计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2024,33(7):230−238 [doi: 10.15888/j.cnki.csa.009557]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于深度强化学习的能源高效 VNF 放置和链接方法①

赵耀鹏, 徐九韵, 脱颖超

(中国石油大学 (华东) 计算机科学与技术学院, 青岛 266580)
通信作者: 徐九韵, E-mail: Jiuyun.xu@computer.org

摘　要: 网络功能虚拟化 (NFV) 技术的出现使得网络功能由虚拟网络功能 (VNF) 提供, 从而提高网络的灵活性, 可扩展性和成本效益. 然而, NFV 面临一个重要挑战是, 如何有效地将 VNF 放置不同的网络位置并链接起来引导流量, 同时最大限度减少能源消耗. 此外, 面对网络服务质量要求, 提高服务接受率对于网络性能也是至关重要的. 为了解决这些问题, 本文研究了 NFV 中的 VNF 放置和链接 (VNFPC), 以最大化服务接受率同时权衡优化能源消耗. 因此, 在 NFV 中设计了一种基于 Actor-Critic 深度强化学习 (DRL) 的能源高效的 VNFPC 方法, 称为 ACDRL-VNFPC. 该方法应用了适应性共享方案, 通过在多服务之间共享同类型 VNF 和多 VNF 共享同一个服务器来实现节能. 实验结果表明, 提出的算法有效权衡了能耗和服务接受率, 并且, 在执行时间方面也得到了优化. 与基准算法相比, ACDRL-VNFPC 在服务接受率, 能耗和执行时间方面性能分别提高了 2.39%, 14.93% 和 16.16%.

关键词: 网络功能虚拟化; 虚拟网络功能放置; 服务功能链; 能源高效

引用格式: 赵耀鹏,徐九韵,脱颖超.基于深度强化学习的能源高效 VNF 放置和链接方法.计算机系统应用,2024,33(7):230–238. http://www.c-s-a.org.cn/1003-3254/9557.html

Energy Efficient VNF Placement and Chaining Approach Based on Deep Reinforcement Learning

ZHAO Yao-Peng, XU Jiu-Yun, TUO Ying-Chao

(College of Computer Science and Technology, China University of Petroleum, Qingdao 266580, China)

Abstract: The emergence of network function virtualization (NFV) technology allows network functions are provided by virtual network functions (VNFs) to improve network flexibility, scalability and cost-effectiveness. However, an important challenge for NFV is how to efficiently place VNFs in different network locations and chain them to steer traffic while minimizing energy consumption. In addition, in the face of network quality of service requirements, improving the service acceptance rate is also critical to network performance. To address these issues, in this study we investigate VNF placement and chaining (VNFPC) in NFV to maximize the service acceptance rate while optimizing the energy consumption trade-off. Therefore, an energy-efficient VNFPC method based on Actor-Critic deep reinforcement learning (DRL), called ACDRL-VNFPC, is designed in NFV. The approach applies adaptive sharing scheme to achieve energy savings by sharing the same type of VNFs among multiple services and sharing the same server among multiple VNFs. The experiment results show that the proposed algorithm effectively trades off the energy consumption and service acceptance rate, and the execution time is also optimized. Compared with the baseline algorithm, ACDRL-VNFPC improves the performance in terms of service acceptance rate, energy consumption and execution time by 2.39%, 14.93%

① 基金项目: 中央高校基本科研业务费自主创新项目 (18CX02140A)
收稿时间: 2024-01-12; 修改时间: 2024-02-07; 采用时间: 2024-02-26; csa 在线出版时间: 2024-05-31
CNKI 网络首发时间: 2024-06-04

and 16.16%, respectively.

Key words: network function virtualization (NFV); virtual network function (VNF) placement; service function chain (SFC); energy efficient

传统的网络服务需要专用硬件来提供, 并按特定顺序引导通过硬件的流量. 然而, 这些专用硬件设备不仅带来了高资本支出和运营支出, 而且管理困难[1,2]. 为了解决以上问题, 研究人员提出了网络功能虚拟化 (network function virtualization, NFV)[1–3]这一新的网络架构概念, 通过将网络功能 (例如防火墙, 网络地址转换和入侵检测系统) 与专用硬件解耦, 转变为通用设备基于软件实现的虚拟网络功能 (virtual network function, VNF)[4], 从而实现网络服务的灵活和弹性管理. 到达网络的服务请求被转化为服务功能链 (service function chain, SFC)[5–7], 它由一组特定有序的 VNF 构成. 如图 1 所示是 SFC 示例. 利用 NFV 技术, 将 SFC 中的 VNF 部署到底层物理网络的服务器或虚拟机上, 以实现灵活高效的网络服务. VNF 放置和链接 (VNF placement and chaining, VNFPC) 过程是指在 NFV 网络中的服务器上放置 VNF 并将它们按序链接起来引导流量, 来提供特定的服务[4].

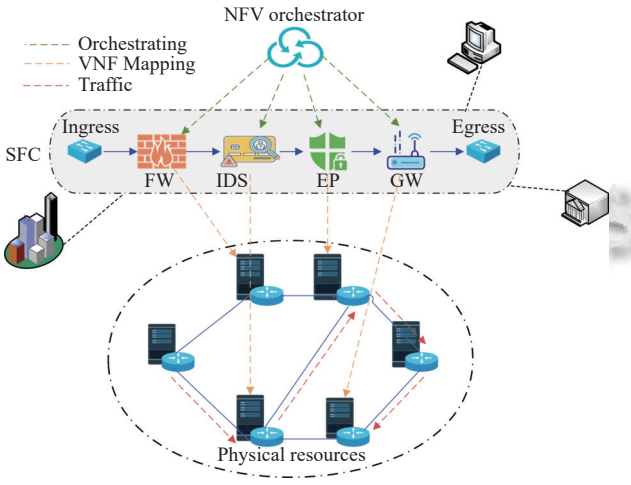


图 1 VNFPC 场景

一般来说, 合理的 VNFPC 方案受到一些优化目标的约束, 例如延迟, 资源利用, 能耗, 运营成本和流量. 之前的研究已经在单目标做了大量工作, 推动 NFV 在学术界和工业界的发展[6–8], 然而, NFV 在多个目标优化进行有效的 VNFPC 需要考虑几个关键挑战. 特别是, 为了将 SFC 部署到异构网络功能, 服务提供商需要在不同的目标之间进行权衡, 例如最小化能耗和最大化服务接受率. 这些目标是相互冲突的, 因为在最小化能耗可能会激活更少的服务器, 从而用户服务可能由于资源或者网络延迟约束被拒绝. 同时, 最大化服务接受率可能需要花费更多资源, 优化服务接受率时可能会增加能耗. 为此, 本文重点讨论了如何在 NFV 网络权衡优化 VNFPC 问题的能耗和服务接受率.

图 1 展示了 VNFPC 场景, 各种网络服务如视频聊天, Web 服务等网络请求被 NFV 编排成 SFC, 其流量从入口节点经过不同 VNF 到达出口节点实现端到端的服务. 具体是将 VNF 映射到不同的物理节点上进行资源分配和优化, 进行流量引导和控制, 来提供高效的网络服务.

现有的大多数研究人员将 VNFPC 问题表述为整数线性规划 (integer linear programming, ILP)[9,10]模型进行最优化求解. 文献[9]面对功耗感知和延迟约束的联合 VNF 放置和路由 (PD-VPR) 问题提出一种 Holu 框架可以在线解决 PD-VPR 问题, 降低能耗同时提高了服务接受率. 在考虑了端到传输延迟和 VNF 亲和力约束时, 文献[10]是基于 VNF 链放置问题 (VNF-CPP) 提供了基于 ILP 在线解决方案, 并验证所提算法的有效性. 此外, 文献[11]将问题表述为混合整数规划 (mixed integer linear programming, MILP) 模型, 通过分解模型开发出一种两级算法实现 SFC 部署, 仿真结果表明该方法实现了接近最优的解决方案. 然而, 由于 VNFPC 是 NP-Hard 特性, 因此 LP 方法仅只能在小规模网络中进行优化, 当网络规模过大时间效率变得很低.

一些研究进一步提出了启发式算法来应对大规模用户服务请求, 并有效指导 LP 问题在有限的时间内求解. 文献[12]优化 VNFPC 的资源分配成本, 通过将问题表述为 LP 模型, 提出了基于背包问题的启发式方法高效解决问题, 并降低了 SFC 的平均成本. 在文献[13]中, 将部署问题建模为 ILP 优化, 并扩展到启发式算法用于估计瓶颈资源占用, 来实现接近最优的部署. 虽然启发式方法很高效, 但解的最优性无法保证, 而且容易陷入局部最优解, 甚至偏离全局最优解.

近年来, 深度强化学习 (deep reinforcement learning, DRL) 也被应用在 VNFPC[14,15]中寻找问题最优解决方案. 该方法不同于传统的方法求解容易陷入局部最优问题, DRL 在解决组合优化问题具有前瞻性的解决方案. 文献[14]中考虑边缘网络资源受限且对于服务的延迟容忍性低的问题, 结合强化学习与时延提出一种面向时延优化的 SFC 的部署方法, 使得时延敏感类业务获得更好体验. 文献[15]的工作中, 为了在服务功能链放置 (SFCP) 问题中提出了基于 DRL 的多目标优化服务功能链放置 (MOO-SFCP) 算法, 具体是将 SFCP 建模为马尔可夫决策过程 (Markov decision process, MDP), 并使用两层策略网络作为智能代理与环境交互进行学习. 上述方案学习模型明显提高求解问题的效率, 但其方案需要严格优化函数进行约束.

综合之前的研究, 在 NFV 中进行 VNFPC 面临以下问题. 首先是解决方案如线性规划或启发式算法很难准确高效地找到最优解决策略. 其次, 之前的研究大多是单个优化目标的约束, 很少研究涉及到多目标优化, 特别是, 优化目标之间存在冲突的. 综上所述, 本文提出一种新的算法 ACDRL-VNFPC, 具体的贡献点如下.

(1) 首先, 将 VNFPC 问题化为一个优化问题, 它考虑了能耗, 服务接受率, 延迟和资源需求.

(2) 接着, 通过建立一个目标函数来联合考虑能耗和服务接受率, 旨在最大化服务接受率同时降低能耗.

(3) 然后, 为了获得优化问题的解决方案, 提出了基于 Actor-Critic 深度强化学习方法 (ACDRL-VNFPC), 并在该方法中采用了一种适应性共享策略, 该策略不仅允许多个服务请求之间共享未充分使用的同类型 VNF, 通过实例化更少的 VNF 减少服务器资源消耗; 而且允许服务器中的资源被不同类型的 VNF 共享, 这样激活了更少的服务器同时降低了能耗.

(4) 最后, 通过构建仿真环境与其他方法进行对比. 实验结果表明, 提出的算法可以获得问题的解, 并且可以权衡优化能耗和服务接受率, 同时执行时间方面也得到了优化.

1 系统模型和问题描述

1.1 物理网络模型

物理网络被建模为无向加权图$G_p=(N_p,E_p)$, 其中N_p和E_p分别表示物理节点和物理通信链路的集合. 每个物理节点v_i (包括交换机和服务器), $v_i\in N_p$具有有限的物理资源. 计算资源分别用$R^p_{v_i}=(cpu^p_{v_i},mem^p_{v_i},str^p_{v_i})$表示每个节点$v_i$的可用的 CPU, 内存和存储的资源. 物理的通信链路用E_p来表示, 其中e_{pq}表示在物理网络中两个物理节点v_p和v_q直接相连, 则链$(p,q)\in E_p$ (即$e_{pq}=(p,q)$), 否则$(p,q)\notin E_p$. 对于每个物理链路$e_{pq}\in E_p$, 用$BW^p_{e_{pq}}$来表示链路e_{pq}的可用带宽容量, 并且$L^p_{e_{pq}}$代表了物理链路的传输时延. 最后用$\Phi^{res}_{v_i}$代表节点资源单位成本, $\Phi^{bw}_{e_{pq}}$来表示链路带宽的单位成本.

1.2 SFC 请求模型

假设有m个 SFC 请求, 每个 SFCsfc_i请求被建模为一个加权有向图$sfc_i=(f_i^1,f_i^2,\cdots,f_i^j,\cdots,f_i^n)$, 其中$f_i^j$表示在 SFC sfc_i中的第j个 VNF 节点, 用E^v_i表示 VNF 之间相连的虚拟链路集合. 分别用$r^v_{f_i^j}=(r^{cpu}_{f_i^j},r^{mem}_{f_i^j},r^{str}_{f_i^j})$表示 VNF f_i^j的 CPU, 内存和存储的资源需求. 通过$l^d_{f_i^j}$来表示 VNF f_i^j的处理时延. SFCsfc_i中的虚拟链路用l_i^{uv}, $l_i^{uv}\in E_i^v$表示两个 VNF f_i^u和f_i^v之间的链路, 链路之间的带宽需求用$bw^v_{l_i^{uv}}$来表示.

1.3 问题公式化

在 NFV 中进行 VNFPC 需要对能耗和服务接受率提出了更严格的要求. 因此, 本文需要权衡优化最小化能耗和最大化服务接受率. 针对 VNFPC 问题的研究目标是寻找一个最优策略π^*来最大化服务接受率和最小化能耗, 目标函数如下:

$$\pi^*=\arg\max_{\pi}\left\{\psi\cdot Ac(t)+(1-\psi)\cdot\frac{1}{P_{v_i}(t)}\right\} \tag{1}$$

其中, ψ表示权重因子, 用于实现服务接受率/能耗的权衡优化. 例如ψ越小, 越强调能耗优化, 反之是服务接受率.

服务接受率$Ac(t)$: 为了满足用户服务质量要求, 服务提供商将成本消耗降至最低, 并尽可能多地部署 SFC. 通过接受的 SFC 请求与所有 SFC 请求的比率来衡量请求接受率. 在时间t $(0\leqslant t\leqslant T)$服务接受率$Ac(t)$表示为:

$$Ac(t)=\frac{\sum_{t=0}^{T}sfc_i^{\text{accept}}}{\sum_{t=0}^{T}sfc_i^{\text{accept}}+\sum_{t=0}^{T}sfc_i^{\text{refuse}}} \tag{2}$$

其中, sfc_i^{accept}表示已经接受并成功部署的 SFC 请求数量, sfc_i^{refuse}表示由于资源或延迟被拒绝的 SFC 请求的数量.

能耗$P_{v_i}(t)$: 能耗主要来源于服务器节点的能耗, 包

括空闲状态和工作状态的能耗. 在时间t服务器节点/机器的能耗与服务器的资源利用率相关[8]. 因此, 在时间t服务器v_i的能耗$P_{v_i}(t)$表示为:

$$P_{v_i}(t)=P_{v_i}^{\text{idle}}(t)+(P_{v_i}^{M}(t)-P_{v_i}^{\text{idle}}(t))\times U_{v_i}(t) \tag{3}$$

其中, $P_{v_i}^{\text{idle}}(t)$为服务器v_i处于空闲状态时的能耗, $P_{v_i}^{M}(t)$为当服务器满载时的最大能耗, $U_{v_i}(t)\in[0,1]$表示服务器的资源利用率.

上述目标函数需要遵循以下约束.

式 (4) 定义了决策变量$x_{f_j}^{v_i}$, 当$x_{f_j^i}^{v_i}=1$表示 VNF f_i^j部署到服务器节点v_i, 否则未部署. 决策变量$x_{f_j^i}^{v_i}$表示为:

$$x_{f_j^i}^{v_i}=\begin{cases}1, & \text{if VNF } f_i^j \text{ is deployed on } v_i\\ 0, & \text{otherwise}\end{cases} \tag{4}$$

式 (5) 约束是为确保每个 VNF f_i^j不可分割, 且仅部署一次.

$$\forall i,j\quad \sum_{i=1}^{|N_p|}\sum_{j=1}^{n}x_{v_i}^{f_i^j}\leqslant 1,\forall f_j^i\in sfc_i,v_i\in N_p \tag{5}$$

当选择适当的服务器部署 VNF 时, VNF 之间的虚拟链路也需要映射到适当的物理链路. 决策变量$y_{e_{pq}}^{l_i^{uv}}$表示虚拟链路l_i^{uv}与物理链路e_{pq}的映射关系. 式 (6) 的决策变量$y_{e_{pq}}^{l_i^{uv}}$表示为:

$$y_{e_{pq}}^{l_i^{uv}}=\begin{cases}1, & \text{if } l_i^{uv} \text{ is mapped in } e_{pq}\\ 0, & \text{otherwise}\end{cases} \tag{6}$$

式 (7) 为确保每个虚拟链路仅映射到唯一的物理链路上, 且仅映射一次.

$$\forall u,v,p,q\quad \sum_{i=1}^{m}\sum_{u,v}^{n}y_{e_{pq}}^{l_i^{uv}}\leqslant 1,\forall l_i^{uv}\in sfc_i,e_{pq}\in E_p \tag{7}$$

式 (8) 定义了物理节点计算资源容量的约束. 任何部署成功的 VNF, 其所有的资源请求不超过物理节点的可用资源需求.

$$\begin{cases}\forall i,j\quad \dfrac{\sum\limits_{f_i^j\in sfc_i}x_{f_i^j}^{v_i}\cdot r_{f_i^j}^{cpu}}{cpu_{v_i}^{p}}\leqslant 1\\ \forall i,j\quad \dfrac{\sum\limits_{f_i^j\in sfc_i}x_{f_i^j}^{v_i}\cdot r_{f_i^j}^{mem}}{mem_{v_i}^{p}}\leqslant 1\\ \forall i,j\quad \dfrac{\sum\limits_{f_i^j\in sfc_i}x_{f_i^j}^{v_i}\cdot r_{f_i^j}^{str}}{str_{v_i}^{p}}\leqslant 1\end{cases} \tag{8}$$

式 (9) 定义了通信资源的容量约束. 为了确保虚拟链路l_i^{uv}的带宽资源需求不超过物理链路e_{pq}的可用剩余带宽资源.

$$\forall i,j,u,v,p,q\quad \frac{\sum\limits_{f_i^j\in sfc_i,l_i^{uv}\in e_{pq}}y_{e_{pq}}^{l_i^{uv}}\cdot bw_{l_i^{uv}}^{v}}{BW_{e_{pq}}^{p}}\leqslant 1 \tag{9}$$

式 (10) 引入了延迟约束, 使用变量D_i^t表示在时间t SFC sfc_i的最大可容忍的服务延迟. 请注意, 这里不仅考虑物理节点之间的通信时延, 也考虑 VNF 的处理时延.

$$\forall i,j,u,v,p,q\quad \sum_{i=1}^{m}\sum_{j=1}^{n}x_{f_i^j}^{v_i}\cdot l_{f_i^j}^{d}\cdot\alpha+\beta\cdot\sum_{e_{pq}\in E_p}y_{e_{pq}}^{l_i^{uv}}\cdot L_{e_{pq}}^{p}\leqslant D_i^t \tag{10}$$

其中, $\alpha,\beta\in[0,1]$表示可调参数, 当$\beta=0$表示 VNF 被部署到同一个物理节点, 仅存在 VNF 处理时延.

2 算法设计

在 DRL 中, 智能体学习做决策通过探索未知环境并应用收到的反馈信息, 在每步学习中, 智能体会观察当前状态并基于某个策略采取行动, 然后智能体被反馈一个奖励. 本节首先介绍 MDP 模型, 然后介绍通过 Actor-Critic 网络进行训练, 最后介绍 ACDRL-VNFPC 算法.

2.1 MDP 模型

MDP 通常提供了一个长期问题中的行动制定的数学框架, 其中输出一般是智能体的随机行为. 此外在 MDP 模型中设定了假设. 即智能体对环境具有全部的感知能力, 并且当前状态排除了任何不确定性[16,17]. 接着定义 MDP 四元组$(\mathcal{S},\mathcal{A},\mathcal{P},\mathcal{R})$. 其中, $\mathcal{S}$表示状态空间, $\mathcal{A}$表示动作空间, $\mathcal{P}$表示状态转移概率, 而$\mathcal{R}$表示奖励. 在每个时间t $(t=1,2,\cdots,T)$, 来描述这几个关键元素.

1) 状态空间: 状态空间$\mathcal{S}_t$包括物理网络节点和链路资源信息和 SFC 资源请求信息. 具体地定义状态$s_t=(s_t^p,s_t^v)$, 分别表示物理网络状态和 SFC 请求信息.

2) 动作空间: 动作空间$\mathcal{A}_t$被定义为$a_t\in\mathcal{A}_t$则是从可用资源超过 VNF f_i^j选择一个节点v_i, 否则, $a_t=\bar{\zeta}$直接到达终止状态.

3) 状态转移概率: 转移概率$\mathcal{P}_t$表示为, 通过s_{t-1}和a_{t-1}转移到下一状态s_t的概率.

4) 奖励: 奖励$\mathcal{R}_t$是为了鼓励智能体以最大化长期平均收益为目标来进行放置. 当 SFC sfc_i在时间t被接

受时获得的收益.

$$rev(sfc_i)=\sum_{i=1}^{m}\sum_{j=1}^{n}x_{f_i^j}^{v_i}\cdot\Phi_{v_i}^{res}+\sum_{e_{pq}\in E_p}y_{e_{pq}}^{l_i^{uv}}\cdot\Phi_{e_{pq}}^{bw} \tag{11}$$

在中间步骤 (即$t<T$), 当满足资源约束时, 智能体获得激励的奖励$r_t=\xi rev_t(sfc_i)$, 否则获得抑制的奖励$r_t=-\xi rev_t(sfc_i)$, ξ为奖励系数.

2.2 Actor-Critic 网络训练

Critic 网络表示价值网络给采取的动作进行评估. 若ω表示价值网络的训练参数, 用价值网络$q(s,a;\omega)$来近似$Q_\pi(s,a)$. 更新 Critic 网络为了更好地估计回报值, 这使得 Critic 判断更加准确. 而在实际训练运用 TD (temporal difference) 算法来更新ω (在价值网络q). 通过计算损失$L(\omega)$, 即预测值q与目标 TD target y_t差值, 其被表示为如下:

$$L(\omega)=\frac{1}{2}[q(s_t,a_t;w)-y_t]^2 \tag{12}$$

其中, 损失是价值网络参数ω的函数, 并使用梯度下降更新ω, 使得y_t-q_t值更小, Critic 判断更准确. 因此, Critic 网络参数ω按照式 (13) 进行更新:

$$\omega_{t+1}=\omega_t-\alpha\cdot\frac{\partial L(\omega)}{\partial\omega}|_{\omega=\omega_t} \tag{13}$$

其中, α表示学习率.

Actor 网络表示策略网络来生成相应的动作和环境交互. 若π和θ分别表示策略网络和 Actor 网络的训练参数, 用策略网络$\pi(a\mid s;\theta)$来近似$\pi(a\mid s)$来负责评估 Actor 的表现, 并指导下一时刻的动作. 因此, 在 Actor 网络中, 状态价值函数可近似为:

$$V(s_t;\theta,\omega)=\sum_a\pi(a\mid s)\cdot Q_\pi(s,a)\approx\sum_a\pi(a_t\mid s_t;\theta)\cdot q_\pi(s_t,a_t;\omega) \tag{14}$$

更新策略网络π为了使得V函数更大, 并运用梯度下降更新θ(在策略网络π). 首先是函数V关于θ计算梯度, 用$g(a,\theta)$表示. 其次对θ做梯度上升, 更新状态价值函数V值.

$$\theta_{t+1}=\theta_t+\beta\cdot g(a,\theta_t) \tag{15}$$

其中, β表示学习率.

具体的 Actor-Critic 网络训练过程如算法 1 所示.

算法 1. Actor-Critic 网络训练

输入:当前网络状态s_t, 最大迭代轮次$iter_{\max}$.
输出: 在线训练的 Actor 网络参数θ.

/**训练阶段**/
1) 初始化神经网络参数θ和ω;
2) 初始化迭代次数$iter$=0;
3) **while** $iter<iter_{\max}$ **do**
4) **for** t=1,2,$\cdots$,T **do**
5) 观察状态s_t, 并随机抽样出动作a_t;
6) 执行动作a_t, 状态s_t更新到s_{t+1}得出r_t获得四元组(s_t,a_t,r_t,s_{t+1})保存到经验池;
7) 通过式 (13) 更新价值网络q参数ω;
8) 使用式 (15) 进行梯度上升更新策略网络π参数θ;
9) **end for**
10) **end while**
11) **return** θ;

2.3 ACDRL-VNFPC

如图 2 所示, ACDRL-VNFPC 整体框架包含 6 个步骤, 当请求 SFC 到达时, 我们首先对物理网络进行特征提取, 包括节点和链路的可用资源信息和 SFC 中的 VNF 资源请求信息. 在步骤 1 中将环境中的状态信息传递给学习代理 (即算法 1 训练 Actor-Critic 网络); 其次在步骤 2 根据最大状态值获得动作部署 SFC 请求 (即算法 2 所描述 VNF 放置和链路映射); 步骤 3 将采取的动作奖励反馈给环境; 接着, 在步骤 4 将四元组(s_t,a_t,r_t,s_{t+1})存储到经验回放池; 在步骤 5 计算损失函数并更新 Critic 网络; 最后步骤 6 通过 Critic 判断更新 Actor 网络.

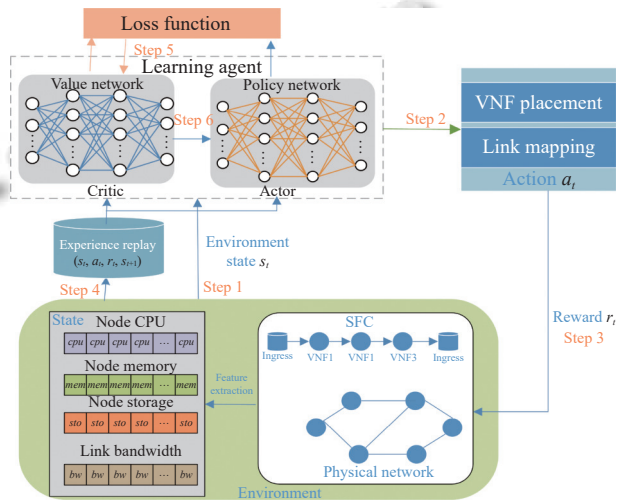


图 2 ACDRL-VNFPC 的框架

算法 2. VNF 放置和链路映射

输入: 物理网络G_p, SFC 请求sfc_i.
输出: 在线 VNF 放置集$placeSet_{\text{result}}$, 链路集$linkSet_{\text{result}}$.

/**VNF 放置**/
1) 初始化 Actor 网络参数θ;

2) $placeSet_{\text{result}}=\varnothing$, $linkSet_{\text{result}}=\varnothing$;
3) **for** t=1,2,⋯,T **do**
4) 按可用节点资源$R^p_{v_i}$和链路可用带宽资源$BW^p_{e_{pq}}$排序;
5) 当 SFC 请求随机到达;
6) **if** $a_t==\bar{\zeta}$ **then**
7) 到达终止状态, 拒绝当前sfc_i, 撤销之前动作a_t;
8) **return** false;
9) **else**
10) 当sfc_i和sfc_j具有相同类型的f^j_i可共享f^j_i处理能力;
11) 执行f^j_i放置到v_i动作a_t;
12) 更新放置结果集$place_{\text{result}}.add(a_t)$;
13) **continue**;
14) **end if**
/**链路映射**/
15) 从排序的候选链路根据 Dijkstra 算法在a_t 和a_{t-1}寻找路径最短的链路p_t;
16) **if** $\exists p_t$ **then**
17) 将虚拟链路映射到物理链路p_t;
18) 更新链路集$linkSet_{\text{result}}.add(p_t)$;
19) **continue**;
20) **end if**
21) **end for**
22) **return** $placeSet_{\text{result}}$, $linkSet_{\text{result}}$;

算法 2 是 VNF 放置及链路映射, 输入当前的物理网络和 SFC 请求, 输出 VNF 放置结果集合和链路映射结果集合. 首先对于物理节点资源和链路带宽资源按非递减的排序, 倾向于先选择较多资源的节点和链路. 然后对于随即到达的 SFC 中 VNF 进行放置, 当没有可供选择的物理节点, 则到达终止状态$\bar{\zeta}$, 并且拒绝该 SFC, 撤销之前放置动作a_t. 在实际放置中利用多服务共享相同类型的 VNF, 减少 VNF 实例数, 节省资源消耗. 从已执行动作a_t和a_t根据 Dijkstra 算法寻找最短链路, 如果存在则更新链路结果集.

3 实验与性能分析

3.1 仿真环境设置

本研究的实验全部在仿真环境上进行, 配置如下: 仿真计算机使用的是 NVIDIA GTX 3060 GPU, Intel(R) Core(TM) i7-11800H CPU 3.20 GHz, 仿真使用的软件 PyCharm 2021.2.2, Python 3.9.

根据 Waxman 模型[18]生成物理网络和虚拟网络的拓扑结构. 在每一轮次中, 随机生成 5 000 个 SFC 请求, 随机确定源节点和目的节点. 具体是每个 SFC 请求由 5–10 个不同数量的 VNF 均匀分布. 此外, 假设每次 SFC 请求到达率服从泊松分布参数为$\lambda=5$. 具体的仿真参数如表 1 所示.

为了评估提出的放置算法的有效性和效率, 并将其与增强 First-Fit (IFF) 启发式算法[19], 邻近节点蒙特卡洛 (NeMC)[20]树搜索、VNF 放置和网络缩放 (VPANS)[21]这 3 种方法进行对比. 算法的具体描述如表 2 所示. 请注意, ACDRL-VNFPC 与对比算法均使用最短路径算法来选择链路.

表 1 仿真参数设置

参数	值	描述
$\lvert N_p\rvert,\lvert E_p\rvert$	50, 250	物理网络的节点数和链路数
$R^p_{v_i}$, $BW^p_{e_{pq}}$	[50, 100]	节点资源和链路资源
$\Phi^{res}_{v_i}$, $\Phi^{bw}_{e_{pq}}$	[0.01, 0.05]	单位节点资源和带宽资源的成本
$P^{\text{idle}}_t(v_i)$, $P^M_t(v_i)$	50, 150	物理节点空闲与最大能耗
$r^v_{f^j_i}$, $bw^v_{l^{uv}_i}$	[5,10]	SFC请求中节点和链路资源需求
α	0.005	Critic的学习率
β	0.05	Actor的学习率
γ	0.90	折扣因子
ξ	0.1	奖励系数

表 2 算法描述

算法	描述
IFF (Baseline)	首先根据节点的可用资源进行排序, 然后对于SFC中的VNF每次选择第1个满足需求的节点
NeMC	NeMC属于强化学习算法范畴, 应用树的分支结构从根节点开始遍历, 并对当前节点的邻接矩阵的值进行评估, 选择最好的节点进行行为决策
VPANS	根据资源能效降序为每个VNF进行排序创建节点偏好列表, 去掉处理能力不足以被节点剩余处理能力满足, 或者不满足链路带宽约束的VNF. 因此每次根据偏好列表优先部署偏好列表的VNF

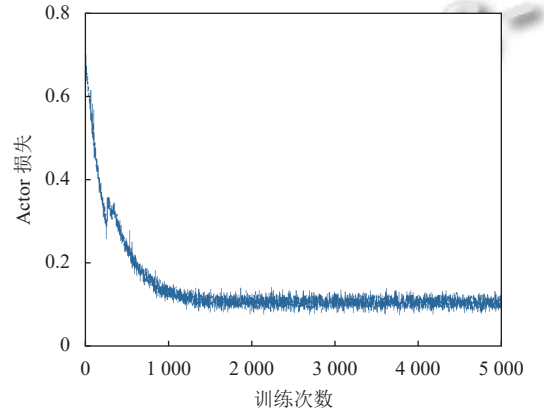
3.2 结果分析

在本研究中, 考虑分别用服务请求接受率, 总能耗和执行时间指标来衡量算法性能.

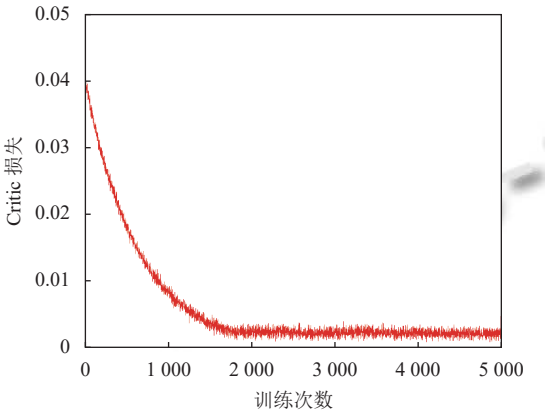
图 3(a) 和 (b) 显示了 Actor-Critic 网络训练过程损失变化, 可以看到损失随着训练数量的增加, 损失呈下降趋势. 当训练次数小于 1 500 次时, 损失值比较高, 这是因为初始模型的参数是随机初始化, 模型是未收敛, 导致损失值比较高. 随着训练次数的增加, 模型不断收敛. 根据奖励和状态值不断学习更新神经网络, 损失值逐渐稳定. 在训练次数大约 4 000 次后, Actor 和 Critic 损失分别收敛到约 0.10 和 0.002 5 局部最优. 这表明提出的 ACDRL-VNFPC 算法近似效果很好.

图 4 展示了 5 000 个 SFC 请求输入到网络中各算法的表现. 如图 4 所示, 服务接受率随着 SFC 请求数量

增多, 各算法的接受率呈下降趋势. 这是因为当 SFC 请求中的 VNF 被成功放置后, 底层物理资源被消耗逐渐减少. 由于 IFF 和 NeMC 优先考虑资源最小化, 但未考虑节点之间负载均衡, 因此算法的整体性能表现相对较差. ACDRL-VNFPC 算法由于应用了适应性共享, 在资源方面考虑了共享, 优化了节点部署, 因此 ACDRL-VNFPC 算法长期来看表现最好. 当 SFC 请求数量达到 1 500 时, ACDRL-VNFPC 算法请求接受率均高于对比算法. 提出的算法 ACDRL-VNFPC 在服务接受率较 IFF 算法性能提升了 2.39%, 较 NeMC 算法性能提升了 3.02%, 较 VPANS 算法性能提升了 2.57%. 这表明本文提出算法明显提高整体网络性能和服务质量.



(a) Actor 损失变化



(b) Critic 损失变化

图 3 训练次数对于 Actor 与 Critic 损失的影响

图 5 展示了随着 SFC 请求数量增加, 4 种算法能耗呈上升趋势. 这是因为随着要部署 SFC 请求数量增多, 节点需要处理更多的数据流量和服务请求, 导致需要启用更多的服务器来放置 VNF, 从而消耗更多的能源. 由于 ACDRL-VNFPC 基于 RL 方法学习最优部署策略, 通过部署到合理的服务器降低服务器的资源利用率, 从而减少能耗. 此外, ACDRL-VNFPC 考虑了适应性共享策略, 使得不同类型的 VNF 共享同一个服务器, 减少实例服务器的数量的同时降低了能耗成本, 因此, ACDRL-VNFPC 算法在能耗方面性能表现最好. ACDRL-VNFPC 方法的能耗较 IFF, NeMC 和 VPANS 分别降低 14.93%, 6.58%. 和 9.64%. ACDRL-VNFPC 的性能表现是由于神经网络的良好拟合能力和泛化能力.

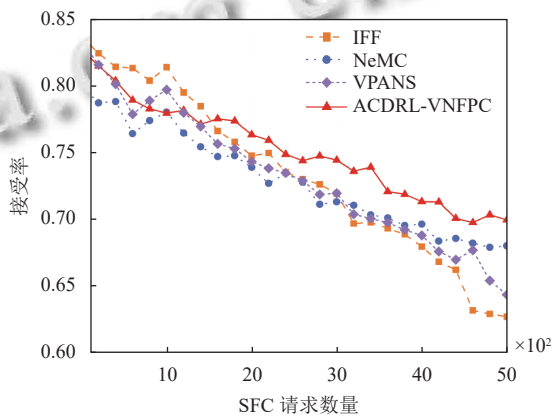


图 4 SFC 请求数量对于服务接受率的影响

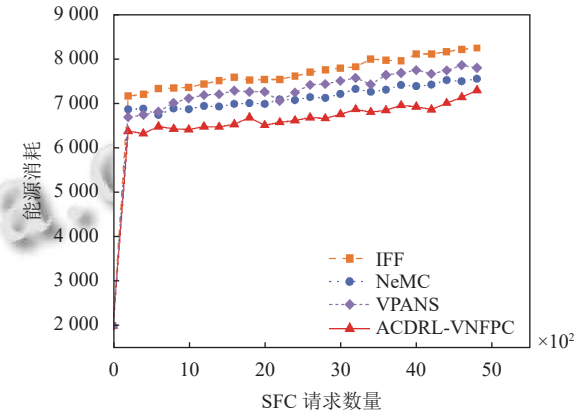


图 5 SFC 请求数量对于能耗的影响

图 6 展示了执行时间随着 SFC 请求数量增多, 4 种算法均呈上升趋势, 其中执行时间指的是获得解决方案的时间, 包括模型的训练时间和部署时间, 这个指标反映了问题规模的扩展能力, 即面对在大量 SFC 请求时, 算法是否可以在合理时间范围内获得解决方案. 从图 6 可以看出, IFF 算法虽然在 SFC 请求数量较少时, 性能表现较好, 然而面对大量的 SFC 请求时, IFF 算法执行时间增速较大, 因此, IFF 整体性能是最差. ACDRL-VNFPC

方法随着 SFC 请求数量的增加, 执行时间仍是以较低时间水平获得解决方案. 提出的算法在执行时间方面较 IFF, NeMC 和 VPANS 分别提升了 16.16%, 2.42% 和 8.73%. 这说明该算法面对大量网络请求的求解时间上是可接受的.

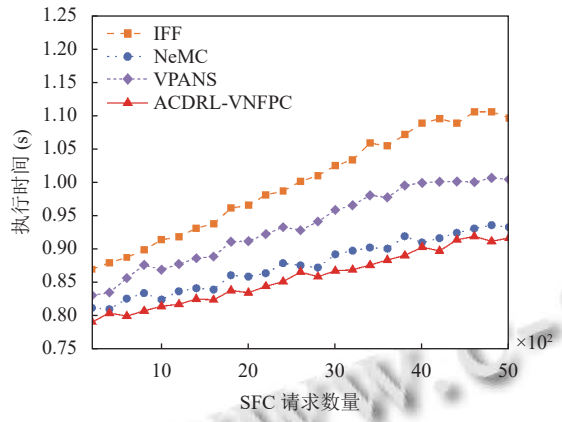


图 6 SFC 请求数量对于执行时间的影响

4 结束语

本研究是基于深度强化学习方法在 NFV 网络中 VNF 放置和链接问题中的应用. 为了满足服务质量前提下, 不降低服务接受率的同时可以权衡优化能耗. 为此, 不仅提取了物理网络信息, 而且考虑了 SFC 中的 VNF 特性, 并提出了基于 Actor-Critic 的深度强化学习的 VNFPC (ACDRL-VNFPC) 的算法. 算法应用了适应性共享方案, 不仅多服务之间共享相同 VNF, 而且多个 VNF 共享相同的服务器, 使得算法减少资源消耗同时降低能耗. 实验结果表明, ACDRL-VNFPC 方法在接受率, 能耗和执行时间方面都有很好的性能. 未来研究方向致力于复杂的云边场景下在线放置和流量路由的解决方案.

参考文献

1 王进文, 张晓丽, 李琦, 等. 网络功能虚拟化技术研究进展. 计算机学报, 2019, 42(2): 415–436. [doi: 10.11897/SP.J.1016.2019.00415]

2 Kaur K, Mangat V, Kumar K. A review on virtualized infrastructure managers with management and orchestration features in NFV architecture. Computer Networks, 2022, 217: 109281. [doi: 10.1016/j.comnet.2022.109281]

3 Kadam SS, Ingle DR. Literature review on redistribution of routing protocols in wireless networks using SDN along with NFV. Proceedings of ICSCS 2021. Springer, 2022. 553–575.

4 Barakabitze AA, Ahmad A, Mijumbi R, *et al*. 5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges. Computer Networks, 2020, 167: 106984.

5 Sun J, Zhang Y, Liu F, *et al*. A survey on the placement of virtual network functions. Journal of Network and Computer Applications, 2022, 202: 103361. [doi: 10.1016/j.jnca.2022.103361]

6 Wang SY, Cao HT, Yang LX. A survey of service function chains orchestration in data center networks. Proceedings of the 2020 IEEE Globecom Workshops. Taipei: IEEE, 2020. 1–6.

7 Adoga HU, Pezaros DP. Network function virtualization and service function chaining frameworks: A comprehensive review of requirements, objectives, implementations, and open research challenges. Future Internet, 2022, 14(2): 59. [doi: 10.3390/fi14020059]

8 Houidi O, Soualah O, Houidi I, *et al*. Energy efficient VNF-FG embedding via attention-based deep reinforcement learning. Proceedings of the 19th International Conference on Network and Service Management (CNSM). Niagara Falls: IEEE, 2023. 1–7.

9 Varasteh A, Madiwalar B, van Bemten A, *et al*. Holu: Power-aware and delay-constrained VNF placement and chaining. IEEE Transactions on Network and Service Management, 2021, 18(2): 1524–1539. [doi: 10.1109/TNSM.2021.3055693]

10 Mohamed R, Leivadeas A, Lambadaris I, *et al*. Online and scalable virtual network functions chain placement for emerging 5G networks. Proceedings of the 2022 IEEE International Mediterranean Conference on Communications and Networking (MeditCom). Athens: IEEE, 2022. 255–260.

11 Yaghoubpour F, Bakhshi B, Seifi F. End-to-end delay guaranteed Service Function Chain deployment: A multi-level mapping approach. Computer Communications, 2022, 194: 433–445. [doi: 10.1016/j.comcom.2022.08.005]

12 Ikhelef A, Saidi MY, Li SP, *et al*. A knapsack-based optimization algorithm for VNF placement and chaining problem. Proceedings of the 47th IEEE Conference on Local Computer Networks (LCN). Edmonton: IEEE, 2022. 430–437.

13 Luo JZ, Li J, Jiao L, *et al*. On the effective parallelization and near-optimal deployment of service function chains. IEEE Transactions on Parallel and Distributed Systems, 2021, 32(5): 1238–1255. [doi: 10.1109/TPDS.2020.3043768]

14 孙春霞, 杨丽, 王小鹏, 等. 结合深度强化学习的边缘计算网络服务功能链时延优化部署方法. 电子与信息学报, 2024, 46(04): 1363–1372.

15 Liu HT, Ding SD, Wang SY, *et al*. Multi-objective optimization service function chain placement algorithm based on reinforcement learning. Journal of Network and Systems Management, 2022, 30(4): 58. [doi: 10.1007/s10922-022-09673-5]

16 Christianos F, Papoudakis G, Albrecht SV. Pareto actor-critic for equilibrium selection in multi-agent reinforcement learning. Transactions on Machine Learning Research. https://openreview.net/forum?id=3AzqYa18ah. (2024-10-24)

17 高媛, 方海, 赵扬, 等. 基于自然梯度 Actor-Critic 强化学习的卫星边缘网络服务功能链部署方法. 电子与信息学报, 2023, 45(2): 455–463. [doi: 10.11999/JEIT211384]

18 Pham TM, Nguyen TM, Nguyen XTT, *et al*. Fast optimal resource allocation for resilient service coordination in an NFV-enabled Internet-of-Things system. Proceedings of the 2022 International Conference on Advanced Technologies for Communications (ATC). Ha Noi: IEEE, 2022. 141–146.

19 Nguyen DHP, Lien YH, Liu BH, *et al*. Virtual network function placement for serving weighted services in NFV-enabled networks. IEEE Systems Journal, 2023, 17(4): 5648–5659. [doi: 10.1109/JSYST.2023.3257776]

20 Wang ZH, Zhuang L, Zhou FJ, *et al*. Energy efficient VNF placement algorithm using reinforcement learning in NFV-enabled network. Proceedings of the 2023 IEEE International Conference on Control, Electronics and Computer Technology (ICCECT). Jilin: IEEE, 2023. 625–629.

21 Chen MH, Sun Y, Hu HL, *et al*. Energy-saving and resource-efficient algorithm for virtual network function placement with network scaling. IEEE Transactions on Green Communications and Networking, 2021, 5(1): 29–40. [doi: 10.1109/TGCN.2020.3042675]

(校对责编: 张重毅)