分类号：　　单位代码：10389

密　级：（不填）　　学　　号：3211239028

福建農林大學

硕士专业学位论文

低延迟服务功能链调度算法研究

学位类别：电子信息硕士

专业领域：网络与信息安全

研究方向：不区分研究方向

学生姓名：刘智伟

指导教师：舒兆港 副教授

林东豪 高级工程师

完成时间：二〇二四年七月

Research on Low latency Service Function Chain Scheduling Algorithm

A Thesis Submitted to

Fujian Agriculture and Forestry University

in Partial Fulfillment of the Requirements

for

Professional Master's Degree in…

College of ……

Fujian Agriculture and Forestry University

Fujian, P.R. China

Submission Date:

独创性声明

本人声明，所呈交的学位（毕业）论文，是本人在指导教师的指导下独立完成的研究成果，并且是自己撰写的。尽我所知，除了文中作了标注和致谢中已作了答谢的地方外，论文中不包含其他人发表或撰写过的研究成果。与我一同对本研究做出贡献的同志，都在论文中作了明确的说明并表示了谢意，如被查有侵犯他人知识产权的行为，由本人承担应有的责任。

学位（毕业）论文作者亲笔签名：　　日期：

论文使用授权的说明

本人完全了解福建农林大学有关保留、使用学位（毕业）论文的规定，即学校有权送交论文的复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存论文。

保密，在　　年后解密可适用本授权书。　□

不保密，本论文属于不保密。　□

学位（毕业）论文作者亲笔签名：　　日期：

指导教师亲笔签名：　　日期：

目 录

摘 要

随着网络技术的不断发展，5G 网络相较于传统网络得到了进一步的升级，这使得网络服务呈现出了更多样化的趋势，同时低延迟网络应用场景也呈指数方式增长，满足用户不断增长的多样化需求已成为当前网络通信行业的必然趋势。为此，业界引入了软件定义网络（Software Defined Network, SDN）技术。SDN 的主要思想是将专有网络设备中的硬件和软件解耦合，使得软件不再依赖于特定的硬件设备，同时将解耦合的软件抽象为独立的虚拟网络功能（Virtualization Network Function，VNF）。这种虚拟化技术让 VNF 可以根据需要自适应地放置在物理网络节点上，从而使网络节点能够结合实际需求提供相应的网络功能，这不仅提高了网络的灵活性和满足服务质量需求，还能够合理利用网络资源，降低运营商的运营成本。

网络功能虚拟化（Network Function Virtualization, NFV）问题，也称为 NFV 资源分配问题，由 VNF 组成、VNF 放置和 VNF 调度组成。当前，现有研究中关于 NFV 资源分配问题主要针对 VNF 放置或 VNF 调度单独展开。但在实际的网络中，VNF 放置与 VNF 调度是相互影响，二者需要联合考虑的。此外，为了提高网络服务质量，服务的运行截止时间、节点处理时延、链路带宽资源和传输时延等约束都需要得到充分考虑。针对这些问题，本文进行了以下工作：

（1）针对延迟敏感的服务，本文提出了一种结合强化学习和自适应鲸鱼优化算法的新型算法，以实现服务功能链（Service Function Chain，SFC）上 VNF 放置和调度的最小化处理延迟，并确保满足 SFC 的约束需求。该算法通过自动调节鲸鱼优化算法的参数，并引入交叉、变异和邻域算子来提高算法性能，加速收敛过程，以达到最优解。仿真结果显示，与现有的 HGWOA 算法和 GABL 算法相比，新算法在性能上分别提升了 10%和 16%。

（2）在现有工作基础上，本文考虑了更符合实际网络应用场景的带宽资源需求、截止时间需求以及调度所需的时延，将 VNF 放置

与 VNF 调度的联合问题拆分为三个子问题，目标是在保证满足 SFC 资源需求的前提下，最小化 SFC 拒绝率。为此，本文提出了基于马尔可夫过程的 VNF 调度模型。该模型通过基于复合规则的 D3QN（Dueling Double DQN)算法，在每个调度时间点上选择 SFC 进行处理，并通过路由优化算法对虚拟节点和路由进行选择，从而实现最小化 SFC 拒绝率。仿真结果表明，与基于单一规则的调度策略、基于 DQN 的调度策略以及基于遗传算法的调度策略相比，该算法降低 SFC 的拒绝率约为 8%。

关键词：网络功能虚拟化，VNF 调度，强化学习，进化算法，深度强化学习

ABSTRACT

With the continuous development of network technology, 5G networks have been further upgraded compared to traditional networks, leading to a trend towards more diversified network services. Additionally, the growth of low-latency network application scenarios has exponential growth, meeting the diversifying needs of users has become an inevitable trend in the current telecommunications industry. To this end, the industry has introduced Software Defined Networking (SDN) technology. The main idea of SDN is to decouple hardware and software in proprietary network devices, allowing software to no longer depend on specific hardware devices, and to abstract the decoupled software as independent Virtual Network Functions (VNFs). This virtualization technology allows VNFs to be adaptively placed on physical network nodes as needed, enabling network nodes to provide the corresponding network functions according to actual demands. This not only improves network flexibility and meets quality of service requirements but also utilizes network resources effectively and reduces operational costs for operators.

Network Function Virtualization (NFV) issues, also known as NFV resource allocation problems, consist of VNF composition, VNF placement, and VNF scheduling. Current research on NFV resource allocation problems mainly focuses on VNF placement or VNF scheduling in isolation. However, in real networks, VNF placement and VNF scheduling are interdependent and need to be considered jointly. Moreover, to improve network service quality, constraints such as service runtime deadlines, node processing delays, link bandwidth resources, and transmission delays must be fully considered. In response to these issues, this thesis presents the following work:

(1) For latency-sensitive services, this thesis proposes a novel algorithm that combines reinforcement learning with an adaptive whale optimization algorithm to minimize processing delays for VNF placement and scheduling on Service Function Chains (SFCs) and to ensure that SFC constraints are met. The algorithm automatically adjusts the parameters of the whale optimization algorithm and introduces crossover, mutation, and

neighborhood operators to enhance performance and accelerate convergence to an optimal solution. Simulation results show that the new algorithm outperforms existing HGWOA and GABL algorithms by 10% and 16%, respectively.

(2) Building on the current work, this thesis considers the more realistic network application scenarios of bandwidth resource requirements, deadline demands, and scheduling delays, dividing the joint problem of VNF placement and VNF scheduling into three sub-problems. The goal is to minimize SFC rejection rates while ensuring that SFC resource requirements are met. To this end, this thesis proposes a VNF scheduling model based on Markov processes. The model uses a Dueling Double DQN (D3QN) algorithm with composite rules to select SFCs for processing at each scheduling time point and employs a routing optimization algorithm to select virtual nodes and routes, thereby minimizing SFC rejection rates. Simulation results show that compared to scheduling strategies based on single rules, DQN, and genetic algorithms, the proposed algorithm reduces the SFC rejection rate by approximately 8%.

KEY WORDS: network function virtualization, VNF scheduling, reinforcement learning, evolutionary algorithms, deep reinforcement learning

1 绪论

1.1 研究背景和意义

随着网络技术的不断进步，相较于传统网络，5G 网络[1]为用户提供了更多样化的服务应用场景，尤其在低延迟网络应用方面，如自动驾驶和虚拟现实等领域。然而，传统网络架构存在两个主要问题。首先，许多网络服务（如防火墙、WAN 优化器等）与所谓的“中间盒”[2]硬件紧密耦合。不同的网络功能需要对应的专用硬件支持，这导致网络功能的僵化和灵活度的降低，使得运营商在运营和资本支出方面面临巨大压力[3]。其次，这种网络模式无法迅速满足多样化的服务需求。

为了解决这些问题，现有研究在 5G 网络中引入了软件定义网络技术和网络功能虚拟化技术。SDN[4]的主要功能是将数据平面与控制平面分离，使得网络更具灵活性；而 NFV[5]技术则将专用设备中的硬件和软件解耦，使软件不再依赖于特定硬件设备。通过将解耦的软件抽象为独立的网络模块，即虚拟网络功能[6]，这些模块可以根据需求灵活地部署在物理网络节点上。这不仅提升了网络的灵活性和服务质量（Quality of Service, QoS），还实现了网络资源的合理利用，同时减少了对专用硬件设备的依赖，进而降低了运营商的运营和资本支出。

在 NFV 的架构中，VNF 实例按照一定顺序排列组合形成服务功能链 [7]，并通过在网络基础设施上部署这些 SFC 来提供网络服务。然而，在支持 NFV 的网络基础设施上配置 SFC 相对复杂，特别对于延迟敏感的 SFC（例如，触觉互联网服务），这些 SFC 需要按照特定顺序进行组合，并在严格的服务期限内完成[8]。为了满足这些严格的时间要求，服务提供商必须有效地将 SFC 的 VNF 放置在合适的网络节点上，并进行合理的 VNF 调度和流量路由，这一挑战被称为 NFV 资源分配（NFV Resource Allocation, NFV-RA）问题[9]。一般而言，NFV-RA 可以分为三个子问题：VNF 组合（VNF-Chain Composition, VNF-CC），VNF 放置（VNF-Placement, VNF-P），以及 VNF 调度（VNF-Scheduling, VNF-SCH）。第一个子问题是解决 SFC 的组成，第二个子问题则解决如何将 SFC 中的 VNF 放置到支持 NFV 的网络节点上，并将 VNF 之间的虚拟链路部署在底层链路上。第三个子问题则解决确定运行给定服务所需的 SFC 中 VNF 的执行方案。显然，在实际应用中，VNF 的放置和调度对网络的服务质量有着重大影响。因此，如何高效地解决这两个问题是当前的研究热点。

在此背景下，本文将 VNF 放置和 VNF 调度视为一个联合优化问题，并提出了两种解决方案：（1）针对延迟敏感的服务功能链，提出了一种基于强化学

习和自适应鲸鱼优化算法的方案。该算法在保证满足服务功能链约束的前提下，通过自动调节鲸鱼优化算法的参数，并利用交叉、变异和领域算子来提高算法性能，从而达到最优解，最小化 VNF 处理延迟；（2）将 VNF 放置与 VNF 调度的联合问题拆分为三个子问题，并建立基于马尔可夫过程的 VNF 调度模型。在满足服务功能链资源需求的情况下，以最小化 SFC 拒绝率为目标，通过基于复合规则的 D3QN 算法，在每个调度时间点上选择合适的 SFC 进行处理，并通过路由优化算法选择虚拟节点和路由，从而有效降低 SFC 拒绝率。总之，本文提出的解决方案不仅有效应对了 5G 网络中的服务需求和低延迟问题，还为提高网络资源利用效率和降低运营成本提供了重要的理论和实践参考，具有显著的理论和实际意义。

1.2 国内外研究现状

1.2.1 VNF 放置问题

在现有研究中，网络中的单个节点有能力承载多个网络功能。因此，当网络需要部署多个 VNF 时，如何合理地将它们放置在网络节点上，实现 VNF 的共享，从而降低运营成本，成为 VNF 放置问题中的一个重要挑战。目前，国内外学者已提出了许多模型和方法来解决这个问题。

线性规划是 VNF 放置问题中的常见解决方法之一，例如 Yanal 等人[10]将 VNF 放置问题描述成整数线性规划（Integer linear programming，ILP）模型，原型测试表明，所提出的策略能够提高网络切片的可用性和可靠性水平。Mohammad 等人[11]将问题映射为整数线性规划模型，并采用两步放置算法和基于禁忌搜索的算法，用于解决大规模和地理分布式 NFV 基础设施中 NFVO 和 VNFM 的放置问题。Hendrik Moens 等人[12]建立了 NFV 环境中虚拟化网络功能资源分配模型，通过特定软件实现了整数线性规划模型，并在具有定义硬件规格的服务器上进行了评估。Morin C 等人[13]通过利用整数线性规划并引入基于网络抽象的启发式方法来解决虚拟网络功能链放置问题。Ali Hmaity 等人[14]提出三种整数线性规划模型来解决 VNF 放置问题，并评估延迟对 VNF 分布的影响。Li 等人[15]为了解决云计算数据中心中 VNF 放置问题，将该问题构建为整数线性规划模型，提出两阶段启发式算法，实验表明该算法在网络资源利用率上具有更好的效果。Sun 等[16]在 VNF 放置问题的基础上，考虑端到端的延迟和带宽资源消耗，并使用基于广度优先搜索的优化算法去解决该问题。Hyodo 等人[17]将 VNF 放置问题表述为一个整数线性规划模型，最小化放置 VNF 和链路使用的成本，还引入了一种启发式算法用于求解，同时允许放宽 VNF 访问顺序约束和

配置。Alhussein 等人[18]将联合问题表述为混合整数线性规划（Mixed integer linear programming，MILP）问题，提出低复杂度启发式算法，并展示仿真结果以证明有效性。Hantao Guo 等人[19]在研究中使用的方法包括提出 MILP 模型和基于斯坦纳树问题和马尔可夫决策过程的启发式算法，实验结果表明，该启发式算法优于 MILP 的最优解。Feng 等人[20]引入了一种先进的启发式算法，涉及 VNF 迁移到另一个可用节点，有效地提高了 SFC 的利用率和接受率。Rankothge 等人[21]提出了两种 VNF 布局算法，以响应新的服务请求，并根据网络流量的变化调整 VNF 布局和位置。

1.2.2 VNF 调度问题

VNF 调度问题是指，当 VNF 被放置在节点上后，需尽可能短的时间内有效地处理它们，以确保在即定的截止时间内完成尽可能多的 SFC，或者最大限度地减少所有服务的完成时间。关于 VNF 调度问题也有大量研究，例如，Riera 等人[22]最初将 VNF 调度问题公式化为车间调度问题，并在没有给出多项式时间解的情况下提出了其数学模型。Qian 等人[23]介绍了一种分组调度算法，该算法考虑了分组队列和 SFC 链的特点，具有较低的复杂性。Chen 和 Wu[24]设计了一个处理和延迟模型，捕捉中间盒中处理流的通信延迟行为，同时开发了两种相应的启发式调度算法。Yi 等人[25]提出了一种基于最小加代数理论的 VNF 调度新模型。该模型通过在不同业务之间共享已部署的 VNF 实例，从而提高资源利用率。Assi 等人[26]提出了一种有效且节能的 VNF 布局和调度方法，利用启发式算法来解决该问题。

除了上述方法，在现有的研究中也有许多进化算法来解决调度问题。例如：Pezzella 等人[27]中使用遗传算法来解决车间调度问题。Mijumbi 等人[28]阐述了在线虚拟函数放置和调度问题，并提出了三种贪婪算法和一种基于禁忌搜索的启发式算法。Qu 等人[29]将 VNF 调度以及相应的资源优化问题描述为混合整数线性规划问题。以最大限度地减少整个 VNF 时间表的完成时间/延迟为目标，提出了一种遗传算法的解决方案，结果表明，能够有效满足具有严格延迟要求的服务，从而增加运营商的收入。Pham 等人[30]提出了一种基于匹配的算法来解决 NP-Hard 的 VNF 调度问题，这种方法可以保证稳定调度，并且能使所有网络服务都得到满意的分配。

VNF 调度问题也是一个组合优化问题，在现有文献中，强化学习已被用于解决组合优化问题。例如，Li 等人[31]提出了一种强化学习算法来确定每个决策状态下的可变动作集，捕捉动作的不同执行时间，以实现感知延迟的 VNF 调度。Bello 等人[32]介绍了一种使用神经网络和强化学习来解决组合优化问题的框架。Lei 等人[33]将车间调度问题解释为由独立学习代理处理的序列决策问题。智能体

通过利用概率调度策略，以提高由标准调度目标函数测量的联合策略的性能。Luo S 等人[34]通过建立深度 Q 网络来解决动态柔性车间调度问题。Liu 等人[35]提出了一种分层分布式结构来解决动态调度问题，引入了专门的状态和动作表示来处理动态调度中的可变规范。此外，Liu 等人[35]还开发了一种可供选择的奖励成形技术，以提高学习效率和调度效率。虽然上述车间调度问题与 VNF 调度有一些相似之处，但它们没有考虑机器之间的传输延迟，这是一个显著的区别。

1.2.3 VNF 组合优化问题

在现有的大多数研究中，虚拟网络功能的放置和调度通常是分开进行的，这种方法难以满足严格的服务要求。由于服务功能链在网络功能虚拟化环境中动态到达，且 VNF 的放置位置未定，各节点在 CPU 计算能力和存储资源上存在差异。因此，将 VNF 放置在不同节点上会导致 SFC 完成时间的显著差异。显然，VNF 的放置和调度是紧密相关的，只有综合考虑这两方面并进行全局优化，才能真正满足现代网络的性能需求和质量要求。这不仅是技术发展的方向，也是 NFV 实际应用中面临的重大挑战之一。

为了解决 VNF 调度与放置的联合问题，已有大量研究提供了各种解决方案。例如，Wang 等人[36]在他们的研究中，着重于支持 NFV 的移动边缘计算系统中 SFC 的调度问题，并将其建模为灵活的车间调度问题，以最小化总体调度延迟为目标，提出了一种基于深度强化学习的算法——深度 Q 网络调度。Riggio 等人[37]则将无线接入网络中的无线 VNF 放置问题形式化为整数线性规划问题，提出了一种称为“无线网络嵌入”的 VNF 放置启发式方法来解决这个问题。Yao 等人[38]设计了一种在线抢占算法，通过调度多个 SFC 以最小化其完成时间，验证了该算法的有效性。Wang 等人[39]将 SFC 的组成和放置问题结合在一起，建模为整数规划问题，同时开发了一种称为 Jcap 的启发式方法，在两个阶段内解决问题，实验结果显示，该方法有效提高了 VNF 实例化的利用率，并减少了链路资源的消耗。Dos Santos 等人[40]提出了一种算法，能够同时处理 VNF 的按需放置和调度，同时考虑软件需求，以利用传统的分布式算法来实现高效调度。

除了这些方法之外，还有一些启发式算法和强化学习算法被用来解决 VNF 组合优化问题。例如，Gamal M 等人[41]通过最大化可接受的服务请求数量、最小化瓶颈链路数量和总体处理时间，解决了一个多目标优化问题，并开发了两种基于遗传算法的新算法。Cao 等人[42]专注于 6G 网络中的 NFV 联合 SFC 嵌入和调度问题，提出了一种新的 SFC 嵌入和调度算法，该算法优先选择资源能力更强、资源丰富的物理节点来承载 SFC。Kuai 等人[43]则联合考虑了给定网络服务的 VNF 放置和调度，以优化最大-最小公平性，同时确保不同服务链的延迟要求，提出了一种基于离线近端策略优化的深度强化学习方法。

1.2.4 研究现状小结

综上所述，大多数实验都采用了启发式算法，它们的性能很大程度上取决于问题的复杂性，对于复杂问题，它们可能过早收敛，陷入局部最优。为此，将启发式算法用于解决 VNF 放置与调度问题，算法可能会随着网络规模的扩大而恶化。例如，粒子群优化和遗传算法等元启发式算法在迭代过程中可能会遇到收敛速度较低的问题，这导致计算成本和运行时间增加。

此外，严格的端到端时延要求对于 5G 业务的开通也非常重要。不同端到端服务的延迟约束结合 VNF 调度问题也带来了额外的挑战，而在上述国内外文献中并未充分讲述延迟约束问题，例如在节点和链路上的处理时间、服务截止时间和链路带宽资源需求等。

最后，VNF 调度与放置是网络功能虚拟化中的两个重要方面，它们共同决定了 VNF 在网络中的部署位置和资源分配。VNF 的调度涉及到决定何时以及如何迁移 VNF 实例，而放置则关注 VNF 应该放置在何处。这两个方面相互依赖，互相影响，它们的决策会直接影响到网络性能、资源利用率和运营成本。因此，上述文献中，将 VNF 放置 VNF 调度问题分开讨论并不能很好的满足严格的服务要求。

1.3 研究内容及创新

本文主要研究了 VNF 调度和 VNF 放置的联合优化问题，图 1-1 为章节结构图，具体内容如下：

第一章，介绍了本文的背景与意义，以及详细介绍了 VNF 放置、VNF 调度以及它们联合优化的国内外研究现状，同时指出了现有研究存在的不足之处。

第二章，对相关的技术背景进行了概述，详细介绍了进化算法理论和深度强化学习理论，为后续章节内容做好铺垫。

第三章，针对 VNF 放置与调度联合问题，以最小化最长完成时间为目标，提出了以强化学习自调节鲸鱼优化算法，并通过实验验证了该算法的优势。

第四章，为更好的满足在现实情况中服务时间和链路带宽资源限制等约束，将 VNF 放置与调度问题拆分为多个子问题逐一解决，并建立基于马尔可夫决策过程的 VNF 调度模型，采用基于复合规则的 D3QN 算法来最小化 SFC 的拒绝率。最后，通过实验验证了该算法的优势。

本文的创新之处如下：

（1）针对元启发式算法解决 VNF 放置与调度联合问题时的不足，本文开发了将鲸鱼优化算法与强化学习相结合算法。该算法利用强化学习自动调节鲸鱼

优化算法的参数，且综合使用交叉、变异和领域算子来提高算法性能，加快收敛速度，从而寻找最优解。仿真结果表明，与 HGWOA 算法和 GABL 算法相比，本文的方法性能提升了 10%和 16%。

（2）针对时延敏感的服务，本文充分考虑端到端服务截止时间、带宽资源需求和处理/传输时间等约束，提出一种 VNF 放置与调度联合算法，旨在最大限度地降低 SFC 的拒绝率。该算法通过使用基于复合规则的深度强化学习算法与启发式算法相结合解决 VNF 放置、VNF 调度以及节点选择和路由优化等问题，以完成最小化 SFC 拒绝率。最后，本文将所提出的算法与基于单一规则的调度策略、基于 DQN 的调度策略以及基于遗传算法的调度策略相比。仿真结果表明，与遗传算法相比，所提出的算法可以将 SFC 的拒绝率降低约 8%。

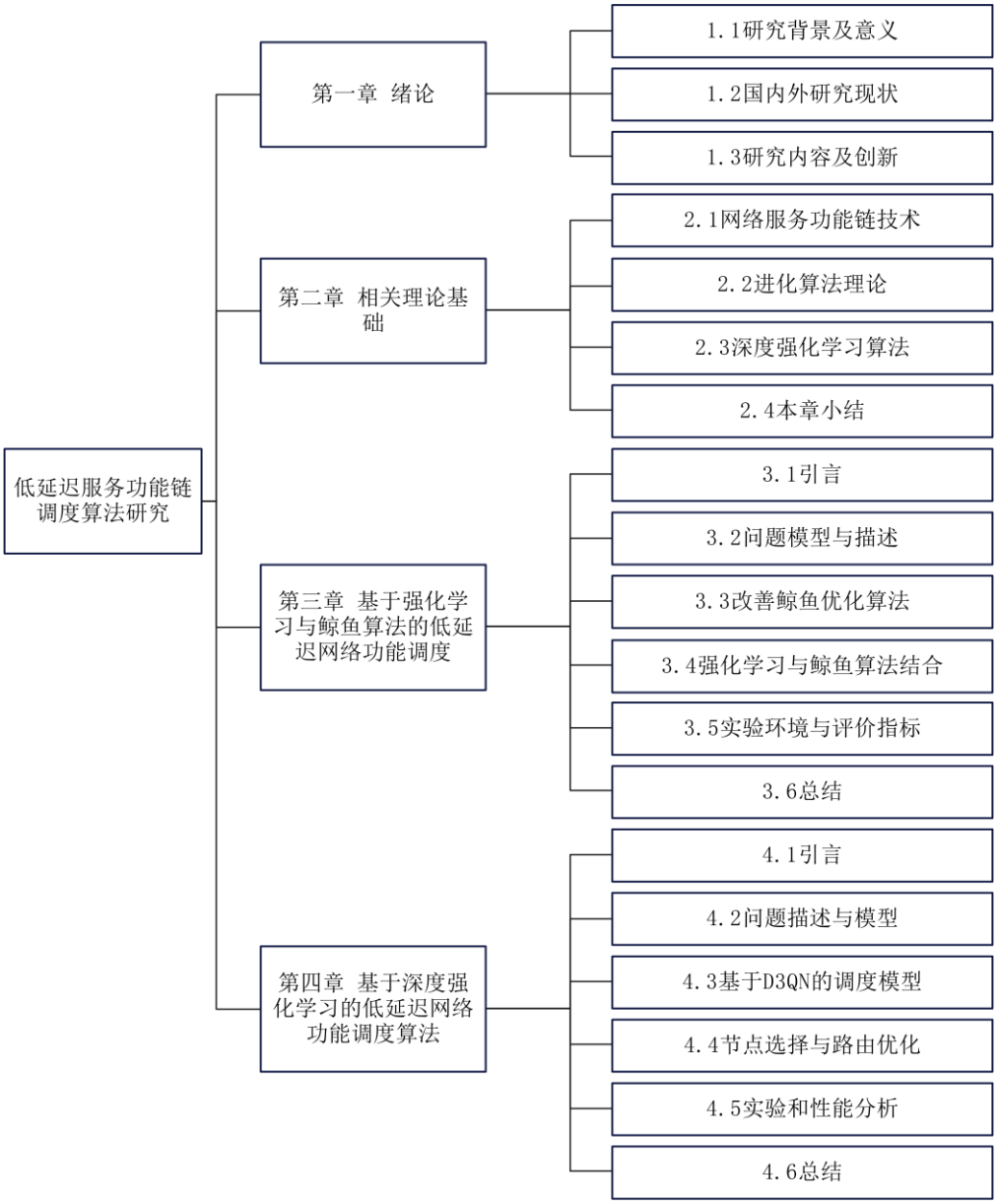

图 1-1 章节结构图

2 相关理论基础

2.1 网络服务功能链技术

2.1.1 软件定义网络

软件定义网络是一种全新的网络架构[44]，旨在通过中央化的网络控制方式来提高网络的灵活性和可编程性。SDN 通过将网络的控制层（决定数据如何在网络中流动）与数据层（实际携带网络用户数据的部分）分离，使网络管理更加简便。这种分离允许管理员从中央位置动态调整网络流量，以满足不断变化的需求，而无需物理接触各个网络设备。SDN 主要是由控制器，南向接口，北向接口和网络设备组成。控制器是 SDN 的大脑，作为一个能够运行在网络中的任何标准服务器上的软件应用程序，主要负责下达网络流量管理的决策，它能通过南向接口实现与下层网络设备的通信功能，同时能够通过北向接口给上层应用提供编程接口。南向接口能够通过通信协议实现控制层和数据层之间的数据传输，其中最广泛使用的协议是 OpenFlow[45]，它使得控制平面可以与数据平面分离。北向接口能够使应用层与控制层进行交互，允许应用层通过控制器编程来请求网络服务，例如路由、负载均衡或防火墙服务。网络设备则是指在 SDN 环境中，扮演数据层的角色（如交换机和路由器），负责数据的实际转发。它们根据控制器下发的指令来转发流量。相比传统的网络，SDN 通过其极高的灵活性和集中化管理能力，实现了对网络的动态控制和优化，从而提升了网络的可扩展性、可管理性和服务质量。此外，SDN 提供了更好的网络资源利用率，能够有效降低运营成本，并促进创新，因为新的网络功能可以作为软件更新迅速部署，而不需要更换硬件。同时 SDN 的集中控制还可以改善网络的可视化和分析，从而提高安全性和性能监控。

2.1.2 网络功能虚拟化

网络功能虚拟化是通信行业的一项创新技术，它在应对网络运营商因依赖专有硬件而面临的高成本、空间限制以及资源管理等方面非常有效。在传统的网络中，网络服务依赖于大量专用硬件，每增加一个服务就意味着新的硬件投入，导致采购、安装、运维和能源开销过多，以及管理这些硬件所需的专业培训成本。这种做法不仅缩短了硬件的生命周期，还增加了经济负担。专有设备过时的速度迫使运营商陷入一个购买与部署循环，这在快速提升的互联网服务

市场中显得尤为不利。为了解决上述传统网络问题，业界提出了网络功能虚拟化技术。

网络功能虚拟化为运营商提供了灵活性，通过与 SDN 相结合，将传统的网络功能如路由、防火墙、负载均衡器等软件化，让运营商能够购买或自主开发可在物理服务器上运行的网络功能软件。这类软件设计上旨在高效地运行于标准化硬件之上，实现从专用硬件向更开放的硬件环境过渡。选择在虚拟化环境中部署这些网络功能，可以为电信服务提供商带来额外的好处，包括资源使用的灵活性、提高能源利用效率，以及系统可伸缩性和敏捷性的增强。通过这种方式，NFV 成为了推进通信行业朝着更加灵活和成本效益更高方向发展的关键驱动力。简而言之，NFV 的实施为运营商开辟了一条能够快速响应市场变化、促进创新，同时确保服务品质和顾客满意度的新途径。

其次，欧洲电信标准组织 ETSI 发布了 NFV 参考架构[46-48]，这使得 NFV 更加的标准化，同时也得到更好的发展，NFV 体系架构如图 2-1 所示：

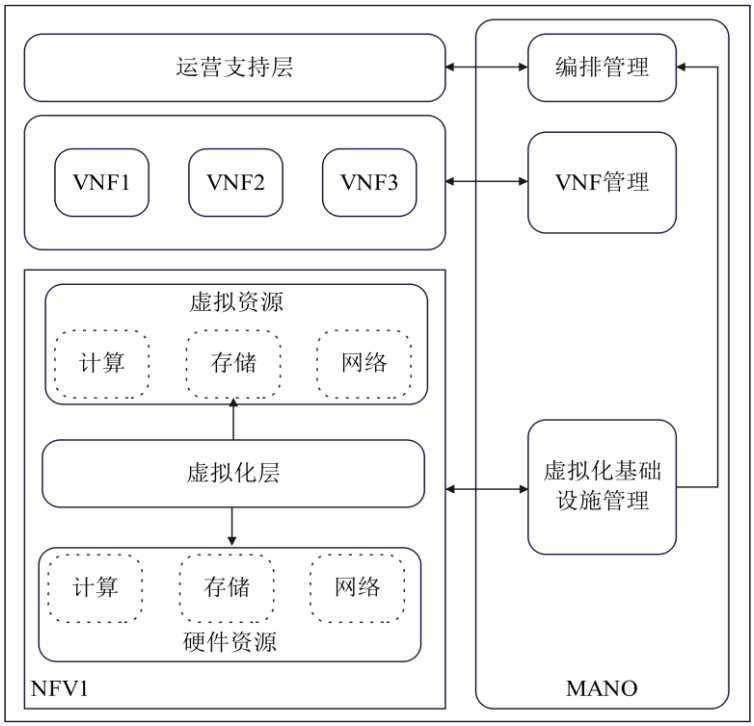

图 2-1 NFV 体系架构

可以看到 NFV 构架主要是由虚拟网络功能，网络功能虚拟化设施（NFV Infrastructure, NFVI）和网络功能虚拟化管理编排器（Management and Orchestration, MANO）[49]组成的。虚拟网络功能，是用来提供网络服务，如防火墙、负载均衡器或 WAN 加速器。在图 2-1 中，多个 VNF 并列排列，代表它们可以在虚拟化环境中灵活部署。网络功能虚拟化设施是 NFV 环境的核心，包含了支撑虚拟网络运行的全部硬件和软件资源。这些资源不仅包括数据中心与云服务（无论是公有、私有还是混合云）之间的网络互联，还涉及到计算、存

储和网络这三大硬件资源。这些硬件通过虚拟化层为虚拟网络功能提供必要的处理能力、存储空间和网络连接。虚拟化层本身位于硬件之上，它逻辑上隔离和分配物理资源给 VNF，从而实现资源的高效利用。管理与编排主要负责协调和自动化虚拟网络功能以及虚拟化基础设施的关键组件。MANO 涵盖了 VNF 的部署、配置、管理和优化，确保它们能够在 NFVI 上高效运行。此外，它通过与传统的网络管理系统集成，支持复杂网络服务的全生命周期管理，实现资源的动态分配、故障恢复和服务保障，从而推动网络服务提供商实现更高效、灵活的网络运营和创新。

2.1.3 NFV 资源分配

在 NFV 架构中，协调器通过控制虚拟网络功能管理器和虚拟基础设施管理器来评估物理节点是否具备执行虚拟网络功能的条件，从而进行虚拟网络资源的分配。NFV 资源分配通常分为三个阶段：

(1) VNF 的组合：

VNF 的组合是指将一系列虚拟网络功能按照特定的顺序排列组合，形成一个完整的端到端网络服务流程，这被称为服务功能链。在这个过程中，需要选择合适的 VNF（例如防火墙、负载均衡器、WAN 优化器等），并决定它们在虚拟网络中的执行顺序。这个步骤的目的是确保数据包能够按照预定的路径依次通过每个网络功能，得到所需的处理。这一阶段的主要挑战在于如何高效地设计 VNF 链，以满足服务质量需求和特定的业务目标。

(2) VNF 的放置：

VNF 的放置是指在第一阶段 VNF 链组合完成的基础上，选择最佳的物理或虚拟资源位置来部署这些虚拟网络功能。正如图 2-2 所示，VNF 的放置是一个 NP-Hard 问题，其主要任务是如何在网络功能虚拟化环境中，决定每个 VNF 的最佳放置位置，以便在满足服务质量要求的同时，优化网络资源的使用，降低延迟，提高系统的整体性能和效率。在实际应用中，这一过程需要综合考虑网络节点的计算能力、存储容量、网络带宽等资源，以及网络拓扑结构和服务需求的动态变化。

在图 2-2 中，可以看到一个通过网络功能虚拟化基础设施层实现的端到端的服务功能链。这个服务链由一系列虚拟网络功能构成，包括防火墙、负载均衡器、加密、数据包检查和解密模块，这些 VNF 被按顺序放置到虚拟化的物理资源上。整个过程由编排器管理，它负责在虚拟资源中嵌入和协调这些 VNF，确保网络服务从一端到另一端的顺畅运行。通过虚拟化层，物理资源被抽象化，允许 VNF 独立于具体的硬件平台运行，而编排器通过标准化接口确保了 VNF 的生命周期管理和网络服务的整体性能。

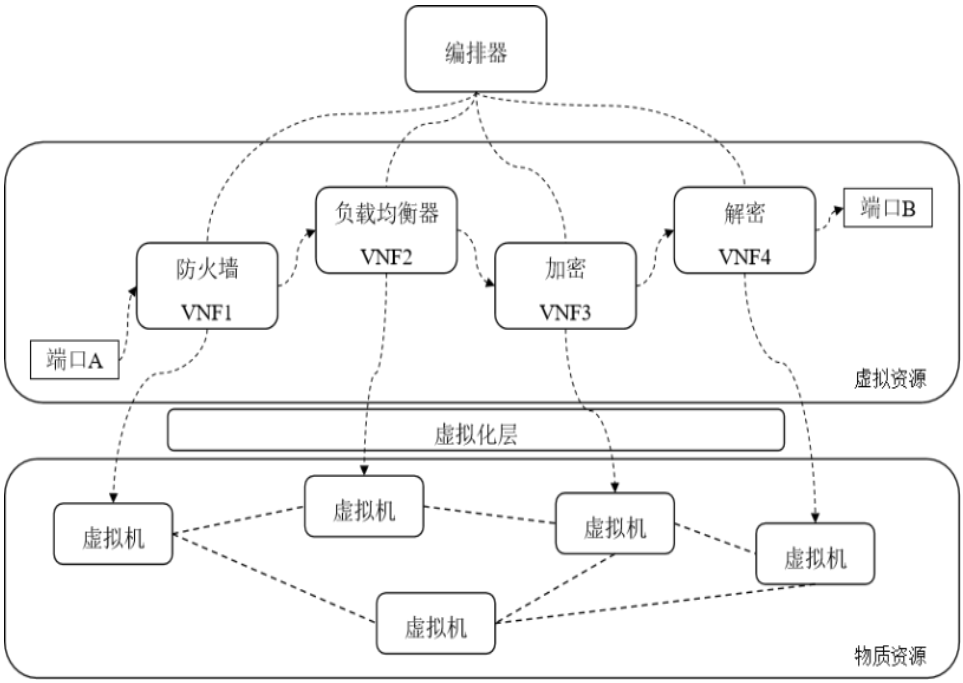

图 2-2 VNF 的放置

(3) VNF 调度：

VNF 调度是网络功能虚拟化环境中的一个关键过程，它涉及到如何有效地管理、配置和优化虚拟网络功能在虚拟化基础设施上的部署。这个过程不仅需要考虑资源的可用性和网络服务的需求，还要确保满足特定的性能指标，如响应时间和吞吐量。因此，应该对 VNF 的执行进行适当的调度，以最大限度地减少网络服务的总执行时间，从而获得更高的性能。

在整个资源分配过程中，NFV 的成功实现依赖于服务功能链的有效资源分配，其中 VNF 调度是关键的最终步骤，它在整个流程中扮演着不可或缺的角色。大部分研究都选择在 VNF 放置完成后，对 VNF 调度进行独立研究。通常采用一种固定的方法，将每个 VNF 直接绑定到一个具体的物理节点上，这种一一对应的方法，可能会限制优化的可能性，导致无法找到最佳解，也不符合实际情况。为了有效的资源分配策略能够显著提高网络的效率、降低运营成本，并提升用户体验，本研究放宽了 VNF 与虚拟机（VM，Virtual Machine）之间的限制，将 VNF 放置和调度视为一个联合问题，对这两个问题进行了综合考虑和研究。

2.2 进化算法理论

服务功能链的放置和调度联合问题是一个 NP-Hard 问题[50]，因此在该领域大部分的工作都是集中在启发式或者元启发式算法的设计，以及在求解混合整数线性规划模型时使用最小复杂度的网络，例如比较常用的有禁忌搜索算法、

遗传算法、鲸鱼优化算法、人工蜂群算法等，但单一的算法由于存在容易陷入局部最优解和收敛速度慢等问题，所以 VNF 的联合问题不能被很好的解决。为此，混合算法的提出是将两种或两种以上的优化算法结合起来，利用各自的优点来改进搜索过程的一种方法，这种方法可以有效地增加算法的多样性，减少被局部最优解困扰的机会。例如，可以将遗传算法（优秀于全局搜索）与梯度下降法（优秀于局部搜索）结合，以期望在全局搜索能力和局部搜索精度之间取得平衡。本文通过将遗传算法与鲸鱼算法相结合来解决联合调度问题。章节 2.2.1 和章节 2.2.2 将分别介绍遗传算法与鲸鱼算法。

2.2.1 遗传算法

遗传算法(Genetic Algorithm，GA) [51]是 1975 年由美国 Michigan 大学的 Holland 教授及其科研团队根据达尔文进化论中“优胜劣汰”原则提出的一套完整的理论和方法，它以其独特的并行搜索能力和强大的全局优化性能著称，能够在广泛的搜索空间中自动寻找到最优或近似最优解。这使得它在多个领域中发挥着重要作用，特别是在组合优化、机器学习和控制系统设计等方面，已成为现代人工智能技术的重要支柱。在遗传算法中有 5 个重要的参数包括：

(1) 种群大小：指的是每一代中解的数量。

(2) 交叉概率：交叉率决定了从当前种群中选取多少个体进行交叉（配对和重组）以产生后代。高交叉率促进了基因的多样性，但过高的交叉率可能导致好的解被破坏。

(3) 变异概率：变异率是种群中个体发生变异的概率。变异可以引入新的基因变体到种群中，增加多样性。太低的变异率可能导致算法过早收敛于局部最优解，而太高的变异率则可能导致搜索过程随机化，降低算法效率。

(4) 选择方法：选择方法决定了如何从当前种群中选取个体以进行交叉和变异。常见的选择方法包括轮盘赌选择、锦标赛选择和排名选择等，各有优缺点，影响算法的收敛速度和能力。

(5) 精英策略：精英策略是指将当前种群中的一部分最优个体直接保留到下一代。这可以保证解的质量不会因为遗传操作而降低，有助于保留已找到的最优解。

图 2-3 展示了遗传算法的过程。

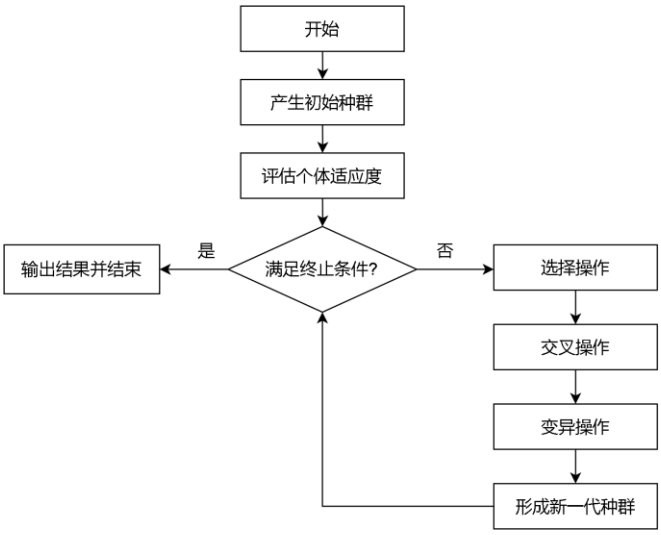

图 2-3 遗传算法流程图

2.2.2 鲸鱼优化算法

鲸鱼优化算法（Whale Optimization Algorithm, WOA）是由 Mirjalili 等人[52]在 2016 年提出的新型群体智能优化算法，灵感来源于座头鲸的独特捕食行为，学习了座头鲸的“气泡网捕食法”策略。该策略通过吐出气泡圈来包围猎物并逐渐将其引向水面，从而更容易捕捉。WOA 算法正是利用这种行为的数学模型来解决优化问题。

(1) 包围捕食：

在鲸鱼捕食的过程中，由于在搜索空间中最优设计的位置并非事先已知，WOA 算法假设当前的最佳候选解即为目标猎物或接近最优解。在确定了最佳搜索代理后，其他搜索代理将努力更新它们的位置并逐渐靠近最佳搜索代理。这一行为可以通过以下方程表示：

$$X(t+1)=X^{*}(t)-A\times|C\times X^{*}(t)-X(t)| \tag{2-1}$$

其中 A 和 C 为系数向量 $X^{*}(t)$ 为当前最优解 $X(t)$ 为当前解，A 和 C 的计算公式如下：

$$A=2a\times r-a, C=2\times r \tag{2-2}$$

其中 a 为收敛因子在迭代过程中从 2 线性下降到 0，r 为[0,1]中的随机数。

(2) 气泡攻击：

对于气泡攻击来说主要有两种方式分别是收缩包围机制（其本质上与包围捕食相同）和螺旋更新方式，其方式是通过计算鲸鱼与最优位置的距离创建的一个螺旋等式其公式如下：

$$X\left(t+1\right)=X^{*}\left(t\right)+e^{bl}\times\cos(2\pi l)\times\left|X^{*}\left(t\right)-X\left(t\right)\right| \tag{2-3}$$

鲸鱼在一个不断缩小的圆圈内绕着猎物游动，同时沿着螺旋形路径游动。为了对这种同时发生的行为进行模拟，假设有 50%的可能性在收缩包围机制和螺旋模型之间进行选择，以便在优化过程中更新鲸鱼的位置，则当概率 p 小于 50%时对应公式（2-1），当概率大于 50%时对应公式（2-3）。

(3) 搜索猎物：

与包围机制不同的是对于搜索机制来说主要是由系数 A 来确定，当|A|>1 时鲸鱼以一种随机的方式通过彼此的位置搜索猎物，不再选择猎物来更新自身位置。当|A|≤1 时它们采用了一种群体策略，随机选取群体中的个体，用其替代原先的猎物。这一策略的效果是迫使鲸鱼们远离猎物所在的位置，从而增强 WOA 算法的整体全局寻优能力。其公式如下：

$$X(t+1) = X_{rand} - A \times |C \times X_{rand} - X| \qquad (2\text{-}4)$$

其中 X_{rand} 是随机鲸鱼位置向量。主要步骤及流程如图 2-4 所示：

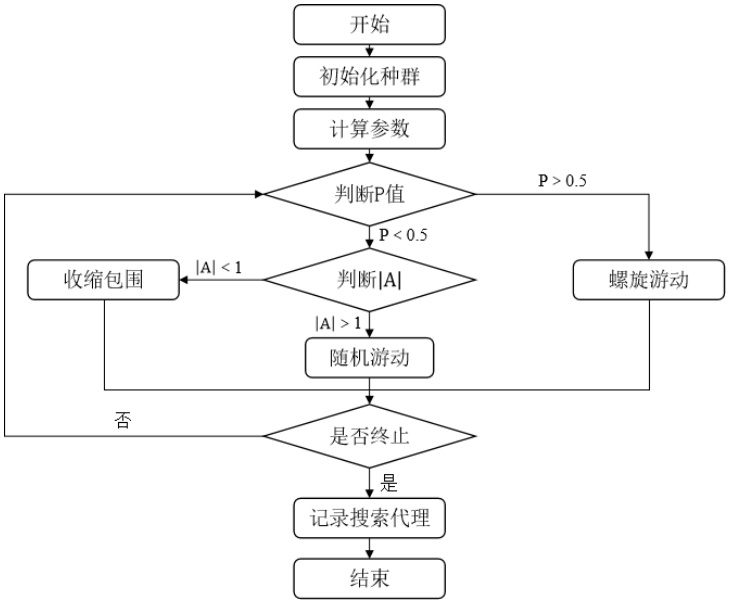

图 2-4 鲸鱼算法流图

步骤 1：设定鲸鱼群体的数量和算法的最大迭代次数 t_{max}。对每条鲸鱼，初始化其在搜索空间中的位置。

步骤 2：计算每条鲸鱼的适应度，并记录当前找到的最优鲸鱼位置。

步骤 3：计算控制搜索行为的参数 a、p 以及系数向量 A 和 C。如果随机概率 p 小于 0.5，则执行气泡网捕食策略，否则，进入下一步。

步骤 4：判断|A|是否小于 1，如果是则模拟座头鲸包围猎物的行为，根据相关公式更新鲸鱼的位置。否则，则通过全局搜索策略更新鲸鱼的位置，模拟鲸鱼探索猎物的行为。

步骤 5：更新所有鲸鱼的位置后，再次评估它们的适应度。如果发现更优的解，则用这个新的最优解替换之前保留的最优解。

步骤 6：检查是否达到了最大迭代次数 t_{max}。如果是，则算法结束，输出当

前最优解。否则，返回步骤 3 继续迭代。

2.3 深度强化学习算法

近年来，强化学习（Reinforcement Learning，RL）[53]技术在人工智能和机器学习等领域得到了广泛的研究和关注。强化学习是一种独特的机器学习方法，其核心原理是通过智能体与环境的持续交互来学习最佳的行为策略。

在强化学习中，智能体根据当前环境的状态做出动作，并将这个动作反馈给环境。环境会根据智能体的动作变化，产生一个新的状态和相应的奖励。如果智能体的动作带来了正面的奖励，这一动作将在后续的决策中被优先考虑，从而增加智能体重复该动作的倾向；相反，如果动作的结果是负面的，智能体则会降低选择该动作的可能性。通过这种不断试验和调整的过程，智能体逐渐学会在各种环境状态下做出最佳决策，从而最大化其累积奖励。

强化学习的目标是找到一个策略，即在给定状态下选择动作的规则，使得智能体能够获得最大的长期奖励。这种学习过程通常通过值函数或策略梯度的方法来实现。深度强化学习（Deep Reinforcement Learning，DRL）则是强化学习与深度学习的结合，它利用深度神经网络来处理高维状态和动作空间，从而应对复杂的环境和决策问题。

DRL 的应用领域非常广泛，从游戏中的自动化决策到机器人控制，再到资源管理和优化问题，其表现出了强大的适应能力和学习能力。在网络功能虚拟化和服务功能链优化中，DRL 也展现出了巨大的潜力，能够有效地解决资源分配、路径优化和延迟管理等复杂问题。

通过这种智能体与环境的持续交互和学习机制，强化学习不仅可以解决传统方法难以处理的复杂决策问题，还为自适应和自主学习系统提供了强有力的支持。这使得深度强化学习成为了当前机器学习领域中一个备受瞩目的研究方向。

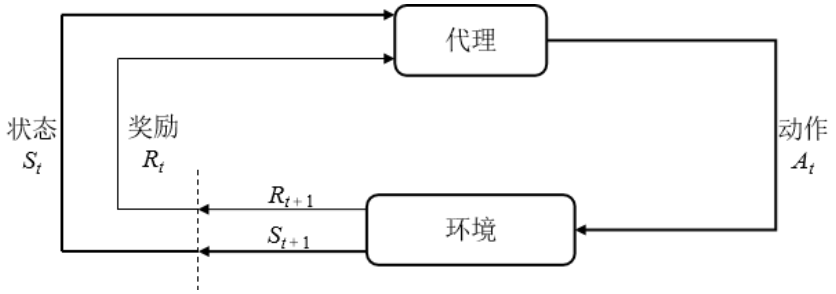

图 2-5 强化学习流程

强化学习执行流程如图 2-5 所示，代理作用是观察环境并获得状态 S_t，依据它的决策对状态做出相应的动作 A_t，此时能得到一个奖励 R_t，同时，因为环境

发生改变，代理会得到一个新的状态，并继续执行下去。

2.3.1 Q-learning

Q-learning 学习是强化学习中的一种，在网络部署、网络路由[54-57]等领域进行了充分应用，其目标是通过迭代更新估计的 Q 值来学习最优策略，从而在每个状态下选择最优的动作。Q 值保存在 Q 值表中，用于记录智能体的学习经历。初始 Q 值表是通过算法生成的零值矩阵，其行数等于状态数量，列数等于动作数量。Q 值的计算是综合考虑智能体经历的状态、选择的动作和获得的奖励，具体公式如下：

$$Q(s_t,a_t)=Q(s_t,a_t)+\alpha[r_{t+1}+\gamma\max_a Q(s_{t+1},a_{t+1})-Q(s_t,a_t)] \quad (2\text{-}5)$$

其中 $Q(S_t,a_t)$ 是在 S_t 状态下执行动作 a_t 的估值，α 是学习速率（0 到 1 之间的值），表示新信息相对于旧信息的权重 r_{t+1} 表示执行动作 a_t 后的及时奖励，γ 为折扣因子（0 到 1 之间的值），表示对未来奖励的重视程度，$\max_a Q(s_{t+1},a_{t+1})$ 则表示在执行动作 a_{t+1} 后 Q 表中状态 s_{t+1} 的最高期望 Q 值。

在传统 Q-learning 算法中，一个表格（Q-table）被用来存储每个状态的 Q 值，这在状态空间较小时是可行的。然而，面对复杂问题时，状态空间的急剧膨胀使得大量状态无法被有效观察和学习，导致学习过程耗时且难以收敛。为此 DQN (Deep Q-Learning)算法通过 Q-Learning 算法与神经网络相结合，有效解决了这一问题。

2.3.2 DQN

DQN 的概念首先由 Mnih 等人[58]提出，它可以被称为具有权重的神经网络函数逼近器。DQN 通过将原始数据直接（状态特征）输入，每个状态-动作对的函数值作为输出，从而可以处理状态空间大且连续的复杂决策过程。DQN 的训练的和进步主要体现在以下两个方面。第一，在 DQN 模型中通过策略 π 与环境交互选择出最优动作，再由动作作用于环境当中形成一个新的环境并获得奖励，这也是四元组的形成：$<S,A,R,S_{t+1}>$。该四元组将会被储存在经验池中以便用于 DQN 的学习，当经验池的容量满时，旧的经验也会被新的经验所取代，通过多次转换来更新参数，因此也可以实现更好的数据效率。第二便是目标网络，在目标网络中每次训练都会更新网络参数，使其在训练中更加的稳定。

DQN 网络计算动作价值公式如下所示[59]，其中 γ 为折扣因子[0,1]，a' 为学习率，θ' 为网络参数。

$$y_i=r_i+\gamma a' maxQ(s',a';\theta') \quad (2\text{-}6)$$

2.3.3 Double DQN

在传统的 DQN 中它可以处理那些大而连续状态的空间的复杂的决策过程，但是它也存在一些问题，例如过估计[60]。过估计问题的原因在于，神经网络的学习过程中，可能会出现偏差和方差的问题。偏差指的是模型本身的拟合能力不足，无法准确地拟合真实的 Q 值函数。方差是模型在训练过程中对于训练数据过度拟合，从而导致对于未知数据的泛化能力不足。这使得估计的值函数比真实值函数要大，从而实际最差的值可能变得估计最好，而实际最好的值可能变得估计最差。为了避免这个问题，可以采用 Double DQN 来解决[61]。Double DQN 是在 DQN 中新加入了一个新网络，其结构和原始的网络相同，但是使用了不同的网络参数。这两个网络的用处有所不同，原来的网络用来控制智能体收集学习到的经验和选择动作，而新的网络用来计算动作的价值，这种选择与评估的解耦减少了高估，使得学习更加的稳定，其公式如下：

$$y_i = r_i + \gamma Q(s', arga' maxQ(s', a'; \theta); \theta') \tag{2-7}$$

2.3.4 Dueling DQN

在 DQN 中神经网络输出的是代理选择动作的价值，但是单纯对动作价值的评估会导致输出不准确。因为动作的价值 $Q(S,A)$ 是和状态和动作有关，但这二者之间对价值的影响力度不一样，因此 Dueling DQN 算法从网络结构上改进了 DQN，神经网络输出的动作价值函数可以分为状态价值函数和优势函数，这样的设计有助于模型学习更加准确地区分每个状态的价值和每个动作相对于平均动作的优势，而不是简单地评估 Q 值[62]。其公式如下：

$$Q_\pi(s,a) = V_\pi(\mathrm{s}) + A_\pi(s,a) \tag{2-8}$$

其中 $V_\pi(\mathrm{s})$ 表示状态价值函数，它等于该状态下所有动作概率的平均值即所有动作值乘以其概率的总和，这反映了环境状态本身的价值。$A_\pi(s,a)$ 优势函数的目的是衡量在特定状态下，一个动作相对于平均动作的优势。理论上，如果一个动作的优势值高，那么这个动作在当前状态下比其他动作具有选择优势，而 $Q_\pi(s,a)$ 值表示为该状态下的动作值，那么优势函数 $A_\pi(s,a) = Q_\pi(s,a) - V_\pi(\mathrm{s})$ 为每个动作对应其平均高或差多少，因此那些比平均值大的动作值会更高而差的就会更低能够加快网络收敛速度。结合上述描述，Dueling DQN 能够更有效地学习到那些对于执行特定动作来说价值不大（即优势值较低）的状态，同时更准确地评估那些对于采取任何动作都有高期望回报的状态。这种区分使得 Dueling DQN 在很多任务中相比传统 DQN 和其他变体表现更好，特别是在需要

精细评估状态价值和动作价值的环境中。综上所述，采用 D3QN,作为本文网络模型进行训练效果会更好。

2.4 本章小节

本章深入探讨了SDN和NFV这两个在现代网络技术中不可或缺的构成部分，并详细陈述了NFV-RA问题。其次对遗传算法、鲸鱼优化算法进行了说明，然后阐述了深度强化学习算法的框架，了解到D3QN在解决复杂问题时的有效性。通过结合SDN的网络流量控制、NFV的功能虚拟化以及SFC的服务定制能力，现代网络架构能够更加灵活地响应变化多端的业务需求，同时降低运营成本并加速服务的创新。

3 基于强化学习与鲸鱼算法的低延迟网络功能调度

3.1 引言

网络服务通常由多种虚拟网络功能组成，例如防火墙、WAN 优化器等。这些虚拟网络功能需要按照特定的顺序依次部署在相应的节点上进行处理，才能完成网络服务。然而，随着信息化时代的到来，网络服务的数量急剧增加。如果没有一个有效的 VNF 放置和调度方案，网络服务的流量可能会堵塞，从而导致网络服务质量的下降。因此，当多个网络服务同时请求物理网络资源时，合理的 VNF 放置和调度方案显得尤为关键。

尽管当前已有大量关于 VNF 放置和 VNF 调度的研究，但大多将这两个问题分开处理并未联合考虑。然而，这种方法在实际应用中并不理想。VNF 的放置和调度是相互影响的动态过程，将它们分开处理，难以满足网络服务的质量要求。此外，一些研究没有充分考虑 VNF 在网络链路上的传输延迟，这未能反映实际情况中的复杂性。

为了应对上述挑战，本文提出了一种基于强化学习的改进鲸鱼算法。该算法的主要改进有以下三点：1）多样性提升：采用 POX 交叉和负载均衡变异的策略，提高可行解的多样性，防止算法陷入局部最优；2）动态参数调整：利用强化学习技术，根据环境的实时变化动态调节鲸鱼算法的参数，从而加快求解速度和提高算法效率。3）性能优势：通过实验研究表明，本文提出的改进算法与已有的算法相比性能上分别提升了 10%和 16%。这说明该算法能够提供更有效的 VNF 调度方案，并显著提升了网络服务质量。

本章的结构安排如下：第 3.2 节详细描述了问题的背景及其建模；第 3.3 节介绍了改进的鲸鱼优化算法；第 3.4 节讨论了如何将强化学习与鲸鱼算法相结合；第 3.5 节进行了一系列实验，并与现有算法进行对比分析，以验证改进算法的优越性；第 3.6 节对本章内容进行总结。

3.2 问题模型与表述

3.2.1 问题描述

虽然在现有的 NFV 资源分配问题中，主要是由 VNF 组成、VNF 放置和 VNF 调度这三个问题组成，但结合实际资源与服务功能链约束来看，对 QoS 影响最大的还是 VNF 放置和 VNF 调度问题。

VNF 的组成是利用流量虚拟化带来的灵活性，根据客户发起的功能需求，将不同的 VNF 功能按照不同的顺序，组成相对应的服务功能链，而这些 SFC 都有执行的顺序性和依赖性，即在一条 SFC 链中只有当前一个 VNF 处理完后，才能开始下一个 VNF 的处理，例如，假设有一条 SFC 链由 VNF1，VNF2，VNF3，VNF4 这 4 个 VNF 依次按顺序组成，则该链的执行顺序为 VNF1→VNF2→VNF3→VNF4。

由上述 VNF 端到端的组成完成后，便作为 VNF 放置输入。VNF 放置寻求在考虑一组请求的 SFC 的情况下，根据 VNF 和网络节点中的约束条件，以合适的方式在网络基础设施中分配 SFC 中的 VNF，将其放置到网络节点中。

一般而言 VNF 调度是在 VNF 组成和 VNF 放置阶段完成基础上进行的，其目标是在满足 SFC 的约束条件的基础上，最小化整体 SFC 的完成时间。为了更好的理解这个过程，本文首先假设在虚拟网络中有 3 个支持 NFV 的节点和由 3 条虚拟链路组成，网络拓扑如图 3-1 所示：

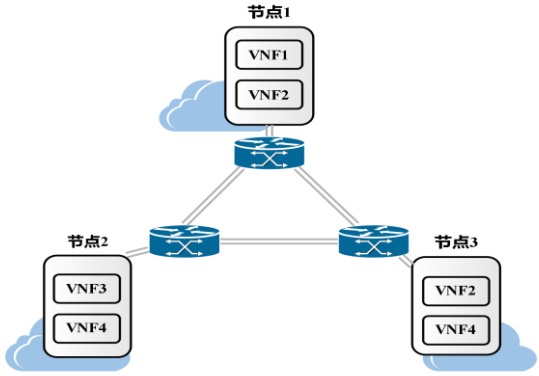

图 3-1 网络拓扑

假设有三条 SFC 进入网络进行调度，分别是 SFC1：VNF1→VNF3，SFC2：VNF2→VNF3，SFC3：VNF1→VNF4，同时 VNFi 必须要放置在具有 VNFi 功能的节点上才能正常运行，其中 SFC1 与 SFC3 的每个 VNF 处理时间为 1，SFC1 的传输时间为 1，SFC3 的传输时间为 2，SFC2 的每个 VNF 处理时间为 2，传输时间为 3。假设这 3 条 SFC 已经完成了放置阶段，VNF 放置处理结果如图 3-2 所示：

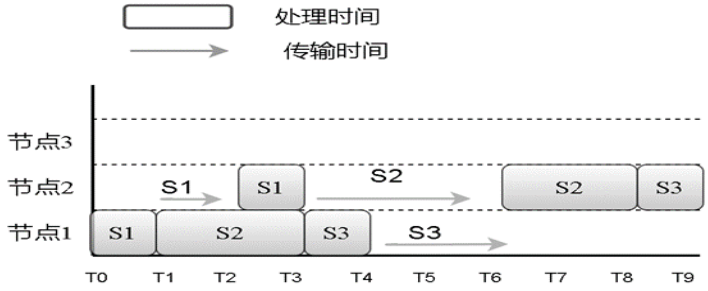

图 3-2 VNF 放置

此时完成时间为 T=9，此时需要对这 3 条 SFC 进行调度来减少整体的完成时间，例如 VNF 调度处理结果如图 3-3 所示：

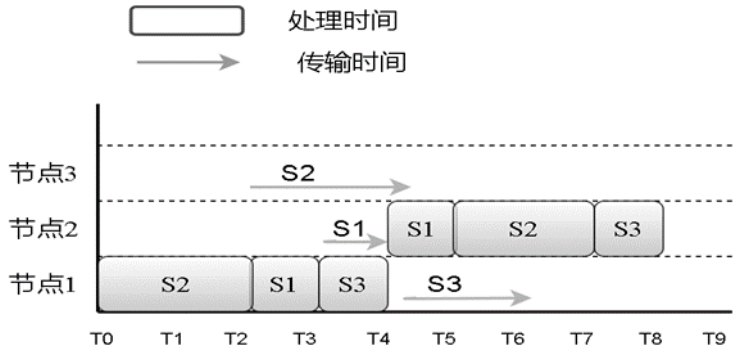

图 3-3 基于放置的 VNF 调度

可见，在结合 VNF 放置阶段的基础上能有效减少整体完成时间。如果将 VNF 放置与 VNF 调度进行结合，那么整体完成时间将减少至 T=7 时效果更好，例如 VNF 放置与调度联合处理结果如图 3-4 所示：

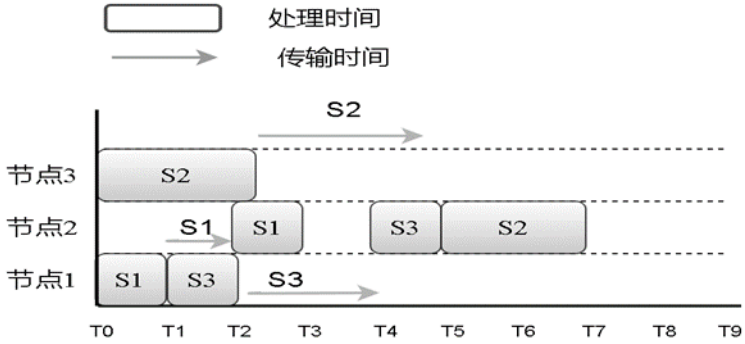

图 3-4 VNF 放置与调度联合

综上所述，单纯的在原来 VNF 放置后的基础上进行调度无法得到较好的效果，只有当 VNF 放置与调度相结合，即在 VNF 放置阶段就开始考虑到调度问题才能最大化效益，下一小节将对网络模型及其约束条件进行描述。

3.2.2 网络模型

在物理网络图 G(N,L)中 N 表示为虚拟节点用来托管和运行不同类型的 VNF，L 表示为连接每两个节点之间的虚拟链路。现有一组 SFC，并且每个 SFC 都由 j 个不同功能的 VNF 组成，其中 SFC 上的 VNF 都得按照一定顺序进行调度，同时每个 VNF 只能在一个节点上进行实例化操作，而每个节点在同一时刻也只能对一个 VNF 进行处理，即在节点处理 VNF 过程中不能中断。模型参数说明如下表 3-1 所示：

表 3-1 参数及意义

参数	意义
S	表示 SFC 的数量
V_s	表示 SFC 上 VNF 的个数
n	表示第 n 个节点
S_j^i	表示第 i 个 SFC 的第 j 个 VNF
f	表示 VNF 的类型
δ	表示时间戳
$f_{S_j^i}$	表示虚拟网络功能 S_j^i 的类型
$p_{S_j^i}^n$	表示虚拟网络功能 S_j^i 在节点 N 上的处理时延
$T_{S_j^i}^{S_{j+1}^i}$	表示虚拟网络功能 S_j^i 与 S_{j+1}^i 的传输时延
$K_{S_j^i}^{n,\delta} \in (0,1)$	表示在第 δ 时间戳时第 i 个 SFC 的第 j 个 VNF 是否在节点 N 上运行，是为 1，否则为 0

本文优化目标及其所需要的约束如下：

(1) 确保 SFC 上的 VNF 类型与放置对象节点上的 VNF 实例的类型相同：

$$\sum_{n\in N,\ s\in S} K_{S_j^i}^{n,\delta} f = f_{S_j^i} \tag{3-1}$$

(2) 表示当 SFC 的第 j 个 VNF 还在处理时第 j+1 个 VNF 不能处理：

$$\sum_{n\in N,\ s\in S} K_{S_j^i}^{n,\delta} \le 1 - \sum_{n'\in N, s\in S} K_{S_{j+1}^i}^{n,\delta} \tag{3-2}$$

(3) 表示当节点的 VNF 实例正在处理当前 SFC 的 VNF 时，不能再处理其他 SFC 的 VNF：

$$K_{S_j^i}^{n,\delta} = 1 - K_{S_{j'}^{i'}}^{n,\delta}, n \in N \tag{3-3}$$

最大完成时间 T 如下所示：

$$T = \sum_{i=1}^{S}\sum_{j=1}^{V_s} K_{S_j^i}^{n,\delta}(p_{S_j^i}^n + T_{S_j^i}^{S_{j+1}^i}) \tag{3-4}$$

为此在满足以上约束的同时，本文的目的是最小化完成时间 $\min(T)$ 。

3.3 改进鲸鱼优化算法

为了提高鲸鱼优化算法整体局部搜索能力和收敛速度，本文通过编码与解码、种群初始化、交叉与变异和领域结构等几个方面进行改进。

3.3.1 编码与解码

由于在网络系统中 SFC 的数量较大，使得在 VNF 调度变得更加复杂。为此，在用 WOA 解决调度问题时，处理好编码和解码非常关键，一个好的编码可以减少不可行解的产生，缩短代码的运行时间，提高整体的效率。本文采用的是分段式编码过程，由机器选择（Machine Selection，MS）和操作选择（Operations Sequencing，OS）两个部分组成一个完整的编码，OS 存放所需处

理 VNF 的排序向量，MS 是每个 VNF 的节点选择的向量，编码如图 3-5 所示，图中有 3 条 SFC，每条 SFC 都有两个 VNF 组成，其中 O(3,1)表示为第 3 个 SFC 中的第 1 个 VNF 最先处理，选择的节点则对应这 SFC3 区域段的第 1 块，即为 1 号节点。

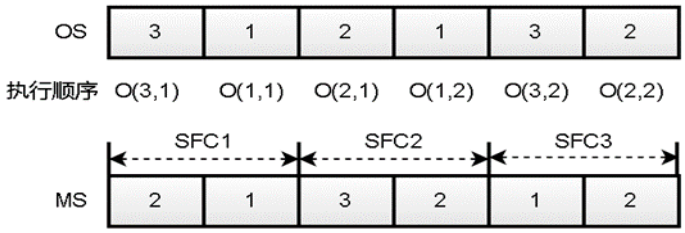

图 3-5 编码图

在传统解码过程中，由于并非每个解向量都对应不同的处理时间，解空间中的 VNF 与节点处理时间之间仍然存在着多对一的放置关系。改进的解码方法可以将结果放置到更好、更少的点，显著缩小搜索空间，减少不可行解的产生。为此，本文引用一种插入式的贪婪解码算法[63]，其主要是根据 OS 编码部分通过从左向右依次进行解码，同时在不移动已经排布好的 VNF 的前提下，插入到对应节点上最佳时刻安排处理，以此方式直至所有的 VNF 排布完成。贪婪解码过程如图 3-6 所示，假设通过编码，解码后，S3 的第二个 VNF 是在 S1 之后完成的，而进行贪婪的解码将 S3 调度到 S2 之前这样可以减少整体的完成时间。

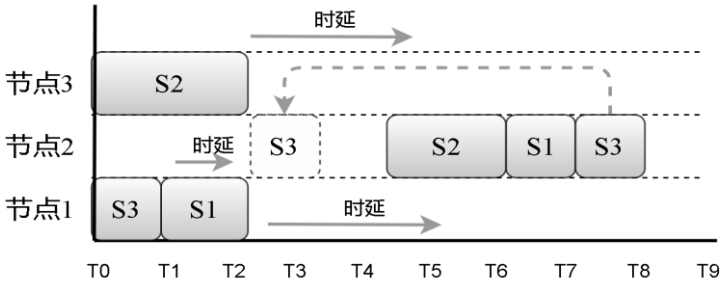

图 3-6 贪婪解码

3.3.2 种群初始化

在进化算法中，种群的初始化质量对算法的求解速度与质量有极大的影响。大部分文献采取的随机初始化方法使得初始解的分布随机，机器负载不均匀，种群质量欠佳，从而导致需要更多次迭代与较大的种群规模才能达到较为理想的目标值。为避免浪费大量的计算资源，本文使用了一种 GLR 初始化方法。该方法的 MS 编码包括如下 3 个部分：全局选择(Global selection, GS)用于保证加工机器上的负载平衡；局部选择(Local selection, LS)保证单个工序加工时间最短或

者优先选择机器负荷最小的机器进行加工；随机选择(Random selection, RS)保证种群的多样性。OS 编码全部由随机编码构成。

3.3.3 交叉与变异

在改进搜索的过程中本文引用了遗传算法中的交叉与变异来保持可行解的多样性，同时本文通过使用领域结构来提高局部搜索能力[64]。

交叉：针对不同的编码层采用不同的交叉方法。对于 OS 编码层使用 POX 交叉对于 MS 编码层使用多点交叉的方式进行编码，通过以上的 OS 和 MS 的交叉操作，可以提高算法的全局搜索能力，并且保证交叉后的解任然是可行解，避免产生不合法的解。

变异：为解决变异的问题同时使节点负载均衡化来减少完成时间，本文在变异操作阶段设计了针对网络中节点的负载平衡变异策略，其中节点负载率定义如公式（3-5）所示。

$$L_k = \frac{\sum_k \sum_i P_{ik} X_{ik}}{T} \tag{3-5}$$

L_k 为负载率，X_{ik} 为在节点 K 上所有的处理 VNF 数，P_{ik} 表示在 K 上每个 VNF 的处理时间，T 为整体的最大完成时间。操作过程如下表格 3-2 所示：

表格 3-2　变异流程

步骤	描述
步骤 1	对鲸鱼个体进行解码，获得其目标函数及各节点负载率。
步骤 2	对最高负载率节点上的 VNF 进行时间上的排序，并选择处理时间最长的 VNF。
步骤 3	查看当前工序可选节点集合，对可选节点的负载进行排序。
步骤 4	在 OS 编码不变的情况下，改变 MS 中当前 VNF 对应的节点，重新解码求得新的 MS 编码。
步骤 5	如果新的 MS 编码降低了负载高节点的负载率，则保存新的 MS 编码；否则保持原本的 MS 编码不变。
步骤 6	选择下一个高负载 VNF，重复步骤 3、4 和步骤 5，直至该节点上的所有 VNF 都被遍历完成。
步骤 7	输出保存的新的 MS 编码，实现负载均衡并可能降低整体的完成时间。

3.3.4 领域结构

为了更好的探索局部搜索的准确性和有效性，本文采用变邻域搜索策略来提高当前最优个体的质量。同时使用了三种领域结构：N1：在 OS 排序部分中任选两个元素，所选元素需对应不同 SFC 的 VNF，然后对所选元素进行交换操作。N2：在 OS 排序部分中任选两个元素，然后将后一元素插入到前一元素之前的位置。N3：在 MS 分配部分任选 1 至 3 个元素，该元素对应工序的可选节点数应多于 1，然后将该 VNF 分配至不同节点上。在每次邻域搜索操作之后评估新的调度解决方案。如果新的调度方案比原来的调度方案更好，则将新的调度方案设置为原来的调度方案。领域结构具体过程如表格 3-3 所示：

表格 3-3 领域流程

步骤	描述
步骤 1.设置参数	设置参数 λ、λ_{max}、i、i_{max} 和 X，其中 λ_{max} 为最大迭代次数，λ 为当前迭代次数 i 和 i_{max} 为参数分别为 1 和 3，X 为当前最优解。
步骤 2. 进行循环	对 i 进行循环，当 i 为 1 时使用 N1 领域搜索，i 为 2 时使用 N2 领域搜索，i 为 3 时使用 N3 领域搜索，如果新的调度解优于 X 则替代，否则 i +1。
步骤 3. 判断条件	如果 $i > i_{max}$ 则 t +1 并回到步骤 2 继续进行循环。
步骤 4. 结束条件	如果 $\lambda > \lambda_{max}$ 则结束。

3.4 强化学习与鲸鱼算法结合

近年来，强化学习技术在人工智能、机器学习等领域中得到了广泛的研究与关注，强化学习是一种特殊的机器学习方法，它主要工作原理是通过当前环境作为输入给智能体，在通过输出动作反馈给环境，在这个智能体与环境的不断交互的过程中将会产生一些奖励，一个好的动作将会产生一些不错得奖励来驱使智能体在后续选择中加强这个动作的趋势，反之则会减弱，以此来达到最佳决策的目的。

Q-learning 学习作为强化学习中的一员，具体公式如下：

$$Q(s_t,a_t)=Q(s_t,a_t)+a[r_{t+1}+\gamma \max_{a} Q(s_{t+1},a_{t+1})-Q(s_t,a_t)] \quad (3\text{-}6)$$

由前面第 2 章节鲸鱼优化算法介绍中得知，在传统的鲸鱼优化算法中有两个很重要的参数分别是 P 和 A，这两个参数均涉及于鲸鱼优化算法的全局搜索和局部搜索能力之间的平衡，但在传统鲸鱼优化算法中，参数 P 采用的是一种

等概率阈值法，即阈值为 0.5 并通过与随机生成 0 到 1 之间的值比较来进行策略选择，而这种等概率的策略选择方式可能会使得鲸鱼无法选择适合当前群体的捕食策略，使得算法出现收敛速度慢、陷入局部最优等问题；其次就是参数 A，它与收敛因子 a 线性相关，而传统鲸鱼优化算法的收敛因子 a 随迭代次数的增加线性递减，这种线性的变化方式不利于算法有效协调全局和局部搜索的能力。

为此，为了解决上述两个问题，本文采用了基于强化学习的鲸鱼改进算法，通过环境的改进和智能体的判断，自适应的改变 P 值和 A 值来更好的协调全局搜索和局部搜索能力，同时提高收敛速度。下一小节中首先给出了强化学习的状态特征设置，然后依次介绍了动作和奖励设置，最后介绍了运行流程。

3.4.1 状态、动作及奖励设置

(1) 状态设置：

在 VNF 调度过程中本文的目标是最小化完成时间，为此本文将以种群的适应度值和负载率为重点进行 4 个状态设置：

(1) 种群的平均适应度值：

$$f = \frac{\sum_{i=1}^{N} f(x_t^i)}{N} \tag{3-7}$$

(2) 种群的最佳适应度值：

$$f_{\max} = \max f(x_t) \tag{3-8}$$

(3) 种群的负载标准差：

$$L_{std} = \sqrt{\frac{\sum_{k=1}^{m} (Lk - Lave)^2}{m}} \tag{3-9}$$

(4) 种群的多样性：

$$D = \sum_{i=1}^{N} \left| f(x_t^i) - \frac{\sum_{j=1}^{N} f(x_t^j)}{N} \right| \tag{3-10}$$

其中 $f(x_t^j)$ 表示第t代的种群中第i个个体的适应度值， $\max f(x_t)$ 表示第t代种

群中最优适应度的个体，同时为了提高网络的稳定性和收敛性，本文将数据进行归一化处理，具体公式为 $x_{new}=\frac{x-x_{min}}{x_{max}-x_{min}}$，其中 x_{new} 为归一化后的新值，x、x_{min}、x_{max}，分别为所需归一化值，x 中的最小值，x 中的最大适应度值。由于种群的平均适应度值和负载标准差能更好的反应整体的种群状态，来提高整体的种群质量从而获得最佳的个体，因此W1 与W4 之间的比重应比W2 与W3 略大一点W1=W4=0.3，W2=W3=0.2，W1+W2+W3+W4=1。

(2) 动作设置：

本文设置一组由两部分组成的动作,由于鲸鱼算法参数 P 是随机由 0 到 1 之间生成的一个值，为此对于 P 而言本文设置动作为在 0 到 1 范围内生成 10 个动作每个动作间隔为 0.1，其次对于 A 的动作来说，虽然在鲸鱼优化算法中 A 为线性收敛的，但其本身还是与固定值 1 相比来进行判断，为此，本文将其设置为一个大小为 2 的动作集合分别是[-1，1]。

(3) 奖励设置：

由于目标是最小化完成时间，因此需要考虑当前状态 t+1 和上一状态 t 之间变化，通过变化设置奖励告知智能体哪些行动能够获得更高的奖励，其中主要是通过最佳种群适应度 f_{max}，平均适应度值 f 来定义。通过不同时刻种群适应度与平均适应度发生改变来比较计算奖励值，奖励计算过程如算法 3-1 所示：

算法 3-1 奖励设置

输入：f_{max}，f

输出：Reward

```
if f_max(t+1) < f_max(t) then
    Reward +=10
else if f_max(t+1) = f_max(t) then
    if f(t+1) < f(t) then
        Reward +=5
    else if f(t+1) = f(t) then
        Reward +=0
    else
        Reward −=5
    end if
else
    Reward −=10
end if
```

3.4.2 算法流程

基于上述改进的鲸鱼和强化学习结合整体的运行流程如下算法 3-2 和图 3-7 所示：

在算法 3-2 中，第 1 至 3 行，执行参数初始化，随后算法进入主循环。

第 4 至 6 行，通过强化学习对当前状态进行分析，进而生成动作，并据此调整鲸鱼算法中的参数 P 和 A。同时，随机生成参数 Pr。

第 7 至 16 行，根据参数 P、A 和 Pr 的值来选择相应的操作执行。

第 17 行，会比较整体的适应度，如果发现适应度有所降低，则进行替换。

第 18 至 21 行，进行整体参数的更新，为下一次循环迭代做准备。

算法 3-2 RLWOA 流程

输入：State

输出：State+1，Reward，Q-table 更新

1 初始化鲸鱼算法参数，初始化 RL 中的状态和动作
2 初始化种群并根据公式计算种群适应度，负载标准差，形成状态 State
3 **While** $t \le t_{max}$ **do**
4 通过 State 状态选择动作
5 通过动作生成 P 和 A 值
6 生成 0 至 1 之间的随机数 Pr
7 **if** P_r<P **then**
8 **if** A= −1 **then**
9 随机个体与新个体进行交叉
10 **end if**
11 **if** A=1 **then**
12 选取全局最优个体与新个体交叉
13 **end if**
14 **else**
15 对当前个体进行负载变异
16 **end if**
17 对全局最优个体进行局部搜索若适应度值减小则替换
18 更新状态 State+1
19 计算奖励 Reward
20 更新 Q 值表
21 t=t+1
22 **end**

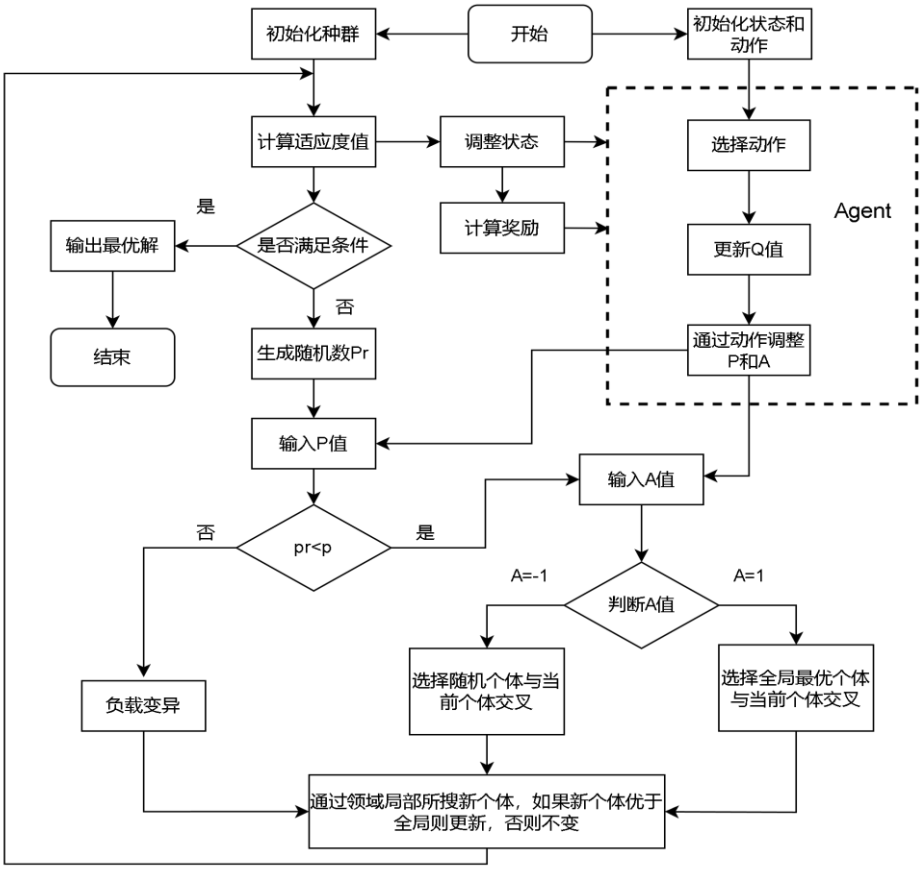

图 3-7 RLWOA 流程图

3.5 实验环境与评价指标

3.5.1 参数设置

为了测试该算法的能够有效的求解虚拟网络功能调度问题，本文将与文献[65]中的 GABL 算法和文献[66]中的 HGWOA 算法进行比较，种群数量设置为 100，迭代次数设置为 50，其次强化学习折扣率 γ 设置为 0.2，学习率 α 设置为 0.75，贪婪率为 0.8。本实验在 windows10 系统，i5 处理器上，均使用 Matlab 编写仿真程序实验运行。

3.5.2 实验分析

设置了 6 个全连接的虚拟网络节点，并且生成 10 条服务功能链进行调度，每条服务功能链由 4 至 6 条 VNF 组成，节点的处理时延为 1 至 6ms，链路的传输时延为 1 至 5ms，先将该 10 条服务功能链放入网络中进行虚拟网络功能。图 3-8 至图 3-10 分别是 RLWOA、GABL 和 HGWOA 中 SFC 的整体时延。

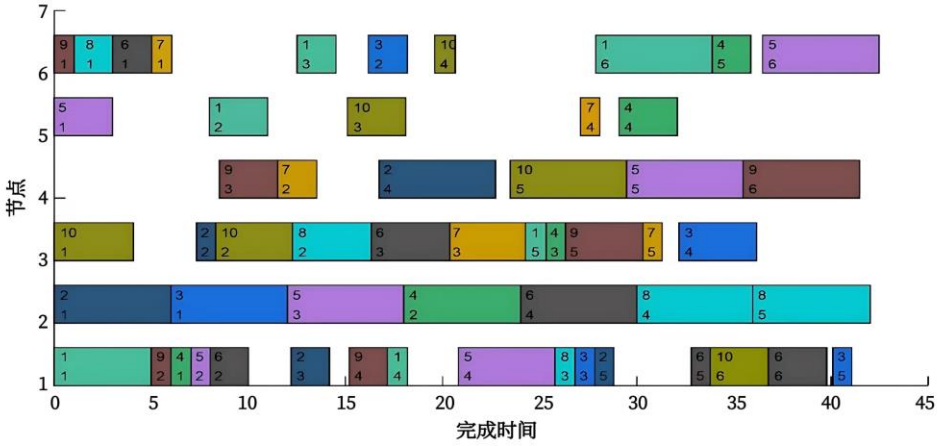

图 3-8 RLWOA 的完成时间

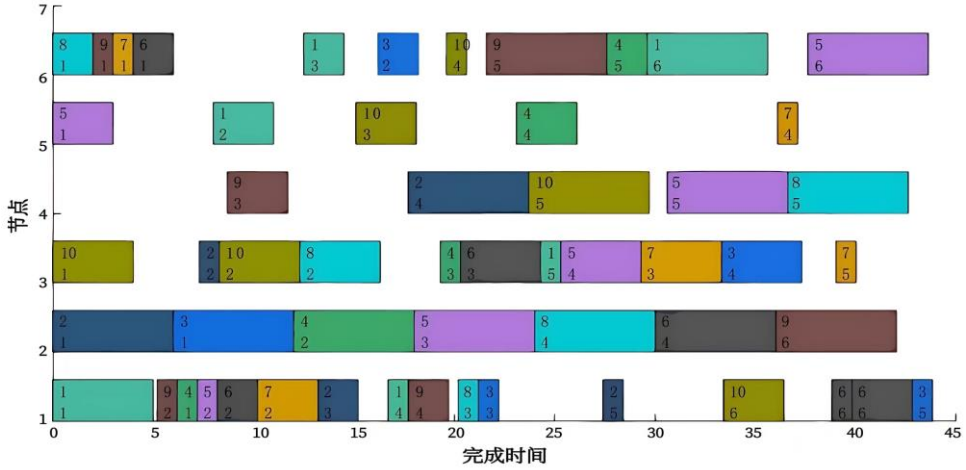

图 3-9 GABL 完成时间

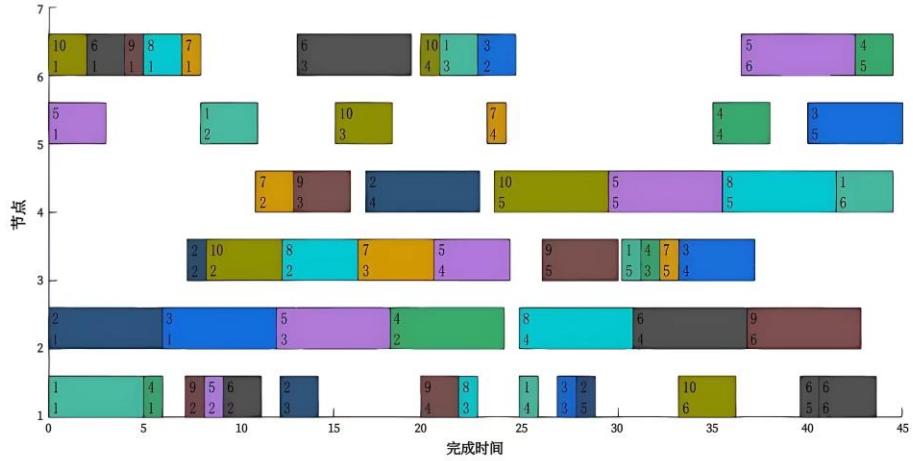

图 3-10 HGWOA 完成时间

图 3-8 至图 3-10 中 X 轴表示整体 SFC 的最大完成时间，Y 轴表示虚拟节点，上述方块中不同颜色表示不同的 SFC，其中每个方块上的数字表示第几个 SFC 中的第几个 VNF，例如左上方为 1，左下方为 1，则表示为第 1 条 SFC 的第 1 个 VNF，从图中可以看出使用 RLWOA 时最低的完成时延是 42.5ms，GABL 时最低的完成时延是 45ms，HGWOA 的最低完成时延是 43.8ms，这表明使用 RLWOA 进行虚拟网络功能调度时效果将会更好。

3.5.3 算法迭代比较

以下将显示三种算法在实验过程种的迭代图，算法迭代对比如图 3-11 所示：

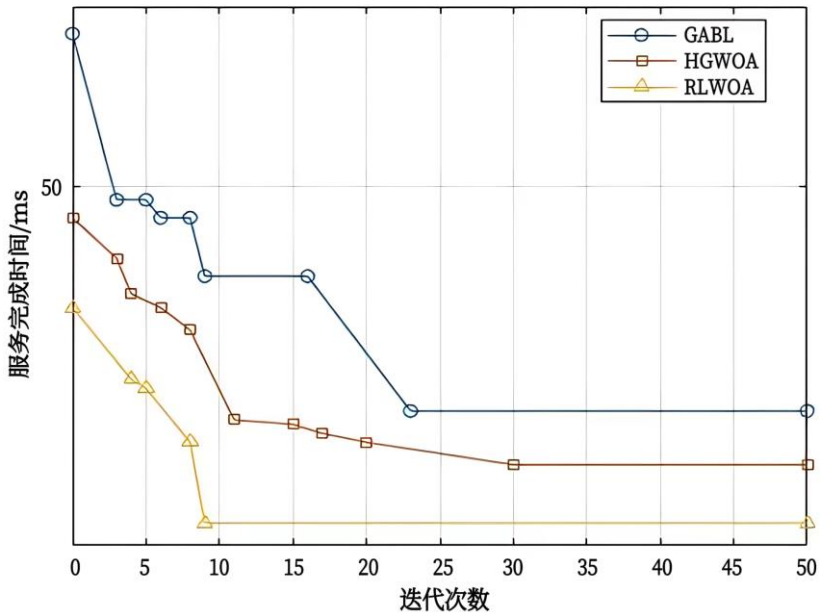

图 3-11 3 种算法迭代对比

从图 3-11 中，当 X 轴值为 0 时，可以看出本文提出的 RLWOA 算法与 GABL 算法和 HGWOA 算法相比值最低，说明 RLWOA 算法具有较好的初始解，也可以表明该种群初始化过程较好。此外，不难看出使用 RLWOA 能够有更好的收敛效率，说明使用强化学习进行训练能够加快收敛速度，同时避免陷入局部最优，找到更优的解。

除上述实验以外本文将采用文献[67]的网络结构进行模拟实验。首先将网络由 20 个节点全连接组成，服务功能链数量设置为 10，20，30 等至 100 个，同时有 10 个不同的虚拟网络功能，组成上述服务功能链，每个服务功能链最多由 10 个虚拟网络功能组成，且每个虚拟网络功能可以在 1 至 4 个节点上进行实例化。其次，其节点的上虚拟网络功能的处理时间为 2 至 10ms，节点与节点之

前的传输时间具有随机行为 1 至 10ms，算法比较结果如下图 3-12 所示：

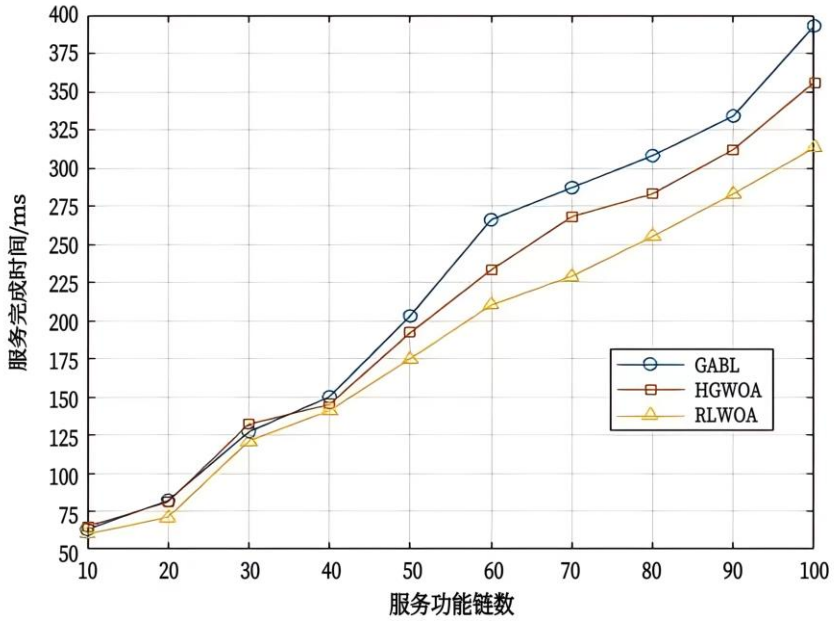

图 3-12 不同服务功能链下算法比较

从图 3-12 中可以看出随着服务功能链的不断增加 RLWOA 算法依旧保持着较好的调度效果，使得服务时间上有所减少，整体来看比 HGWOA 算法提升了 10%的性能，与 RLWOA 算法相比大约提升了 16%。由此可见 RLWOA 算法能够找到更优的调度方案，在更短的时间完成服务功能链。

3.6 总结

本文研究了虚拟网络功能调度问题，并以最小化完成时间为优化目标建立了相应的数学模型。为了解决这一问题，结合了强化学习与鲸鱼优化算法，提出了一种新的调度方法。在所提出的基于强化学习的改进鲸鱼优化算法中，对鲸鱼算法进行了多方面的优化，包括种群初始化、编码与贪婪解码策略、交叉与变异操作等。此外，引入了强化学习技术用于自适应调节算法参数，从而显著提升了算法的整体调度能力。

通过实验验证，本算法在收敛速度和调度质量方面均表现出色。实验结果显示，相比于其它方法，改进的算法能够更有效地优化虚拟网络功能的调度，显著降低了网络服务的完成时间，提高了整体系统的效率和性能。

因此，本文提出的基于强化学习的鲸鱼优化算法不仅在理论上创新，而且在实践中具有显著的应用潜力，为解决复杂的调度问题提供了一种有效新途径。

4 基于深度强化学习的低延迟网络功能调度算法

4.1 引言

随着网络服务日益普及，用户对服务质量的期望也在不断提高。这使得网络运营商在研究服务功能链任务的放置与调度时面临着多样化的优化目标，这些目标根据用户需求的不同而异。从数据中心到5G网络基站，它们的主要优化目标是在确保网络稳定运行的同时，寻求降低能耗和运营成本的策略。而对于互联网触觉服务、微型手术机器人、智能驾驶和虚拟现实服务商等应用，则优先考虑实时性和低延迟作为核心网络性能指标。因此，优化工作首要关注如何减少数据传输延时，以满足对低时延数据传输的需求。

在第 3 章中，本文提出了一种基于强化学习与鲸鱼优化算法相结合的虚拟功能调度模型。该模型通过强化学习动态调节鲸鱼优化算法的参数，以最小化最大时间延迟为目标，解决了 VNF 放置与调度的联合问题。然而，对于启发式算法而言，处理问题时，尤其在处理复杂问题可能导致收敛速度缓慢，这会增加计算成本和运行时间，同时还会陷入局部最优解。此外，如果以最小化最大延迟为目标，未能充分考虑到实际网络服务的生命周期（Time To Live, TTL）和网络节点及链路带宽等资源约束条件，这可能影响网络服务的质量。

因此，本文以最小化网络服务迟到率为目标，首先将低延迟 VNF 调度问题表述为混合整数线性规划问题，然后将原问题重新定义为马尔科夫决策过程问题，并提出了基于深度强化学习框架的解决方案。使用深度强化学习框架来设置网络状态和参数、定义动作空间和奖励机制，以确保智能代理在不同环境下做出正确决策，是一个具有挑战性的任务。本文提出的深度强化学习调度算法致力于解决 VNF 的放置和调度问题，保证延迟敏感的网络服务能够在严格的服务截止时间内完成，从而最小化未完成服务功能链的数量。其主要贡献包括：

1.将 VNF 调度问题建模为混合整数线性规划问题，考虑了 VNF 的联合放置和调度，同时优化了路由选择，旨在提高服务功能链的接受率和降低路由开销。

2.设计了基于深度强化学习的状态、动作和奖励模型（D3QN），使智能代理能够根据当前环境选择最佳动作，以最大化奖励。此外，结合启发式算法用于路由优化，以在满足 SFC 要求的同时最小化传输时间和路由开销。

3.通过一系列实验验证了提出的深度强化学习算法的有效性，并将其性能与元启发式算法、复合调度规则以及标准 DQN 进行了比较。仿真结果表明，所提算法在解决低延迟感知的 VNF 调度问题上具有显著优势。

本章节的结构如下：第 4.2 节建立了问题的详细描述和数学模型，第 4.3 节介绍了基于 D3QN 的调度模型，第 4.4 节通过实验分析验证了算法的有效性，

最后在第 4.5 节进行总结和讨论。

4.2 问题描述与模型

服务功能链是基于客户需求在网络服务阶段开始时由不同的虚拟网络功能组成的网络功能链。这些 SFC 具有序列化和依赖的执行顺序（下一个 VNF 只能在前一个完成后开始处理）。例如，存在一个执行顺序为：VNF11→VNF12 →VNF13→VNF14 的 SFC，在其中 VNF12 只有在 VNF11 执行完成后才开始运行。

VNF 放置基于 SFC 组件的完成进行。其主要目的是找到满足 VNF 约束条件的节点位置，并为后续的调度阶段做准备。VNF 调度针对每个 SFC 的流量进行，以使更多的 SFC 能够在指定的截止时间内完成。因此，创建一个虚拟网络来放置 SFC 中的 VNF，并在物理服务器上部署的虚拟机上运行它们。本文考虑了，网络上沿服务链进行部署 SFC，指导它们之间流量的同时确保它们的执行顺序和所需带宽，以及最终截止时间。假设每个虚拟机上的每个 VNF 实例可以被多个 SFC 共享，但每个虚拟机一次只能处理一个 SFC 的流量。在本文的剩余部分，通过示例细化问题，并讨论对调度的影响。

假设一个 NFV 基础设施由四个支持 VNF 的虚拟节点 N 和五个链路 L 组成，链路的可用带宽为 15 Mbps，网络拓扑如图 4-1 所示：

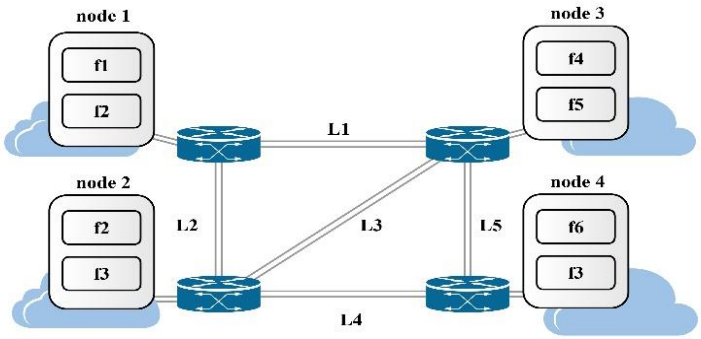

图 4-1 网络拓扑

给定一组由多个 VNF 组成的延迟敏感 SFC，虚拟网络功能由 $F=\{f_1,f_2,...,f_m\}$，同时用 VNF_i 代表 VNF 所对应的虚拟网络功能,其中 $1\le i\le m$,例如，VNF_i 代表 VNF 的功能为 f_1，并且每个 VNF_i 必须放置到拥有相同网络功能的节点 N 上。在网络中每个节点有着不同的网络功能。由于每一个 VNF 在网络中可能有着不同的处理能力，所以处理时间 $p_t=w/p_v$，w 为流量的大小，p_v 为该 VNF 的处理能力。除了节点的处理时间以外，流量在链路上的传输时间也可以表示为 D_t，

$D_t = w/b$，其中 b 表示所需的链路带宽。根据上述描述，在此实例中的 SFC 可由一个 5 元组表示 $SFC = \{VNF, w, b, p_t, D\}$，其中 VNF 表示 SFC 所需要的 VNF 的集合，p_t 为节点的处理时间，D 为 SFC 的截至时间。假设在实例中有 3 个 $SFC = \{SFC_1, SFC_2, SFC_3\}$，$SFC_1 = \{(VNF_1, VNF_4), 24Mb, , 12Mbps, 2T, 10T\}$, $SFC_2 = \{(VNF_2, , VNF_5), 12Mb, 12Mbps, 1T, 3T\}$, $SFC_3 = \{(VNF_2, , VNF_4), 24Mb, , 6Mbps, 2T, 8T\}$,如图 4-2 这些 SFC 均在 $T = 0$ 时到达网络，在第一种按顺序接受情况下如图 4-3 所示：

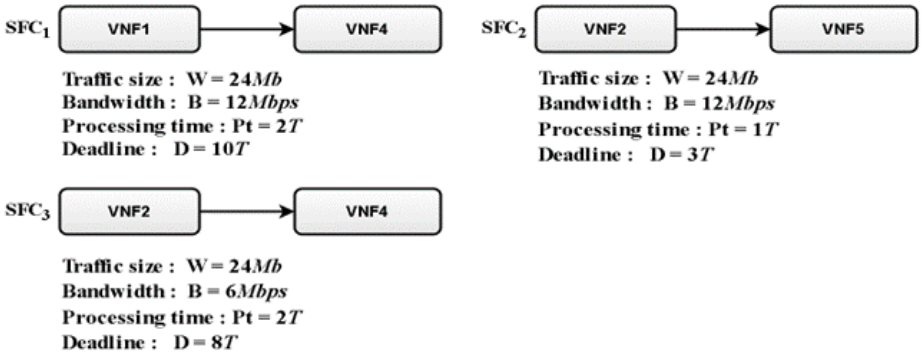

图 4-2 SFC 相关信息

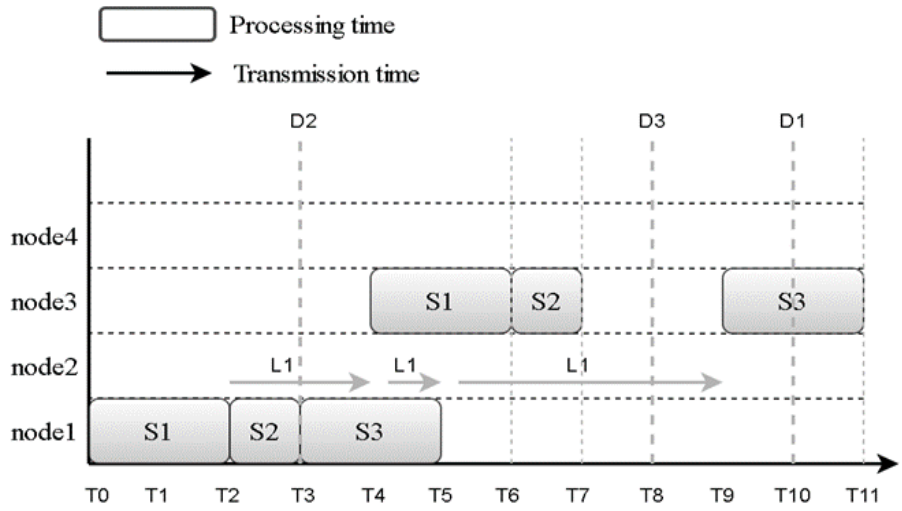

图 4-3 VNF 按顺序处理

当同时放置到节点 1 时，按照 SFC_1,SFC_2,SFC_3 的顺序分别处理 3 个 *SFC*，当在 *T=0* 到 *T=2* 时，节点 1 完成了对 SFC_1 的第一个 VNF 的处理，流量开始通过虚拟链路 L1 传输，传输时长为 2s，即 *T=4* 时，节点 3 开始处理下个 *VNF*，最终在 *T=6* 时完成 SFC_1，完成时间小于 SFC_1 的截至时间 D，满足传输延迟要求。随后，看 SFC_2 当 *T=2* 时节点 1 处理完 SFC_1 后开始处理 SFC_2 的第一个 *VNF*，处理时间为 1s，因此在 *T=3* 时完成，后续也向虚拟链路 L1 开始流量传输。但是，由于此时虚拟链路 L1 还在传输 SFC_1 的流量，虚拟链路的剩余带宽

$(15Mbps-12Mbp<12Mbps)$ 不足以满足 SFC_2 的带宽需求。因此，SFC_2 只能进行等待，直到 $T=4$ 时，SFC_1 传输完成后 SFC_2 开始进行传输，并在 $T=5$ 的时间点到达节点 3，由于此时 SFC_1 的处理还未完成，只能再次进行等待并在 $T=7$ 时完成，但是此时已远超过 SFC_2 的截止时间，不被接受。最后 SFC_3 在 T=3 处开始处理并于 T=5 处完成，虚拟链路 L1 的剩余带宽为 $25Mbps$ 满足 SFC_3 所需带宽进行传输，至 T=9 时到达节点 3 并最后在 T=11 处完成处理，与 SFC_2 相同，最终处理完成时间大于 SFC_3 的截止时间 D 也不能被接受。

很明显在这种情况下在同时进入网络的 3 个 SFC 中有两条都无法满足其延迟需求而被拒绝。在文献[68]中也描述了对调度影响的子问题，但约束条件和解决方法与本章节不同。受此启发，本章节通过描述几种情况对网络进行调度，使所有 SFC 都尽量在截止时间前完成，而这些情况也是需要联合解决的问题。

4.2.1 VNF 的放置

在上小节中 SFC_2 和 SFC_3 的完成时间都远大于了其截止时间，主要原因是因为 SFC 的处理顺序和虚拟链路带宽的不足导致了过多的等待，延迟了 SFC 的处理时间。为了减少在节点和链路上的等待时间，这里把 SFC_3 的第一个 VNF 放置到节点 2 中，在 T=0 的时刻 SFC_1 和 SFC_3 同时进行处理如图 4-4 所示，当 T=2 时 SFC_1 和 SFC_3 均完成了处理，SFC_1 依旧在 L1 的虚拟链路上传播，而 SFC_3 则在 L3 虚拟链路上传播，当 T=7 时 SFC_1 到达了节点 2 并且开始处理，于 T=6 完成，与上小节相同 SFC_2 也在 T=7 处完成，而 SFC_3 在 T=7 处开始在 T=9 时完成处理虽然此时 SFC_3 依旧没有达到预期的处理时间但相比于上一个情况完成时间提前。

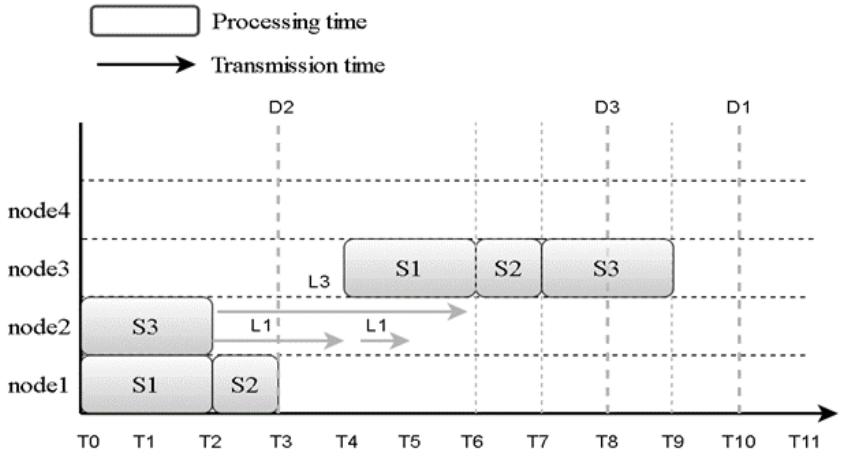

图 4-4 VNF 的放置

4.2.2 VNF 的调度

上述中 SFC_2 依旧未能在截止时间内完成，主要是因为每次 SFC_1 均为第一个处理，这使得对时间延迟非常敏感的 SFC_2 均需等待两个时间单位。因此，在本小节中考虑进行调度，让 SFC_2 先进行流量的处理如图 4-5 所示，在 T=0 时刻 SFC_2 和 SFC_3 分别在节点 1 和节点 2 上开始处理，在 T=1 时，SFC_2 处理完成，通过在虚拟链路 L1 上传输并在 T=2 处达到节点 3 在 T=3 时完成了流量处理，此时满足了延迟需要可以在网络中并被接纳，而 SFC_1 和 SFC_3 也分别在 T=7 处和 T=9 时完成。

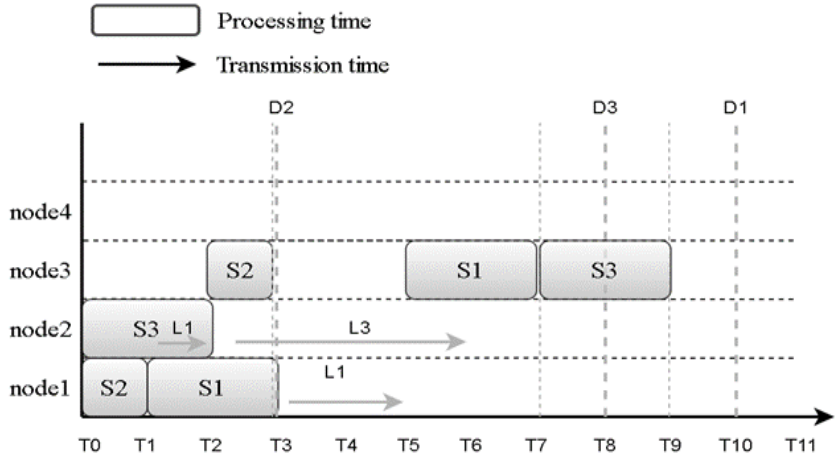

图 4-5 VNF 的调度

4.2.3 流量的路由

在上述小节中，虽然 SFC_1 和 SFC_2 均满足了延迟需求，但是 SFC_3 还是有 1 个时间单位超出截止时间，主要是因为 SFC_1 优先到达节点 3，并且此时节点 3 处理空闲状态，因此会在 SFC_3 到达之前进行处理，让 SFC_3 进行了等待，为了解决这个问题。本文进行路由的选择如图 4-6 所示，前面与和上小节相同在 T=0 时 SFC_2 和 SFC_3 分别在节点 1 和节点 2 上开始处理，在 SFC_2 处理完成后轮到 SFC_1，并在 T=3 时完成第一个 VNF，随后 SFC_1 流量开始向虚拟链路 L2 和 L3 传输而不是 L1，更换了传输路由，这使得在 T=6 时 SFC_3 率先到达节点 3 中并开始，直到 T=8 时完成此时 SFC_3 满足了延迟需求，而 SFC_1 等待 SFC_3 处理完成后开始并在 T=10 处完成，此时延迟需求也得到满足。这样 SFC_1, SFC_2, SFC_3 都在规定时间内完成，均可以被网络所接受。

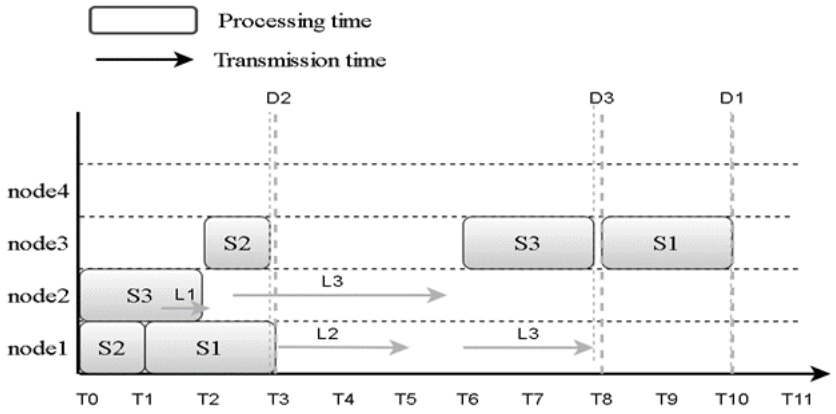

图 4-6 流量路由

4.2.4 问题定义

在物理网络图 G(N,L)中，其中 N 表示为虚拟节点用来托管和运行不同类型的 VNF，L 表示为连接每两个节点之间的虚拟链路。现有一组 SFC，并且每个 SFC 都请求其流量通过网络进行处理，其需要满足以下需求：a)每个 SFC 的 VNF 都必须放置到能处理其功能的节点上 b）每个 SFC 的 VNF 都是按顺序来处理，在一条 SFC 中上一个 VNF 未处理完成时下一个 VNF 是不能接着处理 c）链路的带宽和截止时间必须满足流量的需求。本文的目的就是在满足上述需求的情况下找到最佳的节点对 VNF 进行放置和调度，以便使网络中 SFC 的接收率达到最大化。

本文涉及的主要参数及意义如表 4-1 所示。

表 4-1 参数及意义

参数	意义
$G(K,E)$	表示 SFC 的转发图
F	表示 VNF 类型集合
f_t	表示 VNF 实例化类型 $f_t \in F$
$C_{i,j}$	表示虚拟链路 i, j 之间的可用带宽
S	表示 SFC 集合
N	表示节点集合
δ	表示时间戳
$b_{k,k+1}$	表示 SFC 中第 K 个 VNF 与第 K+1 个 VNF 之间所需带宽
w_s	表示 SFC 的流量大小 $s \in S$
D_s	表示 SFC 的截止时间 $s \in S$
f_s^k	表示 SFC 上第 K 个 VNF 的类型 $s \in S$, $f_s^k \in f_t$
f_n	表示节点 N 上可以实例化的类型 $s \in S$, $f_n \in f_t$

V_S	表示组成 SFC 所需 VNF 的数量 $s \in S$
$p_{n,s}^{k}$	表示在节点上 SFC 中第 K 个 VNF 的平均处理时间 $k \in k \ s \in S \ n \in N$
p_s^e	表示在虚拟链路 e 上的传输时间
pd_s	表示 SFC 中已经完成处理 VNF 的数量
end_s	表示 SFC 中当前最后一个 VNF 处理后的时间 $s \in S$
u_n^t	表示在节点 N 上处理完最后一个 VNF 的时间点 $n \in N$
$u_{n,k}^t$	表示在节点 N 上所有已经完成处理的 VNF 的处理时间总和 $k \in k \ n \in N$

为联合 VNF 放置和调度问题定义了以下决策变量。

$K_{k\,s}^{n\,\delta} \in \{0,1\}$ 是一个二进制变量,指的是 SFC_s ($s \in S$)上的 VNF($k \in k$)在时间戳 δ ($\delta \in \delta$)时是否放置在节点 n($n \in N$)上开始处理，是则为 1 否则为 0。

$l_{i\,j}^{s\,e} \in \{0,1\}$ 是一个二进制变量，指的是 SFC_s ($s \in S$)是否将虚拟链路 e($e \in E$)放置在链路 $(i, j) \in L$ 上，是则为 1，否则为 0。

$j_s \in \{0,1\}$是一个二进制变量,指的是 $SFC_{s \in S}$ 是否被物理网络所接受, 即是否在规定时间内完成，是则为 1，否则为 0。

接下来，将考虑优化目标及其约束所需的约束条件。本章目标是在确保约束的同时调度 SFC，因此目标是最大化 SFC 接受率，这等同于最小化 SFC 拒绝率：

$$Maximize \Sigma_{s \in S} j_s \tag{4-1}$$

其约束条件为：

$$st \begin{cases} C1: K_{k\,s}^{n\,\delta} f_s^k = f_n \\ C2: \sum_{n \in N} K_{k+1\,s}^{n\,\delta} \le 1 - \sum_{n' \in N} K_{k\,s}^{n'\,\delta} \\ C3: K_{k\,s}^{n\,\delta} = 1 - K_{k'\,s'}^{n\,\delta}, k, k' \in K, s, s' \in S \\ C4: \sum_{k \in K} p_{n\,s}^k + \sum_{e \in E} p_s^e \le D_s, s \in S \\ C5: l_{i\,j}^{s\,e} b_{k,k+1} \le C_{i\,j}, e \in E, (i, j) \in L \\ C6: \sum_{k \in K} K_{k\,s}^{n\,\delta} = 1 \end{cases}$$

本章的优化目标主要受到 C1~C6 约束。C1 是为了确保 SFC 上的 VNF 类型与放置到节点的 VNF 实例的类型相同。C2 规定，在 SFC 的 VNF 仍在处理时，不能处理 VNF。C3 确保一个节点的 VNF 实例在处理当前 SFC 的 VNF 时，不能处理另一个 SFC 的 VNF。C4 确保 SFC 的剩余最短完成时间可以满足截止时间。C5 要求 SFC 所需的虚拟链路容量必须小于物理链路的容量。C6 规定 SFC 的 VNF 一次只能放置到一个节点上。

4.3 基于 D3QN 的调度模型

在第二章中已经介绍过 Double DQN 和 Dueling Double DQN。为了得到更好的模型，本章节提出使用 D3QN 作为训练的网络模型，期望能够取得更好的结果。在本节中，首先介绍 D3QN 的状态定义，随后介绍每个调度点的候选调度规则（动作）和奖励定义。接着，解释节点和路由选择过程。最后，讨论 D3QN 的网络结构和训练方法。

4.3.1 状态设置

通常情况下，SFC 或网络节点的数量通常非常大。如果在每一刻都使用每个 SFC 或网络的状态特征作为指标，可能会导致大量的输入量，这可能会给 D3QN 的适应和训练带来困难。因此，本文提出通过取每个 SFC 的特征，并计算它们的平均值以便于 D3QN 的训练，并使其更容易扩展到其他环境。因此，状态定义如下：

(1) *ARC* 表示每个 SFC 内成功放置的 VNF 相对于 VNF 总数的概率：

$$ACR = \frac{\sum_{1}^{s} \frac{pd_S}{V_S}}{s} \tag{4-2}$$

(2) *AUR* 代表平均节点上处理 VNF 所花费的总时间与节点运行的总时间的比较：

$$AUR = \frac{\sum_{1}^{n} \frac{\sum_{1}^{k} (u_{n,k}^{t})}{u_n^t}}{n} \tag{4-3}$$

(3) *EOR* 表示当前调度时间点的潜在 SFC 拒绝率，即如果 SFC 未完成部分（即剩余未处理的 VNF）的估计最短剩余处理时间超过了指定的截止时间，即使此时 SFC 还没有超过截止时间，也被视为潜在的拒绝。计算过程如算法 4-1 所示：

算法 4-1 EOR 的计算过程

输入: V_S , pd_S , D_s， end_S
输出: *EOR*
1: $N = 0$
2: **for** k = 1: n **do**
3: **if** $pd_S < V_S$ **then**
4: T= 0
5: **for** i = pd_S +1: V_S **do**

6: T += ($p_{ns}^{k} + p_{s}^{e}$)
7: **end for**
8: **if** end_S +T > D_s **then**
9: N +=V_S − i
10: Break
11: **end if**
12: **end if**
13: **end for**
14: $$EOR = \frac{N}{\sum_{1}^{S} V_S}$$
15: **return** *EOR*

在算法 1 中，第三行表示寻找尚未完成放置的 SFC，而第五到第十行代表估计未完成 SFC 中剩余 VNF 的处理时间，并确定 SFC 是否可以在预期的处理时间内完成处理。

(4) *AOR* 指的是超过处理截止时间的 SFC 与 SFC 总数的比例。计算过程如算法 4-2 所示：

算法 4-2 AOR 的计算过程

输入: V_S , pd_S , D_s , end_S
输出: *AOR*
1: N = 0
2: **for** k = 1: n **do**
3: **if** end_S > D_s **then**
4: N +=1
5: **end if**
6: **end for**
7: $$AOR = \frac{N}{\sum_{1}^{S} V_S}$$
8: **return** *AOR*

在算法 4-2 中，第 3 到 7 行代表循环中的所有 SFC，以确定当前处理时间是否已经超过了截止时间。如果超过了，N 值增加 1，并除以 SFC 的总数。

为了提高代理在处理不同网络环境时的性能和泛化能力，本文采取了一种将状态值归一化到[0, 1]区间并计算其平均值的方法。这一策略旨在提升代理的网络适应性。通过设定 SFC 的平均完成率(ACR)和节点的平均利用率(AUR)，本研究能够更精细地捕获网络状态的动态变化，从而丰富奖励机制，避免奖励稀疏问题。ACR 和 AUR 不仅能有效地反映 SFC 的当前完成情况，而且还能帮助

代理做出更加准确的决策。实际完成率(AOR)和预期完成率(EOR)分别反映了网络中现有未完成 SFC 的数量和预计将保持未完成的 SFC 数量，为代理提供了直接的网络状态洞察。然而，由于这些指标变化有限，可能会导致奖励稀疏，因此通过结合 ACR 和 AUR 可以为代理提供一个更全面的网络状态理解，以优化其决策过程。

4.3.2 状态设置

在现有研究中，有些使用单一规则进行调度，但是单一规则调度算法可能无法有效应对所有网络环境，本文引入了复合规则调度策略，设计了 5 种专为智能代理定制的规则，以适应不同网络状态。这些规则的设计旨在解决网络状态多样性带来的挑战，通过在每个调度周期中基于网络当前状态选择奖励值最高的规则，确保选出最合适的 SFC 进行处理。此策略不仅提高了调度算法的灵活性和效率，也为路由优化和网络性能提升奠定了基础。

规则 1：首先按顺序比较每个 SFC 的截止时间，并选择具有最小截止时间的 SFC 作为最高优先级。如果存在相同的截止时间，则选择具有最大 end_s 值的 SFC 作为最高优先级，因为如果一个 SFC 的当前 end_s 值较大，意味着用于处理后续 VNF 的可用时间较少。这种方法旨在优先处理最紧迫的任务，从而确保延迟敏感的服务能够及时完成，提高整体网络效率和服务质量。

规则2：本文将使用基于最低松弛度的调度作为高优先级，因为松弛度反映了任务的紧迫性，松弛度越低，任务的可用时间越少，紧迫性越高。这种方法通过识别并优先处理那些具有更高完成紧迫性的任务，旨在提高网络服务的及时性，确保重要任务能够在截止时间前得到有效处理。

规则3：本文根据是否存在end_s值超过截止时间的SFC将其分为两种情况。如果存在这样的 SFC，那么具有最小松弛时间的 SFC 将被赋予最高优先级。如果没有，每个 SFC 的剩余时间将被其剩余操作数除，然后选择最小值赋予最高优先级。具体过程如算法 4-3 所示：

算法 4-3 规则 3 运算过程

输入: V_S , pd_S , D_s , end_S

输出: SFC

1: O←{S| pd_S <V_S && D_s <end_S }

2: P← {S| pd_S <V_S }

3: **if** Isempty(O) **then**

4: $SFC = \arg\min_{S \in P} \frac{D_s - end_S}{Vs - pd_S}$

5: **else**

6: $SFC = \arg\min_{S \in O}[D_s - end_S - \sum_{v=pd_S}^{v_S}(p_{ns}^k + p_s^e)]$

7: **end if**

规则 4：规则 3 类似，此情况也处理两种场景。如果 O 为空，剩余时间将被 SFC 的估计平均处理时间除。否则，使用最小松弛时间作为选择标准。规则 4 具体如算法 4-4 所示：

算法 4-4 规则 4 运算过程

输入: V_S , pd_S , D_s， end_S

输出: SFC

1: O← {S| pd_S <V_S && D_s <end_S }

2: P← {S| pd_S <V_S }

3: **if** Isempty(O) **then**

4: $SFC = \arg\min_{s \in p} \frac{D_s - end_s}{\sum_{v=pds}^{v_S}(p_{ns}^k + p_s^e)}$

5: **else**

6: $SFC = \arg\min_{S \in O}[D_s - end_S - \sum_{v=pds}^{v_S}(p_{ns}^k + p_s^e)]$

7: **end if**

规则 5：预期最短处理时间指的是计算处理每个未完成 SFC 所需要的时间。这涉及到评估 SFC 中每个未处理 VNF 的处理时间，并累加以得到整个 SFC 的完成时间预估。这种计算帮助确定哪些 SFC 应被优先处理，尤其在资源有限且需满足紧迫截止时间的情况下。

以上是关于动作的设置，通过 5 个不同的规则，让智能体能根据当前环境做出最优决策，选择出最适合当前的规则以此为后续节点选择与路由优化做铺垫。下小节将对奖励设置进行分析。

4.3.3 奖励设置

由于本文的目标是最小化拒绝率，因此设计了基于此目标值的奖励函数 R_t。每个 SFC 都必须尽可能在其截止时间内完成，以减少拒绝率并获得更多奖励。然而，由于拒绝率的变化只有在整个 SFC 超过其截止时间或已被处理时才能知道，因此奖励可能变得稀疏。为此，本文通过考虑当前状态的四个关键状态特

征 AOR、EOR、ACR、AUR 以及下一个状态 AOR' 、 EOR' 、 ACR' 、 AUR' 的值来定义回报 R_t。具体如下所示：

$$R_t=\begin{cases}1 & \text{if } AOR'<AOR\\ -1 & \text{if } AOR'<AOR\\ f(EOR,EOR',ACR,ACR') & \text{if } AOR'=AOR\end{cases} \tag{4-4}$$

当 $AOR'=AOR$ 需要定义函数 $f(EOR,EOR',ACR,ACR')$ 去决定奖励rt的值。

$$rt=\begin{cases}1 & \text{if } EOR'<EOR\\ -1 & \text{if } EOR'<EOR\\ \mathrm{g}(ACR,ACR') & \text{if } EOR'=EOR\end{cases} \tag{4-5}$$

当 $EOR'=EOR$ 时rt为：

$$rt=\begin{cases}0 & \text{if ACR}'>\text{ACR}\\ 1 & \text{if ACR}'>1.1\times\text{ACR}\\ -1 & \text{otherwise}\end{cases} \tag{4-6}$$

对于 AUR 状态来说,其奖励 rt' 如下设置：

$$rt'=\mathrm{e}^{\text{AUR}'-\text{AUR}} \tag{4-7}$$

最终奖励R_t被设置为如下所示：

$$R_t=rt+rt' \tag{4-8}$$

以上即为关于奖励的设置，后续小节中将通过启发式算法与深度强化学习相结合来解决节点选择与路由优化问题。

4.4 节点选择与路由优化

根据上述描述，D3QN 动作可以用来选择最高优先级的 SFC 进行放置。然而，为了满足延迟要求，选择 SFC 的最佳节点和路由也是一个关键问题。因此，本节将描述如何进行节点和路由选择。

4.4.1 节点的选择

节点选择的目的是选择一个能够处理VNF类型实例并尽早开始处理其流量的节点。这是因为只有尽早处理流量，才能缩短完成时间并满足延迟要求。这一点可以通过以下公式描述：

$$\min_N \max(N_T, A_t) \tag{4-9}$$

N_T代表节点N上最后一个VNF处理完成的时间，A_i代表选定的SFC的流量到达节点N的时间。$\max(N_T, A_i)$用来表示流量到达该节点的最早开始时间。因此，$\min_N \max(N_T, A_i)$表示在所有可用节点中选择最早可用的节点。这一选择过程旨在确保SFC尽可能早地被处理，以缩短完成时间并满足延迟要求。

4.4.2 路由优化

上述公式代表了节点选择，但目前无法确定 A_i，即流量从前一个节点通过链路到达下一个候选节点时的到达时间。因此，使用 Dijkstra 算法在确保物理链路容量满足虚拟链路需求时，找到两个节点之间最短时间路径，以确定 A_i。由于目的是找到两点之间的最短时间路径，路径权重被设置为链路传输时间加上等待时间，权重设置如图 4-7 所示。

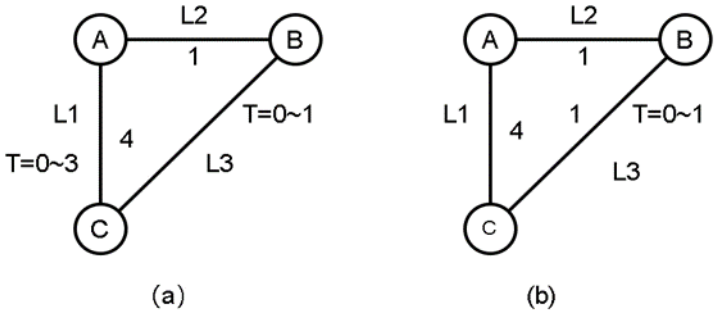

图 4-7 Dijkstra 的权重设置

在图 4-7(a)所示的流量是从源节点 A 到目的节点 C 的流量，链路上的流量传输时间为一个时间节点。由于 *T=0* 到 *T=3* 期间资源不足，链路 L1 需要等待 3 t 节点，因此 L1 的权重为 4，而 L2 的权重为 1，因为它有足够的资源。节点 A、B 和 C 的值分别为 0、1 和 4。当流量从走 A 向 B 同时以 B 为起点，如图 4-7(b)所示，因为 L3 的处理时间从 *T=0* 到 *T=1*，且节点 B 的值为 1，L3 不需要等待，所以 L3 的权重为 1。因此，节点 A、B 和 C 的值更新为 0、1 和 2。因此，从 A 到 C 的最短时间路径为 2。具体的节点和路由选择算法如下算法 4-5 所示：

算法 4-5 节点选择和路由选择

输入： N_T ， SFC
输出： selected node
1:M = []
2: **for** m= 1: *N* **do**
3: **if** 该节点满足约束 **then**
4: L = [], $A_i = end_s$
5: L← Dijkstra(SFC)
6: **for** s = L **do**

7: A_i+= p_s^e
8: **end for**
9: **end if**
10: M← max(N_T, A_i)
11: **end for**
12: selected node = $Arg\min M$

在算法 4-5 中，第 2 到 5 行通过循环所有节点来寻找满足约束条件的所有节点。第 5 行表示，使用Dijkstra算法从SFC的前一个节点找到候选节点的最短时间路径，并保留经过的节点。这一过程包括计算从起始节点到每个候选节点的最短时间，同时确保路径上的物理链路容量能够满足SFC的需求。通过这种方式，可以为SFC选择最优的节点和路由，以满足延迟要求，提高网络效率。从第 6 行到第 10 行，计算从前一个节点到所有可用节点的时间作为A_i。最后，将A_i与N_T比较，找到一个较大的值并赋值给M，其中M表示节点的最早开始时间，第 12 行表示找到最早开始节点的当前VNF的下一个处理节点。

4.4.3 D3QN 训练过程

算法 4-6 D3QN 的训练过程

输入： S_t
输出： S_{t+1}，目标网络参数更新，R_t
1: 初始化经验池 D
2: 初始化在线网络参数 θ
3: 初始化目标网络参数 $\theta' = \theta$
4: **for** 最大迭代次数 **do**
5: 初始化状态集合{ ACR， AUR， EOR， AOR }= {0，0，0，0，0}将训练参数输入进网络。
6: **for** I = 1: $\sum_{i=1}^{n} V_S$ **do**
7: 通过网络模型选择 Q 值最高的动作 A。
8: 执行动作 A，选择最高优先级的 SFC，将其返回给环境进行调度，并生成 S_{t+1}。
9: 通过公式 4-4 至 4-8 计算奖励 R_t
10: 将四元组 $< S, A, R, S_{t+1} >$ 存储在经验池 D 中。
11: 从经验池 D 中随机抽取一个批次。
12: 使用梯度下降更新在线网络参数。
13: 更新目标网络参数。 $\theta' = \theta$
14: **end for**
15: **end for**

在算法 4-6 中，第 7 行，这种方法使得智能体能够在给定的状态下，基于预测的未来奖励最大化原则，选择最佳动作。这是强化学习算法中常用的决策制定机制，旨在优化长期性能，通过不断学习和调整，使智能体能够在复杂环境

中做出更精确的决策。

第 8 行，这个过程是智能体与环境交互的核心部分，通过执行选择的动作，智能体能够观察到环境的变化（即状态的转移），并根据这些变化来优化其决策策略，从而在未来的决策中能够更有效地达成目标。

第 10 行，这一步骤是深度强化学习中经验回放的一部分，旨在通过存储智能体的经验（状态转移、动作、奖励等）来提高学习效率和稳定性。智能体可以从这个经验池中随机抽取数据进行学习，从而避免了连续经验之间的强相关性，帮助智能体能更好地泛化和适应环境。

第 11 行，这是深度强化学习训练过程中的一个关键步骤，旨在通过从以往的经验中随机抽取数据样本来进行学习，这种方法有助于打破数据间的时间相关性，提高学习的稳定性和效率。通过这种方式，智能体能够从多样化的经验中学习，从而更好地泛化其决策策略到不同的状态和情境中。

第 12 行中通过计算损失函数相对于网络参数的梯度，并利用这些梯度来更新网络参数，从而使损失函数最小化。这种方法有助于智能体通过逐渐学习得到更有效的策略，以在给定的任务中达到更好的性能。梯度下降是优化深度神经网络参数的常用技术，关键在于选择合适的学习率和批次大小来确保训练的稳定性和效率。

最后第 13 行中通过定期地将在线网络（即正在学习更新的网络）的参数复制到目标网络（用于预测下一状态值的网络），可以减少学习过程中的波动性。这种方法有助于平滑学习过程，提高算法的稳定性和性能。

4.5 实验和性能分析

4.5.1 参数设置

为了进行评估，设计了一个与文献[69]中类似的网络模型。一个是由 15 个完全连接的节点组成的中等规模网络，并在网络中引入了 15、20、25、30 和 35 个随机生成的 SFC。另一个是由 30 个完全连接的节点组成的大型网络，并在网络中引入了 30、35、40、45 和 50 个随机生成的 SFC。每个链路的可用带宽固定在 50*Mbps*。在本文中，假设每个 VNF 至少可以在一个 VM 节点上处理，每个 VM 节点有能力托管 2-3 个 VNF。对于每个 SFC，使用随机流量生成([25-75]*Mbits*)、带宽需求([15-25] *Mbps)*和变量 VNF([2-4])作为服务组合。它们的截止时间被设置为其处理和传输延迟之和的 4/3 倍，没有考虑任何等待延迟[70]。本实验在 windows10 系统，i5 处理器上，使用 Python 编写仿真程序运行。

4.5.2 D3QN 迭代

测试 D3QN 在包含 15 个节点和 35 次 SFC 插入的预定义实例上。D3QN 在前 300 个训练步骤中获得的总奖励结果显示在图 4-8 和图 4-9 中。

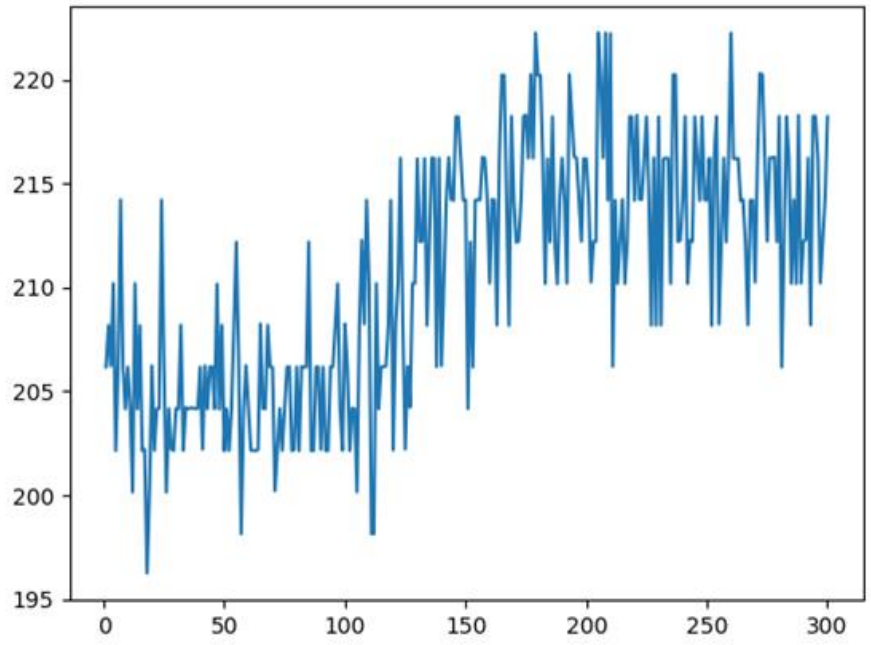

图 4-8 D3QN 迭代图

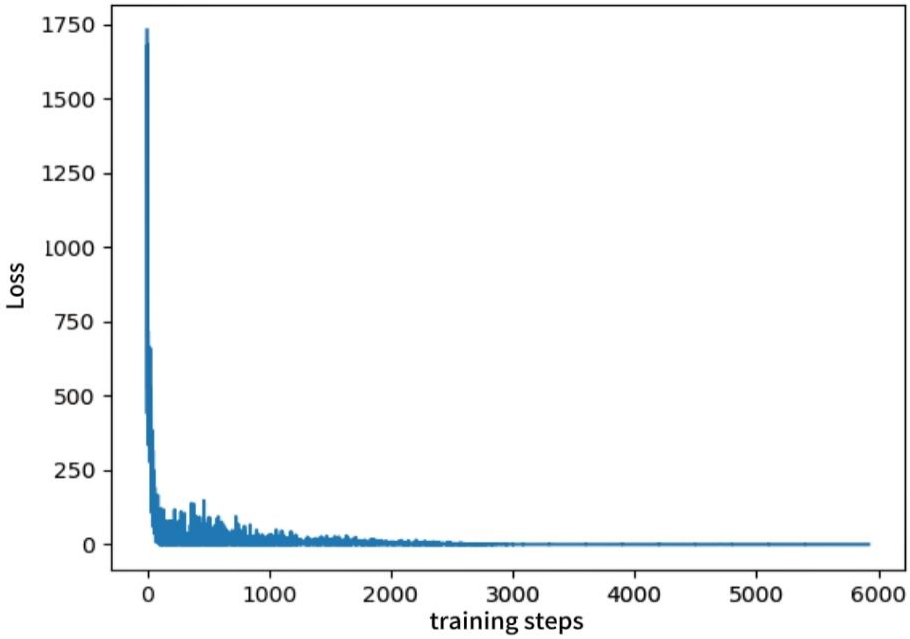

图 4-9 D3QN 损失图

如图 4-8 和图 4-9 所示，图 4-8 展示了训练频率与奖励值的变化。垂直轴代表奖励值，水平轴代表训练频率。在 0-100 步内，新的训练奖励值出现波动。神经网络随机选择动作，随着训练的进行，奖励值增加并最终趋于最大值。波动的原因是在训练的后期仍然存在随机选择动作的小概率。图 4-9 显示了训练过程中损失值的变化。可以看到，整体损失值呈下降趋势，中间有小幅波动。最终损失值趋近于 0，表明模型的预测结果与训练数据中的真实结果非常接近。

4.5.3 与符合规则的比较

上小节中通过 D3QN 的迭代图和 D3QN 对 D3QN 的损失图进行了分析和判断，对 D3QN 收敛效果进行了验证，可以看出 D3QN 在所实验的虚拟环境中是存在一定的稳定性。在该小节中，为验证 D3QN 的有效性，将其与每个复合规则进行了比较。为了确保 D3QN 的普遍性，在 15 节点和 30 节点网络上都进行了比较，即小型网络和中型网络。此外，为消除随机性，本文对每次比较训练了 50 次，并取平均值。这种更有效的比较可以避免随机性，平均值更具有说服力，SFC 拒绝率如图 4-10 和图 4-11 所示：

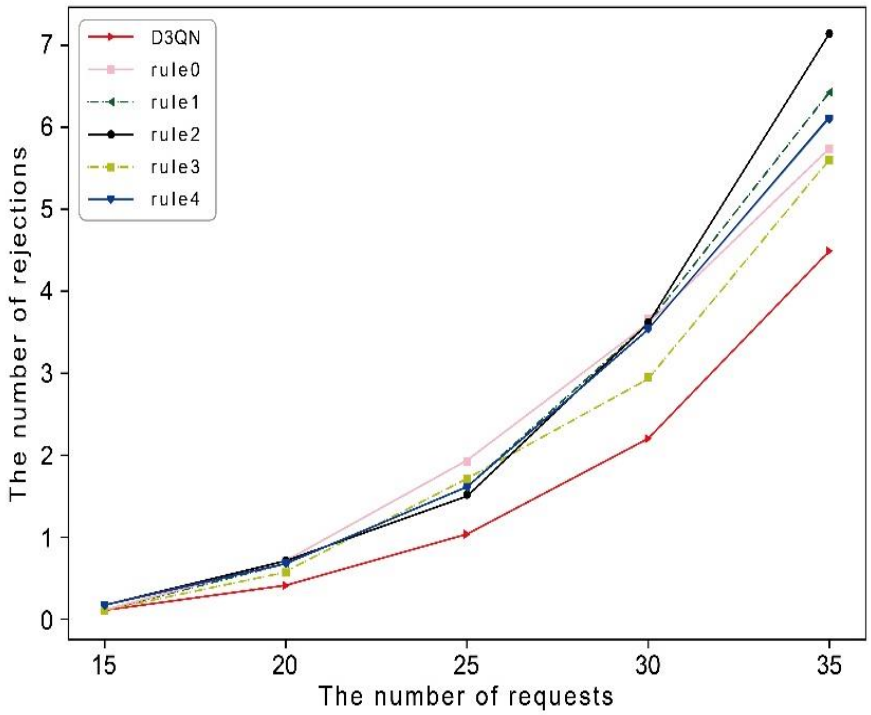

图 4-10 在节点数为 15 时 SFC 的拒绝率

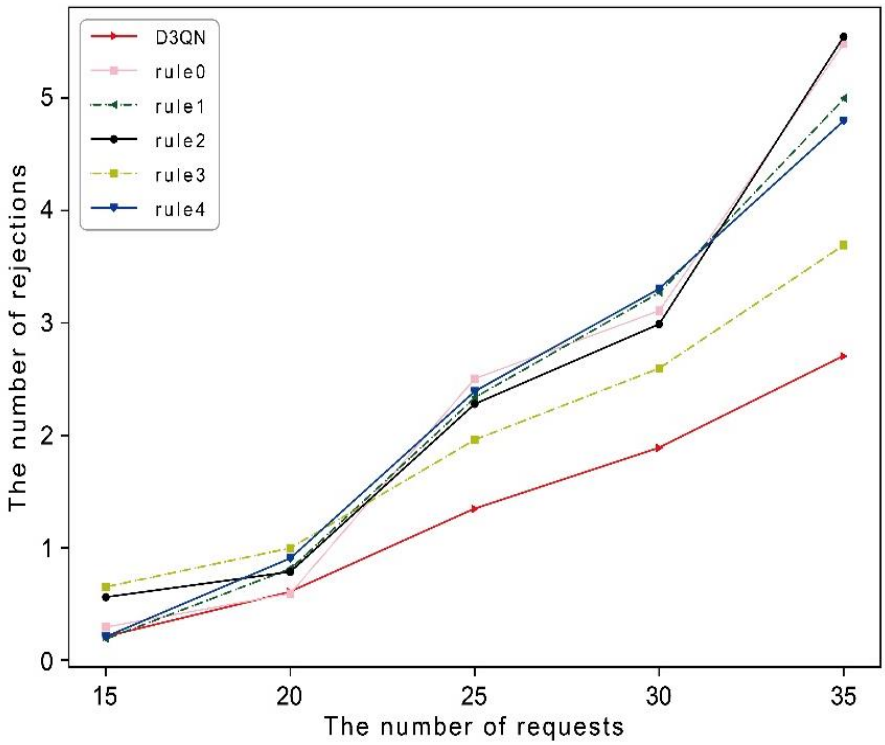

图 4-11 在节点数为 30 时 SFC 的拒绝率

基于上述实验结果，可以看出，与每个单独的复合调度规则相比，D3QN 保持了最低的拒绝率。这突出了训练有素的智能代理选择当前最优规则进行调度在不同网络环境中的能力，从而降低了拒绝率。当然，每个规则之间在性能上存在一定的差异，每个规则的合理设计和 D3QN 操作中适当的选择过程是影响算法性能的主要原因。总的来说，D3QN 能够在每个调度时间点选择更有利于当前情况的复合规则，使其比单一规则更有效。

4.5.4 与其他算法比较

除了与单一复合规则比较外，本文还与随机选择算法（即每个调度点以相同概率随机选择一个复合调度规则）和遗传算法进行了比较。此外，为了比较 D3QN 处理离散空间的优越性，D3QN 与传统的 DQN 进行了比较，后者在每个状态使用相同的可用动作集，并将结果与节点数为 15 和 30 进行了比较，SFC 拒绝率如图 4-12 和图 4-13 所示。

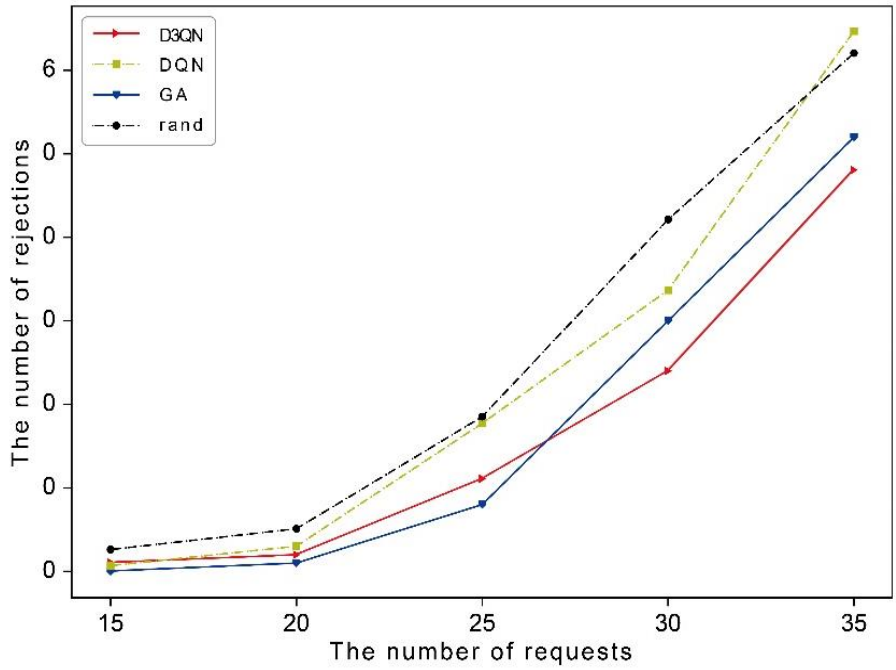

图 4-12 在节点数为 15 时 SFC 的拒绝率

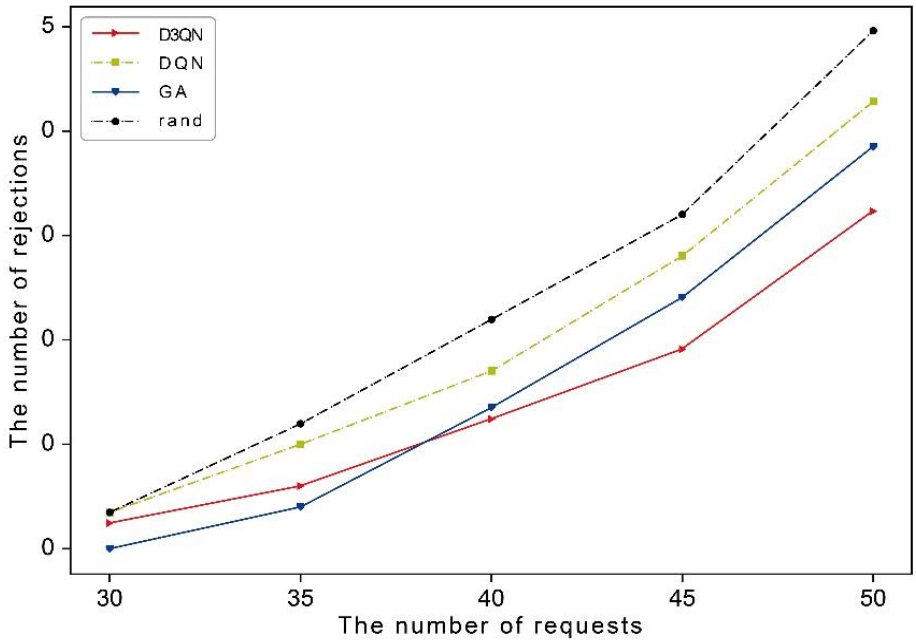

图 4-13 在节点数为 30 时 SFC 的拒绝率

根据上述图表，与随机动作选择策略和遗传算法相比，D3QN 在所有实例中几乎总是能够获得更低的总拒绝数，这意味着 D3QN 能够在每个重新调度阶段为复合规则选择选择适当的动作，以实现更好的调度结果。此外，在实例中，D3QN 的代理性能优于 DQN，这可能是因为 DQN 无法在网络环境中准确区分

不同状态，从而不可避免地降低整体性能。此外，D3QN 的双分支结构可以更好地反映每个不同动作的优势并选择最优动作。总之，D3QN 代理在处理离散状态空间方面比 DQN 代理更合理、更有效。

4.5.5 节点选择性能分析

为了验证节点选择的有效性，将与随机选择可用节点的平均单个 VNF 路由成本和延迟率进行比较。平均单个 VNF 路由成本是从一个 VNF 到下一个 VNF 的路由所需的资源，*ASV* 计算如公式 4-10 所示：

$$ASV = \frac{\sum_{1}^{S}\left(\frac{\sum_{1}^{k} B_{k,k+1}^{s} \times L_{k,k+1}}{k} \times j_s\right)}{S} \tag{4-10}$$

其中 $B_{k,k+1}^{s}$ 代表从 SFC $s \in S$ 的 VNF_k $k \in k$ 到 VNF_{k+1} $k+1 \in k$ 的带宽需求，$L_{k,k+1}$ 代表这两个 VNF 之间经历的路由路径。接下来，本文将在节点数为 30 时，比较有无节点选择时节点选择的性能，如图 4-14 和图 4-15 所示。

在图 4-14 和图 4-15 中， YD3QN 代表节点选择功能，ND3QN 代表无节点选择功能。可以看出，SFC 接受率和单个 VNF 路由成本都有良好的表现，这表明当有良好的节点选择时，VNF 调度可以在更短的时间内完成，以提高 VNF 接受率。同时，从图 4-15 中可以看出，良好的节点选择可以有效减少 VNF 路由的成本并增加运营商的收益。

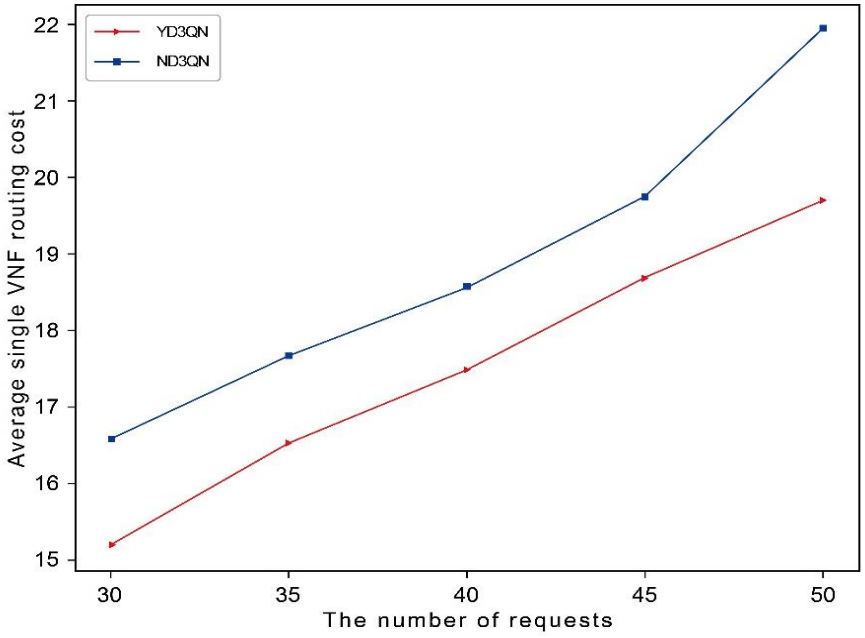

图 4-14 在节点数为 30 时 SFC 的拒绝率

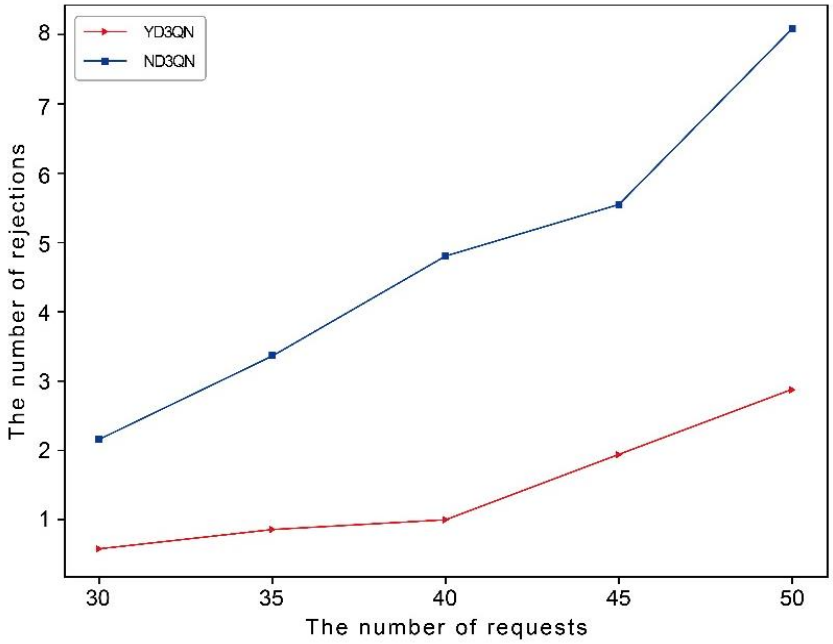

图 4-15 平均单一 VNF 的路由消耗

4.6 总结

本文提出的方法是基于规则的 D3QN 调度来解决网络中的调度问题。指定了 5 个规则，通过规则选择未处理的 VNF，并在每个调度时间点通过节点和路由选择将其分配给节点。此外，D3QN 用于训练，以在每个调度节点选择更合适的规则。本文还比较了两种网络环境，以验证 D3QN 的有效性和通用性。结果表明，经过训练后，D3QN 比其他复合规则、随机选择策略和遗传算法具有更好的性能。此外，与传统的 DQN 相比，D3QN 具有显著的优势，这进一步证明了 D3QN 在处理离散状态空间方面的优越性。

5 结论与展望

5.1 主要工作内容

本文深入研究了 VNF 放置与 VNF 调度的联合优化问题，并采用了两种不同的方法进行解决：

(1) 基于强化学习与鲸鱼优化算法的联合优化：为了最小化网络服务的完成时间，本文构建了相应的数学模型，并结合强化学习与改进的鲸鱼优化算法来解决VNF联合优化问题。在改进的鲸鱼优化算法中，本文采取了多项措施，包括种群的初始设置、编码和基于贪婪策略的解码，以及交叉和变异过程。此外，引入强化学习用于自适应调节参数，全面提升了算法的调度性能。实验结果显示，该算法不仅在收敛速度上表现出色，而且在网络服务的调度效果上也显著优于其他方法。

(2) 基于规则的 D3QN 调度算法：本文还采用基于规则的 D3QN 调度算法来应对网络调度挑战。首先设定了一系列规则，用于选择未处理的虚拟网络功能，并在每次调度中根据算法比较选择合适的节点和路由来进行分配。为了在不同的调度场景中选择最优规则，本文利用 D3QN 进行训练。通过在多种网络环境下的对比实验，验证了 D3QN 算法的稳定性、有效性和适用性。研究结果显示，经过训练的 D3QN 在性能上明显优于其他复合规则和随机选择策略。与传统的 DQN 相比，D3QN 展现出显著的优势，特别是在处理离散状态空间方面表现出高效性。此外，与遗传算法相比，D3QN 同样展示出更出色的性能。

5.2 不足与展望

虽然本文在 VNF 放置与调度联合处理领域已经取得了一些重要进展，展现了其算法在 VNF 放置与调度联合动态调整策略中的有效性，但在研究过程中也暴露出一些局限性和未来研究的可能方向：

(1) 基于强化学习与鲸鱼优化算法的联合优化：该算法在实际应用中仍存在一些潜在的缺点和限制。强化学习需要大量的训练数据和时间来优化参数，这在快速变化的网络环境中可能导致算法响应较慢。鲸鱼优化算法中的交叉和变异过程可能引入额外的计算复杂度，影响算法的实时性能。在未来的研究中，需要进一步优化算法设计，提高算法的实时性和鲁棒性。

(2) 基于规则的 D3QN 调度算法：规则的设定和选择对算法性能具有重要影响。如果规则设计不合理或训练数据不足，可能导致算法无法找到最优解。D3QN 算法在训练过程中可能面临过拟合或欠拟合的问题，需要采取适当的正则化策略来避免。在未来的研究中，需要进一步优化规则设定和算法设计，提高算法的稳定性和泛化能力。

在未来对 VNF 放置和调度优化问题的研究，可以从以下几点展开：

(1) 考虑更多实际因素的调度算法：未来的研究可以进一步考虑更多实际网络环境中的因素，例如节点容量限制、VNF 迁移、资源抢占等。这些因素的考虑将使得调度算法更加贴近实际网络场景，提高算法的实用性和适应性。

(2) 优化调度规则：对调度规则进行优化是一个重要的方向。通过深入研究网络功能的特性和网络流量的模式，可以设计更加智能和适应性强的调度规则，从而进一步提高 VNF 的放置和调度效率。

(3) 探索更先进的算法：除了已有的算法，未来可以进一步探索基于先进的深度强化学习的方法，通过深度神经网络来学习复杂的 VNF 放置和调度策略，以实现更高效的网络资源利用和服务质量提升。

(4) 面向多场景的适应性算法：由于网络环境和需求的多样性，未来的算法应该具备更好的适应性，能够在不同场景下实现高效的 VNF 放置和调度。因此，可以设计面向多场景的适应性算法，以满足不同应用场景的需求。

总的来说，未来的研究将更加注重提高 VNF 放置和调度的实际效用，同时结合更先进的算法和技术，以应对日益复杂和多样化的网络环境和服务需求。

参考文献

[1] Sherry J, Hasan S, Scott C, et al. Making middleboxes someone else's problem: Network processing as a cloud service[J]. ACM SIGCOMM Computer Communication Review, 2012, 42(4): 13-24.

[2] Mijumbi R, Serrat J, Gorricho J L, et al. Network Function Virtualization: State-of-the-Art and Research Challenges[J]. IEEE Communications Surveys & Tutorials, 2017, 18(1): 236-262.

[3] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, et al. Management and orchestration challenges in network functions virtualization[J]. Communications Magazine, IEEE, 2016, 54(1): 98–105.

[4] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee. Network function virtualization: Challenges and opportunities for innovations[J]. IEEE Communications Magazine, 2015, 53(2): 90–97.

[5] Alahmad Y, Agarwal A, Daradkeh T. Cost and availability-aware VNF selection and placement for network services in NFV[C]. International Symposium on Networks, Computers and Communications (ISNCC), Piscataway, 2020,1-6.

[6] LaazizL, kara N, Rabipour R, et al. FAST-SCALE: A fast and scalable evolutionary algorithm for the joint placement and chaining of virtualized services[J]. Journal of Network and Computer Applications, 2019, 148: 102429.

[7] Fu X, Yu F R, Wang J, et al. Service Function Chain Embedding for NFV-Enabled IoT Based on Deep Reinforcement Learning[J]. IEEE Communications Magazine, 2019, 57(11): 102–108.

[8] Alameddine H A, Qu L, Assi C. Scheduling service function chains for ultra-low latency network services[C]. International Conference on Network & Service Management. IEEE Computer Society, 2017,1-9.

[9] J. G. Herrera and J. F. Botero. Resource allocation in NFV: A comprehensive survey[J]. IEEE Transactions on Network and Service Management, 2016, 13(3): 518–532.

[10] Alahmad Y, Agarwal A. VNF placement strategy for availability and reliability of network services in NFV[C]. 2019 Sixth International Conference on Software Defined Systems (SDS). IEEE, 2019: 284-289.

[11] Abu-Lebdeh M, Naboulsi D, Glitho R, et al. NFV orchestrator placement for geo-distributed systems[C]. 2017 IEEE 16th International Symposium on Network Computing and Applications (NCA). IEEE, 2017: 1-5.

[12] Moens H, De Turck F. VNF-P: A model for efficient placement of virtualized network functions[C]. 10th international conference on network and service management (CNSM) and workshop. IEEE, 2014: 418-423.

[13] Morin C, Texier G, Caillouet C, et al. VNF placement algorithms to address the mono-and multi-tenant issues in edge and core networks[C]. 2019 IEEE 8th International Conference on Cloud Networking (CloudNet). IEEE, 2019: 1-6.

[14] Hmaity A, Savi M, Musumeci F, et al. Virtual network function placement for resilient service chain provisioning[C]. 2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM). IEEE, 2016: 245-252.

[15] Li D, Hong P, Xue K, et al. Virtual network function placement considering resource optimization and SFC requests in cloud datacenter[J]. IEEE Transactions on Parallel and Distributed Systems, 2018, 29(7): 1664-1677.

[16] Sun G, Xu Z, et al. Low-latency and resource-efficient service function chaining orchestration in network function virtualization[J]. IEEE Internet of Things Journal, 2020, 5760

[17] Hyodo N, Sato T, Shinkuma R, et al. Virtual network function placement for service chaining by relaxing visit order and non-loop constraints[J]. IEEE Access, 2019, 7: 165399-165410.

[18] Alhussein O, Do P T, Li J, et al. Joint VNF placement and multicast traffic routing in 5G core networks[C]. 2018 IEEE Global Communications Conference (GLOBECOM). IEEE, 2018: 1-6.

[19] Guo H, Wang Y, Li Z, et al. Cost-aware placement and chaining of service function chain with VNF instance sharing[C]. NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium. IEEE, 2020: 1-8.

[20] Feng H, Shu Z, Taleb T, et al. An aggressive migration strategy for service function chaining in the core cloud[J]. IEEE Transactions on Network and Service Management, 2023, 20(2): 2025-2039.

[21] Rankothge W, Le F, Russo A, et al. Optimizing resource allocation for virtualized network functions in a cloud center using genetic algorithms[J]. IEEE Transactions on Network and Service Management, 2017, 14(2): 343-356.

[22] Riera J F, Escalona E, Batalle J, et al. Virtual network function scheduling: Concept and challenges[C]. 2014 international conference on smart communications in network technologies (SaCoNeT). IEEE, 2014: 1-5.

[23] Li X, Qian C. Low-complexity multi-resource packet scheduling for network function virtualization[C]. 2015 IEEE Conference on Computer Communications (INFOCOM). IEEE, 2015: 1400-1408.

[24] Chen Y, Wu J. Flow scheduling of service chain processing in a NFV-based network[J]. IEEE Transactions on Network Science and Engineering, 2020, 8(1): 389-399.

[25] Yi, B., Wang, X. and Huang, M., 2017. A generalized VNF sharing approach for service scheduling[J]. IEEE Communications Letters, 22(1): 73-76.

[26] Assi C, Ayoubi S, El Khoury N, et al. Energy-aware mapping and scheduling of network flows with deadlines on VNFs[J]. IEEE Transactions on Green Communications and Networking, 2018, 3(1): 192-204.

[27] Pezzella F, Morganti G, Ciaschetti G. A genetic algorithm for the flexible job-shop scheduling problem[J]. Computers & operations research, 2008, 35(10): 3202-3212.

[28] Mijumbi R, Serrat J, Gorricho J L, et al. Design and evaluation of algorithms for mapping and scheduling of virtual network functions[C]. Proceedings of the 2015 1st IEEE conference on network softwarization (NetSoft). IEEE, 2015: 1-9.

[29] Qu L, Assi C, Shaban K. Delay-aware scheduling and resource optimization with network function virtualization[J]. IEEE Transactions on communications, 2016, 64(9): 3746-3758.

[30] Pham C, Tran N H, Hong C S. Virtual network function scheduling: A matching game approach[J]. IEEE Communications Letters, 2017, 22(1): 69-72.

[31] Li J, Shi W, Zhang N, et al. Delay-aware VNF scheduling: A reinforcement learning approach with variable action set[J]. IEEE Transactions on Cognitive Communications and Networking, 2020, 7(1): 304-318.

[32] Bello I, Pham H, QVLe, et al.Neural combinatorial optimization with reinforcement learning[J], arXiv:1611.09940,2016.

[33] Lei, Kun, et al. A multi-action deep reinforcement learning framework for flexible Job-shop scheduling problem[J].Expert Systems with Applications 205 (2022): 117796.

[34] Luo S. Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning[J]. Applied Soft Computing, 2020, 91: 106208.

[35] Liu R, Piplani R, Toro C. Deep reinforcement learning for dynamic scheduling of a flexible job shop[J]. International Journal of Production Research, 2022, 60(13): 4049-4069.

[36] Wang T, Zu J, Hu G, et al. Adaptive service function chain scheduling in mobile edge computing via deep reinforcement learning[J]. IEEE Access, 2020, 8: 164922-164935.

[37] Riggio R, Bradai A, Harutyunyan D, et al. Scheduling wireless virtual networks functions[J]. IEEE Transactions on network and service management, 2016, 13(2): 240-252.

[38] Yao H, Xiong M, Li H, et al. Joint optimization of function mapping and preemptive scheduling for service chains in network function virtualization[J]. Future Generation Computer Systems, 2020, 108: 1112-1118.

[39] Wang Z, Zhang J, Huang T, et al. Service function chain composition, placement, and assignment in data centers[J]. IEEE Transactions on Network and Service Management, 2019, 16(4): 1638-1650.

[40] dos Santos G M, Batista D M. On-demand placement and scheduling of virtual network functions with software requirements[C]. 2020 IEEE Latin-American Conference on Communications (LATINCOM). IEEE, 2020: 1-5.

[41] Gamal M, Jafarizadeh S, Abolhasan M, et al. Mapping and scheduling for non-uniform arrival of virtual network function (VNF) requests[C]. 2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall). IEEE, 2019: 1-6.

[42] Cao H, Du J, Zhao H, et al. Resource-ability assisted service function chain embedding and scheduling for 6G networks with virtualization[J]. IEEE Transactions on Vehicular Technology, 2021, 70(4): 3846-3859.

[43] Kuai Z, Wang T, Wang S. Fair virtual network function mapping and scheduling using proximal policy optimization[J]. IEEE Transactions on Communications, 2022, 70(11): 7434-7445.

[44] Hu F, Hao Q, Bao K. A survey on software-defined network and openflow: From concept to implementation[J]. IEEE Communications Surveys & Tutorials, IEEE, 2014, 16(4): 2181–2206.

[45] Lara A, Kolasani A, Ramamurthy B. Network innovation using openflow: A survey[J]. IEEE communications surveys & tutorials, 2013, 16(1): 493-512

[46] Schaffrath G, Werle C, Papadimitriou P, et al. Network virtualization architecture: Proposal and initial prototype[C]. Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures, 2009: 63-72.

[47] Chowdhury NMMK, Boutaba R. A survey of network virtualization[J]. Computer Networks, Elsevier, 2010, 54(5): 862–876.

[48] Chowdhury NMMK, Boutaba R. Network virtualization: state of the art and research challenges[J]. IEEE Communications magazine, IEEE, 2009, 47(7): 20–26.

[49] Gember A, Krishnamurthy A, John S S, et al. Stratos: A Network-Aware Orchestration Layer for Middleboxes in the Cloud[J]. CoRR, arXiv:1305.0209, 2013.

[50] GONG D, XU B, ZHANG Y, et al. A similarity-based cooperative co-evolutionary algorithm for dynamic interval multi-objective optimization problems[J]. IEEE Transactions on Evolutionary Computation, 2020, 24(1): 142-156.

[51] HOLLAND J H. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence[M]. MIT press, 1992: 22-56.

[52] Nasiri J, Khiyabani F M. A whale optimization algorithm (WOA) approach for clustering[J]. Cogent Mathematics & Statistics, 2018, 5(1): 1483565.

[53] Wang H, Gu M, Yu Q, et al. Adaptive and large-scale service composition based on deep reinforcement learning[J]. Knowledge-Based Systems, 2019, 180: 75-90.

[54] 龚铭凡,徐海祥,冯辉,等.基于改进 Q-Learning 的智能船舶局部路径规划[J].船舶力学,2022,26(06):824-833.

[55] 韩忻辰,俞胜平,袁志明,等.基于 Q-Learning 的高速铁路列车动态调度方法[J].控制理论与应用,2021,38(10):1511-1521.

[56] 唐伦,贺兰钦,连沁怡,等.基于改进深度强化学习的虚拟网络功能部署优化算法[J].电子与信息学报,2021,43(06):1724-1732.

[57] 赵蓓英,姬伟峰,翁江,等.基于启发式 Q 学习的 FANET 可信路由算法[J].计算机工程,2022,48(05):162-169.

[58] Mnih V, Kavukcuoglu K, Silver D, et al. Playing atari with deep reinforcement learning[J]. Computer Science, 2013，351-362.

[59] Sutton R S, Barto A G. Reinforcement Learning: An Introduction[J]. IEEE Transactions on Neural Networks, 1998, 9(5):1054.

[60] Hasselt H. Double Q-learning[J]. Advances in neural information processing systems, 2010, 7(1):23.

[61] Hasselt H V, Guez A, Silver D. Deep reinforcement learning with double Q-learning[J]. Computer ence, 2015,30(1):2094-2100.

[62] Wang Z, Schaul T, Hessel M, et al. Dueling network architectures for deep reinforcement learning[C]. International conference on machine learning. PMLR, 2016,1995-2003.

[63] 张超勇,管在林,刘琼,等.一种新调度类型及其在作业车间调度中的应用[J].机械工程学报,2008,(10):24-31.

[64] Gholipoor N, Saeedi H, Mokari N, et al. E2E QoS guarantee for the tactile internet via joint NFV and radio resource allocation[J]. IEEE Transactions on Network and Service Management, 2020, 17(3): 1788-1804.

[65] 刘光远,曹晶仪,庞紫园,等.一种低时延虚拟网络功能放置及调度优化算法[J].西安交通大学学报,2023,57(02):121-130.

[66] 李西兴,杨道明,李鑫,等.基于混合遗传鲸鱼优化算法的柔性作业车间自动导引车融合调度方法[J].中国机械工程,2021,32(08):938-950.

[67] 王琛,汤红波,游伟,等.一种 5G 网络低时延资源调度算法[J].西安交通大学学报,2018,52(04):117-124.

[68] Shu Z, Taleb T, Song J. Resource allocation modeling for finegranular network slicing in bey ond 5G systems[J]. IEICE Transactions on Communications, 2022, 105(4): 349-363.

[69] Qu L, Assi C, Shaban K. Delay-aware scheduling and resource optimization with network function virtualization[J]. IEEE Transactions on communications, 2016, 64(9): 3746-3758.

[70] Promwongsa N, Ebrahimzadeh A, Glitho R H, et al. Joint VNF placement and scheduling for latency-sensitive services[J]. IEEE Transactions on Network Science and Engineering, 2022, 9(4): 2432-2449.

攻读学位期间的学术论文与研究成果

[1] xxxx, et al. "Low-latency Virtual Network function Scheduling Algorithm Basedon Deep Reinforcement Learning." Computer Networks 2024, 110418.(SCI 2 区，IF 5.6,第一作者)

[2] 发表专利：核心网中网络切片的主动迁移策略。第三作者

[3] xxxx, et al. "An aggressive migration strategy for service function chaining in the core cloud." IEEE Transactions on Network and Service Management 2023,20(2): 2025-2039.(SCI 2 区，IF 5.3，第五作者)

[4] xxxx, et al. "Bandwidth Resources Prediction for Dynamic Networking Slicing Based on Time Series Analysis." 2022 International Conference on Machine Learning, Cloud Computing and Intelligent Mining (MLCCIM). IEEE, 2022,1-5(EI 会议，第五作者)

致谢

时光荏苒，转眼间我在研究生的学习生涯即将画上圆满的句号。在此，我谨以此篇致谢，向在我求学道路上给予关心、支持与帮助的各位老师、同学、家人和朋友表示最诚挚的感谢。

首先，我要衷心感谢我的导师。在论文的选题、研究、撰写过程中，导师严谨的治学态度、敏锐的思维和对知识的执着追求给予了我极大的启发和鼓舞。在学术上，导师为我提供了宝贵的意见和建议，使我受益匪浅；在生活中，导师关心我的成长，给予我无微不至的关怀。在此，向您表示由衷的敬意和感激。

感谢实验室的全体老师和同学们，在学术探讨、实验操作和日常生活方面给予我的帮助和支持。特别感谢我的师兄师姐们，你们的经验分享让我少走了很多弯路。同时，感谢我的师弟师妹们，你们的活力和热情让我感受到了团队的力量。

感谢我的家人，是你们一直以来的关爱和支持让我在求学的道路上勇往直前。你们的理解和鼓励是我克服困难、不断前行的动力源泉。

感谢我的朋友们，在研究生生涯中，你们陪伴我度过了许多难忘的时光。你们的关心和鼓励让我感受到了友谊的温暖，使我更有信心面对挑战。

最后，感谢我国高等教育政策，为我提供了良好的学术环境和学习机会。在此，衷心祝愿我国的科研事业蒸蒸日上，繁荣昌盛！

谨以此致谢，献给我最尊敬、最感激的导师、家人、同学和朋友。虽然篇幅有限，但我的感激之情无法用言语表达，愿这份感激化作前行的动力，继续在学术和人生的道路上努力拼搏，不负众望。