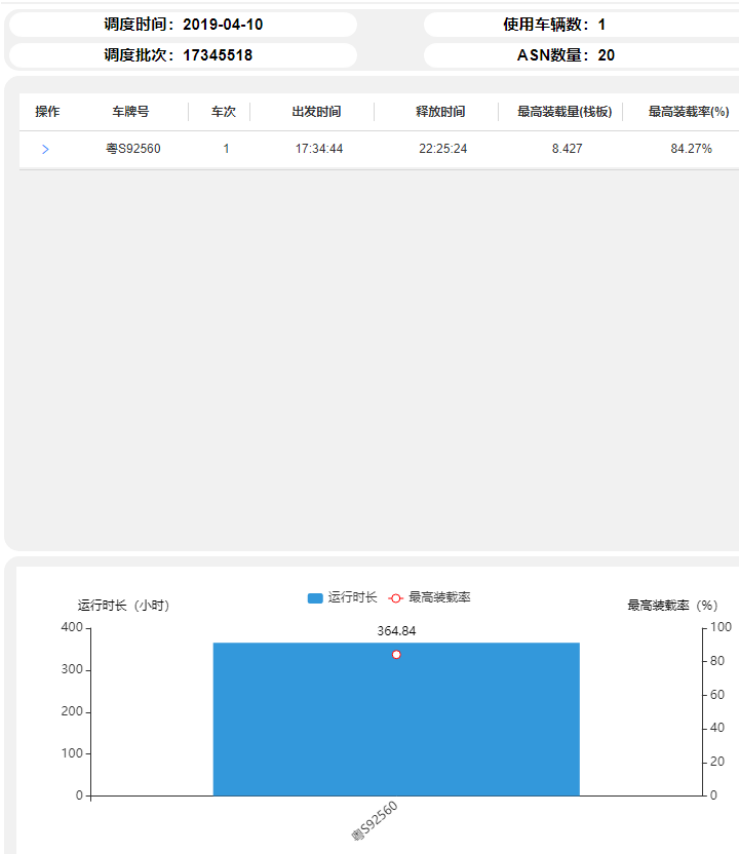

图 4-2 所得排班车辆的指标展示图

图 4-3 各车辆路径规划排班甘特图

（4）基础地图模块　　本模块基于百度地图接口在浏览器端实现相应的功能。采用 JavaScript 技术在浏览器端调用百度地图基于 Web 服务的 API。依据百度地图接口实现鼠标滑轮滚动的放大、缩小功能以及拖拽功能，实现地理位置的检索并定位功能、实现地理位置的标注功能。本系统在该模块地图上标注了厂商位置和名称、所有供应商的位置和名称以及规划车辆的路线信息与实时更新的车辆行驶图标。在该模块下将服务器计算所得的各个车辆的规划路线通过百度地图按照配送顺序进行路径绘制。由于地图路径规划依据某企业真实数据，规划路径在地图上展示有某企业的各个供应商信息、地理位置以及企业本身的相关信息，因此基于保密原因该路线规划地图无法进行展示。可以在第三章实验部分看到算法在数据集中计算所得规划路径的展示图。

4.3.2 数据处理功能模块

本路径规划系统采用 SSM 框架，使用 Postgre 数据库，依据路径规划算法和公开数据集 Li 数据集的格式设计有六个数据信息表，分别是订单信息管理表、厂区与供应商站点管理表、厂区间距离信息管理表、车辆信息管理表、供应商与物料信息管理表、路线信息管理表。由于供应商和厂商的位置相对固定，因此厂区间距离信息管理表的设计，使得算法在预加载后地理位置信息后可以在 $O(1)$ 时间获得两地距离信息。本系统各数据表功能和信息如下所示。

（1）订单信息管理表

本表包含 Li 数据集字段，例如：最早取货时间、最晚取货时间、服务时间、装

货点位置、卸货位置、物流量，以及唯一键的订单号 ASN_NO。取货和送货时间按照日期格式存储，但是在算法处理的时候会预先将日期转换成毫秒的整型数值，以方便算法日期的表达形式和计算方式进行统一简化，计算后的结果也可以方便的转换回各种格式的时间。表中的取货点和卸货点采用站点 ID 来表示，通过站点 ID 可以关联到厂区与供应商站点管理表，即在该表中可以获取站点的详细信息包括经纬度和住址，并且通过厂区间距离信息管理表来获取各个站点间的距离值和时间值。为提高系统的鲁棒性，本订单信息管理表增加 data_status 字段用于区分该订单是异常订单还是正常订单，同时把预处理得出的订单异常原因写入 remark 字段中。

（2）厂区与供应商站点管理表

该表记录供应商的信息以及与该供应商相关的订单 ID 编号。用户可以通过站点（包含厂区站点和供应商站点）ID 获得该表详细信息，该表包含供应商的开放时间窗，即标准装货时间和标准卸货时间，也可以获得该供应商的休息时间窗，即开始不可用的时间和结束不可用时间。还能获得该供应商的经纬度信息和详细的地址。该表是记录供应商约束的主要信息表。

（3）厂区间距离信息管理表

该表格记录各个站点（包含厂家和供应商）之间的距离，可通过唯一键站点 ID 来查询得到。当算法对订单预处理出现新地点时会更新该表格信息。该表包含两个地点间的距离信息和时间信息，通过这两个维度的信息可以为算法提供时空临近情况，该表信息在算法初始化阶段会记录在 hashmap 中，形成距离矩阵后算法即可在 $O(1)$ 时间下快速取出距离和时间值。

（4）车辆信息管理表

该表格记录车辆的详细信息，是体现车辆约束的表，包含车的容量、车型，增加了车的有效性字段，用来判断该车是否正在被安排任务，还包含车牌号、司机姓名、司机电话等信息。在系统中，算法通过车容量进行车辆路径规划安排和装载率计算。车型字段则是为以后进一步研究多车型下的车辆路径规划问题提供支持。

（5）供应商与物料信息管理表

该表格记录每一个物料所在的位置信息，包括物料编码、所在位置 id 以及物料数量。通过该表可以知道当前物料是运送到哪个供应商的，再通过 site_id 即可获得对应供应商的相信地址信息、经纬度信息以及开放时间窗信息等。

（6）路线信息管理表

该表格为标准输出表，包含订单 ID、车辆 ID、到达站点、物流量、发车时间、发车站点等信息，算法经过上述数据的输入和处理后得到最终的各个车辆的配送

计划。记录了车辆从车场出发，在哪些时间到达了哪个站点并装货/卸货了哪些物品的整个过程。最终在地图上显示车辆行驶路线，并且该规划路线会发送至司机终端，司机按照路线进行货物的配送。

4.4 系统设计要点

本系统采用 B/S 架构，前端使用 HTML、CSS、jQuery 技术绘制界面，采用 Echarts 绘制数据直方图，使用 JavaScript 和 canvas 绘制甘特图。在 PC 端功能模块对订单和其他信息的展示均具有显示、查询、导出、删除等功能，后台基于 java 语言采用 SSM 框架，使用 mybatis 将数据持久化到服务器数据库中。数据库采用高稳定性的 Postgre 数据库，并依据带时间窗取送货车辆路径规划问题设计相应数据库表结构。具体技术介绍如下。

4.4.1 SSM 框架

本系统采用 SSM（Spring、SpringMVC、MyBatis）框架，利用 Spring 的控制反转特性将对象之间的依赖关系交给了 Spring 控制，方便解耦。SpringMVC 是使用了 MVC 设计思想的轻量级 web 框架，对 web 层进行解耦，将流程控制代码放到 Controller 层处理，将业务逻辑代码放到 Service 层处理，使系统开发更简洁。Mybatis 是一个基于 Java 的持久层框架，可以直接在 xml 中通过 SQL 语句操作数据库，支持自己定制 SQL、存储过程以及高级映射，相对 hibernate 等 orm 框架来说更加直观，特别在业务场景比较复杂、sql 好多联合关联的情况下使用 Mybatis 能有效的操作数据库。

4.4.2 JQuery 插件

本系统前端的界面效果、图表绘制以及后台与前端的交互形式，大量使用了 JavaScript 技术。而 jQuery 则是 JavaScript 的一个函数库，是基于 JavaScript 语言写出来的一个框架，使用 jQuery 能极大简化前端 JavaScript 的编写。浏览器端使用 Ajax（异步 JavaScript 和 XML）与后台进行通信，使得可以在不刷新整个页面的情况局部更新数据, 通信格式为 json 格式数据。浏览器端的甘特图通过 JavaScript 根据车辆运行时间绘制各辆车的甘特图并将获得的车辆状态信息填充入图中。装载率图的绘制采用百度的开源可视化库 Echarts。

4.4.3 百度地图 API

百度地图为各个平台包括浏览器端、安卓端、iOS 端的开发者分别提供了一套 API 用于提供地图功能。本系统采用百度地图浏览器端基于 JavaScript 版本的 API。浏览器端地图针对用户的鼠标和输入信息直接调用百度地图 API 将信息请

求发送至百度地图服务器，然后获得返回结果并在本地浏览器端展示。路径规划信息则需要浏览器与企业服务器间进行通信，并通过 AJAX 获得服务器返回的路径规划计算结果并在百度地图上采用驾车路线规划来一段一段绘制车辆路线。在此过程中，通过创建 Map 基础类，使用 centerAndZoom 初始化厂商位置，设置 enableScrollWheelZoom 来开启鼠标滚轮缩放，使用 enableContinuousZoom 启动地图拖拽功能。通过 DrivingRoute 类创建一个驾车路线的实例，对从服务器获得的所有车辆路线，按照行驶路线先后顺序进行地图的各段绘制，最后得到所有车辆访问点的地图展示图。

4.5 应用价值

随着全球迈向数字化和智能化时代，智能制造对制造业的影响越来越大，通过将新一代信息技术与制造技术相结合的方式，达到缩短产品研制周期、降低资源能源消耗、降低运营成本、提高生产效率、提升产品质量的效果。本文以现实工业制造中的零部件管理为背景，结合企业生产制造各个环节的环境信息、流程信息、零部件信息，以物流管理智能化为主要方向，优化供应链中生产制造原材料的物流配送，以达到物流的智能优化配送，降低生产制造中的物流成本。针对某企业现实场景的配送问题，本课题完成带时间窗取送货车辆路径规划系统。相对于原系统基于遗传算法的车辆路径规划方案，本课题研究的改进大邻域搜索方案将车辆平均装载率提高了 13%，有效的降低了车辆的运输成本，优化车辆的合理配置。

本系统可应用于包括制造业零部件运输、快递行业物品的运输、冷链物流，为企业和物流公司提供合理的物流配送安排。

4.6 本章小结

本章介绍了带时间窗取送货车辆路径规划系统。讲述了该系统的概述和架构以及系统运行的整体流程逻辑，之后介绍了系统的两大功能模块包括 PC 端功能模块和数据处理模块，并对相应模块的子功能进行了详细描述。最后介绍了本章使用到的技术包括后端采用的 SSM 框架以及前端使用的 jQuery 技术以及百度地图 API。

系统接收的数据为实际生产数据，执行过程中将实际生产订单数据进行预处理，然后使用改进大邻域搜索的车辆路径规划算法求得最终的车辆安排，最后显示在前端界面中，系统的前端可查看各个数据表信息，并显示有最终车辆排班图以及车辆在各个时间段（装货、卸货、在途等）状态的甘特图。实现了对物流过程中车辆的安排和管理，体现了本论文带时间窗取送货车辆路径规划系统的实用性。

结　论

带时间窗的车辆路径规划问题是个 NP-hard 问题，但在实际场景中经常出现，具有极大的现实意义，本文在阅读了国内外的带时间窗取送货车辆路径规划问题的研究成果，提出了改进大邻域搜索算法求解该问题，之后依据现实场景需求，加入休息时间约束、白/夜班约束等条件，构建更符合现实场景的 PDPTW 算法，并在此技术上设计了带时间窗取送货的车辆路径规划系统，主要成果如下：

（1）提出改进大邻域搜索算法。算法外部引入 K-Restarts 策略使算法结果更的稳定，算法内部提出操作算子间自适应策略，在算子内引入参数向量自适应策略。在以车辆数目为第一目标的情况下，依据实验一数据显示，本算法在 Li_100 数据集下计算所需车辆数与该数据集最优安排车辆数一致。在以平均装载率为第二目标的实验二数据显示，本算法所求路径的平均装载率均优于模拟退火嵌套禁忌搜索的算法；依据实际情况，算法增加了对休息时间窗和白/夜班换班约束的处理。针对休息时间窗约束，采用分段讨论并引入线性惩罚函数的方式。对白/夜换班情况，按照每天白/夜班对订单聚类，然后求解班次内路径安排，之后联合调优各班次路线，最终得到满足条件的解。

（2）本系统基于 SSM 框架，使用 HTML、CSS、Javascript，Echarts 等技术，将改进大邻域搜索算法应用到系统中。包含两大功能模块，PC 端功能模块和数据处理模块。能完成订单信息的管理核对所安排的订单依据算法求出合理的路线规划方案。

本课题关于带时间窗取送货的车辆路径规划问题，研究内容从数学模型、算法方案、实际拓展需求、系统设计等方面进行，该过程中设计完成了本算法和其他算法的对比实验，设计完成了车辆路径规划系统，未来将有以下几个方面的改进：

（1）将本算法推广至更一般的情况，即带时间窗的同时取送货车辆路径规划问题，寻求更一般化和易拓展的解决方案。

（2）拓展量子优化算法在该领域的应用，未来的路径规划问题将有希望通过量子计算机来解决，而各个约束条件下如 VRP、CVRP、VRPTW、PDDPTW 等领域问题，能否很好的转换到量子优化算法里，并用量子计算机来解决，这是一个很有意义的改进方向。

（3）将本系统应用于实际场景，以此来测试和完善系统功能，提高系统鲁棒性，并且需要持续不断的对算法进行研究和优化，以达到更好的效果。

参考文献

[1] Donald J, Bowersox C, Dvaid J, et al. Supply Chain Logistics Management[M]. New York : McGraw-Hill Education, 2012.

[2] Chopra S Meindl P. Supply Chain Management–Strategy, Planning and Operation[M]. 北京 : 清华大学出版社, 2001 : 22-36.

[3] Venkateshan P, Mathur K. An Efficient Column-Generation-Based Algorithm for Solving a Pickup-and-Delivery Problem[J]. Computers & Operations Research, 2011, 38(12) : 1647-1655.

[4] Cordeau J-F. A Branch-and-Cut Algorithm for the Dial-a-Ride Problem[J]. Operations Research, 2006, 54(3) : 573-586.

[5] Hernández-Pérez H, Salazar-González J-J. The One-commodity Pickup-and-DeliveryTraveling Salesman Problem:Inequalities and Algorithms[J]. Networks, 2007, 50(4) : 258-272.

[6] Ropke S, Cordeau J-F. Branch and Cut and Price for the Pickup and Delivery Problem with Time Windows[J]. Transportation Science, 2009, 43(3) : 267-286.

[7] Bettinelli A, Ceselli A, Righini G. A Branch-and-Price Algorithm for the Multi-Depot Heterogeneous-Fleet Pickup and Delivery Problem with Soft Time Windows[J]. Mathematical Programming Computation, 2014, 6(2) : 171-197.

[8] Haddad M N, Martinelli R, Vidal T, et al. Large Neighborhood-Based Metaheuristic and Branch-and-Price for the Pickup and Delivery Problem with Split Loads[J]. European Journal of Operational Research, 2018, 270(3) : 1014-1027.

[9] Bent R, Van Hentenryck P. A Two-Stage Hybrid Algorithm for Pickup and Delivery Vehicle Routing Problems with Time Windows[J]. Computers & Operations Research, 2006, 33(4) : 875-893.

[10] Clarke G, Wright J W. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points[J]. Operations research, 1964, 12(4) : 568-581.

[11] Tavakkoli-Moghaddam R, Safaei N, Gholipour Y. A Hybrid Simulated Annealing for Capacitated Vehicle Routing Problems with the Independent Route Length[J]. Applied Mathematics and Computation, 2006, 176(2) : 445-454.

[12] Gillett B E, Miller L R. A Heuristic Algorithm for the Vehicle-Dispatch Problem[J]. Operations Research, 1974, 22(2) : 340-349.

[13] Jung S, Haghani A. Genetic Algorithm for a Pickup and Delivery Problem with Time Windows[J]. Transportation Research Record, 2000, 1733(1) : 1-7.

[14] Landrieu A, Mati Y, Binder Z. A Tabu Search Heuristic for the Single Vehicle Pickup and Delivery Problem with Time Windows[J]. Journal of Intelligent Manufacturing, 2001, 12(5-6) : 497-508.

[15] Huang Y-H, Ting C-K. Ant Colony Optimization for the Single Vehicle Pickup and Delivery Problem with Time Window[C] // Proceedings of 2010 International Conference on Technologies and Applications of Artificial Intelligence. Hsinchu, Taiwan, 2010 : 537-543.

[16] Sombuntham P, Kachitvichayanukul V. A Particle Swarm Optimization Algorithm for Multi-Depot Vehicle Routing Problem with Pickup and Delivery Requests[C] // Proceedings of the International MultiConference of Engineers and Computer Scientists 2010. London, UK, 2010 : 1998-2003.

[17] Shi L, Gu H-Y, Xi Y-G. Quick LNS Algorithm for Solving PDPTW Problem[J]. Control Engineering of China, 2007, 14(5) : 558-561.

[18] Li H, Lim A. A Metaheuristic for the Pickup and Delivery Problem with Time Windows[J]. International Journal on Artificial Intelligence Tools, 2003, 12(2) : 173-186.

[19] Alaia E B, Harbaoui I, Borne P, et al. A Comparative Study of the PSO and GA for the m-MDPDPTW[J]. International Journal of Computers Communications & Control, 2018, 13(1) : 8-23.

[20] Créput J-C, Koukam A, Kozlak J, et al. An Evolutionary Approach to Pickup and Delivery Problem with Time Windows[C] // Proceedings of International Conference on Computational Science. Berlin, Heidelberg, 2004 : 1102-1108.

[21] Velasco N, Castagliola P, Dejax P, et al. A Memetic Algorithm for a Pick-up and Delivery Problem by Helicopter[M] // Bio-inspired Algorithms for the Vehicle Routing Problem. Berlin, Heidelberg : Springer, 2009 : 173-190.

[22] Cherkesly M, Desaulniers G, Laporte G. A Population-Based Metaheuristic for the Pickup and Delivery Problem with Time Windows and Lifo Loading[J]. Computers & Operations Research, 2015(62) : 23-35.

[23] 王超, 刘超, 穆东, et al. 基于离散布谷鸟算法求解带时间窗和同时取送货的车辆路径问题[J]. 计算机集成制造系统, 2018, 24(3) : 570-582.

[24] Solomon M M. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints[J]. Operations Research, 1987, 35(2) : 254-265.

[25] Wilson N H M, Colvin N J. Computer Control of the Rochester Dial-a-Ride System[M]. Technical Report R-77-31, Department of Civil Engineering, Massachusetts Institute of Technology, Cambridge : Massachusetts Institute of Technology, Center for Transportation Studies, 1977.

[26] Wilson N H M, Weissberg R W, Hauser J. Advanced Dial-a-Ride Algorithms Research Project[R]. 1976.

[27] Nanry W P, Barnes J W. Solving the Pickup and Delivery Problem with Time Windows using Reactive Tabu Search[J]. Transportation Research Part B: Methodological, 2000, 34(2) : 107-121.

[28] Ropke S, Pisinger D. An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows[J]. Transportation Science, 2006, 40(4) : 455-472.

[29] Hezer S, Kara Y. Solving Vehicle Routing Problem with Simultaneous Delivery and Pick-up using Bacterial Foraging Optimization Algorithm[C] // Proceedings of the 41st International Conference on Computers & Industrial Engineering. Los Angeles, 2011 : 380-385.

[30] Nalepa J, Bocho M. A Parallel Algorithm with the Search Space Partition for the Pickup and Delivery with Time Windows[C] // Proceedings of 2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC). Krakow, Poland, 2015 : 92-99.

[31] Nalepa J, Blocho M. A Parallel Memetic Algorithm for the Pickup and Delivery Problem with Time Windows[C] // Proceedings of 2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP). St Petersburg, Russia, 2017 : 1-8.

[32] Ghilas V, Demir E, Van Woensel T. An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows and Scheduled Lines[J]. Computers & Operations Research, 2016 (72) : 12-30.

[33] Al Chami Z, Manier H, Manier M-A, et al. An Advanced GRASP-HGA Combination to Solve a Multi-Period Pickup and Delivery Problem[J]. Expert Systems with Applications, 2018(105) : 262-272.

[34] Peng Z, Al Chami Z, Manier H, et al. A Particle Swarm Optimization for Selective Pickup and Delivery Problem[C] // Proceedings of 2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI). Volos, Greece, 2018 : 947-952.

[35] Bertsimas D, Chang A, Mišić V V, et al. The Airlift Planning Problem[J]. Transportation Science, 2019, 53(3) : 773-795.

[36] 贾永基, 谷寒雨, 席裕庚. 求解 PDPTW 问题的一种快速禁忌搜索算法[J]. 控制与决策, 2004, 19(1) : 57-60.

[37] 潘立军, 符卓. 求解带时间窗取送货问题的遗传算法[J]. 系统工程理论与实践, 2012, 32(1) : 120-126.

[38] Wang C, Mu D, Zhao F, et al. A Parallel Simulated Annealing Method for the Vehicle Routing Problem with Simultaneous Pickup–Delivery and Time Windows[J]. Computers & Industrial Engineering, 2015(83) : 111-122.

[39] 黄务兰, 张涛. 基于改进全局人工鱼群算法的 VRPSPDTW 研究[J]. 计算机工程与应用, 2016, 52(21) : 21-29.

[40] 王旭坪, 李新宇, 张珺. 考虑时空距离的异车型同时集送车辆路径优化[J]. 管理学报, 2018, 15(6) : 918.

[41] 张庆华, 吴光谱. VRPSPDTW 建模及模因求解算法[J]. 计算机应用, 2019 : 1-9.

[42] Nagata Y, Bräysy O. A Powerful Route Minimization Heuristic for the Vehicle Routing Problem with Time Windows[J]. Operations Research Letters, 2009, 37(5) : 333-338.

[43] Rachman A, Dhini A, Mustafa N. Vehicle Routing Problems with Differential Evolution Algorithm to Minimize Cost[C] // Proceedings of the 20th National Conference of Australian Society for Operations Research. Australian Society for Operations Research(ASOR) , 2009 : 78-91.

[44] Kirkpatrick S, Gelatt C D, Vecchi M P. Optimization by Simulated Annealing[J]. Science, 1983, 220(4598) : 671-680.

[45] Shaw P. Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems[C] // Proceedings of International Conference on Principles and Practice of Constraint Programming. Berlin, Heidelberg, 1998 : 417-431.

[46] Pisinger D, Ropke S. A General Heuristic for Vehicle Routing Problems[J]. Computers & Operations Research, 2007, 34(8) : 2403-2435.

致　谢

时间过得飞快，匆匆两年半就快要结束了，在读研的这段日子里我学到了很多也收获了很多，曾在一开始迷茫过，也在后来明确方向努力学习和奋斗。

在这期间非常感谢导师黄荷姣老师的教导，课题上黄老师每礼拜例会的认真指导让我对课题不断了解和深入探索，让我学会做研究的一个过程是怎样的，在平时，黄老师和我们耐心细致的交流，让我们知晓所做工作的不完备之处以及提高的方法，也让我们学到了很多为人处世的道理，非常感谢黄老师的教导。同时，也非常感谢贾小华老师对我研究课题的方向给出的指导性意见，贾老师的严谨治学和广博的知识让我非常钦佩。感谢堵宏伟老师以及花忠云老师在我课题的开题和中期给予的指导建议。非常感谢实验室老师在我研究生期间的指导和帮助让我收获很多。

感谢实验室小伙伴在研究生期间的相处和帮助。大家一起举办了实验室第一次元旦晚会，一起努力拿到了羽毛球团体季军，以及组团的聚餐。感谢方波每周固定的叫上我去打羽毛球，让我掌握了另一项运动同时还锻炼了身体，感谢蔡航以及实验室其他的同学在平时课程和课题中的互相讨论和互相进步。还有感谢舍友高昕在研究生的这几年一起组团讨论学习和游戏，带给了我一个欢快而充实的研究生生活。

最后也是最重要的，感谢我的父母。感谢他们每周时不时的了解我的情况，在低谷的时候不断支持我，在开心的时候分享我的喜悦，，他们的鼓励是促进我不断学习的动力，也是支撑我不断走出低谷的力量。没有他们的支持就没有现在的我。