西南财经大学

SOUTHWESTERN UNIVERSITY OF FINANCE AND ECONOMICS

硕士学位论文

MASTER′S DISSERTATION

基于遗传粒子群混合算法的车辆路径问题研究

A Research on VRP Based on GA-PSO

学位申请人　陈洲宇

指 导 教 师　田野 教授

学 科 专 业　物流与供应链管理

学 位 类 别　全日制学术型硕士

分类号 _____密级_____

U.D.C_____

基于遗传粒子群混合算法的车辆路径问题研究

A Research on VRP Based on GA-PSO

学位申请人：陈洲宇

学　　号：2191202Z9014

学 科 专 业：物流与供应链管理

研 究 方 向：路径优化

指 导 教 师：田野 教授

定 稿 时 间：2022 年 5 月

摘 要

物流业作为一个现代综合型服务业，是国民经济至关重要的组成部分，对于国民经济发展有不可或缺的作用。运输配送是物流中的重要环节，同时运输费用也是物流成本的重要构成部分，在车辆路径问题中的关键是降低配送成本，即降低物流运输费用。物流运输费用的降低意味着配送经济且有效，不仅提高了客户满意度，同时也使得企业资源配置更为灵活。现如今信息化以及数字化已成为当代物流企业的一大趋势，信息化与数字化源于计算机技术的发展，对于物流企业竞争力的强化具有重要意义，因此本篇论文考虑借助数字化以及信息化对车辆的配送路径进行安排调度。

现今对于车辆路径问题求解方法的研究多集中于启发式算法，本文旨在对带时间窗的多车场车辆路径问题模型进行优化求解，该模型以配送总成本最小作为目标函数，同时基于现实情况对模型考虑了多车型配送以及尾单配送。在对前人的研究进行整理分析之下，本文通过分解法与全局优化法相结合，在第一阶段分解法中利用聚类算法与遗传算法得到初始解；在第二阶段全局优化法中，采用两种启发式算法混合对问题进行求解，在算法选择上，本文利用粒子群算法求解速度快、鲁棒性好以及遗传算法灵活、无方向性易于搜索全局最优解的特点将两者进行结合。通过使用双层编码，利用 K-means 聚类算法与遗传算法分别对初始解中的客户分配方案以及车辆配送路径进行优化，提高了算法的收敛速度，随后提出改进的遗传粒子群混合算法对初始解迭代，通过群体适应度方差在相邻两次迭代中变化小于某阈值时引入自适应杂交变异遗传算法，目标是分散过于聚集的粒子群跳出局部最优，增大继续搜索得到更优解的概率。

通过算例测试结果以及与其他算法的对比分析发现，本文提出的算法对复杂度较高的车辆路径问题模型具有良好的全局搜索能力，优化效果较佳，同时本文利用客户的定位和订单数据以及物流企业的车场车辆数据开发出一

套简单的以最小的配送成本为目标的物流配送路径模拟系统，该系统嵌入本篇论文所提出的混合算法，具有一定的理论意义与实际价值，能为物流企业的车辆调度提供一定的便利性，做出一定的贡献，符合现代物流企业对于信息化、数字化、可视化等的需求。

关键词：车辆路径问题；遗传算法；粒子群算法；K-means 算法

Abstract

Logistics industry, as a modern integrated service industry, is a vital part of the national economy. Transportation is an important link in logistics, The key in vehicle routing problem is to reduce logistics transportation cost. The reduction of logistics transportation costs means that transportation is economical and efficient, which not only improves customer satisfaction, but also makes enterprise resource allocation more flexible. Nowadays, informatization and digitalization have become a major trend of contemporary logistics enterprises. Saving logistics and transportation costs with digitalization and informatization has become the top priority to improve enterprise competitiveness.

At present, the research on solving methods of vehicle routing problem mostly focuses on heuristic algorithm, this paper aims to optimize and solve the multi-depot vehicle routing problem with time window (MDVRPTW) model. After sorting out and analyzing the previous studies, this paper combines decomposition method with global optimization method, and uses clustering algorithm and genetic algorithm (GA) to get the initial solution in the first-stage decomposition method. In the second-stage global optimization method, two heuristic algorithms are used to mix to solve the problem. In terms of algorithm selection, this paper uses genetic particle swarm optimization hybrid algorithm(GA-PSO), PSO has the characteristics of fast solving speed, good robustness and fast solving speed, and GA is flexible and easy to search the global optimal solution without direction. Firstly, the customer allocation scheme and the vehicle distribution route in the initial solution are optimized by k-means clustering algorithm and genetic algorithm respectively using double-layer coding, which improves the convergence speed of the algorithm. PSO was adopted to iterate the initial solution, and adaptive GA was introduced when the variance of population fitness was less than a certain threshold in two adjacent iterations. The goal was to break out of the local optimum for the dispersed particle swarm which was too concentrated, and increase the probability of obtaining a better

solution through continuous search. Through the test results and comparison with other algorithms, it is found that the proposed algorithm has better global search ability and better optimization effect.

Finally, using the client's location and order data and logistics enterprise's depot and vehicle data, this paper developed a set of distribution with the minimum cost as the objective function of simple logistics distribution path simulation system, the system has certain theoretical significance and practical value, can provide logistics enterprises vehicle scheduling with certain convenience, make a certain contribution, In line with modern logistics enterprises for information, digitalization, visualization and other needs.

Key words： VRP; Genetic algorithm; Particle swarm algorithm; K-means

目 录

1.绪论

1.1 研究背景和研究意义

1.1.1 研究背景

物流业作为一个现代综合型服务业，集合仓储、运输、配送、包装、信息等多个产业，是国民经济至关重要的组成部分，对于国民经济发展有不可或缺的作用。现代企业通过提升物流能力以强化自身竞争力，各个国家政府也在积极扶持加快物流发展以提高综合国力(刘春,朱俊林，2003)。

根据国家发展改革委整理数据如表 1.1 所示，2015~2020 年全社会物流总额维持稳步增长，在此之下社会物流总费用持续增加，由 10.8 万亿元增至 14.9 万亿元。其中，运输费用由 5.8 万亿元增至 7.8 万亿元、保管费用由 3.7 万亿元增至 5.1 万亿元、管理费用由 1.4 万亿元增至 1.9 万亿元。各费用占总费用比例维持稳定几乎无变化。

表 1.1 2015-2020 国内社会物流总费用情况及构成比例表

时间	物流总额（万亿元）	运输费用（万亿元）	保管费用（万亿元）	管理费用（万亿元）	物流总费用（万亿元）	运输费用占总费用比例	保管费用占总费用比例	管理费用占总费用比例
2015	219.2	5.8	3.7	1.4	10.8	53.70%	34.26%	12.96%
2016	229.7	6.0	3.7	1.4	11.1	54.05%	33.33%	12.61%
2017	252.8	6.6	3.9	1.6	12.1	54.55%	32.23%	13.22%
2018	283.1	6.9	4.6	1.8	13.3	51.88%	34.59%	13.53%
2019	298.0	7.7	5.0	1.9	14.6	52.74%	34.25%	13.01%
2020	300.1	7.8	5.1	1.9	14.9	52.35%	34.23%	12.75%

运输配送是物流中的重要环节，同时运输费用也是物流成本的重要构成部分，在车辆路径问题中的关键是降低配送成本，即降低物流运输费用。物流运输费用的降低意味着配送经济且有效，不仅提高了客户满意度，同时也使

得企业资源配置更为灵活。因此如何合理安排车辆的配送路径，节约企业的物流运输费用成为了提高企业竞争力的重中之重。

本文旨在通过建立带时间窗的多车场车辆路径问题模型，并用所设计的改进的遗传粒子群混合算法求解以对车辆路径进行优化，通过算例证明算法的可行性，并设计开发带时间窗的多车场车辆路径问题系统，在一定程度上为企业节约成本。

1.1.2 研究意义

研究车辆路径问题具备理论与实际意义。

从理论意义看，车辆路径问题是一个经典的 NP 难问题，NP 是指非确定性多项式（Non-deterministic Polynomial，NP），所谓非确定性即可通过一定数量的运算去解决多项式时间内可解决的问题。通俗来讲，NP 难问题或许无法找到一个已知的快速解决方案，但可以在多项式时间内验证一个解是否正确。该问题涉及运筹学理论、组合优化、算法等学科的知识，许多学者提出了一些启发式优化算法来解决此类问题，本篇论文拟在设计一套改进的混合启发式算法对具有特殊性与复杂性的带时间窗的多车场车辆路径问题进行求解以提高求解精度，为对该类高复杂度问题的求解提供一定的新颖思路。

从实际意义看，合理的车辆配送方案能够有效减少企业的运作成本，提高企业的效率与竞争力，本篇论文研究的带时间窗的多车场车辆路径问题模型增加了对多车型以及尾单配送的考虑，更加贴合现实企业所面临的实际配送问题，能够为企业的车辆调度路径优化提供一定的参考作用。

1.2 国内外研究现状

1.2.1 粒子群算法相关研究文献综述

粒子群算法的问世为求解车辆路径问题提供了强有力的工具，因其计算步骤简易且具有良好的鲁棒性等优点，运用在解决各类优化问题上都有良好的求解结果，因此应用粒子群算法求解车辆路径问题成为了近年来热门的研

究方向。

Shi 与 Eberthart（1998）为了改善算法的搜索能力和效率，将惯性权重的概念引入至基本粒子群算法中，以此完善了粒子群算法的速度更新公式，在很大程度上改善了算法的灵活性和效率。Clerc（1999）在基本粒子群算法中引入了收敛因子的概念用以更新速度，引入收敛因子的粒子群算法被作为是对粒子群算法引入特殊的惯性权重的一种特殊例子。张建科（2006）提出一种飞行时间自适应调整的粒子群算法，该算法的思想来源于：对于传统的粒子群算法，粒子的飞行时间都固定为 1。在实际操作的迭代初期，粒子的飞行时间应当长一些以此来增大粒子的移动步长从而加快算法的收敛速度；在后期则减少飞行时间缩小粒子的移动步长进行精细搜索。王俊伟（2006）提出一种通过梯度加速以更新粒子的粒子群算法，通过引入梯度信息确定移动步长从而更新粒子的位置及速度，提高粒子移动的效率，加快了算法的收敛速度。刘建华（2009）引进了用来约束粒子惯性权重的相似度概念，使相似度低的粒子的惯性权重相对较大，否则反之。通过动态调整惯性权重，处于群体历史最优位置及其附近的粒子就可以进行精细查找而不承担全局搜索的探测，大大提高了算法的搜索效率。张淑丽、张涛、崔岩、刘仁贵（2019）提出一种利用混合动力学对粒子的初始化过程进行改进的粒子群算法，该算法根据 Logistic 映射产生的向量对越界粒子进行处理，增加了种群的多样性，有效提高粒子群算法的收敛性和全局最优性。

一些文献将粒子群算法同其他算法相融合用以求解车辆路径问题。Angeline（1998）在自然界生物种群世代的演化中得到思路，将演化机制引进粒子群算法形成混合群体算法，该算法在每次迭代中将半数适应度低的粒子进行速度与位置的更新，调整至搜索区域中较好的位置。Lovbjerg、Rasmussen、Krink（2000）在粒子群算法中引入杂交思想，提出一种带有繁殖和子种群的粒子群混合算法。该算法的思想是每次迭代根据设定好的杂交概率在种群中选取一定数量的粒子，并随机将这些粒子两两匹配杂交产生同样数量的子代粒子，随后用子代粒子取代父代粒子得到新种群。Higashi、Iba（2003）为避免算法陷入局部最优，在位置及速度更新规则中引入了高斯变异，算法以一定的概率选出变异粒子再以高斯分布更新位置，如此将扩大初期搜索范围，同时中后期变异率逐渐减小，效率得以提高。叶建芳、王正肖、潘晓弘（2008）

提出免疫粒子群优化算法，将生物免疫机制引入算法，增加抗体浓度选择以及抑制机制对粒子群算法进行优化，能够更好地获得优质全局解。陈婷（2009）提出基于变异的粒子群算法，引入变异算子对粒子群算法陷入停滞时使各粒子朝新方向进行搜索。

1.2.2 遗传算法相关研究文献综述

遗传算法(Genetic Algorithm，GA)具有自组织、自适应性，可以同时处理多个群体中的个体，从串级进行搜索，搜索范围广，利于全局搜索，但该算法属于随机算法，结果稳定性较差。为提高遗传算法的性能，克服实际问题中所遇到的困难，利用遗传算法与其他优化算法相结合成为了目前改善遗传算法性能的主要手段。

Mühlenbein、Schomisch、Born（1991）采用了混合策略，把遗传算法与禁忌搜索算法(Tabu Search，TS)混合，通过测试用例计算得到了满意的结果。王璇（2010）分别将遗传算法与粒子群算法和人工免疫算法相结合形成混合遗传算法，遗传粒子群混合算法通过在遗传算法结束选择、交叉与变异操作之后引入粒子群算法保持了粒子群的多样性，而免疫遗传算法受到生物免疫系统启发，通过对遗传算法的种群接种疫苗保留适应度值最高个体作为记忆细胞提高了算法的搜索速度。该文献的混合遗传算法成功避免了早熟现象的发生，加快了算法的收敛速度，最后验证了算法的有效性。马超、邓超、熊尧、吴军（2013）提出一种遗传粒子群混合算法，在粒子群算法偶数代中引入遗传算子进行粒子更新，并在粒子聚集程度超过一定阈值时对一定粒子进行变异处理，较好地克服了遗传算法收敛精度不高的缺点。田欣（2016）将模拟退火算法的 Metropolis 准则引用至遗传算法中，对遗传操作之后的解进行判断接受与否，极大降低遗传算法陷入局部最优的概率。陈璐璐、邱建林、陈燕云、陆鹏程、秦孟梅、赵伟康（2017）通过对遗传算子改进，依据适应度值对粒子进行分类并执行删除与复制操作，并引入粒子群思想来更新粒子的速度和位置，相当程度上抑制了粒子群算法时常陷入局部最优。

对遗传算法自身的改进研究同样层出不穷。Tsutsui、Fujimoto、Ghosh（1997）提出了一种名为分岔遗传算法的新的遗传算法，该算法通过种群的收敛状态

和所得到的解将搜索空间划分为多个子空间，计算结果表明该算法具有较好的性能。张铃、张钹（1997）将统计启发式搜索算法(Statistic Heuristic Search Algorithm，SA)的理论用于遗传算法中，提出了统计遗传算法，新的算法不但提高了精度同时降低了计算的复杂度，克服了早熟现象的发生。李大卫、王莉、王梦光（1999）提出了基于优先关系的杂交算子，该优先关系同时将距离以及时间窗考虑进去，证明了使用该优先关系得到的杂交算子优势更为明显，求解带时间窗的车辆路径问题得到的结果更为出色。邹彤、李宁、孙德宝、李菁（2004）提出一种新的遗传算法编码表示，将染色体用基因序列表示，通过基因序列中的排序值大小顺序决定配送路径，经过算例验证能够有效解决多车场车辆路径问题。徐晓华、陈崚、陈宏建（2006）从人类进化演变过程中人口增长的规律获取灵感，即人类从低级进化至高级、人口数量由少至多对应解的质量由劣至优、种群规模由小到大，由此提出了一种可变种群规模的遗传算法，通过利用离散的 Logistic 模型控制种群规模进行求解，证明了该算法对比其他改进的遗传算法具有一定的优越性。王运涛、刘钢、薛俊芳（2022）对遗传算子进行改进，根据染色体的适应度值以及进化迭代次数设置了调节杂交概率与调节变异概率，当染色体的适应度值大于本次迭代种群平均适应度值时赋予该染色体较小的杂交概率和较大的变异概率，如此提高了该优秀染色体被保留的概率，加快算法在局部方向的收敛速度；反之同理，降低了劣质染色体被保留的概率。这种调节机制能够良好地扩大算法的搜索范围。

1.2.3 车辆路径问题相关研究文献综述

车辆路径问题（Vehicle Routing Problem，VRP）最早由 Dantzig 和 Ramser 提出，其定义可以解释为：已知所有顾客的货物需求及位置坐标，企业在满足一定的如时间限制、车辆载重量限制、行程里程等约束条件下设计适当的路线完成顾客的需求。要求尽可能少的成本，即尽可能少的车辆数、里程或耗时。

带时间窗的车辆路径问题(Vehicle Routing Problem With Time Windows，VRPTW)顾名思义就是在传统的车辆路径问题中考虑了客户接收服务的时间窗约束，Solomon（1987）最先在车辆路径问题中把时间窗考虑进去，运用多

种启发式算法进行求解，通过对各个结果的对比发现，其中一种插入启发式算法能够得到满意的结果。同年 Kolen、Rinnooy、Trienekens（1987）通过分支定界法求解以最短路为优化目标的带时间窗的车辆路径问题，得到了一个确定下界。Desrochers、Desrosiers、Solomon（1992）利用线性规划松弛与列生成法通过动态规划成功求解了包含 100 个客户的 VRPTW，规模为当时其他研究的六倍。Potvin、Kervahut、Garcia、Rousseau（1996）提出一种禁忌搜索算法用以求解带时间窗的车辆路径问题，在耗时与里程数上更优于 Solomon 的插入启发式算法。Taillard、Badeau、Gendreau、Guertin、Potvin（1997）针对具有软时间窗的车辆路径问题提出了一种通过在两条路径之间交换客户序列来创建邻域的禁忌搜索算法，并成功求解。Zhao、Wu、Wang、Ma、Wang、Sun（2004）采用粒子群算法求解带时间窗的车辆路径问题,首先分别利用节约算法(Saving Algorithm)、最近邻算法(Nearest Neighbor Algorithm)和 Solomon 提出的插入启发式算法初始化参数，通过粒子群算法对客户进行排序与车辆分配，并采用最近邻算法对路线进行优化，结果表明，粒子群算法可以很好地解决带时间窗的车辆路径问题，采用插入启发式算法的粒子群优化算法效率最高。张丽艳、庞小红、夏蔚军、吴智铭、梁硕（2006）年将粒子群优化算法于模拟退火算法(Simulated Annealing，SA)相结合，提出了一种用于求解带时间窗的车辆路径问题的混合粒子群算法，该法首先使用粒子群算法得到粒子的优化结果，而后调用模拟退火算法评价每个粒子的适应度并生成解，计算结果表明该混合粒子群算法明显优于遗传算法、改进粒子群算法。

随着单车场车辆路径问题研究成果越来越成熟，学者们提出更复杂的多车场车辆路径问题(Multi-Depot Vehicle Routing Problem,MDVRP)，Renaud、Laporte、Boctor（1996）提出一种禁忌搜索算法，在 23 个测试用例中共生成 19 次最优解。邹彤等（2004）年提出一种通过编码表示出客户以及各车场所需车辆及路径的遗传算法，有效地实现对多车场配送车辆路径问题的优化。

同单车场车辆路径问题的研究历程相同，基于趋近成熟的多车场配送车辆路径问题的研究成果，学者开始引入时间因素展开更深层次的研究，及带时间窗的多车场车辆路径问题。Wen、Meng（2008）提出一种改进粒子群算法，采用非线性时间递减函数改变惯性权值，使全局搜索与局部搜索达到平衡。Wang、Wu（2012）提出了一种引入种群熵的概念的自适应粒子群算法，

该算法通过种群熵对种群的多样性进行描述，并且在迭代过程中根据种群熵的变化情况对种群进行自适应调节，使得算法达到全局搜索与局部搜索的良好平衡，提高算法的性能，对于求解带时间窗的多车场车辆路径问题计算结果表明，该算法跳出局部最优能力强于遗传算法和粒子群算法。王小会（2015）针对利用粒子群算法求解带时间窗的多车场车辆路径问题时产生不可行解较多的问题设计了优化不可行解的粒子群算法，该算法引入变异算子增强了粒子的寻优能力。计算结果表明优化后的粒子群算法精度以及收敛速度都得到提高，能够对带时间窗的多车场车辆路径问题进行高效求解获得最优解，孙艺婕（2020）提出了改进节约算法与混合遗传算法结合的两阶段算法用于求解带时间窗的车辆路径问题，通过节约算法获得初始解提高了算法整体收敛速度，并在遗传算法中对遗传算子引入代沟概率 GGap，确保适应度值最高的个体为算法的最优解。

1.2.4 研究述评

在综合整理国内外对车辆路径问题的研究成果后发现，虽然更为复杂的车辆路径问题不断被提出及研究，但是仍有一些缺陷需要去考虑。

（1）目前对于车辆路径问题的相关研究文献多集中于单车场问题或是单一的多车场问题，模型较为简单，没有很好地考虑到例如时间要求或是交通路况、车型不统一等更为复杂的实际背景，研究提出的方案结果不具备较强的实用性。

（2）在多车场车辆路径问题的算法选择上，大部分文献选择单一的改进的启发式算法或是多种基本的启发式算法结合，这种方式有助于结果的改进优化，但仍有提高的空间，在算法选择设计上可考虑多种改进的算法相结合，针对算法各自的特点进行组合并进行针对性改进，以更好地发挥优点克服缺点，使得求解结果更趋近于问题的全局最优解。

1.3 研究内容和研究方法

1.3.1 研究内容

本篇论文对带时间窗的多车场车辆路径问题从定义到建立数学模型以及求解算法的选取和优化进行了详细的阐述，基于本文提出的改进的遗传粒子群混合算法通过算例进行求解，验证了算法对贴合现实的模型求解的有效性，并设计实现了以该算法为基础的带时间窗的多车场车辆路径问题模拟系统。

下述为本篇论文的具体框架：

第一章是绪论，最初详细介绍描述了国内的物流业所处背景和意义，说明车辆路径规划的重要性以作为本文研究的铺垫。随后对国内国外有关粒子群算法、遗传算法以及车辆路径问题的研究现状进行综述，最后对本文的研究内容、研究方法、技术路线等进行展示说明。

第二章是对本篇论文所涉及的相关基础理论描述并展开分析。详尽叙述了车辆路径问题的定义、基础理论和求解方法，由求解方法引出粒子群算法与遗传算法，并对两者的定义、特点等进行描述。此章节为本篇论文问题研究、模型建立和算法改进的基础。

第三章是在结合第二章的理论基础上建立了以配送总成本最小为目标函数的带时间窗的多车场车辆路径问题模型，同时该模型基于现实更为复杂的情况，考虑了多车型以及尾单配送，并根据该模型确定了分布式求解方法以及混合算法的选择，为下一章模型求解以及算法研究打下基础。

第四章是遗传粒子群混合算法选择策略与研究。首先采用 K-means 聚类算法和遗传算法分别获得初始客户分配方案和车辆配送路径得到初始解，其次通过改进的遗传粒子群混合算法对初始解进行优化，之后通过引入算例对算法求解模型的有效性进行检验。

第五章是基于本篇论文算法的带时间窗的多车场车辆路径问题模拟系统的设计与实现。该系统可实现人工建立数据库、导入信息、配送总成本以及对车辆配送路径的展示等。

第六章的主要内容为总结归纳研究内容及不足并展望未来，通过对本篇论文的主要研究内容进行总结，同时引出此研究中未考虑的问题及不足，并

对本篇论文的研究方向进行展望。

1.3.2 研究方法

参考相关研究所采用的研究方法，本研究拟采用以下两种研究方法以支撑模型与求解车辆路径问题算法的设计：

（1）文献研究法

文献研究法的作用和目的是对研究对象和主题形成科学的认知。需要研究者通过收集、鉴别与整理文献材料，并加以科学、理性的分析和研究。本篇论文通过在知网以及谷歌学术等学术研究平台搜索关键词:“车辆路径问题”、“粒子群算法”、“遗传算法”等的相关文献，对所得文献检索结果进行整理与分析，认识把握国内外的研究现状，对本文采用的理论背景和算法提供文献支撑。

（2）定量研究法

定量研究法是一种根据有关特定研究对象的信息对其总体得出结果而进行的方法，定量研究中的信息都是由某种数字通过处理来呈现的。本篇论文构造运筹学模型通过 Python 求解模型，得到优化结论。

1.3.3 技术路线图

技术路线如图 1.1 所示。

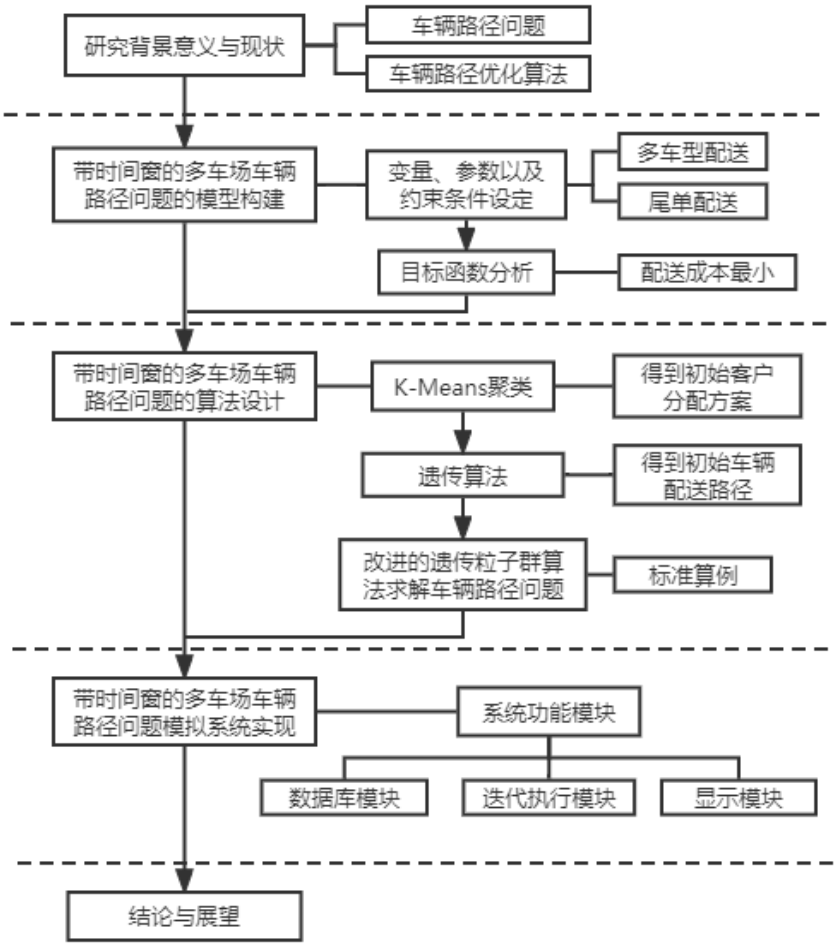

图 1.1 技术路线图

2.相关的基础理论

2.1 车辆路径问题的相关理论

2.1.1 车辆路径问题要素构成

车辆路径问题的要素构成包括：目标函数、约束条件、车辆、客户、配送车场、配送网络等。

（1）目标函数：目标函数是指通过设计变量来体现所需求的目标形式，即设计变量的函数。基本的目标包括配送总成本最低、运输路程最短，除此之外还有可供选择的多个目标：车辆调用最少、客户满意度高等。

（2）约束条件：约束条件是指针对决策目标的各项限制，通常以不等式或者是方程式的形式呈现。对常见的约束条件包括以下几条：1.满足所有客户对货物的需求量。2.车辆总承重务必小于车辆最大载重量限制。3.尽可能遵守客户提出的配送时间要求。4.车辆起点从车场开始进行配送，在配送任务结束后返回起点车场。5.每一个客户只由一辆车提供服务。

（3）车辆：车辆即用来装载并配送货物的载具。主要的属性包括：最大装载量、最远行驶距离、行驶速度以及所在配送中心等。

（4）客户：顾客属性包括：坐标、货物的需求量、种类以及接受服务的时间要求等。

（5）配送车场：配送车场可用于仓储、配送等活动，在不同的问题中配送车场的个数可以是一个也可以是多个，即配送中心或需要满足所有客户的需求或需要满足部分客户的需求。

（6）配送网络：配送网络是车辆路径规划的重要组成部分，是囊括了配送活动中相关设施与组织的大集合。其包括：配送车场、客户、车辆配送路径以及权值。

某车辆路径问题结果示意图如下图 2.1 所示。

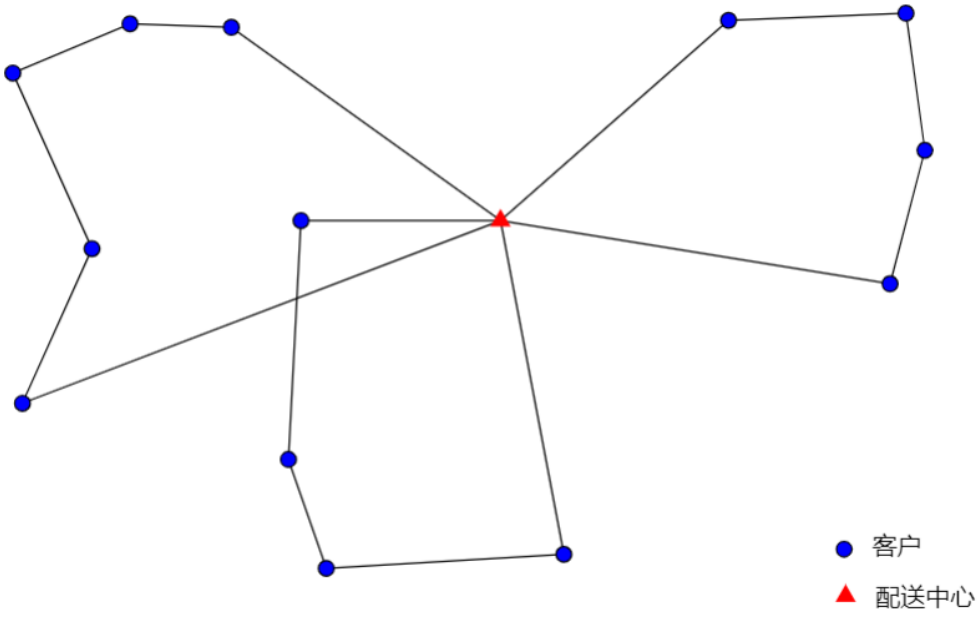

图 2.1 某车辆路径问题结果图

2.1.2 车辆路径问题的分类

车辆路径问题在提出过后，经过实际情况构建的各种约束条件被引入车辆路径问题中，由此出现了该类问题的多种变形，一般被分类为以下几种：

（1）按照客户对于配送时间是否有特定的要求进行配送，车辆路径问题能够分为不带时间窗的车辆路径问题与带时间窗的车辆路径问题，其中时间窗包括硬时间窗与软时间窗。

（2）按照配送中心的数量进行配送，车辆路径问题能够分为单车场配送车辆路径问题与多车场配送车辆路径问题。

（3）按照车型进行配送，车辆路径问题能够分为单车型车辆路径问题与多车型车辆路径问题，大多数车辆路径问题为单车型车辆路径问题。

（4）按照配送任务要求，车辆路径问题能够分为取货车辆路径问题、送货车辆路径问题、同时取送货车辆路径问题。

（5）按照目标函数的数量，分为多目标车辆路径问题和单目标车辆路径问题。

（6）按照配送中心与车辆是否存在约束关系，可将车辆路径问题分为封

闭式车辆路径问题和开放式车辆路径问题，封闭式车辆路径问题限制车辆在所有配送结束后需返回原配送中心，而开放式车辆路径问题则不要求车辆必须返回原配送中心。

下文将详细对多车场配送车辆路径问题以及带时间窗车辆路径问题进行介绍。

2.1.3 多车场配送车辆路径问题概述

（1）多车场配送车辆路径问题的概念

随着城市经济以及交通运输的发展，互联网技术的井喷式爆发，企业间的竞争趋向白热化，客户的需求趋向多样化，需求量也不断攀升。仅仅依靠一个配送中心完成路程过长、交通复杂且客户分散的配送任务难以保证配送的服务质量，影响客户的满意度，随之将产生高昂的配送成本，影响企业的竞争力。因此现今大部分大中型企业都采取建立多个配送中心来提高运输效率以及服务质量，多车场配送车辆路径问题示意图如下图 2.2 所示。

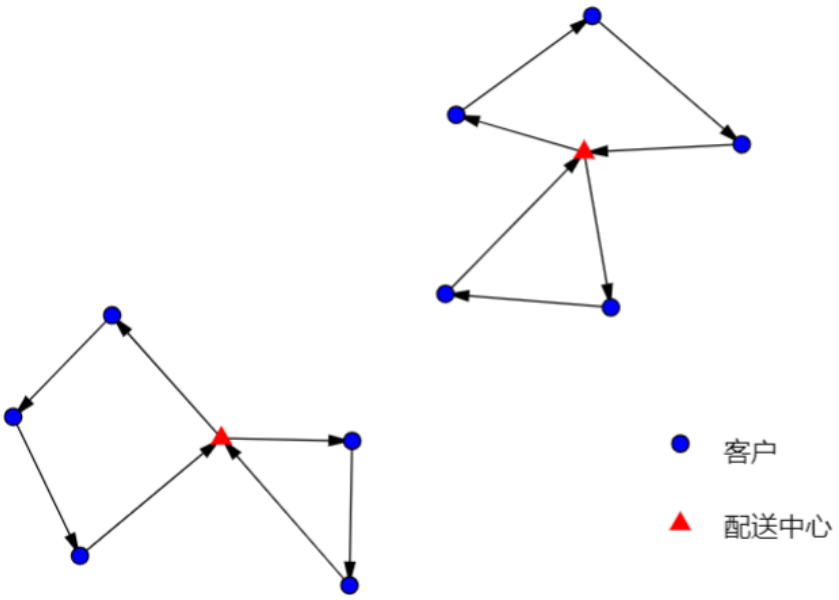

图 2.2 多车场配送车辆路径问题示意图

显而易见，多车场配送车辆路径问题需要考虑的情况比单车场配送车辆路径问题更为复杂，研究难度也更大，于是求解多车场配送车辆路径问题的

方法相比单车场配送车辆路径问题更加精细繁琐。求解该类问题需要两个步骤：第一步，在车辆最大载重量的约束条件下，针对分散的各个客户分配至适合的配送中心；第二步，对每辆车规划合理的配送路线以完成配送任务。

（2）多车场配送车辆路径问题的目标优化分类

多车场配送车辆路径问题常有如下几个目标函数优化方向。

①配送成本最低：现如今各大电商、物流企业实现盈利的方式大多来源于为客户提供高质量和高满意度的服务，物流配送是服务的主要方式，科学规划车辆配送方案能够大大节省企业在物流成本上的耗费。多数车辆路径问题都以配送成本最低作为目标函数展开优化。

②运输路程最短：运输路程的长短直接影响配送总成本，对运输路程进行优化与降低配送总成本等同，这也是大多数车辆路径问题的主要目标函数之一。恰当安排车辆配送路径能够有效降低车辆的行驶里程，实现配送距离最小化，节约配送总成本。

③配送时间最短：归咎于企业间竞争压力不断增大，为提高配送效率和客户满意度，无论是企业还是客户方都在配送时间上加以要求，如果配送超过所约束的时间，将产生与时间所对应的惩罚成本，降低客户的满意度，进而影响整个企业自身的发展。

④车辆数量最少：车辆的使用数同样影响配送任务的配送成本，若配送任务所安排的配送车辆数过多，将会增大出车成本进而增大配送总成本。但该目标函数具有一定局限，若配送任务所安排的配送车辆数过少，可能会导致部分客户的需求得不到满足，仍将引起额外的成本支出。因此恰当调度管理运力同样是企业需要考虑的优化目标。

2.1.4 带时间窗的车辆路径问题概述

（1）带时间窗的车辆路径问题的概念

时间窗所指的是一个具体的时间区间，上下界分别为提供配送服务的最早时间和提供配送服务的最晚时间。在国民经济飞速发展的当下，客户的需求量越来越大，企业之间为抢占资源市场以及客户对服务质量的追求大幅提高，时间窗成为了车辆路径问题的关键因素，与企业所拥有的资源规模息息

相关。

带时间窗的车辆路径问题一般使用以下两类目标函数来进行优化，一是目标函数是配送时间最短的单目标数学模型，适用于如冷链物流、应急物流的车辆运输规划；其二是以最小化配送成本与时间成本之和作为总目标函数的多目标数学模型，通常为方便计算，会通过将时间成本这一目标函数变为时间窗约束条件以简化模型，适用于时效性物流的车辆运输规划。

（2）带时间窗的车辆路径问题的分类

对于带时间窗的车辆路径问题，路径规划不仅要考虑车辆的配送路径即客户分配，还需要考虑车辆的出发时间以及服务客户的先后顺序。时间窗包括以下几种类别。

①硬时间窗：指配送车辆不得违反客户所要求的提供服务时间区间，要求配送车辆必须在指定的时间区间内到达客户点并为客户提供服务，如图 2.3 所示，若车辆晚于最晚服务时间 LT 抵达客户点，客户有权拒收，此时将产生较高的惩罚成本 MaxS;若车辆早于客户要求的最早服务时间 ET 抵达客户点，客户同样有权拒收，随之产生惩罚成本 MaxS。

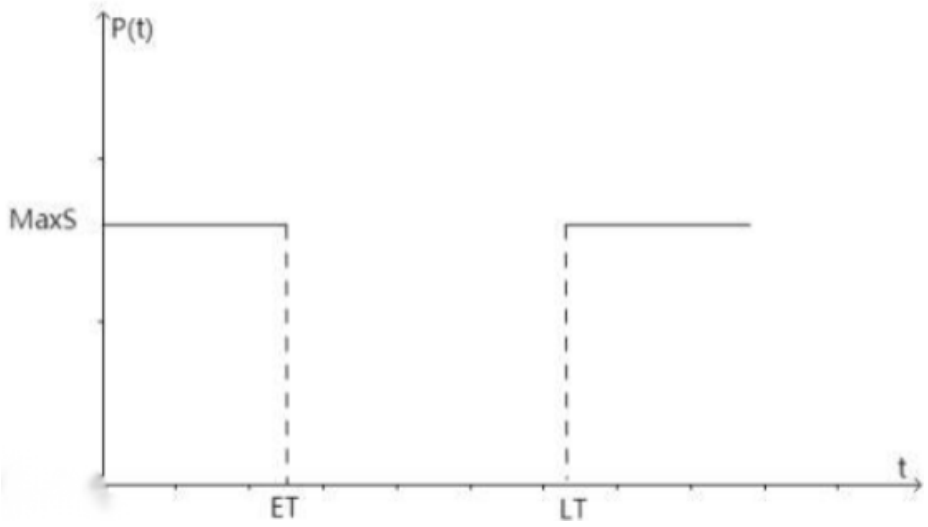

图 2.3 硬时间窗

②软时间窗：指接受配送车辆不一定在时间区间内到达客户点为客户提供服务，如图 2.4 所示，如果车辆早于最早服务时间 ET 抵达客户点，则会依照早到时间的长短 ET-EET 产生对应的机会成本 P（t）；若车辆晚于最晚服务时间 LT 抵达客户点，即使客户同意接收货物，但迟到也降低了客户的满意度，

此时需要根据迟到时间的长短 ELT-LT 引入一个惩罚机制，越长的迟到时间意味着越昂贵的惩罚成本 P（t）。以处罚代替等待与拒收是软时间窗与硬时间窗的最大不同。

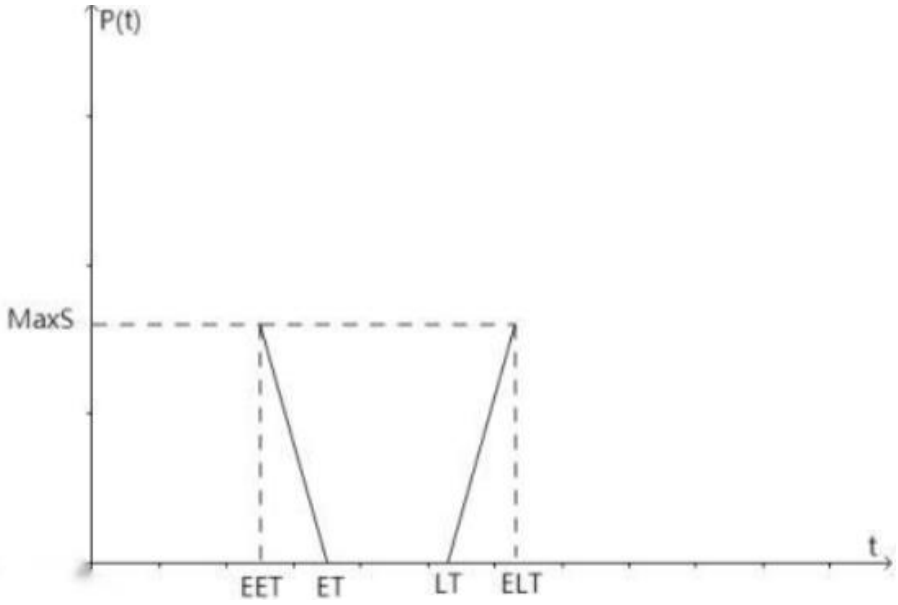

图 2.4 软时间窗

③混合型时间窗：即在一个配送问题中既存在软时间窗又存在硬时间窗。如图 2.5 所示，对于软时间窗来说，客户可接受车辆在弹性时间范围内[EET,ELT]抵达客户点提供服务，惩罚成本根据抵达时间的长短随之产生；对于硬时间窗来说，若车辆未在允许的弹性范围内抵达客户点提供服务，客户可以要求拒绝服务，这种情况将会发生非常昂贵的惩罚成本 MaxS。

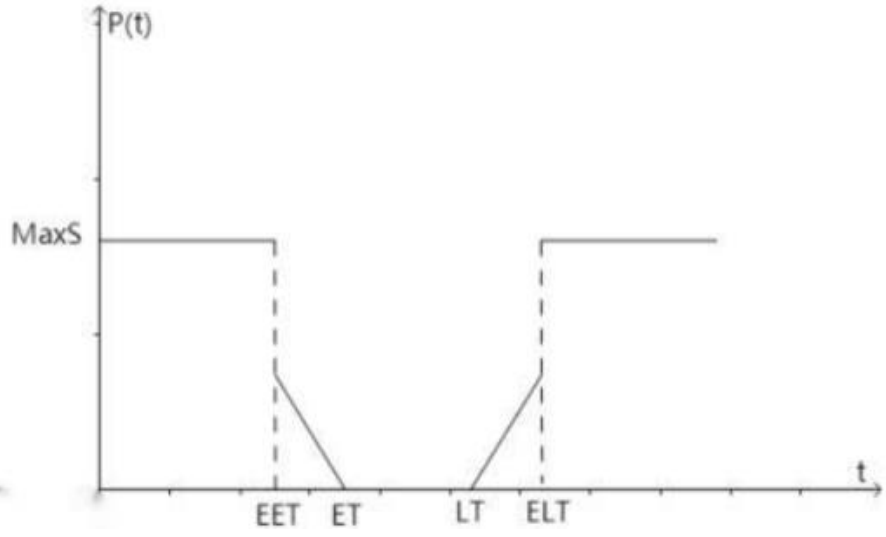

图 2.5 混合时间窗

2.1.5 车辆路径问题的求解方法

对车辆路径问题进行求解的算法分别是精确算法与启发式算法。

（1）精确算法

在车辆路径问题的研究初始，学者们主要是通过简易的数学模型来进行求解，用于求解该数学模型的此类算法就是精确算法。精确算法在小规模的车辆路径问题上通常可以精确求出最优解，但考虑到车辆路径问题的规模扩大以及实际当中所涉及的约束条件更为复杂的情况，仅仅通过精确算法进行求解效率过低，可能无法很好地得出最优解。

精确算法包括分支定界法、割平面法、列生成法与动态规划法等。

①分支定界法最早由 Land、Doig（1960）提出，该方法的基本思想是将对应的不包括整数约束的车辆路径问题(B)的最优解看作起始点，如果该最优解为整数解，则该解就是包含整数约束的原车辆路径问题(A)的最优解，若非整数解则对 B 的解的相邻整数进行分支形成两个子问题并求解。如若求解两个子问题得到的最优解为整数解，则结束对该问题的分支求解，若非整数解则重复分支求解。该方法只适用于求解小规模的车辆路径问题，在面对大规模时，该算法的运算过程相当缓慢繁琐，求解时间过长。

②割平面法于 1958 年问世，该方法通过在求解对应的不包括整数约束的车辆路径问题(B)上增加约束条件(割平面)来割掉 B 可行域的片段，其中被分割的片段中不能包括整数解，直至分割后获取的可行域片段中出现某个整数坐标的极点正好是包含整数约束的原车辆路径问题(A)的最优解即分割结束。该算法同样因运算时间过长不适用于求解大规模的车辆路径问题。

③列生成法本质上是单纯形法的一种形式，Desrosiers、Soumis、Desrochers（1984）将列生成法与分支定界法相结合求解了带时间窗的车辆路径问题，但同样受困于运算时间过长。

④动态规划法由 Bellman（1966）提出，该算法将问题划分为多个阶段并转化为依次求解多个具有递推关系的单阶段问题，通过简化的单阶段问题计算求得最优解，同样仅适用于求解小规模的车辆路径问题。

精确算法通过严格的数学方法求解，当问题规模越大，求解的难度以及时间呈现指数级增长，所以该类算法在实际中只能运用于求解小规模的车辆路径问题，并且这类算法通常都被设计为解决某一特定问题，因此在实际中应用受限。

（2）启发式算法

针对大规模的车辆路径问题，可将求解最优解的目标转变为为求解近似最优解的目标，这样既降低了求解大规模的车辆路径问题的计算难度又可以达到求解大规模的车辆路径问题所需要的计算精度。该类算法能够在较短的时间内给出符合实际需要的满意解，同时容易操作和修改，弥补了精确算法的短板，这类算法统称为启发式算法。

自从 Savelsbergh（1985）证明了带时间窗的车辆路径问题属于 NP 难问题后，启发式算法便成为了此类问题的重要研究方向。启发式算法包括传统启发式算法与现代启发式算法。

①传统启发式算法

节约算法由 Clark、Wright（1964）提出，该算法的求解步骤是在已知初始解的基础上，根据算法计算出所有顾客对的节约额，按照所有顾客对的节约额从大到小的排列顺序来组织连续运输，如连接该顾客对满足配送条件(车载重量、里程限制等)则在解中加入该顾客对，直至所有顾客对遍历结束。

最近邻算法是一种贪婪算法，该算法的思想是每一步都在未被访问的顾客节点中选择距配送中心最近的顾客节点，在满足车辆最大载重量以及车辆最大里程的限制下将其加入路径中，不断重复该步骤直至客户节点全部加入路径。该算法具有短视性，每一步都直接影响路径的走向，进而导致错过最优解。

最近插入法融合了节约算法与最近邻算法的思想，首先建立从配送中心到距离配送中心最近的顾客再回到配送中心的往返式子回路，在剩余的顾客当中以满足距子回路某点最近为条件搜索节点，随后在子回路当中确定一条使得节点两头距离子回路最近节点的距离总和减去这条弧的距离最近的弧，将这个节点加入到子回路之后删去该弧，相当于在子回路当中增添新节点之后得到所增加路途最短的效果，重复上述步骤除非全部节点都已增添进子回路。

扫描法由 Gillet 与 Miller（1974）年提出，该算法的主要思想是建立一个极坐标系，以配送中心作为极坐标的原点，并将所有的顾客点设置在极坐标系中，之后以车辆最大载重量以及里程限制作为约束条件从零点顺时针或逆时针扫描顾客点并进行顾客分组，直至需求总量超过车辆最大载重量或行驶距离超过里程限制，结束该条线路。重复上述步骤，当全部顾客都已分组时结束扫描输出分组情况。

②现代启发式算法

现代启发式算法的思想来源于自然界生物的行为以及规律，该类算法特点是高质量、高效率。对规模庞大且复杂度高的车辆路径问题进行求解的效果较佳。其中包括遗传算法、模拟退火算法、禁忌搜索算法、粒子群算法和蚁群算法等。

遗传算法是一种随机搜索算法，其基本思想是对自然界的生存法则以及自然演化规律进行模拟，算法首先是将问题的可行解按照某种编码方式形成初始种群，随后通过建立符合实际的适应度函数来计算每个个体适应度值并挑选若干个个体纳为进化种群，根据设定的概率对种群进行进化并迭代，最终经过“适者生存”法则筛选留下的最大适应值个体即问题的最优解。

模拟退火算法是一种建立在热力学原理之上的模拟固体退火过程的算法。该算法在初期需要设置初始温度，并在满足内循环停止准则之前不断更新解，随后进行“降温”以缩小搜索范围，最终获得最优解。该算法需合理选择初始温度，如果温度过高会浪费大量的搜索时间。

禁忌搜索算法起源于局部搜索算法，相较于模拟退火算法和遗传算法，禁忌搜索算法较为简单。该算法的主要思想是引入禁忌表，为改进局部搜索容易陷入局部最优的缺陷，禁忌搜索算法利用禁忌表将所有经历过的可行解进行记录，随后的搜索便有选择地跳过记录在禁忌表中的点，以此来跳出局部最优点，提高了全局寻优的效率。但该算法的求解质量受初始解质量的影响，若初始解质量不高，将会导致收敛速度过慢。

粒子群算法由 Kennedy 和 Eberhart（1995）首次提出，该算法通过模拟鸟类、鱼群等动物群体觅食行为来寻求问题的最优解。主要思想是将每只鸟抽象成为有记忆且互相信息共享的粒子，通过群体最优位置及自身每个粒子的最优位置来确定自身的运动方向及速度并进行迭代最终搜索到最优解。该算

法需要设置的参数较少、计算简单。

蚁群算法(Ant Colony Optimization,ACO)是 Colorni、Dorigo、Maniezzo（1992）年提出的一种利用蚂蚁觅食的行为机制来求解复杂优化问题的算法。该算法的基本原理来源于蚂蚁觅食行为可以在没有指挥的情况下快速地发现从蚁巢到食物源的最短路径，这是由于每只蚂蚁在寻找路径的过程中能够留下其余蚂蚁可以识别的“信号素”，其他蚂蚁可以通过识别“信号素”快速聚集。蚁群算法通过制定符合实际情况的转移规则，使得人工建立的蚁群在转移时不断留下“信号素”并进行迭代，在满足终止条件后可以得到“信号素”浓度最高的一条路径，即最优解。该算法需要考虑蚂蚁数目的大小以及选择合适的信息素挥发系数，如果蚂蚁数目较小，容易出现过早停滞现象。挥发系数设定过小，算法的随机性会降低，全局搜索能力也随之减弱。

2.2 粒子群算法描述

上文对粒子群算法的概念已经进行过阐述，粒子群算法在初期会初始化生成满足种群规模大小的随机粒子，迭代过程中根据追踪粒子自身历史极值即个体极值和群体历史极值即群体极值来更新自己的速度和位置。

2.2.1 粒子群算法的数学描述

粒子群算法是一种基于种群的算法，种群由 N 个粒子组成，在 D 维搜索空间中寻求全局最优解。每个粒子所处位置就是问题的一个解，记为$x_i =$（x_{i1}，x_{i2}，…，x_{iD}），通过计算每个粒子的适应度值来评价粒子的优劣。个体历史最优位置$p_i =$（p_{i1}，p_{i2}，…，p_{iD}）用每个粒子历次迭代中最好的适应度值所对应的位置来表示，群体历史最优位置$p_g =$（p_{g1}，p_{g2}，…，p_{gD}）用整个群体中历史最好的适应度值所对应的位置来表示。每一次迭代中，粒子通过改变速度来改变位置，使搜索区域尽可能遍历解空间，粒子 i 速度记为$v_i =$（v_{i1}，v_{i2}，…，v_{iD}）。粒子群的进化方程可描述为：

$$v_{ij}^{(k+1)} = \omega v_{ij}^{k} + c_1\lambda_1^k(p_{ij}^k - x_{ij}^k) + c_2\lambda_2^k(p_{gj}^k - x_{ij}^k) \quad (1)$$

$$x_{ij}^{(k+1)} = x_{ij}^{k} + v_{ij}^{(k+1)} \quad (2)$$

式中，i 表示第 i 个粒子，$i = 1，2，\cdots\cdots，N$，N 表示种群的规模，取值与所需要优化问题的规模相关；下标 j 表示粒子的第 j 维，$j = 1，2，\cdots\cdots，D$，D 表示解空间的维数，即自变量的个数；k 表示第 k 代；ω为惯性权重，它可以调整全局和局部搜索之间的平衡，当ω取值较大时，粒子保持较大程度的运动惯性，速度较大，全局搜索能力较强；反之速度较小则具备较强的局部搜索能力。$c_1，c_2$称为学习因子或加速因子，一般介于$(0，2)$之间取值且$c_1 = c_2$，通过公式（1）可看出，一是两个学习因子各自对粒子追踪个体历史最优值与群体历史最优值的步长进行调节，若c_1相对大于c_2，那么粒子比起通过群体信息确定最优值更乐意通过自身信息确定最优值，如此可能造成粒子过度迂回于局部范围，收敛速度将会因此放慢；反之若c_1相对小于c_2，那么粒子比起通过自身信息更乐意依靠群体信息确定最优值，如此可能造成粒子在算法初期就陷入局部最优，引起早熟的发生，因此设定合适的学习因子至关重要。λ_1和λ_2是$[0，1]$之间满足均匀分布的相互独立的随机数；二是粒子速度的更新取决于三部分，第一部分是粒子的先前速度，第二部分是“自我认知”部分，即粒子对自身位置的认知能力，第三部分是“社会”部分，表示群体中粒子之间的信息资源共享能力，引导粒子趋向种群最优位置。

粒子的最大速度V_{max}是一个对算法有重大影响的参数，该参数划定了粒子在迭代中每一个维度上移动的范围。若V_{max}取值较大，算法的搜索能力就较强，但由于移动步长太大容易错过最优解所在空间；若V_{max}取值较小，算法的搜索能力较弱，但优于移动步长太小很容易陷入局部最优解。粒子每一维的速度都要控制在$[-V_{max}，V_{max}]$内，若超出范围，则需要进行如下修正：

$$v_i^k = v_{max}, if\ v_i^k > v_{max} \quad (3)$$

$$v_i^k = -v_{max}, if\ v_i^k < -v_{max} \quad (4)$$

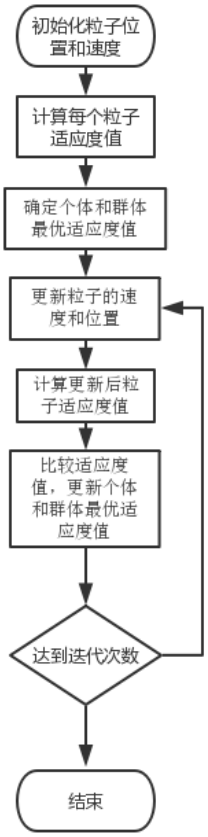

图 2.6 粒子群算法流程

基础粒子群算法流程如图 2.6 所示，步骤如下：

Step 1：完成粒子群参数的初始设定，包括种群规模以及每个粒子的初始位置和速度。

Step 2：依据适应度函数求出每个粒子所处位置的适应度。

Step 3：对每个粒子当前的适应度值与个体历史最好位置的适应度值相比，若前者较优，则用当前的适应度值所对应的位置替换为个体历史最优位置，否则不变。找出当前迭代群体中的最优适应度值，与群体历史最优位置的适应度值进行比较，若前者较优，则将其所对应的位置替换为群体历史最优位置，否则不变。

Step 4：依照式（1）与式（2）对粒子的速度与所处位置进行更新改变。

Step 5：判断迭代次数是否到达最大迭代次数，若到达则终止迭代，否则转至 Step 2。

2.2.2 粒子群算法的特点

粒子群算法广受欢迎来自于该算法所具备的优点，主要包括如下方面：

（1）算法基本原理简单易理解，所需要进行调整的参数较少且易于实现。

（2）算法同时考虑个体以及群体，在搜索过程中，粒子之间不仅会谋求合作同时也相互角逐竞争，整个群体在合作与竞争相结合的机制下搜寻最优解。

（3）算法属于分布式求解。个体在搜索过程中不全部依靠问题中的信息，鲁棒性较强。

随着学者们对粒子群算法研究的深入，粒子群算法已经广泛应用于各个领域，但算法仍具有缺陷，缺陷包括如下方面：

（1）搜索空间总是局限于有限范围当中，并不能对整个可行空间进行遍历搜索。

（2）粒子群算法容易陷入局部最优，导致无法找到最优解。

2.2.3 粒子群算法的改进方法

为了弥补粒子群算法的不足之处，学者们针对粒子群算法提出了不少改进措施，几种典型的粒子群算法改进方式如下。

（1）基于惯性权重的改进

惯性权重ω的取值与算法搜索能力有着重要的关系，它直接控制粒子上一次飞行速度对当前速度的影响，间接决定了算法全局搜索和局部搜索能力，因为当惯性权重取值较大时，粒子的移动步长较大，全局搜索能力较强；反之局部搜索能力较强。因此在不同的迭代周期内决定恰当的惯性权重较为关键。Shi(1998)提出了惯性权重线性递减方法，即ω随着迭代次数的增加而减少，计算公式如下：

$$\omega = \omega_{max} - \frac{k}{k_{max}}(\omega_{max} - \omega_{min}) \tag{5}$$

其中，惯性权重的极大值与极小值分别通过ω_{max}与ω_{min}进行表示；k 代表当前迭代次数，k_{max}代表设定的最大迭代次数。Shi 通过对多个算例进行大量实验得出当ω_{max}取值 0.9，ω_{min}取值 0.4 的时候算法性能较好。

（2）基于粒子位置更新公式的改进

张建科（2006）年提出一种飞行时间自适应调整的粒子群算法加快算法前期收敛速度，减慢后期算法收敛速度。公式如下：

$$x_{ij}^{k+1} = x_{ij}^{k} + v_{ij}^{k+1} t_i^{k+1} \tag{6}$$

式中t_i^{k+1}表示第 i 个粒子在 k+1 次迭代中的飞行时间，飞行时间的确定方案有多种，主要如下：

方案一：
$$t_i^{k+1} = t_{max}(1 - \gamma \frac{k+1}{k_{max}}) \tag{7}$$

方案二：
$$t_i^{k+1} = t_{max}[1 + \gamma \cos(\frac{k+1}{k_{max}}\pi)] \tag{8}$$

方案三：
$$t_i^{k+1} = t_{max}[1 - \gamma(\frac{m-1}{m})^{k+1}] \tag{9}$$

式中，t_{max}表示允许的最长飞行时间，γ表示比例系数，m 为常数。

（3）混合粒子群算法

在对于粒子群算法参数以及位置速度更新公式的改进取得良好的成果之后，学者们发现现实中多数优化问题规模较大，约束条件趋于复杂，粒子群算法等启发式算法由于受自身结构与机制单一化限制难以对优化问题实现高效求解，鉴于该原因，大多学者开始将研究方向转向不同算法结合的混合优化策略。

Lovbjerg（2000）提出具有繁殖和子种群的混合粒子群算法(Hybrid Particle Swarm Optimization,HPSO)。上文已做出介绍，其中子代粒子所处位置与速度的公式：

$$child_1(x) = \rho_i \cdot parent_1(x) + (1 - \rho_i) \cdot parent_2(x) \tag{8}$$

$$child_2(x) = \rho_i \cdot parent_2(x) + (1 - \rho_i) \cdot parent_1(x) \tag{9}$$

$$child_1(v) = \frac{parent_1(v) + parent_2(v)}{|parent_1(v) + parent_2(v)|} |parent_1(v)| \tag{10}$$

$$child_2(v) = \frac{parent_1(v) + parent_2(v)}{|parent_1(v) + parent_2(v)|} |parent_2(v)| \tag{11}$$

其中ρ_i是$[0，1]$之间服从均匀分布的随机数，$parent_1(x)$、$parent_2(x)$以及$parent_1(v)$、$parent_2(v)$分别是父代粒子的位置向量和速度向量，$child_1(x)$、$child_2(x)$和$child_1(v)$、$child_2(v)$分别是子代粒子的位置向量和速度向量。改进后的算法在实现粒子群算法的基础上还使用了演化计算，增强了算法性能。

2.3 遗传算法描述

遗传算法与粒子群算法同样，算法思想都起源于自然界生物系统，遗传算法在迭代过程中基于上一代个体的信息进行交叉、变异等遗传操作演化得到适应度更佳的下一代。

2.3.1 遗传算法的基本流程

遗传算法的基本流程首先是对种群进行初始化并计算每个个体的适应度，若未达到停止条件就重复进行迭代，迭代过程通常是先从种群中选择优秀的个体，之后对每个优秀的个体进行杂交和变异操作产生新种群。

遗传算法流程如图 2.7 所示：

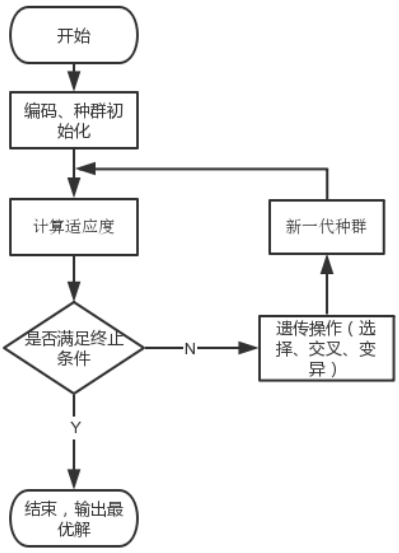

图 2.7 遗传算法流程图

遗传算法实现步骤如下：

Step 1：确定种群规模 N，按照设定的编码规则对初始种群X(0)进行编码，设 t=0。

Step 2：对X(t)中的每个个体适应度值进行计算。

Step 3：将适应度值从大到小进行排列，从X(t)中以种群规模数量 N 挑取

个体纳入进化种群$X'(t)$。

Step 4：依照设定的杂交与变异概率对群体$X'(t)$上进行进化，按照给定的规则挑取 N 个个体纳入下一代种群X(t)。

Step 5：判断迭代次数是否到达最大迭代次数，若到达则终止迭代，否则转至 Step 2。

2.3.2 遗传算法的编码设计

编码即采用便于计算机进行识别的方法表示问题的解，编码方式的选择很大一部分取决于问题的实质，同时对整个遗传过程有很深的作用。

基本的编码设计技术包括下述：

（1）二进制编码

二进制编码是通过某一设定的规则将实数转变成一串以 0 和 1 表示的字符。其中，字符串类似于种群进化的基因，首先将解进行二进制编码，在杂交和变异等操作结束后通过解码把结果恢复成问题的解后计算适应度值。比方（0，1，1，0，1，0）就是一串 6 个字符表示的二进制编码染色体。

（2）浮点数编码

浮点数编码利用浮点数表示问题的解，决策变量的数量即为基因的数量，染色体的长度就相当于决策变量的长度。浮点数即实数，相对二进制编码而言，浮点数精度更高，在变异操作上更具多样性，但搜索能力不如二进制码。

（3）格雷编码

格雷编码是一种特殊的二进制编码。设有二进制字符串α_1，α_2，…，α_n，其对应的格雷字符串表示为α_1'，α_2'，…，α_n'，从二进制编码转换为格雷编码的公式为：

$$\alpha_i' = \begin{cases} \alpha_1 & (i = 1) \\ \alpha_{i-1} \oplus \alpha_i & (i > 1) \end{cases} \tag{12}$$

式(12)中$\oplus$是异或操作，$\alpha_{i-1} \oplus \alpha_i$表示若$\alpha_{i-1}$的值若与$\alpha_i$不相同则为 1，否则为 0。相对于二进制编码而言，该编码方式增强了局部搜索能力。

（4）符号编码

符号编码是指通过只有代码意义而无数值意义的符号集合例如

$\{A，B，C，D\cdots\}$进行编码的方法。在使用符号编码时，不同的符号之间不能比较大小但是染色体中符号的顺序不同可以表示不同意义。该编码方法便于在遗传算法中与所求解问题的专业知识相结合，但对杂交、变异等遗传算子的设计要求较高。

2.3.3 遗传算法的初始种群

对于种群规模的选取，过大会增加算法的计算量，过小则会增大迭代次数，出现早熟现象，因此需要根据实际问题合理选择种群的规模。

2.3.4 遗传算法的杂交与变异

（1）杂交

以不确定的概率在种群内选取两个个体进行对位交换产生两个新个体。实际上，杂交就是两个个体以一个随机的概率对遗传信息的一部分进行互换。下面以二进制编码为例介绍常用的几种杂交方法。

①单点杂交

在两个个体的编码串中随机选取一个点位，对两个编码串点位之前或者是编码串点位之后的部分进行互换进而生成两个新个体。假设有个体X_1和个体X_2：

$$X_1\ 11010110 \qquad Y_1\ 11010101$$

$$\rightarrow$$

$$X_2\ 01100101 \qquad Y_2\ 01100110$$

随机选取杂交点位为 5，进行单点杂交后得到新个体Y_1和Y_2。

②两点杂交

两点杂交类似于单点杂交，区别在于两点杂交选取两个杂交点位，随后将两杂交点位之间的元素进行互换。假设有个体X_1和个体X_2：

$$X_1\ 11010110 \qquad Y_1\ 11100110$$

$$\rightarrow$$

$$X_2\ 01100101 \qquad Y_2\ 01010101$$

随机选取杂交点位为 2，6 进行两点杂交操作后得到新个体Y_1和Y_2。

③一致杂交

一致杂交以设置禁忌字符串的形式从一对随机父代个体当中按照顺序选择某个基因遗传至新个体内。当禁忌字符串中的禁忌字符为 0 时，子代个体相应位置Y_1遗传父代X_1 相应位置的基因；当禁忌字符串中的禁忌字符为 1 时，子代个体相应位置Y_1遗传父代X_2 相应位置的基因，通过以上规则执行产生子代新个体Y_1；反之若禁忌字符串中的禁忌字符为 1，子代个体相应位置Y_1遗传父代X_1 相应位置的基因；若禁忌字符串中的禁忌字符为 0，子代个体相应位置Y_1遗传父代X_2 相应位置的基因，通过以上规则执行产生子代新个体Y_2。如下为示例：

X_1 1 1 0 1 0 1 1 0　　Y_1 1 1 0 0 0 1 0 0

屏蔽字 0 1 0 1 0 1 1 0　→

X_2 0 1 1 0 0 1 0 1　　Y_2 0 1 1 1 0 1 1 1

（2）变异

变异是通过把染色体中单个或多个基因替代为其他基因进而形成新个体。对于二进制编码，变异即将某一基因的二进制值取相反值，即把 0 变为 1 或者把 1 变为 0。变异能够起到相当的克服早熟发生的作用，尽可能保证种群多样性不丧失。接下来通过二进制编码举例介绍几种常见的变异操作。

①基本变异

按某种概率在字符串中挑取单个或者是多个基因执行变异操作被称为基本变异：

X_1 1 0 0 1 0 1 1 0　→　Y_1 1 1 0 1 0 0 1 0

变异位置选择 2 和 6，进行基本变异后得到新个体Y_1。

②逆转变异

在字符串中任意挑选两点，两点之间被称为逆转区间，按照某种概率对逆转区间内的基因执行逆方向排列：

X_1 1 1 0 1 0 1 1 0　　Y_1 1 1 1 0 1 0 1 0

挑选 3 和 6 两点，则逆转区间为[3，6]，对个体X_1执行逆转变异后得到新个体Y_1。

2.3.5 遗传算法的种群选择

种群的选择决定父代种群中哪些个体能够被选来复制或遗传至下一代，模拟了生物进化的“优胜劣汰”、“适者生存”规律。种群选择方法受算法的杂交、变异、适应度值等因素的影响较小，在选择时具备通用性。常见的方法有以下几种：

（1）轮盘赌选择策略

计算新个体的适应度值，根据适应度值的大小分配个体的概率，根据概率方式选择个体作为下一代种群的父代。假设种群规模为 N，个体适应度值分别为f_i，$i=1$，2，$\cdots$，N。个体的概率分别是$p_i=\frac{f_i}{\sum_{j=1}^{N}f_j},i=1,2,\cdots,N$,对累计概率进行统计，$P_i=\sum_{j=1}^{i}p_j$，$i=1$，$2$，$\cdots$，$N$。产生均匀分布随机数$rand_j$，如果$P_{i-1}<rand_i\leq P_i$，则个体i被选择为下一代种群的个体。

（2）精英保留策略

顾名思义，精英保留策略就是对整个种群中最优秀的个体进行保护，使其安全顺利成为下一代种群中的一员。该选择策略保护本代种群中适应度值最高的个体不受遗传操作的干扰，其余个体在进行遗传操作之后将受保护的最优秀的个体替代经过遗传操作后适应度值最低的个体。对于最优秀个体的保护将加快算法的收敛速度。

2.3.6 遗传算法的特点

遗传算法是一种简单、容错性强且鲁棒性好的算法，其主要有以下特点：

（1）与其他优化算法相比，遗传算法是在整个解空间寻优搜索，搜索范围较大，因此可以有效避免陷入局部最优，具备良好的全局搜索能力。

（2）遗传算法容错能力极佳，初始解的优劣对其产生的影响较小。对于与最优解差距甚远的初始解，仍可以通过选择、杂交、变异操作迅速剔除。

遗传算法同时也存在一些不足之处：

（1）遗传算法局部搜索能力较弱，容易导致其在后期收敛速度缓慢，甚至无法收敛至全局最优解。

（2）无方向性。遗传操作中个体都是独立的，个体之间并不共享信息，仅仅依靠选择操作筛选出部分优质个体而并不能对群体中所有个体的质量做出保障。变异以及杂交操作可能会使个体的适应度值降低，由此减慢了算法的收敛速度。

2.3.7 遗传算法的改进策略

随着对遗传算法的不断深入研究，对于基本遗传算法在求解过程中遇到的问题以及不足，学者们提出了不同的改进策略，改进策略多是围绕遗传算法参数的选择、编码方式以及遗传算子展开。

（1）参数的选择

遗传算法中的参数包含种群规模 S、迭代次数 T、杂交概率P_c以及变异概率P_m。参数的不同设置对于遗传算法的性能影响较大，众多学者为了取得合适的遗传算法参数设置进行了大量的研究，其中自适应参数设置法占据大多篇幅，自适应参数设置法考虑的是对于遗传算法在求解某问题的不同阶段如何对某些参数进行自动调节。这种自适应参数遗传算法相比基本遗传算法具有更高的鲁棒性及效率。

①种群规模。种群规模对于遗传算法所求得最优解的好与坏有重要影响。种群规模过小，算法的有效性将大打折扣；种群规模过大，虽然可以降低遗传算法陷入局部最优解的概率从而提高获取全局最优解的概率，但也会因此拉长算法的求解时间，导致算法求解效率下降。因此对于需要求解的问题，种群规模的选取极其重要，通常在车辆路径问题上，种群规模取值在客户数的 2 至 3 倍较合适。

②杂交概率与变异概率。杂交与变异影响着遗传算法的方向性与随机性。方向性提高意味着遗传算法在区域内搜索的能力提高，随机性提高意味着遗传算法开辟新搜索区域的能力提高，算法的性能由这两者所决定，这两者又是由杂交概率P_c以及变异概率P_m所决定。杂交概率P_c决定进行杂交操作的概率，较高的杂交概率会提高算法搜索新区域的概率，但过高的杂交概率也意味着算法无法对当前区域进行细致搜索；较低的杂交概率则会使算法停滞在当前搜索区域。变异是遗传算法中的辅助操作，目的是维持种群的多样性，较

低的变异概率可以降低种群中优良基因的丢失概率，同时又提供算法搜索新区域的可能性；而较高的变异概率会导致算法纯粹的进行随机搜索。基本遗传算法中杂交概率与变异概率是固定不变的，一般来说，杂交概率取值在 0.25 至 1 之间，变异概率的取值通常在 0.05 左右。在自适应杂交概率与变异概率方面，学者们普遍设置函数使得杂交概率与变异概率在算法初期较大，提高算法的搜索概率，而在进入算法后期时减小杂交概率与变异概率，由此将种群中的优良基因尽可能保留下来。

（2）编码方式

在进行遗传算法应用前，首先需要实施的工作就是对编码方式的确定，具体采取哪种编码方式与需要解决的问题相关，对于不同的问题编码方式也有所区别。编码方式决定了遗传算法是否可以有效合理地对所求解问题进行描述。与问题契合的编码方式可以缩短编码长度，提高算法的性能。

为提高遗传算法的性能，学者们提出了不同的改进编码方式，如变种群规模方式，其思想是在算法初期使用短编码长度以及大规模的种群，使算法迅速搜寻到优质解所在区域，对初期算法的效率有很大提升。在算法后期增加编码长度并缩减种群规模，提高算法的收敛精度。

（3）杂交算子

杂交算子在遗传算法中具有重要地位，常见的杂交方式包括单点杂交、多点杂交、混合杂交、顺序杂交等。常见的杂交方式进行杂交操作是无方向性的，由此学者们针对不同问题提出了不同的杂交算子改进方法，例如在求解车辆路径问题上，刘敏（2006）设计了一种提高适应度的路径杂交算子，增强了算法的性能。

2.4 本章小结

本章对相关基础理论进行分析，详尽叙述了车辆路径问题的定义、基础理论和求解方法，由求解方法引出粒子群算法与遗传算法，并对两者的基本概念、基本原理、特点、步骤与流程以及改进策略等进行描述。此章节为本文研究问题、模型建立以及算法改进建立了基础。

3.带时间窗的多车场车辆路径问题模型与求解研究

3.1 带时间窗的多车场车辆路径问题模型分析

3.1.1 问题描述

在基本的车辆路径问题上添加约束条件进行延申便出现带时间窗的多车场车辆路径问题，其内容相比基本的车辆路径问题增加了三个方面：第一个方面是恰当地对多个车场与客户完成匹配，第二个方面是能够同时满足尽量多的不同客户的配送时间要求，第三个方面是多车场车辆路径问题规模更大，所以求解时间也更长，为使模型达到更加贴合实际的目的，将在带时间窗的多车场车辆路径问题中对多车型配送进行考虑，同时为更大程度减少配送成本，将对每个车场的尾单即最后一辆配送车辆所运输的货物，根据该车所需行驶里程考虑是否采取骑手配送。

因此如何合理对高复杂度问题进行求解并提高求解效率成为需要重点关注的问题，即在达到最大的车辆装载率以及满足车辆里程限制的情况下，如何规划出尽量满足客户要求的服务时间范围的若干条车辆配送路径使得配送总成本最低。

3.1.2 模型假设

根据上节问题描述，首先需要对模型做以下基本假设：

（1）所有车场位置以及各车场的车辆数已明确，所有客户的位置以及需求量已明确。

（2）各个车场货源量充沛，不存在缺货、少货情况的发生。

（3）配送车辆车型存在多种，车辆的参数例如最大行驶里程、最大载重量、行驶的平均速度以及启动成本因车型不同而有所不同。

（4）不考虑所装载货物的大小和形状，只考虑货物重量，且货物重量统一。

（5）不考虑在配送路程中发生天气影响、道路拥堵等特别情况的发生。

（6）骑手的费用只考虑固定的启动成本，不考虑配送距离成本以及最大里程限制。

（7）各个客户所要求的时间窗已知，若车辆在客户要求的最早送达时间前到达客户点，则需要等候至最早服务时间才可开启服务，同时会根据等候时长产生一定的机会成本；若车辆晚于客户要求的最晚送达时间送达，则会根据迟到时间产生一定的惩罚成本。

3.1.3 参数说明

构造数学模型涉及符号和变量等参数的定义，如下列表 3.1 与表 3.2 所示：

表 3.1 主要符号定义说明

符号	定义
f	数学模型总目标函数
N	客户总数目
M	车场总数目
R_m	车场 m 拥有车型数
S_m	车场 m 拥有骑手数目
K_{mr}	车场 m 拥有的车型 r 数目（$r=1，\cdots，R_m$）
q_r	车型 r 的最大载重量
D_r	车型 r 的行驶里程限制
v_r	车型 r 的平均行驶速度
C_r	车型 r 的启动成本
q_s	骑手的最大载重量
C_s	骑手的启动成本
g_i	客户 i 的需求量（$i=1，2，\cdots，N$），且$g_i \le q$
d_{ij}	从客户 i 到客户 j 的距离
ET_i	车辆最早到达客户 i 的时间
LT_i	车辆最晚到达客户 i 的时间
t_{ij}	车辆从客户 i 到客户 j 所需时间

T_i	到达客户 i 的时间
ST_i	为客户 i 提供服务的所需时间
P_a	车辆在ET_i之前到达客户 i 每单位时间所产生的等待成本
P_b	车辆在LT_i之后到达客户 i 每单位时间所产生的处罚成本

表 3.2 变量定义说明

变量	定义
x_{ij}^{mkr}	$x_{ij}^{mkr}=\begin{cases}1，\text{由车场}m\text{的车型}r\text{的第}k\text{辆车从客户}i\text{配送至客户}j\\0，\text{否则}\end{cases}$；
y_{ij}^{ms}	$y_{ij}^{ms}=\begin{cases}1，\text{由车场}m\text{的第}s\text{个骑手从客户}i\text{配送至客户}j\\0，\text{否则}\end{cases}$；

3.1.4 约束条件

在构建数学模型的过程中，通常需要添加一些约束条件来确保模型所要求的解的可行性。在车辆路径问题上，需要考虑的约束条件来自于现实当中车辆安排调度的具体情况，例如配送车辆的最大里程和载重量限制、车场封闭或是开放等。本文研究的模型必须满足以下几个约束条件：

（1）每个客户点必须仅用一辆车或一个骑手完成配送服务；

（2）每辆配送车辆或骑手装载的货物重量总和不得大于车型要求的最大装载量限制；

（3）每辆配送车辆运输总距离严格小于车型要求的最大行驶里程数限制；

（4）每辆配送车辆的起点以及终点必须是车场，且起点以及终点为同一车场。

（5）每个骑手的起点必须是车场，但终点为某一客户。

3.2 带时间窗的多车场车辆路径问题模型构建

$$\mathrm{f}=\min\sum_{\mathrm{i=1}}^{\mathrm{N+M}}\sum_{\mathrm{j=1}}^{\mathrm{N+M}}\sum_{\mathrm{m=1}}^{\mathrm{M}}\sum_{r=1}^{R_m}\sum_{k=1}^{K_{mr}}d_{ij}x_{ij}^{mkr}+\sum_{\mathrm{i=N+1}}^{\mathrm{N+M}}\sum_{\mathrm{j=1}}^{\mathrm{N}}\sum_{m=1}^{M}\sum_{r=1}^{R_m}\sum_{\mathrm{k=1}}^{K_{mr}}x_{ij}^{mkr}*C_r$$
$$+\sum_{m=1}^{M}S_m*C_s+\sum_{\mathrm{i=0}}^{\mathrm{N}}\mathrm{P_a}*\max\left((ET_i-T_i),0\right)+\sum_{\mathrm{i=0}}^{\mathrm{N}}\mathrm{P_b}*\max\left((T_i-LT_i),0\right)\quad(1)$$

$$\sum_{j=1}^{N}\sum_{k=1}^{K_{mr}} x_{ij}^{mkr} \le K_{mr},\ i=m\in\{N+1,\ N+2,\ \cdots,\ N+M\}$$

$$,\ r\in\{1,\ \cdots,\ R_m\} \tag{2}$$

$$\sum_{j=1}^{N}\sum_{k=1}^{K_{mr}} x_{ij}^{mkr} = \sum_{j=1}^{N}\sum_{k=1}^{K_{mr}} x_{ij}^{mkr} \le 1,\ i=m\in\{N+1,\ N+2,\ \cdots,\ N+M\}$$

$$,\ r\in\{1,\ \cdots,\ R_m\} \tag{3}$$

$$\sum_{\mathrm{i=1}}^{\mathrm{N+M}}\Big(\sum_{\mathrm{m=1}}^{\mathrm{M}}\sum_{\mathrm{k=1}}^{K_{mr}} x_{ij}^{mkr} + \sum_{s=1}^{S_m} y_{ij}^{ms}\Big) = 1, j\in\{1,\ 2,\ \cdots,\ N\},\ r\in\{1,\ \cdots,\ R_m\} \tag{4}$$

$$\sum_{i=1}^{N} g_i \sum_{j=1}^{N+M}\sum_{k=1}^{K_{mr}} x_{ij}^{mkr} \le q_r,\ m\in\{N+1,\ N+2,\ \cdots,\ N+M\}$$

$$,\ r\in\{1,\ \cdots,\ R_m\} \tag{5}$$

$$\sum_{i=1}^{N} g_i \sum_{j=1}^{N+M}\sum_{s=1}^{S_m} y_{ij}^{ms} \le q_s,\ m\in\{N+1,\ N+2,\ \cdots,\ N+M\} \tag{6}$$

$$\sum_{i=1}^{N+M}\sum_{j=1}^{N+M}\sum_{\mathrm{k=1}}^{K_{mr}} d_{ij}x_{ij}^{mkr} \le D_r, m\in\{N+1,\ N+2,\ \cdots,\ N+M\}$$

$$,\ r\in\{1,\ \cdots,\ R_m\} \tag{7}$$

$$\sum_{j=N+1}^{N+M}\sum_{k=1}^{K_{mr}} x_{ij}^{mkr} = \sum_{j=N+1}^{N+M}\sum_{k=1}^{K_{mr}} x_{ij}^{mkr} = 0, i=m\in\{N+1,\ N+2,\ \cdots,\ N+M\}$$

$$,\ r\in\{1,\ \cdots,\ R_m\} \tag{8}$$

$$T_j = T_i + ST_i + t_{ij},\ i,\ j\in\{1,\ 2,\ \cdots N+M\} \tag{9}$$

$$t_{ij} = {}^{d_{ij}}/_{v_r}, i,\ j\in\{1,\ 2,\ \cdots N+M\},\ r\in\{1,\ \cdots,\ R_m\} \tag{10}$$

公式（1）为距离成本、车辆启动成本、骑手启动成本、等待成本与处罚成本之和，保证了费用最小化；公式（2）限制了从某一个车场出发的某配送车型总数量小于等于K_{mr}；公式（3）确保每辆配送车辆由一个车场发出并在配送任务完成后返回至该车场；公式（4）确保每个客户点仅接受单独一辆配送车辆或骑手完成一次配送；公式（5）确保每辆配送车辆所装货物重量小于其最大载重量限制；公式（6）确保每个骑手所装货物重量小于其最大载重量

限制；公式（7）保证每辆配送车辆（不包括骑手）行驶路程小于其最大行驶里程限制；公式（8）确保配送车辆的配送路径上不会出现从一个车场行驶至另一个车场的情况。公式（9）表示路径中连续两节点 i，j 之间的车辆抵达时间联系。公式（10）表示路径中连续两节点 i，j 之间的运输时间运算公式，由两节点 i，j 之间的距离除以车辆的平均行驶速度得到。

3.3 带时间窗的多车场车辆路径问题求解分析

学者们对于求解带时间窗的多车场车辆路径问题所使用的方法，主要包括全局优化法与分解法。

全局优化法即将全部车场看作一个整体，联合多个车场同步进行优化，在对车场进行配置的同时对车辆进行配送路径的构建。该法优点在于是对整个问题从多方面同时进行优化，求解方向始终趋向全局最优解。缺点则是问题的规模并未得到有效降低，优化时间将非常长。

分解法即把求解多车场问题转化成求解多个单车场问题。通常包含两阶段，首先通过为每个车场分配对应客户，直至所有客户都已确定所属车场，再对每个车场的现有客户如何安排车辆进行有次序的配送进行求解最终获得最优解，该方法具有降低问题求解规模以及复杂度的优点，但若前期客户分配不合理则会导致后期可能无法得到优良的解。分解法的关键是使用何种方法对多车场问题拆分成为若干个单车场问题，拆分的方法有节约算法、客户聚类法等。客户聚类法是一种将一组无标记对象进行划分得到多个群体的机器学习算法。群体内部的个体相似性尽量大，群体之间的相似性尽量小，在该问题中客户聚类法通过各客户到各个车场的距离大小建立客户车场对，随即完成客户分配。

本文对于求解带时间窗的多车场车辆路径问题将结合上文分解法以及全局优化法，在阶段一采用分解法，利用 K-means 聚类算法对客户分配方案进行优化生成初始客户分配方案，采用遗传算法对车辆配送路径进行优化生成初始车辆配送路径最终得到初始解；在阶段二则采用全局优化法对初始解进行优化，鉴于目前学者们对于车辆路径问题多采用混合算法进行求解，本文积极借鉴前人研究经验，采用两种启发式算法混合对问题进行求解，在算法

选择上，本文利用粒子群算法求解速度快以及遗传算法灵活、无方向性易于搜索全局最优解的特点将两者进行结合。通过改进的遗传粒子群混合算法对优化后的初始解进行迭代，多车场联合进行优化以求得全局最优解。此方法不仅减少问题的求解规模以及优化时间，同时有利于找到全局最优解。

3.4 本章小结

本章最开始介绍描述了带时间窗的多车场车辆路径问题，为使模型更加贴合现实情况，在该问题上考虑了多车型以及尾单配送，并结合该问题建立了以配送总成本最小为目标函数的数学模型，积极借鉴前人研究经验，对该模型的求解同时结合了分解法与全局优化法，降低了问题规模并提高了计算精度，并确定了以粒子群算法与遗传算法结合作为本篇论文模型求解的混合算法。

4. 混合算法求解带时间窗的多车场车辆路径问题

4.1 混合算法选择策略

本文对于求解带时间窗的多车场车辆路径问题采用分解法与全局优化法的思想，为提高运算和求解效率，提出两阶段求解方法，降低运算复杂度，缩短运算时间提高效率。

阶段一采用 K-means 聚类法对所有客户进行分类并完成车场分配，再通过遗传算法优化配送路径得到初始解。

阶段二采用改进的遗传粒子群混合算法对初始解进行优化最终得到最优解。对于传统的粒子群算法，处于种群最优位置的粒子单向发送信息至其他粒子，除此之外各个个体独立搜索并不分享其他信息，这样使得算法全程都是在当前迭代次数的最优解的引导下搜索，容易导致种群丧失多样性，算法收敛速度快，容易陷入局部最优；而遗传算法能够有效避免该情况的发生,所有个体在迭代的过程中进行搜索，其求解过程具备无方向性，是整个种群多个点同时开始搜索，不是从单点进行搜索，不论搜索域是单峰还是多峰分布，整个群体都在均匀地向最优区域靠拢，能够有效寻找到最优解，同时遗传算法根据随机概率来进行选择、杂交、变异，通过不确定的随机概率来持续对搜索方向进行调节，这种概率性特点使得算法能够较为灵活地进行搜索。基于粒子群算法求解速度快、容易陷入局部最优的特点，将粒子群算法与遗传算法相结合，利用遗传算法灵活、无方向性易于搜索全局最优解的特点进行算法的优化。

4.2K-means 聚类算法

在对现今的文献进行整理分析后，结合本篇论文所需要研究的问题，本篇论文对所有客户的位置进行收集，首先在第一阶段采取 K-means 聚类算法对客户进行聚类，将各个客户群体与最近车场进行匹配，得到初始的客户分配。首先需要确定客户群体的分类数目，而所需要分为多少个客户群体由有多少个车场所决定，之后对数据进行划分并在经过最初设置的迭代次数之后输出最优的聚类方案。根据第三章提出的问题，配送网络中包括 M 个车场以及 N 个客户，采取 K-means 聚类算法完成对客户群体的划分，K-means 聚类算法流程如下。

Step 1：完成对 N 个客户的定位。

Step 2：确定 M 个最初的聚类中心，聚类中心在 N 个客户中任意选取，且$M \leq N$。

Step 3：对任意客户点，分别求其至 M 个聚类中心的距离，把所有客户点归类至距离其最近的聚类中心从属的分类。

Step 4：根据划分完成的客户群体再次计算确定 M 个新的聚类中心。

Step 5：返回 Step3 直至所有聚类中心的位置不再更新时，终止迭代。

Step 6：将 M 个车场根据距离其最近的聚类中心原则划分至 M 个聚类中心，且需保证车场与聚类中心一一对应，随后完成划分输出聚类结果。

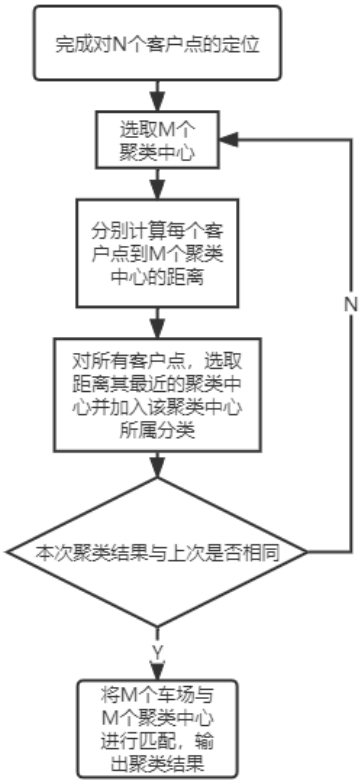

图 4.1 K-means 聚类算法流程图

4.3 改进的遗传粒子群混合算法

4.3.1 改进的遗传粒子群混合算法求解设计

本文在阶段二采用两种启发式算法混合的全局优化法多初始解进行优化。由于粒子群算法全程都是在当前迭代次数的最优解的引导下搜索，容易导致种群丧失多样性从而陷入局部最优，难以获得全局最优解。因此本篇论文为克服粒子群算法这一缺点，利用遗传算法具备良好的全局搜索能力这一特性，在迭代的过程中引入遗传算法，判断是否陷入局部最优解，若搜索陷入停滞则引入遗传算法促使各个粒子跳出局部最优解再次进行搜索，增大算法探索到更优解区域的可能性。对这两种算法的混合使用能够有效增强发现全局最优解的能力。

（1）陷入局部最优判定

基本的遗传粒子群混合算法通常以全局最优解变化停滞代数超过某一阈值判定粒子陷入局部最优解，而该判定方法具有一定的局限性。本文引入吕振肃、侯志荣（2004）提出的适应度方差识别粒子陷入局部最优。

最优粒子群随着算法的推进不断进化，个体之间的差异不断缩小。个体的适应度大小由个体的位置所决定，因此根据粒子群整体适应度变化可以判断种群目前所处状态。

参照吕振肃等（2004）的定义，设粒子群的粒子数量为n，个体i的适应度为f_i，f_{avg}为当前代数种群的平均适应度，σ^2为粒子群的群体适应度方差，则σ^2可定义为：

$$\sigma^2 = \sum_{i=1}^{n}\left(\frac{f_i - f_{avg}}{f}\right)^2 \tag{1}$$

其中f定义为归一化定标因子，它的作用是为了限制σ^2的大小，一旦σ^2小于某一阈值C_{lo}时则判定粒子群陷入局部最优，f的取值采取如下公式：

$$f = \begin{cases} max_{1\le i\le N}|f_i - f_{avg}|, max|f_i - f_{avg}| > 1 \\ 1, else \end{cases} \tag{2}$$

式（1）说明：群体适应度方差反映的是粒子群中个体之间的密集程度，σ^2越小，代表粒子群中个体的密集程度越大；σ^2越大，代表粒子群中个体的密集程度越小，个体间越分散。随着迭代次数的增加，粒子群中个体的适应度将越来越接近，但文献对于阈值C_{lo}的合理取值并没有明确标准，根据董勇、郭海敏（2011）进一步的数值实验研究发现：当陷入局部最优解时，连续的多次迭代中群体适应度方差σ^2变化极小。若群体适应度方差σ^2在相邻迭代中变化很小，则可以说明粒子很可能陷入局部最优解，此时需要对当前粒子群进行变化以跳出局部最优值。

本篇论文利用相邻两次迭代中群体适应度方差σ^2的变化范围来判定粒子群是否陷入局部最优解。若相邻迭代的群体适应度方差σ^2的变化小于某一定值，就说明粒子群陷入局部最优，需要引入遗传算法对密集分布的粒子进行打散。

（2）编码方案和求解方案

编码是算法与所求解问题的重要联接，针对不同类型的问题编码方式也随之不同。本篇论文需要让多车场车辆路径问题中的可行解与粒子群算法中

的粒子一一对应，在对前人相关研究文献进行整理分析后，本文决定采用双层编码方式，第一层粒子与第二层粒子长度一致都是 N 维向量，其中 N 表示客户数目。第一层粒子代表客户与车场的分配方案，方案的取值范围由车场数 M 决定；第二层粒子代表对客户进行配送的优先级，客户被车辆服务的先后顺序按照优先级值由小到大排列，即优先级值越小，越早接受配送服务；反之优先级值越大，越晚接受配送服务。随后将车场的车辆按照最大行驶里程限制从小到大排列，若车辆行驶里程限制低于车场的总配送里程不超过 50 公里（默认骑手行驶里程），则将该车辆纳入出车方案中，否则比较下一辆车，若都不满足则选用行驶里程限制最高的车并更新总配送里程，再从第一辆车去比较，直至满足车场的总配送里程，由此得到初始的出车方案。例如，假设 M=3,N=8，每个车场分别有 2 辆车型不一的车标号为[1 2 3 4 5 6]，第一层粒子为[1 2 3 1 3 2 3 2]，第二层粒子为[4 1 6 2 8 3 7 5]，第一层粒子表示客户 1 与客户 4 由车场 1 提供服务，客户 2、客户 6、客户 8 由车场 2 提供服务，客户 3、客户 5、客户 8 由车场 3 提供服务；第二层粒子则在第一层粒子的基础上进行配送，根据第二层粒子解读车场 1 的配送路径为客户 4-客户 1，车场 2 配送路径为客户 2-客户 6-客户 8，车场 3 对应的配送路径为客户 3-客户 7-客户 5。假设初始的出车方案为[1 3 4 5]，若客户 3、客户 7 满足车场 3 车辆 5 的最大载重量限制，而加上客户 5 会超过最大载重量限制，则会派出骑手对客户 5 进行尾单配送，因此车场 3 中车辆 5 的配送路径为车场 3-客户 3-客户 7-车场 3，以及骑手的配送路径为车场 3-客户 5。

在粒子群算法中对于解的变换方式，许伊萍（2015）提出使用交换子以及交换序列对粒子群算法进行解的变换用以求解旅行商问题，本篇论文从中引入交换子以及交换序列的概念以对本文的问题进行求解。交换子定义为$s = Swap（i，j）$，表示交换子 s 使粒子中位置 i 上的元素与位置 j 上的元素相互交换。如粒子为[4 1 6 2 8 3 7 5]，对其进行一个交换子操作$s = Swap（3，4）$，其新的粒子为[4 1 2 6 8 3 7 5]。在粒子群算法中，粒子的速度可以通过交换子来表示，其作用为改变 x 的位置。

交换序列定义为一组有先后顺序的交换子的集合，即$ss = [Swap_1, Swap_2, \cdots]$，表示对粒子进行连续多次交换操作。如粒子为

[4 1 6 2 8 3 7 5]，交换序列为$ss = [(3，6),(1，7)]$，则粒子经过交换序列 ss 的作用后，新的粒子为[7 1 3 2 8 6 4 5]。

解释了交换子与交换序列，然后重新对粒子的速度公式进行定义并更新，具体公式如下。

$$v_{ij}^{(k+1)} = v_{ij}^{k} \oplus r_1(p_{ij}^{k} - x_{ij}^{k}) \oplus r_2(p_{gj}^{k} - x_{ij}^{k}) \quad (3)$$

$$x_{ij}^{(k+1)} = x_{ij}^{k} + v_{ij}^{(k+1)} \quad (4)$$

式中，$p_{ij}^{k} - x_{ij}^{k}$表示有一个交换序列 ss，该交换序列 ss 使x_{ij}^{k}在通过 ss 变换后得到p_{ij}^{k}，同理$p_{gj}^{k} - x_{ij}^{k}$也是一个交换序列 ss；符号⊕表示两个交换序列合并为一个交换序列，如$[(3，6),(1，7)] \oplus [(4，8)] = [(3，6),(1，7),(4，8)]$；符号+表示对一个粒子按照交换序列 ss 执行交换操作，因此，位置更新公式$x_{ij}^{(k+1)} = x_{ij}^{k} + v_{ij}^{(k+1)}$，表示$x_{ij}^{k}$经过交换序列$v_{ij}^{(k+1)}$作用后得到新的粒子$x_{ij}^{(k+1)}$；$r_1$和$r_2$为 0-1 之间的随机数，表示执行交换操作的可能性大小，意在随机放弃部分交换子。通过式（3）（4）进行粒子位置以及速度的更新无需对得到的结果转为实数，使得算法更好地应用于车辆路径问题。

因此在每次迭代中，粒子根据式（3）（4）更新位置，同时对所有更新后的粒子进行检查，若该粒子是不可行解，则取消该粒子此次位置的更新。之后计算种群中所有粒子的适应度，若个体更新后的适应度优于更新前的适应度，则用更新后得到的解与适应度替换更新前的解与适应度，并更新个体的最优适应度，否则不变；同时对更新前的群体最优适应度与更新后的群体最优适应度进行比较，若更新后的群体最优适应度优于更新前的群体最优适应度，则用更新后得到的群体最优适应度与解替换更新前得到的群体最优适应度与解。

（3）初始种群

初始种群是之后迭代的基础，若种群规模过小，迭代中易陷入局部最优；若种群规模过大，会增加算法运算的难度，降低计算速度，使得求解效率低下。因此选择恰当数量的种群尤为重要。本研究一共由 48 位客户以及 3 个配送中心构成，将种群规模设置为 100。

初始种群中的客户分配方案即第一层粒子已经由一阶段的聚类得出；对于配送路径即第二层粒子，本文首先通过从对$[0，N]$中的数进行随机排列得到不同排列组合从而生成配送路径，而后引入遗传算法通过选择、杂交、变异操作对配送路径以及分配方案进行迭代优化，经过优化的双层粒子便形成初始种群。

（4）适应度函数

适应度函数是判断粒子优劣的依据，本篇论文以第三章中的带时间窗的多车场车辆路径问题模型中的目标函数作为本文算法的适应度函数。本文的目标函数是从配送总成本出发，目标函数越小，适应度越大；反之适应度越小，最佳路线就是目标函数最小的配送路径。

（5）选择算子

遗传算法的选择依据围绕各粒子的适应度大小，通过对所有粒子的适应度进行计算比较，选择适应度值大的粒子代替适应度值小的粒子来进行迭代。随着迭代，将越来越接近最优解。本文采用轮盘赌选择策略，计算出种群中每个个体的适应度大小，并对各个适应度进行归一化计算，得到的归一化值即为粒子被选择的概率，本篇论文的适应度函数即目标函数，目标函数越小，适应度越大，被选择的概率越高。

（6）杂交算子

遗传算法的杂交是指按照一定杂交概率选择杂交点，交换杂交点左或右的片段产生不同的粒子，从而产生新种群。本文以单点杂交作为杂交算子对模型进行求解，采取的自适应杂交概率公式如下：

$$P_c=\begin{cases}P_{cmax}\left[\omega_{PSO}\cdot\cos\left(\frac{\pi}{2}\cdot\frac{t_{PSO}}{T_{PSO}}\right)+\omega_{GA}\cdot\cos\left(\frac{\pi}{2}\cdot\frac{t_{GA}}{T_{GA}}\right)\right], & P_c>P_{cmin}\\ P_{cmin,} & P_c\le P_{cmin}\end{cases} \tag{5}$$

式中，P_c为初始预设的杂交概率，并且在迭代的过程中不断下降，P_{cmax}为初始预设的最大杂交概率，P_{cmin}为初始预设的最小杂交概率，t_{PSO}为当前粒子群代数，T_{PSO}为最大粒子群代数，同理可得t_{GA}为当前遗传代数，T_{GA}为最大遗传代数，ω_{PSO}与ω_{GA}分别为粒子群算法与遗传算法的权重，两者总和为 1。该自适应杂交概率可以确保在遗传算法与粒子群算法在迭代初期杂交概率较大并且保持缓慢的下降速度，从而可以较大概率对种群造成扰动，增强算法

的搜索能力，加快种群进化速度；伴随迭代次数增加，杂交概率逐步降低，最后维持在一个常量，从而可以确保优秀粒子的稳定，避免造成破坏，同时收敛速度得到加快，提高找到全局最优解的可能性。

将父代种群根据杂交概率选择任意且不重复的两个粒子中的全部或部分客户分配方案互换完成杂交操作，重复该操作直至种群所有粒子均完成杂交，得到新的种群，若杂交后的个体为不可行解，则取消个体此次杂交操作。

（7）变异算子

遗传算法的变异是对粒子的任一点用其他的点进行替换产生不同于之前的粒子。本文选取基本变异算子作为变异方法，同时为客户分配方案与配送路径设置了不同的变异策略分别命名为基本变异一和基本变异二，其变异概率同样采取与杂交概率相同的自适应变异概率公式如下所示：

$$P_{m1}=\begin{cases}P_{m1max}\left[\omega_{PSO}\cdot\cos\left(\frac{\pi}{2}\cdot\frac{t_{PSO}}{T_{PSO}}\right)+\omega_{GA}\cdot\cos\left(\frac{\pi}{2}\cdot\frac{t_{GA}}{T_{GA}}\right)\right], P_{m1}>P_{m1min}\\ P_{m1min}, \quad P_{m1}\le P_{m1min}\end{cases} \tag{6}$$

$$P_{m2}=\begin{cases}P_{m2max}\left[\omega_{PSO}\cdot\cos\left(\frac{\pi}{2}\cdot\frac{t_{PSO}}{T_{PSO}}\right)+\omega_{GA}\cdot\cos\left(\frac{\pi}{2}\cdot\frac{t_{GA}}{T_{GA}}\right)\right], P_{m2}>P_{m2min}\\ P_{m2min}, \quad P_{m2}\le P_{m2min}\end{cases} \tag{7}$$

式（6）与式（7）中各符号意义与式（5）同理。对于客户分配方案即每第一层粒子采取基本变异一，随机获取变异点并以基本变异一概率进行变异操作，重复该操作直至种群所有粒子均完成变异；对于配送路径即第二层粒子采取基本变异二，由于对该粒子片段进行直接的基本变异可能造成变异后的个体不合法，因此本文对于基本变异二的策略是随机选取该片段两个变异点以基本变异二概率进行交换完成变异操作，重复该操作直至种群所有粒子均完成变异。两种变异完成之后即得到新的种群，若变异后的个体为不可行解，则取消个体此次变异操作。

4.3.2 改进的遗传粒子群混合算法求解流程

通过 K-means 聚类算法以及利用选择、杂交、变异算子搜索解的遗传算法得到的初始解，采用粒子群算法对解进行迭代优化，当陷入局部最优时再次引入遗传算法跳出局部最优，若达到种群的迭代次数后结束求解，输出最

优解，改进的遗传粒子群混合算法流程如图 4.2 所示：

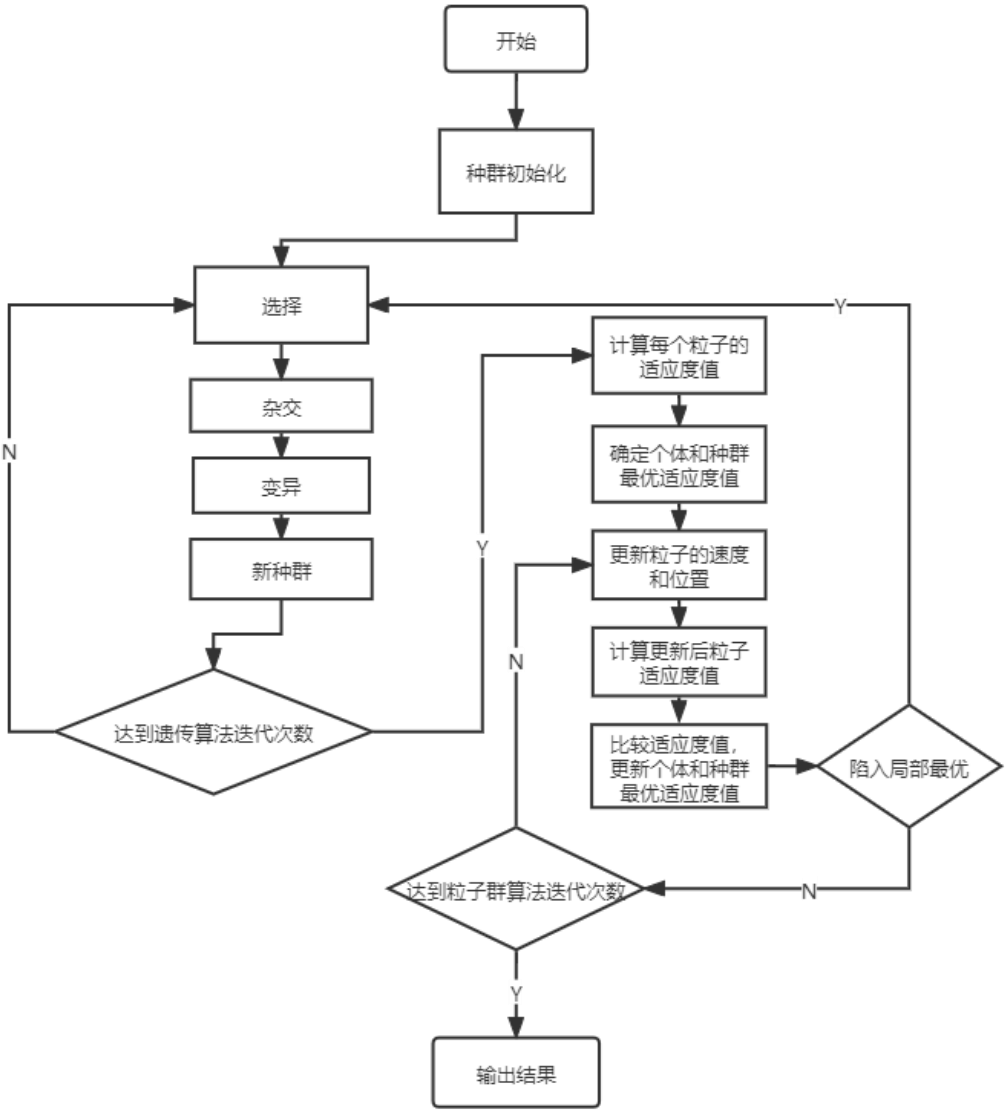

图 4.2 改进的遗传粒子群混合算法流程图

改进的遗传粒子群混合算法分为以下六个步骤。

Step 1：首先选择适合的编码方式对所求解的实际问题数据进行编码，进行编码时要与实际问题结合考虑以提高计算机的运算效率。

Step 2：分别设置遗传算法和粒子群算法的迭代次数，确定问题的适应度函数及目标函数。通过 K-means 聚类算法获得初始客户分配方案、随机生成配送路径并通过遗传算法选择、杂交、变异进行优化获得初始配送路径方案，

由此形成初始种群。

Step 3：根据 Step2 设置的适应度函数，计算所有粒子的适应度值并确定个体和种群的最优适应度值。

Step 4：更新粒子的速度和位置并计算更新后粒子的适应度值，对更新前粒子的适应度值与更新后粒子的适应度值进行比较，若后者优于前者，个体最优适应度值将变更为更新后粒子适应度值，否则不变；同理对种群最优适应度值进行更新。若种群的适应度方差在相邻两次迭代的变化低于预先设定的阈值，则陷入局部最优，通过遗传算法选择、杂交、变异进行优化跳出局部最优，得到新种群，各个粒子最优适应度值为当前新种群中粒子最优适应度值。

Step 5：重复 Step3 与 Step4 直到迭代次数满足终止条件，停止迭代。

Step 6：种群停止迭代，输出最终结果。

4.4 改进的遗传粒子群混合算法求解算例分析

本节通过引入标准算例对算法可行性进行验证，分析算法的有效性并与其他算法进行对比分析。本节程序使用 Python3.8 编写，在内存为 16G，CPU 为 AMD Ryzen 5 4600U with Radeon Graphics 2.10 GHz，Windows 10 系统上运行程序。

4.4.1 算例设计

本节算例取自 VRP 问题研究网站 https://www.bernabe.dorronsoro.es/vrp/中 MDVRPTW 内 20 组算例中的第一组算例。算例中包括 3 个车场以及 48 个客户，客户以及车场的坐标、车辆的最大载重量、最大续航里程、需求、时间窗以及服务时间均已设定，本算例的配送网络为适应多车型配送模型，从原本由 3 个拥有同一批车型的车场组成且各拥有 3 辆车修改为由 3 个拥有不同车型的车场组成且各拥有 3 辆车，48 个客户的时间窗以及服务时间各不相同，单位早到惩罚成本$p_a = 5$，单位晚到惩罚成本$p_b = 8$，车辆需要在客户时间窗内进行配送，如果未在规定的时间窗内送达，将按照违约时长产生对应的惩

罚成本。算例中客户以及车场分布如图 4.3 所示。

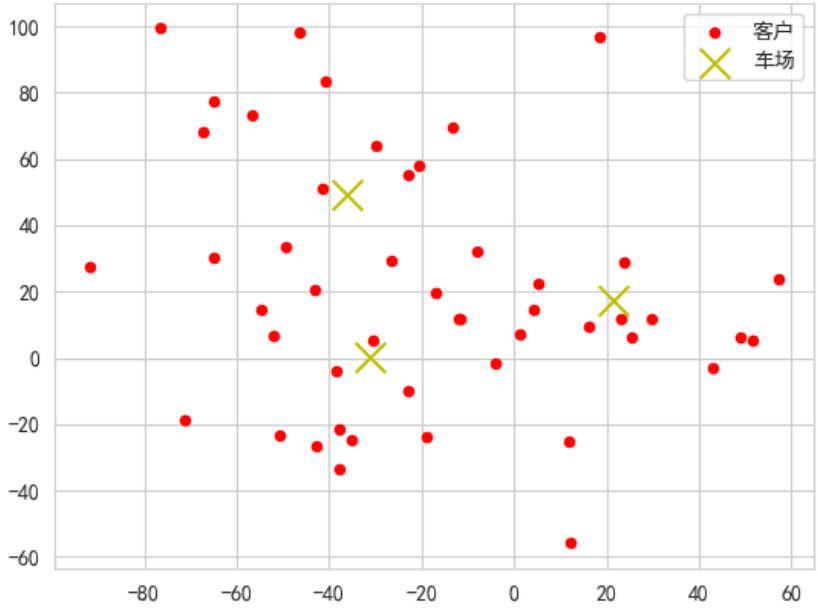

图 4.3 配送网络

客户参数主要包括横纵坐标、需求量、时间窗、服务时间，客户的参数如表 4.1 所示。

表 4.1 客户参数

编号	横坐标	纵坐标	需求量	最早配送时间	最迟配送时间	服务时间
1	-29.73	64.136	12	399	525	2
2	-30.664	5.463	8	121	299	7
3	51.642	5.469	16	389	483	21
4	-13.171	69.336	5	204	304	24
5	-67.413	68.323	12	317	458	1
6	48.907	6.274	5	160	257	17
7	5.243	22.26	13	170	287	6
8	-65.002	77.234	20	215	321	5
9	-4.175	-1.569	13	80	233	7
10	23.029	11.639	18	90	206	1
11	25.482	6.287	7	397	525	4

12	-42.615	-26.392	6	271	420	10
13	-76.672	99.341	9	108	266	2
14	-20.673	57.892	9	340	462	16
15	-52.039	6.567	4	226	377	23
16	-41.376	50.824	25	446	604	18
17	-91.943	27.588	5	444	566	3
18	-65.118	30.212	17	434	557	15
19	18.597	96.716	3	319	460	13
20	-40.942	83.209	16	192	312	10
21	-37.756	-33.325	25	414	572	4
22	23.767	29.083	21	371	462	23
23	-43.03	20.453	14	378	472	20
24	-35.297	-24.896	19	308	477	10
25	-54.755	14.368	14	329	444	4
26	-49.329	33.374	6	269	377	2
27	57.404	23.822	16	398	494	23
28	-22.754	55.408	9	257	416	6
29	-56.622	73.34	20	198	294	8
30	-38.562	-3.705	13	375	467	10
31	-16.779	19.537	10	200	338	70
32	-11.56	11.615	16	456	632	1
33	-46.545	97.974	19	72	179	21
34	16.229	9.32	22	182	282	6
35	1.294	7.349	14	159	306	4
36	-26.404	29.529	10	321	500	13
37	4.352	14.685	11	322	430	9
38	-50.665	-23.126	15	443	564	22
39	-22.833	-9.814	13	207	348	22
40	-71.1	-18.616	15	457	588	18
41	-7.849	32.074	8	203	382	10
42	11.877	-24.933	22	75	167	25

43	-18.927	-23.73	24	459	598	23
44	-11.92	11.755	3	174	332	4
45	29.84	11.633	25	130	225	9
46	12.268	-55.811	19	169	283	17
47	-37.933	-21.613	21	115	232	10
48	42.883	-2.966	10	414	531	17

车场、车辆与骑手的相关参数分别如表 4.2、4.3、4.4 所示。

表 4.2 车场参数

编号	横坐标	纵坐标
49	-31.201	0.235
50	21.387	17.105
51	-36.118	49.097

表 4.3 车辆参数

车辆编号	所属车场	车辆载重（kg）	续航里程（km）	启动成本（元/辆）	行驶速度（km/h）
1	49	200	500	200	40
2	49	100	300	100	50
3	49	150	400	150	45
4	50	100	300	100	50
5	50	200	500	200	40
6	50	150	400	150	45
7	51	150	400	150	45
8	51	200	500	200	40
9	51	100	300	100	50

表 4.4 骑手参数

骑手载重（kg）	启动成本（元/辆）
30	400

4.4.2 算例结果与分析

首先根据聚类算法将算例中的客户分配至各车场得到初始客户分配，客户分配结果如表 4.5 以及图 4.4 所示。

表 4.5 客户分配结果

车场	客户数量	客户编号
49	18	2 12 15 17 18 21 23 24 25 26 30 31 36 38 39 40 43 47
50	18	3 6 7 9 10 11 22 27 32 34 35 37 41 42 44 45 46 48
51	12	1 4 5 8 13 14 16 19 20 28 29 33

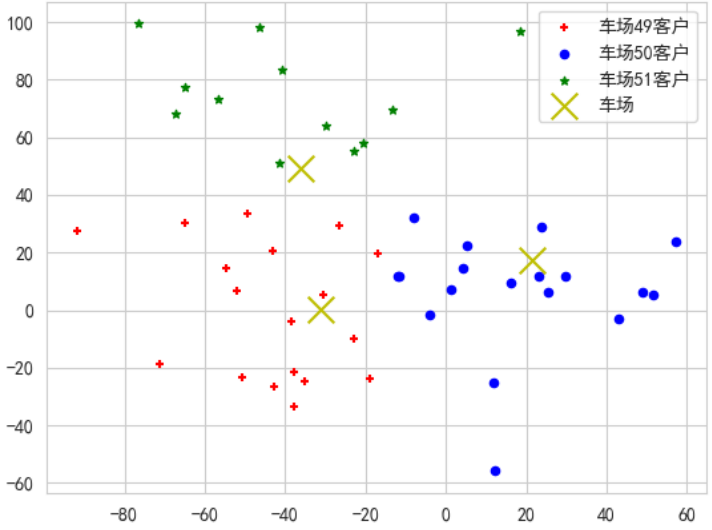

图 4.4 配送网络

本文经过多次数值实验测试，最终得到遗传算法以及粒子群算法各计算表现较佳的参数，如表 4.6 和表 4.7 所示。

表 4.6 遗传算法参数

符号	参数	参数设置
P_c	杂交概率	0.4
P_{cmax}	最大杂交概率	0.45
P_{cmin}	最小杂交概率	0.35
P_{m1}	基本变异一概率	0.3
P_{m1max}	最大基本变异一概率	0.35
P_{m1min}	最小基本变异一概率	0.25
P_{m2}	基本变异二概率	0.1
P_{m2max}	最大基本变异二概率	0.12
P_{m2min}	最小基本变异二概率	0.08
ω_{GA}	遗传算法权重	1/3
T_{GA}	迭代次数	50

表 4.7 粒子群算法参数

符号	参数	参数设置

r_1	个体交换序列保留概率	0.6
r_2	种群交换序列保留概率	0.7
C_{lo}	相邻两次迭代群体适应度变化阈值	0.01
ω_{PSO}	粒子群算法权重	2/3
T_{PSO}	迭代次数	500
S	种群规模	100

根据以上给定参数运行程序得出方案结果，配送总成本为 6151.34 元，车场 49 派出车辆 2 辆，车场 50 派出车辆 2 辆，车场 51 派出车辆 2 辆，共由 6 辆车提供服务，并由三名骑手分别完成尾单的派送。配送路线方案结果如图 4.5 所示，具体线路如表 4.8 所示，种群最优值迭代优化过程如图 4.6 所示。

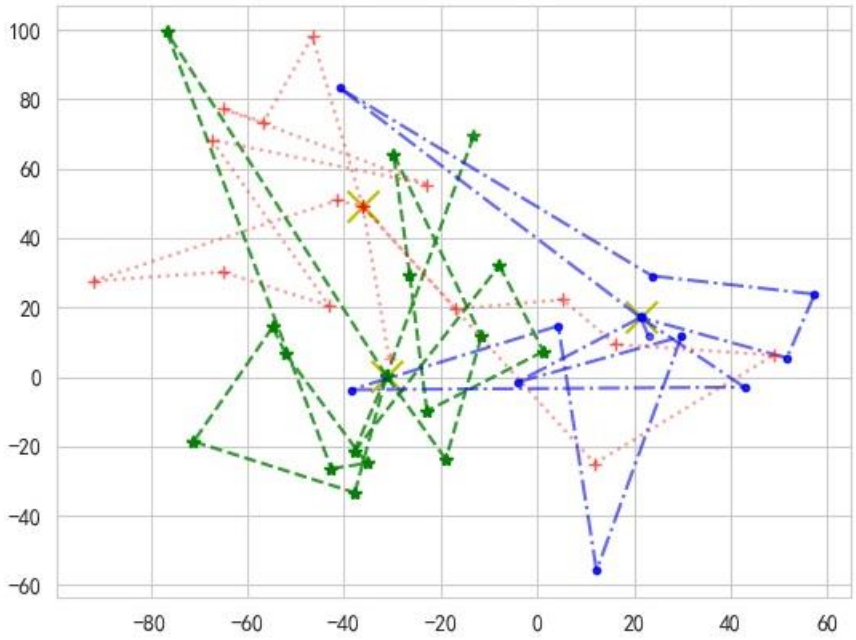

图 4.5 配送路线方案结果示意图

表 4.8 配送车辆路线

车场	车辆编号	路线
49	1	47-41-35-39-36-1-32-43
	3	13-12-24-15-25-40-21
50	5	9-45-46-37-30-48
	6	20-22-27-3
51	8	33-29-8-28-5-23-18-17-16
	9	42-6-34-7-31
骑手	1	38-44-4

骑手	2	26-19-10
骑手	3	11-14-2

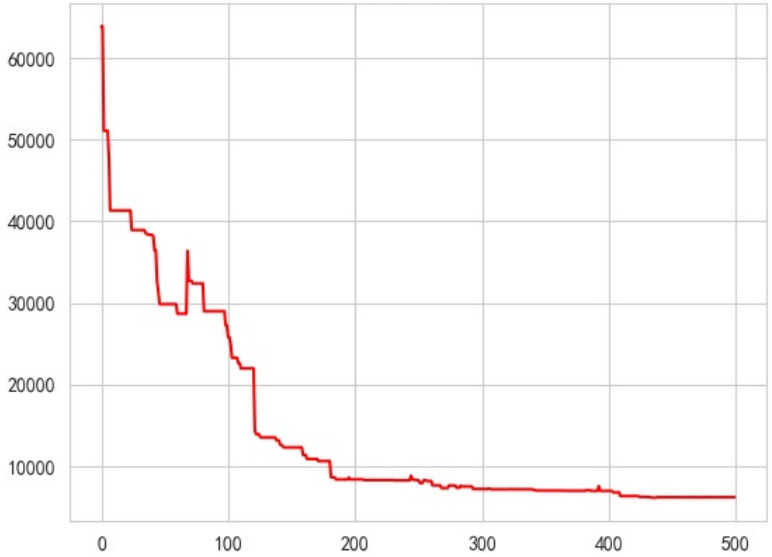

图 4.6 迭代优化过程示意图

4.4.3 算法对比

由于本文所建立的带时间窗的多车场车辆路径问题模型还考虑了多车型以及尾单配送，为验证本论文基于该模型的特殊性与复杂性所提出的改进的遗传粒子群混合算法对该模型求解的性能，分别采用粒子群算法、基本遗传粒子群混合算法以及改进的遗传粒子群混合算法优化该问题，初始客户分配方案均已根据 K-means 聚类算法进行优化，基本遗传粒子群混合算法全局最优解停滞代数超过阈值$C_{PSO}=80$即陷入局部最优，其他参数设置与改进的遗传粒子群混合算法保持一致。采用上述 3 种算法反复计算 100 次，通过对计算后的数据进行整理得出 3 种算法对该模型的优化结果比较以及优化结果箱型图比较分别如图 4.7 和图 4.8 所示。

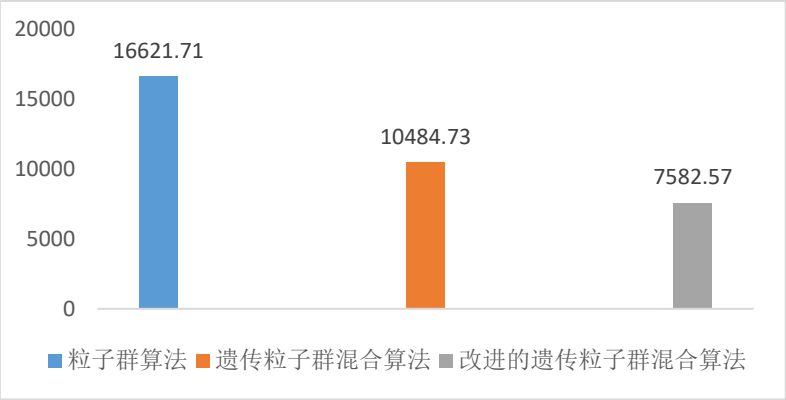

图 4.7 优化结果比较

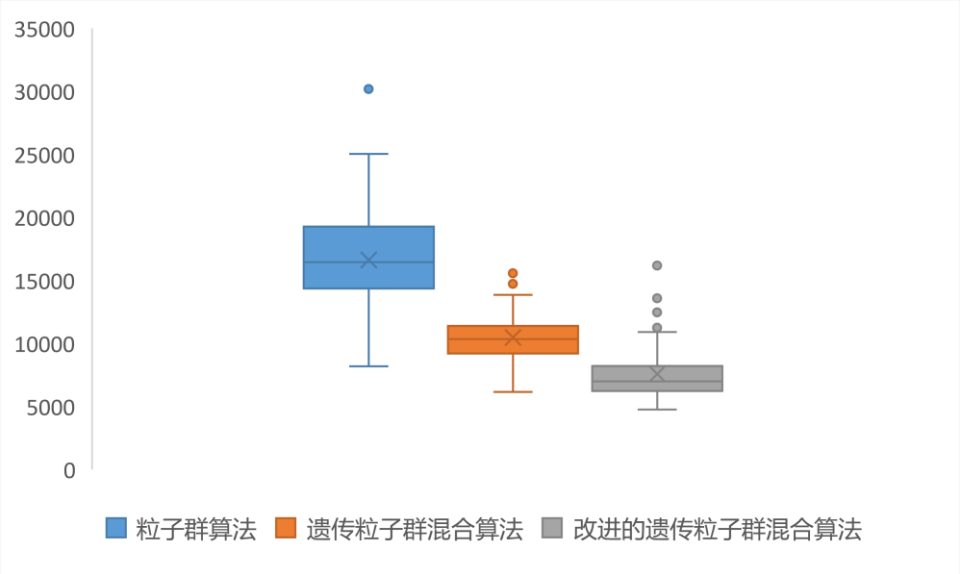

图 4.8 优化结果箱形图比较

图 4.7 的结果柱状图显示，粒子群算法的平均配送总成本为 16621.71 元，遗传粒子群混合算法的平均配送总成本为 10484.73 元，改进的遗传粒子群混合算法的平均配送总成本为 7582.57 元。因此可明显看出，遗传粒子群混合算法对比粒子群算法平均配送总成本下降程度达 58.5%，而改进的遗传粒子群混合算法相比基本遗传粒子混合算法平均配送总成本下降程度达 38.3%。通过图 4.8 优化结果箱形图比较可看出，将异常值剔除后，改进的遗传粒子群混合算法运行的 100 次求解结果都在一个很小的范围内集中分布，波动较小，且相比粒子群算法与遗传粒子群混合算法求解结果更优，且中位数分布较低，数据呈右偏分布，说明大部分求解结果都具有较低的值。由此得出结论，改进

的遗传粒子群混合算法可以很好地对本文模型进行优化求解，大大降低了配送总成本，对该模型具有更佳的求解效果。

4.5 本章小结

本章论文利用分布式的求解方法将分解法与全局优化法相结合，首先第一阶段采用分解法通过 K-means 聚类算法获得问题的初始客户分配方案，并通过遗传算法对初始配送路径进行优化，最终得到初始解；其次在第二阶段通过全局优化法利用改进的遗传粒子群混合算法对初始解进行优化，该算法在粒子群算法的基础上，通过群体适应度方差在相邻两次迭代中变化小于某阈值时引入遗传算法，最终得到问题的最优解。引用算例验证算法对本论文考虑多车型配送的带时间窗的多车场车辆路径问题模型求解的有效性，并与粒子群算法、基本遗传粒子群混合算法分别对该模型进行 100 次计算实验并对所有结果整理建立柱状图与箱形图对比分析发现本篇论文提出的改进的遗传粒子群混合算法在对模型的求解效果上更佳。

5.带时间窗的多车场车辆路径问题系统设计及实现

5.1 带时间窗的多车场车辆路径问题模拟系统

5.1.1 开发目的和意义

物流行业是我国经济中的一个重要经济增长点，物流行业推动我国经济的增长，同时经济增长也带动了物流技术的发展与变革，从某种程度上来说，商流、物流、资金流、信息流构成了流通过程中的四大部分，同时这四大部分相互作用且密不可分，信息流涉及信息化、数字化等。

其中，信息化是以通讯、计算机技术为基础以实现信息之间的共享与传输来优化服务；数字化则是以计算机技术通过运用数据采集和分析手段实现物理世界的网络虚拟化，通过仿真技术、数据库、智能计算等技术对物流行业的支持，数字化能够将物流所涉及的对象、要素以及活动进行控制处理并显示，因此现代物流企业需要具备信息化、数字化、可视化以及网络化等特征，它们对物流活动的高效运作至关重要，已成为现代物流企业的生存法则，推动着物流朝向智能化方向发展，向智能物流进发，而这些特征的载体则是使得物流企业服务高效有力的物流系统。物流配送所面临的环境随着经济发展越加复杂，如交通道路网络错综复杂、客户点零落分散、客户对配送时间要求不一等。因此，对于如何设计规划低成本高质量的物流配送方案极为关键，合理地将现代数学方法与计算机技术相结合开发出有效的物流配送规划软件十分有利于物流企业的车辆调度配送，同时达到提高客户服务质量、降低企业营运成本、实现物流智能化的目的，对于物流企业自身发展甚至整个国民经济水平的发展都有良好的促进作用。

本章意在利用客户的定位和订单数据以及物流企业的车场车辆数据开发出一套简单的以最小的配送成本为目标的物流配送路径模拟系统，具有一定的理论意义与实际价值，能为物流企业的车辆调度提供一定的便利性，做出一定的贡献，符合现代物流企业迫切需要加快传统物流运输配送与信息化、数字化、可视化等的融合。

5.1.2 系统功能及特点

本章的核心是开发出一套高效的计算机系统，主要针对现代企业车辆配送路径进行设计规划，该系统的作用是对车辆的管理调度进行合理安排，功能主要包括车场分配、规划车辆运输路线等。

本系统具有的特点包括以下几方面：

（1）本系统采用 QT Designer 实现视图部分即用户界面显示、采用 Python 实现逻辑部分即车辆路径优化功能，并通过 QT Designer 与 Python 交互实现车场、车辆以及客户数据的存储设置。

（2）本系统包含三个模块：数据库模块、迭代执行模块、用户界面显示模块。其中数据库模块负责车场信息、车辆信息、客户信息、参数的存储设置；迭代执行模块负责算法迭代优化的逻辑实现；用户界面模块负责路径优化结果、迭代过程以及配送总成本的显示。

（3）本系统采用的改进的遗传粒子群混合算法对车场、客户及车辆无其他限制。根据用户输入并点击开始，系统将执行算法开始自动迭代，最终自动将路径优化结果、迭代过程以及配送总成本显示在界面中。

（4）本系统求解所耗费的时间与用户录入的车场信息、客户信息以及参数有关，总可在短时间内得到满意的结果。

5.1.3 系统模块介绍

本章系统总体分为三个功能模块：数据库模块、迭代执行模块、用户界面显示模块，系统界面如图 5.1 所示。其中数据库模块细分为车场信息模块、车辆信息模块、客户信息模块；用户界面显示模块细分为客户、尾单信息以及算

法迭代参数的设置模块、路径优化结果、迭代过程和配送总成本显示模块。优化执行模块属于后台程序。

图 5.1 带时间窗的多车场车辆路径问题系统界面

（1）数据库模块

首先将车场信息、车辆信息、客户信息、尾单信息以及迭代参数信息录入数据库中，构建生成用于进行车辆路径优化的原始数据。其中车场信息包括车场编号以及坐标；车辆信息包括车辆编号、所属车场、载重量、车速、里程限制以及使用成本；客户信息包括客户编号、需求量、坐标、服务时间、时间窗、早到成本以及晚到成本；尾单信息包括骑手启动成本以及最大载重量；迭代参数信息包括种群规模、粒子群算法的迭代次数、适应度方差变化阈值。用户每次初始化数据可以通过自行输入或是从现有数据中传入，界面如图 5.2 所示。

图 5.2 数据库界面图

（2）迭代执行模块

该模块在后台程序中采用本篇论文的改进的遗传粒子群混合算法进行迭代。

（3）用户界面显示模块

用户界面显示模块的功能可向用户展示路径规划结果、迭代过程以及配送成本等，并获取每个车场的车辆配送路径调度安排，随后车场可以据此发出物流配送路径安排，送货司机听从此安排进行配送。用户界面结果如图 5.3 所示，其中 x1 到某车场和 x 到某车场的区间表示尾单派送。

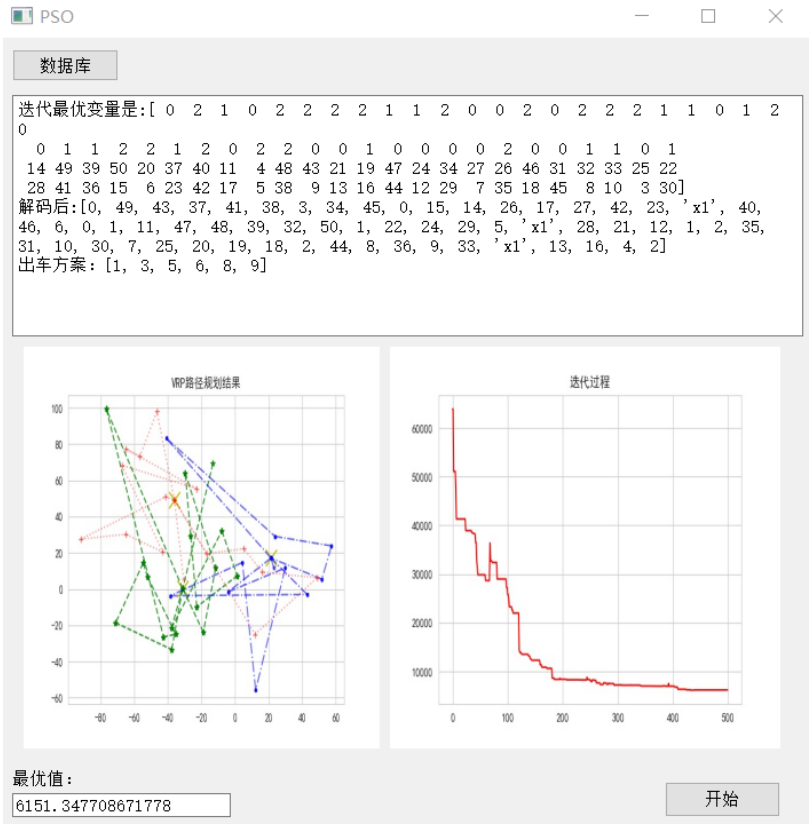

图 5.3 用户界面结果显示

5.2 本章小结

本章通过介绍信息化以及数字化已成为当代物流企业的一大趋势，对于物流企业竞争力的强化具有重要意义，阐述了开发车辆路径优化系统的实际意义，并详细地对该系统各个模块做出介绍，通过用户对数据及参数信息进行输入并执行系统，改进的遗传粒子群混合算法将能够找到合理的车辆配送解决方案，证明了该系统在企业的运用中具有一定的价值。在现实中物流配送情况或许更为复杂，系统仍具有一定缺陷有待后续研究。

6.总结与展望

6.1 总结

本文首先介绍了物流配送在国内物流业的重要地位，引出了研究车辆路径问题的重要意义，详细地阐述了车辆路径问题并总结了其求解方法，随后对粒子群算法以及遗传算法的基本概念、基本思想、流程步骤以及改进策略做出详细介绍，以此为基础对带时间窗的多车场车辆路径问题进行了分析，基于更为复杂的现实情况考虑了多车型以及尾单配送，建立了数学模型，接着介绍了本文分布式求解方法以及混合算法的选择策略，针对基本粒子群算法的不足，提出了基于群体适应度方差和自适应杂交变异概率的改进的遗传粒子群混合算法。利用 Python 进行编程，通过 VRP 标准算例进行验证，最后证明了该算法对该模型求解的有效性。本文主要研究完成了以下几个方面的内容：

（1）本文对前人的研究进行了总结，对车辆路径问题进行了详尽地阐述，从标准的车辆路径问题到引入不同约束条件的车辆路径问题，最后引出本文研究的带时间窗的多车场车辆路径问题并基于现实情况考虑了多车型以及尾单配送，由此建立了与现实相贴合的数学模型。

（2）本文的编码方式采用了双层编码以区别客户分配方案与车辆配送路径，同时对于多车场的车辆路径问题采取两阶段法首先利用 K-means 聚类算法与遗传算法分别优化了初始解中的客户分配方案以及车辆配送路径，大大提高了粒子找寻最优解的速度。

（3）本文在粒子群算法中引用前人利用粒子群算法求解旅行商问题的方法引入交换子与交换序列的概念替代了粒子的速度更新公式，使得算法更好地适用于车辆路径问题这类离散化问题。

（4）在前人对遗传算法与粒子群算法的各种改进基础上提出了本文对于遗传粒子群混合算法的改进思路，在连续迭代的适应性方差变化小于某一阈值时引入遗传算法，同时在遗传算法中引入自适应杂交变异概率，使得杂交概率与变异概率随着粒子群迭代次数与遗传算法迭代次数而变化，在算法收敛速度得到改善的同时尽可能保证了优秀粒子的延续。

（5）通过 Python 编程对改进的遗传粒子群混合算法进行了算例验证并与其他算法进行了对比，证明了该算法在求解本文具有特殊性与复杂性的车辆路径问题上的有效性与可行性。

6.2 展望

车辆路径问题本身极其复杂，问题规模的增加将大大增加求解该问题的复杂程度，本文虽然在该问题上进行了探讨但碍于作者水平有限以及时间的限制，有些问题还有待更深入地研究：

（1）本文对于连续迭代的适应度方差变化阈值以及自适应杂交概率与变异概率中遗传算法与粒子群算法权重分配不够细致，不能够完全囊括运用于所有车辆路径问题，将来需要根据具体问题来明确定值的设定。

（2）本文对于遗传算法的改进仅限于杂交概率与变异概率，虽然相比基本遗传算法有较好的改进，但算法仍存在不足，在种群规模以及遗传算子上仍有改进的空间，有待进一步研究。

（3）在车辆配送问题上实际情况错综复杂，例如交通路况的不确定，配送路径可能会由于交通拥堵情况发生改变；客户的时间窗在配送过程中发生变化等等。诸如此类约束条件复杂的车辆路径问题具有更高的实用性同时也更具挑战性，本文碍于作者水平以及时间有限并未涉及。

参考文献

[1] Angeline, P. J. (1998, May). Using selection to improve particle swarm optimization. In 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360) (pp. 84-89). IEEE.

[2] Bellman, R. (1966). Dynamic programming. Science, 153(3731), 34-37.

[3] Clarke, G., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. Operations research, 12(4), 568-581.

[4] Clerc, M. (1999, July). The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. In Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406) (Vol. 3, pp. 1951-1957). IEEE.

[5] Colorni, A., Dorigo, M., & Maniezzo, V. (1992, September). An Investigation of some Properties of an" Ant Algorithm". In Ppsn (Vol. 92, No. 1992).

[6] Desrochers, M., Desrosiers, J., & Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. Operations research, 40(2), 342-354.

[7] Gillett, B. E., & Miller, L. R. (1974). A heuristic algorithm for the vehicle-dispatch problem. Operations research, 22(2), 340-349.

[8] Higashi, N., & Iba, H. (2003, April). Particle swarm optimization with Gaussian mutation. In Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706) (pp. 72-79). IEEE.

[9] Kennedy, J., & Eberhart, R. (1995, November). Particle swarm optimization. In Proceedings of ICNN'95-international conference on neural networks (Vol. 4, pp. 1942-1948). IEEE.

[10] Kolen, A. W., Rinnooy Kan, A. H. G., & Trienekens, H. W. (1987). Vehicle routing with time windows. Operations Research, 35(2), 266-273.

[11] Land, A. H., & Doig, A. G. (2010). An automatic method for solving discrete

programming problems. In 50 Years of Integer Programming 1958-2008 (pp. 105-132). Springer, Berlin, Heidelberg.

[12] Lovbjerg, M., Rasmussen, T. K., & Krink, T. (2000). Hybrid particle swarm optimization with breeding and subpopulations [R]. In IEEE International Conference on Evolutionary Computation, San, Diego (Vol. 1, No. 11, pp. 394-407).

[13] Mühlenbein, H., Schomisch, M., & Born, J. (1991). The parallel genetic algorithm as function optimizer. Parallel computing, 17(6-7), 619-632.

[14] Potvin, J. Y., Kervahut, T., Garcia, B. L., & Rousseau, J. M. (1996). The vehicle routing problem with time windows part I: tabu search. INFORMS Journal on Computing, 8(2), 158-164.

[15] Renaud, J., Laporte, G., & Boctor, F. F. (1996). A tabu search heuristic for the multi-depot vehicle routing problem. Computers & Operations Research, 23(3), 229-235.

[16] Savelsbergh, M. W. (1985). Local search in routing problems with time windows. Annals of Operations research, 4(1), 285-305.

[17] Shi, Y., & Eberhart, R. (1998, May). A modified particle swarm optimizer. In 1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360) (pp. 69-73). IEEE.

[18] Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations research, 35(2), 254-265.

[19] Taillard, É., Badeau, P., Gendreau, M., Guertin, F., & Potvin, J. Y. (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. Transportation science, 31(2), 170-186.

[20] Tie-jun, W., & Kai-jun, W. (2012, December). Adaptive particle swarm optimization based on population entropy for MDVRPTW. In Proceedings of 2012 2nd International Conference on Computer Science and Network Technology (pp. 753-756). IEEE.

[21] Tsutsui, S., Fujimoto, Y., & Ghosh, A. (1997). Forking genetic algorithms: GAs with search space division schemes. Evolutionary computation, 5(1), 61-80.

[22] Wen, L., & Meng, F. (2008, December). An improved PSO for the multi-depot vehicle routing problem with time windows. In 2008 IEEE Pacific-Asia

Workshop on Computational Intelligence and Industrial Application (Vol. 1, pp. 852-856). IEEE.

[23] Zhao, Y. W., Wu, B., Wang, W. L., Ma, Y. L., Wang, W. A., & Sun, H. (2004). Particle swarm optimization for vehicle routing problem with time windows. In Materials Science Forum (Vol. 471, pp. 801-805). Trans Tech Publications Ltd

[24] 陈璐璐,邱建林,陈燕云,陆鹏程,秦孟梅 & 赵伟康.(2017).改进的遗传粒子群混合优化算法. 计算机工程与设计(02),395-399.

[25] 陈婷.(2010).基于变异的粒子群算法的 MDVRPTW 研究(硕士学位论文,华东师范大学).

[26] 董勇 & 郭海敏.(2011).基于群体适应度方差的自适应混沌粒子群算法. 计算机应用研究(03),854-856.

[27] 李大卫,王莉,王梦光.(1999).遗传算法在有时间窗车辆路径问题上的应用. 系统工程理论与实践(08).

[28] 刘春,朱俊林.(2003).中国城市物流中心的布局现状分析. 资源开发与市场(06),368-370.

[29] 刘建华.(2009).粒子群算法的基本理论及其改进研究(博士学位论文,中南大学).

[30] 刘敏.(2006).多目标遗传算法在车辆路径优化中的应用研究(硕士学位论文,湘潭大学).

[31] 吕振肃,侯志荣.(2004).自适应变异的粒子群优化算法. 电子学报(03),416-420.

[32] 马超,邓超,熊尧 & 吴军.(2013).一种基于混合遗传和粒子群的智能优化算法. 计算机研究与发展(11),2278-2286.

[33] 孙艺婕.(2020).带时间窗的城市物流多中心配送路径优化研究(硕士学位论文,浙江海洋大学).

[34] 田欣.(2016).基于改进遗传算法的移动机器人路径规划研究(硕士学位论文,郑州大学).

[35] 王俊伟.(2006).粒子群优化算法的改进及应用(博士学位论文,东北大学).

[36] 王小会.(2015).多车场带时间窗车辆路径问题的粒子群优化算法. 兰州工

业学院学报(02),51-55.

[37] 王璇.(2010).遗传算法的改进及其应用研究(硕士学位论文,华北电力大学（北京）).

[38] 王运涛,刘钢 & 薛俊芳.(2022).基于改进遗传算法的拆卸序列规划. 现代制造工程(01),137-142+136.

[39] 徐晓华,陈崚 & 陈宏建.(2006).可变种群规模的遗传算法. 系统仿真学报(04),870-872+876.

[40] 许伊萍.(2015).浅谈旅行商问题与粒子群优化算法. 电子制作(09),268.

[41] 叶建芳,王正肖 & 潘晓弘.(2008).免疫粒子群优化算法在车间作业调度中的应用. 浙江大学学报(工学版)(05),863-868+879.

[42] 张建科.(2006).几类改进的粒子群算法(硕士学位论文,西安电子科技大学).

[43] 张丽艳,庞小红,夏蔚军,吴智铭 & 梁硕.(2006).带时间窗车辆路径问题的混合粒子群算法. 上海交通大学学报(11),1890-1894+1900.

[44] 张铃,张钹.(1997).统计遗传算法. 软件学报(05).

[45] 张淑丽,张涛,崔岩 & 刘仁贵.(2019).基于混沌粒子群算法的物流配送路径优化方法. 电子设计工程(15),108-112+117.

[46] 邹彤,李宁,孙德宝,李菁.(2004).多车场车辆路径问题的遗传算法. 计算机工程与应用(21),82-83.

附 录

部分算法代码

```
variance_old = 0
fitness_value_list = []  # 每一步迭代的最优解
distance_matrix = self.calculate_distance(X, Y)
isolution = self.initialize_solution(distance_matrix)
# 编码
# 初始化种群的各个粒子的位置，作为个体的历史最优解 pbest
XXX = np.zeros((size, customer_num * 2), dtype=int)
for i in range(size):
    for j in range(customer_num):
        if openkmeans == 1:
            XXX[i][j] = isolution[j]
        else:
            XXX[i][j] = np.random.randint(0, dis_center)
    XXX[i][customer_num:] = np.random.choice(list(range(dis_center,
customer_num + dis_center)), size=customer_num,
                                             replace=False)
# 计算每个粒子对应适应度
pbest_noen = XXX
pbest = self.encoding(XXX, size)
pbest_fitness = np.zeros((size, 1))
for i in range(size):
    pbest_fitness[i] = self.fitness_func(distance_matrix, xi=pbest[i],
carcase=ini_carcase[i])
gbest_fitness = pbest_fitness.min()
if openinfirst == 1:
```

```
        GAX = self.mutation1(XXX, mut1)
        GAX1 = self.mutation2(GAX, mut2)
        GAX2 = self.crossover(GAX1, c)
        enGAX1 = self.encoding(GAX2, size)
        GAX1_fitness = np.zeros((size, 1))
        for i in range(size):
            GAX1_fitness[i] = self.fitness_func(distance_matrix, xi=enGAX1[i],
carcase=ini_carcase[i])
        XXX = GAX2
pbest_noen = XXX
for i in range(size):
        if GAX1_fitness[i] < pbest_fitness[i]:
            pbest_fitness[i] = GAX1_fitness[i]
else:
        pass
# 计算全局适应度和对应的解 gbest
gbest = XXX[pbest_fitness.argmin()]
gbest_fitness = pbest_fitness.min()
# 记录算法迭代效果
fitness_value_list.append(gbest_fitness)
# 下面开始迭代
final_gbest = []
final_gbest_fitness = 1000000000
for i in range(iter_num):
        for j in range(size):  # 对每个粒子迭代
            pbesti = pbest_noen[j].copy()
            xi = XXX[j].copy()
            # 计算杂交序列
            ss1 = self.get_ss(pbesti, xi, r1)
            ss2 = self.get_ss(gbest, xi, r2)
            ss = ss1 + ss2
            # 执行杂交操作
            xi = self.do_ss(xi, ss)
```

```
# 解码
XXXX = []
XXX1 = dict()
XXX2 = dict()
XXX3 = dict()
XXX4 = dict()
XXX5 = dict()
XXX6 = dict()
for n in range(customer_num):
    if (xi[n] == 0) and (xi[n] in dis_center_list):
        XXX1[xi[n + customer_num]] = n + dis_center
    elif (xi[n] == 1) and (xi[n] in dis_center_list):
        XXX2[xi[n + customer_num]] = n + dis_center
    elif (xi[n] == 2) and (xi[n] in dis_center_list):
        XXX3[xi[n + customer_num]] = n + dis_center
    elif (xi[n] == 3) and (xi[n] in dis_center_list):
        XXX4[xi[n + customer_num]] = n + dis_center
    elif (xi[n] == 4) and (xi[n] in dis_center_list):
        XXX5[xi[n + customer_num]] = n + dis_center
    elif (xi[n] == 5) and (xi[n] in dis_center_list):
        XXX6[xi[n + customer_num]] = n + dis_center
for k in sorted(XXX1):
    XXXX.append(XXX1.get(k))
XXXX.append(0)
XXXX.insert(0, 0)
if XXX2 != {}:
    XXXX.append(1)
    for l in sorted(XXX2):
        XXXX.append(XXX2.get(l))
    XXXX.append(1)
    if XXX3 != {}:
        XXXX.append(2)
        for m in sorted(XXX3):
```

```
                    XXXX.append(XXX3.get(m))
                XXXX.append(2)
                if XXX4 != {}:
                    XXXX.append(3)
                    for m in sorted(XXX4):
                        XXXX.append(XXX4.get(m))
                    XXXX.append(3)
                    if XXX5 != {}:
                        XXXX.append(4)
                        for m in sorted(XXX5):
                            XXXX.append(XXX5.get(m))
                        XXXX.append(4)
                        if XXX6 != {}:
                            XXXX.append(5)
                            for m in sorted(XXX6):
                                XXXX.append(XXX6.get(m))
                            XXXX.append(5)
        routebags = []
        routebag = 0
        for l in range(1, len(XXXX)):
            if ((XXXX[l] in dis_center_list) and (XXXX[l - 1] in dis_center_list))
or (l == len(XXXX) - 1):
                routebags.append(routebag)
                routebag = 0
            else:
                routebag += t[XXXX[l]]
        routedistances = []
        routedistance = 0
        for l in range(1, len(XXXX)):
            if l == len(XXXX) - 1:
                routedistance += distance_matrix[XXXX[l - 1]][XXXX[l]]
                routedistances.append(routedistance)
            elif (XXXX[l] in dis_center_list) and (XXXX[l - 1] in
```

```
dis_center_list):
                    routedistances.append(routedistance)
                    routedistance = 0
                else:
                    routedistance += distance_matrix[XXXX[l - 1]][XXXX[l]]
            ini_carcase = []
            for m in range(len(routedistances)):
                carincenter = [car for car in carsp if car[1] == m]
                carincenter.sort(key=lambda x: (-x[3]), reverse=True)
                while (len(carincenter) != 0) and (routedistances[m] > routeflag):
                    for l in range(len(carincenter)):
                        if routedistances[m] - carincenter[l][3] <= routeflag:
                            routedistances[m] -= carincenter[l][3]
                            ini_carcase.append(carincenter[l][0])
                            carincenter.pop(l)
                            break
                        elif l == len(carincenter) - 1:
                            routedistances[m] -= carincenter[l][3]
                            ini_carcase.append(carincenter[l][0])
                            carincenter.pop(l)
                        else:
                            continue
            # 确保满足车容量和里程限制
            xii = []  # 已解码
            xii.append(0)
            carn = 0
            for o in range(len(XXXX) - 1):
                if XXXX[o] in dis_center_list:
                    dflag = 0
                    start = XXXX[o]
                    sum = 0
                    distan = 0
                    Q = carsp[ini_carcase[carn] - 1][2]
```

```
                distance = carsp[ini_carcase[carn] - 1][3]
            s = XXXX[o]
            end = XXXX[o + 1]
            sum += t[end]
            distan += distance_matrix[s][end]
            if dflag == 1:
                if (sum > driver_c):
                    xii.append('x')
                    sum = t[end]
            elif (sum > Q) or ((distan + distance_matrix[end][start]) > distance):
                if carn == len(ini_carcase) - 1:
                    dflag = 1
                    xii.append('x1')
                    sum = t[end]
                elif carsp[ini_carcase[carn + 1] - 1][1] == carsp[ini_carcase[carn] - 1][1]:
                    xii.append(start)
                    sum = t[end]
                    distan = distance_matrix[start][end]
                    carn += 1
                    Q = carsp[ini_carcase[carn] - 1][2]
                    distance = carsp[ini_carcase[carn] - 1][3]
                else:
                    dflag = 1
                    xii.append('x1')
                    sum = t[end]
            elif (XXXX[o + 1] in dis_center_list) and (XXXX[o] in dis_center_list):
                carn += 1
            xii.append(end)
        # 判断是否更优
        fitness_new = self.fitness_func(distance_matrix, xii, ini_carcase)
        fitness_old = pbest_fitness[j]
```

```
            if fitness_new < fitness_old:
                pbest_fitness[j] = fitness_new
                pbest[j] = xii
                pbest_noen[j] = xi
        # 判断全局是否更优
        gbest_fitness_new = pbest_fitness.min()
        gbest_new = XXX[pbest_fitness.argmin()]
        variance = self.ca_variance(pbest_fitness)
        # print(variance)
        if i == 0:
            variance_old = variance
        if gbest_fitness_new < gbest_fitness:
            gbest_fitness = gbest_fitness_new
            gbest = gbest_new
        # 陷入局部最优
        if open1 == 1:
            # print(variance_old,variance,d_value,abs(variance_old - variance))
            if (abs(variance_old - variance) <= d_value) and (abs(variance_old -
variance) != 0):
                d_v = abs(variance_old - variance)
                if gbest_fitness < final_gbest_fitness:
                    final_gbest = gbest
                    final_gbest_fitness = gbest_fitness
                for k in range(times):
                    if k == 0:
                        GAX1 = self.select(XXX, pbest_fitness)
                    else:
                        GAX1 = self.select(XXX, enXXX_fitness)
                    if mut1 > mutmin1:
                        mut1 = mutmax1 * (wga * np.cos(np.pi / 2 * k / times) +
                                          wpso * np.cos(np.pi / 2 * i /
iter_num))
                    else:
```

```
                    mut1 = mutmin1
                if mut2 > mutmin2:
                    mut2 = mutmax2 * (wga * np.cos(np.pi / 2 * k / times) +
                                      wpso * np.cos(np.pi / 2 * i /
iter_num))
                else:
                    mut2 = mutmin2
                if c > cmin:
                    c = cmax * (wga * np.cos(np.pi / 2 * k / times) +
                                wpso * np.cos(np.pi / 2 * i / iter_num))
                else:
                    c = cmin
                GAX2 = self.mutation1(GAX1, mut1)
                GAX3 = self.mutation2(GAX2, mut2)
                GAX4 = self.crossover(GAX3, c)
                enXXX = self.encoding(GAX4, size)
                enXXX_fitness = np.zeros((size, 1))
                for j in range(size):
                    enXXX_fitness[j] = self.fitness_func(distance_matrix,
xi=enXXX[j], carcase=ini_carcase[j])
                    XXX = GAX4
            pbest_noen = XXX
            pbest_fitness = enXXX_fitness
            gbest_fitness = pbest_fitness.min()
            gbest = XXX[pbest_fitness.argmin()]
        else:
            pass
    variance_old = variance
    # 加入到列表
    fitness_value_list.append(gbest_fitness)
```

后 记

论文的写作在此告一段落，回首整个写作过程，每一个阶段都富有挑战性，从最初对研究方向的选择着眼于硕士阶段对代码编写能力的检验，到有关于算法的选择与改进以及对于文献的收集整理再至最后论文的撰写修改，一直在导师的指引帮助下逐步改进完善，其中对于算法代码的编写便花费了不少功夫。

作为一个代码编写的门外汉，在导师的指引以及自我的学习过程中不断精进提升编写能力，由起初眼里密密麻麻的数字与字母到一排排逻辑缜密的函数与定义，过程中收获到的不仅是编写能力，更重要的是自己的思维与逻辑能力。认识到了算法对于理论以及实际应用的重要性，同时也发现编程能力对学术研究和行业实践有着巨大作用，此次研究写作收获颇丰。

致　谢

时光荏苒，白驹过隙之间硕士研究生三年的学习就要结束，我始终坚信这三年时光将是我人生浓墨重彩的一笔，在这三年的扎根汲取养分之中，身旁有许许多多的人给予我支持与帮助，在此要向他们表示由衷感谢！

首先需要感谢我的研究生导师田野教授，在我弥足珍贵的三年硕士研究生生涯中，给予我的教诲与指导。田野教授学识过人、诲人不倦，与学生打成一片使我由衷感到尊敬。从最初的论文选题至论文收尾中，田野教授总会为我梳理并提供思路，给予我建议及指导，对我顺利毕业起到关键作用。同时田野教授针对学生的就业传授了许多经验与建议，使我受益匪浅。我十分幸运且庆幸在学习生涯甚至是人生中遇上如此良师，在此我向田野教授表示衷心感谢！

感谢我的好朋友文彦博、祝笛、吴一檬、李嘉祺在我情绪糟糕之时给予我的帮助与支持陪伴，拥有你们使我永远勇敢，披荆斩棘向前迈步。

感谢我的研究生好朋友李一帆与室友黄鹏等同学，你们对我学业的帮助以及校园生活的陪伴是我在西财这几年收获的宝贵财富。

其次我非常感谢我的父母及其家人，是他们对我的关怀与教育帮助我完成学业，感谢母亲杨妍女士为我的成长做出的牺牲！

在学者们的研究努力下算法对车辆路径问题的求解越发精确快速，愿此后诸君人生路径不求精确快速，但始终在向最优解逼近，莫听穿林打叶声，何妨吟啸且徐行。

经世济民
孜孜以求

西南财经大学
SOUTHWESTERN UNIVERSITY OF FINANCE AND ECONOMICS

校址：四川成都温江柳台大道555号
电话：028-87092032　传真：028-87092632
邮编：611130
网址：http://www.swufe.edu.cn
Address: Liulin Campus (Main Campus): 555, Liutai Avenue,
Wenjiang District, Chengdu, Sichuan, P. R. China
Tel: 86-28-87092032　Fax: 86-28-87092632　Postcode: 611130