基于双演员-评论家的大型邻域搜索新型自适应机制

余少华 a，毛文浩[b]，吴子高*[b]，Jakob Puchinger[c]，[d]

[a]南京理工大学智能科学与技术学院，中国南京 *210094*[b]华北电力大学机械工程系，中国保定 *071003*[c]诺曼底高等商学院 *Métis* 实验室，法国克利希 *92110*[d]巴黎萨克雷大学中央理工-高等电力学院工业工程实验室，法国吉夫叙尔伊韦特 *91190*

arXiv:2601.11414v1 [cs.GT] 16 Jan 2026

摘要

自适应大邻域搜索（ALNS）是一种广泛应用于组合优化问题的启发式方法。ALNS 通过迭代使用破坏与修复算子来探索解空间，并根据自适应机制调整各算子的使用概率，以寻找最优解。然而，经典 ALNS 的自适应机制在选择算子时未考虑破坏算子与修复算子之间的相互作用。为克服这一局限，本研究提出了一种新颖的自适应机制。该机制通过双演员-评论家（DAC）模型增强算法的适应性，并充分考虑新解的质量由破坏算子和修复算子共同决定这一事实。它在权重调整过程中有效利用这两类算子之间的相互作用，大幅提升了 ALNS 算法的适应性。在该机制中，破坏过程和修复过程被建模为相互独立的马尔可夫决策过程，从而更准确地引导算子选择。此外，我们使用图神经网络从问题实例中提取关键特征，并进行有效的聚合与归一化，以增强算法对不同规模和特征问题的迁移能力。通过一系列实验，我们证明，所提出的 DAC-ALNS 算法显著提高了解的求解效率，并表现出出色的迁移能力。关键词：自适应大邻域搜索；深度强化学习；行动者-评论家模型；图神经网络

电子邮箱地址：shaohua.yu@njust.edu.cn（余少华），mowenhao32@gmail.com（毛文浩），zgwu@ncepu.edu.cn（吴子高*），jpuchinger@em-normandie.fr（Jakob Puchinger）

1. 引言

元启发式算法是一类用于求解优化问题的算法，尤其适用于 NP-hard 问题。它们无法保证找到最优解，但通常能在合理的时间内找到近似最优解。由于在处理复杂问题时具有较高的效率和灵活性，元启发式算法已成为求解现实问题的热门方法。自适应大邻域搜索（ALNS）算法是一种广泛使用的元启发式算法，由 Ropke 和 Pisinger 于 2006年提出，并已广泛应用于物流、制造业及其他领域的各类优化问题。

自适应大邻域搜索算法的核心思想是交替使用不同的破坏与修复算子来构建新解。破坏算子负责移除当前解中的部分内容，而修复算子则着重于重建剩余部分。这些算子的选择概率通过权重调整机制进行自适应调整，并采用类似轮盘赌的算法进行选择。自适应大邻域搜索结合了邻域搜索的直观性与自适应机制的灵活性，使其能够根据问题特征自动调整搜索策略。然而，大量数值实验揭示了一个重要现象：在经典自适应大邻域搜索算法中，自适应机制的效果并没有预期的那么显著。根据 Turkeš 等人（2021）的研究，在自适应大邻域搜索算法中加入经典自适应机制，目标函数值平均仅提高了 0.14%。这一发现表明，尽管经典自适应机制在理论上具有优势，但其实际改进效果有限。因此，设计更高效的自适应机制至关重要。

在经典自适应机制中，选择破坏或修复算子时，并未充分考虑这两类算子之间的相互作用。选择过程主要依赖算子在各自破坏或修复集合中的即时权重。例如，选择修复算子时，不会考虑先前使用的破坏算子类型及其对问题解的影响。这种独立的选择策略可能导致算子组合不当，从而影响算法性能。此外，经典自适应机制过度依赖单一指标：算子在集合中的权重及其对目标函数的即时贡献。这种方法忽略了解题过程中当前状态对问题解的潜在影响，因此可能无法充分评估算子的实际效果，进而影响自适应机制的整体效率。

为克服这些局限，需要制定更细致、更全面的自适应策略。该策略应考虑破坏与修复算子之间的相互依赖关系，以及整体求解过程中状态对算子选择的影响。因此，本文提出一种基于双演员-评论家模型（DAC）的自适应大邻域搜索算法新型自适应机制，以更好地实现算子的自适应选择。该新型自适应机制为破坏过程和修复过程分别构建策略网络，并利用价值网络评估算子选择，从而制定更优的优化策略。在此过程中，我们使用图神经网络（GNNs）提取路径特征，作为破坏过程的状态空间；并将所选破坏算子及图神经网络提取的特征作为修复过程的状态空间，以便更好地评估求解过程中当前状态对算子选择的潜在影响。

本文的主要贡献如下：

1. 我们使用 DAC 优化自适应大邻域搜索的自适应机制，在算子选择过程中有效利用破坏算子与修复算子之间的相互影响，并充分考虑当前状态对问题求解的潜在影响。2. 我们使用图神经网络提取并聚合图特征，为破坏执行器网络和修复执行器网络构建状态空间，使网络能够适应规模和特征各异的问题实例，从而提高算法的适应性和迁移能力。3. 我们使用容量约束车辆路径问题（CVRP）和带时间窗车辆路径问题（VRPTW）的标准数据集，对所提出的自适应机制进行了大量数值实验，验证了所提方法的有效性。

本文其余部分的结构如下：第 2 节介绍相关工作，第 3 节介绍方法论，第 4 节为数值实验，第 5 节总结全文。

2. 相关工作

2.1. 自适应大邻域搜索

自适应大邻域搜索算法是一种高效的元启发式方法，它通过引入自适应机制改进了大邻域搜索算法（Shaw 1998）。该算法利用权重调整机制和轮盘赌选择方法，自适应地选择不同的算子，引导可行解的破坏与修复过程，从而提升全局优化能力（Ropke 和 Pisinger 2006，Lutz 2015，Mourad 等，2021）。

自适应大邻域搜索算法提出以来，学者们一直在探索优化其标准组件的各种方法，包括初始解生成、破坏与修复算子、自适应机制以及接受和终止准则。

在自适应大邻域搜索中，初始解生成以及破坏与修复算子的选择往往具有很强的问题特定特征，其设计必须紧密依赖于问题解的结构特征。在初始解生成方法的设计方面，大多数研究采用简单的启发式方法来获得初始解。一些学者提出了多起点自适应大邻域搜索方法，同时生成多个初始解，以增加初始解的多样性，进而增强搜索方向多样性（Han and Chu 2016, Martí et al. 2019）。破坏与修复算子的典型设计包括随机移除、贪婪移除、随机插入和贪婪插入（Qu and Bard 2013）。一些研究通过轮盘赌机制独立选择破坏与修复算子，例如标准形式的自适应大邻域搜索（Ropke and Pisinger 2006）。另一些研究则设计了成对的破坏-修复算子，使其能够在自适应大邻域搜索中组合使用（Belhaiza 2019）。根据 Yu et al.（2025），前者优于后者。

自适应大邻域搜索中的自适应机制、接受准则和终止准则通常具有通用性。根据 Mara et al.（2022）的研究，关于自适应大邻域搜索中接受准则和终止准则的研究很多。例如，基于贪婪机制、Metropolis准则、记录到记录准则、阈值接受准则和帕累托支配的接受准则（Santini et al. 2018），以及基于迭代次数、未改进迭代次数、运行时间限制和退火温度的终止准则（Breunig et al. 2019；Yu et al. 2022），已广泛应用于自适应大邻域搜索。然而，目前关于自适应大邻域搜索中自适应机制的研究较为有限。在 Mara et al.（2022）调查的251篇相关文章中，有250篇使用轮盘赌作为自适应机制，占99.21%；仅有1篇使用随机通用抽样策略（Chowdhury et al. 2019），占0.40%。过去两年中，关于自适应机制的研究同样较为有限，Yu et al.（2025）提出了一种基于基于破坏算子的修复权重调整方法的自适应大邻域搜索算法自适应机制。

尽管研究表明，目前研究的自适应机制类型相对较少，其中权重调整轮盘赌是主要的自适应机制，但这并不意味着现有自适应机制表现出色。大量数值实验揭示了一个显著现象：经典自适应大邻域搜索算法中的自适应机制未能达到预期效果。根据 Turkeš 等人（2021）的研究，将经典自适应机制应用于自适应大邻域搜索

算法中，目标函数值平均仅提高了 0.14%。这一发现表明，尽管自适应机制在理论上具有优势，但其实际效果有限。因此，设计更高效的自适应机制尤为重要。

2.2. 自适应大邻域搜索中的深度强化学习

人工智能方法是一种创新的优化技术，通过构建计算机系统来模拟和理解人类智能、行为和规律。Aydin 等人（2024）指出，对于元启发式算法，可以通过景观分析确定多个特征来表征问题，并利用人工智能技术构建自适应算子选择方案，以应对效率和可扩展性方面的问题。

研究人员近期开始探索将强化学习技术应用于元启发式算法的算子选择过程。Kalatzantonakis 等人（2023）表明，通过学习环境和奖励机制，强化学习能够优化自适应大邻域搜索的参数设置，包括邻域结构的权重和扰动策略，从而提高搜索效率和质量。Durgut 等人（2021）提出了一种基于Q学习的强化学习算法，并将其集成到标准人工蜂群算法中，以指导算法的算子选择模块，从而解决集合并背包问题。该方法的核心是将问题状态映射到一组最匹配的算子，从而提高整个搜索过程的多样性，并构建最优算子序列。实验结果验证了该方法的有效性。

深度学习擅长从原始高维数据中提取有用特征，增强状态表示能力，使算法能够处理更复杂的问题，并更好地适应不同的环境（Mnih 等人（2015））。Kallestad 等人（2023）介绍了基于强化学习的深度学习方法，并提出了一种深度强化学习辅助的自适应大邻域搜索算法。该方法使用近端策略优化算法（Schulman 等人（2017））学习如何选择破坏-修复算子对的策略。实验表明，这种基于深度强化学习的自适应方法在包括容量约束车辆路径问题在内的问题上有效。Reijnen 等人（2024）进一步利用深度强化学习，在搜索过程中选择破坏-修复算子对、调整参数并控制接受准则。该方法旨在从搜索状态中学习，并为下一次迭代配置自适应大邻域搜索，以获得更有效的解。他们在带随机权重和时间窗的方向问题（由国际人工智能联合会议竞赛提出）上评估了该方法。结果表明，该方法优于经典自适应大邻域搜索。

基于深度强化学习与图神经网络相结合的方法也受到关注，并被用于优化自适应大邻域搜索。Johnn et al.（2023）强调了图神经网络在泛化到比训练期间观察到的问题实例规模更大的实例方面的潜力，这表明该方法的表现优于经典自适应大邻域搜索的自适应层。

尽管Q学习算法原理简单、易于实现，并且从理论上能够收敛到最优策略，但对于状态空间规模较大的问题，可能会面临维数灾难，即Q表变得极其庞大，从而导致计算和维护困难。深度Q网络算法通过引入深度神经网络来近似Q值函数，解决了这一问题，适用于状态空间较大的问题。然而，该算法在训练过程中仍会出现高估现象，从而降低智能体的学习效率。

我们还注意到，现有与深度强化学习相结合的自适应大邻域搜索算法，仅针对破坏-修复算子对的选择建立马尔可夫决策过程模型（Kallestad et al. 2023，Reijnen et al. 2024），或仅针对破坏算子建立模型（Johnn et al. 2023），随后使用深度强化学习算法进行选择。然而，根据Yu et al.（2025）的研究，经典自适应大邻域搜索自适应机制优于破坏-修复算子对方法，但不如基于破坏的修复权重调整方法；后者在进行选择时考虑了破坏算子与修复算子之间的交互作用。

因此，充分考虑破坏与修复算子对新解质量的联合影响，将破坏与修复过程分别建模为马尔可夫决策过程，并充分发挥破坏与修复算子在自适应大邻域搜索算法权重调整过程中的作用，将显著有利于自适应大邻域搜索问题的求解。

近端策略优化（PPO）和行动者-评论家模型（蚁群算法）通过降低训练过程中的方差，提高了训练稳定性，并加快了收敛至更优策略的速度。值得注意的是，在某些情境下，AC 算法可视为 PPO 的一种特殊形式（ Huang et al. 2022 ）。AC 算法的结构高度灵活，由两个相互独立的部分组成：行动者（负责学习策略）和评论家（负责评估价值函数）。这种结构设计使我们能够分别定制和优化这两个网络，从而更好地满足特定需求。因此，基于 DAC 设计一种新的自适应机制具有很大潜力。

3. 方法论

在求解车辆路径问题或其他优化问题时，自适应大邻域搜索是一种有效的元启发式方法，但其性能很大程度上取决于破坏与修复算子的选择。传统自适应大邻域搜索的自适应机制仅依据各算子的即时权重，忽略了当前状态对算子选择和问题解的影响，这可能会影响算子评估和算法效率。为此，我们提出一种基于双演员-评论家的自适应机制，以改进自适应大邻域搜索算法。该机制结合图神经网络，分别为破坏过程和修复过程构建策略网络，并使用价值网络评估算子选择过程。该机制将图神经网络提取的路径特征用作破坏过程的状态空间；所选破坏算子与图神经网络提取的特征共同构成修复过程的状态空间。DAC-ALNS 算法将更全面地考虑破坏算子和修复算子选择之间的相互依赖关系，包括破坏算子的选择对后续修复算子选择的潜在影响，并更好地利用整个解的求解过程中的数据来指导算子选择。

本节首先介绍 DAC-ALNS 算法的基本架构，其次介绍 DAC-ALNS 算法中涉及的马尔可夫决策过程。

3.1. DAC-ALNS 算法架构

本节首先介绍 Actor-Critic 算法以及如何通过 DAC 选择算子，然后介绍 DAC-ALNS 的算法框架。

(1) Actor-Critic算法

在 Actor-Critic 算法中，演员网络的作用是学习策略。它通过智能体与环境之间的交互建立状态空间，并根据策略选择动作。另一方面，评论家网络负责评估演员网络学到的策略，引导演员网络持续改进和优化策略。评论家网络通过估计状态-动作价值函数来指导策略更新。因此，行动者和评论家的作用互为补充，彼此交互且相互依赖。

图1展示了行动者-评论家算法的示意图。

（2）基于 DAC 的自适应大邻域搜索算子选择

为了增强自适应机制的有效性，我们利用Actor-Critic算法组件的高可扩展性，将破坏与修复算子建模为两个独立的神经网络。具体而言，我们分别为破坏和修复过程设计了一个演员网络，各自负责选择最合适的策略。此外，这两个演员网络共享一个评论家网络，用于评估策略选择并评估当前状态。因此，我们采用双Actor-Critic方法替代经典自适应大邻域搜索算法中的自适应层，从而更全面地考虑破坏与修复算子之间的相互依赖关系，并更有效地利用整个解生成过程中的数据来影响算子选择。

基于 DAC 的算子权重调整机制如图2所示。

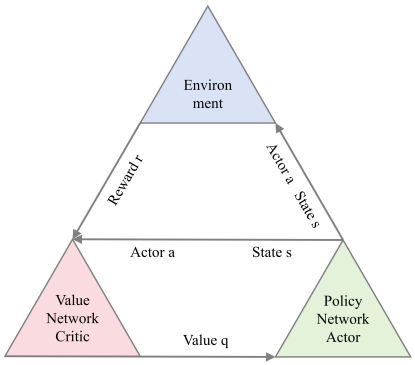


图1：Actor-Critic算法示意图

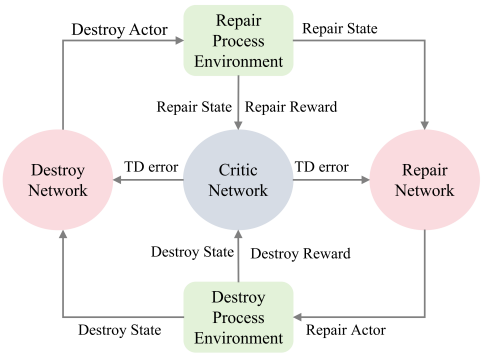


图2：基于 DAC 的算子权重调整机制

(3) DAC-ALNS算法框架

我们将基于 DAC 的算子自适应选择机制嵌入自适应大邻域搜索的自适应层中（自适应大邻域搜索算法架构借鉴了 Lutz（2015）的经验），从而得到基于双演员-评论家的自适应大邻域搜索算法。

所提出的 DAC-ALNS 在训练和部署阶段的伪代码分别见算法 1和算法2。

3.2. 马尔可夫决策过程

强化学习的数学基础是马尔可夫决策过程（MDP）。MDP 通常由状态空间、动作空间和奖励函数等组成。强化学习是一种序贯决策过程，旨在寻找一条决策规则（即，

算法 1 训练阶段：双重演员-评论家自适应大邻域搜索

训练
输入： 训练步数 M
步 $\leftarrow 0$
当步 $< M$ **时执行**
  初始化问题实例
  初始化可行解 x
  $x_{best} \leftarrow x$
  初始化状态 $s_{t_1} \leftarrow 0, s_{t_2} \leftarrow 0$
  初始化演员网络参数 θ 和评论家网络参数 ω
  while不满足停止条件 **do**
    根据状态 s_t，使用策略 π_{θ_1} 选择并执行破坏算子 d
    使用被销毁的解和节点更新状态 s_{t_2+1}，并获得奖励 r_t
    根据状态 s_{t_2+1}，使用策略 π_{θ_2} 选择修复算子 r
    $x_t \leftarrow r(d(x))$
    如果 接受 (x_t, x) **则**
      $x \leftarrow x_t$
    结束如果
    如果 $c(x_t) < c(x_{best})$ **则**
      $x_{best} \leftarrow x_t$
    结束如果
    步 骤 $\leftarrow$ 步 骤 $+ 1$
    更新状态 s_{t_1+1} 并获得奖励 r_t
    更新评论家网络参数 ω 使用时序差分
    使用梯度上升法更新行动者网络参数 θ
  结束循环
结束循环
返回优化参数 θ_1, θ_2

算法2 训练阶段：双重演员-评论家自适应大邻域搜索

1：部署：2：输入：选择策略 π_θ3：可行解 x；$x_{best} \leftarrow$ x 4：当不满足停止条件时执行 5：根据状态 s_{t_1}，使用策略 π_{θ_1} 选择破坏算子 $d \in \Omega^-$ 6：根据状态 s_{t_2}，使用策略 π_{θ_2} 选择修复算子 $r \in \Omega^+$ 7：$x_t \leftarrow r(d(x))$ 8：如果接受 (x_t, x)，则 9：$x \leftarrow x_t$10：结束 if 11：如果 $c(x_t) < c(x_{best})$，则 12：$x_{best} \leftarrow x_t$13：结束 if 14：结束 while 15：返回：x_{best}

最大化系统累积奖励的策略，换言之，即最大化价值（Nazari 等，2018）。

接下来，我们将介绍 DAC-ALNS 算法在容量约束车辆路径问题和带时间窗的车辆路径问题中涉及的状态空间、动作空间和奖励函数。

3.2.1. 状态空间

在马尔可夫决策过程中，状态表示封装了智能体感知到的环境信息和动态变化，为深度强化学习算法中的决策和奖励评估提供必要的输入。状态空间的设计质量直接影响学习算法的收敛速度和最终性能。

本文采用图神经网络提取紧凑且信息丰富的图级特征，在保留关键特征的同时显著降低状态复杂度。

状态表示（容量约束车辆路径问题）。我们将每个 CVRP 解（部分解或完整解）表示为一个图$G = (V,E)$，其节点特征包括平面坐标、需求量和表示服务是否完成的二值指示变量。边集结合静态的k近邻或全连接拓扑与动态生成的弧，并使用距离倒数作为边权重。节点特征经过两层图卷积网络（GCN）处理以生成嵌入，随后通过注意力加权平均池化和逐元素最大池化进行聚合，形成简洁的图级嵌入$g \in \mathbb{R}^{2d}$。

为纳入搜索阶段的上下文，我们附加两个标量：破坏比例 ρ （被移除的客户比例）以及破坏算子的归一化标识符 d 。因此，破坏状态（移除前）定义为 $s^{\text{des}} = [g|0|0]$，修复状态（移除后）定义为 $s^{\text{rep}} = [g'|\rho|\tilde{d}]$, ，其中 g' 根据部分修改后的解重新计算。由此得到的嵌入可同时供共享评论家网络和演员网络使用。

状态表示（*VRPTW*）。与容量约束车辆路径问题类似，每个 VRPTW 解（部分解或完整解）都编码为图 $G = (V,E)$，它将静态的 k-近邻（或全连接）骨干结构与由动态路线派生的弧相结合。边采用逆距离加权或直接距离加权。节点包含 10 维特征，结合了空间坐标、需求量、服务状态、归一化的时间窗约束、服务时长，以及紧迫度和等待时间等时间指标。

一个两层图卷积网络将这些特征处理为嵌入，并通过注意力加权平均和逐元素最大池化进行聚合，得到图级嵌入 $g \in \mathbb{R}^{2d}$。为了捕捉解

˜修改上下文，我们再次纳入破坏比例 ρ 和归一化破坏算子 ID d，将破坏状态定义为 $s^{\text{des}} = [g|0|0]$，将修复状态定义为 $s^{\text{rep}} = [g'|\rho|\tilde{}d]$,，其中 g' 反映了部分破坏的解状态。

DAC-ALNS 下的状态空间概要如表 1 所示。

表 1：DAC-ALNS 框架下的状态空间概要

方面	CVRP	带时间窗的车辆路径问题
图与边	$G = (V,E)$；静态k-近邻 + 路径-诱导动态弧；边权重 $w_{uv} = 1/d_{uv}$。	$G = (V,E)$；静态 k-NN + 从当前路径生成的动态弧路径；边权重$w_{uv} = 1/d_{uv}$。
节点特征	$[x_i, y_i, q_i, \delta_i]$ 维度：4 注：$\delta_i \in \{0,1\}$ (已服务状态)。	$[x_i, y_i, q_i, \delta_i, \frac{e_i-S}{H}, \frac{l_i-S}{w\|\|_\infty}, \frac{w_i}{\|s\|_\infty}, \frac{s^iH}{\|}, \frac{t_i}{H}, \frac{wt_i}{H}]$ 维度：10 注：$H = E - S$ (时间范围)。
两个框架的共同组件		
嵌入	两层图卷积网络；注意力加权平均和逐元素最大池化 $\Rightarrow g \in \mathbb{R}^{2d}$。	
上下文标量	破坏比例 $\rho \in (0,1)$；归一化破坏算子索引 $\tilde{d}$.	
状态向量	$s^{\text{des}} = [g \parallel 0 \parallel 0]$（初始状态）； $s^{\text{rep}} = [g' \parallel \rho \parallel \tilde{d}]$ （中间状态，来自部分解的g'）。	

3.2.2. 动作空间

DAC-ALNS 算法的动作空间由两类基本动作组成：破坏操作和修复操作。破坏操作涉及从当前解中移除一部分元素，而修复操作则在破坏后将元素添加回解中，从而形成新的可行解。这两类动作共同定义了深度强化学习的动作空间，即算法在每一步可以选择执行的动作集合。此外，我们引入了一种破坏规模策略，使 DAC-ALNS 算法能够调整破坏规模。

我们的破坏与修复操作适用于问题特定约束。由于 CVRP 和 VRPTW 的算子集合相同，现总结如下：随机移除、最差移除、随机修复、贪婪修复、后悔修复。详情请参见附录 A。

3.2.3. 奖励函数

我们使用总路径成本作为核心信号（越低越好）。令C_{before} 表示应用当前算子之前的成本，C^{destroy} 表示销毁执行器后立即产生的成本，C^{repair} 表示修复后立即产生的成本，以及

C_{best} 当前的全局最优（迄今为止的最优）成本。

（1）销毁奖励函数

销毁执行器会移除部分解，为后续修复留出空间。其奖励结合了质量标志和潜力项。在我们的实现中，定义了两个基础奖励参数 α_1 和α_2。如果销毁后解的总成本低于或等于当前全局最优（表明具有较高潜力），则给予更高的基础奖励 α_2 。如果成本低于当前解但尚未达到全局最优，则应用基础奖励 α_1 。此外，还会加入由 β 缩放的潜力项，以反映其与全局最优解之间的差距。如果销毁后成本增加，则奖励为零。销毁过程的奖励函数定义如下：

$$R_{\text{destroy}} = \begin{cases} \alpha_1 + \beta \dfrac{C_{\text{best}} - C_{\text{destroy}}}{\max(\varepsilon,\, C_{\text{best}})}, & \text{if } C_{\text{best}} < C_{\text{destroy}} < C_{\text{before}}, \\ \alpha_2 + \beta \dfrac{C_{\text{best}} - C_{\text{destroy}}}{\max(\varepsilon,\, C_{\text{best}})}, & \text{if } C_{\text{destroy}} \le C_{\text{best}}, \\ 0, & \text{if } C_{\text{destroy}} \ge C_{\text{before}} \end{cases} \tag{1}$$

（2）修复奖励函数

修复执行器重构解，我们将质量改进项与探索奖励相结合。探索部分通过以下方式促进使用频率较低的修复算子：

$$\text{explore} = \delta\left(1 - \frac{f_{\text{rep}(id)}}{\max(1,\, f_{\max})}\right), \tag{2}$$

其中， $\delta \ge 0$ 控制探索强度，$f_{\text{rep}(id)}$ 表示索引为 id 的当前修复算子的使用次数，$f_{\max} = \max_j f_{\text{rep}(j)}$ 表示所有修复算子中的最大使用次数。对于解的质量，我们采用三个等级及其基础值 $r_1, r_2, r_3 \in \mathbb{R}$（通常为 $r_1 > r_2 > r_3$）：

$$R_{\text{repair}} = \begin{cases} r_1 + \text{explore}, & \text{if } C_{\text{repair}} < C_{\text{best}}, \\ r_2 + \text{explore}, & \text{if } C_{\text{best}} \le C_{\text{repair}} < C_{\text{before}}, \\ r_3 + \text{explore}, & \text{if } C_{\text{repair}} \ge C_{\text{before}}. \end{cases} \tag{3}$$

换言之，优于当前最优值的修复执行器会获得最高的正基础奖励 r_1；成本虽优于操作前但未超过最优值的修复执行器会获得较低的正基础奖励 r_2；未能改善操作前成本的修复执行器则会受到惩罚或获得少量奖励 r_3。随后，这三种情况下都会加上探索奖励。

4. 数值实验

为评估 DAC-ALNS 模型，我们开展了两阶段实验研究。首先，将该算法与经典基准测试进行比较，以展示其解的高质量。随后，我们开展了泛化实验，以确认该模型在未见数据上保持稳定，并在问题规模增大时展现出可扩展性。

图3展示了所提出的 DAC-ALNS、Lutz 于 2015年改进的经典 ALNS，以及 AC-ALNS 基线的框架。

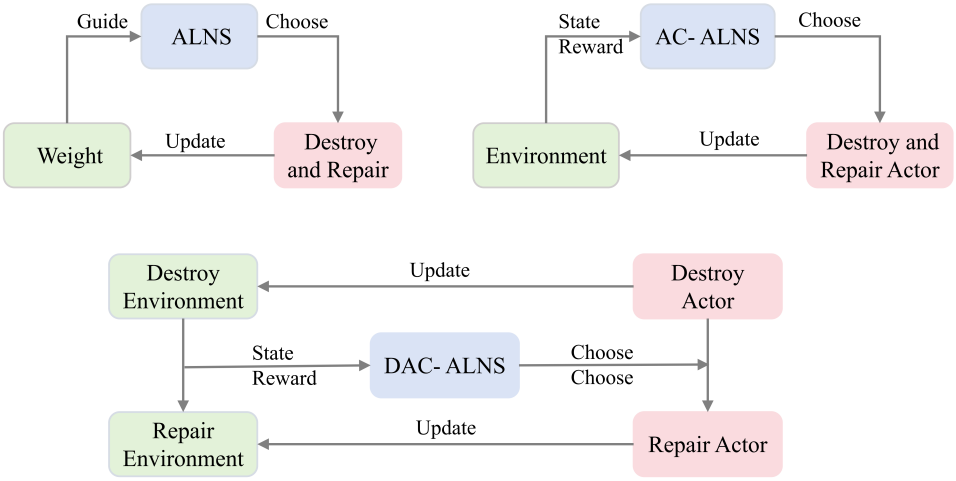


图3：ALNS、AC-ALNS 和 DAC-ALNS 运行逻辑的示意图。

4.1. 实验设置

4.1.1. 数据集与基准测试

我们采用了经典的容量约束车辆路径问题和带时间窗的车辆路径问题数据集，这不仅确保了实验的可复现性，也便于后续研究者将其提出的算法与我们的算法进行比较。

1. **容量约束车辆路径问题**：我们采用了 Augerat（1995）提出的经典实例（A、B 和 P 组），这些实例具有多样的客户分布和容量约束，有助于我们全面评估 DAC-ALNS 的性能。

2. **带时间窗的车辆路径问题**：我们选用了知名的Solomon基准测试集（C、R和RC组）（Solomon，1987），该测试集涵盖聚类分布、随机分布和混合地理分布，以及多种时间窗设置。选择该测试集有助于更有效地评估DAC-ALNS在带时间窗的车辆路径问题上的表现。

实例分层。 为全面评估DAC-ALNS在不同问题规模下的表现，我们根据客户节点数（n）将测试实例划分为小型、中型和大型三类，详情见表2。

表2：对比实验的数据集划分

规模	**节点范围（n）**	**实例数量**	
		容量约束车辆路径问题	**带时间窗的车辆路径问题**
小型	$n < 40$	18	18
中型	$40 \le n < 60$	18	18
大型	$60 \le n < 80$	18	18

表 3 总结了泛化实验的数据集划分。

表 3：泛化分析的数据集划分

泛化类型	**问题**	**迁移协议（源 → 目标）**
分布内	CVRP 带时间窗的车辆路径问题	A/B/P 组（中等） → 其余 A/B/P 组 C/R/RC 组（中等） →其余 C/R/RC 组
分布外	CVRP 带时间窗的车辆路径问题	A组（小型/中型） → A组（大型） R组（小型/中型） →R组（大型）

注：对于带时间窗的车辆路径问题，测试实例通过从原始Solomon基准测试集中进行随机分布采样生成，以严格遵循分层节点范围（$n < 40$、 $40 \le n < 60$以及 $60 \le n < 80$）。此外，泛化分析采用直接源自比较数据集的数据集，以保持实验一致性。

4.1.2. 基线方法

为系统评估 DAC-ALNS，我们将其与两种基线方法进行比较，旨在验证其在容量约束车辆路径问题和带时间窗的车辆路径问题中的解的质量，并展示双重架构相较于

AC-ALNS 架构：

1. 自适应大邻域搜索（基线）：传统元启发式算法的经典范式。作为车辆路径问题领域成熟的最先进水平启发式算法，它可作为参照标准，用于评估我们 DAC-ALNS 算法的解的质量和效率。
2. AC-ALNS（基线）：该变体代表一种标准的深度强化学习范式（将单一行动者-评论家框架与自适应大邻域搜索相结合），并作为消融研究的基线。通过将其与我们的模型进行比较，我们可以有效地分离双重机制的影响，从而验证双行动者架构相较于传统单行动者方法的必要性。

4.1.3. 评估指标

我们采用三个主要指标来评估算法的解质量和计算效率。

1. 最佳差距：该指标反映算法最终的搜索潜力，通过比较不同方法在多次独立运行中找到的最优解，展示它们之间的性能差距。2. 平均差距：采用同一实例在多次独立运行中的平均差距，评估模型表现的一致性，并反映其整体求解稳定性。3. 时间比：该比率量化相对计算开销，并提供与基线方法直接比较的效率基准。

为了比较不同算法，我们衡量待测试方法（称为 $\mathcal{A}$）与参考方法（称为 $\mathcal{B}$）之间的相对性能差距。我们按如下方式计算该差距：

$$\mathrm{Gap}(\mathcal{A},\mathcal{B}) = \frac{\mathrm{Obj}_{\mathcal{A}} - \mathrm{Obj}_{\mathcal{B}}}{\mathrm{Obj}_{\mathcal{B}}} \times 100\% \tag{4}$$

其中，$\mathrm{Obj}_{\mathcal{A}}$ 和 $\mathrm{Obj}_{\mathcal{B}}$ 分别是方法 $\mathcal{A}$ 和方法 $\mathcal{B}$ 找到的最佳目标函数值。差距为负表示方法 $\mathcal{A}$ 优于方法 $\mathcal{B}$（即得到了更好的解）。如果时间比大于 1，则表示方法 $\mathcal{A}$ 的计算成本更高。

我们采用显著性水平为 $\alpha = 0.05$ 的单侧威尔科克森符号秩检验，以判断性能差异是否具有统计显著性。

4.1.4. 实现细节

所提出的 DAC-ALNS 和所有基线算法均使用 Python 3.10 实现，并在单个 CPU 核心上运行。所有实验均在配备双路 AMD EPYC 7Y43 处理器（96 个核心，运行频率为 2.25–3.35 GHz）的计算平台上进行。每个问题实例均进行了 10 次独立试验。主要超参数的完整配置汇总于表 4。

表 4：关键超参数配置

类别	参数	价值
训练	最大迭代次数	$\max(2000, 50 \times n)$
	随机种子集合	$\{0, 100, \ldots, 900\}$
	探索率 (ϵ)	0.1
ALNS 算子	移除比例 (ρ)	$[0.1, 0.4]$
	最坏移除数量	$[5, 20]$
	后悔修复 (N)	2
深度强化学习智能体	学习率 (α)	1×10^{-3}
	折扣因子 (γ)	0.9
	熵系数	0.01
奖励与 状态	奖励项 (r_1,r_2,r_3)	$(20, 10, -0.1)$
	图神经网络嵌入维度	64

4.2. 对比实验

4.2.1. 总体性能分析

表5展示了所有容量约束车辆路径问题和带时间窗的车辆路径问题实例的总体性能指标，并将 DAC 与自适应大邻域搜索和蚁群算法基线方法进行了比较。

表5：容量约束车辆路径问题和带时间窗的车辆路径问题实例的综合性能比较

数据集	Size	DAC 与蚁群算法对比			DAC 与自适应大邻域搜索			蚁群算法 vs 自适应大邻域搜索		
		Best	Avg	Time	Best	Avg	Time	Best	Avg	Time
CVRP	小型	0.00%	−0.06%	1.05	−0.02%	0.01%	2.33	−0.02%	0.07%	2.28
	中型	−0.04%	−0.06%	1.32	−0.06%	0.01%	2.23	−0.02%	0.07%	1.69
	大型	−0.22%	−0.19%	1.58	−0.31%	−0.11%	2.47	−0.09%	0.07%	1.57
带时间窗的车辆路径问题	小型	−0.01%	0.04%	1.08	0.00%	−0.04%	1.78	0.01%	−0.08%	1.73
	中型	−0.10%	−0.57%	0.93	−0.11%	−0.49%	1.77	−0.01%	0.10%	1.92
	大型	−0.03%	−0.08%	1.00	−0.14%	−0.44%	1.94	−0.11%	−0.37%	1.93

注： 负值表示前列方法优于后列方法。时间$(\mathcal{A}, \mathcal{B}) = T_{\mathcal{A}}/T_{\mathcal{B}}$。 —

如表5所示，DAC 的解质量持续优于两种基线方法。在绝大多数情况下，DAC 在解质量方面的最优差距和平均差距均为负值。有趣的是，

总体趋势表明，随着问题规模增大，这一优势愈发明显。例如，在大型车辆路径问题中，DAC 的表现比自适应大邻域搜索（ALNS）好 0.31%（最佳差距）。在带时间窗的车辆路径问题中，提升幅度更大，平均差距分别达到 −0.44%（大型）和 −0.49%（中型）。与所提出的双执行器框架相比，蚁群算法（AC）基线方法的平均差距经常为正值（例如，中型带时间窗的车辆路径问题中为 +0.10%），凸显了其不稳定性。由于 DAC 所需的策略推理和决策逻辑比 AC 和 ALNS 更多，因此通常会带来更大的效率开销。

不过，这种开销并非在所有情况下都更高。例如，在中型带时间窗的车辆路径问题上，DAC 比蚁群算法更快（TimeRatio = 0.93），这表明在某些情况下，学习策略能够减少低效的搜索步骤。总体而言，运行时间结果显示，搜索效率与决策开销之间存在权衡。不过，在相同迭代次数下，DAC 始终能取得更好的结果，这至少说明它所做的选择更为成功。

4.2.2. 比较性能的统计分析

表6详细列出了胜负平记录和统计显著性检验，以验证这些改进的有效性并考察性能趋势。

表6：详细统计比较：DAC 与基线方法（自适应大邻域搜索和蚁群算法）

数据集	规模	N	DAC 与自适应大邻域搜索			DAC 与蚁群算法（消融）		
			胜/负/平	p 值	显著性	胜/负/平	p 值	Sig.
CVRP	小型	18	5 / 6 / 7	0.5000	n.s.	7 / 4 / 7	0.0653	n.s.
	中型	18	11 / 7 / 0	0.2475	n.s.	13 / 5 / 0	0.1231	n.s.
	大型	18	12 / 6 / 0	0.0982	n.s.	11 / 7 / 0	0.1419	n.s.
	全局		**54 /19 / 7**	**0.0861**	**不显著**	**31 /16 /7**	**0.0298**	*
带时间窗的车辆路径问题	小型	18	6 / 7 / 5	0.5417	n.s.	4 / 10 / 4	0.8016	n.s.
	中型	18	13 / 4 / 1	0.0123	*	12 / 4 / 2	0.0037	**
	大型	18	15 / 3 / 0	0.0020	**	10 / 8 / 0	0.2899	n.s.
	全局		**54 34 /14 / 6**	**0.0006**	***	**26 /22 /6**	**0.0278**	*

注：**胜/负/平**：胜/负/平。**显著性**：基于威尔科克森符号秩检验的显著性。符号：不显著（$p > 0.05$）；*（$p < 0.05$）；**（$p < 0.01$）；***（$p < 0.001$）。

(1) DAC 与自适应大邻域搜索（绝对性能）对比。对比结果表明，存在明显的规模效应。在小型实例上，DAC 与自适应大邻域搜索在统计上没有显著差异（例如，小规模容量约束车辆路径问题上的 $p = 0.5000$），且平局频繁出现。一个可能的原因是，这些实例相对简单，因此两种方法通常都能快速找到已知最佳解，性能也很快趋于饱和。随着实例规模增大、约束增多，DAC 开始显著优于自适应大邻域搜索。在大规模带时间窗车辆路径问题数据集上，DAC 在 18 次

运行中赢得了 15 次（$p = 0.0020$），这不太可能仅由随机性解释。汇总后的带时间窗车辆路径问题结果也呈现出相同趋势：DAC 显著优于自适应大邻域搜索（$p = 0.0006$）。总体而言，这些发现表明，随着问题复杂度提高，DAC 的扩展能力更强。

(2) DAC 与 AC（机制有效性）。针对简化基线（AC）的消融研究揭示了实例复杂度如何影响所提机制的实用性。在小规模带时间窗的车辆路径问题中，AC 仍具竞争力，甚至在 18 次试验中赢得 10 次，这表明当底层结构相对简单时，单流策略可能已经足够。随着实例规模增大，这一优势便不再持续。在中型和大型设置中，DAC 的双执行器设计越来越有益，结果也逐渐倾向于 DAC。例如，在中型带时间窗的车辆路径问题中，DAC 取得 12 胜、4 负。在基准测试中汇总结果后，DAC 在容量约束车辆路径问题（$p = 0.0298$）和带时间窗的车辆路径问题（$p = 0.0278$）上均显著优于 AC，尽管 AC 在较小规模的问题上表现出色。总体而言，证据表明，低复杂度情况下单流架构或许足以应对，而在更具挑战性的路径规划实例中，所提出的双重机制对于应对规模更大、维度更高的搜索空间至关重要。

4.3. 泛化实验

为评估所提出算法迁移至未见过的问题实例时的表现，我们开展了分布内和分布外评估。我们还考虑了一种迁移变体，称为 DAC-Transfer（DAC-T）。对于 DAC-T，所有网络参数仅在源数据上进行学习，并在测试期间保持固定；也就是说，该模型在未见过的问题实例上严格采用纯推理设置。

对于每个源数据集，我们将数据划分为训练部分（2/3）和测试部分（1/3）。为减少过拟合，我们采用提前停止策略：如果最优值在超过总迭代次数2%的时间内没有改善，则停止训练。

4.3.1. 泛化性能结果

表7详细说明了迁移模型的性能差距和时间效率。

表7中的一项显著结果是，DAC-T在分布外实例上表现尤为出色，有时甚至优于分布内数据。例如，在分布外的容量约束车辆路径问题任务中，DAC-T的最佳差距比自适应大邻域搜索提升了 −0.60%。此外，与在线训练的DAC相比，DAC-T在保持时间比约为或低于1.0的同时，取得了相当的解的质量（且如所有设置下均为负差距所示，表现始终更好）。这表明，DAC-T能够在实现卓越推理时间效率的同时取得出色的性能。

表7：泛化性能：DAC-T 与基线方法对比

实验类型	场景	DAC-T 与自适应大邻域搜索对比			DAC-T 与 DAC 对比		
		最佳差距	平均差距	Time	最佳差距	平均差距	Time
分布内·	CVRP	–0.16%	–0.32%	2.23	–0.09%	–0.29%	0.99
	带时间窗的车辆路径问题	0.00%	–0.07%	1.49	0.00%	–0.02%	0.85
分布外	CVRP	–0.60%	–0.57%	2.47	–0.09%	–0.52%	1.00
	带时间窗的车辆路径问题	–0.38%	–0.72%	2.00	–0.05%	–0.29%	1.16

注： 负值表示 DAC-T 优于基线方法。时间$(\mathcal{A},\mathcal{B})$ $=T_{\mathcal{A}}/T_{\mathcal{B}}$。 —

4.3.2. 泛化的统计显著性

为验证这些改进并非源于随机波动，我们在表8中报告了统计显著性检验。

表8：泛化的统计显著性：DAC-T 与基线方法对比

比较	场景	N	胜/负/平	p值	Sig.
DAC-T 与自适应大邻域搜索	容量约束车辆路径问题（输入）	6	6 / 0 / 0	0.0156	*
	带时间窗的车辆路径问题（输入）	6	4 / 1 / 1	0.1726	n.s.
	容量约束车辆路径问题（输出）	8	7 / 1 / 0	0.0078	**
	带时间窗的车辆路径问题 (外部)	8	8 / 0 / 0	0.0039	**
DAC-T 与 DAC	容量约束车辆路径问题 (内部)	6	5 / 1 / 0	0.0312	*
	带时间窗的车辆路径问题（输入）	6	2 / 2 / 2	0.3575	n.s.
	容量约束车辆路径问题（输出）	8	7 / 1 / 0	0.0117	*
	带时间窗的车辆路径问题（输出）	8	5 / 3 / 0	0.0547	n.s.

注： **胜/负/平**：胜/负/平。**显著性**：基于威尔科克森符号秩检验的显著性。符号：不显著（$p > 0.05$）；*（$p < 0.05$）；**（$p < 0.01$）；***（$p < 0.001$）。

统计检验进一步证实了迁移策略的稳定性。在分布外实例上，DAC-T 在所有带时间窗的车辆路径问题（VRPTW）案例中均胜过自适应大邻域搜索（ALNS）（$p = 0.0039$），并在 87.5% 的容量约束车辆路径问题（CVRP）案例中获胜（p =0.0078）。更令人意外的是，DAC-T 在 CVRP 上也显著优于在线训练的DAC（$p = 0.0312$，且 $p = 0.0117$），这表明，当搜索空间变为高维时，在小型实例上学到的结构知识可能比从头重新学习更有用。

4.3.3. 迁移能力洞察与机制讨论

DAC-T 在分布偏移下始终表现出色，这表明该策略并非只是记忆依赖于规模的模式，而是学会了能够迁移到不同实例规模的算子选择行为。一种合理的解释是，在较小实例上训练有助于模型获取可复用的结构先验，掌握应如何选择和组合 ALNS 算子。随着规模增大，在线

训练通常会因搜索空间维度更高、改进信号更稀疏而变得更加困难，而 DAC-T 则能在推理时直接利用这些已学得的先验。从实际操作角度来看，这支持了一种“在小规模数据上训练，在大规模场景中推理”的工作流程；对于时间紧迫的物流场景而言，这种方式尤其具有吸引力，因为大规模在线学习成本高昂，甚至根本不切实际。

5. 结论

我们提出了一种 DAC框架，旨在将经典自适应大邻域搜索元启发式用于车辆路径问题时，同时提升搜索效率和泛化能力。我们的核心思想是将破坏与修复阶段视为两个相互耦合但角色不同的决策过程。借助基于图神经网络的状态表示，策略能够利用实例结构，并根据不断演化的解轨迹调整算子选择，而不再完全依赖手工设计的自适应权重更新。在标准基准测试（容量约束车辆路径问题和带时间窗的车辆路径问题）上的大量实验得出以下结论：

1. 高约束场景下的优势：DAC在容量约束车辆路径问题上仍具有竞争力，并且在带时间窗的车辆路径问题上表现出明显的统计优势（$p < 0.001$）。这表明，当搜索空间受到严格时间窗约束的影响时，学习策略尤其有效。在这些情况下，优化长期回报似乎能够促使算子选择更具前瞻性，从而减少传统自适应方案中经常出现的短视行为。

2. 解耦双执行器设计的价值：消融研究表明，双执行器结构始终优于单流基线（蚁群算法）。将破坏与修复分离后，每个策略都能够实现专业化，从而更好地匹配这两个阶段在邻域搜索中所发挥的不同组合优化作用。

3. 稳健的零样本迁移：DAC-T 的迁移性能是一项核心发现。仅在小型实例上训练的模型，随后以冻结的、仅用于推理的模式运行，便可扩展到更大型的分布外实例；在若干情况下，其性能甚至超过在线的、针对特定实例的训练。这一结果支持一种实用的“小规模训练，大规模推理”范式，从而降低真实物流应用场景中的部署时间计算开销和延迟。

展望未来，我们计划缩小基准测试假设与实际运营要求之间的差距。基于 DAC-T 的低延迟推理能力，我们将把这一方法扩展到更丰富的车辆路径问题变体，并纳入贴近现实的工业约束，包括异构车队、动态需求以及电动汽车

路径规划因素。此外，将 DAC 融入实时决策支持流程，对于评估其在现代供应链中典型的随机扰动和不确定性下的鲁棒性也至关重要。

致谢

本研究得到国家自然科学基金 [项目（编号： 72301137]）的资助。感谢程浩在初期探索阶段所付出的努力。

数据可用性

代码仓库https://github.com/Shaohua-Yu/dac-alns包含已实现方法的数据和源代码。

参考文献

Augerat, P.（1995）。车辆路径问题的多面体方法。博士论文，格勒诺布尔国立综合理工学院-INPG。Aydin, M. E.、Durgut, R.、Rakib, A. 和 Ihshaish, H.（2024）。面向数据驱动自适应算子选择的基于特征的搜索空间表征。*Evolving Systems*，15(1)：99–114。Belhaiza, S.（2019）。用于解决现实需求响应式交通问题的混合自适应大邻域启发式算法。*Algorithms*，12(2)：39。Breunig, U.、Baldacci, R.、Hartl, R. F. 和 Vidal, T.（2019）。电动两级车辆路径问题。*Computers* & 《*Operations Research*》，103：198–210。Chowdhury, S.、Marufuzzaman, M.、Tunc, H.、Bian, L. 和 Bullington, W.（2019）。用于解决动态旅行商问题的改进蚁群优化算法：无人机野生动物监测案例研究。*Journalof ComputationalDesign and Engineering*，6(3)：368–386。Durgut, R.、Aydin, M. E. 和 Atli, I.（2021）。基于强化学习的自适应算子选择。*InformationSciences*，581：773–790。Han, A. F.-W. 和 Chu, Y.-C.（2016）。求解具有最低配送量的拆分配送车辆路径问题的多起点启发式方法。*Transportation Research Part E: Logistics and Transportation Review*，88：11–31。Huang, S.、Kanervisto, A.、Raffin, A.、Wang, W.、Ontañón, S. 和 Dossa, R. F. J.（2022）。优势演员-评论家算法是近端策略优化的特例。*arXiv preprint arXiv:2205.09123*。Johnn, S.-N.、Darvariu, V.-A.、Handl, J. 和 Kalcsics, J.（2023）。图强化学习在自适应大邻域搜索元启发式中的算子选择。载于优化与学习国际会议，第200–212页。Springer。

Kalatzantonakis, P.、Sifaleras, A. 和 Samaras, N.（2023）。用于容量约束车辆路径问题的强化学习-可变邻域搜索方法。*Expert Systems with Applications*，213：118812。Kallestad, J.、Hasibi, R.、Hemmati, A. 和 Sörensen, K.（2023）。用于解决组合优化问题的通用深度强化学习超启发式框架。*European JournalofOperationalResearch*，309(1)：446–468。Lutz, R.（2015）。自适应大邻域搜索。Mara, S. T. W.、Norcahyo, R.、Jodiawan, P.、Lusiantoro, L. 和 Rifai, A. P.（2022）。自适应大邻域搜索算法及其应用综述。*Computers* &《*Operations Research*》，146：105903。Martí, R.、Aceves, R.、León, M. T.、Moreno-Vega, J. M. 和 Duarte, A.（2019）。智能多起点方法。《*Handbook of Metaheuristics*》，第221–243页。Mnih, V.、Kavukcuoglu, K.、Silver, D.、Rusu, A. A.、Veness, J.、Bellemare, M. G.、Graves, A.、Riedmiller, M.、Fidjeland, A. K.、Ostrovski, G. 等（2015）。通过深度强化学习实现人类水平的控制。《*Nature*》，518(7540)：529–533。Mourad, A.、Puchinger, J. 和 Van Woensel, T.（2021）。将自主配送服务整合到客运系统中。《*International Journal of Production Research*》，59(7)：2116–2139。Nazari, M.、Oroojlooy, A.、Snyder, L. 和 Takác, M.（2018）。用于解决车辆路径问题的强化学习。《*Advances in Neural Information Processing Systems*》，31。Qu, Y. 和 Bard, J. F.（2013）。具有可配置车辆容量的异构取送货问题。《*Transportation Research Part C: Emerging Technologies*》，32：1–20。Reijnen, R.、Zhang, Y.、Lau, H. C. 和 Bukhsh, Z.（2024）。使用深度强化学习对自适应大邻域搜索进行在线控制。载于《*International Conference on Automated Planning and Scheduling*论文集》，第34卷，第475–483页。Ropke, S. 和 Pisinger, D.（2006）。用于带时间窗的取送货问题的自适应大邻域搜索启发式算法。《*European Journal of Operational Research*》，175(3)：1830–1844。Santini, A.、Ropke, S. 和 Hvattum, L. M.（2018）。自适应大邻域搜索元启发式中接受准则的比较。《*Journal of Heuristics*》，24：783–815。Schulman, J.、Wolski, F.、Dhariwal, P.、Radford, A. 和 Klimov, O.（2017）。近端策略优化算法。*arXiv preprint arXiv:1707.06347*。Shaw, P.（1998）。使用约束规划和局部搜索方法解决车辆路径问题。载于《约束规划原理与实践国际会议》，第417–431页。Springer。Solomon, M. M.（1987）。带时间窗约束的车辆路径与调度问题算法。《*Operations Research*》，35(2)：254–265。

Turkeš, R., Sörensen, K. 和 Hvattum, L. M. (2021). 元启发式算法的荟萃分析：量化自适应大邻域搜索中自适应性的影响。《*European Journal of Operational Research*》, 292(2):423– 442.

Yu, S., Campbell, A. M., Ehmke, J. F. 和 Puchinger, J.(2025). 一种采用机器人的混合式城市配送系统。《*European Journal of Operational Research*》.

Yu, S., Puchinger, J. 和 Sun, S. (2022). 基于货车与机器人的混合取送货路径规划问题。《*European Journal of Operational Research*》,298(3):894–914.

附录 A. 详细动作空间

容量约束车辆路径问题和带时间窗的车辆路径问题的动作空间采用统一的破坏与修复操作集合。

(1) 破坏操作

- **随机移除**：算法根据预定义的移除比例范围，从当前解中随机选取一定比例的节点并将其移除。

- **最差移除**：算法根据预定义的移除比例范围，以贪婪方式从当前解中移除一定比例的节点（移除对解贡献最差的节点）。**(2) 修复操作**

- **随机修复**：随机修复从未服务客户列表中随机选取客户，并将其插入任意当前可行路径；计算时会考虑所有车辆的路径。

- **贪婪修复**：贪婪修复会为所有当前未服务点确定成本最低的插入位置；计算时会考虑所有车辆的路径。

- **后悔修复**：重新插入移除的节点时，该方法会寻找当前最优位置，并预测未来成本，从而对解进行综合优化。事实证明，这一策略能够提升自适应大邻域搜索算法的性能，尤其是在处理大型复杂组合优化问题时。通过采用“后悔值”的概念，算法能够更深入地探索解空间，避免过早收敛，并找到更优的全局解。