单位代码：10293 密　级：公开

南京邮电大学

专 业 学 位 硕 士 论 文

论文题目：若干车辆路径规划问题的启发式优化算法研究

学　　号	1220055509
姓　　名	储君
导　　师	赖向京
专业学位类别	电子信息硕士
类　　型	全日制
专业（领域）	电子信息
论文提交日期	2023 年 5 月

Research on heuristic optimization algorithms for several Vehicle Routing Problems

Thesis Submitted to Nanjing University of Posts and Telecommunications for the Degree of

Master of Engineering

By

JUN CHU

Supervisor: Prof. XiangJing Lai

May 2023

摘要

路径规划问题是经典的优化问题，该问题是在满足一定约束条件下找到一组总成本最低的路线。路径规划问题在 1959 年首次被提出，随着时间的变迁，从最初的精确算法到现在的元启发式算法，越来越多的算法都可以用来解决路径规划问题。本文针对路径规划问题，研究了该问题的三个变种问题，分别为容量约束的车辆路径规划问题，带有时间窗和容量约束的车辆路径规划问题，带有距离约束和容量约束的车辆路径规划问题，以上问题都属于 NP-hard 问题，所以对每个问题分别提出了一个高效的启发式算法：

（1）对于容量约束的车辆路径规划问题，本文提出了全新变邻域搜索算法，并用变邻域下降作为该算法的局部搜索策略，该算法结合了模拟退火的算法框架以及 Metropolis 接受准则。实验结果表明，该算法有较高的效率。

（2）对于带有时间窗和容量约束的车辆路径规划问题，该问题是带有运载限制的多车辆问题的变种问题，同样使用全新变邻域搜索算法，通过增加邻域数量来扩大邻域结构，从而提升算法的搜索性能，通过大量实验证明该算法具有高效性。

（3）对于带有距离约束和容量约束的车辆路径规划问题，该问题是带有运载限制的多车辆问题的变种问题，本文提出随机变邻域搜索算法，其中包括交换邻域，插入邻域和一个全新邻域—交换交叉邻域，经过扰动策略跳出局部最优解。实验结果表明，该算法有良好的计算效率。

本文对可以解决上述三种问题的 LKH 算法进行了加速优化操作，LKH 算法是目前解决路径规划问题效率高，精度准的启发式算法之一。

（4）LKH 算法作为实验的对比算法，本文对该算法进行了优化操作，通过优化罚函数重复计算的问题，可以使 LKH 算法在更短的时间内求出最优解。实验结果表明，在不改变搜索结果的前提下，优化后的 LKH 算法具有更高的效率。

关键词：路径规划，启发式算法，变邻域搜索，容量约束的车辆路径规划，时间窗限制，LKH 算法

Abstract

The vehicle routing problem is a classic optimization problem that involes finding a set of routes with the lowest total distance under certain constraints. The vehicle routing problem was first proposed in 1959, as optimization theory has progressed,the solution is from the exact alogirthm to the meta heuristic algorithm, more and more algorithms can be used to solve the vehicle routing problem. The paper studies three variants of the vehicle routing problem, namely the capacitated vehicle routing problem, the capacitated vehicle routing problem with time window and the distance constrained capacitated vehicle routing problem, all of these problems are NP-hard problems, so we propose an efficient heuristic algorithm for each problem:

(1)For the capacitated vehicle routing problem,this paper proposes a new variable neighborhood search,by integrating a randomized variable neighborhood descent method as its local search method. This algorithm combines the framework of simulated annealing and metropolis acceptance criterion. The experimental results show that the proposed algorithm is efficient.

(2)For the capacitated vehicle routing problem with time windows, this problem is a variant of the CVRP. For this problem, we also propose the new variable neighborhood search algorithm, this algorithm expand the neighborhood structure by increasing the number of neighborhoods, thereby improving the performance of the algorithm. Numerous experiments have shown that the algorithm is efficient.

(3)For the distance constrained capacitated vehicle routing problem, this problem is a variant of the CVRP, this paper proposes a randomized variable neighborhood search including the swap neighborhood, the insertion neighborhood and the swap-cross neighborhood, and the solution jump out of the local optimal traps by perturbation procedure. The experimental results show that the proposed algorithm has good efficiency.

This paper improved the LKH algorithm which can solve the above problems. The LKH algorithm is currently one of the most effective algorithms for the vehicle routing problem.

(4)For the LKH algorithm, this paper proposes an accelerated optimization operation to increase the computational efficiency of the LKH algorithm. The experimental results show that the improved LKH algotirhm is efficient.

Key words:vehicle routing problem,heuristic algorithm,variable neighborhood search,capacitated vehicle routing problem,time windows,LKH

目　录

第一章 绪论

1.1 最优化问题

最优化问题是一种常见的数学问题，它的目标是在一定的约束条件下，寻找一个使得某个函数取得最小值或最大值的解。在机器学习、数据分析、工程优化等领域中，最优化问题都是非常重要的，因为它们可以帮助我们找到最优的决策或模型参数。

最优化问题通常可以用如下形式表示：

$$\min f(x) \tag{1.1}$$

$$\text{s.t.} \quad g_i(x) \leq 0, \quad i = 1, \ldots, m \tag{1.2}$$

$$h_j(x) = 0, \ j = 1, \ldots, p \tag{1.3}$$

其中，*f(x)*是目标函数，$g_i(x)$和$h_j(x)$分别是不等式约束和等式约束，x 是变量向量。我们的目标是寻找一个变量向量 x，使得目标函数 *f(x)*最小化，并且满足所有约束条件。

最优化问题的解可以是全局最优解或局部最优解。全局最优解是指在所有满足约束条件的解中，使得目标函数取得最小值的解。而局部最优解则是指在某个区域内，使得目标函数取得最小值的解。在实际应用中，我们通常关心的是找到一个接近全局最优解的解。

连续优化和组合优化都是最优化问题的两个重要分支。最优化问题的本质是在一个给定的空间中，找到一个最优的解决方案，以满足一定的目标函数和约束条件。而连续优化和组合优化则分别针对不同类型的问题进行研究。在连续优化中，变量是连续的，而在组合优化中，研究的对象往往是离散的集合或者变量，在本文的研究中，研究对象就是组合优化问题，该类问题属于 NP-hard 问题，是无法在多项式时间内求解的。因此，在实际应用中，通常需要结合具体问题的特点和需求，采用启发式算法、元启发式算法等方法来求解，以获得较为实用的解决方案。

最优化问题在各种领域都有广泛的应用。例如，在机器学习中，最小化代价函数的过程就是一个最优化问题；在工程优化中，设计一个最优化的结构或者流程也可以转化为一个最优化问题。因此，学习最优化问题的理论和算法，对于深入理解各种领域的应用问题具有重要意义。

在本篇文章中，我们将深度研究容量约束的车辆路径规划问题以及该问题的变种问题的启发式优化算法研究。

1.2 问题的来源，研究背景和研究意义

本小节将介绍容量约束的车辆路径规划问题（Capacitated Vehicle Routing Problem）以及该问题的 2 个变种问题——带有时间窗和容量约束的车辆路径规划问题（Capacitated Vehicle Routing Problem with Time Windows）和带有距离约束和容量约束的车辆路径规划问题（Distance Constrained Capacitated Vehicle Routing Problem），将分别从问题的来源，问题的研究背景和问题的研究意义来进行系统的阐述。

货物运输在当今社会是一项重要任务，我们每天在物流上花费大量资金。因此，减少在交通上的运输成本是至关重要的，即使在某条路径上做出微小的改变，对整体运输成本的改善都是巨大的。

容量约束的车辆路径规划问题(CVRP)源于物流配送领域。该问题首先由 Dantzig 和 Ramser[1]在 1959 年提出，其核心目的是将一定数量的货物配送到多个客户，使得配送总距离最小，同时保证车辆的容量限制不被超过。随着物流配送业的不断发展，CVRP 问题也被广泛应用于城市物流配送、货运运输、医疗服务配送等领域。CVRP 问题源于现实生活中的物流配送问题。在实际的物流配送中，物流企业需要安排车辆进行货物的配送，如何安排配送路线和调度方案，减少物流成本，提高配送效率是物流企业面临的一个重要问题。此外，由于不同地区的城市环境、客户分布、道路交通等因素的差异，如何解决这些不确定性因素对配送方案的影响也是一个需要解决的问题。因此，CVRP 问题成为了一个非常具有挑战性和实际应用价值的研究课题。

CVRP 问题的研究不仅仅局限于物流配送领域，同时也涉及到城市交通管理、电力调度、医疗服务配送等领域。这些领域都需要设计合理的配送方案。同时，随着信息技术的不断发展，大量的实时配送数据可以用于 CVRP 问题的求解为 CVRP 问题的研究提供了更加丰富和多样化的数据来源。因此，CVRP 问题的研究具有非常广阔的研究前景和应用价值。

带有时间窗和容量约束的车辆路径规划问题(CVRPTW) 是 CVRP 问题的变种问题，它与 CVRP 问题的不同之处在于，在配送过程中，每个客户都有一个时间窗口，在这个时间窗口内必须完成配送任务。因此，CVRPTW 问题既要考虑车辆的容量限制，又要考虑时间窗口的限制，这使得问题的求解更加困难和复杂。CVRPTW 问题最早由 Bodin[2]提出，他将问题描述为"Time-Windowed Vehicle Routing Problem with Capacity Constraints"。他在研究中考虑了车辆的容量限制和时间窗口约束，提出了一种改进的车辆路径规划方法，并通过实例验证了方法的有效性。CVRPTW 问题的研究来源与 CVRP 问题类似，都源于实际生活中的物流配送问题。在实际的物流配送中，由于不同客户的需求量和配送时间的不同，配送任务的时间窗口

也会有所不同。为了更加高效地完成配送任务，物流企业需要合理地安排车辆的行程路线和调度方案以尽可能地减少物流成本，提高配送效率。而 CVRPTW 问题恰恰是这样一个问题，它可以用于解决物流配送中时窗口约束下的配送路线规划和调度问题。

与 CVRP 问题相比，CVRPTW 问题的难度更高，因为它不仅要考虑车辆的容量限制，还要考虑时间窗口的限制。在实际的物流配送中，时间窗口的限制往往是一种固定的约束条件，这使得 CVRPTW 问题更加现实和有意义。同时，CVRPTW 问题的研究也可以应用于其他领域如公共交通、航班调度等领域。因此，CVRPTW 问题的研究具有重要的现实应用价值和研究意义。

带有距离约束和容量约束的路径规划问题(DCCVRP)是 CVRP 问题的变种问题，与传统的 CVRP 问题不同，DCCVRP 考虑了车辆行驶距离的限制。在这个问题中每个车辆必须在行驶距离不超过一定限制的情况下完成所有客户的配送任务。DCCVRP 问题最早由 Laporte[3] 提出，他们在研究中将问题描述为"Distance Constrained Capacitated Vehicle Routing Problem"。他们考虑了车辆的容量限制和行驶距离的限制，提出了一种改进的车辆路径规划方法，并通过实例验证了方法的有效性。DCCVRP 的问题来源与 CVRP 问题类似，都源于实际生活中的物流配送问题。在实际的物流配送中，车辆的行驶距离受到道路条件、油耗、时间等多种因素的制约。如果车辆行驶的距离太远，将导致配送成本的增加，时间成本的增加，同时也会增加交通事故的风险。因此，物流企业需要对车辆行驶距离进行限制，以保证配送任务的高效完成和成本的最小化。

DCCVRP 问题的研究可以应用于实际物流配送中的路线规划和调度问题，同时也可以应用于其他领域，如城市规划、公共交通等领域。在实际生活中，交通拥堵、道路施工等因素也会影响车辆的行驶距离，因此，DCCVRP 问题的研究具有重要的现实意义和应用价值。近年来，随着计算机技术的不断发展，以及各种优化算法的出现，DCCVRP 问题的研究也得到了越来越广泛的关注和研究。

1.3 问题研究现状

在运输管理中，路径规划问题（VRP）具与许多实际应用的经济密不可分，所以该问题有重要意义，因此，许多研究人员一直致力于解决该问题。

车辆路径问题（VRP）最初由 Dantzig 和 Ramser 作为卡车派遣问题进行了介绍，因为其在运输物流中具有高度的实用性，此后受到了广泛的研究。VRP 的关键点在于确定一组路线，以满足所有客户的要求和操作约束，同时使每辆车从单个中转站出发并结束，目标是最小化

总运输成本。针对 VRP 问题，容量约束的车辆路径规划问题（CVRP）是由 Dantzig 于 1959 年提出的。1987 年，Solomon 在 CVRP 问题中加入了时间窗的约束，并且引入了一组算例，称为“所罗门算例”[2]。Laporte 为 VRP 问题提出过精确算法，同时 Larsen 使用了 Dantzig-Wolfe 分解的精确方法[3]来解决 CVRPTW。Lysgaard[4-6]等人提出了用于解决 CVRP 的分支和切割算法。VRP 问题无论是过去还是现在都被广泛研究。该类问题统一被定义为 NP-hard 问题，所以，近 10 年提出了大量启发式算法用于研究该问题，所以大部分 VRP 研究采用启发式方法来解决大规模问题。使用最多的元启发式算法包括禁忌搜索、模拟退火、遗传算法、大邻域搜索和可变邻域搜索。由于有大量学者致力于研究经典 VRP，因此许多启发式算法都已经被证明足够优秀可以找到最优解。Toth 在 2003 年[7]提出了基于受限邻域概念的颗粒禁忌搜索策略。Golden 在 1998[8]年以及 Wasil 在 2005 年[9]结合了记录到记录的旅行和可变长度邻域列表，发现了许多新的最优解。Mester 在 2005 年[10]提出了主动引导进化策略 (AGES) 并获得了许多最优解。这部分是因为使用了高质量的初始解决方案。其他方法，如 Prins 在 2009 年[11]提出的贪心随机自适应搜索过程和 Tarantilis 以及 Kiranoudis 在 2002 年[12]提出的阈值接受算法也得到了成功应用。可变邻域搜索（Variable Neighborhood Search[13]）已被证明是解决不同变体 VRP 最成功的元启发式算法之一，其中的文献包括但不限于 Chen 在 2010 年[14]提出的 IVNS，Fleszar 和 Osman 在 2009 年[15]提出的 VNS，Imran 和 Wassan 在 2009 年[16]提出的算法，Bräysy 在 2007 年[17]提出的改进版变邻域搜索算法，Polacek 在 2004 年[18]以及 Polat 在 2015 年[19]也提出用 VNS 算法来解决 VRP 问题。VNS 也被认为是解决许多其他组合优化问题的有竞争力的方法。例如，最新的研究包括列车调度和时刻表问题[20-21]以及相关的 VRP 的变种问题[22]。

近几年，随着人工智能的发展，深度强化学习(DRL)在研究 VRP 问题上也略有建树，具体地说，基于 DRL 的模型已经可以解决了部分 NP 难优化问题，包括最大切割问题(MC)[23]，旅行商问题(TSP)[24]和 VRP 问题。DRL 模型具有两个特点，使其在处理 VRP 问题时具有很大的应用前景。首先，DRL 能够估计有用的模式，这些模式在大规模问题中难以通过手动启发式方法[25]找到。其次，由于快速的路线生成过程[26]，因此 DRL 对于解决时间敏感的 VRP 具有很大的潜力。尽管如此，使用 DRL 解决路由问题时还面临着一些挑战：（1）尽管推断速度很快，但 DRL 模型的训练过程非常耗时[27]。因此，加速 DRL 模型的收敛速度是一个挑战。（2）需要交互式环境和大量实例来训练 DRL 模型。需要模拟器生成数据并与模型交互。（3）现有基于 DRL 的模型的解决方案质量不可与最先进的启发式算法（如 LNS）相媲美，这表明需要改进解决方案的质量。

还有其他一些学习框架被提出来解决 VRP 问题。Elias Khalil[28]在 2017 年使用了一个图

嵌入结构[29]和一个深度学习算法[30]。实验结果表明，在一组随机生成的小规模 VRP 问题实例中，他们的模型略逊于部分启发式算法（最远插入算法），但对于一些真实世界的数据集，深度 Q 学习算法略优于其他算法。Chaitanya[31]和 Alex Nowak[32]以有监督的方式训练了一个图卷积网络（GCN），直接输出一条路径的邻接矩阵，然后通过波束搜索将其转换为可行解。Michel Deudon[33]将 Irwan Bello[34]中的长短期记忆(LSTM)[35]架构替换为 Transformer[36]架构，实现了更有效的学习方法。

1.4 本章小结及全文结构

本章节中，首先对最优化问题进行了详细的描述，本章介绍了车辆路径规划问题的研究背景和意义，以及介绍了该问题的相关研究现状。车辆路径规划问题在物流、运输、旅游和城市交通等领域有着广泛的应用，是优化物流和交通运输效率的重要工具。当前，启发式算法是解决路径规划问题的主要方法之一，例如遗传算法、模拟退火算法、禁忌搜索算法等。

文章大体结构如下：

第一章详细介绍了车辆路径规划问题的研究目标和研究意义，并且探讨了一些目前解决效率最高的启发式算法。

第二章详细介绍了容量约束的车辆路径规划问题（CVRP）的问题描述和相关特点，并提出了解决该问题的变邻域搜索算法。通过计算大量的算例，并与当前最好的算法进行对比，分析了该算法的优点和缺点。

第三章详细介绍了带有时间窗和容量约束的车辆路径规划问题（CVRPTW），提出了解决该问题的算法。通过对算例进行计算并与其他算法进行结果对比，突出了所提出算法的先进性和有效性。

第四章详细介绍了带有距离约束和容量约束的车辆路径规划问题（DCCVRP），提出了解决该问题的启发式算法。通过对算例进行计算以及和其他启发式算法进行结果比对，对比结果体现了所提算法的先进性和高效性。

第五章详细介绍了对比算法 LKH 算法，并分析了 LKH 算法解决 CVRP 问题时的不足之处。并且，对 LKH 算法进行了优化加速操作，解决了该算法在解决部分算例问题上效率低的缺点。对比实验结果表明，优化后的 LKH 算法在效率上取得了很好的表现。

第六章对本文所做的工作进行了总结，并提出了算法未来的改进方向以及每种问题可能的发展方向，如进一步优化算法的初始化方法，改进邻域结构，引入深度学习等方法来提高算法的效率和精准度。

第二章 容量约束的车辆路径规划问题的算法研究

容量约束的路径规划问题（CVRP）是许多时刻调度的核心问题，可以应用于许多领域，如物流、配送、货运等，在本章节中，主要介绍了 CVRP 问题的问题描述，数学模型，以及对该问题的启发式算法研究。实验表明，本文提出的算法可以高效得解决此类问题。

2.1 容量约束的车辆路径规划问题

在本小节中，将详细介绍容量约束的路径规划问题（CVRP）包括问题描述、数学模型和研究现状。

2.1.1 研究现状

容量约束的路径规划问题（CVRP）是经典的组合优化问题，该问题在现实生活中有许多的应用场景，比如邮局的配送路线和校车的接送路线等等。同时，大量的文章在研究该问题，从现有的文章来看，解决该问题的算法分为了 2 大类，一类是精确算法，另一类是启发式算法。CVRP 问题的精确算法首次由 Laporte 和 Nobert[37]在 1987 年提出，精确算法可以保证在有限的步数内找到最优解，以下引用的文献是使用精确算法来解决 CVRP 问题的文章[38-41]。如今，用精确算法求解 CVRP 问题已经可以求解 360 个客户点的算例，最具有代表性的精确算法有分支切割算法[42]，列生成算法[43]，这些精确算法非常复杂，代码量达到了 10000 行以上，而且运行条件对计算机的性能要求非常高。所以，相比于精确算法，目前更多的研究都用到了启发式算法上。启发式算法是相对于精确算法提出的，相比于精确算法，启发式算法更简单，运算时间更短，可以解决的算例规模更大。

在解决 CVRP 问题上，最具代表性的启发式算法有基于贪婪算法进行迭代，不断优化可行解的 C-W 节约里程法[44]，带集合划分的迭代局部搜索算法(ILS-SP)[45]，基于统一混合遗传搜索算法求解[46-47]的，基于变邻域搜索算法求解的 Lin-Kernighan-Helsgaun[48](LKH-3)算法，基于 SISRs 算法求解的(Christiaens&Vanden Berghe，2020[49])。到目前为止，具有多样化控制的混合自适应迭代局部搜索方法[50]可以被视为 CVRP 的最先进启发式方法之一。LKH 算法最初用在解决 TSP 问题方面，该算法经过更新已经可以用于解决 CVRP 问题，CVRPTW 问题等等，是个求解精准的求解器。

在本章节中，我们将详细介绍提出的一种新的变邻域搜索算法去解决 CVRP 问题，通过对算例进行计算来测试该算法的性能同时也会和 LKH 算法进行详细的比较，从比较结果得知该算法对于小算例具有非常高的效率。

2.1.2 问题描述

容量约束的路径规划问题（CVRP）是经典的旅行商问题（TSP）的一个变种，即在满足条件的前提下，找出一组总距离最短的路线，该问题多用于物流派发及校车接送路线安排上。

CVRP 问题的示意图描述如图 2.1 所示，该示意图是有 13 个客户点和 1 个仓库点的问题，其中v_0为仓库点，其余均是客户点，有 3 辆车从仓库点出发其中v_1,v_2,v_3,v_4,v_5为第一辆车的路线路线，v_6,v_7,$v_8$$v_9$,$v_{10}$为第二辆车的路线，$v_{11}$,$v_{12}$，$v_{13}$为第三辆车的路线，这样就得到了一组可行解。该问题的最优解是使得得到的可行解的距离为最短。

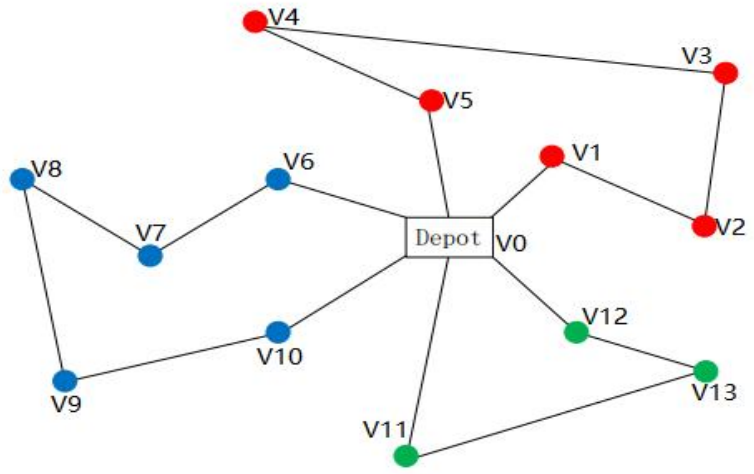

图 2.1 容量约束的车辆路径规划问题示意图

容量约束的路径规划问题描述如下：给定无向完全图 G = (V, E)，其中集合 V 是点集，V = $\{v_0, v_1, v_2, \ldots v_n\}$，其中$v_0$是仓库点，$v_1, v_2, \ldots v_n$是客户点，且各点之间互不重合，同时每个客户点都有一个需求量d_i，集合 E 是边集，E = $\{e_{ij}, i \in V, j \in V\}$，点与点之间的线段都包含在边集之中，每条线段有一个距离长度c_{ij}。在仓库点v_0有k辆车，每辆车的运载上限为 Q，要求每辆车从v_0出发，经过若干客户点之后返回v_0，在满足路线上客户端需求量总和不超过运载能力的条件下，求出一组行驶路线总最短的可行解。

容量约束的路径规划问题的基本数学模型描述如下:

$$\min \sum_{i \in V} \sum_{j \in V} \sum_{k \in K} c_{ij} x_{ij}^{k} \tag{2.1}$$

$$\text{s.t.} \quad \sum_{k \in K} y_{k}^{i} = 1 \tag{2.2}$$

$$\sum_{i \in N} X_{ij}^{k} = y_{j}^{k},\ \sum_{j \in N} X_{ij}^{k} = y_{j}^{k} \qquad j \in N, k \in K \tag{2.3}$$

$$\sum_{k \in K} y_{k}^{i} d_{i} \le Q \qquad i \in N \tag{2.4}$$

其中等式 2.1 为该问题的目标函数，c_{ij}代表了客户点v_i与客户点v_j之间的距离长度，x_{ij}^{k}=1 表示车辆 k 从客户点v_i到客户点v_j。等式 2.2 确保了每个客户点只能被一辆车访问，y_{k}^{i}=1 表示车辆 k 访问过客户点v_i，y_{k}^{i}=0 则表示辆 k 没有经过客户点v_i。等式 2.3 确保了每一个客户点访问前进来的车辆与访问后离开的车辆是相同的。等式 2.4 确保了每辆车访问的客户点的需求总和不超过车辆的运载上限 Q。

2.2 变邻域搜索算法

变邻域搜索算法（Variable Neighborhood Search[13]）是对局部搜索算法的一种改进，是一种改进型的局部搜索算法。变邻域搜索的基本思想是采用多个不同的邻域进行系统搜索。首先采用最小的邻域搜索，当无法改进解时，则切换到稍大一点的邻域。如果能继续改进解，则退回到最小的邻域，否则继续切换到更大的邻域，即通过不同的邻域结构到达不同的解空间，因此，使用多个邻域结构可以增强算法的搜索能力。如图 2.2 所示，邻域 1 和邻域 2 有不同的解空间，当在邻域 1 中找不到更好的解后，则切换至邻域 2 去继续寻找；在邻域 2 中找到更优解之后再退回到邻域 1 继续寻找，直至在邻域 2 中找不到更优解。

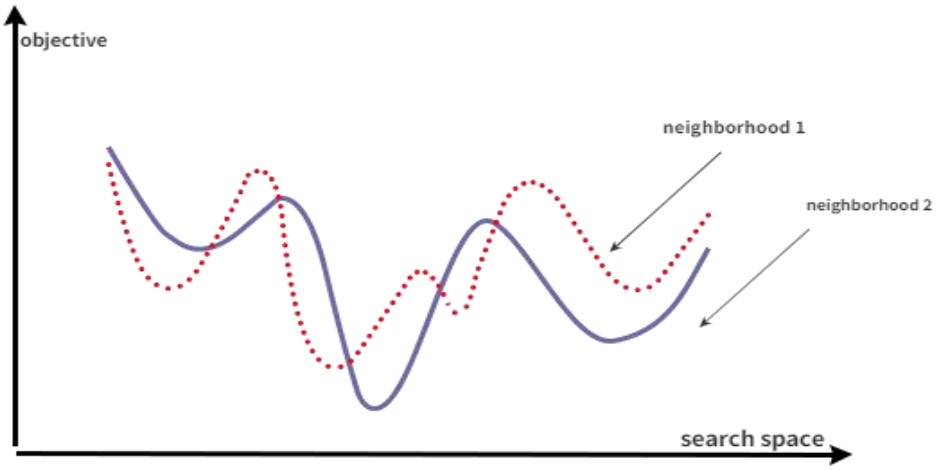

图 2.2 变邻域搜索算法

2.2.1 算法流程图

变邻域搜索算法主要由 4 部分组成，包括初始解的生成，迭代局部搜索，对可行解的扰动策略和接收准则。变邻域搜索算法的大体流程图如图 2.3 所示，如图所示，生成初始解后，先在邻域 1 中搜索，当邻域 1 中找不到更优解后就切换到邻域 2 中搜索，在邻域 2 中找到更优解后就退回邻域 1，否则就对解进行扰动操作。扰动后的解用接收准则来判断，若接受该解，则结束循环，否则继续将扰动后的解放入邻域 1 中继续搜索。

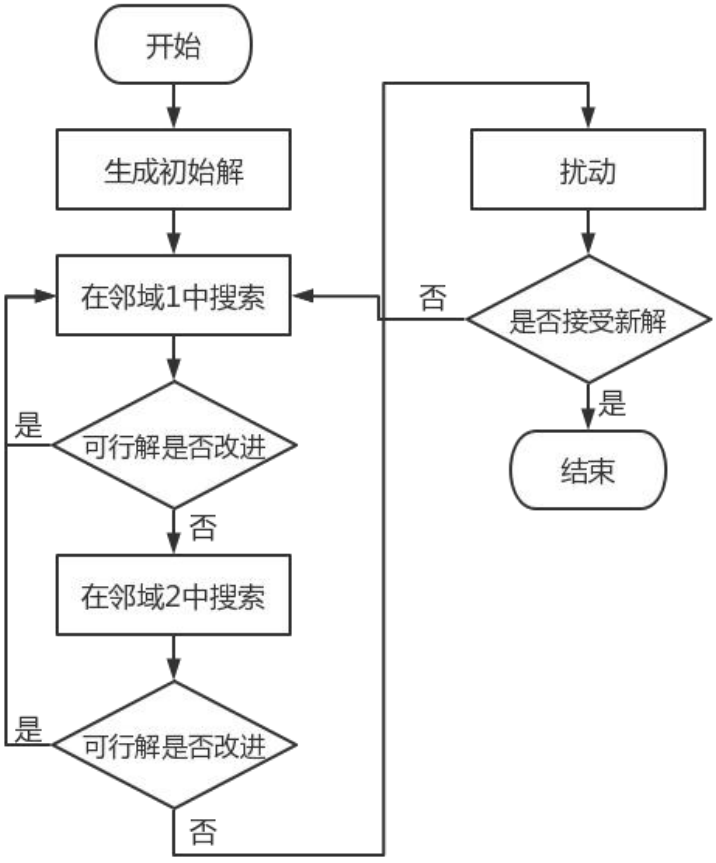

图 2.3 变邻域搜索算法流程图

2.2.2 初始解的生成

变邻域搜索算法中的初始解是通过贪心算法来生成的，具体步骤如算法 1 所示。从第一辆车开始，依次去访问客户点，当车辆无法承载下一个客户点的需求量时，车辆就返回仓库点，此时形成了初始解中的一条线路，下一辆车从上一辆车返回的客户点开始出发依次访问客户点，直到最后一辆车返回。

算法 2.1: 初始解的生成

输入: 点集合 N、需求量集合 D、车辆数量 K、运载能力 Q
输出: 初始解 S

```
demand ← 0, k ← 0
S_k ← 0
for i = 1 to N do
    demand += D[i]
    if demand >= Q then
        S ← S_k, k++, demand ← 0
end
return S
```

从第一辆车开始，依次去访问客户点，当车辆无法承载下一个客户点的需求量时，车辆就返回仓库点，此时形成了初始解中的一条线路，下一辆车从上一辆车返回的客户点开始出发依次访问客户点，直到最后一辆车返回。

2.2.3 局部搜索策略

1. 2-opt 算法

在解决多车辆路径规划问题中，2-opt 算法是一种有效可以优化可行解的局部搜索策略，2-opt 算法的描述如下：在给定的一条路线中，构建 2 条新的路径来取代已经存在的路径，使得新的路线的总长度更短，直到路线的长度无法更新至更短。图 2.4 展示了一次 2-opt 操作，如图所示，经过 2-opt 操作后，在原图中构建了$t_1 \rightarrow t_4$和$t_2 \rightarrow t_3$两条新路线，同时$t_1 \rightarrow t_2$和$t_3 \rightarrow t_4$被新路线取代，这样就通过 2-opt 得到了一条新的路线，需要注意的是，通过 2-opt 得到的新路线要保证路线的可行性。

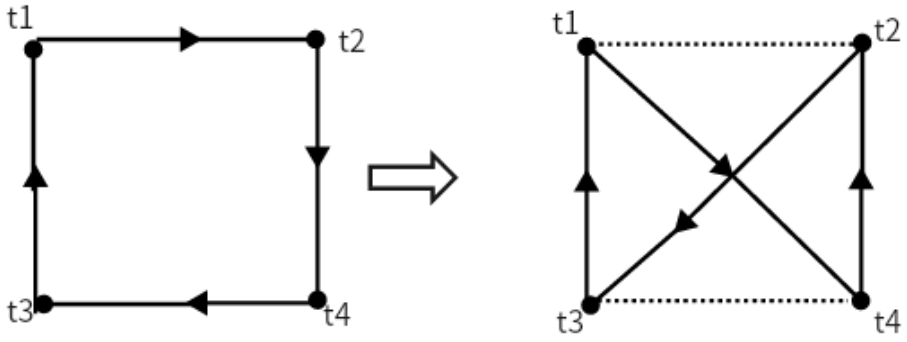

图 2.4 2-opt 操作

2-opt 算法的具体步骤如算法 2 所示，运用了 C 语言中的 rand()函数来进行随机的取点，当取到不同的点时，进行两边交换，计算交换之后路线的长度，若长度更短，则保留操作，将 cnt_opt 清空；该算法保证了每次更新路线之后，都能进行 1000 次的随机取点交换，防止

陷入局部最优。

算法 2.2:CVRP 的 2-opt 算法

输入: 可行路线 S、路线长度 Len
输出: 局部最优解 S*

```
1：Maxcount ← 1000
2：cnt_opt ←0
3：Len* ← 0
4：while cnt_opt≤ Maxcount do
5：    n1 ← rand(), n2 ← rand()
6：    if n1<=n2 then
7：        exchange(n1,n2), Len* ← 0
8：    else if n1 == n2 then
9：        continue
10：   end if
11：       Len* ← Length(S*)
12：   if Len*<Len then
13：       S ← S*, cnt_opt ← 0
14：   else
15：       cnt_opt++
16：end while
17：return S*
```

2. 变邻域下降

在本章节中提出的变邻域搜索算法采用随机变邻域下降（Variable neighborhood descent[47]）为局部搜索策略，在多车辆路径规划的问题上，该算法采用了 2 个邻域，分别是点插入邻域N_1和单点交换邻域N_2。

首先，在进入局部搜索之前给定一个可行解$S = \{R_1, R_2, \dots R_K\}$，点插入邻域$N_1$由公式 2.5 表示

$$N_1(S) = \{S \oplus (v, R_i, R_j) : R_i \neq R_j,$$

$$D(R_j) + d(v) \leq Q\} \tag{2.5}$$

在等式 2.5 中，(v, R_i, R_j)代表了一次点插入操作，将客户点 v 从路线R_i中插入到路线R_j中，$S \oplus (v, R_i, R_j)$表示在可行解集合中执行了一次点插入操作，$D(R_j)$表示了线路R_j中的总需求量，$d(v)$表示了即将移入线路R_j的客户点 v 的需求量。在执行完一次点插入操作后，都会分别对路线R_i和路线R_j进行一次 2-opt 的优化操作。值得注意的是，点插入的操作有可能会删除一条路径。图 2.5 表示执行了一次点插入操作，如图所示，原先路线R_i的顺序为$n_1 \rightarrow n_2 \rightarrow$

$n_3 \to n_4 \to n_5$，路径R_j的顺序为$n_6 \to n_7 \to n_8 \to n_9 \to n_{10} \to n_{11}$，在经过点插入后，将路径$R_i$中的$n_3$插入到路径$R_j$中，点插入操作过后路径$R_i$的顺序为$n_1 \to n_2 \to n_4 \to n_5$，点插入操作过后路径$R_j$的顺序为$n_6 \to n_7 \to n_8 \to n_3 \to n_9 \to n_{10} \to n_{11}$。

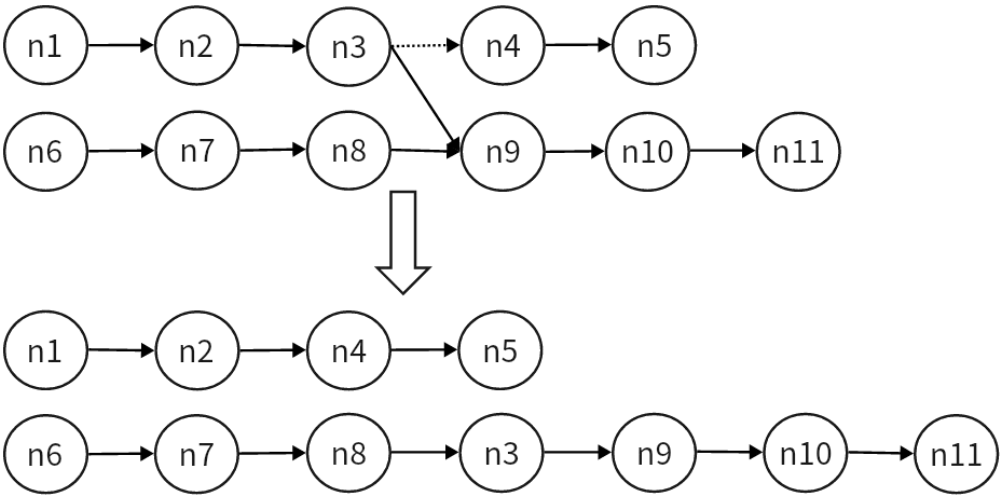

图 2.5 点插入邻域

点交换邻域N_2由公式 2.6 表示

$$N_2(S) = \{S \oplus Swap(u,v): u \in R_i, v \in R_j,$$

$$D(R_j) + d(v) - d(u) \le Q, D(R_i) + d(u) - d(v) \le Q\} \tag{2.6}$$

在等式 2.6 中，$Swap(u,v)$代表一次交换操作，将路径R_i中的客户点 u 与路径R_j中的客户点 v 进行一次交换操作，$S \oplus Swap(u,v)$代表执行了一次交换操作。在执行完一次交换操作后，同样会对路线R_i和路线R_j进行一次 2-opt 的优化操作。图 2.6 表示执行了一次点插入操作，如图所示，原先路径R_i的顺序为$n_1 \to n_2 \to n_3 \to n_4 \to n_5$，路径$R_j$的顺序为$n_6 \to n_7 \to n_8 \to n_9 \to n_{10} \to n_{11}$，在经过点插入后，将路径$R_i$中的$n_3$与路径$R_j$中的$n_7$交换，得到新的路径$R_i$和路径$R_j$，点交换操作过后路径$R_i$的顺序为$n_1 \to n_2 \to n_7 \to n_4 \to n_5$，点插入操作过后路径$R_j$的顺序为$n_6 \to n_8 \to n_3 \to n_9 \to n_{10} \to n_{11}$。

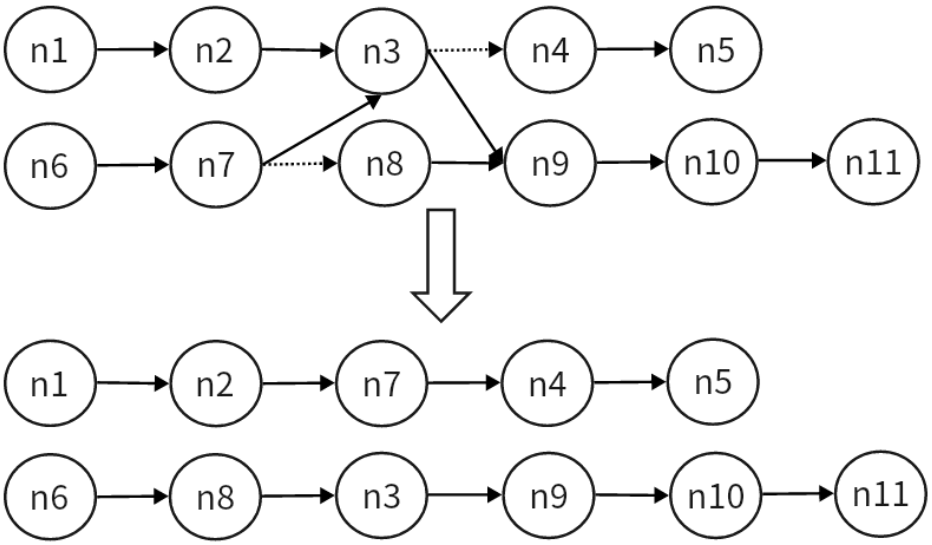

图 2.6 交换邻域

变邻域下降的具体实现如算法 3 所示，其中 $R=\{R_1,R_2,\dots R_k\}$表示是 K 条路径组成的解集合，L 表示 K 条路径的总长度，D（R_i）表示R_i路线上所有客户点的总需求量，F_1表示在邻域N_1中没有找到更优解的次数，F_2表示在邻域N_2中没有找到更优解的次数，F_{max}的值设为 5000 表示在某个邻域中最大的搜索次数。

算法 2.3: 变邻域下降算法（VND）

输入: 可行解 $R=\{R_1,R_2,\dots R_k\}$、长度 L
输出: 改进后的可行解 R*、改进后的长度 L*

```
L*← L
F1 ← 0，F2 ← 0
while F1 ≤ Fmax ∧ F2 ≤ Fmax do
    while F1 ≤ Fmax do 点插入邻域
        Ri ← Choose(R) 从可行解 R 中按序选择一条路线Ri
        Rl ← Choose(R)
        //从可行解 R 中按序选择一条路线Rl，保证Ri ≠ Rl
        v1 ← RandomChoose(Ri) 从路线Ri中随机选择一个客户点v1
        if D(Rl)+d(v1)>Q then
            continue
        end if
        Insert(Ri,Rl,v1)
        Ri*,Rl* ← 2-opt(Ri,Rl) //算法 2.2
        L* ← sum(R*)
        if L* < L then
            #当更新过的总路径小于原路径后，更新原路径里的解
            F1 ← 0, Ri ← Ri*, Rl ← Rl*, L ← L*
        else
            F1 ← F1+1
        end if
    end while 退出点插入邻域
```

```
    F_2 ← 0
    while F_2 ≤ F_max do   点交换邻域
        R_i ← Choose(R)
        R_l ← Choose(R) 保证R_i ≠ R_l
        v_1 ← RandomChoose(R_i) 从路线R_i中随机选择一个客户点v_1
        v_2 ← RandomChoose(R_l) 从路线R_l中随机选择一个客户点v_2
        if D( R_i)-d(v_1)+d(v_2)>Q∧
            D( R_l)-d(v_2)+d(v_1) then
                continue
        end if
        Swap(v_1,v_2) 交换v_1和v_2的位置
        R_i*,R_l* ← 2-opt(R_i,R_l) 对路线R_i和R_l分别进行 2-opt 优化操作
        L* ← sum(R*)
        //当更新过的总路径小于原路径后，更新原路径里的解
        if  L* < L then
            L ← L*, R ← R*, F_2 ← 0, F_1 ← 0
            break
        else
            F_2 ← F_2+1
        end if
    end while 退出点交换邻域
end while 结束变邻域下降
return R*,L*
```

为了解决 CVRP 问题，我们采用了双向链表和数组相结合的数据结构。考虑到点移动邻域需要改变某条路径中的客户点位置，我们选择了双向链表作为数据结构，以方便邻域操作。但是，在进行 2-opt 操作时，需要交换同一条路径中的客户点，因此我们也采用了数组作为数据结构。需要注意的是，双向链表和数组结构必须保持一致以确保正确性。

2.2.4 扰动策略

局部搜索策略能快速定位到最优解，不过这个最优解可能会是局部最优解，为了能搜索到全局最优解，通常采用扰动策略来使当前最优解跳出局部最优解以搜寻更大的解空间。 在本文的变邻域搜索算法中，对在搜索完点插入邻域N_1和点交换邻域N_2后的最优解 S 进行扰动操作。

首先，定义扰动算子（R_i,R_l,v_1,v_2）,对当前的最优解随机选取 2 条路线R_i和R_l，分别在 2 条路线中再随机选取 2 个客户点v_1和v_2，在满足运载能力的前提下交换 2 条路线中选中的客户点v_1和v_2，交换完成后无需对路线R_i和R_l进行优化操作。在经过扰动后，路径总长度会受到改变导致扰动后的解结果变差，在接受准则的判别下，扰动后的解也许被接受成为下一次迭

代的初始解，可以让最优解跳出局部最优从而达到全局最优解。

2.2.5 接受准则

在经过扰动策略之后，由于不同的解扰动的强度并不相同，有时候反而对路线更短的解起不到扰动的效果，因此，为了确保能扩大解的搜索空间，尽可能使当前解跳出局部最优，我们还需要确定是否需要接受扰动过的解作为下一轮迭代的初始解。

本文采用了 Metropolis 算法来决定是否接收扰动后的解作为下一轮迭代的初始解，Metropolis 算法基于一个参数 T，我们称之为温度，在经过扰动后，既不能完全接受当前最优解 S，也不能完全否定扰动解S^*，我们需要一个合理的概率来接受这个不如当前最优解的扰动解。接受概率 p 如公式 2.7 所示，其中$\Delta f = f(S^*) - f(S)$，接受概率与Δf和 T 有着直接的联系。Metropolis 算法可以调整 T 的大小，控制算法收敛速度。T 如果过大，就会导致算法收敛太快，更容易接受达到局部最优值就会结束迭代，如果取值较小，则计算时间会增加。Δf是扰动解S^*与当前最优解 S 的差值，显然，当$\Delta f \le 0$ 时，扰动解S^*比当前最优解 S 更好，所以是百分百可以接受扰动解，当$\Delta f > 0$ 时，扰动解S^*没有当前最优解 S 好，接受概率则根据温度参数 T 的大小来决定，从$e^{\frac{-\Delta f}{T}}$可以看出，当 T 值越大时，$e^{\frac{-\Delta f}{T}}$越大，那么接受扰动解S^*的概率就越大。

$$\mathrm{p}(\mathrm{S}, \mathrm{S}^*) = \begin{cases} 1, \ \Delta f \le 0 \\ e^{\frac{-\Delta f}{T}}, \ \Delta f > 0 \end{cases} \tag{2.7}$$

图 2.7 以坐标轴的形式表现了 Metropolis 接受准则，可以看出，当两个解的差值越大时越不容易接受该解。

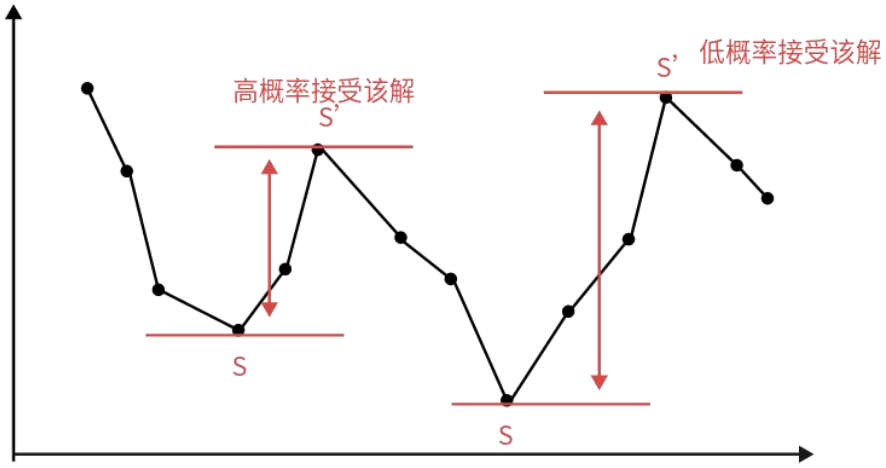

图 2.7 Metropolis 接受准则

在 Metropolis 算法中，温度参数 T 是一个可变的值，这里引入了模拟退火算法的概念，在模拟退火算法刚开始时，T 设置为一个较大的数T_0，因此，在刚开始几次迭代中，更容易接受扰动解S^*，这样做虽然耗费了更多的计算时间，但是可以使解的质量更高。随着迭代次数的增加，需要采用“降温”的操作，减小 T 值来适当缩小解的搜索空间，这里选用较为简单的的指数式下降，令 T = λT，其中λ是一小于 1 的正数，一般设置为 0.8 到 0.99 之间，可以控制 T 值下降的速度，在本文中，λ设置为 0.95。

2.2.6 算法伪代码的实现

随机变邻域搜索算法的具体伪代码算法 2.4 所示。其中$T_{initial}$表示初始温度，在本算法中设置为 5000，以便在迭代前期可以扩大搜索空间，跳出局部最优解。T_{end}表示截止温度，在算法中设置为 1e-8，λ是温度的下降指数，设置为 0.95。

算法 2.4: 随机变邻域搜索算法(random variable neighborhood search)

输入: 完全无向图 G(V,E)、车辆数量 K、运载能力 Q、温度参数($T_{initial}$,T_{end},λ)

输出: 最优解S^*

```
S = 初始解生成    //算法 2.1
S'← S
S* ← S
T ← T_initial
while T >T_end do
    for l=(1,2,...1000) do
        S' =对解 S'进行扰动操作
        S' = 对解 S'变邻域下降操作    //算法 2.3
        if f(S') < f(S) then
            S ← S'
        else
            Δf ← f(S') − f(S)
            r ← rand(0,1)   //r 是 0 到 1 之间的随机数
            if (r < e^(−Δf/T)) then
                S ← S'
            end if
        end if
        if f(S')<f(S*) then
            S* ← S'
        end if
    end for
    T ← T*q
end while
return S*
```

从伪代码中可以看出，首先用算法 2.1 生成出初始解 S，并且将当前初始解记录为最优解。在温度没有到达截止温度时，在循环中对当前解 S' 进行变邻域下降操作，用 Metropolis 接收准则来判断是否接收更新解 S'，当参数 T 小于截止温度T_{end}时，算法停止，此时解S^*为最优解。

2.3 实现结果

在本章节中，本文提出的随机变邻域搜索算法将详细对比在各个文献中广泛使用的 LKH 算法，将从最优解，平均解与计算时间三个维度对比每一个计算的算例。

2.3.1 实验环境与参数

为了测试随机变邻域搜索算法的性能，本文选取了 40 个使用最为广泛的算例，这些算例都是从 VRP 网站（http://comopt.ififi.uni-heidelberg.de/software/TSPLIB95/vrp/）上下载获取。我们将算例分为 A 和 B 两大组，每一个算例都是 X-nXC-kNV 的形式，其中 XC 为客户点和仓库点的数量，NV 为车辆数量，比如 A-n80-k10 表示 A 组算例有客户点和仓库点共 80 个，车辆 10 辆。为了尽可能缩小算法的误差，我们对每个算例进行 10 次计算，并求出其平均值，每个算例的运载能力 Q 都规定为 100。

本文的 VNS 算法（变邻域搜索算法）是用 C 语言编写，编译环境是 windows 下的 dev，运行的机器是 i7-4720HQ 的处理器和 8GB 的运行内存。

2.3.2 算例结果对比

A，B 两组算例的计算结果分别记录在表格 2.1 和表格 2.2 中，由表可见，f_{best}表示算法在 10 次计算中，计算结果最好的 1 次，f_{avg}表示算法在 10 次计算中的平均解，t_{avg}表示算法在 10 次计算中的平均计算时间，值得注意的是，计算时间是从算法开始到第一次找到最优解的时间，单位是秒，opt 表示每个算例的最优解。表格的第一列是每个算例的名称，第 2-4 列是变邻域搜索算法的计算结果，第 5-7 列表示 LKH 算法的计算结果。

表格 2.1 是对 A 组算例的计算结果，表格 2.2 是对 B 组算例的计算结果。

表格 2.1　A 组算例的计算结果与对比

算例	LKH 算法			随机变邻域搜索算法			opt
	f_{best}	f_{avg}	t_{avg}	f_{best}	f_{avg}	t_{avg}	
A-n33-k5	**661**	**661**	6	**661**	**661**	6	661
A-n34-k5	**778**	**778**	12	**778**	782	**9**	778
A-n36-k5	**799**	**799**	20	**799**	**799**	**6**	799
A-n37-k5	**669**	**669**	37	**669**	675	**6**	669
A-n37-k6	**953**	**953**	98	**953**	**953**	**6**	953
A-n38-k5	**731**	**731**	115	**731**	**731**	**8**	731
A-n39-k5	**822**	**822**	93	**822**	**822**	**5**	822
A-n39-k6	**831**	**831**	18	**831**	**831**	**8**	831
A-n44-k6	**939**	**939**	74	**939**	950	**5**	939
A-n45-k7	**1146**	**1146**	210	**1146**	1180	**4**	1146
A-n46-k7	**918**	**918**	15	**918**	945	**6**	918
A-n48-k7	**1073**	**1073**	95	**1073**	1122	**5**	1073
A-n53-k7	**1014**	**1014**	246	**1014**	1066	**19**	1014
A-n54-k7	**1172**	**1172**	386	**1172**	1203	**15**	1172
A-n55-k9	**1074**	**1074**	17	**1074**	1123	**13**	1074
A-n60-k9	**1357**	**1357**	344	**1357**	1367	**13**	1357
A-n62-k8	**1307**	**1307**	451	**1307**	1332	**17**	1307
A-n63-k9	**1628**	**1628**	496	**1628**	1689	**11**	1628
A-n63-k10	**1321**	**1321**	61	**1321**	1366	**11**	1321
A-n64-k9	**1419**	**1419**	14	**1419**	1463	**13**	1419
A-n65-k9	**1183**	**1183**	92	**1183**	1232	**16**	1183
A-n69-k9	**1167**	**1167**	205	**1167**	1204	**8**	1167
A-n80-k10	**1784**	**1784**	824	**1784**	1815	**13**	1784

表格 2.2　B 组算例的计算结果与对比

算例	LKH 算法			随机变邻域搜索算法			opt
	f_{best}	f_{avg}	t_{avg}	f_{best}	f_{avg}	t_{avg}	
B-n31-k5	**672**	**672**	**4**	**672**	**672**	5	672
B-n35-k5	**959**	**959**	**3**	**959**	**959**	5	959
B-n38-k6	**805**	**805**	41	**805**	**805**	**6**	805
B-n39-k5	**549**	**549**	24	**549**	**549**	**6**	549
B-n41-k6	**829**	**829**	**4**	**829**	834	6	829
B-n43-k6	**742**	**742**	**2**	**742**	753	7	742
B-n44-k7	**909**	**909**	**9**	**909**	922	**9**	909

B-n45-k5	**751**	**751**	10	**751**	762	**9**	751
B-n50-k7	**741**	**741**	**6**	**741**	**741**	8	741
B-n50-k8	**1325**	**1325**	92	**1325**	1344	**13**	1325
B-n51-k7	**1032**	**1032**	15	**1032**	1046	**7**	1032
B-n52-k7	**747**	**747**	23	**747**	755	**7**	747
B-n56-k7	**707**	**707**	11	**707**	**707**	**8**	707
B-n57-k7	**1165**	**1165**	733	**1165**	1176	**9**	1165
B-n57-k9	**1599**	**1599**	200	**1599**	1612	**8**	1599
B-n63-k10	**1510**	**1510**	240	**1510**	1534	**10**	1510
B-n64-k9	**1307**	**1307**	451	**1307**	1332	**12**	1307
B-n66-k9	**1320**	**1320**	974	**1320**	1334	**12**	1320
B-n67-k10	**1037**	**1037**	392	**1037**	1052	**12**	1037
B-n68-k9	**1272**	**1272**	35	**1272**	1288	**16**	1272
B-n78-k10	**1224**	**1224**	182	**1224**	1242	**16**	1224

由表2.1和表2.2可以看出，本文提出的变邻域搜索算法可以在更短的平均计算时间内（往往小于20秒）找到每一个算例的最优解，和性能强大的LKH算法相比，在客户点小于100个的前提下，本文提出的算法对最优解的搜寻效率是低于LKH算法的，但是在保证能找到最优解的前提下能以更短的计算时间找到最优解,在个别算例中，变邻域搜索算法运行10次的总时长都要少于LKH算法的平均计算时长。

2.4 本章小结

本章节主要介绍一种新的随机变邻域搜索算法（RVNS），用于解决容量约束的路径规划问题（CVRP）。CVRP是一个NP-hard的组合优化问题，也是实际应用中经常遇到的问题。为了有效地解决这个问题，通常需要采用启发式算法。本章提出的RVNS算法包含四个主要部分，分别是生成初始解、局部搜索、扰动策略以及基于Metropolis算法的接收准则。这些部分的相互作用可以使得该算法在非常短的时间内找到算例的最优解。

为了测试RVNS算法的运行性能，我们采用了44个基本算例进行了实验。实验结果表明，相对于LKH算法，RVNS算法在计算时间上更加有效率，能够在短时间内找到最优解。然而，在搜索结果上LKH算法的效果更好，其搜索精度更高。LKH算法采用了大邻域策略和分支定界策略，需要考虑更多的状态和搜索路径。与此相比，RVNS算法采用局部搜索策略，通过搜索问题的解的邻域来寻找更优的解，可能会找到更多局部最优解，但也有可能陷入局部最优解中。因此，在解决CVRP问题时，LKH算法的解决精度可能更高。

综上所述，本章提出的RVNS算法在小规模的CVRP问题上表现出色，可以有效地找到

最优解。尽管与 LKH 算法相比，VNS 算法可能存在一定的局限性，但其短时间内找到最优解的能力还是非常值得肯定的。因此，在实际应用中，我们可以根据具体情况选择合适的算法来解决 CVRP 问题。

第三章 带有时间窗和容量约束的车辆路径规划的算法研究

在本章节中，我们将详细介绍容量约束路径规划问题（CVRP）的变种问题——带有时间窗和容量约束的车辆路径规划问题（CVRPTW）和带有距离约束和容量约束的车辆路径规划问题（DCVRP）。其中，CVRPTW 在原问题上又增加了时间窗这一约束条件，而 DCVRP 在原问题的基础上增加了距离限制这一约束条件，这两个问题同样可以用变邻域搜索策略进行求解，在大量算例的计算下，我们提出的算法在小算例问题上具有高效快速性。

3.1 带有时间窗和容量约束的车辆路径规划

本章节将具体描述 CVRP 问题的一个变种问题带有时间窗和容量约束的车辆路径规划问题（CVRPTW），将从问题描述，研究背景和算法描述详细介绍这一问题。

3.1.1 研究背景

带有时间窗和容量约束的路径规划问题(CVRPTW)是一个经典的组合优化问题，涉及到配中心向多个客户点配送货物的最优路径规划问题，该问题在物流、配送等领域有着广泛的应用。CVRPTW 问题的复杂度非常高，因为它需要同时考虑车辆的容量限制、时间窗口和路径规划等多个约束条件。在研究这个问题上，许多学者提出了不同的算法和方法，其中可以分为精确算法和启发式算法两种，其中精确算法是一种通过枚举所有可能的解来寻找最优解的方法。其中最常见的精确算法包括分支定界算法、分支限界算法、动态规划算法等。这些算法可以保证找到全局最优解，但是在实际应用中往往由于计算复杂度过高而无法便用。在 CVRPTW 问题上，精确算法基本是基于分支定界法，其中包括 Dantzig-Wolfe 分支定界法：是最早的求解 CVRPTW 问题的算法之一，由 Dantzig 和 Wolfe 在 1959 年提出。该算法将问题分解成子问题，在每个子问题上使用线性规划求解，通过对子问题的求解来找到最优解(Jepsen 和 Spoorendonk[52]在 2011 年运用到 CVRPTW 中)，前沿算法(Frontier-based algorithm[53])由 Savelsbergh 和 Sol 在 1995 年提出。该算法将所有客户按照其时间窗口的开始时间排序，并在每个时间窗口的前沿点处进行分支，将问题分解成多个子问题，通过求解这些子问题来找到最优解，分支定界和修剪算法 (Branch-and-bound with pruning)[54]由 Gendreau 和 Laporte 在 1988 年提出。该算法使用分支定界法求解 CVRPTW 问题，并通过剪枝策略减少搜索空间，从而提高求解效率。

对于精确算法无法解决大算例的缺点，更多启发式算法应运而生，常见的解决 CVRPTW 问题的启发式算法包括禁忌搜索[55]（Tabu Search）：由 Glover 在 1997 年提出。该算法通过维护一个禁忌表来避免搜索陷入局部最优解，同时引入随机性，以在一定程度上跳出局部最优；遗传算法[56]（Genetic Algorithm）：由 Holland 在 1975 年提出。该算法通过模拟生物进化过程来生成新的解，并通过选择、交叉和变异等操作来优化解，蚁群算法[57]（Ant Colony Optimization）：由 Dorigo 在 1997 年提出。该算法模拟蚂蚁寻找食物的行为，通过信息素的作用和启发式信息来指导搜索过程，Particle Swarm Optimization[58]（粒子群算法）：由 Kennedy 和 Eberhart 在 1995 年提出该算法模拟鸟群或鱼群等集体行为，通过每个粒子的位置和速度来搜索最优解。Chiang 和 Russell[59]通过模拟退火算法求解了 VRPTW，Ombuki 和 Berger 等人使用遗传算法解决了 CVRPTW[60-61]。Cordeau[62]等人关于 CVRPTW 分别介绍了精确算法和启发式算法。Pisinger[63]使用自适应大邻域搜索（ALNLS）解决 CVRPTW，其中最常见的操作是拆除边和重组边。Chand[64]等人提出了一种基于遗传算法的启发式算法，该算法可以同时最小化车辆数和总距离。Wang[65]设计了多目标 CVRPTW 排序的混合遗传算法，可以最大得缩短形式总距离并且能满足时间窗要求。Tavakkoli[66]等人使用模拟退火算法解决了一种具有硬时间窗口的 CVRPTW，其中要求最小化车辆数，总行驶距离和违反时间窗的罚函数。Tan[67]等人利用遗传算法求解了 Solomon 提出的 CVRPTW 算例，并取得了较好的计算结果。

元启发式算法结合了多种启发式算法的深度搜索空间。该类算法允许接受结果较差的解，从而可以跳出局部最优解。近年来顺序和并行算法在解决 CVRPTW 问题上使用得较为频繁。Zhong 和 Pan[68]在 Debudaj Grabysz 和 Czech[69]以及最近的 Li[70]成功地将元启发式算法应用到了模拟退火。禁忌搜索元启发式算法是由 Cordeau[71]和 Sin[72]等人提出的。Xuan[73]和 Qi and Sun[74]等人提出了蚁群算法的元启发式。关于元启发式算法的调查可以在 Gendreau[75]的论文中找到。

3.1.2 问题描述

带有时间窗约束的多车辆路径规划问题（CVRPTW）是 CVRP 问题的一个变种，在原有的条件基础上，每一个客户点增加了时间窗的约束限制$[e_i, e_l]$，同时增加了车辆对客户点的服务时间S_t，要求每辆车必须在时间窗内到达客户点，否则就会有惩罚值 P。如图 3.1 所示，其中横坐标是时间 t，纵坐标是惩罚值 P，当车辆在时间窗下限e_i之后并且在时间窗上限e_l之前到达，则惩罚值为 0，当车辆在E_i之后并且在e_i之前到达罚函数呈线性下降的趋势，当车辆在时间窗上限e_l之后到达并且在L_i之前到达惩罚值呈线性上升，在E_i之前到达或L_i之后到惩罚值

分别为 S1 和 S2。

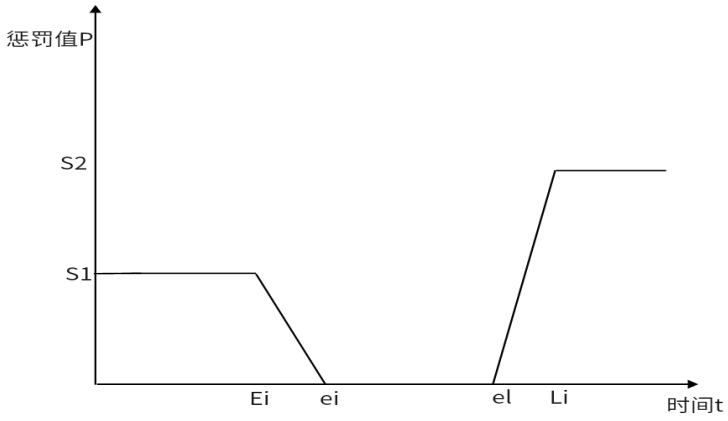

图 3.1 惩罚值 P 与时间 t 关系图

CVRPTW 问题中的时间窗口是客户点对服务时间的限制，又能分为以下两种：硬时间窗口和软时间窗口。硬时间窗口是指客户有一个固定的开始时间e_i和固定的结束时间e_l，要求客户必须在这个时间段内得到服务，如果在时间窗之外到达，客户将无法得到服务。软时间窗口如图 3.1 所示，在时间窗口$[e_i, e_l]$之外仍能得到服务，只是将产生违反时间窗口的成本，即惩罚值。在实际问题中，时间窗口限制通常是硬时间窗口或软时间窗口。通常情况下，为了减少违反时间窗口带来的成本，算法会尽可能地在时间窗口内为客户提供服务。然而，在某些情况下，为了更好地平衡车辆路径的长度和时间窗口限制，算法可能会选择违反时间窗口。这需要在算法中进行明确定义和约束。

CVRPTW 的问题描述具体如下：给定无向完全图 $G = (V, E)$，其中集合 V 是点集，$V = \{v_0, v_1, v_2, \ldots v_n\}$，其中$v_0$是仓库点，$v_1, v_2, \ldots v_n$是客户点，且各点之间互不重合，同时每个客户点都有一个需求量d_i和一个时间窗口$[T_i, T_l]$，集合 E 是边集，$E = \{e_{ij}, i \in V, j \in V\}$，点与点之间的线段都包含在边集之中，每条线段有一个距离长度c_{ij}。从仓库点v_0出发 K 辆车，对于每个客户，必须在其时间窗口内到达并完成配送，且每辆车的配送总量不能超过容量限制 Q，最后每辆车返回仓库点v_0，该问题的目标是最小化配送路线的总长度。

具体的数学模型如下

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{3.1}$$

$$\text{s.t.} \quad \sum_{i \in V} x_{ij} = 1 \qquad j \in V \tag{3.2}$$

$$\sum_{i \in V} d_i x_{ij} \leq Q \tag{3.3}$$

$$\sum_{i \in V} x_{ij} \leq (k-1) + \sum_{i \in V} d_i / Q \tag{3.4}$$

$$u_i + q_i = u_j,\ C_j \geq F_i + c_{ij} \qquad j \in V \tag{3.5}$$

$$T_i \leq C_i \leq F_i \leq T_l \qquad i \in V \tag{3.6}$$

其中表达式 3.1 是 CVRPTW 问题的目标函数，3.2 表示每个客户点被访问一次且仅一次，其中x_{ij}表示客户点 i 到客户点 j 之间是否有车辆经过，如果有则x_{ij}=1，否则x_{ij}=0；3.3 表示每辆车的约束条件，总重量不能超过运载上限 Q；3.4 保证了每个客户点的需求量都被满足；3.5 和 3.6 保证了每个客户必须在其指定的时间窗内被服务。图 3.2 为 CVRPTW 问题的示意图，其中每个客户点都有自身的时间窗约束，车辆要求按每个客户点规定的事件窗内到达该客户点位置。

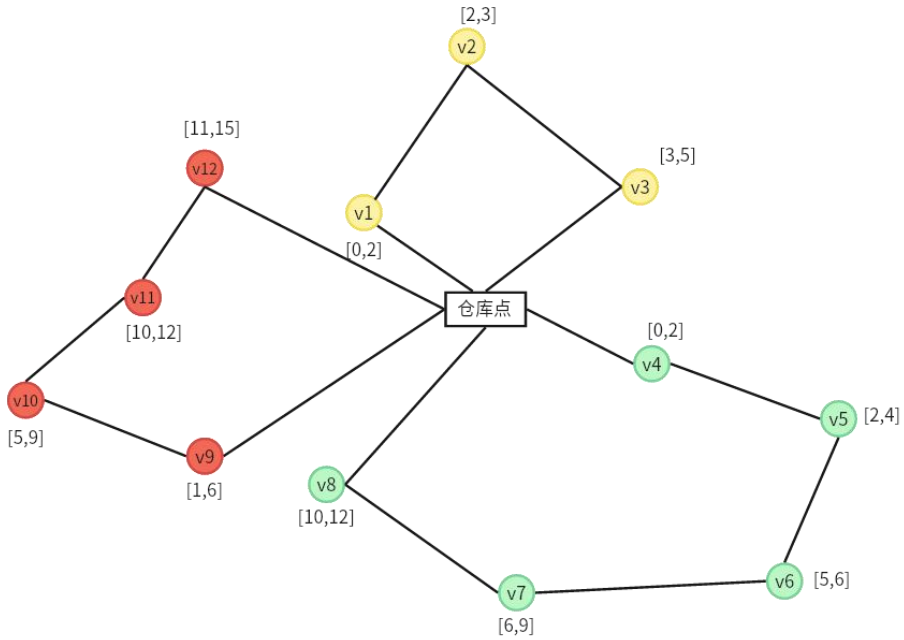

图 3.2 带有时间窗约束和容量约束的车辆路径规划示意图

3.2 改进的变邻域搜索算法

变邻域搜索算法在章节 2.2 中已经做了详细的介绍，本章节在解决 CVRPTW 问题上延续的第二章的变邻域搜索算法，并在其基础上更新了初始解的生成策略和在局部搜索中增加了一个交叉邻域，优化之后的变邻域搜索中有 3 个邻域，当邻域结构增加则算法的搜索空间就会大大增加，从而算法的搜索能力就会更强。如图 3.2 所示，在邻域 1 中找不到最优解则跳至邻域 2，若在邻域 2 中找到最优解则立刻返回邻域 1，同理，在邻域 2 中找不到最优解则跳至邻域 3，若在邻域 3 中找到最优解则立刻返回邻域 1，在邻域 3 中找不到最优解则局部搜索结束，返回当前最优解。

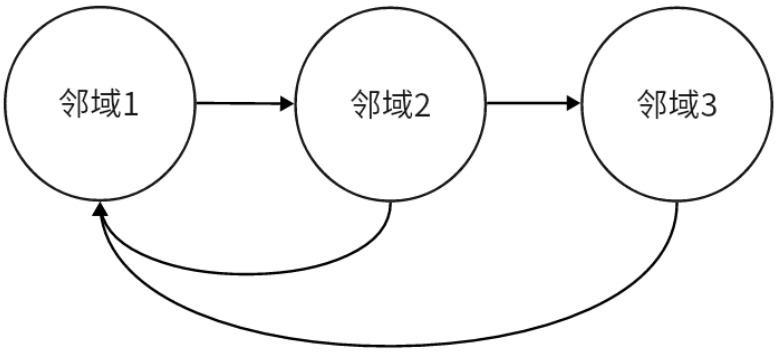

图 3.3 多邻域搜索

3.2.1 初始解的生成

生成初始解是解决 CVRPTW 问题的关键步骤之一，因为它直接影响着后续优化算法的效率和结果，以下是部分生成初始解的常用算法。

1.贪心算法：贪心算法是第二章中解决 CVRP 问题使用的算法，贪心算法简单有效但是生成的解的质量不高。

2.插入法：在本章节中，我们使用插入法生成 CVRPTW 的初始解，插入法是一种基于贪心思想的方法，通过逐步向已有路径中插入新的客户点，来构建一条完整的车辆路径。插入法的关键在于选择插入位置的策略，常见的策略包括最近插入法、最远插入法、最便利插入法等。

3.随机算法：随机法是一种基于随机选择的方法，通过随机生成车辆路径来构建解。在 CVRPTW 问题中，随机法可以随机选择起始点和终止点，逐步添加客户点来生成车辆路径。

本章节使用的算法伪代码见算法 3.1，使用了插入法来寻找初始解，保证了初始解的质量在后续的局部搜索中能更高效得找到最优解。

算法 3.1: 插入法生成初始解

输入: 路径长度矩阵c_{il}，客户点需求量d_i，时间窗口矩阵$[e_i,e_l]$，车辆数量 K
输出: 初始解 S

1：Initialize(S)　//令 S 为空路径集合
2：对于每个车辆$k \in K$，执行下列操作
3：　在 S 中插入路径P_k
4：　选择仓库点v_0，插入到路径P_k中
5：　当路径P_k中容量不超过 Q 时，找到下一个满足时间窗口限制的最近点v_i，并将其插入到P_k中
6：　重复步骤 5，直到无法找到满足限制条件的点或者P_k中容量达到 Q
7：　在路径P_k末尾插入仓库点v_0，完成路径P_k的构建
8：返回初始解 S

上述伪代码中，首先将路径集合 S 初始化为空。然后对于每个车辆 k，在 S 中插入一个空路径P_k，选择仓库点v_0，并将其插入到P_k中，接着，通过邻近矩阵寻找下一个满足时间窗口限制和容量限制的最近点，并将其插入到P_k中直到无法找到满足条件的点或者P_k容量达到 Q。最后，在P_k 的末尾插入仓库点v_0，则完成了路径集合 S 中的一条路径的构建。在算法中，我们用到了邻近矩阵作为选择下一个客户点的依据，通过距离矩阵c_{ij}，找到每个客户点与之最近的 n 个点，其中 n 的个数为客户点总数的 1/5。

在上述代码中，我们引入了邻近矩阵这一概念，在数据结构中，常常称邻近矩阵为邻接表。邻接表是一种数据结构，用于存储图的信息。对于 CVRPTW，每个客户点可以表示为一个顶点，并且在该顶点上需要存储它的需求、时间窗等信息，同时需要记录它与其他顶点（客户点）之间的邻接关系。使用邻接表来表示 CVRPTW 问题的优点在于，它可以节省存储空间和提高查询效率。邻接表可以在 O(1)的时间内查找顶点的邻居，并且可以在 O(E)的时间内遍历整个图，其中 E 是边的数量。此外，邻接表可以与许多启发式算法结合使用，采用邻接表这一数据结构对于后续的局部搜索策略有着很大的帮助。图 3.2 表示稀疏图与邻接表之间的关系

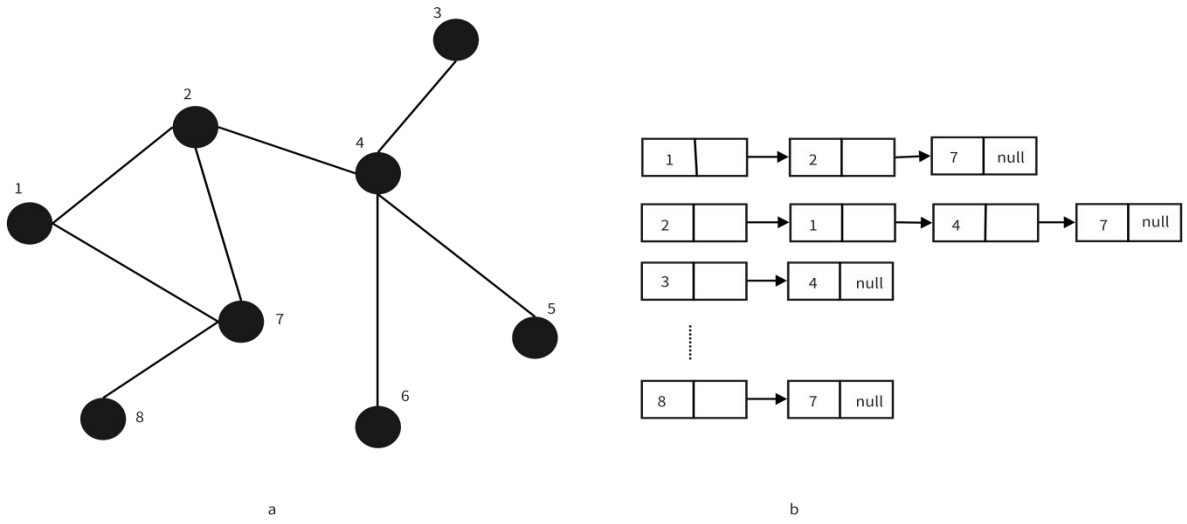

图 3.4 稀疏图及其邻接表

图 3.2 中，a 图是一个包含了 8 个节点的稀疏图，b 图是该稀疏图对应的邻接表。这个列表是用链表和数组相结合来实现。在链表实现中，每个节点包含一个指向下一个节点的指针，以及与其相邻的顶点的标识。在数组实现中，数组的每个元素都是一个链表，存储与该顶点相邻的其他顶点。邻接表的优点在于在图比较稀疏的情况下，它可以节省大量的存储空间。并且可以在 O(1)的事件内找到一个客户点的所有最邻近的点。

3.2.2 局部搜索策略

2-opt 操作在章节 2.2.3 中已经做了详细介绍，但是传统的 2-opt 策略只适用于 CVRP 问题，针对 CVRPTW 问题，因为要考虑到时间窗的问题，所以每个客户点的先后顺序必须要满足条件，因此，在没有改进过的 2-opt 操作就无法适用于 CVRPWT。相比于 2-opt 的操作，优化后的操作需要考虑每一次重新连接的点是否满足时间窗的顺序问题，若满足，则可以执行这次交换，否则撤回交换。算法 3.1 是判断是否满足时间窗的伪代码，已知客户点时间窗的下限e_i和上限l_i，客户点的服务时间s_i以及车辆行驶到客户点的时间t_i ，可以用以下算法检查客户点 i 是否满足时间窗限制。其中，第 2 行判断当前时间是否在时间窗内，如果在，则节点 i 满足时间窗限制。如果不在，则需要根据当前时间的早晚来判断节点是否可达。如果当前时间早于时间窗下限，则需要等待到时间窗开启；如果当前时间晚于时间窗上限，则节点 i 不可达，需要重新安排路线。

算法 3.2: 时间窗满足条件

输入: 客户点的时间窗$[e_i, l_i]$，客户点的服务时间s_i，车辆行驶到客户点的时间t_i

if $t_i + s_i \leq l_i \wedge t_i + s_i \geq e_i$ **then**

#当前时间在时间窗内，节点 i 满足时间窗限制

else if $t_i + s_i < e_i$ **then**

#当前时间小于时间窗下限，需要等待到时间窗开启

$t_i \rightarrow e_i$

else if $t_i + s_i > l_i$ **then**

#当前时间窗晚于时间窗上限，节点 i 不可到达，需要重新安排路线

return false

end if

其中，第 2 行判断当前时间是否在时间窗内，如果在，则节点 i 满足时间窗限制。如果不在，则需要根据当前时间的早晚来判断节点是否可达。如果当前时间早于时间窗下限，则需要等待到时间窗开启；如果当前时间晚于时间窗上限，则节点 i 不可达，需要重新安排路线。

算法 3.3 是针对 CVRPTW 更新过的 2-opt 的伪代码，其中，x 表示当前解，x_{best}表示当前最优解，n 表示顾客节点的个数，f(x)表示解 x 的总成本（即目标函数值）。

算法 3.3: CVRPTW 的 2-opt 策略

输入: 初始化解 x

输出: 2-opt 操作后的最优解x_{best}

```
while not 退出条件 do
    for i=1 to n-1 do
        for j=i+1 to n do
            反转路径 x[i]...x[j]，得到新的解 x'
            if f(x') < f(x_best) 并且满足算法 3.2 then
                令 x_best → x'
                令 x → x_best
                跳出内层循环，重新开始
            else
                恢复路径 x[i]...x[j]，继续内层循环
            end if
        end for
    end for
end while
```

针对 CVRPTW 问题，局部搜索策略采取变邻域下降策略，在本章节中优化了 2.2.3 中的策略，同时选用了 3 个邻域，即交叉邻域N_1，单点交换邻域N_2，单点移动邻域N_3。

交叉邻域N_1是在路径集合 S 中选取 2 条路径r_1、r_2，分别随机选中 2 条路径中的一个客户点进行交叉。交换之后，对新的路径r_1'，r_2'进行约束条件的判断，判断时间窗约束和运载上限是否满足要求，如果满足约束要求，则进行 2-opt 优化操作并且与之前的路径进行长度对比，否则重新选取客户点，直到所有的客户点都不满足要求。邻域N_1的数学表示如下所示

$$N_1(S) = \{S \oplus (R_i, R_j, u, v), R_i \neq R_j$$

$$u \in R_i, v \in R_j\} \tag{3.7}$$

其中R_i和R_j是路径集合中的 2 条路径，客户点 u 和客户点 v 分别从路径R_i和R_j中随机选取，将 2 个选取的客户点作为交叉的起点来生成新的路径R_i'和R_j'。

交叉邻域的示意图如图 3.1 所示。在交叉操作之前R_i路径的顺序是$n_1 \to n_2 \to n_3 \to n_4 \to n_5$，此时$R_j$路径的顺序是$n_6 \to n_7 \to n_8 \to n_9 \to n_{10} \to n_{11}$，在交叉操作之后，新的路径$R_i$'路径的顺序是$n_1 \to n_2 \to n_8 \to n_9 \to n_{10} \to n_{11}$，此时$R_j$'路径的顺序是$n_6 \to n_7 \to n_3 \to n_4 \to n_5$。

交叉邻域的伪代码如算法 3.4 所示,如代码所示，在进行交叉操作之前，首先要判断是否满足算法 3.2 的时间窗要求以及运载上限要求。

算法 3.4: 交叉邻域 Cross 操作

输入: 可行解 S，

输出: 更新解 S*

1: $S_1 \leftarrow \text{RandomChoose(S)}$

2: $S_2 \leftarrow \text{RandomChoose(S)}$ //在可行解中随机选取 2 条可行路线

3: $v_1, v_2 \leftarrow \text{RandomChoose}(S_1)$ //在可行路线S_1中随机选取客户点v_1，v_2

4: $v_3, v_4 \leftarrow \text{RandomChoose}(S_2)$ //在可行路线S_2中随机选取客户点v_3，v_4

5: //判断时间窗和运载需求条件是否满足，若满足则执行下列操作

6: //以下是交叉操作 Cross

7: v_1 →next ← v_4 ，v_4 →pre← v_1

8: v_2 →next ← v_3 ，v_3 →pre← v_2

9: return S*

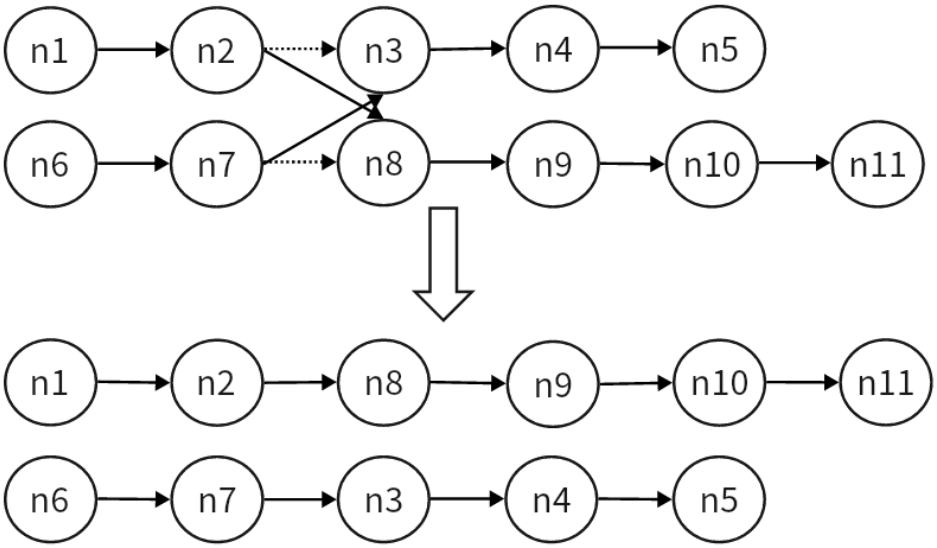

图 3.5 交叉邻域

点交换邻域N_2和点插入邻域N_2在章节 2.2.3 已经做过详细介绍，在这里就不做赘述。局部搜索策略的伪代码见算法 3.5

算法 3.5: CVRPTW 问题的变邻域下降算法

输入: 路径集合 R={$R_1, R_2, \ldots R_k$}、长度 L

输出: 改进后的可行解 R*、改进后的长度 L*

$L^* \leftarrow L$

$F_1 \leftarrow 0$，$F_2 \leftarrow 0$，$F_3 \leftarrow 0$

while $F_1 \le F_{max} \wedge F_2 \le F_{max} \wedge F_3 \le F_{max}$ **do**

    while $F_1 \le F_{max}$ **do** 点插入邻域

        $R_i \leftarrow$ Choose(R) 从可行解 R 中按序选择一条路线R_i

        $R_l \leftarrow$ Choose(R) 从可行解 R 中按序选择一条路线R_l，保证$R_i \neq R_l$

        $v_1 \leftarrow$ RandomChoose(R_i) 从路线R_i中随机选择一个客户点v_1

        if $D(R_l)+d(v_1)>Q \wedge$ 满足算法 3.2 **then**

            continue

        end if

        Insert(R_i,R_l,v_1)

        #对路线R_i和路线R_l分别进行优化后 2-opt 操作

        $R_i^*, R_l^* \leftarrow$ 2-opt(R_i,R_l)

        L* $\leftarrow$ sum(R^*)

        #当更新过的总路径小于原路径后，更新原路径里的解

        if L* < L **then**

            $F_1 \leftarrow 0,\ R_i \leftarrow R_i^*,\ R_l \leftarrow R_l^*, L \leftarrow L^*$

        else

            $F_1 \leftarrow F_1+1$

        end if

    end while 退出点插入邻域

    $F_2 \leftarrow 0$

    while $F_2 \le F_{max}$ **do** 点交换邻域

        $R_i \leftarrow$ Choose(R)

        $R_l \leftarrow$ Choose(R) 保证$R_i \neq R_l$

        $v_1 \leftarrow$ RandomChoose(R_i) 从路线R_i中随机选择一个客户点v_1

        $v_2 \leftarrow$ RandomChoose(R_l) 从路线R_l中随机选择一个客户点v_2

        if D(R_i)-d(v_1)+d(v_2)>Q$\wedge$

            D(R_l)-d(v_2)+d(v_1)$\wedge$ 满足算法 3.2 **then**

                continue

        end if

        $Swap(v_1,v_2)$ 交换v_1和v_2的位置

        #对路线R_i和路线R_l分别进行优化后 2-opt 操作

        $R_i^*, R_l^* \leftarrow$ 2-opt(R_i,R_l)

        L* $\leftarrow$ sum(R^*)

        #当更新过的总路径小于原路径后，更新原路径里的解

        if L* < L **then**

            L $\leftarrow$ L*, R $\leftarrow$ R*, $F_2 \leftarrow 0,\ F_1 \leftarrow 0$

            break

        else

            $F_2 \leftarrow F_2+1$

        end if

    end while 退出点交换邻域

```
44:     while F3 ≤ Fmax do 点交叉邻域
45:         Ri ← Choose(R)
46:         Rl ← Choose(R) 保证Ri ≠ Rl
47:         v1 ← RandomChoose(Ri) 从路线Ri中随机选择一个客户点v1
48:         v2 ← RandomChoose(Rl) 从路线Rl中随机选择一个客户点v2
49:         if D( Ri)-d(v1)+d(v2)>Q∧
50:             D( Rl)-d(v2)+d(v1)∧ 满足算法 3.2 then
51:                 continue
52:         end if
53:         Cross(Ri, Rl, v1, v2) 在v1和v2的位置进行 2 条路径的交叉
54:         #对路线Ri和路线Rl分别进行优化后 2-opt 操作
55:         Ri*,Rl* ← 2-opt(Ri,Rl)
56:         L* ← sum(R*)
57:         #当更新过的总路径小于原路径后，更新原路径里的解
58:         if L* < L then
59:             L ← L*, R ← R*, F2 ← 0, F1 ← 0
60:             break
61:         else
62:             F3 ← F3+1
63:         end if
64:     end while 退出交叉邻域
65: end while
66: return R*,L*
```

由上述伪代码可知，邻域搜索顺序是先搜索点插入邻域，若在交换邻域中找到最优解则返回点插入邻域，否则跳至点交换邻域。若在交换邻域中找到最优解则返回点插入邻域，否则跳至交叉邻域，若在交叉邻域中找到最优解则返回点插入邻域，否则输出当前最优解。

3.2.3 扰动策略

在经过变邻域搜索之后，为了防止陷入局部最优解，通常采取扰动来使 CVRPTW 问题达到全局最优解，扰动算子(R_i, v, u)代表从路径集合 S 中随机选中一条路径R_i，在路径R_i中随机选取 2 个客户点 u 和 v，在满足算法 3.1 的条件下，交换这两个客户点的位置，以尝试寻找更优的解决方案。

3.2.4 算法实现

解决 CVRPTW 问题的算法实现是选用了模拟退火的框架结合变邻域下降的策略，以及概率接受解的策略。与 CVRP 问题相比较，解决 CVRPTW 问题的算法在变邻域下降操作上以及优化 2-opt 的操作上与 CVRP 算法有所差异，算法框架如第二章节中的 2.4 所示。

3.3 实验结果与对比

为了测试本章节提出的算法的性能，我们采用了与其他算法进行对比实验的方法。我们选择了目前最优的求解器 LKH-3 所使用的算法作为对比算法。在 TSPLIB 网站上，我们选取了 40 组不同的测试用例，并在相同的实验环境下，对每个测试用例进行了 10 次运算。我们从最优解、平均解和平均时间三个维度来对比实验结果。算例的名称命名为 X.n.k，其中 X 为算例的编号，n 为客户点的数量，k 为车辆数。例如算例 C101.25.3,代表算例编号为 C101 的算例，其中客户数量为 25，车辆数为 3。与 CVRP 不同，在下列算例中，不同编号的算例对应的运载上限 Q 也是不同的，其中编号在 C101-C109 间的算例所对应的运载上限为 200，编号在 C201-C208 之间的算例所对应的运载上限为 700，编号在 R101-R112 间的算例对应的运载上限为 200。

本文的对比算法 LKH 算法是用 C 语言编写，变邻域搜索算法也是用 C 语言编写，为了控制变量，都是在同一电脑上进行计算。表 3.1 是客户点为 25 个的算例，表 3.2 是客户点为 50 个的算例。

表 3.1 客户点为 25 的算例的计算结果与对比

算例	LKH 算法			随机变邻域算法			opt
	f_{best}	f_{avg}	t_{avg}	f_{best}	f_{avg}	t_{avg}	
C101.25.3	**1913**	**1913**	2	**1913**	**1913**	**1**	1913
C102.25.3	**1903**	**1903**	2	**1903**	**1903**	**1**	1903
C103.25.3	**1903**	**1903**	2	**1903**	**1903**	**1**	1903
C104.25.3	**1869**	**1869**	2	**1869**	1900	**1**	1869
C105.25.3	**1913**	**1913**	**1**	**1913**	**1913**	1	1913
C106.25.3	**1913**	**1913**	2	**1913**	**1913**	**1**	1913
C107.25.3	**1913**	**1913**	2	**1913**	**1913**	**1**	1913
C108.25.3	**1913**	**1913**	2	**1913**	1927	2	1913
C109.25.3	**2147**	**2147**	**2**	**2147**	**2147**	3	2147
C201.25.3	**2147**	**2147**	**2**	**2147**	**2147**	3	2147
C202.25.3	**2147**	**2147**	2	**2147**	2166	2	2147
C203.25.3	**2131**	**2131**	2	**2131**	**2131**	1	2131
C204.25.1	**2147**	**2147**	5	**2147**	**2147**	2	2147
C205.25.3	**2147**	**2147**	5	**2147**	**2147**	2	2147
C206.25.3	**2145**	**2145**	4	**2145**	**2145**	4	2145
C207.25.3	**2145**	**2145**	4	**2145**	**2145**	2	2145
C208.25.3	**6171**	**6171**	5	6220	6456	2	6171

R101.25.8	**5471**	**5471**	5	**5471**	5492	2	5474
R102.25.7	**4546**	**4546**	4	**4546**	4568	2	4546
R103.25.5	**4169**	**4169**	4	**4169**	4185	3	4169
R104.25.4	**5305**	**5305**	5	**5305**	5322	2	5305

表 3.2 客户点为 50 的算例的计算结果与对比

算例	LKH 算法			随机变邻域算法			opt
	f_{best}	f_{avg}	t_{avg}	f_{best}	f_{avg}	t_{avg}	
C101.50.5	**3624**	**3624**	**3**	**3624**	**3624**	**3**	3624
C102.50.5	**3614**	**3614**	**3**	**3614**	**3614**	**3**	3614
C103.50.5	**3614**	**3614**	**3**	**3614**	**3614**	**2**	3614
C104.50.5	**3580**	**3580**	3	**3580**	3605	**2**	3580
C105.50.5	**3624**	**3624**	4	**3624**	**3624**	**2**	3624
C106.50.5	**3624**	**3624**	**2**	**3624**	**3624**	3	3624
C107.50.5	**3624**	**3624**	5	**3624**	**3624**	**3**	3624
C108.50.5	**3624**	**3624**	3	**3624**	**3642**	**2**	3624
C109.50.5	**3624**	**3624**	3	**3624**	**3624**	**2**	3624
C201.50.3	**3602**	**3602**	3	**3602**	**3602**	**2**	3602
C202.50.3	**3602**	**3602**	**3**	**3602**	**3602**	**3**	3602
C203.50.3	**3598**	**3598**	**3**	**3598**	3614	**3**	3598
C204.50.3	**3501**	**3501**	4	**3501**	3522	3	3501
C205.50.3	**3598**	**3598**	**3**	**3598**	3612	**3**	3598
C206.50.3	**3598**	**3598**	**2**	**3598**	**3598**	3	3598
C207.50.3	**3596**	**3596**	**3**	**3596**	**3596**	4	3596
C208.50.2	**3505**	**3505**	4	**3505**	**3505**	**3**	3505
R101.50.12	**10440**	**10440**	5	10489	10503	**2**	10440
R102.50.11	**9090**	**9090**	5	**9090**	9123	**4**	9090
R103.50.9	**7729**	**7729**	5	**7729**	7729	**3**	7729
R104.50.6	**6254**	**6254**	**4**	**6254**	6296	**4**	6254

从上述两表中可以看出，本章节提出的变邻域搜索算法和 LKH 算法都能以较高的效率解决 CVRPTW 问题，对于小算例 CVRPTW 问题，LKH 算法在计算时间上也和本章节提出的算法接近，都具有高效性。不过对于大算例，LKH 算法得益于出色的邻域结构可以有更好的表现。

3.4 本章小结

本章节中详细介绍了 CVRP 问题的变种问题———带有时间窗和容量约束的车辆路径规

划（CVRPTW），采用了变邻域搜索算法解决了 CVRPTW 问题。在该算法中，考虑到每个客户点都是时间窗的约束，所以对传统的 2-opt 局部搜索操作进行了改进，使得 2-opt 能够使用到 CVRPTW 上。在邻域结构上，采取了 3 个邻域分别为交换邻域，点插入邻域和交叉邻域，扰动策略上采取了扰动算子(R_i, v, u)，即在满足时间穿约束的条件下交换路径R_i的客户点 u 和客户点 v，从而能够跳出局部最优解。在实验结果上，我们采取了 40 组不同的算例，这些算例的客户点在 25 和 50 之间，并且将计算结果与经典的 LKH 算法做了比较，从比较结果我们可以看出，在 CVRPTW 问题上，我们提出的变邻域搜索算法和 LKH 算法都有着较高的计算效率和计算精度，当然，在大算例问题上，LKH 算法依靠其强大的邻域结构和优化策略能够有更好的结果。因此，在实际应用中，我们应该根据具体问题的规模和复杂度选择适当的算法。

第四章 带有距离约束和容量约束的车辆路径规划的算法研究

本小节将具体介绍 CVRP 的另一个变种问题带有距离限制的多车辆路径规划（Distance Constrained Capacitated Vehicle Routing Problem），将从问题描述、研究背景和现状以及算法描述详细介绍这一问题。

4.1 带有距离约束和容量约束的车辆路径规划

在本章节中，将详细介绍 CVRP 问题的变种问题——带有距离约束和容量约束的车辆路径规划问题（DCCVRP），将从问题描述和研究背景进行详细阐述。

4.1.1 研究背景

带有距离限制的多车辆路径规划问题（DCCVRP）目前研究广泛，主要是在许多实际应用中，如物流、配送、出租车服务等，车辆的行驶距离是一个重要的限制因素。因此，DCCVRP 问题的研究变得越来越重要。DCCVRP 问题在实际应用中有着广泛的应用，例如：1.物流配送服务，许多公司需要为客户提供定期配送服务，而配送车辆的行驶距离是一个重要的限制因素。2.出租车服务，出租车服务需要设计最优路径方案，以最小化行驶距离和时间，同时满足乘客需求。3.医疗服务，医疗服务需要为患者提供最优路径方案，以最小化车辆数量和行驶距离，同时满足医疗需求。

DCCVRP 问题的研究成果可以为实际应用提供有效的决策支持。近年来，随着启发式算法和优化技术的发展，对 DCCVRP 问题的研究也越来越深入。通过对 DCCVRP 问题的研究，可以进一步优化车辆路径规划和物流配送等领域的服务质量和效率。目前研究 DCCVRP 问题的文章不多，主要包括 Li[76]在 1992 年提出的 DCCVRP 问题的定义，Nagy 和 Salhi[77]在 2005 年提出的新的启发式算法，Cheang[78]等人在 2012 年提出后入先出算法（last-in-first-out）的以及 Almoustafa[79]在 2013 年提出的精确算法。

目前，解决 DCCVRP 问题的算法主要是以下几种：精确算法，启发式算法，元启发式算法。精确算法能保证得到最优解，但是由于 DCCVRP 问题是 NP-hard 性质，精确算法的事件复杂度相对非常高，目前针对该问题的精确算法有分支定界算法，整数线性规划等。精确算法只适用于规模小的算例，所以我们在这里不予讨论。启发式算法是求解 NP-hard 问题常用的算法，Ruiz[80]等人在 2019 年发表的论文中提出了一种基于改进的遗传算法的 DCCVRP 求解

方法。该论文提出了一种基于改进的蚁群算法的 DCCVRP 求解方法，主要针对 DCCVRP 问题的复杂性和求解效率进行了优，Auliani[81]等人在 2021 年发表的论文中提出了一种基于基因优化算法的 DCCVRP 求解方法。该方法主要针对 DCCVRP 问题的复杂性和求解效率进行了优化。元启发式算法是一种将不同启发式算法组合起来以提高求解效果的方法。常见的元启发式算法包括遗传算法与模拟退火算法的组合、蚁群算法与遗传算法的组合等。元启发式算法能够在不同的启发式算法之间寻找最佳平衡，从而得到更好的解决方案。

近年来，针对 DCCVRP 问题的研究一直在持续。例如，2022 年，Gupta[82]等人提出了一种基于深度增强学习的解决方法，该方法通过训练一个深度增强学习网络来学习车辆路线规划策略。能够在较短的时间内快速求解问题，并且在实验中得到了较好的解决效果。同时，一些传统的启发式算法，如遗传算法和蚁群算法等，也在不断改进和优化，以提高求解效果和求解速度。

4.1.2 问题描述

带有距离限制的多车辆路径规划（DCCVRP）是一种组合优化问题，该问题旨在设计车辆路径以及最小化车辆的行驶长度，同时要求车辆满足客户点的需求以及满足每辆车的距离限制。DCCVRP 问题的描述如下：给定无向完全图 $G=(V,E)$，其中集合 V 是点集，$V=\{v_0,v_1,v_2,\dots v_n\}$，其中$v_0$是仓库点，$v_1,v_2,\dots v_n$是客户点，且各点之间互不重合，同时每个客户点都有一个需求量d_i，集合 E 是边集，$E=\{e_{ij},i\in V,j\in V\}$，点与点之间的线段都包含在边集之中，每条线段有一个距离长度c_{ij}。在仓库点v_0有 k 辆车，每辆车的运载上限为 Q，要求每辆车从v_0出发，经过若干客户点之后返回v_0。在满足没条路径的总需求量不超过车辆运载上限 Q 并且每辆车的行驶距离不超过距离上限 D 的前提下，最小化总行驶距离。

DDCVRP 问题的数学模型描述如下

$$\min \sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=1}^{m} c_{ij}x_{ij}^{k} \tag{4.1}$$

$$\text{s.t.} \quad \sum_{k=1}^{m} y_k^i = 1,\ i=1,2,\dots,n \tag{4.2}$$

$$\sum_{i=1}^{n} d_i u_k^i \le Q, k=1,2,\dots m \tag{4.3}$$

$$\sum_{i=0}^{n}\sum_{j=0}^{n} c_{ij}x_{ij}^{k} \le D, k=1,2,\dots m \tag{4.4}$$

在上述公式中，其中c_{ij}表示客户点 i 和客户点 j 之间的距离，x_{ij}^{k}表示车辆 k 是否从客户点 i 到客户点 j，若$x_{i,j}^{k}$=1，则车辆 k 经过客户点 i 到 j，否则$x_{i,j}^{k}$=0。目标函数由等式 4.1 给出，要

求最小化形式距离。公式 4.2 要求每个客户点都被服务，其中y_k^i表示车辆 k 是否经过客户点 i，若$y_k^i = 1$ 则表示车辆 k 经过客户点 i，否则$y_k^i = 0$。公式 4.3 为容量约束，保证每条路径的总容量不超过上限 Q。公式 4.4 是距离约束，其中 D 是距离上限，保证每条路径的长度不超过 D。

4.2 随机变领域搜索算法

本章节将介绍用于解决 DCCVRP 问题的随机变邻域搜索算法，将从初始解的生成，局部搜索算法，扰动策略和算法伪代码详细介绍该算法。

4.2.1 初始解生成

DCCVRP 问题的初始解生成才用贪心算法，其伪代码见算法 4.1。该算法和用于生成 CVRP 初始解的算法类似，只是在运载需求的限制之上增加了距离限制的约束，从第一辆车开始，依次访问客户点，当车辆的容量不够或者距离达到限制后，就换下一辆车继续访问该客户点，直到所有车辆都访问结束。该初始解并不能保证最优性，需要后续算法来不断改进。

算法 4.1: DCCVRP 初始解的生成

输入: 点集合 *N*、需求量集合 *D*、车辆数量 *K*、运载能力 *Q*、距离上限 M、距离矩阵 C

输出: 初始解 *S*

```
1:demand ← 0, k ← 0，c ← 0
2:S_k ← 0
3:for i = 1 to N do
4:     demand += D[i]
5:     c += C[i][i+1]
6:     if demand ≥ Q ∨ c ≤ M then
7:         S ← S_k, k++,
8:         demand ← 0,c ← 0
9:     end if
10:end for
11:return S
```

4.2.2 局部搜索算法

为了解决 DCCVRP 问题，我们使用了局部搜索算法，该算法采用了变邻域搜索和 2-opt 策略的结合。由于 DCCVRP 问题不需要考虑客户点时间窗的限制，所以进行 2-opt 是无需判断时间窗需求，但是经过 2-opt 操作后，可能导致该路径的长度发生改变从而违反距离限制的

约束条件，因此，DCCVRP 问题的 2-opt 操作也有所不同。算法 3.7 为 DCCVRP 问题的 2-opt 操作的伪代码。如伪代码所示，与 CVRP 相比，在 DCCVRP 问题中，每进行一次 2-opt 操作都要重新计算当前路径的长度，判断是否超出了距离上限的约束，如果超过了距离上限 M，则取消这次操作，如果因此在 DCCVRP 问题中，2-opt 操作相比于 CVRP 问题，时间复杂度也会有所增加。

算法 4.2:DCCVRP 的 2-opt 算法

输入: 可行路线 S、路线长度 Len、距离上限 M
输出: 局部最优解 S*

```
Maxcount ← 1000
cnt_opt ←0
Len* ← 0
while cnt_opt≤ Maxcount do
    n1 ← rand( ), n2 ← rand( )
    if n1<=n2 then
        exchange(n1,n2), Len* ← 0
    else if n1 == n2 then
        continue
    end if
        Len* ← Length(S*)
    if Len*<M ∧ Len*<Len then
        S ← S*, cnt_opt ← 0
    else
        cnt_opt++
    end if
end while
return S*
```

变邻域搜索策略包括三个邻域：交换邻域，点插入邻域和交叉交换邻域。交换邻域和交叉邻域的详细介绍在章节 3.1.5 中，这里不再赘述。交叉交换邻域对车辆的利用率和平衡负载有着很好的效果。交换交叉邻域发展自交换邻域，该邻域可以扩大交换邻域的搜索空间，增强算法的搜索能力。交换交叉邻域N_3由公式 3.2 给出,考虑到 DCCVRP 问题的限制条件，公式 3.3 和 3.4 给出在交换交叉邻域搜索中的约束条件。

$$N_3(S) = \{S \oplus (R_i, R_j, x_i, x_j), R_i \neq R_j,\ x_i \in R_i, x_j \in R_j\} \tag{4.5}$$

$$D(R_i) - D(x_i) + D(x_j) \leq Q\ , D(R_j) - D(x_j) + D(x_i) \leq Q \tag{4.6}$$

$$C(R_i) - C(x_i) + C(x_j) \leq M\ , C(R_j) - C(x_j) + C(x_i) \leq M \tag{4.7}$$

其中 3.3 表示容量约束，即经过交换交叉后的路径总容量不能超过上限 Q，3.4 表示距离约束，即经过交换交叉后的路径总距离不能超过上限 M。

交换交叉邻域是根据交换邻域和交叉邻域提出的，该邻域可以更大扩展邻域搜索空间，从而可以找到各个算例的最优解。算法 3.8 是交换交叉邻域的伪代码，在可行解 S 中随机选取两条路线，分别为S_1和S_2；之后在路线S_1中随机选取 2 个不重复客户点，分别为v_1，v_2，同样，在路线S_2中也随机选取不重复的客户点v_3，v_4；若满足容量约束以及距离约束，则对两条路径进行交叉操作，不满足则约束则重新选取客户点；交叉操作结束后则进行交换操作，若满足约束条件，则执行交换操作，否则重新选取客户点，具体代码如下所示：

算法 4.3: 交换交叉邻域

输入: 可行解 S、运载上限 Q、距离上限 M
输出: 可行解 S*
1: $S_1 \leftarrow RandomChoose(S)$
2: $S_2 \leftarrow RandomChoose(S)$ //在可行解中随机选取 2 条可行路线
3: $v_1, v_2 \leftarrow \mathrm{RandomChoose}(S_1)$ //在可行路线S_1中随机选取客户点v_1，v_2
4: $v_3, v_4 \leftarrow \mathrm{RandomChoose}(S_2)$ //在可行路线S_2中随机选取客户点v_3，v_4
5: 若满足运载约束和距离约束，则进行下列操作
6: v_1 →next ← v_4，v_4 →pre ← v_1
7: v_2 →next ← v_3，v_3 →pre ← v_2 //交叉操作
8: 若满足运载约束和距离约束，则进行下列操作
9: v_1 →next← v_3 →next，v_1 → pre ← v_3 →pre
10: v_2 →next← v_4 →next，v_2 → pre ← v_4 →pre //交换操作
11: return S*

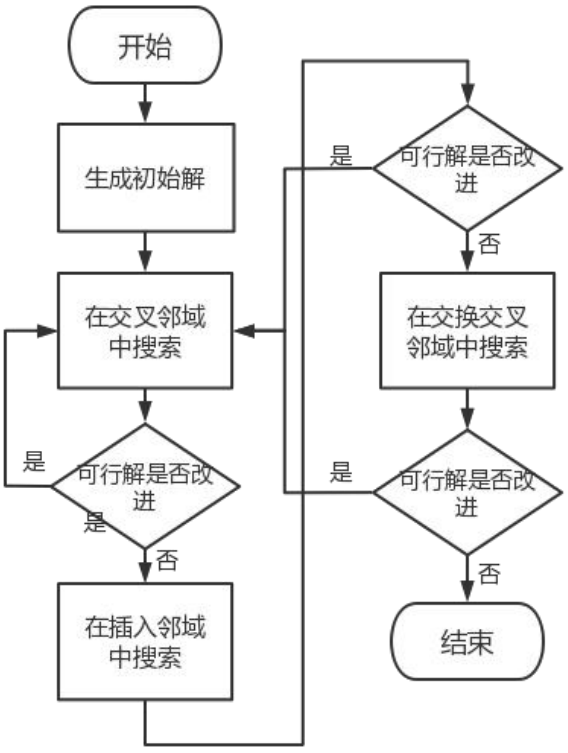

图 4.1 变邻域搜索流程图

图 4.1 中给出了 DCCVRP 问题的变邻域搜索流程图中,在流程图中可以看出，首先使用贪心算法生成初始解，然后进入交换邻域进行邻域搜索，在找到更优解后进入到点插入邻域搜索，找不到更优解则直接跳入下一邻域，将到达停止条件视为找不到更优解。在点插入邻域中进行搜索，在找到更优解之后返回交换领域中，否则进入到交换交叉邻域。在交换交叉邻域中进行邻域搜索，若找到更优解则返回至交换邻域，否则结束邻域搜索。

4.2.3 扰动策略

DCCVRP 在经过变邻域搜索之后，产生的解往往未必是全局最优解，有可能陷入了局部最优，为了使解跳出局部最优，我们往往采用扰动策略。在 DCCVRP 问题中，定义扰动算子为(R_i, v_i, v_j)，其中R_i表示在可行解中随机选中一条可行路线，v_i和v_j表示在选中的可行路线中选取 2 个客户点，并将选中的客户点之间的路线进行反转，图 3.6 表示了这一扰动操作。

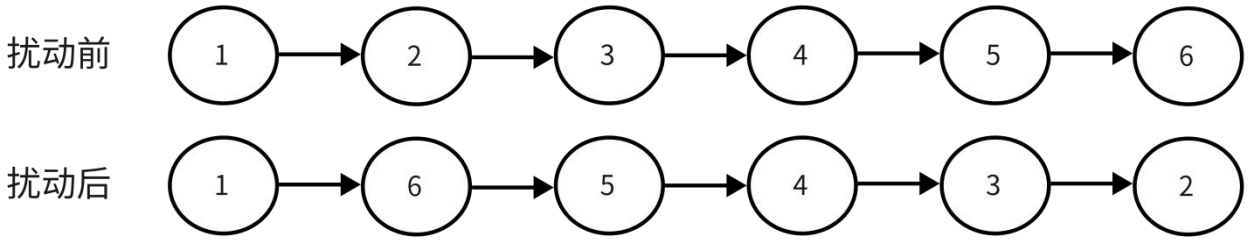

图 4.2 扰动操作

如图可以看出，选中了路径为$v_1 \to v_2 \to v_3 \to v_4 \to v_5 \to v_6$的可行路径，在路径中选中了客户点$v_2$和$v_6$，将选中的客户点之间的路径进行了反转操作，扰动后得到的路径为$v_1 \to v_6 \to v_5 \to v_4 \to v_3 \to v_2$，因为仓库点的影响，反转操作将大大改变原来路径的长度，从而可以增强算法的搜索能力，跳出局部最优解。算法 4.4 用于表示扰动操作

算法 4.4: DCCVRP 的扰动策略

输入: 可行路线 S、行驶上限 M

输出: 更新解 S*

1：$S_1 \leftarrow RandomChoose(S)$//在可行解中随机选取 1 条可行路线

2：$v_1, v_2 \leftarrow RandomChoose(S_1)$ //在可行路线S_1中随机选取客户点v_1，v_2

3：reserve(S_1) //对S_1中的部分路径反转

4：检查反转后的路径是否满足满足距离约束

5：若不满足，则撤销本次反转

6：若满足，则保留该解

7：return S*

4.2.4 算法总体伪代码

用于解决 DCCVRP 问题的算法的伪代码如算法 4.5 所示。该算法结合了随机变邻域搜索策略和模拟退火的算法框架。首先用算法 4.1 生成初始解 S，在对生成的初始解进行扰动操作，即局部路径反转操作，对扰动后的解S'进行变邻域搜索，邻域分别为交换邻域，点插入邻域和交叉交换邻域，经过变领域搜索更新过的可行解S'再与当前最优解S^*做比较，判断两个解的大小，若S'更小，则直接接受该解作为当前最优解并且进入下一次循环；若S'大于当前最优解S^*，则需要用 Metropolis 接受准则来判断是否接受当前的可行解S'作为下一次循环的初始解。算法具体步骤如下所示，其中$T_{initial}$表示初始温度，在本算法中设置为 5000，以便在迭代前期可以扩大搜索空间，跳出局部最优解。T_{end}表示截止温度，在算法中设置为 1e-8，λ 是温度的下降指数，设置为 0.95。

算法 4.5: DCCVRP 的变邻域搜索算法

输入: 完全无向图 G(V,E)、车辆数量 K、运载能力 Q、温度参数($T_{initial}$,T_{end},λ)、行驶上限 M

输出: 最优解S^*

```
S = 初始解生成      //算法 4.1
S'← S
S*← S
T ← T_initial
while T >T_end do
    for l=(1,2,...1000) do
        S' =对解 S'进行扰动操作   //局部路径反转
        S' = 对解 S'变邻域下降操作
        if f(S') < f(S) then
            S ← S'
        else
            Δf ← f(S') − f(S)
            r ← rand(0,1)   //r 是 0 到 1 之间的随机数
            if (r < e^(−Δf/T)) then
                S ← S'
            end if
        end if
        if f(S')<f(S*) then
            S* ← S'
        end if
    end for
    T ← T*q
end while
return S*
```

4.3 实验结果与对比

本章节主要用于测试对 DCCVRP 提出的变邻域搜索算法的算法性能，采用对比的方法来评判算法性能，选取的对比算法为 LKH 算法。首先在 TSPLIB 网站上下载一组 DCCVRP 问题的经典算例，因为 DCCVRP 问题研究稀缺，所以算例数不多，在该实验中，我们选取 8 个具有典型性的算例来进行计算。在相同的实验环境下进行算法计算，对每个算例进行 10 次计算，分别从最优解，平均解和平均计算时间三个维度进行对比。算例的名称命名为 DnC-nK，其中 D 代表是 DCCVRP 算例，nC 表示客户点和仓库点的数量，其中仓库点恒为一，nK 表示车辆数。算例 D022-04 代表是有 21 个客户点和 1 个仓库点，4 辆运输车辆的 DCCVRP 算例。其中 *BKS* 表示为最优解，平均计算时间的单位为秒（s），sol 表示算法计算该算例的计算结果，算例的计算结果在表格 4.1 中展示。

表格 4.1　DCCVRP 的计算结果与对比

ID	运载上限	*BKS*	LKH		VNS	
			sol	time(s)	sol	time(s)
D022-4	6000	375	*BKS*	7	*BKS*	**4**
D023-3	4500	652	*BKS*	12	*BKS*	**4**
D030-3	4500	534	*BKS*	20	*BKS*	**3**
D033-4	8000	942	*BKS*	356	*BKS*	**7**
D051-6	160	548	*BKS*	70	*BKS*	**12**
D076-11	160	905	*BKS*	417	*BKS*	**13**
D101-9	200	856	*BKS*	927	876	**16**
D101-11	200	865	*BKS*	1046	889	**13**

从上表的计算结果可以看出，在客户点是小于 100 的算例中，我们提出的算法能在更短的事件内找到算例的最优解，当算例数大于 100 时，提出的算法在级短的时间内找到非常接近最优解的可行解，LKH 算法则可以在大量计算下找到算例的最优解。

4.4 本章小结

本章节主要介绍了容量约束的车辆路径规划问题的变种问题——带有距离约束和容量约束的车辆路径规划（DCCVRP），本研究采用了随机变邻域搜索算法（Randomized Variable Neighborhood Search）解决了 DCCVRP 问题，并将其与 LKH 算法进行比较。实验结果表明，本章节提出的算法在效率上优于 LKH 算法。

随机变邻域搜索算法是一种基于贪心和随机化思想的启发式算法，它通过在每个解的邻

域中进行搜索，逐步优化解。在本章节中，我们将其应用于 DCCVRP 问题。具体地，我们设计了多种邻域结构，交换邻域，点插入邻域和交叉交换邻域。并通过的方式选择邻域进行搜索。同时，我们采用了一些加速技巧，如模拟退火和局部最优解跳出策略，来提高算法的效率。

我们在多个算例上进行了实验，并将实验结果与 LKH 算法进行了比较。实验结果表明，本章节提出的算法在求解 DCCVRP 问题上具有更高的效率。在 DCCVRP 问题中，我们的算法表现出色，与 LKH 算法相比，平均解决时间减少了约 90%。尽管本章节提出的随机变邻域搜索算法在小规模算例上表现出了更高的效率，但在大规模算例上,我们发现 LKH 算法具有更好的精度。这是因为 LKH 算法采用了更复杂的邻域结构和优化策略，能够更好地克服局部最优解和搜索空间的障碍。因此，在实际应用中，我们应该根据具体问题的规模和复杂度选择适当的算法。

总之，本文提出的变邻域搜索算法在解决 DCCVRP 问题上表现出了良好的效率和性能。虽然在某些情况下 LKH 算法可能具有更好的精度，但我们仍然相信我们的算法具有很好的应用前景，并可以用于实际的路径规划问题中。

第五章 LKH 算法的加速策略研究

在本章节中，将详细介绍本文的对比算法 LKH 算法，以及对该算法进行的加速优化操作，实验表明，优化过的 LKH 算法求解效率可以更高。

5.1 LKH 算法

本章节中将介绍 LKH 算法以及解决 CVRP 问题时影响罚函数的时间操作的问题。

5.1.1 算法描述

LKH-3 (Lin-Kernighan-Helsgaun-3) 算法是一种用于求解旅行商问题 (TSP)的优化算法，是对 LKH-2 算法的改进和升级。它是由 Keld Helsgaun 提出的，目前是 TSP 问题及其变种问题中效果最好的算法之一。LKH-3 算法是一种基于局部搜索的启发式算法，它采用了候选点集合和 k-opt 启发式算法的结合，可以在较短的时间内找到非常优秀的解，同时具有较高的可扩展性和鲁棒性。

候选点集合是 LKH 算法用于客户点连接的重要参考标准，在 LKH 算法中，每一个客户点都会有它对应的候选集，在候选集中的点按照权重依次排列代表候选集中的该点成为最优解中的次序，其中权重按照 LKH 算法中的α值，α值越高，则排序越靠前，越容易被选择为最优解。因为候选集的存在，使得 LKH 算法在进行 k-opt 操作是舍弃掉了大量不满足条件的解。

k-opt 启发式算法是一种基于切割和重组操作的局部搜索策略，通过对当前解的路径进行切割和重组，生成新的解，并选择其中路径长度最短的作为下一步搜索的起点。在 k-opt 操作中，虽然对 k 值的上限没有明确规定，不过 k 的大小直接影响到 k-opt 操作的时间复杂度，具体对时间复杂度的影响为 $O(n^k)$，所以通常 k 值的选择不超过 5，该算法首先将当前路径切除 k 条互不相邻的线段，然后对线段之间的交叉点进行枚举，并尝试进行重组，生成新的解。通过枚举不同的线段和不同的切割和重组操作，可以找到一组可行的解，并选择其中路径长度最短的作为下一步搜索的起点。图 5.1 为 3-opt 的示意图，其中切断了 3 个互不相邻的线段并对其进行了重组，并对所有可能的连线进行了枚举操作。

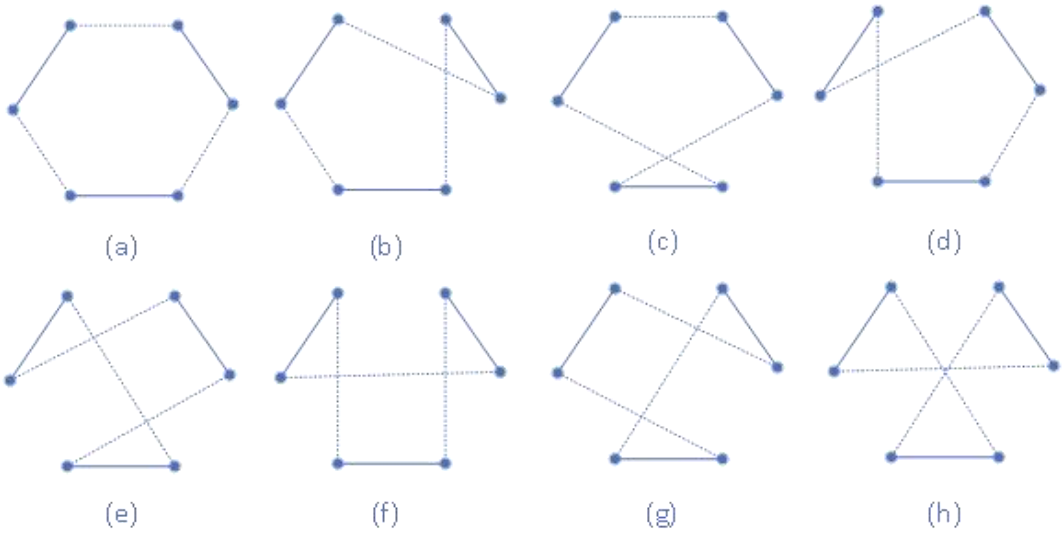

图 5.1 3-opt 操作及枚举

LKH-3 算法在解决 CVRP 问题时步骤如下

1.将 CVRP 问题转换成 TSP 问题，其中增加了 k-1 个虚拟仓库点，虚拟仓库点之间距离为无穷大，k 代表车辆的数量。

2.将图形划分为多个子集，每个子集代表一个车辆的路径。

3.初始时，每个子集只包含起点和终点，每个车辆的容量都为 0。

4.为每个客户点创建自己的候选点集合。

5.采取局部搜索策略，从 2-opt 开始搜索，若找不到更好的则转入 3-opt，一直到 5-opt 操作，从当前解开始进行局部搜索，寻找当前解的局部最优解，并将其作为下一步搜索的起点。

6.如果找到更好的解，则更新当前解，否则继续执行局部搜索策略。

7.重复步骤 5 和 6，直到找到最优解或达到停止条件为止。

在解决 CVRP 问题时，容量约束是一项重要的限制条件，需要在求解过程中进行考虑，如果超过容量限制，则需要派遣额外的车辆或拆分路径来满足需求。LKH-3 算法中运用罚函数 P 来进行容量约束处理，罚函数是一种惩罚机制，可以在目标函数中增加额外的罚项，以惩罚超过容量限制的解，从而将这些解的优先级降低。具体来说，对于每个车辆的路径，我们可以计算路径上所有客户的需求之和，如果这个值超过了车辆的容量限制，就给这个路径添加一个罚项，惩罚超出容量限制的部分。罚项的大小可以根据超出容量的程度来确定，例如可以设置为超出容量的体积或数量与容量限制的比例。在 LKH-3 算法中，罚函数的形式如公式 5.1 所示

$$P = \begin{cases} \lambda(Load_i - Q) & Load_i > Q \\ 0 & Load_i \le Q \end{cases} \tag{5.1}$$

其中$Load_i$是第 i 辆车的负载，$\max(0,Load_i - Q)$表示的是超出运载能力的部分，λ是罚函数的权重，用于调整罚项对目标函数的影响，为了方便计算和提高程序的可读性，在大多数算法

中，λ的值都设置为 1。在 LKH 算法中，λ也设置为 1。

总之，LKH-3 算法可以通过局部搜索策略来寻找 CVRP 问题的解，并在搜索过程通过罚函数操作限制了容量约束，从而得到更加可行的解。

5.1.2 LKH 算法解决 CVRP 问题时的罚函数计算

LKH-3 算法的源代码表明，在每次执行局部优化操作后，该算法会计算罚函数的总体值。相比于旅行商问题，LKH-3 算法解决 CVRP 问题需要更多的时间，因为它需要重复计算罚函数。这样做的目的是为了接受一些罚值大于 0 的解，以便跳出局部最优解并达到全局最优解。但是我们在实验中发现，在客户点低于 500 个的算例中，接受罚值大于 0 的解并没有意义。因为 LKH-3 算法具有强大的局部搜索能力，可以通过优秀的邻域结构跳出局部最优解。因此，在客户点小于 500 个的情况下，每次计算总体罚函数只会增加算法的时间成本，而对精度的影响可以忽略不计。

LKH-3 在解决 CVRP 问题时，用于计算罚函数的伪代码如下所示，其中 startroute 为可行

算法 5.1:LKH 算法用于解决 CVRP 问题的罚函数

```
输入: 可行路线 S、可行路线罚值 P、运载上限 Q
输出:  更新后的罚值 P*
T ← 0    //变量 T 用于保存增加的罚值
N ← startroute //变量 N 用于保存可行路线中的每一条路线的起始客户点
do
    currentroute ← N
    DemandSum ← 0
    DemandSum ← Calculate(N) //计算路径 N 上的总需求量
    if (DemandSum > Q)&&(T +=DemandSum-C)>P then
        startroute = currentroute
        return P
    end if
  #当变量 N 保存的不是可行路线中的第一条路线是退出循环
while(N != startroute)
return T
```

路线 S 中的第一条路径，currentroute 为可行路径 S 中的当前路径，变量 N 用于遍历可行路径 S，在 Calculate()函数中，会对变量 N 进行遍历，可以看出，每进行一次交换，LKH 算法都会重新计算一遍 CVRP 问题的罚函数从而跳出局部最优解。

5.1.3 LKH 算法的加速策略

1.限制性搜索

LKH 算法在进行局部搜索时，对该操作进行限制，具体而言，先判断此次交换是否可以满足运载需求上限，若可以满足，则执行此次局部搜索操作，否则，重新进行搜索。在该操作下，进行局部搜索之后无需进行罚函数的计算，不过该操作之适合于客户点低于 500 个的情况下，当客户点数量更高时，则需要考虑接受罚值大于 0 的解以跳出局部最优。限制性搜索的伪代码如下所示，由算法可以看出，在限制性搜索下，无需关注罚函数的大小，只有满足限制条件才可以进行搜索，因此，搜索效率可以大大提高。实验结果表明，在优化 LKH 算法局部搜索操作后，候选集中的非法解可以被筛选掉 15%-20%。

算法 5.2：优化 LKH 算法的局部搜索

输入： 可行路线 S、运载上限 Q

```
1：if charge(S) then
2：#判断 2-opt 操作是否满足运载限制,若满足则进行 2-opt，否则重新搜索
3：      2-opt(S)
4：else return
5：if charge(S) then
6：#判断 3-opt 操作是否满足运载限制,若满足则进行 3-opt，否则重新搜索
7：      3-opt(S)
8：else return
9：if charge(S) then
10：#判断 4-opt 操作是否满足运载限制,若满足则进行 4-opt，否则重新搜索
11：      4-opt(S)
12：else return
13：if charge(S) then
14：#判断 5-opt 操作是否满足运载限制,若满足则进行 5-opt，否则重新搜索
15：      5-opt(S)
16：else return
```

2.优化罚函数

下面是优化 LKH 算法解决 CVRP 问题时罚函数的优化，其中算法 5.3 用于计算罚函数值。该函数接受两个节点（一个是仓库点，另一个可能是客户点）、当前载重和车辆的运载上限，根据是否满足运载限制返回相应的罚函数值。算法 5.4 则是应用罚函数的 LKH 算法主体，它在执行 k-opt 交换时会计算并考虑罚函数值，从而在搜索过程中满足 CVRP 的约束条件。从算法中可以看出，在每进行一次 k-opt 操作后，算法都会计算当前路径的运载量以及罚函数，相比原原先 LKH 算法，每进行一次交换都会重新计算整条路径，我们优化后的算法明显有更高的效率，但相比之下，搜索的广度不及原先的 LKH 算法，可能会导致在大算例问题上无法找到最优解。

算法 5.3:LKH 解决 CVRP 问题的罚函数

输入： 点$node_1$、点$node_2$、当前载重 *current_load*、运载上限 *Q*

```
if node1==depot && node2!=depot   //如果node1为仓库点
    if current_load + node2.demand > Q
        return MAX
else if node1!=depot && node2==depot //如果node2为仓库点
    if current_load - node1.demand < 0
        return MAX
return 0
```

算法 5.4:优化算法主体

输入： 距离矩阵 distance_matrix、需求量 demands，运载上限 Q

```
#初始化解
solution = initialize(distance_matrix)
while not termination_condition_met( )   //当没有到达停止条件
    #选择一对节点进行 k-opt
    node1,node2=select_to_exchange(solution)
    #计算当前载重
    current_load = calculate_current_load(solution,node1)
    #计算罚函数
    penalty = edge_penalty(node1,node2,current_load,Q) //算法 4.3
    #计算新的路径长度
    new_cost = old_cost-distance_matrix[node1][node2]+penalty
    #如果交换后的解更优，则接受新解
    if new_cost < old_cost
        solution = update_solution(solution,node1,node2)
        old_cost = new_cost
return solution
```

5.1.4 结果对比

考虑到 LKH 算法在解决大算例 CVRP 问题时，才会有明显的计算时间，所以我们在选取计算用例时都选取客户点在 500 个以上的算例，并且从最优解和计算时间上分别对比传统 LKH 算法和优化过后的 LKH 算法，其中 time 取第一次找到最优解的时间，ptime 表示找到最优解时计算罚函数的事件，单位记为秒（s），其中 BKS 表示最优解。为确保实验环境变量，实验都是在同一台电脑上进行。计算的结果如表 5.1 所示。

表 5.1 计算结果与对比

ID	客户点	*BKS*	LKH			优化后的 LKH		
			sol	ptime(s)	time（s）	sol	ptime(s)	time(s)
Li_21	561	16212.3	*BKS*	1060	5300	*BKS*	**467**	**4750**
Li_22	601	14499.04	*BKS*	1310	6553	*BKS*	**839**	**5900**
Li_23	641	18801.13	*BKS*	1396	6980	*BKS*	**726**	**6421**
Li_24	721	21389.43	*BKS*	1730	8650	*BKS*	**1596**	**7618**
Li_25	761	16668.51	*BKS*	3414	13881	*BKS*	**2653**	**12395**
#Improve								**5**

由表 5.1 可以看出，对 LKH 算法进行了加速优化操作过后，总体的计算时间缩短了 10%-15%，其中罚函数的计算时间大大缩短，最优解的质量没有下降，因此，优化后的 LKH 算法在解决 CVRP 问题大算例时具有更好的计算效率。

5.2 本章小结

在本章节中，我们详细介绍了本文的对比算法——LKH 算法，介绍了 LKH 算法在解决 CVRP 问题时局部搜索策略和罚函数的计算，并分析了 LKH 算法计算小算例时并没有采取限制性搜索以及罚函数没有用到累加计算，针对这两个问题，我们对 LKH 算法进行了加速优化，即在不改变计算结果的前提下，提高了 LKH 算法解决 CVRP 问题时的效率。从数据中可以提现，我们对限制搜索和罚函数累加计算的改进是成功的，改进后的 LKH 算法可以在更短的时间内找到 CVRP 问题的最优解。

第六章 总结与展望

6.1 总结

本文主要针对车辆路径规划问题（Vehicle Routing Problem）做了大量的研究，其中包括研究了容量约束的车辆路径规划问题（Capacitated Vehicle Routing Problem），带有时间窗和容量约束的车辆路径规划问题（Capacitated Vehicle Routing Problem with Time Winodw）和带有距离约束和容量约束的车辆路径规划问题（Distance Constrained Capacitated Vehicle Routing Problem），对以上每个问题都提出了一个启发式算法并且成功找到了最优解。本文还改进了对比算法 LKH 算法，通过修改该算法的罚函数从而缩短了找到最优解的计算时间。本文对上述问题的研究成果和贡献如下所述：

（1）对于容量约束的车辆路径规划问题（CVRP）：CVRP 问题是车辆路径规划问题最经典的问题之一，该问题在 1959 年首次被提出。CVRP 问题在现实生活中应用非常广泛，比如在实际的物流配送中，我们要考虑每辆运输车的载重极限，并且要考虑不同社区之间的距离，以及生活垃圾的处理与回收。由于 CVRP 问题的高应用性，从古至今许多学者都为该问题做了深刻的研究，由于该问题是 NP-hard 问题，往往不能求出一个精确解，所以研究的算法也从最先的精确算法到目前的启发式算法，在目前的大量启发式算法中，LKH 算法在求解 CVRP 问题时能够有效得计算出算例的最优解，不过在计算时间方面，LKH 算法不具备快速高效性，所以，我们提出了基于邻域分解的变邻域搜索算法来解决该问题。该算法的算法框架为基础的变邻域搜索，我们引入了N_1，N_2两个邻域，分别为点插入邻域和交换邻域，并将邻域分解技术应用到算法中来提升算法的效率。再结合模拟退火算法，通过概率接受可行解的方式来跳出局部最优解。实验部分，将本文提出的算法与 LKH 算法相比较，体现了我们所提出算法的高效性。

（2）对于带有时间窗和容量约束的车辆路径规划问题（CVRPTW）：CVRPTW 问题时 CVRP 问题的变种问题之一，该问题在 1987 年首次被提出。CVRPTW 问题更接近现实应用，该问题在 CVRP 问题的基础上增加了时间窗这一限制条件，这样可以将该问题适用到外卖配送以及快递上门派送等实际应用中，相比于 CVRP 问题，CVRPTW 问题所提出的时间较晚，所以提出用于解决 CVRPTW 问题的算法也相对 CVRP 问题较少，所用到的研究算法也是从最初的精确算法到目前的启发式算法，在目前较为常见的的几种启发式算法中，部分算法可以求出 CVRPTW 问题的最优解，但是求解的效率并不是很高，所以，我们提出了基于邻域

分解的优化后的变邻域算法来解决该问题。在该算法中，我们引入了N_1，N_2，N_3三个邻域，分别为点插入邻域，交换邻域和交叉邻域。再结合适应了时间窗问题的 2-opt 优化策略和 Metropolis 接受准则，可以精准得计算出部分算例的最优解。实验部分，我们将算法与 LKH 算法相比较，结果提现了所提出算法的精准性和高效性。

（3）对于带有距离约束和容量约束的车辆路径规划问题（DCCVRP）：DCCVRP 问题是 CVRP 问题的一个变种问题，该问题在 CVRP 问题的基础上增加了距离上限这一约束条件，使得该问题更加贴近于现实应用。在现实生活中，车辆往往需要考虑油耗或者新能源车的电量问题，并不能一直行驶下去，所以正确考虑到每辆车的行驶距离上限是非常重要的。DCCVRP 问题在 1992 年被首次提出，由于其提出晚并且和 CVRP 问题高度重合，所以并没有多少研究算法用于 DCCVRP 问题，目前用于解决 DCCVRP 问题的算法多为启发式算法，该类算法解决 DCCVRP 问题上可以有效得求出问题的最优解，但是求解的效率并不是很高，所以，我们提出了基于邻域分解的优化后的变邻域搜索算法来解决该问题。在算法中，我们引入了N_1，N_2，N_3三个邻域，分别为交换邻域，点插入邻域和交换交叉邻域。结合适应了 DCCVRP 问题的 2-opt 操作作为局部搜索，算法总体框架采用了模拟退火算法框架，可以快速准确找到算例问题的最优解。从实验结果可以看出，我们提出的算法具有较高的效率。

（4）对于 LKH 算法：我们首先介绍了 LKH 算法用于解决 CVRP 的机制，然后通过计算罚函数的代码分析了该算法的不足之处——在计算部分算例时耗时过大，我们在 LKH 源代码基础上对该算法进行了优化加速操作。从实验结果可以看出，我们的优化能用更短的计算时间找到最优解。

6.2 展望

基于本文的研究，给出了几个未来可能的研究方向和应用场景。

（1）对于带有运载上限的多车辆路径规划问题（CVRP），在应用场景方面可以将 CVRP 问题用于太阳能板光伏布线的应用中，目前新能源势头正盛，用光伏发电可以大大节省煤炭燃料的开支，而在布线环节，布线成本也是一笔很大的开支，所以将 CVRP 问题运用到光伏布线中也具有很大的现实意义。对于本文所提出的算法对大算例的计算效率不高，对 200 个客户点以上的算例不能找到最优解的问题，可以在变邻域下降（VND）操作中扩大邻域的结构从而达到扩大搜索空间的效果。对于 CVRP 问题，由于其现实应用背景，该问题的研究空间非常大以及研究意义还是非常重要的，在运输链中改善一个小小的环节对于整体的成本都是巨大的，因此该问题值得我们更深刻得研究和讨论。

（2）对于带有时间窗的多车辆路径规划问题（CVRPTW），在应用场景方面可以将CVRPTW 问题运用到机器分割中，可以对一块固定长度的布料进行不等长的分割，机器每分割一次需要停止运行一段时间，可以考虑分割的顺序来最短化机器的分割时间，CVRPTW 问题对于该应用有很大的现实意义。考虑到 CVRPTW 问题的特殊性，可以在其基础上再增加约束条件，比如距离限制和车辆折损等等，这样该问题就更加接近现实问题，值得更深刻得讨论。本文提出的算法在解决大规模算例是结果不佳，考虑到局部搜索简单，可以通过在 2-opt 操作的基础上增加 3-opt 操作来增强算法的搜索能力。目前，人工智能产业发展迅猛，也有越来越多的学者用强化学习等操作来解决 CVRP 问题，同样我们也可以将强化学习等操作用在 CVRPTW 问题上，结合启发式算法也许可以发现更多算例的最优解。

（3）对于带有距离限制的多车辆路径规划问题（DCCVRP），该问题由于提出时间晚，所以研究的学者以及各种文章并不是很多，考虑到 DCCVRP 问题的限制条件和 CVRP 问题的限制条件比较类似，所以可以适当在 DCCVRP 问题的基础上再增加更贴合现实意义的限制条件，比如增加时间窗的约束限制以及某些路段为单行道的特殊限制，这样就能使得 DCCVRP 问题更加具有讨论意义和研究价值。本文提出的随机变邻域搜索算法在面对运载上限过大的算例时，算法的计算效率不高，我们考虑可以将深度学习和启发式算法进行一个结合，从而可以增加算法的性能。

总之，路径规划问题已经被研究得非常久远，我们可以考虑依仗现有的算法，通过改变路径规划问题的限制条件，包括增加部分限制，可以使其变成一种新的更加具有现实意义的问题。另外我们可以寄将启发式算法和强化学习相结合，这种新结合的算法对于解决车辆路径规划问题会具有良好的效果。

参考文献

[1] Dantzig, G. B., Ramser, J. H.,. The Truck Dispatching Problem. Management Science ,1959,6 (1), 80–91.

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

[3] Larsen J., Parallelization of the vehicle routing problem with time windows. Ph.D. thesis, Department of Mathematical Modelling, Technical University of Denmark, Lynghy, Denmark , 1999.

[4] Laporte G., The vehicle routing problem : An overview of exact and approximate algorithms, Europe Journal of Operational Research, 1992,59(3), 345–358.

[5] Bodin B, Golden A. Assad and M. Ball, "Routing and Scheduling of Vehicles and Crews: TheState-of-the-Art, Computers & Operations Research 10, 62-212.

[6] G. Laporte, Y. Nobert, S. Taillefer A branch and bound algorithm for the asymmetrical distance-constrained vehicle routing problem Mathematical Modelling, 1987 ,9 (12) , pp. 857-868.

[7] Toth, P., Vigo, D.. The Granular Tabu Search and Its Application to the Vehicle-Routing Problem. INFORMS Journal on Computing 2003, 15, 333–346.

[8] Golden, B., Wasil, E., Kelly, J., Chao, I. The Impact of Metaheuristics on Solving the Vehicle Routing Problem: Algorithms, Problem Sets, and Computational Results. In: Crainic, T. G., Laporte, G. (Eds.), Fleet management and logistics. Springer US, 1998, Boston, MA, pp. 33–56.

[9] Li, F., Golden, B., Wasil, E.. Very large-scale vehicle routing: new test problems, algorithms, and results. Computers & Operations Research , 2005, 32 (5), 1165–1179.

[10] Mester, D., Br ¨ aysy, O. Active guided evolution strategies for large-scale vehicle routing problems with time windows. Computers & Operations Research, 2005, 32 (6), 1593–1614.

[11] Prins, C.. A GRASP × evolutionary local search hybrid for the vehicle routing problem. In: Pereira, F. B., Tavares, J. (Eds.), Bio-inspired Algorithms for the Vehicle Routing Problem. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, 820 pp. 35–53.

[12] Tarantilis, C., Kiranoudis, C. BoneRoute: An Adaptive Memory-Based Method for Effffective Fleet Management. Annals of Operations Research, 2009, 115 (1-4), 227–241.

[13] 米恬怡, 唐秋华, 成丽新, 张利平. 融合强化学习与变邻域搜索的柔性作业车间调度研究[J/OL]. 工业工程与管理:1-9[2023-05-15].

[14] Chen, P., Huang, H.-K., Dong, X.-Y. Iterated variable neighborhood descent algorithm for the capacitated vehicle routing problem. Expert Systems with Applications, 2010, 37 (2), 1620–1627.

[15] Fleszar, K., Osman, I. H., Hindi, K. S. A variable neighbourhood search algorithm for the open vehicle routing problem. European Journal of Operational Research, 2009, 195 (3), 803–809.

[16] Imran, A., Salhi, S., Wassan, N. A. A variable neighborhood-based heuristic for the heterogeneous flfleet vehicle routing problem. European Journal of Oper-ational Research, 2009, 197 (2), 509–518.

[17] Mester, D., Br ¨ aysy, O. Active-guided evolution strategies for large-scale capacitated vehicle routing problems. Computers & Operations Research, 2007, 34 (10),2964–2975.

[18] Polacek, M., Hartl, R. F., Doerner, K., Reimann, M. A Variable Neighborhood Search for the Multi Depot Vehicle Routing Problem with Time Windows. Journal of Heuristics, 2004, 10 (6), 613–627.

[19] Polat, O., Kalayci, C. B., Kulak, O., G ¨ unther, H.-O. A perturbation based variable neighborhood search heuristic for solving the Vehicle Routing Problem with Simultaneous Pickup and Delivery with Time Limit. European Journal of Operational Research, 2015, 242 (2), 369–382.

[20] Hassannayebi, E., Hessameddin, S. Computers & Operations Research Variable and adaptive neighbourhood search algorithms for rail rapid transit timetabling problem. Computers & Operations Research, 2016.

[21] Sam`a, M., Ariano, A. D., Corman, F., Pacciarelli, D. Computers & Operations Research A variable neighbourhood search for fast train scheduling and routing during disturbed railway traf fifi c situations. Computers & Operations Research, 2016.

[22] Todosijevi´c, R., Hanafifi, S., Uro ˇ sevi´c, D., Jarboui, B., Gendron, B., 2016. A general variable neighborhood search for the swap-body vehicle routing problem. Computers & Operations Research.

[23] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takac, “Reinforcement learning for solving the vehicle routing problem,” in Proc. Adv. Neural Inf. Process. Syst., 2018, pp. 9839–9849.

[24] 高海昌, 冯博琴, 朱利. 智能优化算法求解TSP问题[J]. 控制与决策, 2006(03):241-247+252.

[25] 苗夺谦, 胡桂荣. 知识约简的一种启发式算法[J]. 计算机研究与发展,1999(06):42-45.

[26] J. J. Q. Yu, W. Yu, and J. Gu, “Online vehicle routing with neural combinatorial optimization and deep reinforcement learning,” IEEE Trans. Intell. Transp. Syst., vol. 20, no. 10, pp. 3806–3817, Oct. 2019.

[27] Z. Li, Q. Chen, and V. Koltun, “Combinatorial optimization with graph convolutional networks and guided tree search,” in Proc. Adv. Neural Inf. Process. Syst., 2018, pp. 539–548.

[28] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. 2017. Learning combinatorial optimization algorithms over graphs. In Advances in Neural Information Processing Systems. 6348–6358.

[29] Hanjun Dai, Bo Dai, and Le Song. Discriminative embeddings of latent variable models for structured data. In International Conference on Machine Learning. 2016, 2702–2711.

[30] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 .

[31] Chaitanya K Joshi, Thomas Laurent, and Xavier Bresson. An efficient graph convolutional network technique for the travelling salesman problem. 2019, arXiv preprint arXiv:1906.01227.

[32] Alex Nowak, Soledad Villar, Afonso S Bandeira, and Joan Bruna. A note on learning algorithms for quadratic assignment with graph neural networks. 2017, Stat 1050 , 22.

[33] Michel Deudon, Pierre Cournut, Alexandre Lacoste, Yossiri Adulyasak, and LouisMartin Rousseau. Learning heuristics for the tsp by policy gradient. In International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research. Springer,2018, 170–181.

[34] Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning.2016, arXiv preprint arXiv:1611.09940 .

[35] Sepp Hochreiter and Jürgen Schmidhuber.. Long short-term memory. Neural computation ,1997, 9, 8, 1735–1780.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in Neural Information Processing Systems.2017, 5998–6008.

[37] Laporte, G., & Nobert, Y. Exact algorithms for the vehicle routing problem. In S. Martello, G. Laporte, M. Minoux, & C. Ribeiro (Eds.), Surveys in combinatorial optimization. In North-Holland Mathematics Studies, 1987, 132 (pp. 147–184). North-Holland.

[38] Baldacci, R., Toth, P., & Vigo, D. Recent advances in vehicle routing exact algorithms.2007, 4OR, 5(4), 269–298.

[39] Baldacci, R., Toth, P., & Vigo, D.. Exact algorithms for routing problems under vehicle capacity constraints. Annals of Operations Research, 2010, 175(1), 213–245.

[40] Baldacci, R., Vigo, D., & Toth, P. Exact solution of the capacitated vehicle routing problem. Wiley Encyclopedia of Operations Research and Management Science, 2010.

[41] J.LYSGAARD, A.LETCHFORD, R.EGLESE, A new branch-and-cut algorithm for capacitated vehicle routing problems. Math. Program, 2004, 100,423-445.

[42] R.BALDACCI, A. MINGOZZI, A unifified exact method for solving different classes of vehicle routing problems. Math.Program, 2009, 120,347-380.

[43] Gözde Kizilates, Fidan Nuriyeva, On the Nearest Neighbor Algorithms for the Traveling Salesman Problem.Advances in Intelligent Systems and Computing book series (AISC,volume 225).
[44] G.CLARKE, J.W.WRIGHT, Scheduling of vehicles form a central depot to a number of delivery points.Operations Research, 1964, 12(4) ,568- 581.
[45] Subramanian, A., Uchoa, E., & Ochi, L. S. A hybrid algorithm for a class of vehicle routing problems. Computers & Operations Research, 2013, 40(10), 2519–2531.
[46] Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. A unifified solution framewok for multi-attribute vehicle routing problems. European Journal of Operational Research, 2014, 234(3), 658–673.
[47] Hansen, P., Mladenovi´c, N., Moreno P´erez, J. A. Variable neighbourhood search: methods and applications. Annals of Operations Research , 2010, 175 (1), 367–407.
[48] K. Helsgaun. An effective implementation of the lin‐kernighan traveling salesman heuristic. European Journal of Operational Research, 2000, 126(1):106–130.
[49] Elshaer, R., & Awad, H. A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants. Computers & Industrial Engineering, 2020, 140, 106242.
[50] Duarte A, Mladenovic N, Sánchez-Oro J, et al. Variable neighborhood descent[J]. 2018.
[51] Vinicius R. Maximo, Maria C.V. Nascimento. A hybrid adaptive iterated local search with diversifif- ication control to the capacitated vehicle routing problem. European Journal of Operational Research, 2021, 294(3),1108-1119.
[52] Jepsen, Mads et al. “A non-robust Branch-and-Cut-and-Price algorithm for the Vehicle Routing Problem with Time Windows.” (2006).
[53] Savelsbergh, Martin W. P. and M Marc Sol. “The General Pickup and Delivery Problem.” Transp. Sci.1995, 29 : 17-29.
[54] Brown, D.M., & Frank, M.E. Liquid Phase Methanol LaPorte PDU: Modification, operation, and support studies. Quarterly technical progress report No. 4, 1 April--30 June.
[55] Glover, Fred W. and Manuel Laguna. “Tabu Search.” Handbook of Heuristics, 1997.
[56] Goldberg, D.E., & Holland, J.H. Genetic Algorithms and Machine Learning. Machine Learning, 1998, 3, 95-99.
[57] Dorigo, M., & Gambardella, L.M. Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. Evol. Comput.,1997, 1, 53-66.
[58] Eberhart, R.C., & Kennedy, J. A new optimizer using particle swarm theory. MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995, 39-43.
[59] Chiang W.C. and Russel R.A., Simulated annealing metaheuristic for the vehicle routing problem with time windows, Annals of Operations Research, 1996, 63, 3–27.
[60] Ombuki B., Ross B.J., Hanshar F., Multi-objective genetic algorithms for vehicle routing problem with time windows, App Intell, 2006, 24(1),17–30.
[61] Berger J., Barkaoui M., A parallel hybrid genetic algorithm for the vehicle routing problem with time windows, Computers and Operations Research, 2004, 31, 2037–2053.
[62] Cordeau J.F., Gendreau M., Laporte G., Potvin J.Y. and Semet, F., A guide to vehicle routing heuristics, Journal of the Operational Research Society, 2002, vol. 53(5), 512–522.
[63] Pisinger D. and Ropke S., A general heuristic for vehicle routing problems, Comput Oper Res, 2007, 34,2403– 2435.
[64] Chand P., Mishra B.S.P. and Dehuri S., A multi objective genetic algorithm for solving vehicle routing problem, Int J Info Tech Knowl Mgmt, 2010, 2,503–506.
[65] Wang C.H. and Li C.H., Optimization of an established multi-objective delivering problem by an improved hybrid algorithm, Expert Syst Appl, 2011, 38(4),4361–4367.
[66] Tavakkoli-Moghaddam R., Safaei N. and Shariat M.A., A multi-criteria vehicle routing problem with soft

time windows by simulated annealing, J Indus Eng, 2005, 1(1),28– 36.

[67] Tan K.C., Chew Y.H. and Lee L.H., A hybrid multi objective evolutionary algorithm for solving vehicle routing problem with time windows, Comput Opt Appl, 2006, 34(1),115–15.

[68] Zhong, Y., Pan, X.: A hybrid optimization solution to VRPTW based on simulated annealing. In: Automation and Logistics IEEE International Conference on, 2007, 3113 –3117.

[69] Debudaj-Grabysz, A., Czech, Z.J.: A concurrent implementation of simulated annealing and its application to the VRPTW optimization problem. In: Distributed and Parallel Systems. Volume 777 of The Kluwer International Series in Engineering and Computer Science. Springer US, 2005, 201–209.

[70] Li, Y.: An improved simulated annealing algorithm and its application in the logistics distribution center location problem. Applied Mechanics and Materials 389 ,2013, 990–994.

[71] Cordeau, J., Laporte, G., Hautes, E., Commerciales, E., Gerad, L.C.D.: A unified tabu search heuristic for vehicle routing problems with time windows. J. of the Oper. Res. 2001, Society 52 , 928–936.

[72] Ho, S.C., Haugland, D.: A tabu search heuristic for the vehicle routing problem with time windows and split deliveries. Comp. and Oper. Res. 2001, 31, 1947–1964.

[73] Tan, X., Zhuo, X., Zhang, J.: Ant colony system for optimizing vehicle routing problem with time windows (VRPTW). In: Proc. of the 2006 int. conf. on Comp. Intell. and Bioinf. ICIC'06, Berlin, Heidelberg, 2006, Springer-Verlag 33–38.

[74] Qi, C., Sun, Y.: An improved ant colony algorithm for VRPTW. In: Proc. of the 2008 Int. Conf. on Comp. Sc. and Soft. Eng.2008, 455–458.

[75] Br¨aysy, O., Gendreau, M.: Vehicle Routing Problem with Time Windows, part II: Metaheuristics. Trans. Sc.2005, 39(1), 119–139.

[76] Li, C.L., Simchi-Levi, D., & Desrochers, M. On the Distance Constrained Vehicle Routing Problem. INFORMS, 1992, 40, 790-799.

[77] Nagy, G., & Salhi, S. Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. European Journal of Operational Research , 2005, 162, 126-14.

[78] Cheang, B., Gao, X., null, A. L., Qin, H., & Zhu, W. Multiple pickup and delivery traveling salesman problem with last-in-first-out loading and distance constraints. European Journal of Operational Research,2012, 223, 60-75.

[79] Almoustafa, S., Hana, S., & Mladenovi, N. New exact method for large asymmetric distance-constrained vehicle routing problem. European Journal of Operational Research, 2013, 226, 386–394.

[80] Ruiz, E., Soto-Mendoza, V., Barbosa, A.E., & Reyes, R. Solving the open vehicle routing problem with capacity and distance constraints with a biased random key genetic algorithm. Comput. Ind. Eng., 2019, 133, 207-219.

[81] Auliani, F., Hertini, E., & Nahar, J. Determination distribution route of beverage products with the application of the vehicle routing problem model and sensitivity analysis. Journal of Physics: Conference Series, 2021, 1722.

[82] Gupta, A., Ghosh, S., & Dhara, A. . Deep Reinforcement Learning Algorithm for Fast Solutions to Vehicle Routing Problem. 5th Joint International Conference on Data Science & Management of Data 9th ACM IKDD CODS and 27th COMAD, 2022.