分　类　号：　TP391　　单位代码：10183
研究生学号：　2021532057　　密　　级：公开

吉 林 大 学

硕士学位论文

(学术学位)

基于深度强化学习和局部搜索的车辆路径优化求解方法研究

Research on Vehicle Routing Problem Methods Based on Deep Reinforcement Learning and Local Search

作者姓名：刘畅

专　　业：计算机科学与技术

研究方向：约束求解与优化

指导教师：张永刚　教授

培养单位：计算机科学与技术学院

2024 年 5 月

基于深度强化学习和局部搜索的车辆路径优化求解方法研究

Research on Vehicle Routing Problem Methods Based on Deep Reinforcement Learning and Local Search

作者姓名：刘畅

专业名称：计算机科学与技术

指导教师：张永刚 教授

学位类别：工学硕士

答辩日期：2024 年 5 月

摘　要

基于深度强化学习和局部搜索的车辆路径优化求解方法研究

随着交通运输与物流配送行业的快速发展，物流企业面对的挑战和机遇并存。特别是在电子商务和即时配送服务日益普及的背景下，如何高效、经济地完成货物配送成为了物流企业亟需解决的问题。在学术研究中，此类物流配送问题被定义为车辆路径问题（Vehicle Routing Problem，VRP）。VRP 问题的传统求解方法主要分为精确算法和启发式算法。然而，随着实际应用场景中问题规模的扩大，传统求解方法面临巨大的计算压力。特别是当问题的约束条件发生变化时，传统算法需要重新进行搜索，不仅增加了时间开销，同时也造成了计算资源的大量浪费。近年来，深度强化学习（Deep Reinforcement Learning，DRL）因其在计算机视觉、自然语言处理等领域展现出强大的计算和序列决策能力，已被广泛应用于 VRP 问题的求解中，为这类问题开创全新的求解模式。

本文围绕车辆路径问题的高效优化策略开展研究，结合深度强化学习和局部搜索来求解 VRP 问题。主要研究工作如下：

（1）针对现有 DRL 方法在学习 VRP 特征时存在局限，并且在求解质量方面与传统算法相比仍存在差距的问题。提出联合学习节点和边特征的图注意力模型（Graph Attention network for Node and Edge，GAT-NE），采用基于图注意力的编码器整合求解过程中生成的边信息来优化节点嵌入，提升 VRP 求解的准确性。此外，提出将基于展开基线策略改进的 REINFORCE 算法应用于模型的训练，旨在冻结策略网络参数并降低训练过程中的方差，进而稳定训练并提高模型性能。为了充分探索 GAT-NE 的求解上限，在测试阶段采用贪婪和随机采样两种解码策略，实现速度与精度的权衡。实验结果表明，相比现有的 DRL 方法，GAT-NE 在公共基准数据集和随机生成的数据集上展示出更优的求解质量和更强的泛化能力。

（2）针对传统大邻域搜索（Large Neighborhood Search，LNS）算法在求解 VRP 时需要引入大量领域知识，但求解质量仍然不高的问题。提出基于 LNS 的深度强化学习算法来求解较大规模的 VRP 问题，称之为带有路径重构策略的动作序列算法（Action Sequence Algorithm based on Path Reconstruction Strategy，AS-PRS）。将 AS-

PRS 重构解决方案的过程定义为一个马尔可夫决策过程，并采用强化学习算法来指导智能体在特定状态下选择最优动作，以构造完整解序列并最大化长期奖励。此外，采用两阶段求解框架，基于 GAT-NE 为 AS-PRS 算法提供一个多样化的初始解集，旨在为 AS-PRS 算法的初始阶段提供一个平衡优化与探索的解集，增强算法的全局搜索能力，避免其早期收敛于局部最优。两阶段求解框架不仅提高解决方案的质量，也降低了 AS-PRS 算法所需的求解时间，显示出 AS-PRS 算法求解大规模 VRP 问题的潜力。在随机生成的测试数据集以及 CVRPLIB 的聚类数据集上的实验结果表明，AS-PRS 算法和两阶段框架在求解质量和求解时间上均有一定的提升。

关键词：

车辆路径问题，深度强化学习，大邻域搜索，图注意力网络，两阶段框架

Abstract

Research on Vehicle Routing Problem Methods Based on Deep Reinforcement Learning and Local Search

With the rapid development of the transportation and logistics industry, logistics companies are faced with concurrent challenges and opportunities. Particularly with the growing popularity of e-commerce and instant delivery services, efficiently and economically completing cargo delivery has become an urgent issue for logistics companies. In academic research, such logistics distribution problems are defined as the Vehicle Routing Problem (VRP). Traditional methods for solving the VRP mainly divide into exact algorithms and heuristic algorithms. However, as problem sizes in actual application scenarios increase, traditional solving methods are under significant computational pressure. Especially when the constraints of the problem change, traditional algorithms need to be rerun, increasing time expenditure and leading to considerable waste of computational resources. In recent years, Deep Reinforcement Learning (DRL) has been widely applied to solve VRP issues due to its powerful computation and sequential decision-making capabilities demonstrated in fields such as computer vision and natural language processing, introducing a new solution paradigm for these types of problems.

This paper focuses on efficient optimization strategies for the Vehicle Routing Problem, integrating deep reinforcement learning and local search to solve the VRP. The main research works are as follows:

(1) Addressing the limitations of existing DRL methods in learning VRP features and the existing gaps in solution quality compared to traditional algorithms. We propose a Graph Attention Network for Node and Edge (GAT-NE), employing a graph attention-based encoder to integrate edge information generated during the solving process to optimize node embeddings, thereby enhancing the accuracy of VRP solutions. Additionally, we propose the use of an improved REINFORCE algorithm based on the rollout baseline strategy for model training, aimed at freezing policy network parameters and reducing variance during training to stabilize training and improve model performance. To fully explore the upper solving limit of GAT-NE, we employ both greedy and random sampling decoding strategies during the testing phase to balance speed and accuracy. Experimental results indicate that compared to existing DRL methods, GAT-NE demonstrates superior solution quality, stronger

generalization capability, and faster solving speed on public benchmark datasets and randomly generated datasets.

(2) In response to the traditional Large Neighborhood Search (LNS) algorithms, which require extensive domain knowledge yet still result in suboptimal solution quality when solving VRP, we propose a Deep Reinforcement Learning algorithm based on Large Neighborhood Search to solve larger-scale VRP issues, termed the Action Sequence Algorithm based on Path Reconstruction Strategy (AS-PRS). The process of reconstructing solutions in AS-PRS is defined as a Markov Decision Process, and reinforcement learning algorithms are used to guide agents in selecting optimal actions in specific states to construct complete solution sequences and maximize long-term rewards. Moreover, a two-stage solving framework is adopted. Based on GAT-NE, we provide a diverse initial solution set for the AS-PRS algorithm, aimed at providing a balanced set for optimization and exploration during the initial phase of the AS-PRS algorithm, enhancing the global search capability of the algorithm and avoiding early convergence to local optima. The two-stage solving framework not only improves the quality of the solutions but also reduces the solving time required by the LNS algorithm, demonstrating the potential of the AS-PRS algorithm in solving large-scale VRP problems. Experimental results on randomly generated test datasets and clustered datasets from CVRPLIB indicate that both the AS-PRS algorithm and the two-stage framework show improvements in solution quality and solving time.

Keywords:

Vehicle routing Problem，Deep Reinforcement Learning，Large Neighborhood Search，Graph Attention Network，Two-Stage Framework

目 录

第 1 章 绪论

1.1 研究背景与意义

车辆路径问题（Vehicle Routing Problem，VRP）作为经典的组合优化问题，自 1959 年由 Dantzig 等人[1]首次提出以来，已成为运筹学领域的经典研究课题。VRP 不仅在物流配送和交通运输等领域发挥着关键作用，还被扩展到电路设计、电网布局、供应链管理和机器人系统等多个领域[2]。车辆路径问题旨在制定最优的路线安排，确保指定数量的车辆从配送中心出发，访问所有客户并返回起点，其关键是在满足车辆容量、服务时间等约束条件下最小化总的路线成本，包括减少总的行驶距离和时间。

随着问题规模及约束条件的增加，VRP 问题的求解难度呈指数形式上升。在这种情况下，采用传统的精确求解方法在多项式时间内找到全局最优解变得异常困难。因此，启发式和元启发式算法被广泛应用以寻找问题的近似最优解。但这些方法往往需要大量的领域知识，以及需要通过不断试错来进行调整，难以将已获取的求解知识扩展到新的问题实例中。此外，对于大规模的问题实例，启发式算法可能需要较长的求解时间，难以满足实际应用场景的复杂性和多样性[3]。

近年来，深度神经网络（Deep Neural Network，DNN）在计算机视觉、自然语言处理等领域中取得了突破性的进展，使用 DNN 为 VRP 构造近似求解器已成为当下 VRP 求解方法研究的新趋势[4]。这些方法主要通过监督学习和强化学习对 DNN 参数进行调整，使网络能够理解不同问题实例之间的内在关系和通用属性，从而提高同分布下其他实例的求解质量。得益于 GPU 高效的矩阵运算能力，DNN 在求解速度上通常优于传统启发式算法。与需要大量高质量标签的监督学习不同，深度强化学习（Deep Reinforcement Learning，DRL）是一种集序列决策能力和感知能力于一体的新型网络架构，不依赖预先准备的标签，仅需大量实例即可进行训练，具有很强的泛化性。DRL，近年来在 AlphaGo[5]、AlphaGo Zero[6]、ChatGPT[7]等模型上的表现展示出其强大的学习和决策能力，与 VRP 在离散空间进行序列决策的特征具有天然的贴合度，已成逐渐成为求解 VRP 的主流方法。菜鸟团队提出深度强化学习与自适应大邻域搜索算法（Adaptive Large Neighborhood Search，ALNS）结合的方法[8]，解决了经典算法无法构造在线 VRP 完整解的问题，于 2021 和 2022 连续两年入围 Franz Edelman 杰出成就奖总决赛，并在车辆路径规划问题最权威的评测平台上打破了多项世界纪录[9][10]。

因此，采用 DRL 求解 VRP 问题，具有求解效率高、泛化能力强和扩展性强的优点。虽然 DRL 在求解精度方面可能与某些启发式方法有差距，但其自动化的特性有助于减少人力成本和时间消耗，为传统方法提供有效补充或替代方案。

1.2 国内外研究现状

车辆路径的求解方法可以分为传统求解方法和基于学习的求解方法。随着 DRL 技术的引入，基于学习的求解方法进一步细化为两种主要类型：一种侧重于从头开始构建解决方案的构造型方法，另一种则专注于对已有初始解进行逐步改善和优化的改进型方法。

1.2.1 传统方法求解 VRP

车辆路径问题的传统求解方法分为两大类：精确算法和启发式算法。精确算法包括分支定界[11]、割平面[12]、和动态规划[13]等，通过将 VRP 建模为整数或混合整数线性规划问题，寻求全局最优解[14]。然而，精确算法在求解时往往需要巨大的时间投入，尤其是在问题规模增大时，算法的效率和可行性受到挑战。

相比之下，启发式算法能在短时间内获得较高质量的近似解。特别是，局部搜索算法已成为求解组合优化问题的关键技术。这些算法通过逐步改进当前解决方案来探索解空间，寻找局部最优解，并采用不同策略跳出局部最优状态，以期望找到全局最优解或更好的近似解。在此基础上，Freisleben 等人[15]提出通用的局部搜索启发式算法，用于解决对称和非对称的旅行商问题（TSP），该算法旨在寻找局部最优解，并通过通用策略跳出局部最优以获得更好的性能。为解决相同问题，Hore 等人[16]提出结合随机策略的可变邻域搜索方法，旨在通过多样化的局部搜索规则来提升解的质量。接着，Helsgaun 等人[17]提出 Lin-Kernighan 启发式算法，该算法能够有效解决多达 7397 个节点的 TSP 问题，生成近似最优解。Pisinger 等人[18]提出适应性大邻域搜索方法，通过设计一个自适应层来选择移除和重新插入的启发式算子集，以应对 VRP 及其变体实现更广泛的探索。陈等人[19]研究了一种基于可变邻域下降和可变邻域搜索的混合启发式方法来处理 CVRP，其中引入扰动算子以从局部最优解跳至全局最优解。这种基于破坏-修复策略的适应性大邻域搜索启发式方法被应用于求解累进型 CVRP 问题。林等人[20]采用了一种混合的启发式算法来解决 CVRP，该算法结合模拟退火和禁忌搜索方法的核心优势：前者增强了解的探索和多样性，后者避免了邻域搜索的循环。通

过模拟退火的随机化搜索机制，算法能有效探索广泛的解空间，增强解的多样性和创新性；而禁忌搜索通过其独特的记忆功能，能够防止算法在已探索的邻域内循环。此外，结合这两种策略不仅加强了算法的全局搜索能力，也证明不同启发式搜索技术的结合能够产生互补效果。Szeto 等人[21]提出了人工蜂群算法及其改进版本来解决 CVRP，维护一个信息素表以指导路线构建。特别地，每只蜂基于信息素表构建解决方案，并通过与其他的蜂交流，使用局部和全局信息更新信息素表。辛等人[22]基于分而治之原则，提出进化多目标路线分组的方法，首先利用多目标进化算法将庞大的 CVRP 问题划分为更小、更易管理的子问题。随后，通过局部搜索方法对这些子问题进行精细调整，进一步改善解决方案的质量。

经过多年发展，这些传统算法已在求解车辆路径问题方面取得显著成功，成为了诸如 Concorde[23]、OR-Tools[24]和 LKH3[25]等著名 VRP 求解器的核心。但传统的启发式算法在 VRP 问题上的求解进展缓慢，现有的启发式算法需要复杂的领域知识和不断试错，甚至在面对问题的轻微变化时，也需要重新初始化来重复求解[26]。

1.2.2 基于学习方法求解 VRP

尽管传统启发式算法在解决 VRP 方面已取得了一系列显著成果，如上述研究所示。但这些算法通常依赖于大量的领域知识来构建有效的求解规则，导致面对新问题或数据集时需重新设计算法规则，限制了其通用性和适应性，且这种方式与当今强调的基于数据驱动和学习能力的计算范式存在明显差异。数据驱动的方法，尤其是基于机器学习的方法，通过从大量实例中学习模式和策略，能够自动适应新的问题设置而无需过多的人工干预。因此，近年来涌现出一系列采用深度学习或将其与启发式相结合的 VRP 求解新方法。这些新兴方法，无论在求解速度还是质量方面，均表现出巨大潜力，已迅速成为领域研究热点。

本文将深度学习在 VRP 求解中的应用划分为两大类：基于学习的构造式方法和改进式方法。

（1）构造式方法

神经网络真正能够有效求解组合优化问题的构造式方法是指针网络[27]。2015 年，Vinyals 等人[28]受到 Sequence-to-Sequence 模型和注意力机制启发，提出一种新的可以求解组合优化问题的神经网络结构——指针网络（Pointer Network，Ptrnet）。该网络

通过编码器将问题的输入序列转换为特征向量，并利用解码器结合注意力机制，以自回归的方式逐步构造解。Vinyals 等人以监督学习的方式训练循环神经网络（Recurrent Neural Network，RNN）来预测节点的输出序列，实验结果表明，当节点数小于 30 时，Ptrnet 能产生较好的解。此后，Joshi 等人[29]和 Groshev 等人[30]同样以监督学习方式训练图卷积神经网络（Graph Convolutional Network，GCN），特别是，Groshev 等人展示了如何利用小规模实例上训练的 GCN 来指导启发式算法，并通过这些解作为标签，以此重新训练模型以求解更大规模的问题。Prates 等人[31]利用消息传递神经网络，实现了在最少监督下对问题实例的学习，取得了接近最优解的成果。Sultana 等人[32]提出了一种非欧氏 TSP 网络架构，解决了上述模型仅在欧氏空间上训练和测试，不能适应节点分布不均匀的情况。与深度强化学习训练的模型相比，监督学习训练的模型需要大量最优路径作为标签，不适用于求解大规模问题。

2019 年，Kool 等人[33]受 Transformer[34]模型的启发提出 AM 模型，该模型考虑到节点相邻结构对求解结果的影响，在端到端的模型中引入了多头注意力机制提取节点特征，在不同节点间传递加权信息，使得节点能够从它们的邻域更多的信息。作者将节点选择过程建模为马尔可夫决策过程（Markov decision process，MDP），使用简单有效的 REINFORCE 算法的 Greedy rollout baseline 策略来训练模型的参数。POMO[35]采用了一种端到端的策略来开发构造性启发式算法，利用强化学习的对称性质，该策略能够在训练过程中并行地生成多个最优解，从而有效加快了模型的训练效率。针对 TSP 和 CVRP 问题，POMO 能够迅速产生近似于最优的解决方案，并且在目标函数的性能上相比传统构造方法更加优越。

一般来说，基于学习的构造式方法在求解 VRP 时具有较高的效率，使得它更适合拓展到在线和实时的优化场景。然而，对于 NP 难的 VRP 问题，这类方法在构造解时易陷入局部最优，因此必须结合额外的后处理技术来改善解的质量。这包括：随机采样[36]，即通过重复随机构造过程来探索解空间；波束搜索[37]，即执行附加的最优搜索过程；主动搜索[38][39]，针对特定实例通过额外的梯度更新来精细调整模型权重；数据增强[40]，利用 VRP 问题的等价性和不变性进行多重解采样；结合局部搜索方法等[41]。

（2）改进式方法

与构造式方法不同，改进式方法通常涉及设计或挑选由神经网络决定的改进算子，

它通过在每一步中应用选定的改进算子来不断改变解的构造，从而在解空间中进行探索，以寻找问题的最优解[42]。最近的研究表明，深度学习在车辆路径问题上的应用并不局限于设计各种端到端模型和训练算法，其能充分结合学习方法和经典启发式方法的优势，利用自身强大的学习能力学习启发式方法的搜索规则，进而提高经典启发式的求解质量或以更低的时间开销生成有竞争力的解决方案。文献[43][44]所提方法在优化效果上达到甚至超过了 LKH3、Google OR-Tools 等专业组合优化求解器，文献[44]在求解速度上也超越了 LKH3、Google OR-Tools 等方法[26]。

在启发式算法的应用中，算法性能受到其众多超参数的显著影响，如模拟退火算法的接受准则、遗传算法的种群规模等[45]。尽管启发式算法在执行过程中会产生大量的节点进化信息，但这些有用的信息并没有被充分利用，因此出现了许多使用学习方法提取隐藏的信息、协助启发式算法的研究，通过学习这些信息来定义算法的最优超参数设置，从而优化搜索过程并提高算法性能。如 Calvet 等人[46]提出了一种基于统计学习的方法来解决参数设置问题，用于调整求解多站点车辆路径问题的迭代局部搜索算法的参数。Cooray 和 Rupasinghe[47]使用简单的聚类技术调整变异率提高遗传算法的性能。Žunić 等[48]构建了一个框架，使用如支持向量机这样的预测模型基于历史数据自适应的控制给定模型和算法的参数，通过在带有时间窗口的异构车队车辆路径问题及一系列实际物流问题上的测试，证实了该框架的有效性。

此外，深度学习方法还可以用于增强经典优化算法，如在局部搜索算法中，通过引入扰动使算法能够考虑相对较差的解决方案，并依靠特定的策略引导搜索过程跳出局部最优。在这种情况下，学习模型本身还可以充当经典优化算法中的破坏算子，被集成到传统的优化算法中。DaCosta 等人[49]使用深度学习模型模拟扰动算子来迭代地改进给定的解决方案，直到达到或接近最优。并进一步通过策略梯度算法来学习基于 2-opt 的改进启发式。Chen 等人[50]受求解车辆路径问题的 ALNS 启发，使用近端策略优化（Proximal Policy Optimization，PPO）算法训练一个层次递归图卷积网络（GCN）作为破坏算子，根据最小开销原则将破坏算子移除的节点插入到不可行解中。在 Solomon 基准数据集和合成数据集上的实验证明，Chen 等人的模型在求解质量和计算效率方面均超过 ALNS。Hottung 等人[51]基于 LNS 算法，提出了一种称为神经大邻域搜索（NLNS）的新算法，将学习到的启发式规则结合到一个复杂的高级搜索过程中。实验结果表明，NLNS 学习到的启发式算法能够超越专家制定启发式规则的 LKH3

求解器。

1.3 本文研究内容

本文聚焦于二维欧式空间下的 VRP 问题，即节点间的距离采用欧式距离进行计算，主要研究基于深度强化学习的求解方法。分别以 AM[33]模型和 NLNS[51]模型为基础，通过设计网络结构、修改训练算法等，来捕获 VRP 问题全局特征进而提高求解质量，提出了两种求解方法。本文的主要研究内容和创新点如下：

（1）针对现有 DRL 方法求解 VRP 时解质量低的问题，设计了一个新的端到端深度强化学习模型 GAT-NE。在 AM 模型的基础上，使用联合学习的方法学习节点和边的特征，对节点和边的关系使用注意力机制进行建模，以有效捕获图结构信息。同时引入了层间残差连接，以避免由于深度模型中梯度消失而导致的模型退化。为了改善训练过程，将基于展开基线改进的 REINFORCE 算法应用于训练中，旨在通过策略网络参数的冻结来减少训练过程中的方差，从而稳定训练并提高模型性能。为进一步提高求解质量，采用三种解码策略实现时间和精度的权衡。在随机生成的实例数据集上训练模型，在公共基准数据集和随机生成的实例数据集上进行对比实验，与传统启发式算法和基于学习的求解方法在求解质量、泛化能力和求解速度进行对比，展示出所提方法的改进效果以及泛化到实际问题的能力。

（2）针对传统 LNS 算法在求解较大规模 VRP 问题面临的局部最优困境和计算时间长的问题。在 NLNS 算法模型架构的基础上，提出了一个带有路径重构策略的动作序列算法 AS-PRS，具有对状态特征感知和学习策略决策效率上的优势。此外，AS-PRS 依据多样性与优势平衡原则，从 GAT-NE 输出的解中筛选形成多样化的解集，将此集合作为 AS-PRS 算法的初始解，进一步增强 AS-PRS 的求解能力。采用基于策略梯度的强化学习算法训练 AS-PRS 算法的模型，引导 AS-PRS 更好的探索解空间，最小化路径长度。在随机生成的数据集以及 CVRPLIB 的聚类数据集上，将 AS-PRS 算法与当前先进的 DRL 方法和传统的启发式算法进行对比实验，展示出 AS-PRS 在解决方案质量和计算效率方面发挥了双重作用。

1.4 论文组织结构

本文主要分为五个章节，具体结构如下：

第一章，绪论。首先介绍研究背景与意义，强调车辆路径问题模型对生产、生活

的重要作用。接着阐述了传统方法求解 VRP 问题的发展过程，并指出传统算法存在的局限性，引出利用学习方法求解 VRP 问题的现状。最后，规划了论文整体的研究方向与架构。

第二章，相关理论基础。该部分主要介绍深度强化学习和局部搜索算法应用于 VRP 问题求解的相关知识。首先介绍了 VRP 问题的基本概念和问题定义。其次阐述了本文所采用的局部搜索算法和深度强化学习技术。最后，对编码器-解码器结构的原理进行概述。

第三章，基于深度强化学习的车辆路径优化算法。在 AM 模型的基础上提出新的模型结构捕捉 VRP 问题的全局特征，并对基于策略梯度的强化学习算法进行优化以训练模型。在随机生成的数据集和公共基准数据集进行实验，验证算法的有效性。

第四章，基于路径重构策略的动作序列算法。基于 NLNS 模型针对 CVRP 问题设计了新的编解码器结构，并将重构解的过程建模为马尔可夫决策过程。最后提出一个两阶段的求解思想，进一步增强动作序列算法的求解能力。

第五章，总结与展望。对本文所做的工作进行总结，并对未来基于 DRL 求解 VRP 问题的改进方向做出展望。

第 2 章 相关知识

2.1 VRP 问题

2.1.1 问题定义

旅行商问题（Travelling salesman problem，TSP）和有容量约束的车辆路径问题（Capacitated Vehicle Routing Problem，CVRP）是车辆路径问题的两个最典型的变体，在多种实际场景中具有广泛应用[52]。TSP 要求规划出一条最短的路径，以便访问一系列指定的客户点，且每个点仅能被访问一次，最终返回起点。而 CVRP 则在此基础上增加了额外的约束：它不仅要求规划最短的路线，还兼顾在满足每辆车载重限制的情况下，确保每个有需求的客户都能得到服务。

（1）TSP 问题的数学模型可以表示如下：

$$minimize \sum_{i,j\in N} e_{i,j}x_{i,j} \quad \cdots\cdots (2.1)$$

$$s.t. \sum_{j\in N, j\neq i} x_{i,j} = 1, \forall i \in V$$

$$\sum_{i\in N, i\neq j} x_{i,j} = 1, \forall j \in V$$

$$x_{i,j} \in \{0,1\}, \forall i,j \in V$$

其中，V是城市的集合，$e_{i,j}$是城市i到城市j之间的距离，$x_{i,j}$是决策变量，如果城市i与城市j直接相连，则$x_{i,j} = 1$，否则$x_{i,j} = 0$。

（2）CVRP 问题的数学模型可以表示如下：

$$minimize \sum_{k\in K}\sum_{i,k\in A} e_{i,j}x_{i,j}^{k}(1) \quad \cdots\cdots (2.2)$$

$$s.t. \sum_{k\in K}\sum_{j\in V, j\neq i} x_{i,j}^{k} = 1, i \in V(2)$$

$$\sum_{k\in K}\sum_{i\in V, i\neq j} x_{i,j}^{k} = 1, j \in V(3)$$

$$\sum_{k\in K}\sum_{i\in V} x_{i,0}^{k} = K(4)$$

$$\sum_{k\in K}\sum_{j\in V} x_{0,j}^{k} = K(5)$$

$$s_0^k = D \ \ \forall k \in K(6)$$

$$s_i^k - q_j \le s_j^k + M\left(1 - x_{i,j}^k\right) \quad \forall i,j \in V, i \ne j, k \in K (7)$$

$$x_{i,j}^k \in \{0,1\}, \{i,j\} \in E (8)$$

其中，$V = \{0,1,\dots,i,\dots,m\}$是节点集合，$i = 0$表示仓库节点，$i = \{1,\dots,m\}$表示客户节点；$E = \{a_{ij}: i,j \in V, i \ne j\}$是边集合，表示客户之间和客户与仓库之间可能的连接；e_{ij}为节点i到节点j的距离；K表示车辆数量。如果节点i和j在车辆k的行驶路线中相邻，则将二元决策变量$x_{i,j}^k$定义为 1，否则定义为 0。D表示每辆车的最大载重量；q_i表示客户i的需求；s_i^k表示车辆k访问城市i后的剩余容量。在公式（2.2）中，目标是最小化总的路径长度；约束（2）和（3）表示每个客户只能被服务一次；约束（4）和（5）表示离开车辆和返回车辆之和应该等于车辆总数，即每个车辆都应该从配送中心出发并返回配送中心；约束（6）为车辆的初始载重；（7）确保车辆k在不超载的情况下从一个城市转移到另一个城市，如果 $x_{i,j}^k = 0$（即车辆k不从城市i到城市j），则$M\left(1 - x_{i,j}^k\right)$会变为一个大数$M$，使得不等式$s_i^k - q_j \le s_j^k + M$总是成立，因此不影响解的可行性，确保该约束只在$x_{i,j}^k = 1$时才起作用；约束（8）用于确保决策变量$x_{i,j}^k$是二元变量。

2.1.2 数据集

VRP 问题的基准数据集一般包括以下两个：

（1）随机生成的数据集：TSP 和 CVRP 问题中，节点坐标均从单位正方形[0,1]×[0,1]中随机生成。对于具有 20、50 和 100 个节点的 CVRP 实例，相应的车辆容量分别设置为 30、40 和 50，客户需求从集合从{1, … ,9}中均匀采样的。

（2）公共数据集：TSPLIB[53]和 CVRPLIB[54]公共基准测试数据。

2.1.3 评价指标

VRP 问题的评价指标为：求解长度、最优解差距以及求解时间。

（1）求解长度：在满足可行性条件下算法输出的解序列的路径长度，该值越小，表明算法的求解能力越强。

（2）最优解差距：当前算法预测的路径长度相对于最佳路径长度的百分比，计算如下：

$$\text{最优解差距} = \frac{\text{当前路径长度} - \text{已知最好的路径长度}}{\text{已知最好的路径长度}} \cdots\cdots\cdots\cdots (2.3)$$

最优解差距值越小，表明算法的求解能力越强。

（3）求解时间：算法输出解序列所花费的时间。

2.2 局部搜索算法

2.2.1 局部搜索算法概述

局部搜索算法是解决组合优化问题的重要算法，特别适用于那些解空间庞大以至于难以通过穷举所有可能解来找到最优解的情况[55]。如图 2.1 所示，这类算法的核心思想为：从一个初始解开始，通过探索这个解的“邻域”，即通过对当前解做出小幅度修改可以到达的解集合来寻找更优解。局部搜索算法的求解效率和效果很大程度上依赖于邻域的定义以及如何选择邻居解。然而，传统的局部搜索算法有时会因为陷入局部最优解而无法进一步优化。为了克服这一限制，若干改进算法被相继提出，如模拟退火法引入了随机性以跳出局部最优解，禁忌搜索通过记忆过去的移动来避免循环搜索，而变量邻域搜索通过系统地改变邻域结构来增加解空间的探索深度。这些改进算法的共同点在于它们都试图通过在全面探索和局部最优之间找到平衡，从而增强算法的全局搜索能力。

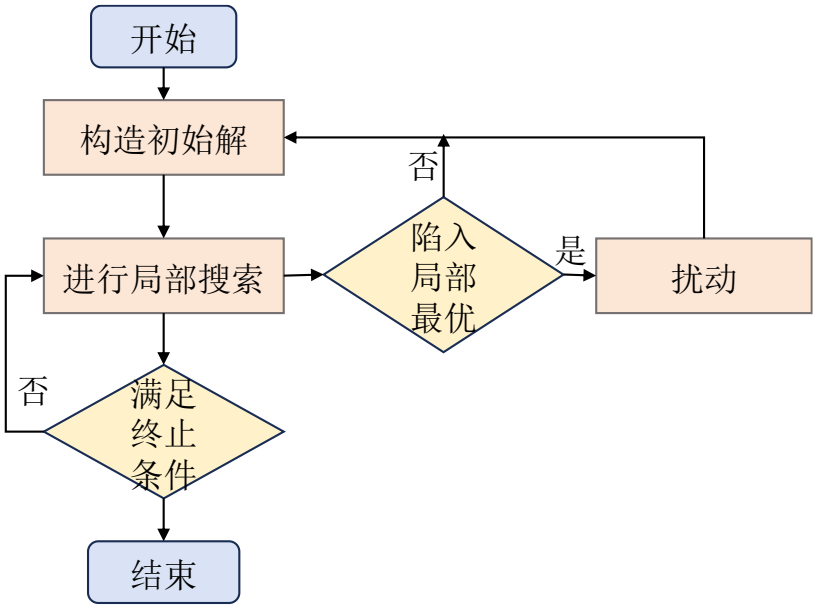

图 2.1 局部搜索算法框架图

2.2.2 大邻域搜索算法

大邻域搜索（Large Neighborhood Search，LNS）算法是局部搜索算法的扩展，如

算法 2.1 所示。LNS 被广泛应用于解决各种复杂的组合优化问题，包括但不限于车辆路径问题、旅行商问题以及其它优化问题。该方法由 Shaw 在 1998 年首次提出[56]，并在随后的研究中得到了广泛的发展和应用。

算法 2.1：大邻域搜索算法

输入：初始解$S_{initial}$，最大迭代次数 $MaxIter$.

输出：最优解S_{best}.

1: $S_{current} \leftarrow S_{initial}$
2: $S_{best} \leftarrow S_{current}$
3: $iter \leftarrow 0$
4: $While\ iter < MaxIter\ do$
5: $S_{destroyed} \leftarrow$ 破坏操作$(S_{current})$ /*移除解的一部分，创建部分未完成的解决方案*/
6: $S_{repaired} \leftarrow$ 修复操作$(S_{destroyed})$ /*应用启发式或元启发式算法修复破坏的解*/
7: If 接受准则$(S_{repaired}, S_{current})\ then$
8: $S_{current} \leftarrow S_{repaired}$
9: $If\ cost(S_{current}) < cost(S_{best})\ then$
10: $S_{best} \leftarrow S_{current}$
11: $End\ If$
12: $iter \leftarrow iter + 1$
13: $End\ While$
14: $Return\ S_{best}$

相比于传统局部搜索算法，LNS 的创新之处在于它扩大了邻域的范围。在传统的局部搜索中，邻域通常由当前解通过执行一系列小的、局部的修改所产生的解集合构成，比如交换、位移或逆转等操作。这些操作仅影响解的一小部分，探索的是相对紧密的邻域空间，容易陷入局部最优解。为了有效地探索更大的邻域而不至于计算成本过高，LNS 算法的设计采用了一种独特的策略。如图 2.2 所示，该策略通过特定的修复和破坏算子来间接定义解空间的邻域结构。

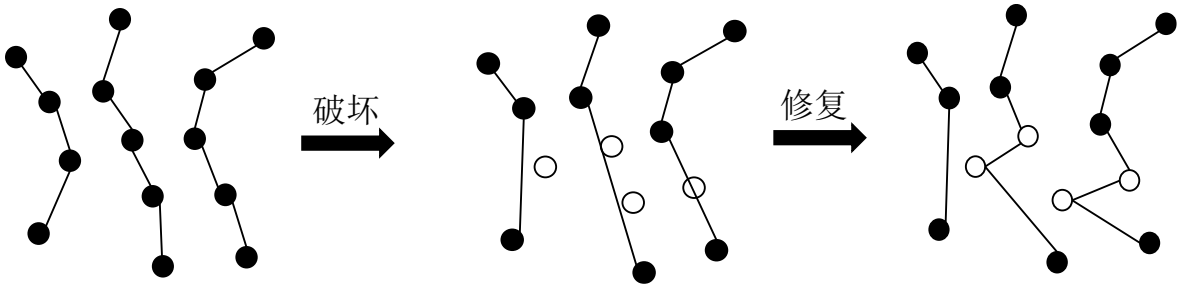

图 2.2 大邻域搜索算法示意图

破坏算子：对于一个特定问题实例的初始解π，LNS 通过对π施加破坏和修复操作来生成一个临近解π'。破坏算子会移除解的某些部分。例如，给定一个车辆路径问

题的初始解，破坏算子可能会去掉几个路线段，导致产生了一个不完整的解。

修复算子：在移除了某些节点后，接下来的任务是将这些节点以某种方式重新加入到解中，以形成一个新的、完整的解决方案。修复算子会对该解进行修补，构造出一个新的可行解π'。最后，使用模拟退火算法中的 Metropolis 接受准则，判断π'是否足以取代当前的解π。重复此过程，直到达到终止条件。

LNS 通过这种破坏和修复的过程，定义了一个非常大的邻域，从原始解出发，几乎可以到达解空间中内任意的新解。由于这种方法比传统的局部搜索范围更广，LNS 在寻求全局最优解时显示出更大的优势。

本文选择 LNS 作为局部搜索模块的基础基于以下原因。首先，LNS 提供了一个清晰的框架，通过其破坏和修复算子来学习邻域函数。其次，修复步骤的复杂度并不随问题实例的大小而变化，使得 LNS 能够处理较大规模的问题实例。此外，LNS 已被广泛成功应用于解决各种优化问题，包括车辆路径优化问题。

2.3 强化学习

强化学习是一种重要的机器学习方法，其根源可以追溯到心理学的行为主义理论，该理论探讨了生物体如何通过环境给予的奖励与惩罚来调整其行为，以期在未来的决策中实现奖励的最大化。这一过程涉及到对环境刺激的响应和预期的形成，是一种基于试错的学习方式。随着深度学习技术的进步，二者的结合已经在多个领域展现出了显著的成果，如击败围棋世界冠军的 AlphaGo[57]，超越星际争霸 II 的人类职业选手的 AlphaStar[58]，以及最近爆火的 ChatGPT 等。除此之外，强化学习在组合优化问题上的应用已成为该领域的研究热点，由于大多数组合优化问题都可以被定义为具有巨大解空间的序贯决策问题，强化学习方法首先对组合优化问题定义环境以及在此环境中运行的智能体（Agent），然后由 Agent 进行求解并进行优化。本文将深入探讨强化学习的基本原理和方法分类，并详细介绍如何利用强化学习解决车辆路径问题。

强化学习利用 Agent 通过马尔可夫决策过程进行与环境交互，Agent 在每一时刻都处于某一特定的状态，并根据其策略选择并执行动作。随后，Agent 会根据其动作效果从环境中获得相应的奖励，并转移到新的状态。强化学习的核心目标在于训练 Agent 以最大化其未来收益的累积值，即最大化总回报。在这个过程中，Agent 不必完全掌握其所处环境的全部动态，而是通过直接或间接的方式学习获得最佳策略。为

了实现这一过程，强化学习需要借助编码器，也就是一种将问题状态转化为数值表示的函数。在组合优化问题的求解算法中，已经提出了多种类型的编码器，包括递归神经网络（RNN）、图神经网络（GNN）、基于注意力机制的网络和多层感知器等，这些编码器能够有效地将问题的状态信息编码，为 RL 算法提供决策的基础。

如图 2.3 所示，使用强化学习解决组合优化问题的流程如下：首先，需要将问题建模为一个马尔可夫决策过程，明确定义状态、动作及其对应的奖励。随后，使用状态编码器将问题的状态输入转换为特征向量，并为每个动作提供一个价值或选择概率。接着，应用强化学习算法调整编码器参数，以在给定的 MDP 框架下作出最佳决策。当 Agent 执行动作后，环境便转移到新状态并给出该动作的奖励。这一过程重复进行直到达到时间限制。一旦 Agent 模型参数训练完成，它便能够搜索新实例的解决方案。

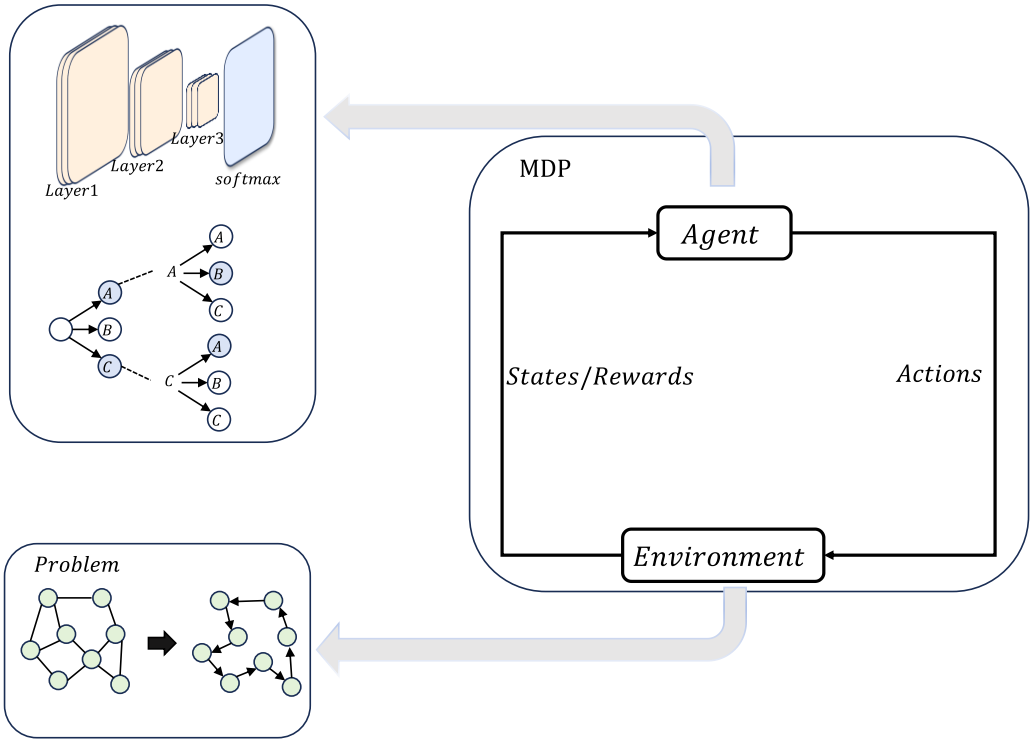

图 2.3 使用 RL 求解组合优化问题

2.3.1 马尔可夫决策过程

为了将强化学习应用于 VRP 问题中，需要将节点选择过程建模为马尔可夫决策过程（Markov Decision Process，MDP）。MDP[59]通常定义为一个五元组$M = < S, A, R, T, \gamma, H >$。

其中，$S = \{s_1, s_2, \ldots, s_t\}$为环境中状态空间，S 有以下两种定义方式：如果解是增

量式构造的，将其定义为问题的部分解的集合（部分路径）；如果解是从问题的次优解开始迭代的改进，则将它定义为次优解的集合。

$A = \{a_1, a_2, \dots, a_t\}$为动作集合，集合中的每个元素对应着$t$时刻 Agent 所选择的动作$a_t$。对于车辆路径问题则表示在当前时刻选择节点到部分解或改进完整解。

R 为奖励函数，是一个从状态和动作映射到实数奖励的函数$R: S \times A \to R$。

T 为转移函数$T(s_{t+1}|s_t, a_t)$，根据做出的动作，控制从一个状态到另一个状态的转移。在车辆路径问题中，转移函数通常是确定的。

γ为折扣因子，$0 < \gamma \leq 1$。不同时刻所产生的奖励对当前时刻的影响是不同的，折扣因子鼓励 Agent 优先考虑短期奖励。

H 为一个情节（episode）的长度，其中 episode 定义为包含序列和动作的集合$\{s_t, a_t, s_{t+1}, a_{t+1}, \dots\}_{t=0}^{H}$。对于构造式的求解方法，episode 的长度是指达到可行解时所需的动作总数，对于迭代改进解的方法，情节长度由设定的最大优化迭代次数决定。

在 MDP 中，Agent 的目标是找到一个将状态映射为动作的策略$\pi(s)$，进而找到求解 MDP 的最优策略π^*使得累积回报$E(\sum_{t=0}^{H} \gamma^t R(s_t, a_t))$最大化，如公式（2.4）所示。

$$\pi^* = argmax_\pi E[\sum_{t=0}^{H} \gamma^t R(s_t, a_t)] \quad \cdots\cdots (2.4)$$

在将车辆路径问题定义为 MDP 之后，需要选择合适的算法指导 Agent 搜索最优策略π^*。搜索最优策略的强化学习方法大致可分为基于模型的方法和无模型的方法，如图 2.4 所示。

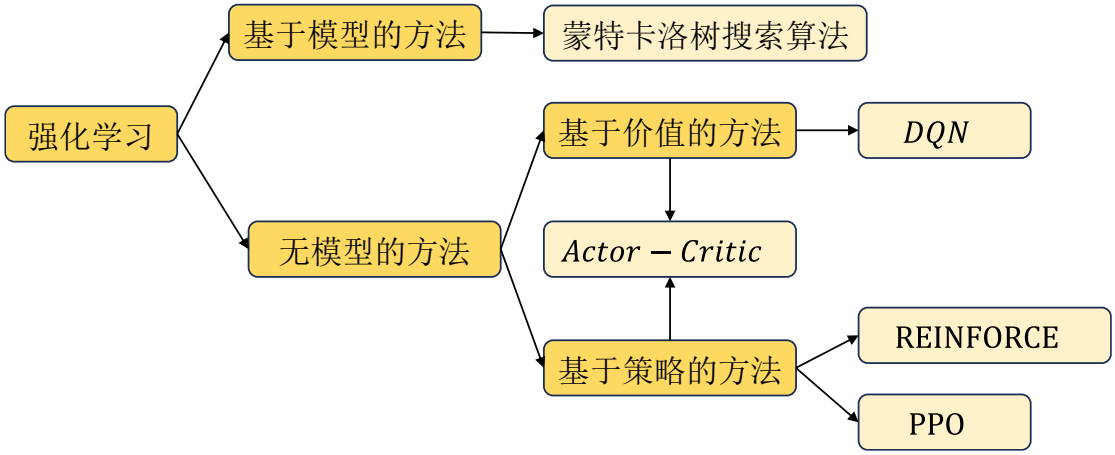

图 2.4 强化学习方法分类图

基于模型的方法适用于转移函数已知或可被学习的环境，算法在决策过程中可以

直接利用环境信息，如蒙特卡洛树搜索算法。无模型方法则仅依赖 Agent 累积的经验来探索策略，这类方法又可以细分为基于价值的方法和基于策略的方法两种。基于价值的方法侧重于对值函数进行近似，值函数用于量化特定状态下各个动作的潜在价值，并依据这些估值来指导动作选择，间接学习最优策略。基于策略方法则直接对策略进行建模和优化，使用梯度上升等技术改进策略。此外，还存在将基于价值的方法和基于策略方法相结合的强化学习算法，如演员-评论家（Actor-Critic）方法，它通过并行地更新策略（演员）和价值函数（评论家）来加速学习过程。

2.3.2 基于价值的方法

强化学习方法将 VRP 建模为 MDP 的目标是使 Agent 学习最优策略π^*，以最大化累计奖励。基于价值的强化学习方法旨在通过近似价值函数和动作价值函数来确定最优策略。

在强化学习方法中，价值函数$V(s)$被定义为从状态s出发通过策略π获得的预期折扣奖励的期望，该值越大则表明当前环境所处的状态越好。

$$V(s) = E[\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi, s_0 = s] \cdots\cdots (2.5)$$

动作价值函数$Q(s,a)$描述了在状态s下，根据策略π采取特定的动作a所获得的预期折扣奖励的期望，该值越大则表明当前状态下所采取的动作越好。

$$Q^{\pi}(s,a) = E[\sum_{t=0}^{\infty} \gamma^t R(s_t,a_t)|\pi, s_0 = s\,, a_0 = a] \cdots\cdots (2.6)$$

价值函数的最优值$V^{\pi}(s)$是通过所有可能动作下的最大Q值来确定的，代表了在状态s下可能获得的最大回报。

$$V^{\pi}(s) = \max_{a} Q^{\pi}(s,a) \cdots\cdots (2.7)$$

基于贝尔曼方程[60]，状态s和下一状态s'的值函数以及动作价值函数的关系为：

$$V^{\pi}(s) = \gamma \sum_{s'} T(s,\pi(s),s')V^{\pi}(s') + R(s) \cdots\cdots (2.8)$$

$$Q^{\pi}(s) = \gamma \sum_{s'} T(s,a,s') \max_{a'} Q^{\pi}(s',a') + R(s,a) \cdots\cdots (2.9)$$

在策略评估阶段，如果策略π_1的性能优于策略π_2，即策略产生的状态价值函数$V^{\pi_1}(s) > V^{\pi_2}(s)$对所有 s 成立，那么策略$\pi_1$在策略改进阶段的性能也将超越策略$\pi_2$。因此，策略$\pi^*$将产生最优的价值函数：

$$V^{\pi^*}(s) = \max_{\pi} V^{\pi}(s), \forall s \in S \cdots\cdots (2.10)$$

同理，最优动作价值函数：

$$Q^{\pi^*}(s,a) = \max_{\pi} Q^{\pi}(s,a), \forall s \in S, \forall a \in A \cdots\cdots (2.11)$$

如果$Q^{\pi^*}(s,a)$是已知的，则基于以上可以获得最优策略π^*。因此，基于价值函数的方法的关键在于寻找最优（动作）价值函数。常见的近似最优价值函数的方法主要有两种：

（1）Q-learning[61]：Q-learning 是一种典型的基于价值的强化学习方法，它通过更新动作价值函数来学习在给定状态下采取何种动作以最大化长期奖励。

（2）Deep Q-learning[62]：Deep Q-learning 是传统的 Q-learning 的扩展，通常使用深度神经网络来近似和学习动作价值函数，使得算法能够处理高维输入并提高学习效率。

2.3.3 基于策略的方法

基于策略的方法不依赖于价值函数的评估，而是直接学习动作的概率分布来寻找最优策略。在基于强化学习求解车辆路径问题的研究中，常见的基于策略的方法有：

（1）REINFORCE 算法[63]：REINFORCE 属于蒙特卡洛方法的一种，它依赖于对完整的序列（或情节）的采样来估计策略π的梯度。在 REINFORCE 算法中，策略通常用某种参数化形式表示（如神经网络），并通过梯度上升法调整这些参数来优化策略。策略π的梯度计算如下：

$$\nabla_{\theta} J(\pi_{\theta}) = E_{\pi_{\theta}}[\sum_{t=0}^{H} \nabla_{\theta} log \pi_{\theta}(a_t|s_t) G(t:H)] \cdots\cdots (2.12)$$

$$G(t:H) = \sum_{t=t'}^{H} \gamma^{t'-t} r_{t'+1} \cdots\cdots (2.13)$$

（2）带基线的 REINFORCE（REINFORCE with baseline）算法[64]：采用 REINFORCE 算法训练模型参数时，通常基于当前策略进行奖励估计可能导致初始阶段性能波动大，产生较大的方差。其次，为增强模型探索新路径的能力，基于概率分布的随机采样机制会产生多个行为轨迹，也会引入额外的方差。为了缓解这些问题，引入了带基线的 REINFORCE 算法降低方差。将公式（2.12）改写为：

$$\nabla_{\theta} J(\pi_{\theta}) = E_{\pi_{\theta}}[\sum_{t=0}^{H} \nabla_{\theta} log \pi_{\theta}(a_t|s_t) G(t:H) - b(s_t)] \cdots\cdots (2.14)$$

（3）近端策略优化（Proximal Policy Optimization，PPO）算法[65][64]：PPO 算法通过重要性采样来校正目标偏差，它通过对比新旧策略产生的样本来衡量两者之间的差异，并剪裁那些差异过大的样本梯度，从而减少样本方差。

（4）演员-评论家（Actor-Critic）算法[66]：Actor-Critic 算法结合了基于价值方法和策略梯度方法的优点，通过一个参数化的值函数（Critic）来代替传统的基线，并使用演员（Actor）网络来学习优化的策略。在这种框架下，演员负责根据当前策略执行动作，而评论家则评估这些动作的好坏，即评估当前策略的价值。这种结构允许算法不必等待整个 episode 完成即可更新参数，能够实现更为频繁和即时的学习。通过这种相互学习的机制，Actor-Critic 方法能够在探索环境和优化策略之间保持平衡，加速学习过程。

2.4 图神经网络

图神经网络（Graph Neural Network，GNN）的核心意义在于其能够有效处理图结构数据，即能够捕捉和利用数据点之间的关系（例如社交网络、分子结构、交通网络等）。与传统的神经网络相比，GNN 特别适合处理具有图结构的数据，保留节点间的结构信息，这是传统神经网络难以实现的。由于许多组合优化问题都具有图结构的特征，非常适合采用图嵌入技术对组合优化问题进行建模。最近，GNN 在信息嵌入和图拓扑的信念传播方面显示出的卓越能力，为图组合问题建模提供了强大支持[67][68]。

通常将组合优化问题表示为$G =< V, E >$，其中V是节点的集合，E是边的集合。图神经网络以$G =< V, E >$作为输入，通过多次迭代学习节点特征（如节点坐标、需求、服务时间等）并聚合邻居的信息来更新节点嵌入，使其包含更广泛的上下文信息，而不仅仅是局部邻域的信息。随后，将更新的节点特征应用于决策过程，如使用解码器或其他策略来规划车辆的最优路径。最后，通过优化算法（如强化学习）进一步调整 GNN 的参数以改进 VRP 的解决方案。

2.5 编码器-解码器结构

在 2.3 节中，本文探讨了强化学习的基础理论及其常见算法。为了解决 VRP 问题，必须把问题的输入数据（例如，VRP 问题的节点坐标等）转化为高维特征向量，并借助神经网络来进行决策制定。然后通过强化学习技术搜索最佳策略，并优化网络参数。在此过程中，负责将原始输入特征转换为高维数据的模块称为编码器，而负责

输出解决方案序列的模块则称为解码器。

近年来，编码器-解码器结构[69]已成为视频理解[70]、图像生成[71]、语音合成[72]等生成型任务的核心架构，如 ChatGPT 对话模型、DALL-E 文本图像生成模型等皆基于此架构。该结构最初于 2014 年在神经网络机器翻译领域被提出，基本原理是通过编码器将源数据转换成深层特征，再由解码器将这些特征转换为目标数据。例如，在机器翻译任务中，编码器处理源语言文本，解码器则生成目标语言文本。最初的编码器-解码器模型基于循环神经网络，而后续研究引入了 LSTM、GRU、CNN 以及基于注意力机制的网络来增强模型性能。当前，在神经网络解决组合优化问题的应用中，基于注意力机制的编码器和解码器结构成为了主流选择。

2.6 本章小结

本章主要介绍了基于深度强化学习求解 VRP 问题涉及到的相关知识。首先给出了车辆路径问题的定义，并对常用的数据集及评价指标进行了说明。接着对强化学习和马尔可夫决策过程进行了介绍，包括基于价值的强化学习方法和基于策略的强化学习方法。最后介绍了深度学习相关的网络结构知识，包括图神经网络和编码器-解码器结构。

第 3 章 基于深度强化学习的车辆路径优化算法

AM 模型是经典的基于自注意力机制的强化学习模型，适用于多种优化问题。但由于 VRP 问题的 NP 难特性，基于强化学习构造的近似求解器与基于领域知识构造的启发式方法相比解的质量仍存在较大差距。本章在 AM 模型的基础上提出融合节点和边特征的图注意力模型（Graph Attention network for Node and Edge，GAT-NE），用于捕捉 VRP 问题的全局特征。随后采用融合得到的注意力系数更新每个节点的信息，并传递到全局的图嵌入（Graph embedding）中。此外，将基于展开基线改进的 REINFORCE 算法应用于模型训练中，通过冻结策略网络参数帮助模型减少训练过程中的方差，从而稳定训练并改善训练过程。

3.1 GAT-NE 模型

本章基于 GNN 构建了一个端到端的深度强化学习框架，以探索 VRP 问题的解决方案，如图 3.1 所示。

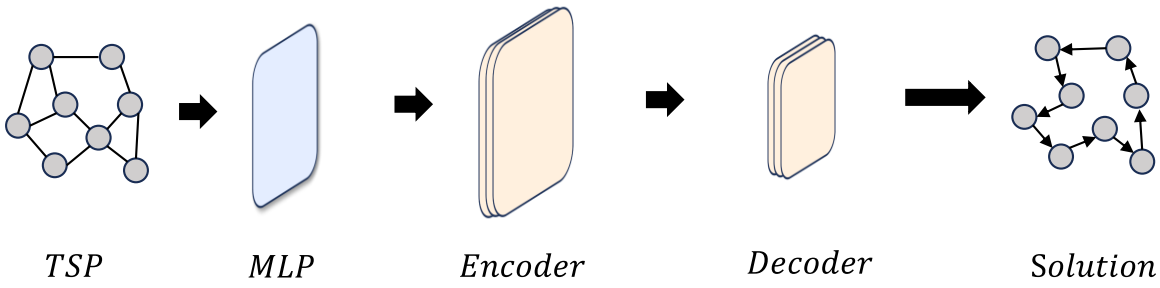

图 3.1 求解 TSP 的端到端模型

首先由多层感知机对 TSP 问题的特征进行初始化。利用基于 GNN 的编码器对这些特征进行编码处理。编码后的特征随后被送入一个集成注意力机制的解码器，以预测选择每个未访问节点的概率。

3.1.1 模型的整体结构与 AM 的对比

与 AM[33]相同，GAT-NE 使用了编码器-解码器结构，编码器中使用去除了位置编码并带有批量归一化的 Transformer 编码器。解码器在每个时刻基于全局图嵌入、前部分解序列的嵌入以及车辆剩余容量构造上下文向量，基于注意力机制对下一时刻的节点选择进行预测。不同点在于 GAT-NE 的编码器联合学习节点和边的信息，利用融

合得到的注意力系数更新每个节点的信息。解码器采用 PtrNet 的方式输出节点，在每个解码时间步将概率值作为“指针”来选择输出节点。尽管在结构上与 AM 解码器相似，本文的解码器却采用了与之不同的方法，利用多头注意力和单头注意力两个子层来增强解码过程中的模型表现力。本章将在第 3.2.2 节介绍编码器结构，在第 3.2.3 节介绍 GAT-NE 的解码流程。

3.1.2 编码器

编码器结构如图 3.2 所示。编码器将问题以图$G=\{V,E,W\}$的形式作为输入。例如，TSP 问题的输入特征为(v_i,e_{ij})，v_i为 TSP 节点的坐标，e_{ij}为节点i,j之间的欧式距离，$i,j\in\{1,\dots,m\}$。

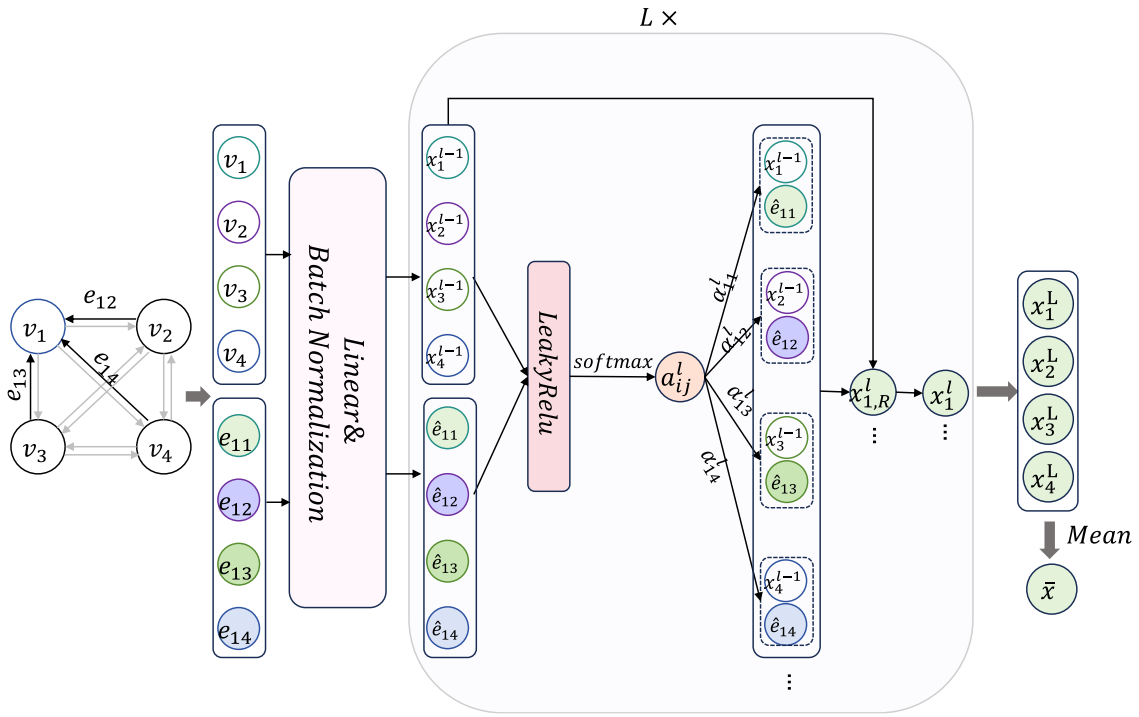

图 3.2 GAT-NE 的编码器结构

随后，通过线性层将特征(v_i,e_{ij})编码为d_x和d_e维的特征向量，再经过批处理模块将特征向量输入到中进行更新。公式（3.1）和（3.2）描述了 TSP 特征嵌入的过程，上标 0 表示节点的初始特征（如二维坐标）。

$$x_i^0=BN(A_0v_i+b_0),\forall i\in\{1,\dots,m\} \quad (3.1)$$

$$\hat{e}_{ij}=BN\left(A_1e_{ij}+b_1\right),\forall i,j\in\{1,\dots,m\} \quad (3.2)$$

其中，A_0和A_1为可训练的权重矩阵，$BN(\cdot)$表示批量归一化（Batch Normalization）操作。GAT-NE 的编码器由L个结构相同但参数不同的编码层组成，节点特征$x^0=$

$\{x_1^0, x_2^0, \ldots, x_m^0\}$和边特征$\hat{e} = \{\hat{e}_{11}, \hat{e}_{12}, \ldots, \hat{e}_{mm}\}$为编码器第一层的输入。经编码器层理后，输出一组新的节点特征$x^1 = \{x_1^1, x_2^1, \ldots, x_m^1\}$，其中$x_i^l \in \mathbb{R}^{dx}$。$\ell \in \{1, \ldots, L\}$用于标记第$l$层的节点嵌入$x_i^l$。在该编码过程中，边特征$\hat{e}$始终保持不变，仅用于更新节点特征。

图 3.2 描述了第l个编码层如何联合学习节点和边的特征，并更新每个节点的特征，具体流程如下：首先将第$l-1$层输出的节点$x_i^{\ell-1}$和$x_j^{\ell-1}$以及边特征$\hat{e}$通过权重矩阵W^ℓ进行转换，然后通过$LeakyReLU$激活函数生成注意力系数α_{ij}^ℓ，最后通过$Softmax$函数对α_{ij}^ℓ进一步归一化，α_{ij}^ℓ表示在第ℓ层节点j对节点$i(i, j \in \{1, \ldots, m\})$的重要程度。随后，利用融合得到的注意力系数$\alpha_{ij}^\ell$更新每个节点$i$的特征，通过其所有邻接节点$j$的特征$x_j^{\ell-1}$以及它们与$i$之间的边特征$\hat{e}_{ij}$结合注意力系数$\alpha_{ij}^\ell$来更新节点$i$的特征。注意力系数$\alpha_{ij}^\ell$计算如下：

$$\alpha_{ij}^\ell = \frac{\exp\left(LeakyReLU\left(g^{\ell T}\left[W^\ell\left(x_i^{\ell-1} || x_j^{\ell-1}\right) || \hat{e}\right]\right)\right)}{\sum_{z=1}^m \exp\left(LeakyReLU\left(g^{\ell T}\left[W^\ell\left(x_i^{\ell-1} || x_z^{\ell-1}\right) || \hat{e}\right]\right)\right)} \quad (3.3)$$

其中，||是连接操作，g^ℓ和W^ℓ为可训练的权重矩阵。此外，本文在每两层编码器之间设置了残差连接。第ℓ层编码器的输出为：

$$x_i^\ell = x_{i,R}^\ell + x_i^{l-1} \quad (3.4)$$

其中，$x_{i,R}^\ell$是第ℓ层编码器的输出的向量，表示如下：

$$x_{i,R}^l = \sum_{j=1}^m \alpha_{ij}^\ell W_1^\ell x_j^{l-1} \quad (3.5)$$

W_1^ℓ表示可训练的权重矩阵。经过L层编码，最终输出的节点嵌入为x_i^{L}，图嵌入为$\bar{x} = \{\bar{x}_1, \ldots, \bar{x}_j, \ldots, \bar{x}_{d_x}\}$，如公式（3.6）所示。

$$\bar{x}_j = \frac{1}{m}\sum_{i=1}^m (x_i^L)_j, j = 1, \ldots, d_x \quad (3.6)$$

3.1.3 解码器

GAT-NE 的解码器采用了一种类似于 Kool 等人[33]提出的注意力机制和基于 Transformer 模型的解码器部分构建的。不同于传统编码器-解码器架构中常用的循环层，GAT-NE 模型利用多头注意力和单头注意力两个子层来进行信息处理。

虽然 Transformer 模型在序列到序列的转换任务中表现突出，但该模型的输出维度是固定的，而组合优化问题要求模型能够根据不同的输入序列长度动态调整输出序

列的长度，因此无法直接将 Transformer 模型应用于组合优化问题。此后，Vinyals 等人[28]提出指针网络（PtrNet），通过特殊的注意力机制，使模型在每一步解码时从输入序列中动态选择元素，适用于输出长度变化的组合优化问题。本文解码器采用 PtrNet 的方式输出节点，在每个解码时间步将概率值作为“指针”来选择输出节点。解码过程如图 3.3 所示。

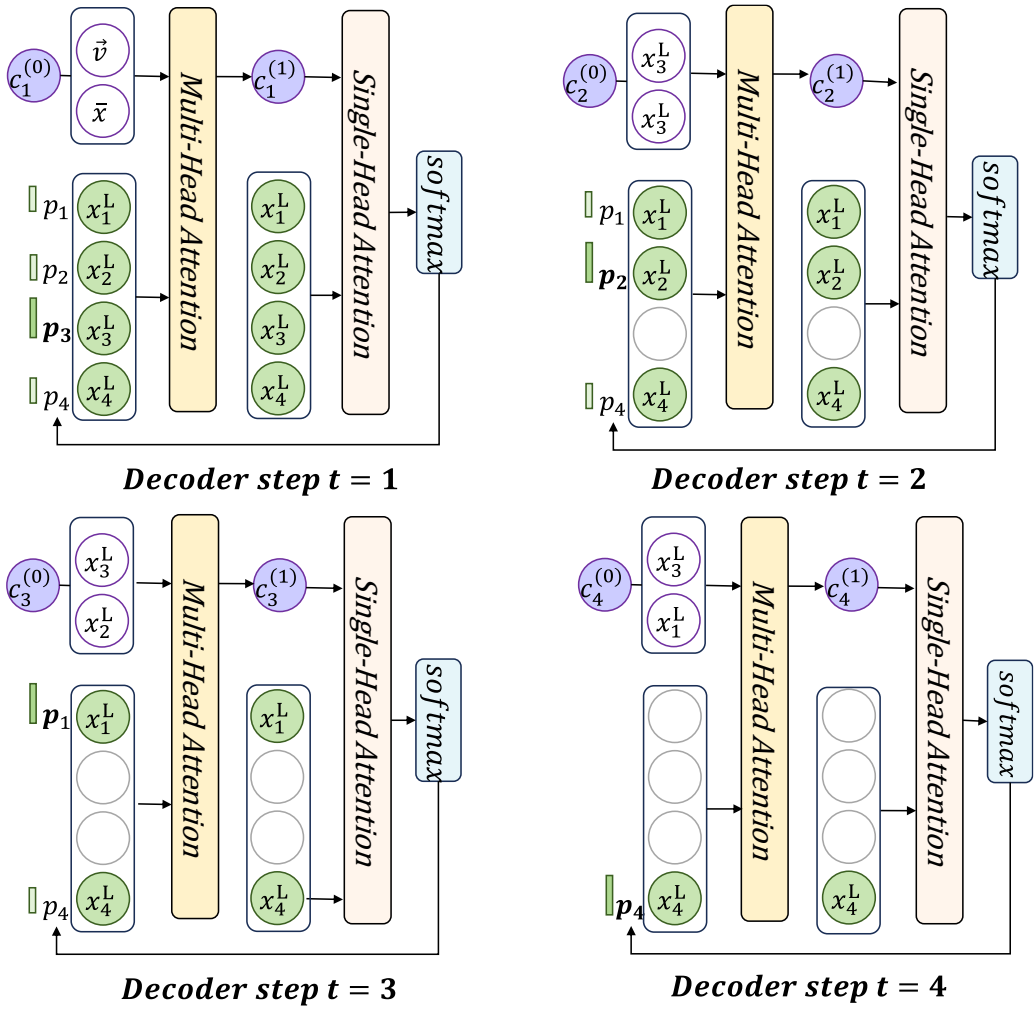

图 3.3 GAT-NE 解码器结构

在解码阶段，首先根据节点嵌入计算t时刻的上下文向量c_t。解码器以图嵌入$\bar{x}$和节点嵌入x_i^L作为输入。当$t=1$时（即第一个解码步骤），上下文向量$c_t^{(0)}$是由图嵌入$\bar{x}$和可学习的占位符参数$\vec{v}$直接相加。当$t>1$时，$c_t^{(0)}$通过图嵌入$\bar{x}$、$t=1$时间步选择的节点嵌入$x_{\hat{\pi}_1}^L$、上一时间步选择的节点嵌入$x_{\hat{\pi}_{t-1}}^L$以及一个可训练的权重矩阵W_x计算，计算方式如下：

$$c_t^{(0)} = \begin{cases} \bar{x} + W_x(x_{\hat{\pi}_1}^L \parallel x_{\hat{\pi}_{t-1}}^L), & t>1 \\ \bar{x} + \vec{v}, & t=1 \end{cases} \quad \cdots\cdots (3.7)$$

随后，解码器将上下文向量$c_t^{(0)}$输入到多头注意层产生一个新的上下文向量$c_t^{(1)}$，该层为解码器的第一层，工作流程如下：

（1）计算 key 向量：$k_i \in \mathbb{R}^{d_v}$；value 向量：$v_i \in \mathbb{R}^{d_v}$；query 向量$q \in R^{d_v}$。用于帮助模型确定节点之间的重要性和关系。

$$q = W^Q c_t^{(0)}, k_i = W^K x_i^{\mathrm{L}}, v_i = W^V x_i^{\mathrm{L}}, i \in \{1, 2, \dots, m\} \cdots\cdots (3.10)$$

其中，$W^K \in R^{d_v \times d_x}, W^Q \in R^{d_v \times d_x}, W^V \in R^{d_v \times d_x} (d_v = d_x/H)$是可训练的权重矩阵，$H$为注意力头的个数。$q$由$c_t^{(0)}$和$W^Q$计算得出，$k = \{k_1, \dots, k_m\}$和$v = \{v_1, \dots, v_m\}$由编码器输出节点嵌入$x_i^{(L)}, i \in \{1, 2, \dots, m\}$和相应的权重矩阵计算。query 向量$q$和 key 向量$k = \{k_1, \dots, k_m\}$用于计算在$t$时间步第一层$Decoder$的注意力系数$u_{i,t}^1 \in R, i = \{1, \dots, m\}$。$u_{i,t}^1$表示在给定时间步$t$时，节点$i$被选择的注意力权重。为了避免在时间步骤$t$之前已经被选中的节点影响注意力机制，本文将它们的注意力系数设为$-\infty$来进行屏蔽，即可在后续避免选择相同的节点，如公式（3.8）所示：

$$u_{i,t}^1 = \begin{cases} \dfrac{q^T k_i}{\sqrt{d_v}}, if \ \ i \neq \hat{\pi}_{t'} \ \forall t' < t \\ -\infty, otherwise \end{cases} \cdots\cdots (3.8)$$

（2）将每个注意力头$h \in \{1, \dots, H\}$输出的注意力系数$\left(\hat{u}_{i,t}^1\right)^h, 1 \leq i \leq m$通过$softmax$进行归一化，并将它们全部串联起来再经过简单的线性层进行处理，输出新的上下文向量$c_t^{(1)}$：

$$\hat{u}_{i,t}^1 = softmax\left(u_{i,t}^1\right) \cdots\cdots (3.9)$$

$$c_t^{(1)} = W_f \left(||_{h=1}^H \sum_{i=1}^m \left(\hat{u}_{i,t}^{(1)}\right)^h v_i^h \right) \cdots\cdots (3.10)$$

该多头注意力层通过并行考虑数据的多种表示形式和特征，增强了模型的表达力并提高了学习过程的稳定性[73]。

解码器的第二层为单头注意力层，该层以多头注意力层输出的上下文向量$c_t^{(1)}$作为输入来计算在时间步t的注意力系数$\hat{u}_{i,t}^{(2)} \in R, i \in \{1, \dots, m\}$。如公式（3.11）所示，本文使用$tanh$激活函数将注意力系数$\hat{u}_{i,t}^{(2)}$限制在$[-C, C]$的范围内，其中$C = 10$。随后，通过$softmax$函数计算$t$时间步选择节点$i$的概率$p_{i,t}$，如公式（3.12）所示。

$$u_{i,t}^{(2)} = \begin{cases} C \cdot \tanh\left(\frac{c_t^{(1)^T} k_i}{\sqrt{d_v}}\right), & if\ i \neq \hat{\pi}_{t'}, \forall t' < t \\ -\infty, & otherwise \end{cases} \quad \cdots\cdots (3.11)$$

$$p_{i,t} = p_\theta(\hat{\pi}_t \mid s, \hat{\pi}_{t'}\ \forall t' < t) = softmax\left(u_{i,t}^{(2)}\right) \quad \cdots\cdots (3.12)$$

最后，根据概率分布$p_{i,t}$，使用随机采样或贪婪解码策略来预测下一个要访问的节点。

3.1.4 解码策略

如 3.1.3 节所述，解码器在每个时间步t都会输出待访问节点的概率分布$p_{i,t}$。根据概率分布$p_{i,t}$，以自回归的方式确定在当前时刻应选择的节点，从而构造一个完成的解序列$\hat{\pi} = (\hat{\pi}_{1,\ldots,}\hat{\pi}_m)$。将这种以自回归构造解序列的过程看作一个优化问题：

$$\max_{\hat{\pi}_1,\ldots,\hat{\pi}_k} \prod_{t=1}^{k} p_\theta(\hat{\pi}_t \mid s, \hat{\pi}_{t'}\ \ \forall t' < t) \quad \cdots\cdots (3.13)$$

通常，需要采用穷举法获取该问题的最优解，其时间复杂度为$O(n!)$。随着节点数量的增加，穷举法的时间消耗变得不切实际。因此，为了平衡求解质量和求解时间，需要在解码阶段引入一些额外解码策略。本文采用的解码策略包括以下三种：

（1）贪婪策略

该策略在每一步中贪婪地选择具有最高概率的节点，并且一旦节点被访问过，就会被屏蔽掉，不再参与后续的选择。

（2）随机采样

在每个时间步t，根据解码器输出的概率分布$p_\theta(\hat{\pi}_{t'}|s, \hat{\pi}_{t'}\ \forall t' < t)$随机选择一个节点。在测试阶段，为了保证 GAT-NE 输出多样化的解序列，通过温度超参数[74]$\lambda \in R$来调整原有的概率分布，如公式（3.14）所示：

$$p_{i,t} = p_\theta(\hat{\pi}_{t'}|s, \hat{\pi}_{t'}\ \forall t' < t) = softmax\left(\frac{u_{i,t}^{(2)}}{\lambda}\right) \quad \cdots\cdots (3.14)$$

温度超参数λ采用网格搜索来确定。对于 TSP 问题，当节点数分别为 20、50 和 100 时，温度超参数的最优设置为 2、2.5 和 1.5。对于 CVRP 问题，当节点数分别为 20、50 和 100 时，温度超参数的最优设置为 2.5、1.8 和 1.2。

（3）主动搜索

Active Search 是一种交互式的搜索策略，该策略在搜索过程中不断更新模型参数，

使得模型能够更好地适应问题的特定情况。具体来说，本文在每个测试样本上执行了10,000 个训练步骤的主动搜索，每步使用 128 的批量大小，从而共采样了 1,280,000 个候选解决方案，有助于增加解的多样性以寻找最优解。在该过程中，学习率设定为1×10^{-5}来微调模型参数。

在训练阶段，本文采用随机采样策略来探索环境获得更好的模型性能。而在测试阶段，使用贪婪解码和主动搜索方法获得高质量的解。

3.2 基于展开基线的 REINFORCE 训练算法

基于展开基线的 REINFORCE 算法（REINFORCE with Rollout baseline）是一种结合传统 REINFORCE 算法和 Rollout 策略评估技术的策略梯度算法，用于解决建模为马尔可夫决策过程的 VRP 问题。该算法的核心目标是训练策略网络 GAT-NE，以实现从状态到动作的最优映射，从而最小化 VRP 解序列的路径长度。

对于 VRP 问题实例s，根据 GAT-NE 产生的概率分布$p_\theta(\hat{\pi}|s)$以及相应的解码策略，即可获得 VRP 问题的完整解序列$\hat{\pi} = (\hat{\pi}_{1,\ldots,}\hat{\pi}_m)$。对于解序列中每一个动作，REINFORCE 算法需要计算从该时间点开始直到情节结束的累积奖励，即路径的总长度$L(\hat{\pi}|s)$。同时，采用 Rollout 策略来生成额外的解序列，并计算这些解序列的路径长度，将这些额外计算的路径长度作为基线$b(s)$，以估算每一步的优势$L(\hat{\pi}|s) - b(s)$，根据优势值计算策略梯度以指导学习过程。

基于展开基线的 REINFORCE 算法估计梯度的计算方式如下：

$$\nabla_\theta Loss(\theta|s) = \mathbb{E}_{\hat{\pi}\sim p_\theta(\hat{\pi}|s)}\left[(L(\hat{\pi}|s) - b(s))\nabla_\theta log p_{\theta(\hat{\pi}|s)}\right] \cdots\cdots\cdots (3.15)$$

其中，基线$b(s)$是一个与选择的动作无关的值，它的作用不是改变策略梯度的期望值，而是减少梯度估计的方差，使得训练过程更加平稳。当策略的求解长度$L(\hat{\pi}|s)$大于基线$b(s)$时则抑制，否则增强该策略。最后，通过应用梯度上升法更新策略网络 GAT-NE 的参数。

基于展开基线的 REINFORCE 算法与传统的 Actor-Critic 算法不同，该算法通过引入一个展开基线策略网络代替 Critic 网络，形成了一种独特的双 Actor 结构。与 DQN 算法中冻结目标 Q 网络参数的策略相同，该算法在每个训练周期中冻结 Baseline 策略网络的参数$\pi_{\theta^{BL}}$，在每个训练周期结束后，通过贪心解码策略比较当前训练策略与 Baseline 策略的效果。当在 10000 个评估实例上根据配对t检验显示出显著改进时

（显著性水平为 5%），才更新基线策略网络的参数。

为了进一步提升算法性能，本文在该算法的基础上实施了一些优化策略，包括：

（1）标准化奖励

本文采用标准化奖励函数的方式，将奖励$L(\hat{\pi}|s)$减去其均值并除以标准偏差，标准化处理后的奖励具有零均值和单位方差，有助于保持策略优化的稳定性和效率。

（2）学习率退火

在训练过程中采用学习率衰减策略，并在每个 epoch 结束时（即整个训练数据集完整地通过了一次前向和反向传播过程）更新：

$$l_{new} = l_{old} \cdot (\beta)^{epoch} \quad (3.16)$$

其中，l_{new}和l_{old}分别表示衰减前后的学习率，β为衰减系数。

（3）正交初始化

在使用正交初始化[75]时，神经网络层的权重矩阵被初始化为近似正交矩阵，通过保持层间权重的正交性来改善模型的训练性能和稳定性。

如算法 3.1 所示，改进的 REINFORCE 算法不仅提高了算法的稳定性和效率，其独特的策略更新机制也保障了训练过程的高效性。

算法 3.1：改进的基线 REINFORCE 算法

输入：训练周期总数E，每个训练周期的步数P_e，PPO 步骤T_s，批量大小B，显著性水平α;带有可训练参数θ的 Actor 网络π_θ,带有可训练参数θ^{BL}的基线策略网络$\pi_{\theta^{BL}}$

输出：GAT-NE 的训练参数集θ

1:$\boldsymbol{Initialization}$: $Xavier\ initialization\ \theta, \theta^{BL} \leftarrow 0$

2:$\boldsymbol{for\ each}\ total\ epoch = 1, \dots, E_t\ \boldsymbol{do}$

3: $\boldsymbol{for\ each}\ step = 1, \dots, P_e\ \boldsymbol{do}$

4: $s_i \sim RandomInstance(\,)\ \forall i \in \{1, \dots, B\}$ /*从某个分布中随机采样一个状态*/

5: $\hat{\pi}_i \sim SampleSolution\ (p_\theta(\hat{\pi}_i|s))\ \forall i \in \{1, \dots, B\}$ /*根据当前策略网络,为每个状态 s 采样一个动作*/

6: $\hat{\pi}_i^{BL} \sim GreedySolution\ (p_{\theta^{BL}}(\hat{\pi}_i|s))\ \forall i \in \{1, \dots, B\}$ /*根据基线策略网络，使用贪婪方法为每个状态选择一个动作*/

7: $Compute\ L(\hat{\pi}_i|s), L\left(\hat{\pi}_i^{BL}\middle|s\right)\ via\ s_i, \hat{\pi}_i, \hat{\pi}_i^{BL}$ /*计算当前策略和基线策略下，每个动作的奖励*/

8: $\bar{L}(\hat{\pi}_i|s) \leftarrow Reward\ Normalization\ L(\hat{\pi}_i|s)$ /*对当前策略下的奖励进行标准化处理*/

9: $\bar{L}\left(\hat{\pi}_i^{BL}\middle|s\right) \leftarrow Baseline\ Reward\ Normalization\ L\left(\hat{\pi}_i^{BL}\middle|s\right)$ /*对基线策略下的奖励进行标准化处理*/

续表 算法 3.1：改进的基线 REINFORCE 算法

10: $Compute\ advantage\ estimates\ \hat{A}_i = \bar{L}(\hat{\pi}_i|s_i) - \bar{L}(\hat{\pi}_i^{BL}|s_i)$ /*计算优势估计*/

11: $\nabla_\theta Loss(\theta|s_i) = \frac{1}{B}\sum_{i=1}^{B} E_{\hat{\pi}\sim p_\theta(\hat{\pi}|s)}[\hat{A}_i \nabla_\theta log p_{\theta(\hat{\pi}_i|s_i)}]$ /*计算损失函数关于θ的梯度*/

12: $\theta \leftarrow Adam(\theta, \nabla_\theta Loss(\theta|s_i))$ /*使用 Adam 优化器根据计算出的梯度更新策略网络参数*/

13: ***end for***

14: ***if*** $OneSidedPairedTTest(p_\theta, p_{\theta^{BL}}) < \alpha$ ***then*** /*比较当前策略与基线策略的性能*/

15: $\theta^{BL} \leftarrow \theta$ /*更新基线策略网络参数为当前策略网络 GAT-NE 的参数*/

16: ***end if***

17:***end for***

3.3 实验及结果分析

3.3.1 实验数据及环境设置

本文为 TSP 和 CVRP 随机生成了服从相同分布的数据集，包括 100000 个 20 节点的实例和 80000 个 50 和 100 节点的实例。对于每个数据集都进行了 100 个训练周期（epochs）的训练。验证集和测试集各包含 10000 个问题实例，并且这些数据都服从与训练集相同的分布。另外，还使用了 TSPLIB 和 CVRPLIB 上的真实世界的实例进行了实验。本章所有实例的训练以及所有测试过程均在 Nvidia GeForce RTX3090 GPU 上执行。模型基于 PyTorch 构建，代码使用 Python3.8 编写。实验设置如表 3.1 所示。

表 3.1 模型的超参数设置

超参数	数值	超参数	数值
编码器层数	4	steps T_s	1
学习衰减率	0.96	多头注意力机制头数	8
REINFORCE 学习率	$1\times10^{-3}(m=20)$ $3\times10^{-4}(m=50,100)$	优化器	Adam
节点嵌入网络的维度	128	边嵌入网络的维度	64

3.3.2 GAT-NE 与其他方法的性能比较

（1）TSP 实验结果

表 3.2 展示了 GAT-NE 模型与基于学习的方法、启发式算法和 Concorde 求解器

在测试集上的性能比较。表中第一部分是 Concorde 求解器的结果，本文以该方法作为基准方法进行比较。Concorde 求解器通过分支定界法，割平面和启发式搜索，高效准确地求解 TSP 问题，确保找到最短的路径访问每个城市一次并返回出发点，被认为是目前世界上最快的 TSP 求解器之一。表中第二部分是传统启发式算法的求解结果，最远插入法、最近插值法和随机插值法的详细信息见文献[33]。表中的第三部分展示了基于学习的方法实现端到端求解的结果。

表 3.2 在节点数为 20、50、100 的 10000 个随机生成 TSP 实例上的测试结果

方法	TSP20			TSP50			TSP100		
	长度	差距	时间	长度	差距	时间	长度	差距	时间
Concorde	3.84	0.00%	1m	5.70	0.00%	2m	7.765*	0.00%	82m
最远插入法	3.93	2.34%	1s	6.01	5.44%	2s	8.35	7.53%	7s
最近插值法	4.33	12.76%	1s	6.78	18.95%	2s	9.46	21.83%	6s
随机插值法	4.00	4.17%	0s	6.13	7.54%	1s	8.52	9.72%	3s
PtrNet[28]	3.88	1.04%		7.66	34.39%				
GCN(Greedy)[29]	3.86	0.52%	6s	5.87	2.98%	55s	8.41	8.31%	6m
AM(Greedy)[33]	3.85	0.26%	0s	5.80	1.75%	2s	8.12	4.57%	6s
MDAM(Greedy)[22]	3.84	0.00%	5s	5.73	0.53%	2s	8.12	4.57%	6s
Ours(Greedy)	3.84	0.00%	1s	5.72	0.35%	4s	8.048	3.64%	13s
AM(Sampling)[33]	3.846	0.01%	5m	5.73	0.53%	24m	7.94	2.25%	1h
GCN(Sampling)[29]	3.847	0.01%	20s	5.71	0.18%	2m	7.92	2.00%	10m
Ours(Sampling)	3.84	0.00%	10m	5.706	0.01%	55m	7.86	1.22%	1h

如 2.1.3 节所述，给出了平均路径长度，平均最优解差距和在 10000 个测试实例的平均值表示。差距表示其他方法得到的结果与 Concorde 求解器提供的结果$L(\pi)$之间的平均最优差距，计算方式如下：

$$\text{最优解差距} = \frac{1}{10000}\sum_{i=1}^{10000}\frac{L(\hat{\pi}|s) - L(\pi|s)}{L(\pi|s)} \quad \cdots\cdots (3.17)$$

对于 TSP100，GAT-NE 模型在贪婪解码与随机采样解码的最优解差距分别为 3.64% 和 1.22%，但贪婪解码策略的运行时间显著优于随机采样解码策略，分别为 13s 和 1h。从求解质量来看，GAT-NE 模型在所有规模的数据集上都展现了非常好的性能。这得益于 GAT-NE 模型对问题全局特征的捕捉，证明了该模块所学习的节点特征优于其他方法。解码策略对于学习方法的求解质量均有着明显帮助。但 GAT-NE 在提升后取得

了更好的结果，在三个测试集上均取得了更低的最优差距，如经过随机采样策略改进的最优解差距分别为 0.00%，00.1%，1.22%。从求解时间来看，基于学习的方法（尤其是采用解码策略的）在较小的数据集上表现出较快的求解速度。然而，随着问题规模的增加，所有方法的求解时间都增加，但采用贪心解码策略的 GAT-NE 模型通常比其他学习方法更快，同时保持较低的平均最优差距。

（2）CVRP 实验结果

CVRP 的定义方式与 3.1.2 节中 TSP 问题的定义方式相同。不同点在于$i = 0$代表仓库节点，$i > 0$代表客户节点；车辆容量为$D > 0$，每个客户节点$i = \{1, \dots, m\}$有一个需求$q_i, 0 < i < D$，仓库节点的需求$q_0 = 0$。所有节点都是随机生成的，生成规则如第 2.1.2 节所述。根据 CVRP 问题的特性，对模型输入、掩码操作以及解码器的上下文向量进行调整即可适应模型，调整的方式与 AM 相同。

输入：将节点特征v_i扩展至三维v_i'，v_i'中包括标准化的需求q_i'和节点特征v_i。

$$v_i' = v_i || q_i' \quad \cdots\cdots (3.18)$$

车辆的剩余容量：在每个时间步$t \in \{1, \dots, m\}$中，如果模型选择的是某个客户节点$\hat{\pi}_t$，则基于该客户节点的需求来更新车辆的剩余容量D_t'，否则车辆剩余容量保持不变，计算方式如公式（3.19）所示。对于已满足需求的客户节点，则不必更新它们的需求。

$$D_t' = \begin{cases} D_{t-1}' - q_{\hat{\pi}_t}', & \hat{\pi}_t = i, i \in \{1, \dots, m\} \\ D, & \hat{\pi}_t = 0 \end{cases} \quad \cdots\cdots (3.19)$$

上下文向量：在每个时间步$t \in \{1, \dots, m\}$，上下文向量$c_t^{(0)}$是由图嵌入$\bar{x}$、上一时间步选择节点的嵌入$\hat{\pi}_{t-1}$及车辆的剩余容量D_{t-1}'计算：

$$c_t^{(0)} = \begin{cases} \bar{x} + W_x(x_{\hat{\pi}_{t-1}}^{(L)} \parallel D_{t-1}'), & t > 1 \\ \bar{x} + W_x(x_0^{(L)} || D_t'), & t = 1 \end{cases} \quad \cdots\cdots (3.20)$$

掩码：如果当前时间步t所选择的节点需求超出车辆剩余容量或在时间步t前已被服务，即$q_i' > D_{t-1}'$或者$i \neq \tilde{\pi}_t, \forall t' < t, i \in \{1, \dots, m\}$。在这种情况下，相应客户节点的注意力系数$u_{i,t}^{(0)}, u_{i,t}^{(1)} = -\infty$。一旦车辆从仓库出发，则不会立即返回仓库，相应的注意力系数$u_{0,t}^{(0)}, u_{0,t}^{(1)} = -\infty$。

表 3.3 与表 3.2 相似，展示了 GAT-NE 及其他学习方法对不同规模 CVRP 实例的测试结果，表中其他学习方法的结果均取自 MDAM。在求解质量方面，GAT-NE 优于

其他学习方法，尤其在 20、50 节点规模的 CVRP 实例上效果更加显著，但随着问题规模的增大，GAT-NE 的求解时间也在增加，未来研究将进一步提高模型的求解能力，取得精度与速度的权衡。

表 3.3 在节点数为 20、50、100 的 10000 个随机生成 CVRP 实例上的测试结果

方法	CVRP20			CVRP50			CVRP100		
	长度	差距	时间	长度	差距	时间	长度	差距	时间
LKH3	6.14	0.00%	2h	10.38	0.00%	7h	15.65	0.00%	13h
PtrNet[28]	6.59	7.33%		11.39	9.73%		17.23	10.10%	
AM(Greedy)[33]	6.40	4.23%	1s	10.98	5.78%	3s	16.80	7.35%	8s
MDAM(Greedy)[22]	6.24	1.63%	7s	10.74	3.47%	16s	16.40	4.79%	45s
Ours(Greedy)	6.26	1.95%	2s	10.80	4.05%	6s	16.69	6.65%	14s
AM(Sample)[33]	6.25	1.79%	6m	10.62	2.31%	28m	16.23	3.71%	2h
NeuRewriter[43]	6.16	0.33%	22m	10.51	1.25%	35m	16.10	2.88%	66m
MDAM(Sample)[22]	6.14	0.02%	3m	10.50	1.16%	9m	16.03	2.43%	31m
Ours(Sampling)	6.14	0.00%	14m	10.43	0.48%	1h	15.97	2.04%	3h

3.3.3 泛化性测试

为了评估所提方法的泛化性，本文采用了两个公共基准数据集：TSPLIB 和 CVRPLIB。这些数据集涵盖了不同规模的 TSP 实例和 CVRP 实例，用于评估模型从解决训练期间随机生成的实例到在测试期间处理实际问题的泛化能力。表 3.4 中汇总了各种方法的性能，包括精确算法、深度强化学习方法以及经典的元启发式算法。其中，S2V-DQN[76]是一种基于深度 Q-learning 的学习方法，该方法基于主动搜索的解码策略求解 TSPLIB 问题实例。对于表 3.4 中的 TSPLIB，使用由 TSP100 训练模型并通过两种解码策略进行测试。在评估各个实例的最佳路径时，采用贪婪策略的思想，使用从 100 个不同训练周期（epoch）得到的模型参数对每个实例分别求解，最终选择出各自得到的最短路径作为该实例的最优解。如表 3.4 所示，所提方法显著优于采用贪婪解码的 AM 方法以及使用主动搜索策略的 S2V-DQN 方法。在大多数测试中，改进的人工蜂群（ABC）算法[21]取得了最优结果（0.00%），显著优于本文所提出的方法（3.5%）。然而，ABC 算法的平均运行时间过长，远大于本文方法，对于需要快速响应的实时环境而言，本文的方法与其相比在解决方案质量和计算速度之间实现了一个有效的平衡。

表 3.4 GAT-NE 与其他方法在 TSPLIB 测试集上的比较结果

TSPLIB	Concorde	RL				元启发式
		Ours(Greedy)	AM(Greedy)[33]	Ours(Sample)	S2V-DQN(Active)[76]	ABC[21]
eil51	426	426	438	426	430	**427**
berlin52	7542	7876	8302	7792	7538	**7542**
st70	675	679	698	674	694	**675**
eil76	538	550	566	539	561	**538**
rat99	1211	1297	1335	1288	1272	**1211**
kroD100	21249	22633	23395	**21891**	22135	
rd100	7910	7937	8513	**7933**	8154	
lin105	14379	14925	16180	14847	15017	**14379**
pr107	44303	46404	52611	45656	**45114**	
pr124	59030	60462	62276	60902	61614	**59030**
bier127	118282	124335	128187	**120673**	121589	
ch130	6110	**6201**	6422	6209	6280	
pr144	58537	60832	64304	**59038**	59443	
ch150	6528	6817	6921	**6595**	6982	
kroA150	26524	28052	29764	**27802**	27889	
pr152	73682	77113	76975	75003	75279	**73682**
u159	42080	44269	49031	**43308**	45442	
平均差距	0.00%	**3.92%**	8.75%	**2.21%**	3.55%	0.00%

对于 CVRPLIB 中的小型实例和中型实例，分别使用由 CVRP50 和 CVRP100 随机实例训练的模型进行测试。表 3.5 和表 3.6 展示了 CVRPLIB 实例的最优解、AM[33] 模型的结果以及本文 GAT-NE 模型的结果。实验结果表明，在大部分情况下，GAT-NE 在减少车辆数量（路线数目）和总路线长度两方面均超越了 AM 模型。此外，将由 CVRP50 和 CVRP100 随机实例训练的模型泛化到 CVRPLIB 中小型实例中，表 3.5 和表 3.6 展示了 GAT-NE 模型在 CVRPLIB 实例上与其他方法的性能对比结果。在大部分测试实例中，本文所提方法在车辆数量（路线数）和总路径长度两个方面，均显示出比 AM 模型更优的表现。在 TSPLIB 及 CVRPLIB 基准测试数据集中，GAT-NE 的最优解差距分别为 2.21%、5.46%和 11.09%，验证了随机实例训练的 GAT-NE 模型泛化到真实世界实例的能力。

表 3.5 在 CVRPLIB 中小规模实例上的泛化能力比较结果

CVRPLIB	节点数	最优解		Ours(Greedy)			AM(Greedy)[33]		
		路线	长度	路线	长度	差距	路线	长度	差距
A-n34-k5	33	5	788	**5**	**828**	5.08%	6	869	10.28%
A-n37-k6	36	6	949	**6**	**997**	5.06%	**5**	998	5.16%
A-n39-k6	38	6	831	**5**	**849**	2.16%	**5**	851	2.41%
A-n44-k6	43	6	937	**6**	**972**	3.74%	7	**987**	5.34%
A-n45-k7	44	7	1146	**7**	**1185**	3.39%	7	1217	6.20%
A-n48-k7	47	7	1073	**7**	**1101**	2.61%	**7**	1176	9.60%
A-n63-k9	62	10	1616	10	1820	12.13%	**10**	**1811**	12.07%
A-n69-k9	68	9	1159	10	1217	4.97%	**9**	**1229**	6.04%
B-n45-k5	44	5	751	**7**	**799**	6.39%	**7**	825	9.85%
B-n50-k8	49	8	1312	**7**	1373	4.65%	**7**	**1329**	1.30%
B-n51-k7	50	7	1032	**8**	**1039**	0.67%	**8**	1167	13.08%
B-n57-k7	56	7	1153	**7**	**1228**	6.50%	**7**	1243	7.81%
B-n68-k9	67	9	1272	**9**	1337	5.11%	10	**1304**	2.52%
E-n31-k7	30	7	379	7	455	20.05%	**6**	**385**	1.58%
E-n51-k5	50	5	521	**6**	**542**	4.02%	**6**	551	5.76%
P-n50-k8	49	8	696	**9**	**725**	4.13%	**9**	729	4.74%
P-n55-k7	54	7	568	**7**	**598**	5.28%	**7**	629	10.74%
P-n60-k15	59	15	968	**15**	**996**	2.89%	16	1009	4.24%
P-n70-k10	69	10	827	**11**	**853**	3.14%	11	891	7.74%
平均差距						**5.46%**			6.83%

表 3.6 在 CVRPLIB 中大规模实例上的泛化能力比较结果

CVRPLIB	节点数	最优解		Ours(Greedy)			AM(Greedy)		
		路线	长度	路线	长度	差距	路线	长度	差距
E-n101-k14	100	14	1067	**14**	1169	9.56%	**14**	**1157**	8.43%
M-n101-k10	100	10	820	**10**	**875**	6.71%	**10**	884	7.80%
M-n121-k7	120	7	1034	8	**1137**	9.96%	**7**	1262	22.05%
M-n151-k12	150	12	1015	**12**	**1103**	8.67%	**12**	1124	10.74%
M-n200-k16	199	16	1274	**17**	**1421**	11.54%	**17**	1453	14.05%
P-n101-k4	100	4	681	**4**	**814**	19.53%	5	861	26.43%
X-n125-k30	124	30	55539	**30**	**59150**	6.50%	32	60842	9.55%
X-n157-k13	156	13	16876	**13**	20264	20.08%	14	**18059**	7.01%
X-n181-k23	180	23	25569	**23**	**28577**	11.76%	**23**	26936	5.35%
X-n223-k34	222	34	40437	37	43534	7.66%	34	43422	7.38%

续表　　表 3.6 在 CVRPLIB 中大规模实例上的泛化能力比较结果

CVRPLIB	节点数	最优解		Ours(Greedy)			AM(Greedy)		
		路线	长度	路线	长度	差距	路线	长度	差距
X-n266-k58	265	58	75478	**62**	**83097**	10.09%	**62**	85541	13.33%
X-n298-k31	297	31	34231	**32**	**38005**	11.03%	**32**	38757	13.22%
平均差距						**11.09%**			12.11%

3.4 本章小结

本章重点探索如何利用 GAT-NE 模型深入挖掘车辆路径问题的全局特性，以提升解决方案的质量。为此，研究了一种结合节点与边信息的深层图注意力编码器，并通过添加层间残差连接来防止深度模型中的梯度消失问题，确保模型奖励值的有效收敛。同时，基于注意力机制创建的上下文向量有助于捕捉问题的核心信息，从而提高模型性能。此外，将基于展开策略的改进的 REINFORCE 算法用于训练模型，增强算法的稳定性与效率。实验结果展示了深度强化学习模型在解决质量和泛化能力上的优势，同时在解决速度上也超过了传统的启发式方法。

第 4 章 基于路径重构策略的动作序列算法

为了在合理的运行时间内进一步获得大规模问题的满意解，本章提出基于路径重构策略的动作序列算法（Action Sequence Algorithm based on Path Reconstruction Strategy，AS-PRS），该算法采用大邻域搜索算法的框架训练模型进行推理求解，进一步优化了 NLNS 的模型架构。同时，将 AS-PRS 算法重构被破坏解决方案的过程形式化定义为一个马尔可夫决策过程，采用 REINFORCE 训练算法指导 Agent 选择要执行的最优动作，通过不断重复这一步骤，重构被破坏的解。此外，将第三章 GAT-NE 模型输出的解筛选形成一个多样化的可行解集作为 AS-PRS 算法的初始解，旨在在局部搜索算法的初始阶段提供一个平衡了优化与探索的解集。

4.1 整体求解方案

本章的整体的求解方案与第三章不同，如图 4.1 所示，采用 GAT-NE 增强 AS-PRS 算法的求解能力。

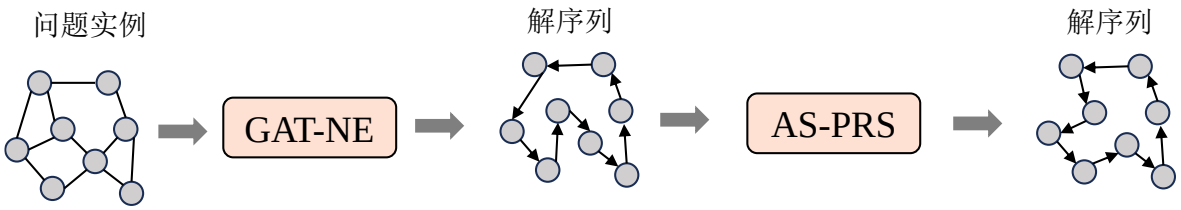

图 4.1 整体求解方案图

AS-PRS 算法是基于大邻域搜索（LNS）算法[78]的框架设计。LNS 算法的求解效果很大程度上依赖领域专家设计的破坏和修复算子的质量。其中，修复算子的设计通常需要采用复杂的启发式方法。为了探索模型不依赖额外领域知识的求解能力，引入一种基于 LNS 算法框架的深度强化学习算法，称之为带有路径重构策略的动作序列算法（Action Sequence Algorithm based on Path Reconstruction Strategy，AS-PRS）。在 LNS 修复被破坏的解决方案的每一步，AS-PRS 都会自动选择执行当前最优的动作实现解决方案的重构。与传统 LNS 算法的依赖领域知识的修复策略不同，AS-PRS 通过深度神经网络来指导修复过程，从而提升求解效率和解的质量。此外，LNS 作为一种局部搜索算法，其最终的迭代结果取决于初始解的选择，特别是处理大规模问题时尤为显著。局部搜索的解空间通常接近初始解，不合适的初始解不仅会增加计算量，也

可能使算法陷入局部最优解。因此，基于 GAT-NE 模型为 AS-PRS 提供一个多样化的初始解集，进一步增强 AS-PRS 探索解空间的能力。AS-PRS 算法整体的工作流程如下：

（1）初始解π的定义是求解过程的起点，初始解是基于 GAT-NE 预先筛选出的可行解集 M 中采用贪婪策略选取的。可行解集 M 由最优的m_{best}个解和与最优解差距最大的$m_{different}$个解构成，规模为$m = m_{best} + m_{different}$，确保了初始解的质量和多样性。

（2）采用破坏算子对初始解进行破坏操作，从而生成多个不完整的解决方案。为提高 AS-PRS 的多样性并适应不同的路径问题场景，采用了两种类型的破坏算子：

1）基于距离的移除[79]（Proximity-based Removal，PR）

该破坏算子的基本原理是，通过连续地去除与当前选定节点距离最短的节点，来形成新的路线，可以理解为是对 Shaw Removal 方法[53]的改进。在具体操作中，PR 算子首先随机挑选一个客户节点v_j，然后在每次迭代中，从未被选中的客户节点集合中移除距离v_j最近的节点，即设置$\mathcal{L} = \mathcal{L} \cup \{j\}$。并找出距离最小的节点$i^* = \min_{i \in N \setminus \mathcal{L}} \{||v_j - v_i||\}$，然后将$j$更新为$i^*$。这里的$\mathcal{L}$代表已移除节点的集合。

2）邻近点移除[79]（Node Neighborhood Removal，NNR）

NNR 算子旨在提升邻域搜索的多样性。首先 NNR 会随机选择一个客户节点，并在随后的迭代过程中，逐步移除围绕该节点的若干邻近节点。在基于网格分布识别邻近节点的情况下，如果一个网格内的邻近节点数未达到设定阈值，那么就会增加网格大小的一定比例，以便纳入更多邻近节点以进行考察。

（3）重构这些被破坏的解。不同于传统的启发式修复方法，AS-PRS 算法的修复过程由训练过的深度学习模型指导，允许算法以更高的准确率和效率重建优化的路径。通过将路径重构策略集成到 LNS 框架中，AS-PRS 不仅增强了原始 LNS 算法的求解能力，也降低了对领域知识的依赖。

4.2 AS-PRS 算法

4.2.1 算法的整体结构与 NLNS 的对比

与 NLNS 相同，AS-PRS 算法的模型框架的输入为被破坏路线的端点以及当前的参考起点，都需要处理关于 VRP 问题的一系列节点信息。它们均基于大邻域搜索算

法，旨在预测并选择下一个访问的节点，但在构建解决方案的方法上存在差异。首先，它们采用不同的技术框架。NLNS 主要基于指针网络且没有明确的编码器-解码器结构，AS-PRS 算法的模型使用编码器-解码器架构，其中编码器采用去除位置编码的 Transformer 编码器并加入批量归一化层，提升了对节点特征的理解和信息捕获能力。而 NLNS 模型则直接处理节点特征而没有经过复杂的前置处理。此外，AS-PRS 将选择节点进行路径重构的过程建模为马尔可夫决策过程（Markov Decision Process，MDP），有助于 REINFORCE 算法指导模型选择最优动作。最后，为 AS-PRS 提供多样化的初始解集增强其求解能力。

4.2.2 AS-PRS 算法的模型结构

图 4.2 展示了一个被破坏后的解决方案。该解决方案由四条不完整的路线组成，其中一条路线仅包含一个节点。每一条未与仓库节点x_0相连的路线的端点$(x_1, \ldots, x_5)$、仓库节点x_0，以及一个参考节点x_3将作为 AS-PRS 算法模型的输入。其中，每个节点的特征向量包含本身坐标的信息和相关的需求信息。模型的目标是确定参考节点x_3所连接的下一个节点。

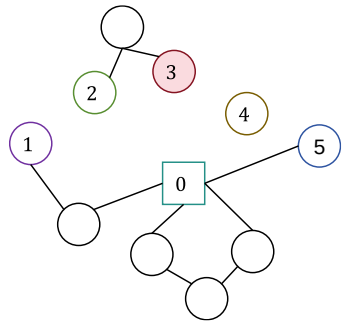

图 4.2 被破坏的解决方案

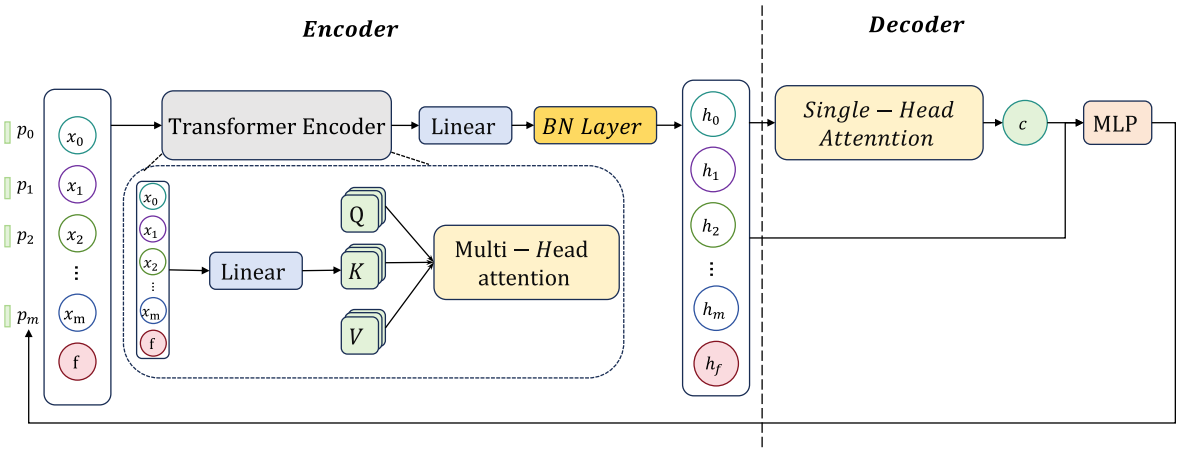

图 4.3 AS-PRS 算法的模型架构细节示意图

AS-PRS 算法的模型架构如图 4.3 所示。在每个时间步t，模型以(X_t, f_t)作为输入，其中X_t代表一个不完整的解决方案，f_t为当前的参考节点。

首先，模型对输入X中的每个元素x_i进行编码，生成嵌入向量h_i。这一编码过程由 Transformer、线性层和批量归一化层组成。参考节点f的嵌入h_f使用与上述编码结构相似但参数不同的模块生成。随后，通过解码器中的单头注意力层及参考节点h_f来计算每个输入$x_0, \dots, x_m$的相关性，并生成对应的上下文向量c，具体的计算步骤如下：使用h_f和每个输入x_i嵌入h_i计算对齐向量$\bar{a}$。对齐向量的计算涉及到h_i和h_f的连接，并通过一个可训练的参数矩阵W^A和向量z^A进行变换，然后应用$tanh$激活函数，最后通过$softmax$函数获得每个h_i对应的权重$\bar{a}$。

$$\bar{a} = softmax(u_0^H, \dots, u_m^H) \cdots\cdots (4.1)$$

$$u_i^H = z^A \, tanh(W^A \, [h_i \, || h^f \,]) \cdots\cdots (4.2)$$

最后，将获得的对齐向量$\bar{a}$通过加权求和的方式得到上下文向量c，即每个h_i乘以其对应的权重$\bar{a}$，然后对m个h_i进行求和。

$$c = \sum_{i=0}^{n} \bar{a}_i h_i \cdots\cdots (4.3)$$

上下文向量c包含了与给定任务相关的输入信息$x_i \in X_t$，并与参考节点的嵌入h_f拼接起来，随后被输入到一个全连接的前馈网络，该网络使用两层$ReLU$激活函数，并输出一个d_h维的向量q。将q与每个嵌入$h_0, \dots, h_m$结合，计算所有动作的输出分布。

$$p_\theta(a_t | \pi_t) = softmax(u_0, \dots, u_m) \cdots\cdots (4.4)$$

$$u_i = z^B \tanh(h_i + q) \cdots\cdots (4.5)$$

4.2.3 基于 MDP 的路径重构策略

本文基于基线的 REINFORCE 算法来训练 AS-PRS 算法的模型，将 AS-PRS 重构被破坏解决方案的过程形式化定义为一个马尔可夫决策过程（MDP）。在每个时间步t中，Agent（学习模型）与 Environment（不完整的解决方案）相互作用，直到重新构造出满足所有节点需求的完整解。同时，使用与第三章相同的掩码机制，避免选择不可行的动作。具体情况如下：

状态：在每个时间步t，状态S描述了当前的不完整解决方案$\pi_t = (X_t, f_t)$。X_t为所有不完整路线的末端节点信息，每个元素节点x是一个四维特征向量(x^X, x^Y, x^D, x^S)。其中，x^X和x^Y是该节点的x和y坐标，x^D为该行程的满足需求之和。如果x为仓库节点

v_0，则将x^D和x^S设置为-1。在输入到网络之前，除了始终为-1 的仓库的x^D值外，x^X，x^Y和x^D的值被规范化到[0,1]范围内。参考节点f_t在每个时间步骤是动态选择的，依赖于前一步骤的操作结果。$t=0$时，参考节点f_t从集合$X_t\backslash\{x_0\}$中随机选取；$t>0$时，如果与f_{t-1}相连的路线仍然不完整，则f_t设置为该路线末端节点；否则，f_t将再次通过随机选择来确定。

动作：在当前时间步t，每个动作a_t涉及从X_t中选择一个节点与参考输入f_t相连，并产生新的不完整的解决方案π_{t+1}。

奖励：当前时间步t，修复的解决方案π_t的总路线长度与π_0的总路线长度的差值。

策略：策略是模型根据当前状态选择动作的规则。本文将策略定义为模型通过学习得到的动作概率分布$p_\theta(a_t|\pi_t)$。

终止条件：重复此过程，直到构造出一个满足所有节点的需求的完整解。

4.2.4 基于基线的 REINFORCE 训练算法

本文基于基线的 REINFORCE 算法训练 AS-PRS 算法的模型来重构被破坏算子o^D破坏的解。为评估模型选取某个动作的效果，定义了一个损失函数。设初始被破坏的解决方案为π_0，经过$Agent$采取一系列动作$a_0,\dots,a_{T-1}$修复后的解决方案为π_T。损失函数$L(\pi_T|\pi_0)$表示π_T相对于π_0之间的差异，即二者的总行程长度之差。$Agent$训练的目标是调整模型的参数θ，以最小化修复π_0的预期损失。

$$J(\theta|\pi_0)=E_{\pi_T\sim p_\theta}(\cdot\,|\pi_0)L(\pi_T|\pi_0) \cdots\cdots (4.6)$$

其中，产生解π_T的概率可分解为：

$$p_\theta(\pi_T|\pi_0)=\prod_{t=0}^{T-1}p_\theta(a_t|\pi_t) \cdots\cdots (4.7)$$

使用基于基线的 REINFORCE 算法来估计梯度，计算方式如下：

$$\nabla J(\theta|\pi_0)=E_{\pi_T\sim p_\theta(\cdot|\pi_0)}\left[\left(L(\pi_0,\pi_T)-b(\pi_0)\right)\right]\nabla_\theta log p_\theta(\pi_T|\pi_0) \cdots\cdots (4.8)$$

为了评估模型采取特定行动的有效性，引入了基线网络$b(\pi_0)$来估算修复一个破坏后解决方案π_0的成本$J(\theta|\pi_0)$，$b(\pi_0)$在训练过程中为模型提供一个稳定的参考标准，有助于稳定训练过程。基线网络$b(\pi_0)$不是独立运行，而是与 AS-PRS（策略网络）交替进行训练的，以最小化预测的修复π_0的成本与实际成本之间的均方误差。具体而言，基线网络$b(\pi_0)$以最近学习的策略产生的不完整解序列作为输入。$b(\pi_0)$对每个输入应

用一个能够识别其位置信息的前馈网络，产生一个连续的估计值。将这些连续的估计值相加，就得到了整个修复成本的估计值$b(\pi_0)$。

4.2.5 搜索策略

AS-PRS 算法的搜索策略如算法 4.1 所示。在搜索开始时，当前最优解π^*被初始化为已知的一个可行解（从可行解集 M 中采用贪心策略选取），同时为算法设置初始温度T_s用于控制搜索过程中的探索深度。搜索主体由两个嵌套的$while$循环组成。外层循环负责在每次迭代中生成一个包含当前最优解副本的解集合B，模拟多个解的并行搜索。内层循环通过降低温度T来逐步改进集合B中的解。与 NLNS 相同，本文使用模拟退火的策略逐渐减少接受较差解的概率。当外层循环结束时，温度T会重新设置为温度T_r，使得算法在后续的迭代中跳出局部最优解。

算法 4.1：$AS-PRS$算法的搜索策略

输入：一个可行解集合 M, 起始温度T_s, 再加热温度T_r, 最低温度T_m, 冷却计划δ, 重置为当前解的百分比Z

输出：最佳可行解π^*

```
1:function: AS − PRS(M, T_s, T_r, T_m, δ, Z)
2:      π* ← get_best(M)
3:      T ← T_s
4:      while search termination criteria not reached do
5:          B ← {π*, …, π*}
6:          while T > T_m do
7:              B ← Repair(Deatory(B)) /*对批次B 中的解施加破坏和修复操作，以改进解决方案*/
8:              π^b ← argmin_{π∈B} {Cost(π)}
9:              if Accept(π^b, π*, T) then
10:                 π ← π^b
11:             if Cost(π^b) < Cost(π*) then
12:                 π* ← π^b
13:             Set the first Z% of elements in B to π
14:             T ← Update(T, δ)
15:         T ← T_r
16:     return π*
```

在每次内层循环中，集合B中的所有解首先经过破坏和随后的修复操作。之后，从B中选出最优解π^b，并依据模拟退火算法的 Metropolis 接受准则来评估解π^b的质量，从而决定是否接受它作为新的当前解π。如果π^b优于π^*，那么采用π^b替换掉可行集 B 中最差的解。最后，更新温度T，B 中前Z%的元素被设置当前解π。参数 Z 控制了搜

索的集中度，即在多大程度上集中探索生成当前解π的邻近解。而集合 B 中的后（1−Z）%的解仍然保留，以增加解的多样性。通过并行运行多个内循环实例，并在每轮迭代后异步交换当前最优解，算法进一步提升了搜索的并行性和效率。

综上所述，本文描述的搜索策略通过结合贪心策略的初始解选择、模拟退火的温度调控以及批次处理的方法，旨在高效地探索和优化解空间，同时并行处理提高了计算速度。

4.3 实验及结果分析

4.3.1 实验数据及环境设置

本文在规模为 20、50、100 的随机数据集以及 CVRPLIB 公共基准测试数据上对车辆路径问题进行了实验。将 CVRPLIB 中具有相同特征的实例聚类为 9 组数据集，每组 20 个实例，组内所有实例服从相同分布，所有测试集仅在测试过程中使用，该聚类数据集来自 NLNS 方法。本章所有训练以及测试过程均在 Nvidia GeForce RTX3090 GPU 上执行。模型基于 PyTorch 构建，代码使用 Python 3.8 编写。实验设置如表 4.1 所示。

表 4.1 模型的超参数设置

超参数	数值	超参数	数值
toRemove	$3(m=20)$ $6(m=50)$ $10(m=100)$	Iter	$150(m=20)$ $200(m=50)$ $250(m=100)$
REINFORCE 学习率	$1\times10^{-3}(m=20)$ $3\times10^{-4}(m=50,100)$	再加热次数	$50(m=20)$ $200(m=50)$ $250(m=100)$
编码器层数	4	Z	0.8
节点嵌入网络的维度	128	批处理大小	300

本文将 AS-PRS 算法与构造算法、LKH3 求解器以及基于学习的方法进行比较。其中构造算法包括随机插入、最近插入和最近邻算法。以 LKH3 求解器作为基准方法进行比较，LKH3 是一个基于启发式算法的求解器，用于近似求解 CVRP 问题，它能够有效地产生接近最优的解决方案，特别是对于大规模实例，从而在计算时间和解的质量之间达到平衡。

在 AS-PRS 算法中，破坏算子所需移除的客户节点数（toRemove）和算法最大迭

代次数（Iter）随问题规模进行动态调整，确保随着问题规模的扩大，算法能够通过增加待删除节点的数量和扩展迭代次数来适应更广泛的搜索空间，从而维持解决方案的优化和搜索效率。

算法中的起始温度T_S和再加热温度T_r均根据可行解集 M 中解的路径长度分布动态设定，即依据解集中路径长度的四分位数进行调整。此方法使得温度参数能够与 M 集合内当前的路径长度分布相适应，进而更有效地管理搜索过程中解的接受标准。同时，冷却策略采用指数下降，确保温度最终平稳降至 1。在算法的迭代过程中，温度T的下降不遵循线性模式，而是采用指数方式减小，旨在缓慢降低初期温度，使算法广泛探索不同的解空间。随着迭代的深入，温度的快速降低促使算法逐步减少新解的探索，聚焦在当前找到的最佳解决方案上。当问题的节点数量不超过 200 时，设定再加热过程为五次；而当节点数量超过 200 时，则设定再加热次数为十次，以便算法可以跳出局部最优解重新开始探索过程。关于运行时间的比较较为复杂，因此在评估提出方法的总体性能时，本研究不考虑模型训练所需的时间。

4.3.2 AS-PRS 与其他方法的性能比较

如表 4.2 所示，将 AS-PRS 获得的实验结果与 LKH3 求解器、LNS 启发式算法以及基于学习的 NLNS 方法进行对比，并以 LKH3 作为基准方法。

表 4.2 AS-PRS 与优化方法进行比较

CVRPLIB	节点	LKH3	Ours		NLNS[51]		LNS[56]	
			长度	差距	长度	差距	长度	差距
XE_1	100	30785	30210	-1.87%	30243	-1.76%	30414	-0.12%
XE_2	128	33620	33572	-0.14%	33584	-0.11%	34122	1.49%
XE_3	161	13984	13890	-0.67%	13977	-0.15%	15002	7.28%
XE_4	180	26604	26720	-0.44%	26716	-0.42%	27246	2.41%
XE_5	203	20768	20545	-1.07%	20593	-0.84%	21038	1.30%
XE_6	236	27731	27627	-0.38%	27694	-0.13%	30132	8.66%
XE_7	241	79626	79402	-0.28%	79413	-0.27%	80383	0.95%
XE_8	269	34521	34256	-0.77%	34272	-0.72%	35256	2.13%
XE_9	297	37031	37058	0.07%	37096	0.18%	39324	6.19%
平均差距		0.00%		-0.52%		-0.11%		3.25%

表 4.2 列出了不同大小的实例测试集（标记为 XE_1 至 XE_9，问题规模逐渐增

加）。对于每个测试集，比较了每种方法的性能，评价指标包括解决方案的平均质量和与 LKH3 求解器的差距（越低越好）。在绝大多数测试实例中，AS-PRS 不仅达到了 LKH3 的性能，甚至在大多数情况下超越了 LKH3 的解决方案质量。尤其是在 XE_5 和 XE_8 中，AS-PRS 的最优解差距分别为-1.07%和-0.77%，这证明了 AS-PRS 在处理复杂路径问题时的强大能力。此外，在多个数据集上，AS-PRS 相比 NLNS 展示出更小的最优差距，这表明 AS-PRS 能够生成更接近最优解的解决方案。证实 AS-PRS 在模型架构、节点特征处理以及求解策略上的改进，是有效且有价值的。相比之下，传统的 LNS 虽然在一些情况下能够提供可接受的解决方案，但求解质量通常不如 AS-PRS 和 NLNS。

4.3.3 两阶段框架的有效性验证

如 4.1 节所述，AS-PRS 算法的初始解由 GAT-NE 模型提供，为验证这种两阶段求解的有效性，本文在解的质量和计算时间两个方面进行了实验评估。将 AS-PRS 与不同的基线方法的求解性能进行比较，将基线方法生成的解序列作为 AS-PRS 的初始解，以此来评估所提出的两阶段框架的效率。这些方法包括构造方法（随机插入、最近插入、最近邻法）以及基于学习方法的 AM。

表 4.3 两阶段框架有效性的对比结果

方法	CVRP20			CVRP50			CVRP100		
	长度	差距	时间	长度	差距	时间	长度	差距	时间
GAT-NE(Greedy)	6.26	5.56%	2s	10.80	7.36%	7s	16.69	7.33%	17s
最远插入法	6.25	5.40%	83s	11.11	10.44%	327s	17.42	12.03%	3205s
最近插值法	6.23	5.06%	74s	10.83	7.65%	242s	17.33	11.45%	2512s
随机插值法	6.09	2.70%	26s	10.49	4.27%	148s	16.21	4.24%	1882s
AM(Greedy)	5.96	0.51%	25s	10.17	1.09%	146s	15.70	0.96%	1848s
AS-PRS	5.93	0.00%	23s	10.06	0.00%	147s	15.55	0.00%	1851s

表 4.3 展示了采用不同初始解的 AS-PRS 的平均路线长度、最优解差距和总计算时间。可以观察到，AS-PRS 在解决方案质量和计算时间上都显著优于其他非学习算法增强的结果。以 CVRP100 为例，基于学习的方法模型增强的结果平均解决方案质量达到 15.55，平均求解时间为 1851 秒。而由传统启发式方法增强的最佳解决方案平均长度为 16.21，求解时间为 1882 秒。此外，比较了 AM 和 GAT-NE 模型在增强 AS-

PRS 求解质量和计算时间方面的差异。在考虑解决方案质量时，本文的两阶段模型在各种规模的 VRP 问题上明显优于其他的组合方法。这一结果证明了 GAT-NE 模型能够在提升 AS-PRS 搜索方法的解决方案质量和计算效率方面发挥了一定的作用。

4.4 本章小结

本章主要介绍了如何通过基于策略梯度的 REINFORCE 算法训练 AS-PRS 学习 LNS 启发式算法，详细介绍了该新型 LNS 算法的工作过程。此外，将第三章的 GAT-NE 模型与 AS-PRS 结合构成一个两阶段框架。GAT-NE 能够找到强的初始解，这有利于 AS-PRS 进行局部搜索，以提高解的质量获得近最优解。在随机生成的测试数据集以及 CVRPLIB 的聚类数据集上，将本文提出的 AS-PRS 和两阶段求解模型与当前先进的 DRL 方法和传统的启发式算法进行对比，验证本文方法的有效性。

第 5 章 总结与展望

5.1 工作总结

车辆路径问题作为研究最为广泛的组合优化问题之一，与交通运输和社会生活密切相关。由于 VRP 的传统求解方法存在求解时间过长、领域知识依赖过重的问题，众多学者转向深度学习方法的研究，为 VRP 求解开辟崭新的方向。通过数据驱动的神经网络对问题求解过程进行学习，使其在面对其他同分布的问题实例时，能够利用已学到的知识快速求解。虽然基于深度强化学习的 VRP 求解方法具有无需标签数据、泛化能力强等优势，但这些方法仍处于探索阶段，在求解质量和求解时间方面仍有待进一步优化。本文主要的研究工作如下：

（1）为了解决现有 DRL 方法求解 VRP 问题的不足，提出联合学习节点和边特征的图注意力模型（GAT-NE）。通过设计注意力模块对节点和边的关系建模，并采用边的特征对节点的嵌入进行更新，高效地捕获问题的全局特征。为了增强模型训练的稳定性和效率，提出采用基于 Actor-Critic 架构改进的 REINFORCE 算法。相较于现有的其他训练方法，该方法通过冻结策略网络的参数降低训练过程中的方差，进而稳定训练并提高模型性能。在随机生成的数据集和公共基准数据集上的实验结果验证了本文 GAT-NE 模型的有效性。

（2）由于 GAT-NE 方法面对大规模车辆路径问题难以训练，因此该类方法在面对较大规模实例时求解质量均存在明显下降的问题。为了解决上述问题，提出了混合式求解方法，首先设计了一个基于大邻域搜索算法的深度强化学习算法（AS-PRS），通过 AS-PRS 算法的路径重构策略改进传统 LNS 算法的修复算子，减少对领域知识的依赖。其次，提出采用 GAT-NE 模型来增强 AS-PRS 算法，通过 GAT-NE 模型为 AS-PRS 算法提供多样化的初始解，有助于更好的探索解空间，提高解决方案的质量。在随机生成的数据集和公共基准数据集上的实验结果验证了本文 AS-PRS 算法与经 GAT-NE 模型增强的 AS-PRS 算法的有效性。

5.2 未来展望

本文基于深度强化学习和局部搜索技术建立了求解车辆路径问题的框架，得到较好的效果，然而仍存在以下不足之处，对此提出以下几点展望：

（1）尽管深度强化学习在求解车辆路径问题方面有所提升，但也暴露出一些局限性。深度强化学习训练的模型需要耗费大量的时间，这对于追求效率的实际应用有一定挑战。考虑到深度学习领域的快速发展，探索新的网络架构和训练策略，如轻量级网络或迁移学习，可能会进一步降低训练时间并提升模型的泛化能力。

（2）本文的局部搜索模块仍然需要使用依赖领域知识的破坏算子来提升解决方案的质量，这限制了算法自主学习和适应新问题的能力。在未来工作中，计划采用神经网络设计一种新型的 LNS 破坏算子，该算子能够自动识别并选择需要破坏的节点位置，从而提高求解效率和解决方案的质量。

（3）原始 VRP 问题的结构不随时间变化，即客户的位置和客户的需求是固定的。然而，大多数实际场景存在很多不可预测的因素，动态环境中的路径规划为 VRP 的研究带来了挑战，因此需进一步加强对动态车辆路径问题的研究，尽可能多地考虑多种动态因素及其之间的相互关系。

参考文献

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

[2] Bullo F, Frazzoli E, Pavone M, et al. Dynamic vehicle routing for robotic systems[J]. Proceedings of the IEEE, 2011, 99(9): 1482-1504.

[3] Ralphs T K, Kopman L, Pulleyblank W R, et al. On the capacitated vehicle routing problem[J]. Mathematical programming, 2003, 94: 343-359.

[4] Kumar S N, Panneerselvam R. A survey on the vehicle routing problem and its variants[J]. 2012.

[5] Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of go without human knowledge[J]. nature, 2017, 550(7676): 354-359.

[6] Tian Y, Ma J, Gong Q, et al. Elf opengo: An analysis and open reimplementation of alphazero[C].International conference on machine learning. PMLR, 2019: 6244-6253.

[7] Kung T H, Cheatham M, Medenilla A, et al. Performance of ChatGPT on USMLE: Potential for AI-assisted medical education using large language models[J]. PLoS digital health, 2023, 2(2): e0000198.

[8] Hu H, Zhang Y, Wei J, et al. Alibaba vehicle routing algorithms enable rapid pick and delivery[J]. INFORMS Journal on Applied Analytics, 2022, 52(1): 27-41.

[9] Beam C M, Milne R J. Introduction: 2021 Franz Edelman Award for Achievement in Advanced Analytics, Operations Research, and Management Science[J]. INFORMS Journal on Applied Analytics, 2022, 52(1): 4-7.

[10] Beam C M, Pekgün P. Introduction: 2022 Franz Edelman Award for Achievement in Advanced Analytics, Operations Research, and Management Science[J]. INFORMS Journal on Applied Analytics, 2023, 53(1): 5-8.

[11] Papadimitriou C H, Steiglitz K. Combinatorial optimization: algorithms and complexity[M]. Courier Corporation, 1998.

[12] Marchand H, Martin A, Weismantel R, et al. Cutting planes in integer and mixed integer programming[J]. Discrete Applied Mathematics, 2002, 123(1-3): 397-446.

[13] Bertsekas D. Dynamic programming and optimal control: Volume I[M]. Athena scientific, 2012.

[14] Ralphs T K, Kopman L, Pulleyblank W R, et al. On the capacitated vehicle routing problem[J]. Mathematical programming, 2003, 94: 343-359.

[15] Freisleben B, Merz P. A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems[C].Proceedings of IEEE international conference on evolutionary computation. IEEE, 1996: 616-621.

[16] Hore S, Chatterjee A, Dewanji A. Improving variable neighborhood search to solve the traveling salesman problem[J]. Applied Soft Computing, 2018, 68: 83-91.

[17] Helsgaun K. An effective implementation of the Lin–Kernighan traveling salesman heuristic[J]. European journal of operational research, 2000, 126(1): 106-130.

[18] Pisinger D, Ropke S. A general heuristic for vehicle routing problems[J]. Computers & operations research, 2007, 34(8): 2403-2435.

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

[20] Lin S W, Lee Z J, Ying K C, et al. Applying hybrid meta-heuristics for capacitated vehicle routing problem[J]. Expert Systems with Applications, 2009, 36(2): 1505-1512.

[21] Szeto W Y, Wu Y, Ho S C. An artificial bee colony algorithm for the capacitated vehicle routing problem[J]. European Journal of Operational Research, 2011, 215(1): 126-135.

[22] Xin L, Song W, Cao Z, et al. Multi-decoder attention model with embedding glimpse for solving vehicle routing problems[C].Proceedings of the AAAI Conference on Artificial Intelligence. 2021, 35(13): 12042-12049.

[23] David A, Robert B, Vašek Chvátal, et al. Concorde TSP solver, 2006. https://www.math.uwaterloo.ca/tsp/concorde.html

[24] Perron L, Furnon V. Google's OR-Tools, 2019. https://Developers.google.com/optimization.

[25] Helsgaun K. An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems[J]. Roskilde: Roskilde University, 2017, 12.

[26] 接中冰. 基于深度强化学习的覆盖旅行商问题求解研究[D].无锡:江南大学,2022.

[27] Kai-Wen L, Tao Z, Rui W, et al. Research reviews of combinatorial optimization methods based on deep reinforcement learning[J]. Acta Automatica Sinica, 2021, 47(11): 2521-2537.

[28] Vinyals O, Fortunato M, Jaitly N. Pointer networks[J]. Advances in neural information processing systems, 2015, 28.

[29] Joshi C K, Laurent T, Bresson X. An efficient graph convolutional network technique for the travelling salesman problem[J]. arXiv preprint arXiv:1906.01227, 2019.

[30] Groshev E, Goldstein M, Tamar A, et al. Learning generalized reactive policies using deep neural networks[C].Proceedings of the International Conference on Automated Planning and Scheduling. 2018, 28: 408-416.

[31] Prates M, Avelar P H C, Lemos H, et al. Learning to solve np-complete problems: A graph neural network for decision tsp[C].Proceedings of the AAAI Conference on Artificial Intelligence. 2019, 33(01): 4731-4738.

[32] Sultana N, Chan J, Sarwar T, et al. Learning to optimise general TSP instances[J]. International Journal of Machine Learning and Cybernetics, 2022, 13(8): 2213-2228.

[33] Kool W, Van Hoof H, Welling M. Attention, learn to solve routing problems![J]. arXiv preprint arXiv:1803.08475, 2018.

[34] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.

[35] Kwon Y D, Choo J, Kim B, et al. Pomo: Policy optimization with multiple optima for reinforcement learning[J]. Advances in Neural Information Processing Systems, 2020, 33: 21188-21198.

[36] Nazari M, Oroojlooy A, Snyder L, et al. Reinforcement learning for solving the vehicle routing

problem[J]. Advances in neural information processing systems, 2018, 31.

[37] Xin L, Song W, Cao Z, et al. Multi-decoder attention model with embedding glimpse for solving vehicle routing problems[C].Proceedings of the AAAI Conference on Artificial Intelligence. 2021, 35(13): 12042-12049.

[38] Bello I, Pham H, Le Q V, et al. Neural combinatorial optimization with reinforcement learning[J]. arXiv preprint arXiv:1611.09940, 2016.

[39] Hottung A, Kwon Y D, Tierney K. Efficient active search for combinatorial optimization problems[J]. arXiv preprint arXiv:2106.05126, 2021.

[40] Kwon Y D, Choo J, Yoon I, et al. Matrix encoding networks for neural combinatorial optimization[J]. Advances in Neural Information Processing Systems, 2021, 34: 5138-5149.

[41] Pan X, Jin Y, Ding Y, et al. H-TSP: Hierarchically Solving the Large-Scale Travelling Salesman Problem[J]. arXiv preprint arXiv:2304.09395, 2023.

[42] 董金虎.基于深度强化学习的车辆路径规划算法研究[D].长春:吉林大学,2022.

[43] Chen X, Tian Y. Learning to perform local rewriting for combinatorial optimization[J]. Advances in Neural Information Processing Systems, 2019, 32.

[44] Lu H, Zhang X, Yang S. A learning-based iterative method for solving vehicle routing problems[C]//International conference on learning representations. 2019.

[45] Bengio Y, Lodi A, Prouvost A. Machine learning for combinatorial optimization: a methodological tour d'horizon[J]. European Journal of Operational Research, 2021, 290(2): 405-421.

[46] Calvet Liñán L, Juan Pérez Á A, Serrat Piè C, et al. A statistical learning based approach for parameter fine-tuning of metaheuristics[J]. SORT: statistics and operations research transactions, 2016, 40(1): 201-224.

[47] Cooray P, Rupasinghe T D. Machine learning-based parameter tuned genetic algorithm for energy minimizing vehicle routing problem[J]. Journal of Industrial Engineering, 2017, 2017.

[48] Žunić E, Đonko D, Buza E. An adaptive data-driven approach to solve real-world vehicle routing problems in logistics[J]. Complexity, 2020, 2020: 1-24.

[49] d O Costa P R, Rhuggenaath J, Zhang Y, et al. Learning 2-opt heuristics for the traveling salesman problem via deep reinforcement learning[C]//Asian conference on machine learning. PMLR, 2020: 465-480.

[50] Chen M, Gao L, Chen Q, et al. Dynamic partial removal: A neural network heuristic for large neighborhood search[J]. arXiv preprint arXiv:2005.09330, 2020.

[51] Hottung A, Tierney K. Neural large neighborhood search for routing problems[J]. Artificial Intelligence, 2022, 313: 103786.

[52] Jingwen L. Deep Reinforcement Learning for Solving Vehicle Routing Problems[D]. National University of Singapore (Singapore), 2022.

[53] Reinelt G. TSPLIB—A traveling salesman problem library[J]. ORSA journal on computing, 1991, 3(4): 376-384.

[54] Uchoa E, Pecin D, Pessoa A, et al. New benchmark instances for the capacitated vehicle routing

problem[J]. European Journal of Operational Research, 2017, 257(3): 845-858.

[55] Chu Y, Luo C, Hoos H H, et al. Improving the performance of stochastic local search for maximum vertex weight clique problem using programming by optimization[J]. Expert Systems with Applications, 2023, 213: 118913.

[56] SHAW P 1998. Using constraint programming and local search methods to solve vehicle routing problems[C] // In: Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming. New York: Springer, 417–431.

[57] Miao Q, Zheng W, Lv Y, et al. DAO to HANOI via DeSci: AI paradigm shifts from AlphaGo to ChatGPT[J]. IEEE/CAA Journal of Automatica Sinica, 2023, 10(4): 877-897.

[58] Zong Y, Lin L, Wang S, et al. Improvements and Challenges in StarCraft II Macro-Management A Study on the MSC Dataset[J]. Journal of Theory and Practice of Engineering Science, 2023, 3(12): 29-35.

[59] He J, Zhao H, Zhou D, et al. Nearly minimax optimal reinforcement learning for linear markov decision processes[C]//International Conference on Machine Learning. PMLR, 2023: 12790-12822.

[60] Silver D, Hubert T, Schrittwieser J, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play[J]. Science, 2018, 362(6419): 1140-1144.

[61] Yan Y, Li G, Chen Y, et al. The efficacy of pessimism in asynchronous Q-learning[J]. IEEE Transactions on Information Theory, 2023.

[62] Sethi V, Pal S. FedDOVe: A Federated Deep Q-learning-based Offloading for Vehicular fog computing[J]. Future Generation Computer Systems, 2023, 141: 96-105.

[63] Chen S Y C. Quantum deep recurrent reinforcement learning[C]//ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2023: 1-5.

[64] Zhang J, Kim J, O'Donoghue B, et al. Sample efficient reinforcement learning with REINFORCE[C]//Proceedings of the AAAI conference on artificial intelligence. 2021, 35(12): 10887-10895.

[65] Zhong H, Zhang T. A theoretical analysis of optimistic proximal policy optimization in linear markov decision processes[J]. Advances in Neural Information Processing Systems, 2024, 36.

[66] Kumar H, Koppel A, Ribeiro A. On the sample complexity of actor-critic method for reinforcement learning with function approximation[J]. Machine Learning, 2023: 1-35.

[67] Gori M, Monfardini G, Scarselli F. A new model for learning in graph domains[C]//Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005. IEEE, 2005, 2: 729-734.

[68] Battaglia P W, Hamrick J B, Bapst V, et al. Relational inductive biases, deep learning, and graph networks[J]. arXiv preprint arXiv:1806.01261, 2018.

[69] Bahdanau D, Cho K H, Bengio Y. Neural machine translation by jointly learning to align and translate[J]. 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings, 2015: 1-15.

[70] Chen S, Yao T, Jiang Y G. Deep Learning for Video Captioning: A Review[C]. IJCAI, 2019(1):1-20.

[71] Ramesh A, Pavlov M, Goh G, et al. Zero-shot text-to-image generation[C]. International Conference

on Machine Learning. 2021: 8821-8831.

[72] Tan X, Qin T, Soong F, et al. A survey on neural speech synthesis[J]. arXiv preprint arXiv:2106.15561, 2021.

[73] Veličković P, Cucurull G, Casanova A, et al. Graph attention networks[J]. arXiv preprint arXiv:1710.10903, 2017.

[74] Bello I, Pham H, Le Q V, et al. Neural combinatorial optimization with reinforcement learning[J]. arXiv preprint arXiv:1611.09940, 2016.

[75] Engstrom L, Ilyas A, Santurkar S, et al. Implementation matters in deep policy gradients: A case study on ppo and trpo[J]. arXiv preprint arXiv:2005.12729, 2020.

[76] Khalil E, Dai H, Zhang Y, et al. Learning combinatorial optimization algorithms over graphs[J]. Advances in neural information processing systems, 2017, 30.

[77] Khan I, Maiti M K. A swap sequence based artificial bee colony algorithm for traveling salesman problem[J]. Swarm and evolutionary computation, 2019, 44: 428-438.

[78] Pisinger D, Ropke S. Large neighborhood search[J]. Handbook of metaheuristics, 2019: 99-127.

[79] 赵磊.卡车与无人机两级配送网络鲁棒优化模型及算法研究[D].长春:吉林大学,2022.