电子科技大学

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

专业学位硕士学位论文

MASTER THESIS FOR PROFESSIONAL DEGREE

论文题目 粒子群优化算法在车辆路径问题中的应用研究

专业学位类别	工 程 硕 士
学　　号	201622060127
作 者 姓 名	谢传聪
指 导 教 师	吴卫平　教 授

分类号 _____ 密级 _____

UDC [注1] _____

学 位 论 文

粒子群优化算法在车辆路径问题中的应用研究

（题名和副题名）

谢传聪

（作者姓名）

指导教师 吴卫平 教 授

电子科技大学 成 都

（姓名、职称、单位名称）

申请学位级别 硕士 专业学位类别 工 程 硕 士

工程领域名称 计算机技术

提交论文日期 2019.11.5 论文答辩日期 2019.12.3

学位授予单位和日期 电子科技大学 2019 年 12 月

答辩委员会主席 _____

评阅人 _____

注 1：注明《国际十进分类法 UDC》的类号。

Application of Particle Swarm Optimization in Vehicle Routing Problem

A Master Thesis Submitted to

University of Electronic Science and Technology of China

Discipline: **Master of Engineering**

Author: **Chuancong Xie**

Supervisor: **Prof. Wei-Ping Wu**

School: **School of Computer Science and Engineering**

(School of Cybersecurity)

摘　要

优化问题是在物流供应链管理中一个非常重要的课题。随着技术的发展，优化问题也变得越发复杂，难以解决。智能算法对于求解这些优化问题有着明显的优势，在计算规模较大，较为复杂的优化问题时，智能算法有着更快的求解效率，适应性更广。

粒子群优化算法(Particle Swarm Optimization, PSO)是群体智能算法的一种，它模拟自然界中鸟类飞行觅食的过程，在搜索空间中寻找出解决问题的最佳方案。PSO 算法由于其参数的数量较少，有着简单易于实现、收敛速度快的优点，但它也同样存在易陷入局部最优的缺陷，这个缺陷也限制了其实际应用，因此对粒子群算法做出相应的改进有着十分重要的意义。车辆路径问题(Vehicle Routing Problem，VRP)作为典型的 NP 难问题，使用智能算法求解是目前主要的解决方案，本文也将 PSO 算法应用到了 VRP 中，对 PSO 算法应用进行了研究。

本文首先介绍了粒子群算法和车辆路径问题的国内外研究现状，其次对粒子群算法的原理和算法流程进行了阐述，并总结出了几种常见的改进方向，介绍了几种常见的标准测试函数。在这些理论基础上，本文根据粒子群算法本身的缺陷，将莱维飞行和反向学习机制引入了粒子群算法中，记录每个粒子飞行过程中的历史最差位置，当粒子陷入停滞时，对其采用反向学习机制，使用历史最差位置牵引粒子逃离局部最优，学习的步长采用莱维分布的随机数。通过仿真对比实验证明了算法的可行性和有效性。然后将所提出的改进粒子群算法应用到两个不同的车辆路径问题实际案例中进行问题的求解，证明了算法具有实际应用价值。最后，根据本文的研究内容，设计并实现了一个药品配送系统，实现了对本文所提出的 PSO 算法的实际应用。

关键词：粒子群优化算法，莱维飞行，反向学习，车辆路径问题，药品配送系统

ABSTRACT

Optimization is an important research target in supply chain management. With recent technological advances, optimization has become complex and more difficult to solve. Swarm intelligence optimization algorithms show good performance in solving optimization problems in terms of application area with high efficiency.

Particle swarm optimization(PSO) algorithm is one of the swarm intelligence algorithms. It simulates the process of birds flying and foraging and finds the best solution for the problem. PSO has the advantages of fast computation speed and few parameters. However, PSO is prone to premature convergence, which will inevitably affect its practical application. Therefore, it is of great significance to improve PSO and expend its application. Vehicle Routing Problem(VRP) is a typical NP-hard problem, which can be solved efficiently with swarm intelligence algorithms. This thesis also applies PSO algorithm to VRP and studies the application of PSO.

Firstly, the research status of PSO and VRP at home and abroad is introduced. Secondly, the principle and flow chart of PSO are described, and several common improvement directions are summarized. On the basis of these theories, this thesis develops a new PSO algorithm combining Levy flight and reverse learning mechanism. The new algorithm records the personal worst position of each particle. When a particle is stagnated, the reverse learning mechanism is adopted. The personal worst position is used to pull the particle out of local optimum, and the learning step is Levy fraction. The effectiveness and feasibility of the algorithm are proven by simulation and comparative experiments. Then, the improved PSO algorithm is applied to solve two different practical VRPs, which proves that it also has practical application value. Finally, according to the research content of this paper, a preliminary drug distribution system is designed and implemented, and the practical application of the PSO algorithm proposed in this thesis is realized.

Keywords: particle swarm optimization(PSO) , Levy flight, reverse learning, vehicle routing problem(VRP), drug distribution system

目 录

第一章 绪 论

1.1 研究工作的背景与意义

优化问题是一种在工程设计中的常见问题，指的是在满足一些特定的约束条件的情况下,选择一组变量，使得目标达到最优值。优化问题一般都能用数学规划的形式来表示，但有时由于问题的规模过于庞大或者问题本身过于复杂，在现有的条件下很难达到理论最优值，故有时也只能最大程度的做出目前情况下的最优决策。简单来讲，解决优化问题的过程就是在满足特定条件的前提下，在大量可行方案中找出最优的那一个的过程。

优化问题在物流等领域中的应用有许多。例如，在物流运输的过程中，生产厂商需要对多个用户需求点，根据自身的车辆以及人力状况来安排每一辆运输车的行驶路线以及货物的运输顺序，在尽可能的达到客户要求的前提下，使自己花费的运输成本最小。这样一来，一方面可以提升物流过程的整体效率，提高客户的满意程度；另一方面还能为企业节省运营成本。

群体智能算法自从被提出后，便被广泛应用到最优化问题的求解上，并引起了国内外学者的研究。粒子群优化算法（Particle Swarm Optimization, PSO）便是其中之一，其算法简单、参数少、收敛快等特点使其在求解最优化问题上的表现出色。在近几年内也被应用到各种领域，如组合优化〔1〕、舆情预测[2]、气温预测[3]、机器人路径规划[4]等。同时还有其他的智能优化算法也被广泛的运用，如遗传算法（Genetic Algorithm, GA）、蚁群算法（Ant Colony Optimization, ACO）、狼群算法（Wolf Pack Algorithm, WPA）等等。尽管这些算法不能保证一定能获得所求问题的理论最优解，但在求解较大规模的问题时，能够找到问题的满意解。

虽然目前关于粒子群算法的研究还在不断的更新，但算法本身以及其实际应用还有很多可以挖掘之处。因此，本文将基于目前已有的研究，在粒子群算法的领域中探究如何更好的改进算法本身，让粒子群优化算法更加完善，从而扩展了算法的应用领域。

1.2 国内外研究现状

1.2.1 粒子群优化算法发展现状

粒子群优化算法自问世以来，相关研究论文也不断地涌现，研究方向也各不相同。本文总结了学者们对于粒子群算法改进的几个大方向，主要集中在以下三个方面。

（1）粒子群算法理论研究

粒子群算法的理论研究也就是对粒子群算法本身的研究。Shi[5]等人将惯性权重加入了粒子群算法的速度更新公式中，并通过大量实验测试了惯性权重对于粒子群算法寻优能力的影响。实验也表明，惯性权重的大小直接决定了整个粒子群在寻优过程中的策略，对于粒子跳出局部最优以及扩大搜索范围有着重要的作用。Chaturvedi[6]等人通过对学习因子 c_1、c_2 的研究，通过对比实验得出学习因子的值在[0,2]时，算法能够取得较好的结果。Jin[7]等人提出了一种基于健康度的粒子群算法（HPSO），该算法通过对每一个粒子计算健康度来区分正常粒子和异常粒子，对于每一代中的异常粒子，通过对其个体最优值进行变异操作，提高其健康度。实验也表明了 HPSO 有较好的搜索能力。

（2）粒子群算法性能改进

粒子群算法本身在求解问题虽然收敛速度较快，但又有着容易陷入局部最优值的缺点，所以学者们不断从如何提高算法的收敛精度与速度的角度出发，对算法进行各种改进。Liang[8]等人提出的综合学习粒子群算法（CLPSO），该算法在更新一个粒子的飞行速度时，使用了整个种群粒子的历史最优信息，因此算法的种群多样性保持的比较好，较好地阻止了算法在搜索过程中的早熟现象。特别是对于多峰优化问题，CLPSO 算法的优化效果是比较理想的。Zhan[9]等提出了自适应粒子群优化算法（APSO）。APSO 具有快速收敛的全局搜索能力，他依据种群的空间分布信息和粒子的优劣实时计算并确定粒子群的进化状态，并且根据种群的不同状态调整算法中的各种参数，从而使算法有较好的搜索性能和收敛速度。APSO 还在算法中引入精英学习策略，使得算法可以跳出局部最优。相关的实验结果也证明了 APSO 具有较好的全局优化能力。王永贵[10]等人针对算法易陷入早熟收敛的问题，修改了惯性权重的变化规律，使其随着迭代呈指数衰减，并且对满足特定条件的粒子进行分裂，使得种群在后期也能有更丰富的多样性，避免早熟收敛。夏学文[11]等人提出了一种具备反向学习和局部学习能力粒子群优化算法，该算法将每次迭代中的最差粒子位置和每一个个体的历史最差位置记录下来，当算法陷入停滞时，对一部分粒子采用反向学习策略，向这些较差的信息学习，使粒子脱离局部最优。同时利用种群中较优粒子的位置引导粒子在局部进行精细搜索，有效的提高了算法的局部搜索能力。

（3）粒子群算法应用研究

粒子群算法由于其易于实现、收敛快等特点，一直以来被广泛应用于各个领域。霍林[12]等人将改进的粒子群算法应用于安卓的恶意应用程序检测中，该文章首先提取了安卓应用程序的信息作为特征，但这些信息的数量十分庞大，特征维

数可高达三到四万维。为了消除冗余的特征，文章采用了一种改进的离散二进制粒子群算法进行混合式特征选择。实验表明，该算法具有较好的收敛结果，能够有效使得安卓恶意应用程序的检测正确率得到提升。张倩[13]等人为了解决农业水资源的配置问题，建立了优化模型，采用了改进的粒子群算法对模型求解。实验结果也表明优化后的经济效益相较于优化前有明显增长，为区域水资源的优化配置解决提供了新的思路。刘彩霞[14]等人将模糊推理技术结合到粒子群算法中，提出了一种改进粒子群算法，并将其应用到机器人路径规划的中。首先，文章处理了障碍物，优化了环境模型；其次，对粒子群算法的参数进行了自适应的动态调整，改进后的算法在复杂环境下路径长度、平滑度和运行时间方面分别提高了17%、14%、7%。张进峰[15]等人对航运中的航速优化问题进行了探讨，通过研究风浪对船舶航行速度的影响，建立了优化模型，并使用改进的多目标粒子群算法对模型求解，实验表明优化模型能有效地降低排放量并且降低运营成本。

1.2.2 车辆路径问题研究现状

在现代物流管理的诸多问题之中，车辆路径问题的研究涉及到很多个学科，在许多领域中都备受学者们的关注。而且车辆路径问题对于很多实际问题的解决有着至关重要的意义，所以也成为了研究的热门之一，自从它被提出到现在，国内外都有着许多的学者在进行着不同方面的研究。

（1）研究起源

车辆路径问题（Vehicle Routing Problem, VRP）[16]是指在已知一些信息的情况下，通过添加一定的约束条件，对运输车辆进行合理的行驶路线以及出行时间的安排等，从而达到既定的目标（如时间最短，成本最小，路程最短等）。VRP领域研究的开端要追溯到 1954 年，Dantzing, Fulkerson 和 Johnson[17]研究了较大规模的旅行商问题（Traveling Salesman Problem, TSP），并且提出了解决方法。1959 年，Dantzing 和 Ramse[18]首先提出了“车辆调度问题（Truck Dispatching Problem）”的概念，该文章讨论了实际生活中的汽油配送问题，研究了汽油配送的车辆如何调度才能使得车辆行驶的总距离最短。Clarke 和 Wright [19]在车辆调度问题的基础之上，将该问题推广到物流运输中经常遇到的优化问题，这就是现如今被我们所熟知的车辆路径问题。在此以后，经典 VRP 开始被广泛地应用在各种领域，包括应用数学、组合优化、物流、交通、计算机等领域。现如今，VRP 已经被扩展成许多种不同的类型，并应用于各种不同的场景，其求解的方法也在不断地涌现和发展。

（2）VRP 变式的发展

经典的 VRP 只在一些简单的条件下计算最优结果，但在实际应用中，很多因素都需要考虑在内，这就使得经典 VRP 并不适用于大多数实际情况。例如在运输的过程中，客户的需求发生变化或者出现新的需求，这些信息都是在路径规划之后才出现。经典 VRP 的模型在规划路径时无法将这些信息考虑在内，所以各种 VRP 的变式也相继的研究出来。根据问题不同，可将 VRP 分为两类：静态确定性 VRP 和动态随机性 VRP。

静态确定性 VRP 在被提出后便成为当时研究的主流。静态确定性 VRP 就是指所有的输入信息都是已知的，且运输任务一旦开始就不再发生变化，所有已知的信息不会改变[20]。但是静态确定性 VRP 模型并不能满足现实中的实际需要，现实生活中的 VRP 会有更加复杂的情况，如交通因素造成的行驶时间变长，随机出现的新客户以及随机改变的客户需求等因素都导致静态的 VRP 不能应用于实际中。因此，学者们开始将研究的重点转向动态随机性 VRP。1977 年，Wilson 和 Colvin[21]首先提出了单车型动态需求的 VRP。早期关于随机性 VRP 的研究并不是太多，但随着之后计算机技术的发展，由于计算机强大的计算能力和编码能力可以帮助学者们进行许多复杂的计算，随机性 VRP 的研究开始爆发式增多。常见的随机动态 VRP 的研究主要在三个方面：随机顾客、随机时间、随机需求。

步入 21 世纪之后，能源与环境问题逐渐成为了全球范围的热点话题，人类活动产生的温室气体二氧化碳（CO_2）也在逐步的导致全球气候的变化。运输行业作为能源消耗和碳排放的大户也引起了学者们的关注。如何在新的背景下解决 VRP 问题也逐渐成为了学者们研究的新方向。2007 年，Palmer[22]首次将碳排放量纳入 VRP 的优化目标中，提出了带时间窗且以碳排放和距离为目标的车辆路径优化模型。从此 VRP 出现了一个以绿色环保为目标的新分支，在现阶段成为了研究的热点。

目前国内也在针对不同条件下的 VRP 进行着多方面的研究。段征宇[23]等在随机时变最优路径问题的基础上提出了一种带硬时间窗的随机时变车辆路径问题的多目标鲁棒优化模型，同时考虑路网交通状态的时变性和随机性。该模型在知道路段行程时间的波动范围的情况下，可以保证车辆在时间窗的范围内送达。文中还设计了一种蚁群算法用来求解该类问题的多目标优化模型，取得了较好的结果。何东东[24]等从低碳环保的角度考虑，通过数学方法将车辆的碳排放量和油耗量进行了量化，并引入带时间窗的 VRP 中，在目标函数中加入了车辆在运输过程中产生的废气污染，进而提出了一种带时间窗的绿色 VRP 模型，并且针对问

题设计了一种改进的禁忌搜索算法，仿真实验的结果证明了该算法解决这类问题时十分有效，为未来的低碳运输提供了技术指导。但是文章是在闭合式的条件下研究该类问题，并没有讨论开放式的情况。王杨[25]等构建了一种基于时变速度的多配送中心多车型联合配送的车辆路径问题模型，基于一种改进的禁忌搜索算法对该问题进行了优化。文章最后采用了北京市公铁联运城市配送案例进行了算法和模型的有效性验证，结果证明了运用该模型能有效地减少成本，该模型更加符合企业中多配送中心模式，优化补货车辆、配送站点车辆以及行车路线的实际需要。杨翔[26]等针对有模糊需求，多中心和开放式三重约束的 VRP 建立了模型，并且提出了新的禁忌搜索算法求解该问题，该算法包含两个阶段，在第一阶段求解包含全部客户的旅行商问题，并作为第二阶段的初始解，并采用合适的编码方式来保证两个阶段解的兼容，最后通过实验证明该算法在可接受时间内能对这类问题进行有效的求解。

（3）VRP 的解决算法研究

VRP 的主流求解方法一共有两种，即精确算法和启发式算法。精确算法适用于求解静态确定性 VRP，但是随着问题的实际情况越来越复杂，需要考虑的条件越来越多，精确算法的计算难度也越来越大；而相对于精确算法求出问题精确的最优解，启发式算法只能求出问题的近似解，所以衡量算法优劣的指标也就分为了速度与精度两个方面。速度体现为算法计算问题时求出最终结果所需要的时间，精度则体现在求出的解是否更加接近于最优解，在 VRP 中则体现为是否离优化目标更近。

随着研究的深入，“元启发”这个概念也被提出，用来命名那些解决组合优化问题的算法，也就是上文中提到的智能算法。Gendreau[27]等将元启发式算法分为以下几类：（1）禁忌搜索算法（2）模拟退火算法（3）确定性退火算法（4）遗传算法（5）蚁群算法（6）神经网络算法。在元启发式算法中，提出较早的算法有遗传算法、蚁群算法、粒子群算法等，对这几种算法的研究相对较多。近期也提出了人工鱼群算法、蝙蝠算法、狼群算法等，学者们也在对这些算法进行着各方面的研究。由于元启发式算法适用性强，能解决大多数情况下的问题，所以成为解决车辆路径问题的首选，国内外学者一直也对其进行着各方面的研究。

金淳[28]等人通过结合领域搜索算法，改进状态转移规则，提出了一种改进分布式多 agent 蚁群算法来求解带时间窗的车辆路径问题，相较于其他算法提高了求解效率，降低了求解的复杂度。孙小军[29]等人建立了一种双目标带时间窗的动态 VRP 模型，并提出了一种新的蚁群算法对模型优化，该算法引入了交通拥堵因子，能够提高计算效率。戚远航[30]等人提出了一种离散的蝙蝠算法来求解带时

间窗的车辆路径问题，算法将约束条件转化为罚函数并加入到目标函数中，并且引入了随机插入策略、普通插入搜索等策略来扩大搜索空间，使算法能更快的收敛。石建力[31]等人分析了需求随机出现和变化并需要分批运送情况下的 VRP，建立了带修正的随机 VRP 模型，并将粒子群算法与局部搜索算法相结合对文中讨论的模型进行了优化。虽然计算效率较低，但是能在大多数情况下提高解的精度。周蓉[32]等人研究了车辆路径问题中装货和卸货共存的情况，并提出了相应的规划模型，将降低配送成本、减少车辆行驶的距离、最小化使用车辆数同时做为优化目标。然后针对该问题提出了一种自适应的并行遗传算法，该算法将种群一分为二采用并行进化的策略，实现了全局和局部的同步搜索，大大提高了搜索的完整度。根据文中的实验结果证明了算法是可行有效的。

Che[33]等人分析了人工鱼群算法的收敛速度不高，计算的结果不稳定等缺点，提出了一种改进的混合人工鱼群算法用于解决较大规模的车辆路径问题。最终实验表明该算法能够提高解决问题的速度并且提升解的精度。Dechampai[34]等人提出了两种不同的启发式算法用来解决提送货一体化的车辆路径问题，两种算法的不同点在于其中一种算法在优化问题前会先对需求点进行分类。实验结果表明先分类的算法所得到的结果较好。Gao[35]等人针对随机需求且负载可变的绿色车辆问题提出了一种改进的模拟退火算法进行求解，该算法使用 K-means 聚类法进行局部搜索。实验结果表明该算法比原始的模拟退火算法能得到更优的解。

综上所述，国内外学者对 VRP 的分类、模型以及求解算法都进行了各种各样的深入探索，且研究结果丰富。为了更加符合实际应用场景，针对车辆路径问题各种变式的研究也发展迅速，研究同时也在考虑更多的实际因素。各种元启发式算法的研究也对不同场景下车辆路径问题的解决提供着理论基础和指导作用。

1.2.3 PSO 在 VRP 中的应用

PSO 作为典型的启发式算法，对于解决 PSO 这种 NP 难问题有着较好的求解结果。Gan[36]等人将碳排放因子引入到车辆路径问题中，提出了一种新的 VRP 模型并使用 CLPSO 对其进行求解，并将结果与 GA 求解的结果进行对比。实验表明 CLPSO 算法有着较好的求解效果。Okulewicz[37]等人提出了一种两阶段的多种群 PSO 算法，并将其应用于动态 VRP 问题的求解中，并在标准测试函数上与其他最新提出的改进 PSO 算法进行比较，结果表明文章所提出的算法得到的结果均值和收敛速度均优于其他算法。Lagos[38]等人针对考虑提送一体的带时间窗 VRP，提出了一种改进的粒子群优化算法，实验结果表明粒子群算法能在可接受的时间范围内求出问题的解。

1.3 研究内容和方法

1.3.1 研究内容

粒子群算法虽然有着实现简单、收敛速度快的优点，但也因此有着容易丧失种群多样性，陷入局部最优值，导致搜索的精度不高的缺点。故本文针对这些缺点对粒子群算法进行改进研究，并应用于车辆路径问题求解的实际应用中，以此来验证算法的有效性。本文的具体研究内容如下：

（1）文献综述和理论基础

首先，通过对相关文献的阅读和梳理，对现阶段粒子群算法、车辆路径问题的研究进行了总结归纳。其次，对粒子群算法的概念和特点以及相关知识和车辆路径问题的基本理论知识进行了阐述与概括。

（2）算法的改进与应用

针对粒子群算法容易陷入局部最优的缺点，本文提出另一种基于莱维飞行的反向学习粒子群算法(Particle Swarm Optimization Algorithm with Reverse Learning and Levy Flight, RLFPSO)。通过对粒子的位置是否进行了更新迭代进行检测，若粒子的个体最优值未更新次数超过了一定的数值，便认为该粒子陷入了停滞，采用反向学习更新其位置。使该粒子向个体最差位置进行学习，学习的步长采用莱维分布的随机变量获得。通过对 7 个标准测试函数的仿真实验，并与标准粒子群算法以及其他改进粒子群算法进行结果对比，表明基于莱维飞行的粒子群算法搜索精度高，收敛速度快。最后，将上述的粒子群算法应用于带时间窗约束的车辆路径问题(Vehicle Routing Problem with Time Window, VRPTW)。通过仿真实验测试并与其他算法在此问题上的求解结果进行对比，验证 RLFPSO 的收敛速度和精度，证明该拥有一定的实用价值。

（3）系统的设计与实现

设计并实现一个简单的药品配送系统。并将文章所提出的 RLFPSO 算法应用到系统中，对实际生产中的问题进行求解，证明该算法在实际生产中的应用价值。

1.3.2 研究方法

本论文主要采用文献分析和仿真实验相结合的方法进行研究，所用到的研究方法如下：

（1）文献分析法。通过阅读相关的书籍和与研究内容相关的文献以了解粒子群算法和车辆路径问题的研究现状和发展趋势，并对粒子群算法的原理、特点、改进方向等进行总结。

（2）数学建模法。将研究的实际问题根据其特点抽象为数学模型，并且定量

的分析模型中的各种影响因素，一遍对问题的研究与求解。

（3）仿真分析法。使用 MATLAB 软件对问题的优化结果进行计算，并进行数据的比较和分析，最终得出结论。

1.4 论文组织结构

本论文总共包含 6 个章节，每章的具体内容如下：

第一章，对本文研究的问题背景，研究目的和意义做了简短的分析，其次总结了粒子群算法和车辆路径问题的研究现状，最后对本文的主要工作和组织结构进行了介绍。

第二章，对粒子群算法相关的理论基础进行了简单介绍，并总结了粒子群算法的常见改进方向。同时也介绍了优化算法中几种常见的标准测试函数。

第三章，针对粒子群算法容易陷入局部最优的缺点，提出一种基于莱维飞行的反向学习粒子群算法，介绍了该算法的思想和原理，并在标准测试函数上将该算法与标准粒子群算法以及其他算法进行了仿真比较实验，证明了该算法在收敛速度和精度上的优势。

第四章，将提出的算法应用到带时间窗约束的车辆路径问题中。首先对车辆路径问题进行了简单的阐述，并介绍了带时间窗的车辆路径问题。然后介绍了粒子群算法在应用过程中的编码方案、算法步骤。最后给出了两个不同的实际问题案例并使用 RLFPSO 算法进行求解。验证了 RLFPSO 在该问题中具有较好的应用价值。

第五章，在上文研究的基础之上，设计并实现一个初步的物流配送服务系统，对整个优化模型的优化过程以及结果进行了可视化，并对系统进行了简单的介绍。

第六章，对本论文所做的研究工作进行总结，提出论文中的不足以及今后如何改进，最后展望了论文进一步研究的方向。

第二章 相关理论综述

2.1 粒子群算法简介

粒子群优化算法是 Kennedy 等在 1995 年所提出的一种基于群体智能理论的启发式算法[39]，它的实质是模仿自然界中的鸟群、昆虫的群居行为。这几类种群之间的个体都会通过信息交流来共同寻觅食物，每一个个体通过将自身所学习到的知识或经验与种群中其他的个体共享，同时也获得其他个体的经验，并基于这些信息不断地调整自己的行为模式。算法的总体思路是将问题的解空间作为鸟群飞行的空间，根据特定问题的不同约束条件，解空间的范围也会有所限制。解空间中的每一个点就代表着一个问题的可行解。再将每一只鸟都抽象为一个粒子，这个粒子在解空间中无质量无体积，但是有速度和位置，每一个粒子都表示问题的一个候选解，食物即是解空间中最优解的位置。那么整个优化的过程就被类比为鸟群在整个解空间中寻找食物的过程。在优化的过程中，模仿鸟类觅食时的行动规则，为空间中的每一个粒子都制定一定的运动方式，使得粒子在优化过程中呈现出与鸟群觅食相同的特征。然后使每一个粒子都具有记忆个体历史最优位置的能力，且粒子与粒子之间的信息能够共享，这样就可以实现种群中每一个个体的自我学习和相互学习，利用个体最优信息和种群最优信息进行种群的迭代来实现问题的优化求解。

2.1.1 基本粒子群优化算法

在基本粒子群算法中，粒子的位置对应问题的一个可行解。粒子的适应度就是讲该可行解带入到目标函数中所得到的函数值。粒子的速度决定粒子在下一次迭代过程中的飞行方向和距离。

假设在一个 D 维的目标搜索空间中，有一个种群规模为 N 的粒子群落，即种群中一共有 N 个粒子，其中，第 i 个粒子的位置表示为一个 D 维的向量$X_i = (x_{i1}, x_{i2}, \dots, x_{iD})(i = 1,2, \dots, N)$，其飞行速度可以表示为$V_i = (v_{i1}, v_{i2}, \dots, v_{iD})(i = 1,2, \dots, N)$。每个粒子的位置在目标搜索空间中都代表一个问题的可行解。在开始时，种群初始化为一组随机的解，即随机地分布在整个搜索空间中。在算法执行时通过下面两个式子，对粒子的状态进行更新：

$$v_{id} = wv_{id} + c_1 rand_1(p_{id} - x_{id}) + c_2 rand_2(p_{gd} - x_{id}) \quad (2\text{-}1)$$

$$x_{id} = x_{id} + v_{id} \quad (2\text{-}2)$$

上式中，v_{id}表示第 i 个粒子在第 d 维上的速度，x_{id}则表示第 i 个粒子在第 d 维上的位置。w 表示惯性权重，该值越大表示粒子的探索能力越强，即全局搜索能

力越强；越小表示粒子的开发能力越强，即局部搜索能力越强。c_1和c_2为学习因子，也被称为加速常数(acceleration constant)，c_1表示个体学习因子，c_2表示社会学习因子。$rand_1$和$rand_2$是相互独立的[0,1]区间上的随机数。p_{id} 表示第 i 个粒子个体到目前为止搜索到的最优解的位置，p_{gd} 表示整个种群到目前为止搜索到的最优解的位置。式（2-1）右边可以分为三部分，第一部分是粒子的惯性，表示粒子有保持上一代时飞行速度速度的趋势；第二部分是个体最优位置引导粒子移动，反映粒子向自身经验进行学习，代表粒子向自身历史最优位置飞行的趋势；第三部分是种群最优粒子引导各个粒子进行移动，反映了种群之间的信息交换，代表粒子有向种群最优粒子飞行的趋势。

为了避免某个粒子的飞行范围过大，需要对粒子的每一个维度设置一个最大飞行速度$V_{\max}^d$，当粒子在第 d 维上的速度大于$V_{\max}^d$或小于-$V_{\max}^d$时，设置粒子在第 d 维上的速度为$V_{\max}^d$或-$V_{\max}^d$。$V_{\max}^d$的取值一般为该维度上决策变量的取值范围的10%-20%。

算法流程图如下图 2-1 所示：

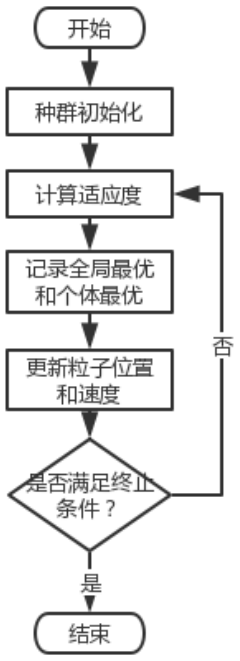

图 2-1 基本 PSO 算法流程图

2.1.2 标准粒子群优化算法

标准粒子群算法(Standard Particle Swarm Optimization, SPSO)[5]是由 Shi 和

Eberhart 在 1998 年提出的。该算法与基本 PSO 算法最大的不同在于采用了线性递减的惯性权重的策略。该文章通过大量实验研究表明惯性权重对于算法的搜索能力有着至关重要的影响。当 w 越大时，算法的全局搜索能力越强；当 w 越小时，算法的局部搜索能力越强。采用线性递减的惯性权重策略，可以使算法在初期扩大搜索范围，进行广而粗的搜索，尽可能的搜索到解空间的可能的最优解；在算法末期较小的 w 使算法在当前最优解附近进行精细搜索，挖掘可能的更优解。该算法中粒子速度更新公式为：

$$v_{id} = wv_{id} + c_1 rand_1(p_{id} - x_{id}) + c_2 rand_2(p_{gd} - x_{id}) \tag{2-3}$$

其中，

$$w = w_{\max} - \frac{(w_{\max} - w_{\min})}{T_{\max}} \times t \tag{2-4}$$

式（2-4）中，$w_{\max}$ 和 $w_{\min}$ 分别是惯性权重的最大值和最小值边界。$T_{\max}$ 是算法的最大迭代次数，t 为算法的当前迭代次数。

自从惯性权重被引入粒子群算法，学者们就对这个参数进行了大量的研究。Nickabadi[40]等人利用种群的搜索成功率来缺东搜索空间中粒子的位置，使得算法中的惯性权重的值根据需要进行自适应调节。结果也表明，这种调整方法对提高算法效率非常有用。

2.2 粒子群优化算法的常见改进方式

粒子群算法自从被提出以来，学者们便纷纷对其进行研究改进，改进的方法也各不相同。通过对大量文献的阅读，本文总结出粒子群算法的几种常见的改进方向。

2.2.1 参数改进

粒子群算法特点是参数不多，但有些参数对算法的性能有着重大的影响，对粒子群算法的改进也集中于这几个参数中：惯性权重 w、学习因子 c_1 和 c_2。Farooq[41]等人提出了一种新的粒子群算法，对粒子的每一维赋予不同的惯性权重的值。同时为了算法的稳定性，加速因子也随着惯性权重的不同而变化。最后通过实验证明，该算法在动态和静态环境下都大大地提高了算法的效率。Spavieri[42]等人将惯性权重的值和粒子的适应度关联起来，将每一代粒子的最优适应度和最差适应度的值和当前世代每一个粒子的适应度值进行计算，分别算出每一个粒子的惯性权重的值，这样能更好的根据每一个粒子的具体情况来制定每一个粒子的搜索策略。实验结果也证明，该算法在 6 个标准测试函数上寻优结果优于其他三个 PSO 算法。

2.2.2 混合 PSO 算法

由于 PSO 算法自身有缺陷，所以通过与其他算法相结合可以弥补不足，发挥多种算法的优势，取得更好的结果。Garg[43]等人将遗传算法与粒子群算法相结合，提出了一种混合算法(PSO-GA)，通过在粒子群算法中使用遗传算法的“交叉”与“变异”的操作帮助粒子群跳出局部最优值，平衡了算法的全局搜索能力和局部搜索能力。Elloumi[44]等人将一群算法与粒子群算法相结合，在粒子群算法中加入了蚁群算法的搜索策略，提出了一种混合算法(ACO-PSO)。该算法在解决路径规划的问题上有较好的表现。Wei[45]等人提出了一种基于狼群活动范围的粒子群算法，狼群算法主要模拟了自然界中的狼群在狩猎过程中的分工合作和竞争机制，该算法分析了狼群算法中活动范围的更新、头狼的生成和更换以及狼群的攻击、搜索等行为，并与粒子群算法相结合。实验表明，混合算法能够极大的提高搜索的速度。Liu[46]等人提出了一种将粒子群和人工蜂群混合的算法。在粒子群算法的迭代过程中，将整个种群分为两个子种群，其中一个子种群采用粒子群算法的搜索策略进行搜索，另一个子种群使用人工蜂群的算法来引导粒子运动。在每一次迭代后将两个子种群所求出的最优解进行对比，将较优的那个作为全局最优位置。结果表明，该算法具有较好的优化性能。

2.2.3 多种群 PSO 算法

在标准粒子群算法中，所有粒子都跟随个体最优位置和群体最优位置飞行，导致整个粒子群快速收敛，种群多样性下降，算法陷入局部最优。在标准粒子群算法中，整个种群只向自身和最优粒子学习，而其他粒子的信息没有得到充分利用，学者们从这个角度也提出了粒子群算法的改进方案。

Wang[47]等人在 PSO 整个进化过程中，把每一代粒子按适应度划分为了两个子种群，并分别采用不同的搜索策略，两个子种群间的信息可以交换。实验证明，这种改进方法提高了信息的利用率，从而使得算法效率得到提升。Xia[48]等人将整个种群划分为许多个小种群，来保持种群多样性。此外使用了自适应重组算子来控制不同种群粒子间的重组和信息交互。通过实验结果的比较证明了该算法具有较好的逃离局部最优的能力，提高了算法的准确性。

2.3 常用的标准测试函数

一个新的或者改进的智能算法被提出后，通常需要在标准测试函数集上做相应的仿真实验来证明算法的可行性和有效性。本节下面将列举几个常见的标准测试函数。其中函数 f_1 和 f_6 为单峰函数，而其余均为多峰函数。多峰函数的全局存

在许多局部极值点很难求得全局最优解。所有这些测试函数的理论最优解为 X=(0,0,...,0)，理论全局最优值为 0。

（1）Sphere 函数：

$$f_1(x)=\sum_{i=1}^{n}x_i^2\ ,\ (-100\le x_i\le 100) \tag{2-5}$$

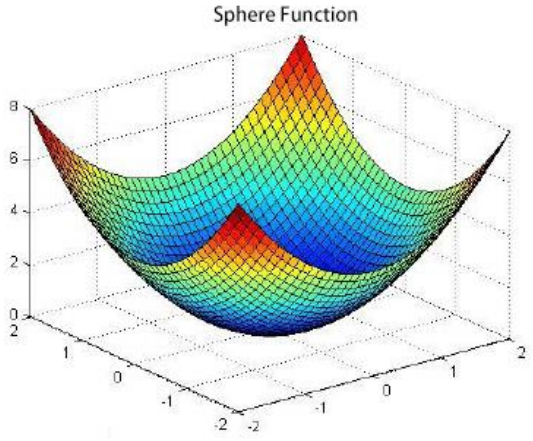

图 2-2 Sphere 函数

（2）Quadric 函数：

$$f_2(x)=\sum_{i=1}^{n}(\sum_{j=1}^{i}x_j)^2\ ,\ (-100\le x_i\le 100) \tag{2-6}$$

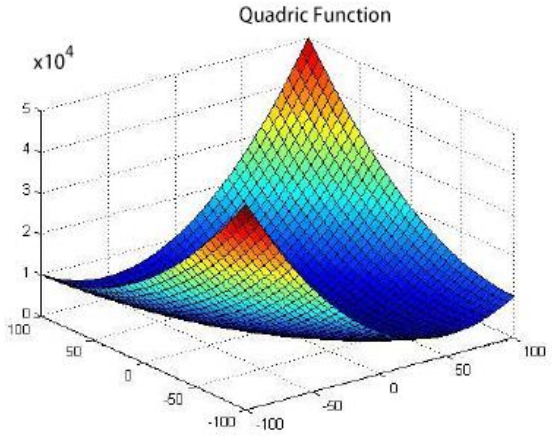

图 2-3 Quadric 函数

（3）Ackley 函数

$$f_3(x) = -20\exp\left\{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right\} - \exp\left\{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right\} + 20 + e \ , \quad (-32 \le x_i \le 32) \tag{2-7}$$

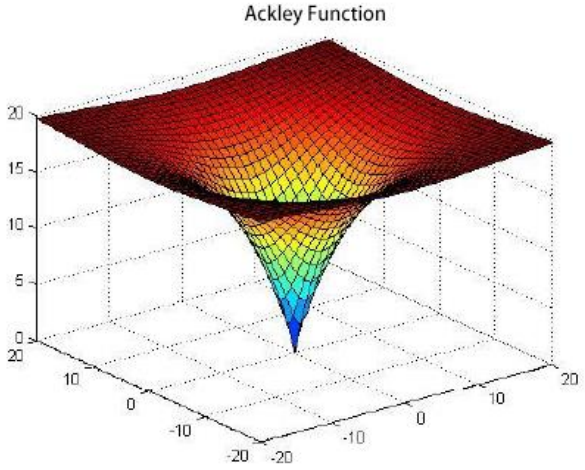

图 2-4 Ackley 函数

（4）Griewank 函数

$$f_4(x) = \frac{1}{4000}\sum_{i=1}^{n}x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1 \ , \ (-600 \le x_i \le 600) \tag{2-8}$$

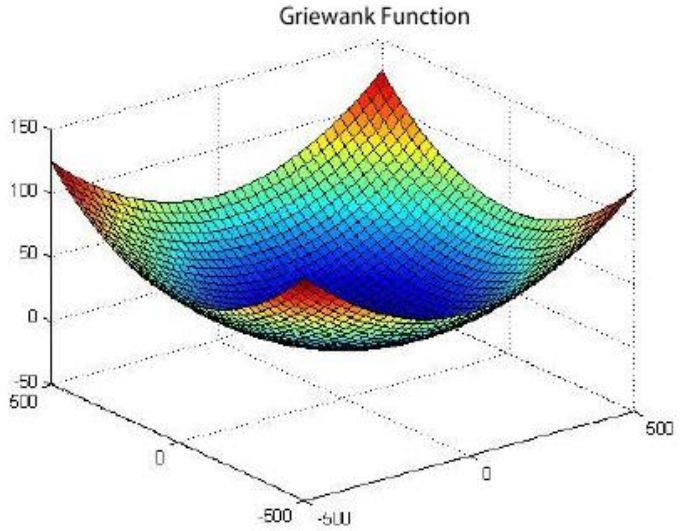

图 2-5 Griewank 函数

（5）Rastrigin 函数

$$f_5(x_i)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10]\ ,\ (-5.12\le x_i\le 5.12) \tag{2-9}$$

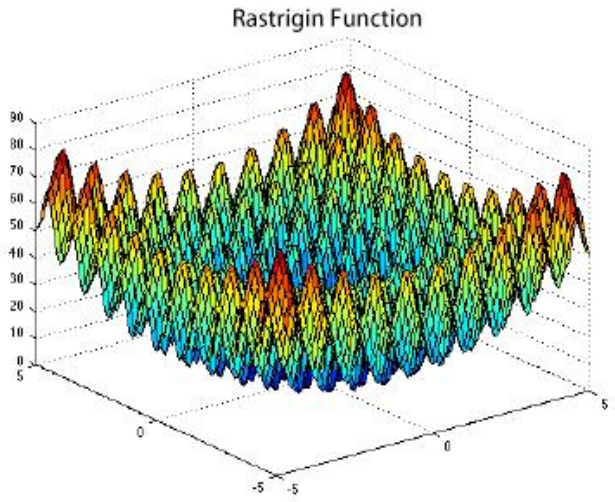

图 2-6 Rastrigin 函数

（6）Rosenbrock 函数

$$f_6(x_i)=\sum_{i=1}^{n}(100(x_{i+1}-x_i^2)^2+(x_i-1)^2)\ ,\ (-10\le x_i\le 10) \tag{2-10}$$

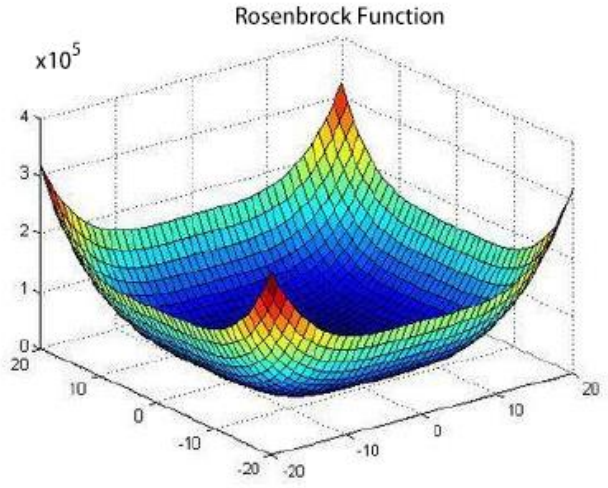

图 2-7 Rosenbrock 函数

（7）Schaffer 函数

$$f_7(x，y)=0.5+\frac{(\sin^2\sqrt{x^2+y^2}-0.5)}{1+0.001(x^2+y^2)^2}，(-100\le x_i\le 100) \tag{2-11}$$

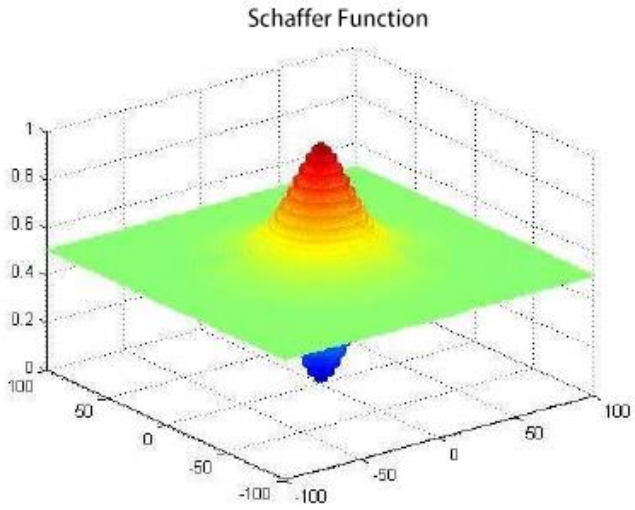

图 2-8 Schaffer 函数

2.4 本章小结

本章主要介绍了粒子群算法的有关理论知识，介绍了基本 PSO 算法和标准 PSO 算法，并对几种常见的 PSO 算法改进方向做了简单总结。最后给出了几个常用的智能算法标准测试函数和其三维图像。

第三章 基于莱维飞行的反向学习粒子群算法

3.1 RLFPSO 算法

粒子群算法虽然收敛速度较快且易于实现，但其本身也存在易陷入早熟收敛的问题。针对粒子群算法的不足，本文提出了一种基于莱维飞行和反向学习的粒子群算法(Particle Swarm Optimization with Reverse Learning and Levy Flight, RLFPSO)。

3.1.1 反向学习机制

标准粒子群算法在整个迭代的过程中，整个种群都只是朝着自身历史最优位置和整个种群的历史最优位置学习，所以整个种群是趋于同一化的，所以才造成了粒子群算法本身的缺陷。一方面，在实际的自然界中，每一只个体鸟的运动不会只受到最优个体的影响，种群中每一只鸟都可能对其他个体产生影响。而在粒子群算法中，每一个粒子的每一代信息都会被记录，只要充分利用这些信息，就可能使得算法效率得到提高。基于此，本文将反向学习的机制引入了粒子群算法中。当粒子由于局部最优值而陷入停滞时，让该粒子向其个体历史最差位置进行学习，快速引导这些粒子逃离最优值，提升种群的多样性，提高算法的搜索能力。

在种群迭代的过程中，若检测到粒子的个体最优位置在一定的迭代次数内没有更新，则可以判断该粒子陷入了停滞状态。此时对其使用反向学习，其他粒子的学习方式不变。第 i 个粒子的个体历史最差位置表示为 $W_i=(w_{i1},w_{i2},\ldots,w_{ij},\ldots,w_{iD})$。则粒子在进行反向学习时的速度更新公式为

$$\begin{aligned} v_{id} &= wv_{id}+c_1 rand_1(p_{id}-x_{id})+c_2 rand_2(p_{gd}-x_{id}) \\ &+c_3 rand_3(x_{id}-w_{id}) \end{aligned} \tag{3-1}$$

其中，w_{id} 表示第 i 个粒子在该时刻，个体历史最差位置在第 d 维上的值；c_3 表示反向学习的学习因子。

3.1.2 莱维飞行

莱维分布是 19 世纪 30 年代被法国数学家 Levy 提出的一种概率分布，一经提出就有需对学者对其进行了大量研究。目前，已经有文章证明了自然界中有许多昆虫、鸟类如蜜蜂、果蝇等的觅食轨迹都符合莱维飞行模式[49]。

近年来，莱维分布被应用到优化领域中，如布谷鸟算法[50]。莱维飞行[51]是一种随机搜索路径，它的轨迹服从莱维稳定分布。其飞行过程由许多短距离搜索中偶尔穿插长距离搜索组成。

采用莱维飞行可以提高算法的全局搜索能力，防止种群多样性减少，因此使用莱维飞行与粒子群算法结合可以有效防止其容易陷入局部最优值的缺点。

由于莱维分布非常复杂，目前还没有完全实现，因此常用 Mantegna 算法进行模拟。Mantegna 算法模拟莱维飞行的步长 S 的计算公式如下：

$$S=\frac{u}{|v|^{1/\beta}} \tag{3-2}$$

其中，β 是莱维分布的参数，通常取值为 1.5，u 和 v 是服从正态分布的随机数，

$$\begin{cases} u \sim N(0,\sigma_u^2) \\ v \sim N(0,\sigma_v^2) \end{cases} \tag{3-3}$$

其中，

$$\begin{cases} \sigma_u=\left\{\dfrac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\Gamma(1+\beta/2)\beta 2^{(\beta-1)/2}}\right\}^{1/\beta} \\ \sigma_v=1 \end{cases} \tag{3-4}$$

最终，粒子运动的步长为，

$$stepsize=\alpha\times S \tag{3-5}$$

其中α为调节因子，根据不同的具体问题调节步长，这样才不会使飞行的步长过大或过小，从而保持在合适的范围内。

莱维飞行有着很强的搜索能力和更大的搜索范围。它与粒子群算法相结合能够明显提高算法的全局寻优能力。下面是在 MATLAB 中将 Mantegna 算法模拟出的莱维飞行 1000 步的图像，如图 3-1 所示。

将莱维飞行与反向学习机制相结合，使用莱维分布获得反向学习的步长，得到 RLFPSO 的速度更新公式为

$$\begin{aligned} v_{id} &= wv_{id}+c_1 rand_1(p_{id}-x_{id})+c_2 rand_2(p_{gd}-x_{id}) \\ &+c_3 r_3(x_{id}-w_{id})\times stepsize \end{aligned} \tag{3-6}$$

式（3-6）中，w_{id}是第 i 个粒子的历史最差位置的第 d 维的值。

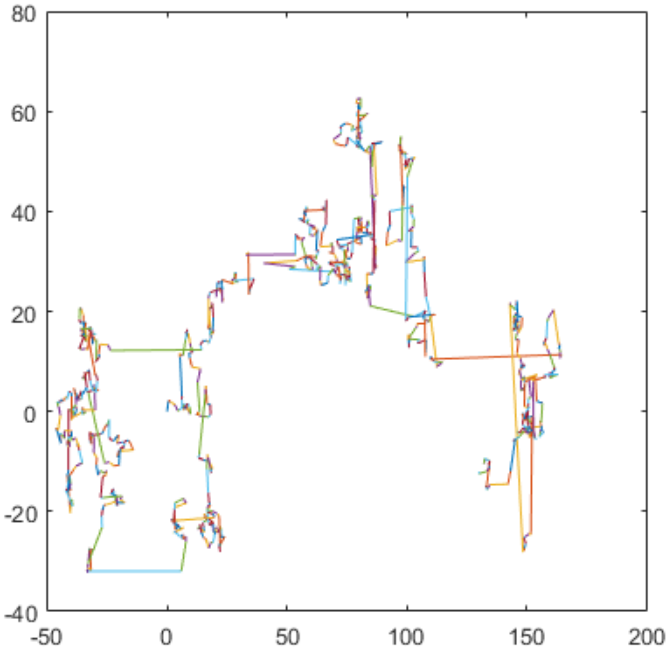

图 3-1 莱维飞行图像

3.1.3 算法主要步骤

综上所述，RLFPSO 的算法步骤描述如下：

输入：种群规模 *Size*，粒子维数 *Dimension*，最大迭代次数 *Tmax*；

输出：平均最优适应度 *Mean*，方差 *Std.*；

步骤 1：种群初始化；

步骤 2：计算粒子的初始适应度值；

步骤 3：根据适应度得到粒子的个体最优位置和群体最优位置；

步骤 4：检查每个粒子的个体最优值连续未更新次数，若未更新次数超过 *Limit*，则按照式（3-6）、式（2-2）更新粒子位置；否则按照式（2-1）、式（2-2）更新粒子位置；

步骤 5：若达到算法终止条件，结束算法；否则返回步骤 2。

3.1.4 算法伪代码

```
Begin
1   初始化参数
2   随机生成初始种群, 初始化 Pi , Pg, Pw
3   for i = 1 to Tmax
4       for j = 1 to Size
5           if f(xi) < f(Pi)
6               Pi = xi;
7               Limit(i) = 0    %记录连续未更新次数
8           else
9               Limit(i)++;
10          end if
11          if f(xi) > f(Pw)
12              Pw = xi;
13          end if
14          获取正态分布随机数 u, v;
15          S = u/|v|^(1/β);      %莱维飞行的步长
16          Stepsize = S*scale;   %根据具体问题调整步长
17          if Limit(i) < 10
18              根据标准 PSO 公式更新粒子速度和位置
19          else
20              采用反向学习策略更新粒子速度和位置
21          end if
22      end for
23  end for
```

3.1.5 算法流程图

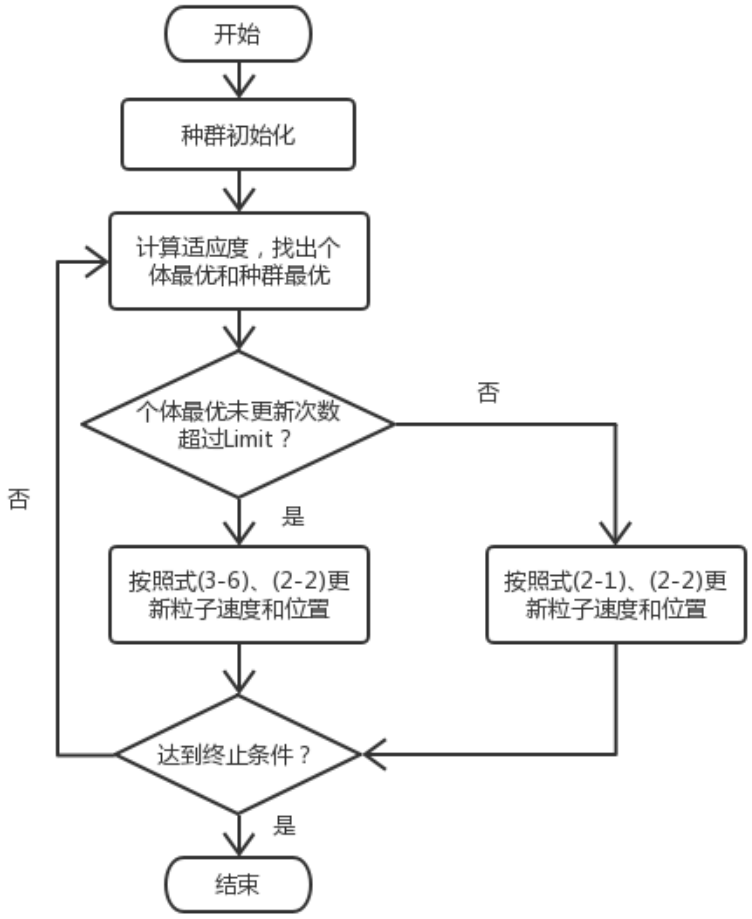

图 3-2 RLFPSO 算法流程图

3.2 时间复杂度分析

衡量算法优劣的一个重要标准就是算法的复杂度。设 RLFPSO 算法的种群规模为 N，粒子的维度为 D，算法在最坏情况下的时间复杂度分析如下：

步骤 1：初始化粒子参数和种群，时间复杂度为 $O(ND)$；

步骤 2：计算粒子适应度值，更新个体最优和群体最优，判断粒子个体最优未更新次数是否大于 $Limit$，并更新粒子位置，时间复杂度为 $O(ND)$；

步骤 3：判断算法是否达到终止条件，时间复杂度为 $O(1)$。

综上，RLFPSO 算法的时间复杂度为 $O(ND)$。

3.3 仿真实验结果分析

3.3.1 测试函数

为了评价 RLFPSO 算法的性能，本文选取了常用的 7 个标准测试函数进行对比实验。函数的详细信息如下表 3-1 所示。

表 3-1 测试函数

函数	名称	搜索范围	属性	理论最优值
F1	Sphere	[-100,100]	单峰	0
F2	Quadric	[-100,100]	单峰	0
F3	Ackley	[-32,32]	多峰	0
F4	Griewank	[-600,600]	多峰	0
F5	Rastrigin	[-5.12,5.12]	多峰	0
F6	Rosenbrock	[-2.5,2.5]	多峰	0
F7	Schaffer	[100,100]	多峰	0

3.3.2 对比算法

为了更直观地评价本文提出的算法的收敛速度和全局搜索能力，本文将选取 3 个不同的 PSO 算法与 RLFPSO 算法在上述测试函数中进行仿真测试比较，包括 SPSO[5]、CLPSO[8]、HPSO[7]。其中 SPSO 是在基本 PSO 算法的基础上，引入了惯性权重，属于在算法参数上做出改进；CLPSO 则是改进了 PSO 算法中粒子本身的学习策略，让粒子的每一维都随机地向种群中的随机粒子进行学习，从而实现种群多样性的提高；HPSO 则是在粒子群算法中引入了一种新的健康度机制，用来识别粒子是否陷入了局部最优，当检测到粒子的健康度过低时则会采取变异操作帮助粒子逃离局部最优位置。

在本文实验中，这些对比算法的参数设置以原文献为准，具体参数设置如下表 3-2 所示。

表 3-2 对比算法参数

算法	参数设置
SPSO	ω:[0.4.0.9], $c_1 = c_2 = 1.49445$
CLPSO	ω:[0.4.0.9], $c_1 = c_2 = 1.49445$
HPSO	$\omega = 0.6, c_1 = c_2 = 1.49, W_s = 3, W_{osc} = 0.8$

3.3.3 实验环境及参数设置

本文的实验的硬件环境为 Intel i5-3210M CPU 2.50GHz，RAM 6.00GB，WINDOWS 10 操作系统。软件环境为 MATLAB 2016b。MATLAB 有着高效的数值计算和符号计算功能，以及完备的图形处理功能，能够实现计算结果和编程的可视化，使得实验的比较结果更加清晰明了。

本文提出的 RLFPSO 算法参数设置如下：学习因子 c_1=c_2=1.49445，反向学习阈值 *Limit*=10，惯性权重 $\omega=0.55$；莱维飞行中的参数 $\beta=1.5$，α=0.01。

由于 RLFPSO 算法在速度更新公式中引入了一个新的参数 c_3，该值的大小也会对算法的效果产生一定的影响。为了确定 RLFPSO 算法中学习因子 c_3 的最佳值，设定粒子维数 *D*=30，种群规模 *N*=40，最大迭代次数 T_{max}=2000，对 c_3 取不同的值进行实验，结果如表 3-3 所示。表中 Mean 代表计算 20 次结果的平均值，Std.表示 20 次求解的标准差。

表 3-3 不同 c_3 算法结果

		F2	F4	F6
c_3=0.4	Mean	1.11E-261	0.00E+000	2.89E+001
	Std.	0.00E+000	0.00E+000	3.90E-002
c_3=0.6	Mean	3.18E-267	0.00E+000	2.89E+001
	Std.	0.00E+000	0.00E+000	3.42E-002
c_3=0.8	Mean	2.69E-269	0.00E+000	2.89E+001
	Std.	0.00E+000	0.00E+000	3.79E-002
c_3=1.0	Mean	4.24E-266	0.00E+000	2.89E+001
	Std.	0.00E+000	0.00E+000	3.78E-002
c_3=1.2	Mean	5.85E-268	0.00E+000	2.88E+001
	Std.	0.00E+000	0.00E+000	4.46E-002
c_3=1.4	Mean	**6.80E-271**	**0.00E+000**	**2.88E+001**
	Std.	**0.00E+000**	**0.00E+000**	4.06E-002
c_3=1.6	Mean	1.44E-268	0.00E+000	2.88E+001
	Std.	0.00E+000	0.00E+000	3.57E-002
c_3=1.8	Mean	2.04E-267	0.00E+000	2.88E+001
	Std.	0.00E+000	0.00E+000	5.06E-002

从表 3-3 中可以看出，当 c_3 的取值为 1.4 时，算法的寻优效果最好。特别是在如 F2 的单峰函数中，效果明显。故本文实验中反向学习因子 c_3 的取值为 1.4。

3.3.4 性能对比实验与分析

本文利用上文表 3-1 中的 7 种算法 F1~F7 进行对比实验。实验中，粒子维度 *Dimension* = 30，种群规模 *Size* = 40,最大迭代次数 *Tmax* = 2000。实验重复进行 30 次，实验统计结果包括算法优化的均值(Mean)以及标准差(Std.)，具体结果如下表 3-4 所示。其中加粗数值表示相应函数在对比算法优化后得到的最优结果。

表 3-4 对比算法在测试函数上的优化结果

		SPSO	CLPSO	HPSO	RLFPSO
F1	Mean	3.50E+001	4.19E-004	2.48E-209	**1.13E-276**
	Std.	1.95E+001	1.10E-004	0.00E+000	**0.00E+000**
F2	Mean	2.94E+003	6.33E+003	9.14E-210	**4.92E-267**
	Std.	1.50E+003	1.47E+003	0.00E+000	**0.00E+000**
F3	Mean	7.24E+000	6.19E-002	**8.88E-016**	**8.88E-016**
	Std.	1.11E+000	1.78E-002	**0.00E+000**	**0.00E+000**
F4	Mean	4.27E+000	2.81E-002	**0.00E+000**	**0.00E+000**
	Std.	1.57E+000	1.13E-002	**0.00E+000**	**0.00E+000**
F5	Mean	8.44E+001	1.03E+001	**0.00E+000**	**0.00E+000**
	Std.	2.04E+001	2.41E+000	**0.00E+000**	**0.00E+000**
F6	Mean	6.03E+001	5.57E+001	2.89E+001	**2.88E+001**
	Std.	1.90E+001	2.00E+001	4.86E-002	**4.74E-002**
F7	Mean	2.50E-003	2.50E-003	2.50E-003	**2.50E-003**
	Std.	2.03E-017	0.00E+000	8.69E-009	**3.26E-017**

从表 3-4 的对比数据中可以看出，在本文所使用的 7 个标准测试函数中，RLFPSO 算法相较于其他几个算法解的精度以及结果的稳定性都有了一定的提升。特别是在如 F1、F2 的单峰函数中，求解精度有了明显的提高。而在多峰函数如 F3、F6 中，RLFPSO 算法的提升较小，与 HPSO 几乎相差无几。在 F7 上 RLFPSO 算法的稳定性有了较大的提升。

为了更加直观的展示各个算法的寻优效果，使用 MATLAB 绘制出 4 种算法在 7 个测试函数上的收敛曲线图如下图 3-3 所示。

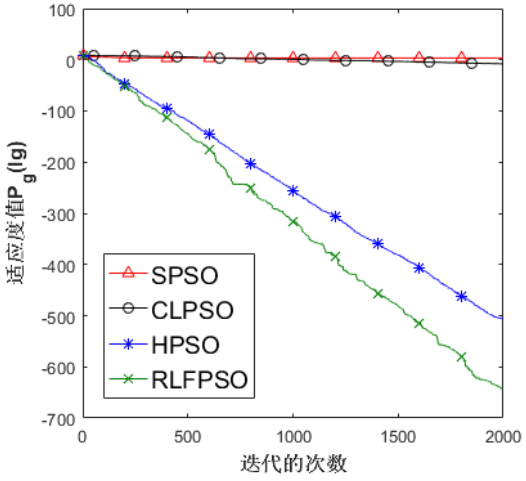

图 3-3(a) F1 收敛曲线

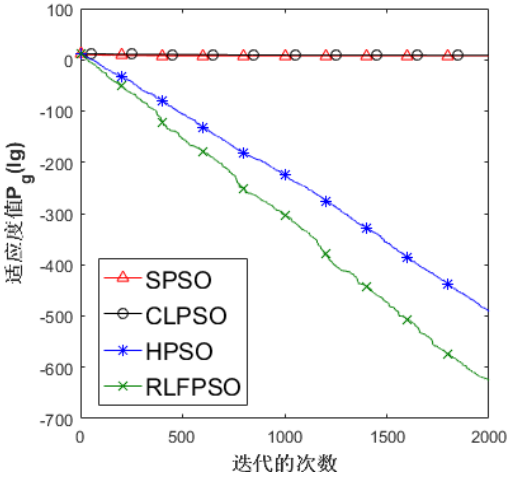

图 3-3(b) F2 收敛曲线

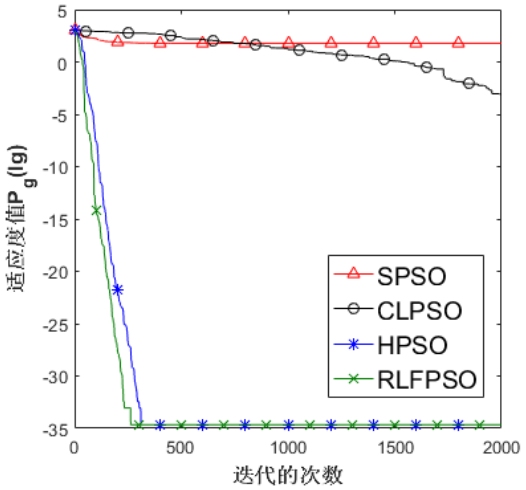

图 3-3(c) F3 收敛曲线

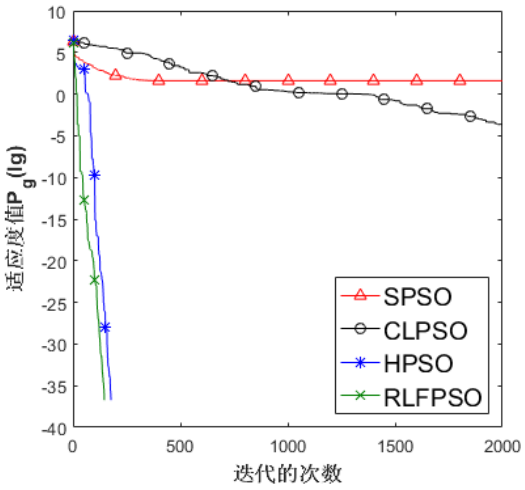

图 3-3(d) F4 收敛曲线

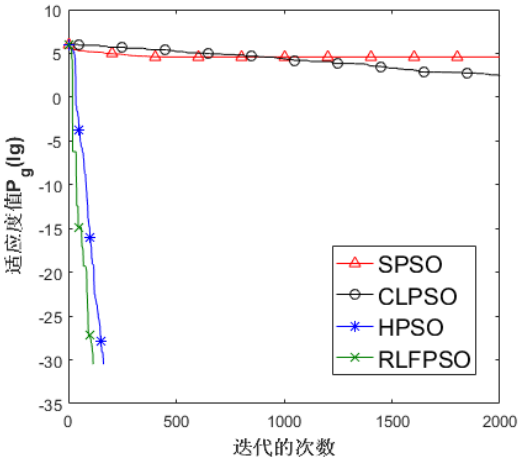

图 3-3(e) F5 收敛曲线

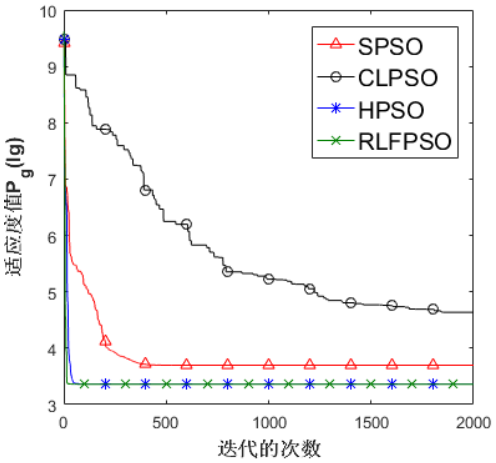

图 3-3(f) F6 收敛曲线

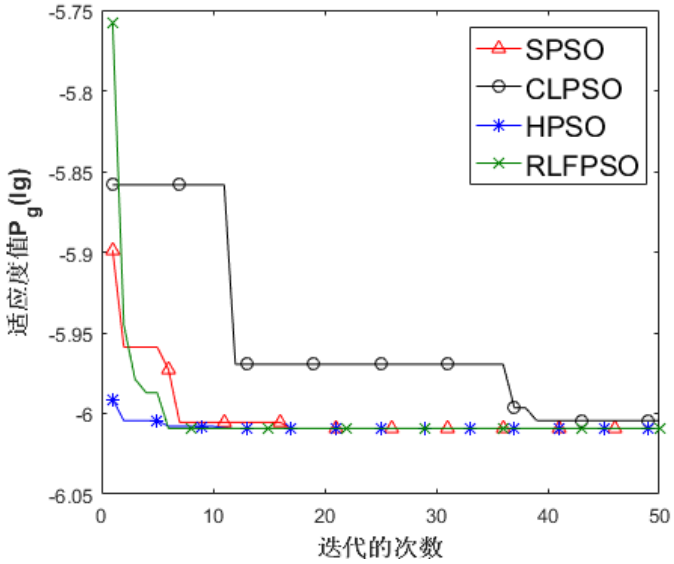

图 3-3(g) F7 收敛曲线

上图 3-3(g)中，对于 F7 四个算法均在前几次迭代中就收敛到相同的适应度值上，并在后续的迭代中全局最优值一直保持不变，可以看出这四个算法在 F7 上体现出相同的求解性能。

从图 3-3 中可以看出，RLFPSO 算法不论是在单峰函数还是多峰函数上，除了 F7 以外，相较于其他几个算法都有着更快的收敛速度，可以在更短的迭代次数内找到全局最优值。因此，通过对表 3-4 以及图 3-3 的结果进行分析可知，RLFPSO 在求解精度以及收敛速度上优于其他 3 个算法。

3.4 本章小结

本章详细介绍了基于莱维飞行的反向学习粒子群算法的具体改进方法以及算法的性能。首先介绍了 RLFPSO 算法的步骤、分析了算法的时间复杂度，然后在 7 个不同的标准测试函数上进行了仿真对比实验，证明了 RLFPSO 算法的有效性。

第四章 RLFPSO 算法在车辆路径问题中的应用

4.1 车辆路径问题介绍

4.1.1 车辆路径问题的概念与组成要素

（1）车辆路径问题的概念

车辆路径问题(Vehicle Routing Problem, VRP)自从被提出来以后，就一直成为了物流领域研究的热门方向。车辆路径问题是指：在给定需求地点、产品需求量和车辆载重等基本信息，并且满足一定的约束条件下，通过规划安排出合适的车辆配送方案和路线，以达到既定的一个或多个目标（例如配送总成本最低、车辆行驶的路程最短等）。

（2）车辆路径问题的组成要素

根据车辆路径问题的概念可以总结出车辆路径问题基本组成要素，包括客户、配送中心、货物、车辆、配送网络、约束条件和目标函数[52]。

a. 配送中心

配送中心是进行产品装配并安排车辆进行货物配送的物流节点。在不同的问题中，配送中心的数量可以根据实际情况变化。

b. 车辆

车辆是配送中心对货物进行配送的主要载体，是 VRP 的必要组成部分。在车辆路径问题的研究中，车辆的不同性能也会对问题的条件造成影响。例如车辆的数量、载重、最大行驶距离和速度等等属性都会被考虑在内，不同的场景中的选择也不同。

c. 货物

物流配送的过程就是将特定的货物准确无误地配送到需要服务的地点。配送的产品特征根据实际情况也不同，货物的体积、重量、以及自身的某些特性都会影响整个物流配送流程。

d. 客户

客户则是整个物流配送过程中服务的对象。客户在车辆问题中的表现形式也不唯一，可以是个体也可以是商户等。在研究 VRP 的过程中，需要考虑客户对货物本身的要求、送达时间的要求以及需求量等因素，尽量在客户的要求得到满足前提下将货物送到指定地点，从而提高客户的满意度。

e. 运输网络

车辆路径问题中的运输网络就是由车辆行驶的道路、客户、配送中心等元素组成。配送中心和客户就是网络中的各个节点，节点与节点之间通过道路相连接。但在 VRP 的实际研究中，配送路径不仅仅可以表示距离，也可以用来表示成本、时间等，根据所研究问题的情况而定。

f. 约束条件

约束条件是对于在整个物流配送过程中由于一些客观或者主观的因素所要考虑的一些限制条件的抽象化。例如配送中心的车辆数、车辆的载重、客户要求的货物送达时间窗等等都是需要考虑的问题。在研究的过程中针对不同的实际情况来对问题设定相应的约束，求解的结果才会更加合理，具有更高的应用和参考价值。

g. 目标函数

目标函数就是整个车辆路径问题所要达到的最终目的的抽象。目标函数根据实际问题，可能只有一个也可以有多个。现阶段研究中常见的目标函数有最短行驶距离、最短行驶时间、最少车辆数、最小总配送成本、最大客户满意度等等。

一个基本的车辆路径问题示意图如下图 4-1 所示。图中圆圈表示需要配送的不同客户点，箭头表示车辆运输的路径。

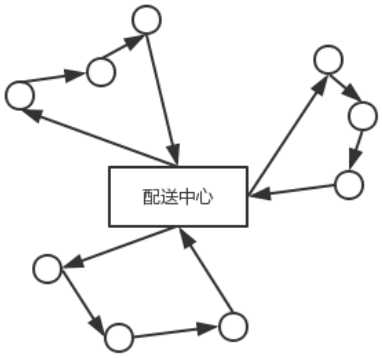

图 4-1 VRP 示意图

4.1.2 车辆路径问题的分类

车辆路径问题从提出到现在，一直是物流配送领域的热门研究方向，因此所涉及的领域也在不断扩大。对于不同的物流配送过程抽象出的 VRP，其组成要素也是不尽相同，所以根据不同的组成要素的特性，可以将 VRP 进行分类。下表 4-1 中列出了几种常见的 VRP 分类。

表 4-1 VRP 的分类

序号	分类依据	VRP 类型
1	相关信息	静态确定性 VRP
		动态随机性 VRP
2	规划周期	周期性 VRP
		非周期性 VRP
3	服务时间	无时间限制 VRP
		软时间窗 VRP
		硬时间窗 VRP
		混合时间窗 VRP
4	任务特征	只提货或送货 VRP
		提送货一体 VRP
5	车场	单车场 VRP
		多车场 VRP
6	车辆数量	单车型 VRP
		多车型 VRP
7	客户需求	确定需求 VRP
		随机需求 VRP
		未知需求 VRP

4.1.3 带时间窗的车辆路径问题

在车辆路径问题中，客户对货物规定的最早到达时间 ET_i 以及最晚到达时间 LT_i 所构成的区间$[ET_i, LT_i]$就是时间窗。带时间窗的车辆路径问题就是指在研究车辆路径问题的过程中将时间窗约束加入到考虑的范围内，在安排车辆配送的路线时，需要尽量使货物在客户要求的时间内送到指定地点。如果车辆早到或者晚到，则会产生惩罚。根据惩罚成本的高低以及客户对时间窗的要求，一般可以将时间窗分为以下两类：硬时间窗和软时间窗。

（1）硬时间窗

硬时间窗是指在物流配送过程中，客户对于货物到达的时间有非常精确的要求并且不可调整。如果运送的货物没有按客户的要求在期望的时间窗内到达，则客户会拒绝接受货物。硬时间窗的惩罚函数为：

$$P(t)=\begin{cases}p_0, & t<ET_i, t>LT_i\\ 0, & ET_i\le t\le LT_i\end{cases} \tag{4-1}$$

式（4-1）中，$P(t)$表示惩罚函数值，p_0表示早到或晚到的惩罚系数，由于是硬时间窗，p_0一般为无穷大，表示不允许发生。ET_i表示客户期望的运输产品到达的最早时间，LT_i表示客户期望的运输产品到达的最晚时间，t代表运输产品实际运输到客户点的时间。

硬时间窗的罚函数如图 4-2 所示。

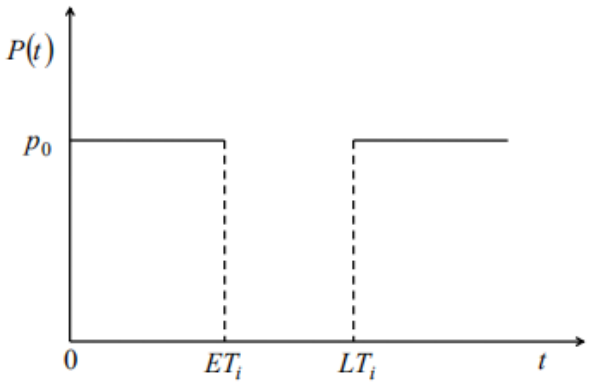

图 4-2 硬时间窗惩罚函数图像

（2）软时间窗

软时间窗是指在货物配送的过程中，客户对货物送达的时间有一个要求，但是相对于硬时间窗来说要求比较宽松。如果产品没有在客户期望的时间内到达，只要没有超出时间窗太久，客户都能够接受，但是配送方需要接收一定的惩罚作为客户损失的补偿，超出时间窗越多惩罚越大。软时间窗的惩罚函数为：

$$P(t)=\begin{cases}p_1\cdot(ET_i-t) & t<ET_i\\ 0, & ET_i\le t\le LT_i\\ p_2\cdot(t-LT_i), & t>LT_i\end{cases} \tag{4-2}$$

式（4-2）中，p_1表示时间窗中早到的惩罚系数，p_2表示晚到的惩罚系数。P1 和 p2 可以相同也可以不同也可以为 0，例如有些企业的需求是需要冷藏的货物，如果货物提前到达会产生额外的仓储费用，在有些行业中（例如疫苗运输行业），货物的存储需要有严格的温度控制，存储的成本很高，此时早到惩罚系数就会大于晚到的惩罚系数。

软时间窗的惩罚函数图像如图 4-3 所示。

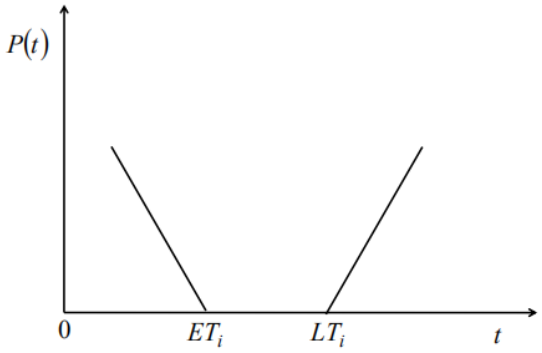

图 4-3 软时间窗惩罚函数图像

4.1.4 VRPTW 数学建模

假设有一个物流配送中心的车辆数为 m，每辆车的容量为 Q，此时有 n 个客户点需要配送，且每个客户的需求量为 q_i (i=1, 2,…, n)，客户 i 到客户 j 之间的距离为 d_{ij}。对于该次配送任务拟定的基本要求如下：1)车辆从配送中心出发，执行完配送任务后需返回配送中心；2)所有的客户点的需求量必须被满足；3)车辆不可超载；4)配送方案行驶路线所花费的总成本要尽可能的小。在我们的日常生活中，此类配送问题的例子比比皆是，而有些问题由于实际因素的影响会更加复杂。从数学的角度讲，VRPTW(Vehicle Routing Problem with Time Window)是非线性规划问题，属于 NP 难问题，智能算法就是解决该类问题的一种重要途径。

将实际的 VRPTW 问题抽象为数学模型，根据配送任务的要求可以作出该问题中的假设条件如下：

（1）各个客户点的位置和需求量已知且确定；

（2）配送中心车辆数已知，车辆有载重上限；

（3）各个客户点之间，客户点与配送中心之间的距离已知；

（4）运输过程中的产品损耗忽略不计；

（5）货物若未在客户期望的时间窗内到达，会产生额外的成本，且成本随超出时间的增多而增大，早到和晚到惩罚系数相同。

根据该次任务的配送要求，总结出该次配送需要满足的约束条件如下：

（1）每个客户点的需求有且仅由一辆车进行配送；

（2）每个客户点的需求量必须得到满足；

（3）车辆完成配送任务后需返回配送中心；

（4）车辆装载的货物重量不能大于其载重量。

根据以上分析，该次车辆路径优化问题的目标即为整个配送活动中所花费成本

的最小值。故目标函数即为整个配送活动所花费的成本，其表达式如下：

$$\begin{aligned}\min C &= C_v + C_t + C_p \\ &= C_1 \cdot m + \sum_{k=1}^{m}\sum_{i=0}^{n}\sum_{j=0}^{n} C_2 \cdot d_{ij} \cdot X_{ij}^{k} + C_p \end{aligned} \tag{4-3}$$

$$s.t. \quad \sum_{j=1}^{n}\sum_{i=0}^{n} q_j \cdot X_{ij}^{k} \le Q \tag{4-4}$$

$$\sum_{k=1}^{m}\sum_{i=0}^{n} X_{ij}^{k} = 1 \tag{4-5}$$

$$X_{ij}^{k} \in \{0,1\} \tag{4-6}$$

式（4-3）表示问题的目标函数，表示该次配送的总成本，由车辆固定成本，运输成本，惩罚成本三部分构成。其中惩罚成本采用软时间窗，C_p 的表达式如下：

$$C_p(t_i) = \begin{cases} C_3 \cdot (ET_i - t_i), & t < ET_i \\ 0, & ET_i \le t \le LT_i \\ C_3 \cdot (t_i - LT_i), & t > LT_i \end{cases} \tag{4-7}$$

式（4-4）表示车辆配送总需求量不得超过载重量；

式（4-5）表示每个客户点有且仅有一辆车进行配送；

在上述数学模型中，m 表示配送中心拥有的总车辆数，n 表示需要配送的客户数，q_j 表示第 j 个客户的需求量，d_{ij} 表示第 i 个客户到第 j 个客户的最短干道距离，X_{ij}^{k} 是模型的决策变量，表示第 k 辆车是否从点 i 驶向点 j，若是则为 1，不是则为 0，Q 表示车辆的最大装载件数，C_1 表示车辆的固定成本，C_2 表示单位运输成本，C_3 表示超出时间窗的惩罚系数，$[ET_i, LT_i]$表示第 i 个客户的期望时间窗，t_i 表示车辆到达客户点 i 的时间。

4.2 RLFPSO 在 VRPTW 中的实现

4.2.1 编码方法

由于本文所研究的车辆路径问题是离散的模型，而粒子群算法的本质是连续的，不适合求解离散的问题。因此需要对算法中的粒子进行编码，来建立连续空间中的粒子和离散空间中的行驶方案之间的关系。本文将采用排序法来实现粒子与行驶方案的映射。具体的编码方案如下：

将设置粒子的位置编码分为两部分，第一部分用来映射车辆是否被使用的情况，将其称为 p_1。设配送中心车辆数总和为 M，所有客户总数为 N，则每个粒子的

p_1 使用 $M \times N$ 个二维实向量表示，二维向量的第一维表示车辆的编号，第 1~N 个实数用 1 表示车辆 1，第 $N+1 \sim 2N$ 个实数用 2 表示车辆 2，以此类推；二维向量的第二维表示粒子在空间中的位置矢量，取[0,1]的随机实数。在随机生成了粒子的位置矢量后，便根据位置矢量中的实数的值对 $M \times N$ 个二维实向量进行排序，然后取前 N 个二维向量来表示所有车辆的使用情况。

例如，若某配送中心共有三辆车，某次配送任务中的客户点数为 4 个。下表 4-2 中表示根据上述编码方案生成的 p_1，表 4-3 表示根据位置矢量排序后的 p_1。

表 4-2 初始化的 p_1

车辆编号	1	1	1	1	2	2	2	2	3	3	3	3
位置矢量	0.46	0.23	0.85	0.96	0.56	0.34	0.76	0.99	0.13	0.44	0.42	0.86

表 4-3 排序后的 p_1

车辆编号	3	1	2	3	3	1	2	2	1	3	1	2
位置矢量	0.13	0.23	0.34	0.42	0.44	0.46	0.56	0.76	0.85	0.86	0.96	0.99

得到了表 4-3 的序列后，截取前 4 个向量的车辆编号所构成的集合便是需要使用到的车辆的编号，在上面的例子中前 4 个向量的车辆编号为 3、1、2、3，故三辆车均被使用，且车辆 1 和车辆 2 被分配 1 个客户，车辆 3 被分配 2 个客户。

在得到了需要使用的车辆后，再使用粒子编码的第二部分将各个客户点随机分配给某一辆车，将其称为 p_2。p_2 使用 N 个二维实向量表示，第一维是从 1 到 N 的整数，用来表示各个客户点，第二位表示粒子的位置矢量，同样取[0,1]的随机实数，在初始化后根据位置矢量将 N 个二维实向量排序。仍然使用上面的例子得到的初始化 p_2 以及排序后的 p_2 如表 4-4 和表 4-5 所示。

表 4-4 初始化的 p_2

客户编号	1	2	3	4
位置矢量	0.64	0.10	0.43	0.28

表 4-5 排序后的 p_2

客户编号	2	4	3	1
位置矢量	0.10	0.28	0.43	0.64

得到表 4-5 中的客户编号序列后，将其与第一部分的车辆的任务数对应起来，便得到了每一个车辆的服务序列，上例中由于车辆 1、车辆 2 都被分配 1 个客户，车辆 3 被分配 2 个客户，故车辆 1 服务顺序是客户 2，车辆 2 服务顺序是客户 4，

车辆 3 服务顺序是客户 3、1。

通过这种编码方式得到的车辆选用情况和车辆服务序列可以保证每个客户点都被服务，且只被同一辆车服务。

4.2.2 约束条件的处理

上文中的带时间窗的车辆路径问题模型中的约束条件主要有三个：一是车辆载重约束，即车辆所载的货物总量不得大于车辆的最大载重量，该约束主要通过将超重方案的适应度值（即成本）设置为无穷大来满足，由于在本文的优化问题中，粒子群算法求解的是最小值，故适应度值为无穷大的解会被排除；二是每个客户点有且仅有一辆车进行配送，该约束通过上述例子编码方案满足；三是时间窗约束，该约束则是采用罚函数法，直接将时间窗约束写入目标函数中。

4.3 案例分析一

4.3.1 算例描述

某医药企业在药物供应链中，会将仓库中的药品配送到所在城市的各个零售点中。本文将运用所提出的带时间窗的车辆路径问题模型，对企业的配送方案进行设计。

本案例采用了该公司 2019 年 9 月 2 日在武汉市内进行配送过程中客户点的需求信息。其中，供应点经度为 114.143962，纬度为 30.626479，需求点为武汉市内的各大药品零售点。

本算例考虑的主要为单一固定的药品仓库，在时间以及车辆载重量的约束条件下，规划出一个最佳的配送方案，有效地将药品配送至各个零售点，并使得配送的总成本最小。各个零售点的位置和需求量信息如下表 4-6 所示。将上述各个客户的位置在地图上进行标注，如下图 4-4 所示。

表 4-6 中客户的经纬度信息，通过百度地图可以获取到各个客户点之间的最短干道距离。将两个客户点的地理位置提交到百度提供的 GIS 服务器，经过百度服务器的远程计算可以获取到两个点之间的最短干道距离。

为了简化收集数据的过程，编写程序，并调用百度地图 API 接口便可以方便进行距离数据的收集。为了方便统计各个物流节点间的距离，将供应点编号为 0。具体数据（部分）如下表 4-7 所示。

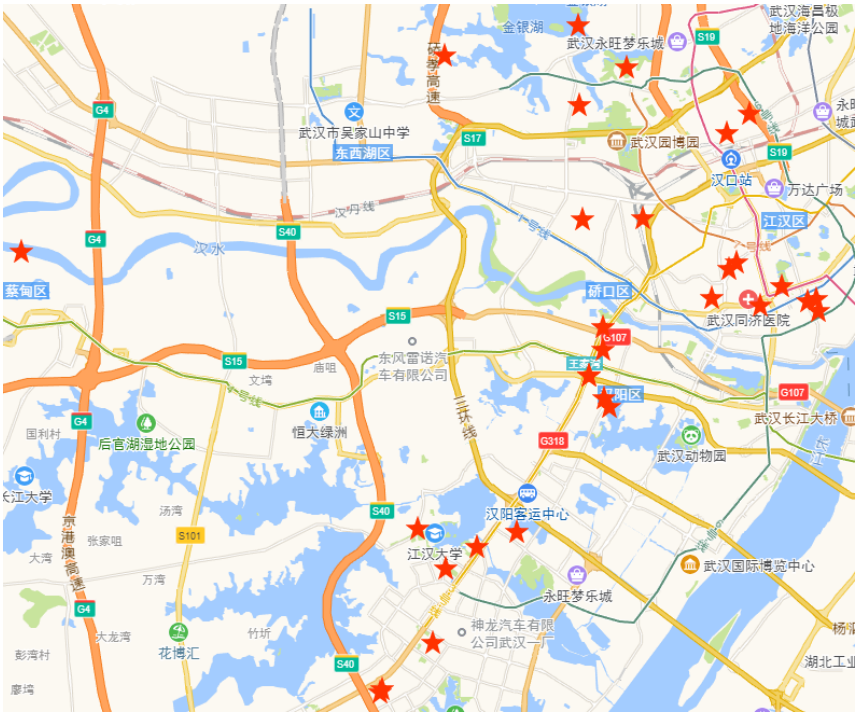

图 4-4 客户位置

表 4-6 客户信息

编号	经度	纬度	需求量	编号	经度	纬度	需求量
1	114.206467	30.632956	4	16	114.142786	30.471799	2
2	114.254244	30.588156	1	17	114.205988	30.655027	4
3	114.163363	30.646647	2	18	114.270996	30.583431	9
4	114.143571	30.472872	4	19	114.159868	30.485396	3
5	114.214067	30.56583	2	20	114.282773	30.576577	2
6	114.226593	30.602053	2	21	114.155075	30.516925	4
7	114.279342	30.579153	3	22	114.253555	30.625453	2
8	114.221572	30.643363	2	23	114.214367	30.552626	2
9	114.260689	30.630642	3	24	114.282009	30.579733	5
10	114.028471	30.592873	7	25	114.264087	30.578004	3
11	114.248794	30.580126	3	26	114.186661	30.515866	3
12	114.173949	30.511712	4	27	114.21598	30.550202	3
13	114.256444	30.589842	4	28	114.209472	30.559011	2
14	114.207454	30.601791	2	29	114.163959	30.506084	3
15	114.209472	30.559011	4	30	114.21407	30.572253	2

表 4-7 各点之间的距离（部分，单位 km）

序号	0	1	2	3	4	5	6	7	…
0	0								
1	9.646	0							
2	7.785	16.006	0						
3	25.278	19.546	25.184	0					
4	10.576	6.16	17.052	13.733	0				
5	5.446	4.488	11.168	18.386	5.882	0			
6	13.019	3.965	19.435	20.97	8.362	7.447	0		
7	3.423	9.825	7.453	25.76	13.256	7.922	10.885	0	
8	6.185	7.629	12.613	23.315	10.811	6.985	7.138	5.222	
9	24.651	25.446	20.992	25.151	19.983	24.103	29.012	26.861	
10	12.178	3.236	16.789	18.377	5.873	4.63	3.903	11.203	
11	19.781	14.049	19.687	5.999	8.186	13.005	16.918	21.991	
12	10.007	0.438	15.212	18.682	6.178	3.822	3.824	9.032	
13	5.364	6.213	10.979	19.103	6.599	3.449	8.655	7.471	
14	11.003	7.212	18.104	13.032	1.336	6.168	9.604	14.108	
15	25.419	19.687	25.325	0.732	13.824	18.643	22.556	27.629	
16	4.023	12.017	7.103	27.748	12.944	8.522	13.077	2.806	
17	11.655	2.601	18.071	20.606	7.326	6.083	3.076	10.68	
18	23.093	17.361	22.999	2.958	11.498	16.317	20.23	25.303	
19	13.614	4.56	20.03	21.046	8.438	8.042	0.925	12.639	
20	21.375	15.643	21.281	6.768	9.78	14.599	18.512	23.585	
…									
30	10.15	5.706	16.598	14.048	1.542	4.662	8.098	12.602	

4.3.2 参数的设置

（1）模型参数的设置

模型中参数的设置如下表 4-8 所示。客户需求的时间窗为下单后的一天之内，会根据不同订单的下单时间而变化，故没有列举在表格中。

表 4-8 模型参数设置

符号	数值	单位	含义
m	6	辆	配送中心最大车辆数
Q	40	件	车辆的最大装载件数
v	30	公里/时	车辆的行驶速度
C_1	200	元/辆	车辆使用的固定成本
C_2	5	元/公里	运输过程中每公里产生的运费
C_3	1	元/分钟	超出时间窗的惩罚成本

（2）算法参数的设置

本案例实验中，所有算法的参数设置均与第三章中的性能仿真实验相同。

（3）实验环境

本案例中的实验环境，与第三章性能仿真实验相同。

4.3.3 实验结果与分析

本文主要从运输过程中的总成本作为目标函数进行考虑，将不同 PSO 算法的优化结果与改进的算法结果进行对比，并通过结果的对比，比较算法的优劣。

（1）算法比较

根据车辆信息、各个客户的需求量、时间窗要求等信息，采用 MATLAB 进行编程，使用不同的粒子群算法对算例进行求解，根据目标函数计算得到的配送总成本最小的方案将被选定为最优方案。在具体实验的过程中，每种算法被独立运行 30 次，并对每种算法 30 次运行的结果进行分析，结果如下表 4-9 所示。

表 4-9 不同 PSO 算法求解结果（元）

算法名称	Mean	Std
SPSO	3992.1	294.3
CLPSO	4201.5	224.1
HPSO	2907.4	309.2
RLFPSO	**2367.9**	**247.2**

上表 4-9 描述了 SPSO、CLPSO、HPSO、RLFPSO 四个算法在上述案例下分别独立求解最小成本 30 次的结果，Mean 和 Std 分别表示求解结果的平均值与标准差，从表中可以看出，RLFPSO 所找出的路径成本平均值相较于其他几种算法更低，但是方差也较小。可以看出，本文提出的 RLFPSO 算法相较于其他算法更

适合求解成本最小化的车辆路径问题。

下表 4-10 给出了 RLFPSO 算法在 30 次求解中，结果最优的一次各个车辆的具体配送路线和车辆承载率以及准时率。

表 4-10 RLFPSO 求解最优结果

车辆编号	服务路线	装载数(件)	装载率	准时率	总成本（元）
1	0-2-10-11-14-16-19-22-26-27-0	26	65%	100%	
2	0-13-4-29-15-3-18-30-25-28-20-9-0	38	95%	83.3%	1,822.0
3	0-17-6-23-24-21-8-12-1-5-7-0	32	80%	100%	

下图 4-5 是使用 RLFPSO 计算 30 次中结果最优的一次的配送方案路线图。

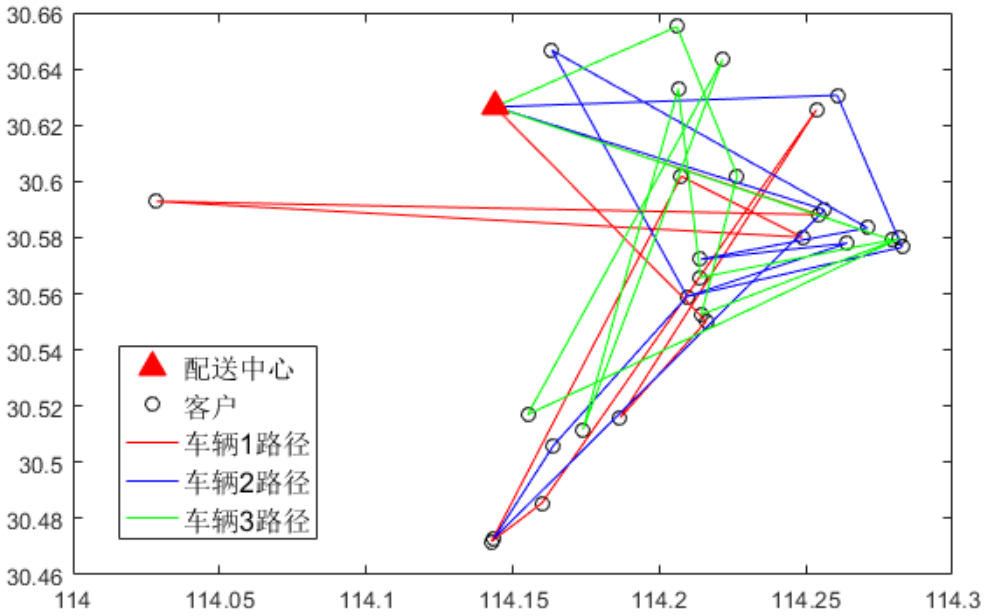

图 4-5 RLFPSO 算法求得的最优路径图

（2）原始方案比较

为了证明由 RLFPSO 算法计算出的结果具有一定的实际意义，先将该公司对该次配送任务的实际执行过程的具体配送方案与 RLFPSO 算法计算出的方案进行对比，所执行的原始配送方案如下表 4-11 所示。

表 4-11 原始配送方案

车辆编号	服务路线	装载数(件)	装载率	准时率	总成本（元）
1	0-4-15-18-28-20-25-0	24	77.5%	66.67%	2,493.9
2	0-29-5-14-27-26-22-30-10-0	24	85%	100%	
3	0-3-1-8-16-9-21-0	17	42.5%	100%	
4	0-2-23-7-6-11-24-13-19-17-12-0	31	97.5%	92.31%	

比较表 4-10 与表 4-11 可以看出，RLFPSO 算法求解 30 次结果的平均成本低于实际配送使用的成本，故 PSORLF 算法在该案例上有着较好的优化性能，可以有效地降低配送活动的成本。

4.4 案例分析二

4.4.1 算例描述

同样为案例一中的医药企业，本案例采用了该公司 2019 年 9 月 3 日在武汉市内进行配送过程中客户点的需求信息。其中，供应点经度为 114.143962，纬度为 30.626479，需求点为武汉市内的各药品零售点。与案例一不同的是本案例中共有 60 个客户点，问题的规模进一步增大。

各个零售点的位置和需求量信息如下表 4-12 所示。将上述各个客户的位置在地图上进行标注，如下图 4-6 所示。

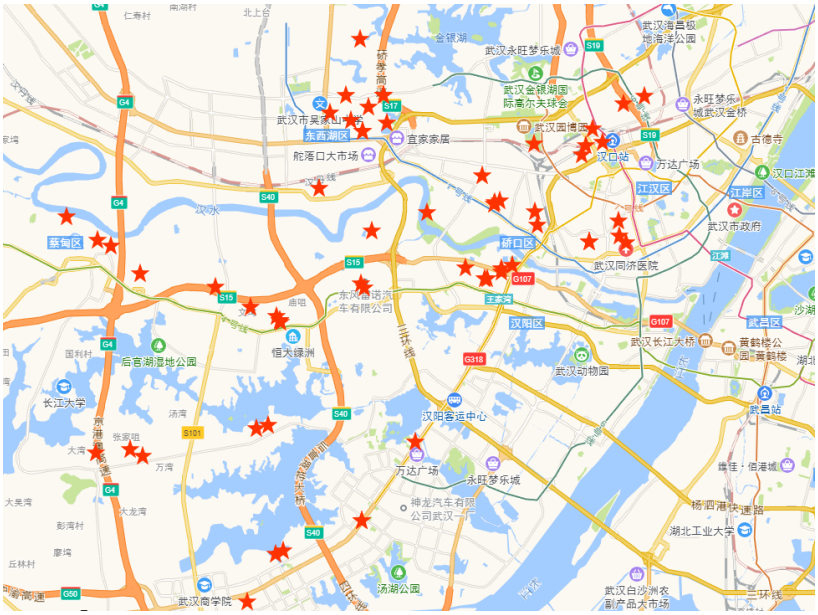

图 4-6 客户的位置

表 4-12 客户信息

编号	经度	纬度	需求量	编号	经度	纬度	需求量
1	114.106093	30.559363	1	31	114.224027	30.588458	2
2	114.16066	30.634255	2	32	114.154558	30.629958	2
3	114.203238	30.569461	2	33	114.201393	30.60587	3
4	114.152166	30.621505	4	34	114.043364	30.583167	2
5	114.245345	30.582431	2	35	114.116533	30.472168	2
6	114.245345	30.582431	2	36	114.209136	30.573152	9
7	114.243967	30.616619	2	37	114.268047	30.633817	4
8	114.178534	30.592909	2	38	114.116701	30.556062	2
9	114.138768	30.627935	4	39	114.222433	30.617062	2
10	114.151732	30.484041	2	40	114.091773	30.566495	2
11	114.104929	30.455343	4	41	114.056798	30.509111	4
12	114.173949	30.511712	4	42	114.06184	30.506765	3
13	114.04876	30.580928	2	43	114.2061	30.595697	5
14	114.151732	30.484041	2	44	114.222908	30.593125	8
15	114.162303	30.624476	3	45	114.260313	30.58217	3
16	114.042565	30.508021	2	46	114.152737	30.56632	4
17	114.155964	30.586478	21	47	114.213794	30.574227	7
18	114.206292	30.596459	3	48	114.118079	30.554758	1
19	114.20865	30.59696	2	49	114.259311	30.631192	6
20	114.257517	30.584726	5	50	114.16066	30.634255	3
21	114.113411	30.517811	2	51	114.119546	30.473312	1
22	114.242208	30.61334	3	52	114.152166	30.621505	2
23	114.246549	30.622317	3	53	114.118386	30.554274	1
24	114.151277	30.654006	1	54	114.145348	30.63429	6
25	114.061	30.571277	4	55	114.147363	30.62543	3
26	114.250708	30.617694	4	56	114.118072	30.554752	2
27	114.257308	30.5899	5	57	114.194553	30.573404	3
28	114.209003	30.571893	2	58	114.151472	30.567834	1
29	114.108679	30.516645	3	59	114.030471	30.591267	2
30	114.13419	30.601558	2	60	114.202441	30.569577	4

通过百度地图可以获取到各个客户点之间的最短干道距离。为了方便统计各个物流节点间的距离，将供应点编号为 0。具体数据（部分）如下表 4-13 所示。

表 4-13 各点之间的距离（部分，单位 km）

序号	0	1	2	3	4	5	6	7	…
0	0								
1	12.927	0							
2	3.136	15.159	0						
3	12.377	11.813	13.666	0					
4	1.485	12.822	2.467	12.675	0				
5	18.733	20.977	16.759	8.007	17.884	0			
6	18.733	20.977	16.759	8.007	17.884	0.001	0		
7	8.639	21.487	10.939	8.664	10.443	4.825	4.825	0	
8	1.230	12.184	9.776	6.762	7.790	9.910	9.910	12.000	
9	20.621	12.121	3.740	14.049	2.094	14.57	14.570	11.832	
10	27.635	17.408	21.758	13.178	19.772	15.859	15.859	18.774	
11	17.234	14.439	28.772	20.192	26.786	22.873	22.873	25.788	
12	14.526	16.313	18.371	9.791	16.385	12.472	12.472	15.387	
13	20.621	6.189	16.996	18.799	14.826	21.947	21.947	23.399	
14	3.180	17.408	21.758	13.178	19.772	15.859	15.859	18.774	
15	21.749	15.402	1.366	11.370	3.643	11.891	11.891	9.153	
16	8.631	10.734	24.366	21.575	22.196	31.801	31.801	30.769	
17	8.417	8.938	9.768	7.959	7.782	11.107	11.107	13.197	
18	8.081	15.862	9.855	7.240	7.568	4.849	4.849	6.216	
19	15.423	15.924	9.519	7.302	7.232	4.513	4.513	5.880	
20	23.515	19.965	15.070	7.455	14.574	3.072	3.072	6.613	
…									
60	12.460	11.896	13.749	0.068	11.611	6.145	6.145	11.112	

4.4.2 参数的设置

本案例中的参数设置见 4.3.2 中的参数设置。

4.4.3 实验结果与分析

（1）算法比较

同样，在实验的过程中，每种算法被独立运行 30 次，并对每种算法 30 次运行的结果进行分析，结果如下表 4-14 所示。

表 4-14 不同 PSO 算法求解结果（元）

算法名称	Mean	Std
SPSO	4.63E+003	2.50E+002
CLPSO	5.85E+003	1.65E+002
HPSO	4.95E+003	5.17E+002
RLFPSO	**3.89E+003**	**1.65E+002**

上表 4-14 描述了 SPSO、CLPSO、HPSO、RLFPSO 四个算法在上述案例下分别独立求解最小成本 30 次的结果，从表中可以看出，RLFPSO 所找出的路径成本平均值相较于其他几种算法更低，且方差也较小。可以看出，本文提出的 RLFPSO 算法相较于其他算法在求解本案例时更具有优势。

下表 4-15 给出了 RLFPSO 算法在 30 次求解中，结果最优的一次各个车辆的具体配送路线和车辆承载率以及准时率。

表 4-12 RLFPSO 求解最优结果

车辆编号	服务路线	装载数(件)	装载率	准时率	总成本（元）
1	0-39-7-36-40-59-15-2-24-55-0	26	65%	88.89%	3,500
2	0-4-26-23-10-21-25-48-42-51-35-47-53-58-33-0	38	95%	100%	
3	0-50-22-6-27-45-5-18-37-49-9-0	35	87.5%	100%	
4	0-34-13-41-11-14-46-17-0	39	97.5%	100%	
5	0-8-57-56-38-1-12-44-0	22	55%	71.43%	
6	0-52-32-19-43-20-31-28-60-3-29-16-30-54-0	39	97.5%	100%	

下图 4-7 是使用 RLFPSO 计算 30 次中结果最优的一次的配送方案路线图。

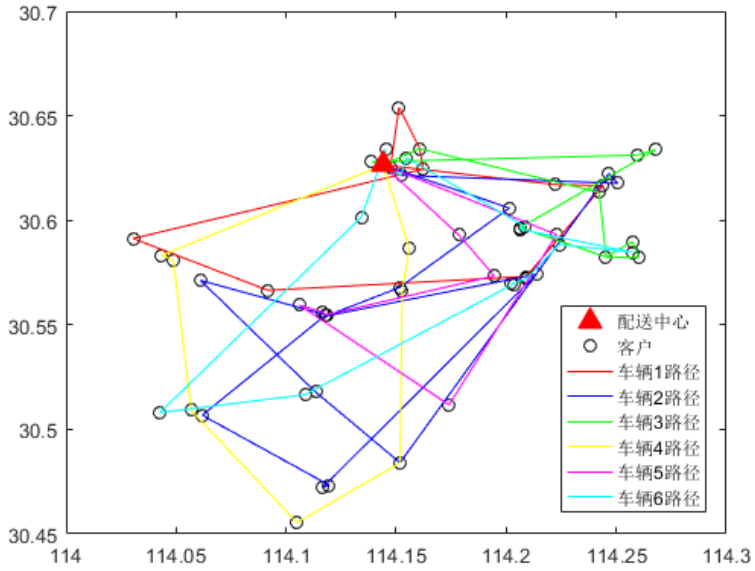

图 4-7 RLFPSO 算法求得的最优路径图

（2）原始方案比较

在该公司对该次配送任务的执行中，所执行的原始配送方案如下表 4-16 所示。

表 4-13 原始配送方案

车辆编号	服务路线	装载数(件)	装载率	准时率	总成本（元）
1	0-21-40-29-42-16-11-41-14-10-12-51-35-0	31	77.5%	66.67%	3,600
2	0-8-58-46-57-3-60-28-47-36-0	34	85%	100%	
3	0-38-48-53-56-1-25-13-34-59-0	17	42.5%	100%	
4	0-31-27-22-33-20-45-43-39-19-7-5-6-18-0	39	97.5%	92.31%	
5	0-26-17-49-37-23-0	38	95%	100%	
6	0-30-24-9-54-32-4-55-52-2-44-50-15-0	40	100%	100%	

比较表 4-15 与表 4-16 可以看出，虽然 RLFPSO 在 30 次求解中最好的一次结果好于原始方案，但这也同样是 30 次求解中唯一一次好于原始方案的结果，所以可以看出，RLFPSO 算法在求解该问题时能够得到较好的结果，但算法的结果的稳定性有待提高。在准时率上均有 4 辆车保证了 100%的准时率，在装载率上差异也

不大。可以看出本文所提出的 RLFPSO 算法在解决带时间窗的车辆问题是所求解出的结果具有一定的参考价值，即 RLFPSO 算法适用于求解该类型的实际问题，具有一定的实际意义。

4.5 本章小结

本章首先介绍了车辆路径问题的相关理论知识，然后介绍了一种车辆路径问题的具体场景，并建立了优化模型。最后将本文提出的 RLFPSO 算法分别应用到了一个规模较小的案例中和一个规模较大的案例中进行求解，将得到的结果与其他 PSO 算法计算得出的方案和原始配送方案进行比较，结果表明本文提出的 RLFPSO 算法适用于求解带时间窗的车辆路径问题，具有一定的实际应用价值。

第五章 药品配送系统的设计与实现

5.1 需求分析

药品在现实生活中的存储和运输是一个庞大而又繁琐的过程。整个药品运输的过程中，从生产企业生产入库存储，到第三方物流企业运输到各地 CDC，整个过程相较于其他普通物流行业来说都较为复杂。不仅如此，在有些药品配送的过程中，还有一定的温度条件限制，导致运输的单位成本很高。在这种情况下，如果只靠人为的来安排配送方案或者不经过优化处理进行随机配送，就会使整个物流的成本上升。利用本文所提出的算法以及模型，并借助一定的开发工具，可以实现一个初步的药品物流配送系统，通过该系统来进行药品配送的路径规划，不仅可以降低配送成本，还能提高人力物力等资源的利用率。基于第四章中对某企业药品配送活动中的案例的求解结果，采用本文提出的 PSORFLF 算法，并结合 Java 进行编程实现一个简单的药品配送系统。

通过分析整个药品配送流程，归纳出药品物流配送系统的部分需求，具体如下：

（1）信息管理

信息管理是药品物流配送系统中的必要部分，在整个药品配送的过程中，涉及到许多单位或个人的信息，例如对服务的客户有编号、名称、地址、联系方式等信息；对于配送中心，有编号、地址、空闲车辆数等信息；对于生产企业来说有药品信息、库存信息等。这些信息都需要完整的录入系统中进行管理，除了方便管理者查询外，这些信息也是其他一些功能实现的数据基础。

（2）订单管理

对于药品生产企业来说，当客户对药品产品有需求时，则会向其发出订单要求，企业需要及时录入这些订单的信息，确认客户信息以及需求信息，以便后续配送过程的实施。

（3）库存管理

对于药品生产企业来说，对于库存中产品的管理也十分重要。库存管理不仅涉及到库存中药品的种类、数量、状态等信息的查看，还关系到后续的订单配送是否能正常运行。库存管理的内容包括对生产企业仓库中各种货物的数量、状态的管理以及记录产品入库、出库等相关的操作。

（4）配送管理

当企业接到一定数量的订单后就需要统一对这些订单进行配送，对整个配送过程进行一定的路线规划可以有效地提高物流配送的效率、降低配送的成本，同时

提高客户的满意程度。同时在配送的过程中也可以进行对车辆物流信息的追踪，并反馈给客户和企业，以保证在一些突发情况或者需求变更的情况下及时的进行配送方案的更改。本文主要研究的问题时配送路径的优化，故在配送管理中只对配送方案安排进行设计。

（5）其他辅助功能

药品物流配送的环节中还包括药品的逆向物流。药品生产企业需要处理药品运输过程中的产品损耗、药品的售后、退换货等过程。要做好这部分的管理，则需要整个系统实现对产品订单配送的售后反馈、评价等功能，有了这些信息，企业才能对整个药品配送的流程做出全方位的评估，同时也便于从中总结整个运作流程中存在的问题，做出相应的改进。

5.2 功能模块设计

跟据上文对药品配送系统的需求分析，结合药品运输的流程和特点，设计药品物流配送系统的功能模块框架如下图 5-1 所示。

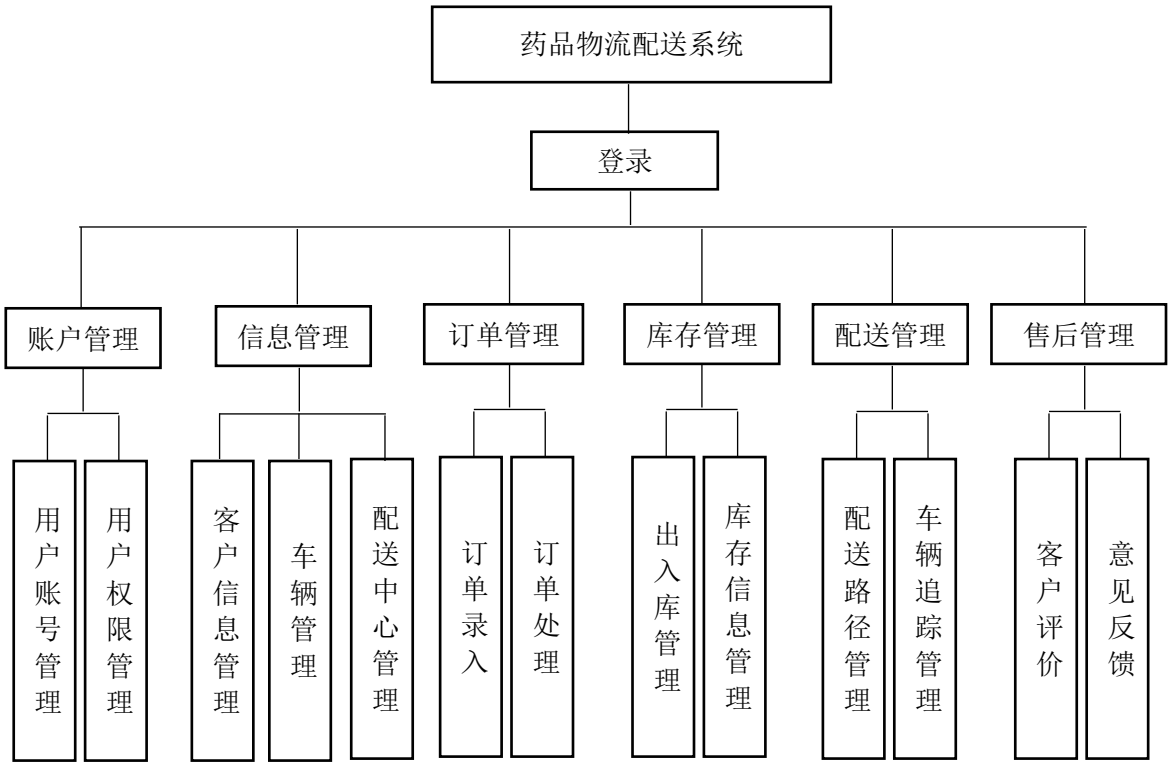

图 5-1 药品物流配送系统功能模块

药品物流配送系统的功能主要包括账户管理、信息管理、订单管理、库存管理、配送管理和售后管理这几大模块，每个模块中包含不同的功能。对每个功能模块具体介绍如下：

（1）账户管理模块

需要对登录系统的所有用户信息进行管理，通过账号管理和权限管理的形式实现。账号管理的部分主要体现在用户可以在系统中注册账户并登录，包括修改密码的功能。权限管理的功能则会为系统中的每一个用户分配权限，系统中的管理员可以对后台的数据进行增删查改等操作，而非管理员的普通用户只能对自己的账号信息进行管理。

（2）信息管理模块

由于系统进行最优配送路径规划需要知道所有客户点、配送中心、车辆、司机等信息，故对系统运行所使用到的所有信息进行管理，主要包括客户信息管理功能和配送中心信息管理功能。该模块可以直接对这两类信息的数据进行增删查改等操作。

（3）订单管理模块

为了确保客户所提出的需求可以得到满足，需将所有需求订单录入到系统中进行管理，这些订单的信息也是后续的配送路径规划中所必要的。该模块主要包括订单录入、查询、处理等功能。

（4）库存管理模块

对各种药品的库存状况进行管理，除了对仓库中药品的数量、状态等信息进行实时更新，还需要对各种药品的出库入库状态进行实时的记录。该模块主要包括出入库管理和库存信息管理功能。

（5）配送管理模块

该模块的功能是本文研究的主要内容，系统根据已经录入的订单信息、客户信息、配送中心信息等对所选的订单进行配送路线的规划，并输出最终的优化结果，即最优配送方案。该模块还有车辆追踪管理功能，主要对在途车辆的物流信息进行实时的监控和管理，由于该功能实现起来较为复杂，且不在本文的研究范围内，故不会进行详细的设计与实现。

（6）售后管理模块

对订单完成的情况，以及整个订单的运输流程进行记录。对订单后客户的反馈进行收集和统计，并录入系统，同时提供查询的功能，为后续完善和改进物流过程提供参考。

5.3 数据库设计

药品物流配送系统中的主体对象包括用户、药品、配送中心、车辆、司机、客户、订单。根据这些主体对象以及需求分析，可以画出实体关系模型图（E-R 图）。

本系统的 E-R 模型图如下图 5-2 所示。

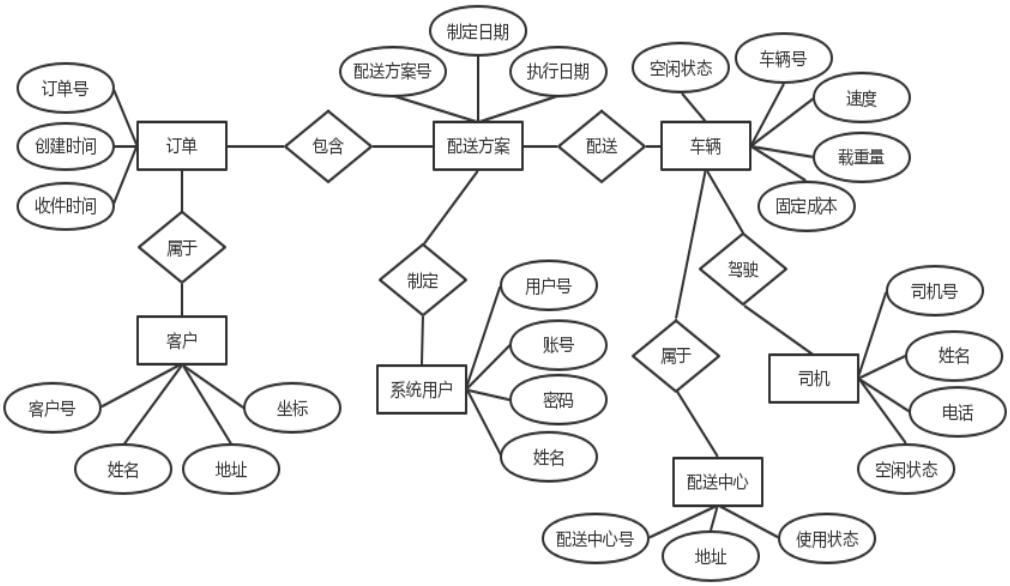

图 5-2 E-R 模型图

为了满足系统业务和功能的需求，并根据上面的 E-R 模型图，存储主体对象信息的表的具体设计如下。

除了上述主体对象的信息表外，再建立一个药品出入库信息记录表，来方便药品库存的管理和查询。通过该记录系统可以自动更新库存中各种药品的库存量状况，也可以更清楚的查阅库存变化信息。

表 5-1 用户信息表(user)

字段名	类型	备注
id	int	用户唯一标识
username	varchar(100)	用户名
password	varchar(100)	密码
name	varchar(100)	名称
authority	int	权限
signDate	date	注册时间

表 5-2 药品信息表(drug)

字段名	类型	备注
id	int	药品唯一标识
name	varchar(100)	名称
stock	double	库存量
tCost	double	运输成本
eCost	double	早到成本
lCost	double	晚到成本

表 5-3 配送中心信息表(transport_center)

字段名	类型	备注
id	int	配送中心唯一标识
longitude	double	经度
latitude	double	纬度
vehicles	int	车辆数
active	int	是否在使用

表 5-4 车辆信息表(vehicle)

字段名	类型	备注
id	int	车辆唯一标识
tcid	int	配送中心 id
capacity	int	最大容量
free	int	空闲状态
speed	double	车速
cost	double	固定成本
driverId	int	司机 id

表 5-5 客户信息表(client)

字段名	类型	备注
id	int	客户唯一标识
longitude	double	经度
latitude	double	纬度
name	varchar(100)	名称
tel	varchar(50)	联系方式
address	varchar(100)	地址

表 5-6 订单信息表(order)

字段名	类型	备注
id	int	订单唯一标识
cid	int	客户 id
did	int	药品 id
amount	double	需求量
earlyTime	date	最早送达时间
lateTime	date	最晚送达时间
finish	int	完成状态
v_id	int	配送车辆

表 5-7 司机信息表(driver)

字段名	类型	备注
id	int	司机唯一标识
name	varchar(100)	司机名称
tel	varchar(50)	联系方式
free	int	空闲状态

表 5-8 配送路线表(route)

字段名	类型	备注
id	int	路线唯一标识
vid	int	车辆 id
createTime	date	创建时间
startTime	date	开始时间
endTime	date	结束时间
uid	int	用户 id
detail	Varchar(100)	详细路线
finish	int	完成状态

表 5-9 库存操作日志表(store_log)

字段名	类型	备注
id	int	日志唯一标识
did	int	药品 id
amount	double	数量
operation	int	出/入库操作
time	date	操作时间

5.4 药品物流配送系统的实现

根据以上需求分析和功能模块的划分，实现药品物流配送系统。下面对系统中几个主要的功能模块的实现过程进行介绍。

5.4.1 开发环境

药品物流配送中的各种信息的管理，实际上是对数据库的管理，包含了数据库的增删查改等操作，故本系统是以一种将优化算法与数据库管理相结合的模式，将药品配送问题的解决进行可视化处理。由于 Java 语言面向对象的特性和丰富多样的类库，使其非常适合进行 Web 端管理系统的开发。根据对整个系统的需求分析和功能模块框架的构建，本系统采用 IntelliJ IDEA 作为开发工具，在 spring boot 框架下进行开发。进行配送路径优化所用到的算法通过 MATLAB 实现，系统中的软件框架采用 Java 语言进行编写，配送路径规划功能通过将 MATLAB 程序打包成 Java 可运行的 jar 包供 Java 程序调用。在执行过程中，Java 程序会调用相应的 MATLAB 算法，并获取返回的结果。

同时 Mysql 作为常用的与 spring boot 框架相结合的数据库，本文将使用其作为系统开发的数据库环境，进行各种有关信息存储和操作。

5.4.2 主要功能的流程

药品配送路径的规划是整个药品配送系统中最为重要的一个环节，它与系统中的其他模块都互相联系。整个配送系统的主要运作流程大致为：药品生产企业根据历史需求信息对未来各个品种药品的需求进行推测，通过查看对应药品的库存状态来进行药品的生产。当生产企业接收到来自各个销售点的药品订单需求时，生产企业根据客户订单的信息、各类药品的库存状态，利用系统所提供的配送路径规划功能指定出符合目前情况的配送成本最低方案，并根据得到的方案执行配送流程。在订单成功送达客户点后，搜集客户满意度、客户评价等反馈信息并录入到系

统中，整个冷链物流配送过程完成。本文所设计的系统中涉及到的主要功能包括基本信息录入（包括客户信息、订单信息、配送中心信息等）的功能以及配送路径规划的功能。

（1）基本信息录入功能流程

在本系统中，药品配送路径规划是最核心的功能，而这个功能的实现离不开各种基本信息的数据基础，例如配送中心信息、客户信息、订单信息等等，而这些信息都需要管理员在系统中进行录入。

系统录入信息的功能流程如下图 5-3 所示。

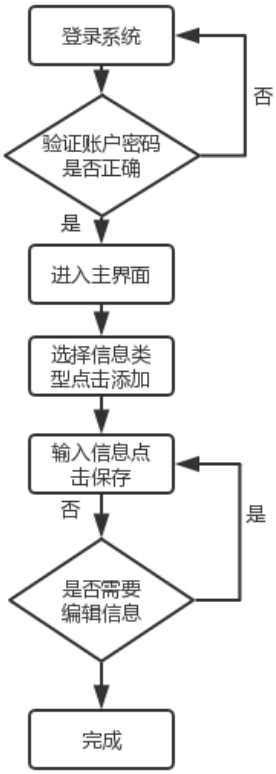

图 5-3 信息录入功能流程图

管理员输入账号密码后系统将会进行验证，若验证失败则需要重新输入，在输入正确后便可进入系统的主界面。在主界面中可以选择需要添加的信息类型，点击添加后便会弹出信息添加输入框，在输入相应的信息后点击保存便可以将信息保存在系统中，并在列表中查看。若需要对信息进行修改、删除等操作，可以直接点击相应的信息便可以进行编辑，点击保存按钮进行保存，从而完成整个对于信息的录入功能。

（2）配送路径优化流程

该功能是整个系统的核心功能，也是本文研究的主要内容，配送路径优化功能的主要流程如下图 5-4 所示。

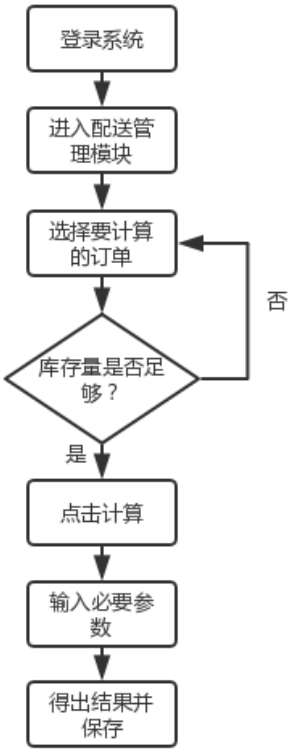

图 5-4 配送路径优化功能流程图

管理员登录药品物流配送系统后，需要对相应的基本信息进行管理，例如输入各个配送中心的信息、客户信息等，并按需要对这些信息作出删除修改等操作。随后需要将所接到的订单录入系统中，随后进入配送管理界面，在配送管理界面中选择将要配送的订单，点击界面中的生成配送方案按钮，系统会调用已经设置好的算法，根据系统中已经输入的信息进行计算并得出最终的最优配送方案，并在界面上显示结果，整个配送方案生成流程完成。

5.4.3 主要功能模块的实现

（1）用户登录

在用户使用系统之前，需要对用户进行认证，验证用户是否有权限进入本系统，如果登录失败则返回登录界面，如果认证成功，将跳转到系统主界面。系统的用户登录界面如下图 5-5 所示。

图 5-5 系统登录界面

（2）系统主界面

用户登录后则进入了系统主界面。用户进入主界面后可以点击相应的模块进入相应的信息查看和输入界面。如下图 5-6 所示。

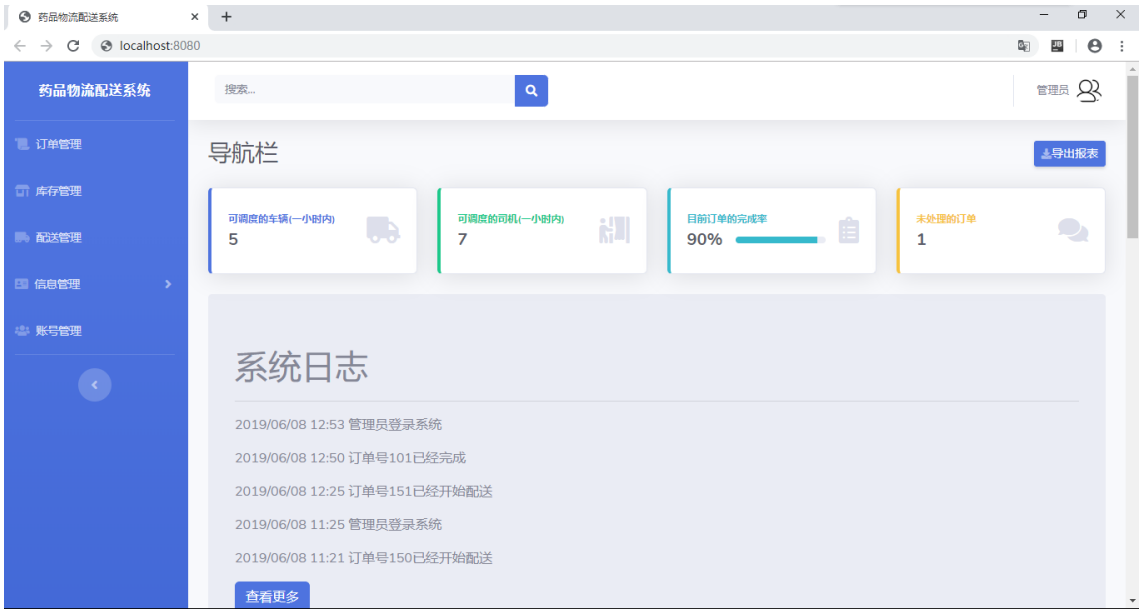

图 5-6 系统主菜单

（3）订单管理模块

点击主菜单中的订单管理，下拉框展开可以看到该模块的两个功能，分别为查看订单和录入订单，点击查看订单可以进入订单查看界面，在界面中会展示所有目前已经被录入系统的订单信息，管理员可以对订单的信息进行查看，点击相应的订

单则会弹出订单信息编辑界面，在列表右上角有订单信息搜索输入框，管理员可以通过输入关键字对特定的订单信息进行搜索。界面中的订单列表带有排序功能，用户可以点击字段，列表中的信息会根据所点击的字段进行排序。

在订单查询界面中可以看到订单目前所处的状态：未配送、配送中或者已完成。这些状态通过数据库中 order 表中的 finish 字段控制。当订单处于未配送状态时，在配送管理页面可以查询出这些订单并对未配送的订单进行路线规划；当订单处于配送中的状态时，页面上会显示该订单目前正在由哪辆车进行配送，并将对应车辆在 vehicle 表中的 free 字段进行修改，以标识车辆目前的处于已被使用的状态；当订单完成配送后，再将对应车辆的 free 字段的值修改回空闲状态，以便后续的路线规划使用。

整个界面如下图 5-7 所示。

图 5-7 查看订单界面

点击订单管理中的录入订单按钮，则会弹出订单信息录入界面，系统用户可以选择新建订单对订单进行录入，录入订单时用户可以选择已有的客户，系统会根据客户信息填写电话和地址，提高订单录入效率。如果该订单的客户在系统中不存在，系统可以在新建订单的同时保存客户的信息。在输入框中输入地址后，界面中的地图会自动定位到相应的地址处，以便添加时查看。在框中输入相关的信息后，点击保存，即可将新的订单信息在订单列表中显示出来。点击查看订单按钮便可以看到新添加的订单。信息录入界面如下图 5-8 所示。

在界面中，可以看到一些订单必要信息的输入框，用户需要根据实际订单信息进行输入，点击最早或最晚服务时间可以弹出日历框进行选择，输入完成并保存后便可以在订单列表中进行查看。点击取消则返回订单列表界面。

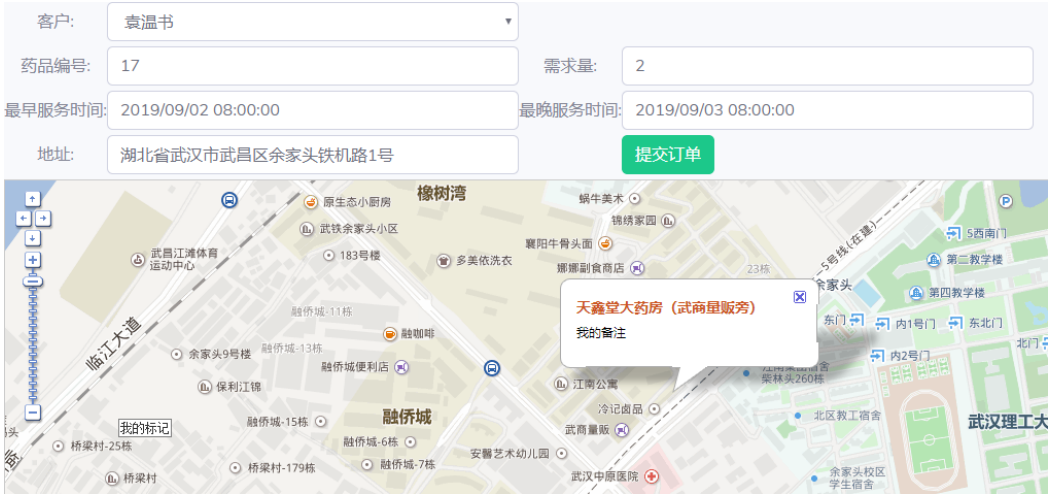

图 5-8 订单录入界面

（2）信息管理模块

点击主菜单中的信息管理模块可以看到该模块的三个功能：客户管理、车辆管理、配送中心管理。点击客户管理便会跳转到客户信息查询界面，界面上会展示客户列表，点击列表上方的添加按钮便可以弹出客户信息录入界面，输入相应的信息后点击保存便可以将客户信息保存到系统中，供用户查询。选择客户列表中的条目可以对相应的客户信息做编辑操作。该模块的列表同样支持搜索和排序功能。客户信息、车辆信息管理与配送中心管理功能类似，本文不再赘述。

信息管理模块主要是对车辆路径问题中的各种主体的信息进行保存和管理，以便后续配送路线的规划使用：

在客户信息管理中，各个客户的位置信息（经纬度）被保存下来，同时在录入订单的过程中，系统会识别数据库中已经保存的客户信息，若客户已存在于数据库中，可以直接选择以方便快速录入。

在车辆管理中，保存了车辆的车速、载重量、固定成本、所属配送中心等关键信息，同时通过 free 字段控制车辆可使用的状态，在配送管理模块中进行计算时，算法只会读取空闲车辆的数量作为参数，避免出现错误。

在配送中心管理中，保存了配送中心的坐标以及车辆数，在查询页面中，不仅会显示每个配送中心的总车辆数，还会显示配送中心的空闲车辆数，该数据是由 vehicle 表中的 free 字段控制，通过对配送中心对应的每辆车遍历查询该字段的状态并计算数量得出。同时配送中心还通过 active 字段来表示该配送中心是否可以使用，无法使用的配送中心在计算配送方案时无法选取。

以配送中心管理为例展示该模块的界面如下图 5-9 所示。

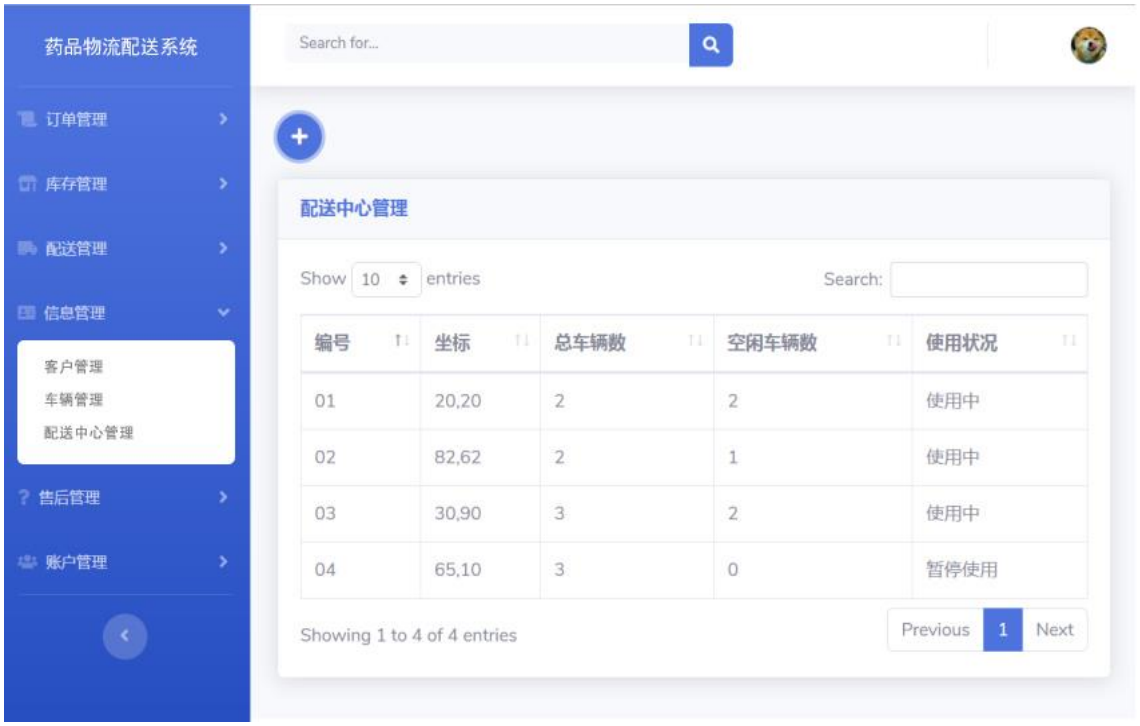

图 5-9 信息管理模块界面

点击列表上方的添加按钮，便会弹出配送中心录入界面，在界面中用户可以录入配送中心的信息。在信息输入完成后点击保存按钮则可以将配送中心的信息存入系统中，并在配送中心查询界面进行显示。在查询界面点击列表中的条目同样会跳转到录入界面，用户可以对所选的信息条目进行编辑。点击返回则回到配送中心查询界面。

在录入配送中心时，坐标的输入是指配送中心的经纬度，中间用逗号隔开，车辆数则是输入配送中心拥有的总车辆数，空闲车辆数通过 vehicle 表中的 free 字段信息自动生成。

配送中心信息录入界面如图 5-10 所示。

录入信息时，用户可以根据实际情况选择配送中心使用状况为使用中或暂停使用，若配送中心的使用状况为暂停使用，则在后续路径优化的过程中改配送中心将不可使用。配送中心的空闲车辆数会根据车辆的实际使用状况自动生成，不需要在录入信息时输入。

本模块中的其他信息模块基本与配送中心信息界面一致，只是在不同实体中有些字段上有所差别，本文不再赘述。

图 5-10 配送中心录入界面

（3）库存管理模块

点击主菜单中的库存管理选项可以看到药品库存管理以及出入库管理两个功能选项。出入库管理主要记录各品种药品的出入库情况，本文主要介绍药品库存管理功能。点击药品库存管理按钮便可以跳转到药品库存管理页面，页面中会显示出目前仓库中现有药品的一些信息，如库存数量、名称、编号等。药品的自身特性等信息也会被记录在该列表中如运输成本等，由于不同品种的药品这些信息都不相同，所以需要分别进行记录，这些信息同时也是后续进行配送方案计算时的必要数据。

同样该模块的功能也支持增删查改等操作，列表查询界面中也包含搜索和排序的功能，在药品库存信息录入时，只需要根据实际情况输入编号、名称、各项成本等信息，药品的库存数量会根据该模块中的另一个功能即出入库管理功能的数据自动生成并进行实时的更改。

同时，当进行后续的配送方案计算的操作时，订单上药品的需求量会与当时该药品的库存量进行比较。若订单所对应的药品库存量不足时，则该订单无法被选取。

库存管理页面如下图 5-11 所示。

图 5-3 药品库存管理界面

（4）配送管理模块

配送管理模块是本系统的最主要功能模块，它的作用是根据录入系统中的订单需求信息、客户信息、库存信息等数据，针对相应的实际问题进行求解，并将最终计算出的最优配送方案计算出来并提供给用户参考。该模块功能直接将系统中的数据作为基础，调用嵌入在系统中的优化算法并进行计算，输出结果并进行前端展示。

点击主菜单中的配送管理，则可以跳转到配送管理主界面。进入主界面后，可以看到目前所有已经录入系统中的订单信息，界面的主要内容与订单管理模块中的订单查询界面基本相同，但在该模块中多两个功能：1)在配送管理界面中，每一个订单条目的最左侧会有一个可勾选框，用户可以点击勾选或取消勾选，当用户在这一次的配送过程中只配送部分订单时，则可以将想用的订单条目上进行勾选，未被勾选的条目不会在计算最优路径是不会进行考虑；2)当用户选择完所有在本次物流过程中需要进行配送的订单后，点击页面上方的计算按钮，便会调用后台中的优化算法，并读取目前系统中的各项数据作为算法运行的数据基础，通过计算得出本次物流过程中运输成本最低的配送方案，并弹出计算结果。

在上述计算过程中，会调用到本文所提出的 RLFPSO 算法，系统中所存储的用户、车辆、订单等信息都会做为算法的参数进行计算。具体包括以下几个关键过程：

a. 调用 getRoute 方法通过对计算时所选择的配送中心以及订单的相关信息进

行查询，获取车辆数以及客户数量，并对车辆进行自动编号，再结合算法中当前粒子的位置获得粒子位置与配送路线之间的对应关系，该方法的具体代码如图 5-13 所示；

b. 得到配送路线后，再调用 getCost 方法，将上面得到的路线作为参数传入，由用户输入单位距离的运费，其余参数通过系统中所管理的信息获取，如车辆运行速度、最大装载量、客户需求的时间窗等。通过计算得到该配送路线运输时所花费的成本。用该成本作为粒子的适应度值，该方法的具体代码如下图 5-12 所示。

c. 根据本文提出的 RLFPSO 算法的流程进行粒子群的迭代，最终得出计算结果。

```
function cost = getCost(costs,routes,speed,freight,r_cost,timeTable,ke,kl,vehicles,cv,req,maxNumber)
%input : routes of vehicles, costs between points %output : total cost
cost = cv*vehicles; %固定成本
ttime = 0; %车辆行驶总时间
number = 0; %车辆装载件数
for i = 1:size(routes,1)
    startpoint = routes(i,1); %设置起点
    endpoint = routes(i,2);
    dis = costs(startpoint+1,endpoint+1);
    time = floor(dis/speed*60); %该路段的运输时间
    number = number + req(endpoint+1,1);
    if (number > maxNumber)
        cost = Inf; %禁止超过载重量
        break;
    end
    ttime = ttime + time;
    if(endpoint == 0)
        arrive_cost = 0;
    elseif(ttime > timeTable(endpoint,1) && ttime < timeTable(endpoint,2))
        arrive_cost = 0;
    elseif(ttime < timeTable(endpoint,1))
        arrive_cost = abs(ttime-timeTable(endpoint,1)) * ke;
    else
        arrive_cost = abs(ttime-timeTable(endpoint,2)) * kl;
    end
    cost = cost + dis*freight + time*r_cost + arrive_cost; %计算总成本
    if(endpoint == 0)  %回到起点
        ttime = 0;
        number = 0;
    end
end
```

图 5-4 getCost 代码

```
function [vehicles,Routes] = getRoute(X,m,n,starts)
%input : particle position, number of vehicles, number of costmers, startsof vehicles
%output : routes of vehicles
p1 = zeros(2,m*n);
p2 = zeros(2,n);
i = 1;
count = 1;
vehicles = 0; %使用的车辆数
%初始化p1
while i<=m*n
    for j = i:i+n-1
        p1(1,j) = count;
    end
    count = count+1;
    i = i+n;
end
for i = 1:m*n
    p1(2,i) = X(i);
end
p1 = sortrows(p1',2);
p1 = p1';
%初始化p2
for i = 1:n
    p2(1,i) = i;
    p2(2,i) = X(m*n+i);
end
p2 = sortrows(p2',2);
p2 = p2';

serve = zeros(1,m); %每辆车服务的客户数
for i = 1:n
    serve(p1(1,i)) = serve(p1(1,i))+1;
end

 for i = 1:m
     if serve(i)>0
        vehicles = vehicles+1;
     end
 end

Routes = [];
p = 1;
m = 1;
while p <= n
    if(serve(m)~=0)
        Routes = [Routes;[starts(2,m),p2(1,p)]];
        for i = 1:serve(m)-1
            Routes = [Routes;[p2(1,p+i-1),p2(1,p+i)]];
        end
        Routes = [Routes;p2(1,p+serve(m)-1),starts(2,m)];
    end
    p = p+serve(m);
    m = m+1;
end
```

图 5-5 getRoute 代码

图 5-13 所示的 getRoute 方法，是第四章中编码方法的实现。首先由编码方法可以得出粒子的维度是 $m*n+n$，其中 m 表示车辆数，n 表示客户数，$m*n$ 是 p_1 的长度，n 是 p_2 的长度。首先初始化 p_1，将粒子的位置矢量 X 按照编码方法拼接成 $m*n$ 个二位向量，并排序，根据前 n 个二维矢量第一维的值获得每辆车需要服务的客户数，并用 serve 数组保存起来。接着初始化 p_2 并根据位置矢量排序，得到车辆服务客户的顺序。最后根据 serve 数组中保存的每辆车服务的客户数以及被服务客户的顺序，就能得到每辆车服务的客户序列，并以（起点编号，终点编号）的形式保存在 route 数组里。

在图 5-12 所示的 getCost 方法中，根据 getRoute 方法中所得到的 route 数组以及其他模块中存储个各种数据信息，计算该配送方案的总成本。由于本文中研究的是带时间窗的车辆路径问题，所以在本系统中总成本主要包含了三个部分：固定成本、运输成本以及超出时间窗所支付的惩罚成本。该方法中，客户点之间的距离使用的是通过调用百度地图提供的 api 计算得到的最短干道距离。通过变量 ttime 记录从出发起行驶的总时间并与客户的需求时间窗比较，计算惩罚成本。通过 number 控制车辆上装载货物的总重量，若 number 超过车辆的载重量 maxNumber 则成本设置为无穷大，表示禁止该情况发生。配送管理的界面如下图 5-14 所示。

图 5-6 配送管理界面

上面的界面中一共有 4 个订单条目，使用本文所提出的优化模型和算法进行求解。将页面中的 4 个需求订单全部勾选，并点击计算按钮，便会弹出一个输入框，

输入计算时必要的参数，如下图 5-15 所示。

图 5-7 输入计算参数

点击计算，就能得到算法计算出的结果，如下图 5-16 所示。

图 5-8 计算结果

上图的计算结果中，分别显示了配送方案中不同配送中心的不同车辆分别的服务顺序路线。配送路线中则显示的将客户编号按配送先后顺序排列的序列。

上述过程中，计算后的路线会保存到配送方案中，以便后续查看。保存之后可以再配送管理模块中的配送查询标签下，查看所有已经保存的路线信息。如下图 5-17 所示。

Show 10 entries Search:

	路线编号	创建者	司机编号	完成状态	创建时间	执行时间	操作
☐	1	易玉堂	1	配送中	2019/09/02 08:00:00	2019/09/03 08:00:00	查看 删除
☐	2	易玉堂	2	已完成	2019/09/02 07:00:00	2019/09/02 10:00:00	查看 删除
☐	3	司子民	-	未配送	2019/09/03 08:00:00	-	查看 删除
☐	4	司子民	3	已完成	2019/09/02 08:00:00	2019/09/03 08:00:00	查看 删除

Showing 1 to 4 of 4 entries Previous 1 Next

图 5-17 配送查询界面

在图 5-17 中，选择某条路线点击查看。可以看到该路线在地图上的详细情况，包括途径的客户点等。如下图 5-18 所示。

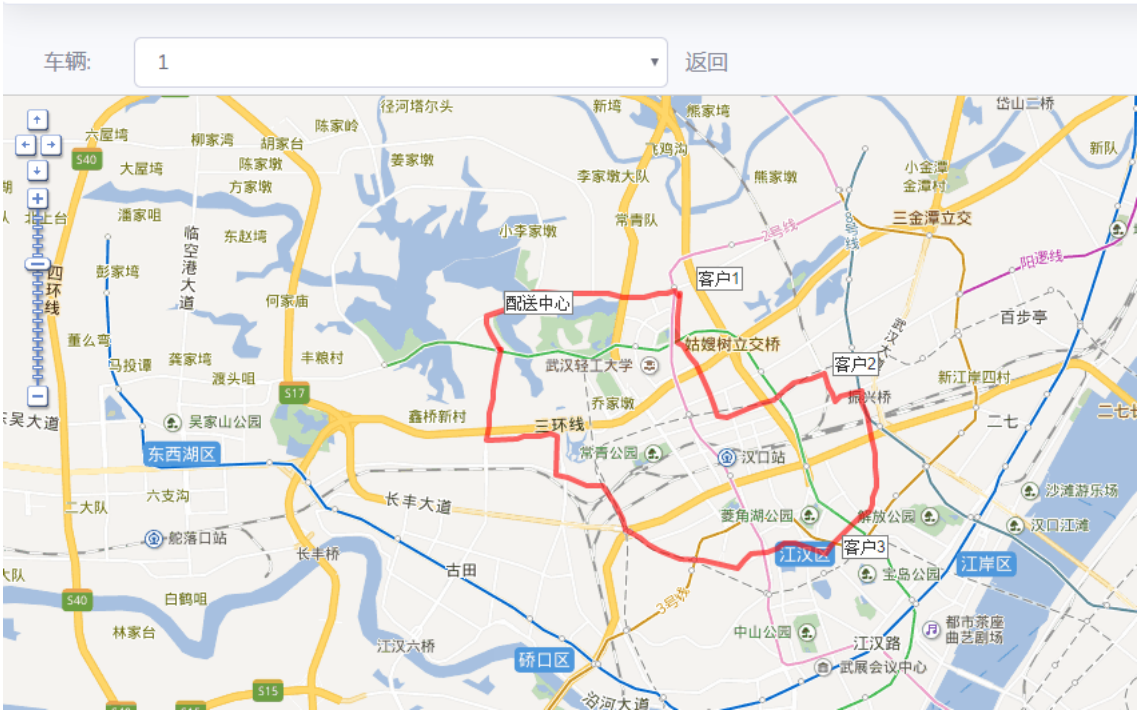

图 5-18 线路查询图

5.5 本章小结

本章首先根据实际对药品物流配送系统进行了需求分析，根据需求对系统进行了功能模块的划分。在此基础上，对整个系统的开发环境进行了说明，接着着重介绍了药品物流配送系统中的几个主要的功能模块的功能流程以及最终实现的过程，并展示了相关的页面，实现了药品物流配送优化的可视化系统。

第六章 总结与展望

6.1 全文总结

本文首先对粒子群算法进行了研究，包括它的基本原理、研究现状以及常见的改进方向。基于此，针对 PSO 算法的不足提出了一种改进的算法，通过仿真实验证明了算法的有效性。最后将改进的算法应用到了带时间窗的车辆路径问题中，证明该算法可以用于解决实际问题。本文的主要工作如下：

（1）对粒子群算法的研究起源、现状进行了归纳和总结。在对相关文献阅读和整理的基础上，对粒子群算法的基本原理、研究现状、特点、以及常见的改进方向等方面进行了阐述，介绍了粒子群算法常见的标准测试函数集，并分别介绍了各个函数的特点。

（2）针对粒子群算法容易陷入局部最优的特点，将莱维飞行和反向学习机制引入到粒子群算法中，提出了基于莱维飞行的反向学习粒子群算法(RLFPSO)。首先，粒子群算法在后期种群多样性下降过快，导致早熟收敛，粒子陷入停滞，当粒子停滞的代数大于某一个阈值时，对其使用反向学习的操作。在此基础上引入莱维飞行策略，使粒子反向学习的步长服从莱维分布，这样能够更快的逃离局部最优值。最后通过理论分析和实验证明了算法的可行性和有效性，在所给出的标准测试函数上，该算法的性能相较于其他算法有着更高的精度和收敛速度。

（3）将 RLFPSO 算法应用到了带时间窗的车辆路径问题中。首先对车辆路径问题和时间窗进行了简单的介绍。然后通过分析，将实际问题抽象为带时间窗的车辆路径问题数学模型，并给出了相应的假设条件和约束条件。最后通过实际案例的计算和优化，证明了 RLFPSO 的有效性，并且具有实际应用价值。

（4）初步实现了一个简单的药品物流配送管理系统，将本文所提出的算法嵌入到系统中，实现整个药品物流配送过程中的信息管理以及配送路径优化的可视化功能。

6.2 研究展望

本文研究的问题中，所提出的算法虽然相较于给出的改进算法有一定的性能提升，整个研究过程还有待完善之处。

（1）本文所给出的算法在单峰函数上有着较为明显的性能提升，但是在少数的多峰函数上的表现不够好，还有改进的空间。

（2）本文将算法应用到了 VRPTW 中，但应用场景仅限于带时间窗的车辆问

题中，其他优化目标以及约束条件下的应用，还有待挖掘。

（3）本文所实现的系统功能还有待扩展，例如利用全球定位系统、5G 技术等对运输途中的车辆、货物等进行实时的追踪和监控，并上传信息，使用户可以更好的掌控货物的信息。又例如可以将最优配送方案的计算结果在地图中可视化显示，这样更加方便直观。

致　谢

三年的研究生生涯转眼间即将结束。在这充实的三年期间，无论在学习上还是生活上，我都有很大的收获。在学位论文即将完成之际，我想对所有在这三年间帮助过我的人表示最诚挚的感谢！

感谢我的指导老师吴卫平教授，感谢吴教授在三年的研究生生涯中对我的指导和教诲。吴教授在我的研究生时光里，无论在生活中还是学习上，他都给予了我很大的帮助。在论文完成的过程中，吴教授也为我提供了许多建设性的意见和想法，可以说这三年来我取得的一切学习成果都离不开吴教授的帮助。感谢吴教授对我的辛勤培养！

感谢实验室中一起学习的同学们，他们对学习的认真，生活的严谨一直以来都是我学习的榜样。在平时的学习生活中，遇到问题他们也会不惜自己的时间帮我解答，给了我许多帮助。这几年在实验室的时光，由他们的陪伴，让我受益匪浅！

感谢我的室友以及一直以来陪伴在我身边的朋友们，在我遇到困难时给予帮助，在我心情低落时给我鼓励，这三年的时光有他们的关心和陪伴，才让我有勇气前进。

感谢一直在背后默默支持着我的家人，你们永远都是我最坚强的后盾，无私的给予我理解和关怀。正因为有他们才让我的研究生生涯一路顺利。

感谢参与我论文评审以及答辩的各位老师，感谢你们的辛苦工作！

最后，再次向所有曾经帮助过我、支持过我的所有老师、朋友、同学表示最衷心的感谢！

参考文献

[1] 吴和海,熊高峰,袁晋蓉,秦跃杰.一种适用于机组组合优化的改进整数编码粒子群算法[J].现代电子技术,2017, 40(11): 167-171.

[2] 陈福集,黄亚驹.基于 SAPSO_RBF 神经网络的网络舆情预测研究[J].武汉理工大学学报(信息与管理工程版),2017, 39(04): 422-426+438.

[3] 许振赐,刘君陶,王国栋,杨建平.基于 PSO_LSSVM 和 Elman 神经网络的北京市气温预测效果比较[J].河南农业科学,2013, 42(03): 157-160.

[4] 郭世凯,孙鑫.基于改进粒子群算法的移动机器人路径规划[J].电子测量技术,2019, 42(03): 54-58.

[5] Y. Shi, R. Eberhart. A modified particle swarm optimizer [C]. IEEE World Congress on Computational Intelligence, Anchorage, Alaska, 1998: 69-73.

[6] K. T. Chaturvedi, M. Pandit, L. Srivastava. Particle swarm optimization with time varying acceleration coefficients for non-convex economic power dispatch[J]. International Journal of Electrical Power & Energy Systems, 2009, 31(6): 249-257.

[7] Q. Jin, K. Wang, Z. Zhao, et al. HPSO Algorithm with High Speed Convergent based on Particle Health Degree[J]. Applied Mathematics & Information Sciences, 2014, 8(4): 1809-1821.

[8] J. J. Liang, A. K. Qin, P. N. Suganthan, et al. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions[J]. IEEE transactions on evolutionary computation, 2006, 10(3): 281-295.

[9] Z. H. Zhan, J. Zhang, Y. Li, et al. Adaptive particle swarm optimization[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2009, 39(6): 1362-1381.

[10] 王永贵,曲彤彤,李爽.基于指数衰减惯性权重的分裂粒子群优化算法[J/OL].计算机应用研究:1-6[2019-10-10]. https://doi.org/10.19734/j.issn.1001-3695.2018.10.0734.

[11] 夏学文,刘经南,高柯夫,李元香,曾辉.具备反向学习和局部学习能力的粒子群算法[J].计算机学报,2015, 38(07): 1397-1407.

[12] 霍林,陆寅丽.改进粒子群算法应用于 Android 恶意应用检测[J/OL].计算机工程与应用:1-10[2019-09-25]. http://kns.cnki.net/kcms/detail/11.2127.tp.20190318.1731.010.html.

[13] 张倩,张建丰,李涛,辛彦林,史娟.粒子群算法的改进及在农业水资源配置的应用[J/OL].排灌机械工程学报: 1-8[2019-09-25]. http://kns.cnki.net/kcms/detail/32.1814.TH.20190425.1646.006.html.

[14] 刘彩霞.基于模糊推理技术 PSO 算法的机器人路径规划研究[J].机电工程,2019, 36(04):

445-450.

[15] 张进峰,杨涛宁,马伟皓.基于多目标粒子群算法的船舶航速优化[J].系统仿真学报,2019, 31(04): 787-794.

[16] 熊伟.车辆路径问题建模与多目标进化算法的研究与分析[J].科学技术创新,2018(32): 28-29.

[17] G. Dantzig, R. Fulkerson, S. Johnson. Solution of a large-scale traveling-salesman problem[J]. Journal of the operations research society of America, 1954, 2(4): 393-410.

[18] G. B. Dantzig, J. H. Ramser. The Truck Dispatching Problem[J]. Manage Science, 1959, 10(6): 80-91.

[19] C. Clarke, J. V. Wright. Scheduling of Vehicles from A Central Depot to A Number of Delivery Points[J]. Operations Research,1964, 12: 568-581.

[20] G. Laporte, Y. Nobert. Exact algorithms for the vehicle routing problem[J]. Annals of Discrete Mathematics, 1987(31): 147-184.

[21] N. H. M. Wilson, N. J. Colvin. Computer control of the Rochester dial-a-ride system[M]. Massachusetts Institute of Technology, Center for Transportation Studies, 1977.

[22] A. PALMER. The development of an integrated routing and carbondioxide emissions model for goods vehicles[D]. Bedfordshire:Cranfield University, 2007: 24-81.

[23] 段征宇,雷曾翔,孙硕,杨东援.随机时变车辆路径问题的多目标鲁棒优化模型与算法[J/OL]. 西南交通大学学报: 1-9. http://kns.cnki.net/kcms/detail/51.1277.U.20181113.2320.012.html.

[24] 何东东,李引珍.多车型绿色车辆路径问题优化模型[J].计算机应用,2018, 38(12): 3618-3624+3637.

[25] 王杨,鲁晓春.时变路网下多配送中心多车型联合配送[J].科学技术与工程,2018, 18(36): 111-119.

[26] 杨翔,范厚明,徐振林,李阳.模糊需求下多中心开放式车辆路径优化[J/OL].计算机集成制造系统: 1-15[2019-03-12]. http://kns.cnki.net/kcms/detail/11.5946.TP.20181221.1624.026.html.

[27] M. Gendreau, G. Laporte, J. Y. Potvin. Metaheuristics for the capacitated VRP[C]. The Vehicle Routing Problem, Monographs on Discrete Mathematics and Applications, Philadelphia. MA SIAM, 1998: 129–154.

[28] 金淳,张雨,王聪.带时间窗车辆路径问题的分布式多 agent 蚁群算法[J].计算机应用研究,2018, 35(03): 666-670.

[29] 孙小军,介科伟.求解带时间窗动态车辆路径问题的改进蚁群算法[J].大连理工大学学报,2018, 58(05): 539-546.

[30] 戚远航,蔡延光,蔡颢,黄何列.带时间窗的车辆路径问题的离散蝙蝠算法[J].电子学报,2018,

46(03): 672-679.

[31] 石建力,张锦.粒子群算法求解需求随机的分批配送 VRP[J].计算机工程与应用,2018, 54(21): 230-239+264.

[32] 周蓉,沈维蕾.装卸一体化车辆路径问题的自适应并行遗传算法[J].中国机械工程,2018, 29(22): 2740-2749.

[33] J. Che, K. Zhou, et al. Application of hybrid artificial fish swarm algorithm based on similar fragments in VRP[C]. MIPPR 2017: Remote Sensing Image Processing, Geographic Information Systems, and Other Applications. International Society for Optics and Photonics, 2018, 10611: 106111L.

[34] D. Dechampai, L. Tanwanichkul, et al. A differential evolution algorithm for the capacitated VRP with flexibility of mixing pickup and delivery services and the maximum duration of a route in poultry industry[J]. Journal of Intelligent Manufacturing, 2017, 28(6): 1357-1376.

[35] Y. Gao, C. Wang, et al. An archived multi-objective simulated annealing algorithm for vehicle routing problem with time windows[J]. International Journal of u-and e-Service, Science an Technology, 2016, 9(12): 187-198.

[36] X. B. Gan, L. J. Liu, J. S. Chen, et al. Comprehensive learning PSO for solving environment heterogeneous fixed fleet VRP with time windows[C]. International Conference on Swarm Intelligence. Springer, Cham, 2016: 424-432.

[37] M. Okulewicz, J. Mandziuk. The impact of particular components of the PSO-based algorithm solving the Dynamic Vehicle Routing Problem[J]. Applied soft computing, 2017, 58: 586-604.

[38] C. Lagos, G. Guerrero, et al. An improved particle swarm optimization algorithm for the VRP with simultaneous pickup and delivery and time windows[J]. IEEE Latin America Transactions, 2018, 16(6): 1732-1740.

[39] J. Kennedy, R. C. Eberhart. Particle swarm optimization[A]. Proc of the First IEEE International Conference on Neural Networks[C]. Perth, Australia: IEEE Press, 1995. 1942-1948.

[40] A. Nickabadi, M. M. Ebadzadeh, R. Safabakhsh. A novel particle swarm optimization algorithm with adaptive inertia weight[J]. Applied Soft Computing, 2011, 11(4): 3658-3670.

[41] M. U. Farooq, A. Ahmad, A. Hameed. Opposition-based initialization and a modified pattern for Inertia Weight (IW) in PSO[C]. 2017 IEEE International Conference on Innovations in Intelligent Systems and Applications (INISTA). IEEE, 2017: 96-101.

[42] G. Spavieri, D. L. Cavalca, R. A. S. Fernandes, et al. An adaptive individual inertia weight based on best, worst and individual particle performances for the PSO algorithm[C]. International Conference on Artificial Intelligence and Soft Computing. Springer, Cham, 2018: 536-547.

[43] H. Garg. A hybrid PSO-GA algorithm for constrained optimization problems[J]. Applied Mathematics & Computation, 2016, 274(11): 292-305.

[44] W. Elloumi, Alimi M A. Combinatory Optimization of ACO and PSO[C]. Conference on Metaheuristics and Nature Inspired Computing. 2008: 184-216.

[45] B. Wei, Q. Peng, X. Chen, et al. An improved particle swarm optimization based on wolves' activities circle[C]. Intelligent Control and Automation. IEEE, 2012: 4557-4562.

[46] J. Liu, X. Zhang, A. Ning. Hybrid optimization algorithm of PSO and ABC[J]. Computer Engineering & Applications, 2011, 1(6): 226-229.

[47] H. Wang, H. Sun, C. Li, et al. Diversity enhanced particle swarm optimization with neighborhood search[J]. Information Sciences, 2013, 223(2): 119-135.

[48] X. Xia, B. Wang, C. Jin, et al. Self-adaptive Multi-swarm Particle Swarm Optimization Algorithm[J]. Journal of System Simulation, 2016(10): 155-159.

[49] 刘长平,叶春明.一种新颖的仿生群智能优化算法:萤火虫算法[J].计算机应用研究,2011, 28(09): 3295-3297.

[50] X. S. Yang, S. Deb. Cuckoo search via Lévy flights[C]//2009 World Congress on Nature & Biologically Inspired Computing (NaBIC). IEEE, 2009: 210-214.

[51] H. Haklı, H. Uğuz. A novel particle swarm optimization algorithm with Levy flight[J]. Applied Soft Computing, 2014, 23: 333-345.

[52] 林俊楷.生鲜食品冷链物流配送车辆路径优化[D].泉州:华侨大学, 2016: 18-20.