指导教师评定成绩 (五级制):
指导教师签字:

使用深度强化学习优化虚拟网络功能放置

摘要

网络功能虚拟化（NFV）引入了一个新的网络架构框架，将传统上部署在专用设备上的网络功能演变为运行在通用硬件上的软件实现。部署 NFV 的主要挑战之一是在 NFV 基础设施中优化 NFV 基础设施的最佳资源配置。虚拟网络功能放置和网络嵌入可以被表述为一个数学优化问题，它涉及到一组表达网络基础设施和服务合同限制的可行性约束。据报道，这个问题是 NP-hard，因此该领域开展的大多数优化工作都集中在设计启发式和元启发式算法上。然而，在高度受限的问题中，如本案例中，推断出一个有竞争力的启发式算法可能是一项艰巨的任务，需要专业知识。因此，一个有趣的解决方案是使用强化学习来模拟一个优化策略。这里提出的工作通过考虑问题定义中的约束条件扩展了神经组合优化理论。由此产生的 Agent 能够通过探索 NFV 基础设施来学习安置决策，目的是使整体功耗最小化。所进行的实验表明，当所提出的策略也与启发式算法相结合时，使用相对简单的算法就能取得极具竞争力的结果。

1 介绍

目前基于特定用途硬件的网络部署有很多缺点：它们的资本和运营支出很高，不允许以简单的方式更新其功能，而且生命周期很短。因此，网络功能虚拟化（NFV）在业界兴起，由于目前虚拟化技术的进步，有望改变网络运营商设计、管理和部署其网络基础设施的方式。为了使这项技术标准化，2012 年，领先的电信网络运营商在欧洲电信标准协会（ETSI）创建了一个规范小组。由此产生的 NFV 架构[1]用处理特定虚拟网络功能（简称 VNF）的软件模块取代了传统的特定服务硬件。这种架构为每个功能提供了模块化和隔离性，因此它们可以在一个通用的虚拟环境中独立运行。

在 NFV 框架中，一个虚拟化的网络服务（NS）在创建时会结合各个 VNF。在基础设施上部署的有序的 VNF 集合的组成被称为前向图。为了在 NFV 环境中全面定义网络服务，有必要指定：构成网络服务的 VNF 组件、它们在图中的各自顺序以及它们在 NFV 基础设施中的位置。这项技术面临的主要挑战之一是在底层网络基础设施中优化资源放置。这一挑战被称为 VNF 前向图嵌入问题（简称 VNF-FGE），是 NFV 资源分配问题之一，其余的资源分配问题还由 VNF 链式组合和 VNF 调度问题。

在本文中，我们专注于优化 VNF-FGE。这个问题包括在物理网络基础设施的基础上有效地映射一组网络服务请求。特别是，考虑到虚拟环境的状态，我们寻求获得 NS 链的最佳位置，从而实现特定的资源目标（例如，剩余资源的最大化，整体功耗的最小化，特定 QoS 指标的优化等）。此外，NFV 的具体方面，如转发延迟、入口/出口比特率和流量链，也必须加以考虑。

我们将 VNF-FGE 形式化为一个受限的组合优化问题，其中 NS 需要被放置在满足服务水平协议的网络基础设施之上。这个问题被表示为 NP-hard[2]；因此，只有在小的实例中才有可能精确地解决它。文献中的大多数现有方法都集中在启发式或元启发式算法的设计上。然而，这些策略存在一个主要的缺点。尽管它们在解决有限制的优化问题方面有很好的效果，但当限制的数量较多，解决方案的可行区域较小时，在某些情况下，它们往往是无效的。

2 背景

2.1 关于 VNF-FGE 问题的文献

以前的一些工作已经着手解决 VNF-FGE 问题。在文献中，我们可以发现它存在于[9]或[10]中。由于它是一个 NP-hard 的优化问题，对于大的实例来说，优化解决它的运行时间是难以承受的。出于这个原因，这个问题已经通过应用以下的

替代方法来解决了。

i) 对于小的问题实例，可以实现最佳的解决方案。例如，使用混合整数线性编程，或者对描述为马尔科夫决策过程的问题分析求解贝尔曼方程。

ii) 另一个选择是启发式方法，尽管不能保证收敛到局部最优，但能够在合理的计算时间内获得有竞争力的解决方案。

iii) 最后，作为启发式方法的延伸，元启发式方法提供了一个与问题无关的高级算法框架，为开发优化算法设定了准则。这些方法通过迭代改进中间的解决方案来找到接近最优的解决方案，并考虑到给定的质量衡量标准。

基于前面的备选方案，在优化 VFN-FGE 时，已经解决了不同的目标。最小化映射在基础设施之上的虚拟网络功能实例的数量[17]，或最大化成功嵌入的服务请求的数量[14]是这些目标中的几个。其他工作涉及到基于功率的安置建议。例如，通过遗传算法方法使功耗最小化[18]，或对未知或不精确制定的资源需求变化提供稳健性[2]，是这些工作中的一部分。

除了前面的参考文献，强化学习也成为一个有前景的研究方向。例如，Mijumbi 等人在 2014 年发表的文章[19]中使用 Q-learning 来控制作为 NFV 管理系统一部分的资源分配。在[20]中，Mijumbi 还采用了一个人工神经网络，在其之前的工作基础上进行资源重新分配决策。2018 年 Yao 等人[21]首次实现了历史网络请求数据和基于策略的 RL 来优化节点映射。他们使用节点和链接的嵌入表示，在每个时间步骤中，都会根据网络属性的变化特征进行更新。这个网络嵌入通过卷积神经网络来选择使长期收益最大化的底层节点。

2.2 神经组合优化

在组合优化中，必须从一个有限的解决方案中找出一个最小化或最大化给定成本函数的解决方案。当解决方案的搜索空间较小时，这项任务可能很容易。然而，在大的问题实例上这并不简单，因为精确的方法在计算上是不可行的，而启发式和元启发式不能保证找到最优解，也不能保证快速收敛。在 Vinyals 等人在 2015 年发表的文章[22]中，作者证明了使用监督学习来解决组合优化问题是可能的。特别是，他们表明，给定一个实例的输入，一个神经网络可以返回问题的近似最优解。在序列到序列模型的最新进展的激励下，他们开发了一个注意力指针网络来处理传统的组合问题。然而，在大多数情况下，这种策略并不适用，因为很难计算出一套最佳的标签（解决方案）来训练模型。沿着同样的思路，Bello 等人 [3]介绍了使用强化学习来解决组合问题，创造了神经组合优化这一术语。

NCO 依赖于基于策略的方法，直接学习一个策略函数，将问题的实例（状态）映射到解决方案（行动）。在 RL 方法下解决组合问题时，行动空间对应于搜索

空间，具有很大的维度。在这种情况下，基于策略的方法在估计模型参数时被证明是有效的。使用 RL，不需要最优标签来训练模型，相反，Agent（RL 中对模型的称呼）从评估环境（在这种情况下，问题描述）上的解决方案中获得的奖励被用来优化其策略。特别是，政策梯度[23]方法被用来执行随机梯度下降，以更好地估计 NN 的权重。一旦 Agent 在学习过程中收敛，给定一个问题的实例给模型，它就会返回一个解决方案。这个解决方案不能被证明是最优的，但在这个意义上是 "接近 "它的，因为它遵循的政策，平均来说，已经获得了最好的结果。这里介绍的工作与 Mirhoseini 等人 [5]有关，他们也是在 Bello 等人[3]的原始论文启发下执行了一个安置模型。在那篇论文中，作者实施了主动搜索，以推断出先前给定的问题实例的近似最佳位置。我们的工作不同之处在于，我们在问题的定义中处理了约束条件。

3 问题的形式化

在以下段落中，将广泛描述简化后的 VNF-FGE 问题的数学形式化，以及所需的符号。

让我们考虑一个网络服务列表，该列表必须以最佳方式放置在一组主机服务器 h∈H 中，并且每个主机服务器都依赖于有限的可用资源 r∈R，即计算、存储和连接的能力。如前文所述，主机服务器通过星形拓扑结构，使用各自的链路连接 i∈L（见图 1）相互连接。链接的属性如带宽或传播延迟也被考虑在内。这个问题的最终目的是发现给定服务链的最佳位置，使基础设施的总耗电量最小。这个解决方案受制于遵守与虚拟资源和链路容量的可用性相关的限制，以及每个服务施加的延迟阈值。

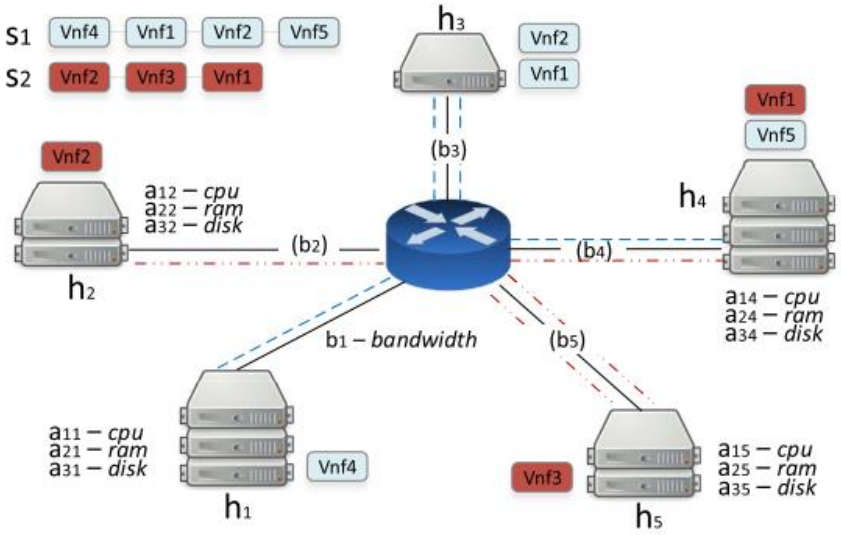

图 1：虚拟化环境中的网络服务分配实例。它描述了一个环境，其中服务链表示为 s1 和 s2，需要以最佳方式放置在总共五个主机中（从h_1到h_5）。每台主机都有自己的a_h计算能力、a_{2h}内存和a_{3h}磁盘能力。此外，每个主机通过专用链路 i 连接到一个共同的交换机，该链路有一个相关的容量约束b_i。

为了表述这个问题，我们采用 Morotta 等人在 2017 年发表的文章中[2]提出的命名法。让我们把{ h_1，h_2,..., h_n}表示为主机服务器 H 的集合，让 V 是资源库中可用的 VNF 的集合。因此，一个网络服务由 m∈{1,...,M}的虚拟网络功能数组组成，它们构成了一个服务链 s=（f_1, f_2,..., f_m），其中 f∈V。所有服务链的组合空间被表示为 S。

问题包括寻找最佳的放置集，表示为x ∈ {0，1}$^{m\times n}$，其中x_{fh}代表一个布尔状态变量，描述功能 f∈V 是否被放置在主机$h \in H$中（在正的情况下为 1，反之为 0）。然后，需要找到问题解决方案的搜索空间是：

$$\Omega=\{x \in \{0，1\}^{m\times n} \text{ s.t. } \sum_h x_{fh} = 1 \; \forall f \in s \; \}$$

Ω中的限制条件规定，一个虚拟网络功能一次只能放在一个主机中。

表 1 问题形式化变量

H	主机的集合
L	链接的集合
V	一组 VNFs
S	一组 NS 链
R	资源的集合
P	一组安置点
a_{rh}	主机 h 中的可用资源量 r
r_{rv}	VNF v 所请求的资源量 r
W_h^{min}	主机 h 的空闲功耗
W_h^{cpu}	主机 h 中每个 cpu 的功耗
W_{net}	链接上每个带宽单位的功耗
b_i	链路 i 的带宽
l_v	由于 VNF v 的计算时间而产生的延迟
l_i^s	由服务链 s 产生的链路 i 上的延时
b_v^s	服务链 s 中的 VNF v 所要求的带宽
l^s	服务链 s 上允许的最大延时

x_{fh}	主机 h 中功能 f 的二进制放置变量
y_h	主机 h 的二进制激活变量
g_i	链接 i 的二进制激活变量

在提出问题的成本和限制函数之前，表 I 中列出了问题的决策变量和参数的摘要。如前所述，要优化的变量集是那些定义放置 x 的变量。为了支持问题的描述，提出了辅助变量。服务器激活变量$y_h \in \{0, 1\}$就是这种情况，如果服务器正在执行任何 VNF，则表示 1，否则关闭电源；链路激活变量$g_i \in \{0, 1\}$，如果链路 i 正在承载流量，则等于 1，否则等于 0。

在功耗方面，主机服务器的特点是线性功耗曲线，与计算利用率成比例增长。每个被激活的服务器（$y_i = 1$）消耗一个最小的功率W_h^{min}，其功率随着分配给该服务器的 VNFs 所需的 CPU 的总和而增加。每个使用中的 CPU 消耗W_h^{cpu}瓦特。关于链接，它们也有一个相关的能源成本。它的计算方法是将每个带宽利用率的成本（表示为W_{net}）乘以每个链路利用的带宽。每个服务器 h 拥有的可用资源 r $\in$R，表示为a_{rh}。VNF v 需要的资源量 r 表示为r_{rv}。作为服务 s$\in$S 的一部分，VNF v 的数据传输所要求的带宽表示为b_v^s。以同样的方式，l_i^s表示由于服务 s 导致的链路 i 的延迟，而l_v表示由于 VNF v 的计算时间导致的延迟。链路 i 中允许的最大带宽表示为b_i。最后，我们用l_s表示每个服务链 s 允许的最大延时。

表 2 需要优化的问题的形式化方程

$$\arg\min_{\boldsymbol{x}\in\Omega}\left(\sum_{h\in H}\left[W_h^{cpu}\cdot\sum_{f\in s}r_{rv}\cdot x_{fh}+W_h^{min}\cdot y_h\right]+\right.$$

$$\left.+\sum_{i\in L}W_{net}\cdot\sum_{f}b_v^s\cdot x_{fh}\right) \quad (1)$$

subject to:

$$\sum_{f\in s}r_{rv}\cdot x_{fh}\le y_h\cdot a_{rh} \qquad \forall\, h\in H, r\in R \quad (2)$$

$$\sum_{f\in s}b_v^s\cdot x_{fh}\le g_i\cdot b_i \qquad \forall\, i\in L \quad (3)$$

$$\sum_{h\in H}\sum_{f\in s}l_v\cdot x_{fh}+\sum_{i\in L}\sum_{f\in s}l_i^s\cdot x_{fh}\le l^s \qquad \forall\, h\in H, i\in L \quad (4)$$

要优化的成本函数列于表二。它代表能耗，并被计算为与激活的服务器有关的功耗和活动链路的总成本。公式（2）中的约束条件决定了服务器中使用的总资源不能超过活动服务器中的可用资源。因此，它在服务器激活变量y_h和分配变量x_{fh}之间设置了一个链接；只有承载某些 VNF 的服务器是活跃的。然后，带宽的容量约束在公式（3）中定义。它以与主机激活变量相同的方式使用链接状态变量 gi。它们将链接的布尔状态与节点激活变量的状态联系起来：如果一个链接被使用，那么其终端节点必须被激活；如果一个节点没有被激活，那么相关的链接也没有被激活。最后，公式（4）中的约束表达了网络服务 s∈S 的延迟要求，规定图中使用的链接的延迟和计算时间导致的延迟的总和必须满足服务的延迟限制l_s。

为了说明这个问题，图 1 中描述了一个例子。在这里，引入了安置的等效表示：$p_s = (p_1, p_2, \ldots, p_m)$其中$p_i$∈H。这个符号对于简化下文中描述的 RL 方程的表述特别有用。继续说，每台服务器通过一个专用链路i连接到一个共同的交换机。目标是放置服务链（表示为s_1和s_2），使整体成本函数最小。在这个例子中，为每个服务计算的放置向量是$p_{s_1} = (h_1, h_3, h_3, h_4)$和 $p_{s_2} = (h_2, h_5, h_4)$。

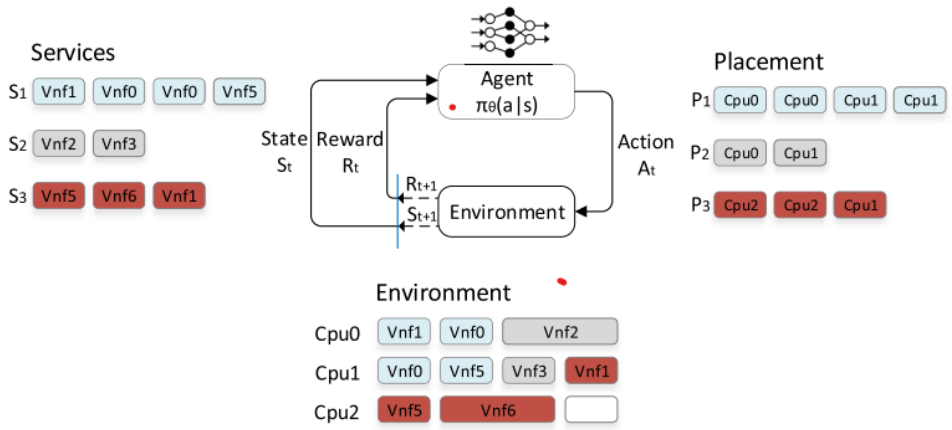

图 2 强化学习行动-奖励反馈回路

4 策略优化方法

在这项工作中，我们使用神经组合优化范式来处理一个简化的 VNF-FGE 问题。为此，我们使用一个神经网络模型来推断安置策略。我们提出的神经结构是一个基于编码器-解码器的序列-序列模型，这种结构在序列预测方面已经取得了突出的成果。其工作原理如下：Agent 接收一个大小可变的m的网络服务 $s = (f_1, f_2, \ldots, f_m)$作为输入，并输出一个安置向量$p_s = (p_1, p_2, \ldots, p_m)$表示对每个

VNF 的分配。底层神经网络用其权重θ表示，推导出一个策略$\pi_\theta(p_s|s)$，该策略概括了所有可能的服务链组合的放置策略，这些组合可以通过 VNF 字典生成。

所提出的神经架构的输出空间限于一个序列，我们用它来预测一个服务要占用的节点。将网络减少到一个星形连接是一种简化，使我们能够利用这种众所周知的架构，它在神经机器翻译等任务中取得了杰出的成果。

在图 2 中，我们描述了针对我们的问题的一般强化学习的行动-奖励反馈回路。在这种情况下，与环境的互动被限制在一个单一的步骤。这意味着，对于一个给定的网络服务请求（即图中红色描述的s_3），将创建一个状态向量输入给 Agent，嵌入环境的状态（以前分配的服务s_1和s_2）和请求的服务。Agent 生成相应的放置向量（行动），表明新服务应该放在哪里。然后，环境评估安置决定，并使用公式（1）计算出一个反馈信号，表明解决方案的质量（奖励）。请注意，对于 Agent 来说，表二中的方程式所描述的环境是一个黑盒子。通过与之互动，Agent 将发现一个解决下限约束优化问题的策略，甚至不需要知道其定义。

如上所述，神经组合优化不能被直接应用，因为它不考虑其机制中的约束。处理约束条件不满意的问题是至关重要的，否则，成本函数不能提供足够的信息来推断出一个有竞争力的政策。为此，我们在 Agent 和环境之间引入了一个新的强化学习接口。这个接口使我们能够在 Policy Gradient 方法的成本函数中使用约束放松技术。如果没有这个贡献，Agent 几乎不可能改善其在受限环境中的行为，因为它不会经历足够的积极奖励。奖励稀少的问题在强化学习中是众所周知的。在本文中，我们扩展了神经组合优化理论，以承担约束优化问题中的这个问题。为了这个目的，奖励信号表明基础设施与额外的反馈信号相辅相成，表明对约束的不满意程度。我们将在下面的章节中看到，我们使用拉格朗日松弛技术将这些约束纳入成本函数中。

4.1 带 Policy Gradients 的约束性优化

如前所述，我们利用 Policy Gradients 来学习随机政策$\pi_\theta(p_s|s)$的参数，给定网络服务$s \in S$作为输入，将高概率分配给成本较低的安置$p_s \in P$，将低概率分配给成本较高的安置。我们的神经网络使用链式规则来分解这个输出概率（见公式 1）。因此，在序列中分配一个 VNF 的可能性取决于先前分配的请求链的 VNF p(<f)和状态向量。为简单起见，我们将认为状态向量完全由网络服务 s 定义。如果考虑到环境的多种状态，那么输入 Agent 的状态将是服务与环境状态的嵌入。

$$\pi_\theta(p|s) = \prod_{f=1}^{m} \pi_\theta(p_f \,|\, p_{(<f)}, s) \qquad (1)$$

为了估计模型的参数，我们在策略梯度方法中定义了一个目标函数，代表每

个权重向量θ所获得的奖励。它决定了所实现的政策的质量，因此是一个需要优化的函数。这个目标函数是为每一个可能的策略所定义的，其形状取决于环境和设计的 NN 模型。因此，该问题不依赖于直接优化公式(1)，而是优化这个决定了 Agent 的策略的新的目标函数。为了这个目的，让我们从定义预期的能源消耗 E 与输入的网络服务 S 以及它们的放置位置开始：

$$J_E^{\pi}(\theta|s) = \mathop{\mathbb{E}}_{p\sim\pi_\theta(\cdot|s)}[E(p)] \tag{2}$$

Agent 需要推断出一个政策，从所有可能的服务组合中放置服务。因此，预期成本被定义为服务分布的期望值。

$$J_E^{\pi}(\theta) = \mathop{\mathbb{E}}_{s\sim\mathcal{S}}[J_E^{\pi}(\theta|s)] \tag{3}$$

此外，还有一个与政策相关的约束不满意的期望值与政策相关联，可以表示如下。

$$J_C^{\pi}(\theta) = \mathop{\mathbb{E}}_{s\sim\mathcal{S}}[J_C^{\pi}(\theta|s)] \tag{4}$$

最基本的问题是，在满足约束条件的前提下，找到使预期能耗最小的策略。

$$\min_{\pi\sim\Pi} J_E^{\pi}(\theta) \quad \text{s.t.} \quad J_{C_i}^{\pi} \leq 0 \tag{5}$$

我们为每一个环境返回的约束不满意信号定义一个函数J_C^{π}。在我们的案例中，有三种信号，并且它们代表了按占用率、带宽和延迟分组的累积约束不满意程度。使用拉格朗日松弛技术，公式（5）中表示的问题被转换成一个无约束的问题，其中不可行的解决方案被惩罚。

$$\begin{aligned} g(\lambda) = \min_{\theta} J_L^{\pi}(\lambda,\theta) &= \min_{\theta}[J_E^{\pi}(\theta) + \sum_i \lambda_i \cdot J_{C_i}^{\pi}(\theta)] \\ &= \min_{\theta}[J_E^{\pi}(\theta) + J_{\xi}^{\pi}(\theta)] \end{aligned} \tag{6}$$

即$J_L^{\pi}(\lambda,\theta)$为拉格朗日目标函数，$g(\lambda)$为拉格朗日对偶函数，$\lambda_i$为拉格朗日乘数。换句话说，就是惩罚系数。在这个方程中，我们引入了一个新项$J_{\xi}^{\pi}(\theta)$，我们将其称为 "预期惩罚"，它只是约束性不满意信号的加权总和

对偶函数是凸的，尽管原始函数和约束条件是非凸的，它给出了原始问题的最优值的下限 [24]。然后，目标是找到能产生最佳下限的拉格朗日系数。这就是所谓的拉格朗日对偶问题。

$$\max_{\lambda} g(\lambda) = \max_{\lambda}\min_{\theta} J_L^{\pi}(\lambda,\theta) \tag{7}$$

在这种情况下，我们对惩罚系数进行手动选择。关于如何获得这些系数的直

观描述见附录 C。得到的拉格朗日函数$J_L^{\pi}(\theta)$是定义推断政策质量的目标函数。

为了计算优化该目标函数的权重 θ，我们采用了蒙特卡洛策略梯度方法和随机梯度下降法。

$$\theta_{k+1} = \theta_k + \alpha \cdot \nabla_\theta J_L^{\pi}(\theta) \tag{8}$$

拉格朗日的梯度是用对数似然法推导出来的。这个推导过程与推导预期报酬没有任何区别，方法在[25]中介绍。

$$\begin{aligned} \nabla_\theta J_L^{\pi}(\theta) &= \mathop{\mathbb{E}}_{p\sim\pi_\theta(\cdot|s)}[(L(p|s)\cdot\nabla_\theta \log \pi_\theta(p|s)] \\ \text{where} \quad L(p|s) &= E(p|s) + \xi(p|s) \\ &= E(p|s) + \sum_i \lambda_i \cdot C_i(p|s) \end{aligned} \tag{9}$$

我们将每次迭代中得到的惩罚性能量成本表示为$L(p|s)$，其计算方法是在能量信号$E(p|s)$中加上所有受约束的不满意信号$C(p|s)$的加权和。

然后用蒙特卡洛抽样对梯度进行近似计算，在其中抽出B个服务样本$s_1, s_2, \ldots, s_B \sim S$。为了减少梯度的方差，并因此加快收敛速度，我们添加一个基线估计值[23]，在公式中用术语$b(s)$表示。

$$\nabla_\theta J_L^{\pi}(\theta) \approx \frac{1}{B}\sum_{j=1}^{B}(L(p_j|s_j) - b_{\theta_\nu}(s_j))\cdot\nabla_\theta \log \pi_\theta(p_j|s_j) \tag{10}$$

在这种情况下使用的基线是依赖于状态的，并由一个辅助序列网络执行，该网络嵌入了状态信息，并预测 Agent 在当前政策$\pi_\theta(p|s)$之后获得的惩罚成本$L(p|s)$。它由权重b_θ参数化，并通过随机梯度下降对其预测$b_\theta V(s)$和从环境中获得的实际惩罚成本之间的均方误差目标进行训练。

$$\mathcal{L}(\theta_\nu) = \frac{1}{B}\sum_{j=1}^{B}\|b_{\theta_\nu}(s_j) - L(p_j|s_j)\|^2 \tag{11}$$

附录 D 描述了带有基线估计器的单时间步 Monte-Carlo Policy Gradients 的算法实现。

4.2 架构细节

在我们提出的 NCO 方法中，Agent 是一个序列到序列的模型[26]。图 3 显示了整体 Agent 架构，它是由一个基于堆叠的长短期记忆（LSTM）单元的编码器-解码器设计形成的。该模型的输入是构成要放置的网络服务的 VNF 序列$s = (f_1, f_2, \ldots, f_m)$。如前所述，这些 NS 链有一个可变的长度$m \in \{1, \ldots, \mathrm{M}\}$。这是使用序列模型的主要原因，因为它是为输入不同大小的链而设计的，无需修改其内

部结构。

解码器是一个注意型 LSTM 模型[27]，它的解码步骤与输入序列的数量相同。在每一步，解码器都会输出主机，以放置同一步在编码器中引入的组件。解码器网络隐藏状态$\rho_t = f(\rho_{t-1}, \bar{\rho}_{t-1}, c_t)$是其自身先前状态与对编码器隐藏状态的注意力机制相结合的函数。上下文向量c_t由输入序列的隐藏状态的总和组成，由对齐分数加权。解码器输出步骤t的上下文向量计算如下：

$$c_t = \sum_f \alpha_{t,f} \bar{\rho}_f \tag{12}$$

集合a_t, f 是定义解码过程中每个源隐藏状态的权重的变量。可变大小的对齐向量具有与源序列相同的步骤数。它是通过对解码器的当前目标隐藏状态ρ_t与每个源隐藏状态$\bar{\rho}_f$的评分来计算的。

$$\alpha_{t,f} = \text{softmax}(\text{score}(\rho_t, \bar{\rho}_f)) \tag{13}$$

分数函数定义为以下形式：

$$\text{score}(\rho_t, \bar{\rho}_f) = v_a^{\intercal} \tanh(W_1 \rho_t + W_2 \bar{\rho}_f) \tag{14}$$

v_a, W_1, W_2是对准模型中要学习的权重矩阵。

最后，baseline 中的$b_\theta V(s)$由一个辅助网络计算：一个连接到多层感知器（MLP）输出层的 LSTM 编码器，Agent 预测按照当前政策产生的惩罚性能源成本。因此，它是一个基于环境状态的数值近似器。

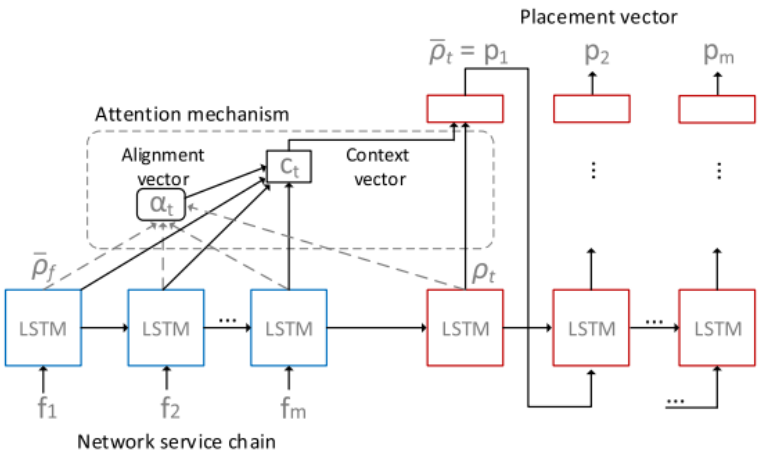

图 3：序列到序列模型，由一个带有 Bahdanau 注意机制的编码器-解码器结构组成。它将一个可变长度的服务链编码在一个表征向量中，解码器网络用它来产生安置决策

4.3 搜索策略

策略梯度有一个主要的缺点，它存在局部收敛的问题。在学习过程中，神经

元的权重是按照目标函数的梯度方向调整的。由于这个函数是非凸的，这种方法很容易收敛到次优的最小值。一旦 Agent 收敛到一个，无论我们延长训练多长时间，它都会保留所取得的政策。为了改善获得的策略，有必要收敛到成本函数中更好的最优值。在这个意义上，在训练过程中应用了熵正则化等技术来增加探索。我们通过在推理中应用 Bello 等人[3]提出的一些搜索方法，获得了明显的改进。由于评估安置的费用不高，Agent 可以在推理时通过考虑多个候选方案并选择最佳方案来模拟一个搜索过程。

我们考虑了两种搜索策略：一种是对多个训练好的模型进行贪婪推理，另一种是采样技术。在第一种策略中，学习了多个模型，在推理时对每个模型的贪婪输出进行评估，以选择最佳模型。另一种方法是利用温度超参数T来控制输出分布的稀疏性，从而改善推理时的探索。取出多个样本，并选择其中最好的一个作为输出。使用这种方法是为了防止模型过于自信，允许在推理中评估近似的政策。

5 实验研究

在本节中，为了评估所提出的优化 VNF-FGE 问题的方法，在两个不同规模的环境中进行了详细的实验研究：一个小型和一个大型基础设施。利用这个测试平台，我们评估了神经网络模型与 Gecode 求解器和 FF 启发式算法的性能[28]。这些环境的选择分别有利于求解器和启发式算法，在下文会进行论证。

除了直接使用神经网络的输出外，我们还利用这些信息来指导启发式算法。由此产生的混合 Agent 利用神经网络的结果，向启发式指出基础设施中的节点必须被占领的顺序。

表三是对环境的简要描述，以及主要参数 3 和结果的总结。关于环境或参数配置的进一步信息，请参考附录 A 和附录 B。

5.1 学习过程

首先，我们进行了实验，以研究在考虑不同占用率的情况下，裸序列到序列模型的学习过程。在图 4 中，显示了 Agent 在一个有足够空间的问题（a）和有大量占用率的小实例（b）上的学习历史。对于每个迭代，介绍了该批次计算的预期能量J_E^π、基线b、惩罚J_ξ^π和拉格朗日J_L^π函数的近似值。

在学习的开始阶段，Agent 产生的随机输出序列违反了许多约束条件，这些行为会受到很高的惩罚。因此，在开始时，Agent 只关注约束条件的满足，而忽略了潜在的功耗。随着学习的进展，Agent 通过随机梯度下降修正其权重，以最小化拉格朗日目标函数（公式 6）。迭代地重复这个过程，Agent 改进其政策，换句话说，减少不满意的约束。这个过程一直持续到达到局部最小值或鞍点为止。

在学习结束时，惩罚信号几乎取消了。这意味着 Agent 在这个小场景中能够推断出的政策是所期望的。获得不符合限制条件的安置的期望值很低，如果这种情况发生，限制条件就会稍微不满意。然而，在严格限制的环境中，图 4(b)，约束条件不满足的概率明显增加。可以看出，收到的平均惩罚与能源成本相当。因此，Agent 专注于满足约束条件，以使获得的惩罚最小。

5.2 评估搜索策略

正如 IV-C 节所指出的，为了改善学习过程中获得的结果，我们在推理时进行了以下搜索方法：对多个 Agent 的贪婪推理和采样方法。产生最佳结果的方法是对多个训练好的 Agent 进行贪婪推理。在这里，每个单独训练的 Agent 收敛到自己的政策，导致不同的安置策略有相同的输入序列。从多个 Agent 的决定中选择最佳解决方案，导致了重大改进。

这项技术已经与抽样方法结合使用，在我们的案例中，抽样方法的效率不如[3]。应用这些技术后得到的结果反映在表三中。

5.3 与其他算法的比较

为了验证所提出的 NCO 方法，我们将模型实时产生的结果与约束解算器 Gecode4 和 FF 启发式的输出进行比较。这个实验是在限制解算器的时间为 5 分钟的情况下进行的。根据[8]中的指导原则，Gecode 求解器对每个实例的求解都是基于经典的 Branch & Bound 范式的算法。VNF-FGE（小的和大的），我们从可行方案的数量和与求解器相比的最优性方面来评估结果。每个问题实例的细节以及每个案例中使用的模型参数的摘要可以在附录 B 中看到。

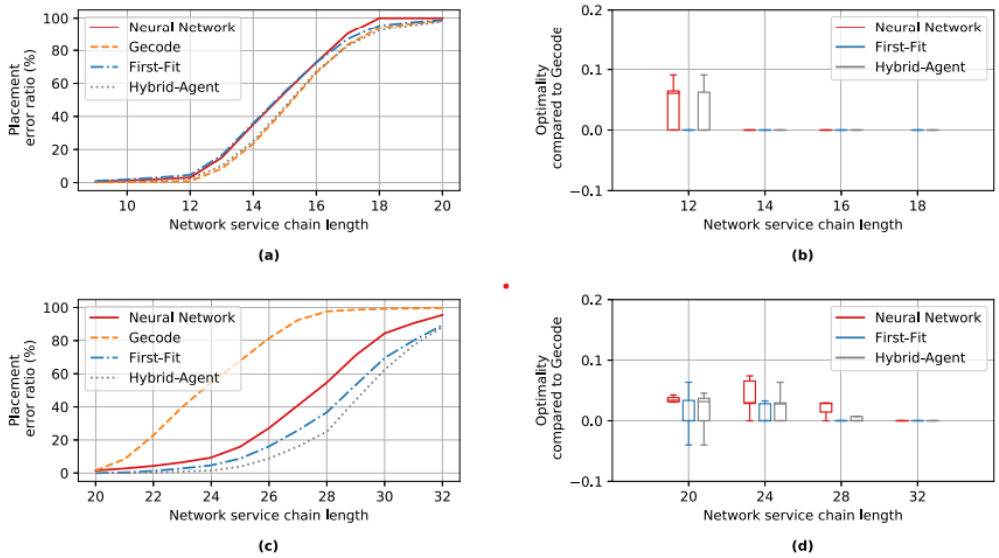

图 5 神经网络模型、Gecode 求解器、First-Fit 启发式和 Hybrid-Agent 在小环境（a）和（b）以及大环境（c）和（d）中的解决方案的有效性比较，包括优化和可行性。在小环境（a）和（b）以及大环境（c）和（d）下的效果

图 5 显示了在小环境（a）和（b）以及大环境（c）和（d）中由神经网络模型、求解器和 FF 启发式得到的指标。这个实验是在以下程序下测量的，从一个空的环境开始，我们找到了一个具有不同长度的单一服务，以收紧占用率，因此加强了定义问题的约束。这个测试平台既不代表正常的操作，即服务是按顺序分配的。

服务是按顺序分配的，其长度也不符合实际。然而，为了比较的目的，我们还是考虑了它。考虑到一个更有限的环境，可以达到同样的效果，但这种方法有助于统一测试的起点。可以注意到，我们增加了链中 VNF 的数量，换句话说，我们限制了有效解决方案的空间，直到所有方法都失败。对于小的问题实例，求解器有更大的概率找到一个可行的解决方案。然而，在大的问题实例中，失败的比例会掉头，启发式是遭受较少的错误陈述。关于最优性差距，对于那些解决方案满足所有约束条件的实例，我们分别比较了求解器、神经网络和启发式算法的解决方案的质量。当环境较小时，求解器获得了较好的结果，但当在足够大的环境中测试时，它又被启发式算法打败了。神经网络本身的表现介于上述两种方法之间。在第二种情况下，它提供了比求解器更好的解决方案（当服务长度在 24 到 26 个元素之间时，实现了多达 50%的可行方案），但在可行方案的数量和优化差距方面，它的竞争力不如启发式。

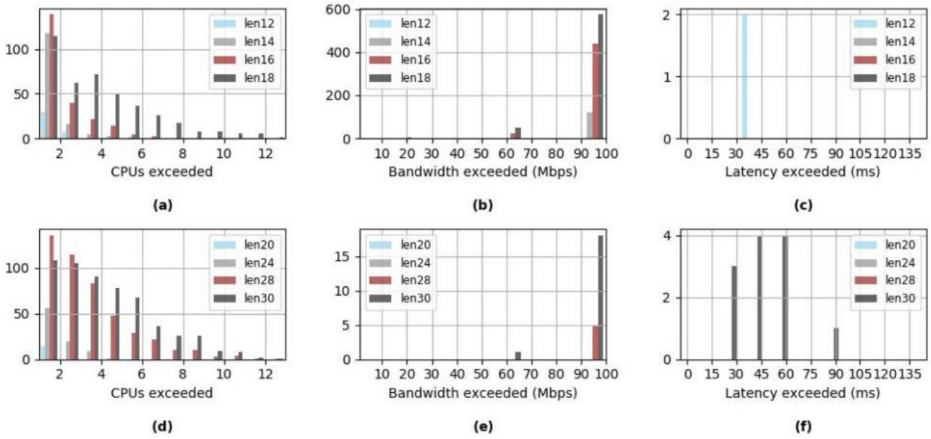

图 6 在小（a）（b）（c）和大（d）（e）（f）环境中，按照不同长度服务功能链的放置，超过的 CPU 数量、带宽（Mbps）和延迟（ms）按顺序排列的直方图

使用神经网络来估计政策是一笔巨大的财富，它总是会实时地产生一个近似于结果的解决方案，同时试图最小化不满意的约束。然而，如果使用解算器等方法失败时，将会没有任何解决方案可用，这种情况也可能在启发式算法中发生。

5.4 混合 Agent 分析

如结果所示，在实际情况下，FF 启发式本身能够产生比神经网络和求解器更好的解决方案。然而，这种算法缺乏定制化，对于每个问题实例，它执行的是确定性的行为。它从拥有更多可用资源的节点开始分配环境。神经网络产生的信息可以作为一种指导，以提高这种启发式算法的性能。由此产生的混合 Agent 已显示出在高度受限的环境中提高了有效解决方案的数量，同时保持了启发式算法的最优性差距。混合 Agent 被设计为如下。它使用神经网络产生的输出序列作为 FF 算法的种子。由神经网络产生的输出指导启发式，指出必须占领的服务器。然而，启发式验证眼前的位置是允许的，否则它就会跳转到占领神经网络建议的下一个服务器。使用这个程序，我们实现了一个 Agent，成为新的参考。如图 5 所示，在小环境中，它的解决方案与求解器相当，而当环境足够大时，它在有效解决方案的数量上优于之前所有的方法。在这种情况下，与启发式方法相比，我们在可行方案的数量上获得了 10%的提升，同时保留了它性能方面的差距。

5.5 不满足限制条件的情况

如前所述，神经网络推断出一个局部最小化目标函数的政策。也就是说，

所实现的策略是，对于所有的服务组合空间S，呈现出能源消耗期望值的局部最小值，包括错位造成的惩罚。可以看出，当基础设施中有足够的可用空间时，神经网络能够概括出满足所有约束条件的安置决策。随着环境占用率的增加，该策略不够成熟，并开始不满足那些认为惩罚最少的约束。这些约束条件不满足的方式是有规律可循的，也就是说，不满足一个变量的可能性很小，而不满足的可能性很大。先验地，我们不可能知道哪些约束条件会被不满足。在图 6 中，描述了按占用率、带宽和延迟分组的不满意约束的直方图。直方图是由一组 640 个样本生成的，在之前描述的场景中。可以看出，根据不同的实例，Agent 容易违反一些约束，而不是其他。例如，尽管占用率通常是最不满意的约束，但在小场景（b）中，当服务有相当长的长度（18 个元素）时，带宽约束的具体情况是不满意的主要原因。不满意的约束和不满意的模式都不能被预测。然而，通过修改分配的拉格朗日乘数λ_i来改变这种模式是可能的。Agent 可以有倾向性，以便先于其他约束满足所需的约束。这种行为，不能得到也不能转移到启发式。正如在这种情况下，不满足约束的情况将被嵌入到算法本身中。

6 结论

在本文的工作中，我们通过强化学习方法解决了 VNF-FGE 问题，以模拟网络功能虚拟化基础设施中的安置政策。为此，我们扩展了神经组合优化理论，以考虑问题中的限制。由此产生的神经网络模型能够学习放置决策，目的是使整体功耗最小化。

所进行的实验表明，当所提出的代理与启发式算法一起使用时，我们得到了一个混合代理，它提高了启发式算法本身的性能。在本文中，人工智能被用来增强一个简单的 First-Fit 启发式算法，在不需要该问题的专业知识的情况下取得有竞争力的结果。

为改善神经网络获得的策略，一个令人兴奋的扩展是将问题呈现为一个完全定义的马尔科夫决策过程。在这种情况下，由于安置决策是一步一步做出的，代理人能够从中间状态中学习，而不是从单一输入中一次性推断出来。这是 Nazari 等人[6]为解决车辆路由问题而采取的策略。在他们的模型中，他们不需要在推理时进行过多的后处理，以获得在运行时间和性能上与特定问题启发式方法相当的解决方案。

7 附录

7.1 环境描述

小规模环境描述：

Host Properties										
No. CPUs	10	9	8	7	6	6	6	6	6	6
Link BW (Mbps)	1000	1000	500	400	300	300	300	300	300	300
Link Latency (ms)	30	50	10	50	50	50	50	50	50	50

大规模环境描述：

Host Properties										
No. CPUs	10	10	9	9	8	8	7	7	6	6
Link Bw (Mbps)	1000	1000	1000	1000	500	500	400	400	300	300
Link Latency (ms)	30	30	50	50	10	10	50	50	50	50
	6	6	6	6	6	6	6	6	6	6
	300	300	300	300	300	300	300	300	300	300
	50	50	50	50	50	50	50	50	50	50

VNF 字典描述：

VNFD Properties								
No. CPUs required	4	3	3	2	2	2	1	1
Bw required (Mbps)	100	80	60	20	20	20	20	20
Processing latency (ms)	100	80	60	20	20	20	20	20

与环境中的电力消耗有关的参数如下：

$$
\begin{array}{ll}
W_h^{min} & 200 \ (watt) \\
W_h^{cpu} & 100 \ (watt) \\
W_{net} & 0.1 \ (watt \ / \ mbps)
\end{array}
$$

下面介绍为惩罚不同约束函数而选择的拉格朗日乘数：

$$
\begin{array}{ll}
\lambda_{occupacy} & 1000 \\
\lambda_{bandwidth} & 10 \\
\lambda_{latency} & 10
\end{array}
$$

7.2 超参数

该模型中配置的主要超参数如下表所示。

Hyperparameter	Value
Learning rate (agent)	0.0001
Batch size	128
No. LSTM layers	(1, 3, 4)*
LSTM hidden size	(32, 64, 128)*
Embedding size	10
Learning rate (baseline)	0.1
Gradient clipped by norm	1
Temperature	15
No. samples with temp.	16
No. models inference	6
*Refer to Table III to consult in which scenario the different dimensionalities where used.	

在实验和调整结构的过程中，我们注意到，编码器-解码器结构的复杂性需要足够大，以嵌入问题的特征。然而，一旦满足了这个要求，增加其复杂性并不能带来更好的性能。在这种情况下，降低学习率、增加批量大小和鼓励探索会带来更好的结果。在本文中，我们专注于加强在推理时评估一些搜索策略的解决方案。我们一共评估了 6 个不同的模型，其中 16 个样本是在使用温度参数的窒息输出分布上提取的。这些在推理中使用的后处理技术并没有增加大量的计算时间，而且提供了明显的改进。根据我们的经验，从一个略微窒息的分布中取样是报告了最佳结果的技术。从多个模型中推断，本身也有好处，然而推断出的政策之间存在着明显的关联性。因此，在所使用的模型数量和所经历的改进之间存在着一种关系。

7.3 关于选择拉格朗日乘数的直觉

在本附录中，我们旨在提供一些关于选择拉格朗日乘数的直觉。让我们把公式（5）中描述的原始问题的最优值称为p^*，即目标函数在满足约束条件下所能获得的最小值。使用拉格朗日松弛技术，这个问题被转化为一个无约束的问题，其中不可行的解决方案被惩罚，见公式（6）。由此产生的对偶函数$g(\lambda)$总是凸的，即使原始函数和约束条件是非凸的，并且它给出了原始问题p^*的最优值的下限。

一般来说，我们的目标是找到能产生最佳下限d^*的拉格朗日乘数。这被称为拉格朗日对偶问题，在一般情况下，它被表示在图 7a 中。

我们所描述的问题有一些特殊之处。所有的约束函数都是正定的，因为它们分别被定义为占用率、带宽和延迟不满意的期望值。换句话说，一个可行的解决方案将是所有这些约束不满意信号的期望值都等于零。然而，在整个权重空间 θ 中，可能不存在符合这一要求的神经网络配置。

如果它满足这一要求，那么$d^*=p^*$，此时强对偶性成立，因为存在一个能够删

除惩罚项的点（互补松弛性）。

一般来说，一个可行的点不能被证明存在。在这种情况下，对偶函数不断随λ增加，因为函数的每一点都会带来越来越多的惩罚。因此，$d^* = \infty$。 图 7（b）中描述了这两种情况。在第一种情况下，存在一个$\lambda_{\min}$，在这一点上我们可以保证惩罚足够大，以确认对偶函数d^*的最优值对应于原始问题p^*的最优值。这个值是一个先验的未知数，但可以很容易地估计出来。看起来λ在$\lambda_{\min}$和∞之间的所有值都同样有效，但我们将在后面说明。

在第二种情况下，目标是不同的。一个原始的最优值p^*并不存在。我们的目标是找到拉格朗日乘数 λ，在惩罚J_C^π的期望值和奖励J_E^π的期望值之间建立起理想的承诺。为了澄清这个概念，让我们分析一下λ的极端值。如果$\lambda = 0$，那么拉格朗日J_L^π=J_E^π，优化器将最小化预期功耗而不注意约束。同样地，如果$\lambda \to \infty$，根据大数法则，拉格朗日将收敛于惩罚函数J_C^π。因此，Agent 要推断的政策是在不考虑解决方案的最优性的情况下，通过平均值实现更少的错误陈述的政策。在这种情况下，没有λ能使对偶函数$g(\lambda)$最大化，从而为我们提供一个参考点，即为了达到可行区域，或者在这种情况下，在约束不满意方面呈现最小的区域，超过其余的函数。

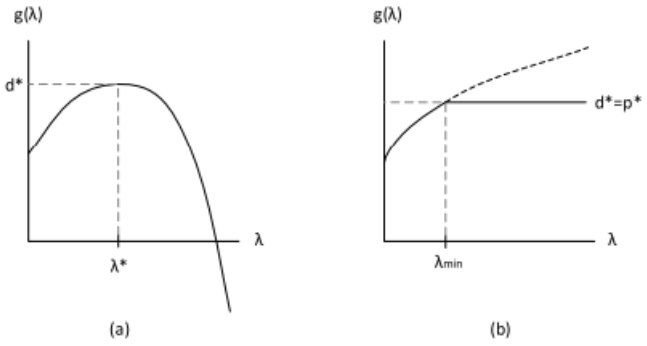

图 7. 在以下情况下的一般二元函数表示。(a)的可行解区域内存在利益，(b)的可行解区域内没有补贴或不存在可行解区域

在实践中，如果神经网络是一个足够好的近似器，那么就存在一个权重配置，其中$J_C^\pi \approx 0$。这给了我们一个直觉，即寻找拉格朗日乘数的区域是$\lambda + min$。一旦惩罚系数被设定，我们就会得到要优化的拉格朗日函数$J_L^\pi(\theta)$。这个拉格朗日函数是一个非凸函数，因此，不能保证优化器会达到全局最优值。通常情况下，这个函数被优化，直到达到一个鞍点或局部最优。先验地，我们没有关于收敛点的信息，因此需要对λ进行微调，以便在最优性和约束不满足之间设定所需的承诺。请注意，在这个微调过程中，我们的行动是沿着在极端情况下（$\lambda = 0$和$\lambda = \infty$）

的学习过程中取得的局部最优或鞍点的轨迹进行的。理论上，我们应该体验到有利于我们想通过控制λ来实现的目标的好处。然而，在现实中，没有任何东西能保证在这个过程中实现具有更好或更坏性能的不同优化路径。

7.4 算法实现

带基线估计器的单步蒙特卡洛策略梯度算法实现：

Algorithm 1

procedure TRAIN AGENT NETWORK (LEARNING SET $\mathcal{S}$, BATCH SIZE B)
  Initialize agent and critic networks with random weights θ and θ_ν
  for epoch = 1,2... **do**
    reset gradients: $d\theta \leftarrow 0$
    $s_j \sim \text{SampleInput}(\mathcal{S})$ for $j \in \{1, .., B\}$
    $p_j \sim \text{SampleSolution}(\pi_\theta(\cdot|s))$ for $j \in \{1, .., B\}$
    $b_j \leftarrow b_{\theta_\nu}(s_j)$ for $j \in \{1, .., B\}$
    compute cost function : $L(p_j)$ for $j \in \{1, .., B\}$
    $g_\theta = 1/B \cdot \sum_{j=1}^{B}(L(p_j) - b(s_j)) \cdot \nabla_\theta \log \pi_\theta(p_j|s_j)$
    $\mathcal{L}(\theta_\nu) = 1/B \cdot \sum_{j=1}^{B} \|b_{\theta_\nu}(s_j) - L(p_j|s_j)\|^2$
    $\theta \leftarrow \text{Adam}(\theta, g_\theta)$
    $\theta_\nu \leftarrow \text{Adam}(\theta_\nu, \mathcal{L}(\theta_\nu))$
  end for
  return θ and θ_ν
end procedure

参考文献

[1] Network Functions Virtualisation (NFV); Architectural Framework, document ETSI GS NFV 002 (V1.2.1), 2014.

[2] A. Marotta, F. D'Andreagiovanni, A. Kassler, and E. Zola, "On the energy cost of robustness for green virtual network function placement in 5G virtualized infrastructures," Comput. Netw., vol. 125, pp. 64–75, Oct. 2017.

[3] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," 2016, arXiv: 1611.09940. [Online]. Available: https://arxiv.org/abs/1611.09940

[4] M. Deudon, P. Cournut, A. Lacoste, Y. Adulyasak, and L.-M. Rousseau, "Learning heuristics for the TSP by policy gradient," in Proc. Int. Conf. Integr. Constraint Program., Artif. Intell., Oper. Res. Amsterdam, The Netherlands: Springer, 2018, pp. 170–181

[5] A. Mirhoseini et al., "Device placement optimization with Reinforcement Learning," in Proc. 34th Int. Conf. Mach. Learn., vol. 70, 2017, pp. 2430–2439.

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

[7] C. Tessler, D. J. Mankowitz, and S. Mannor, "Reward constrained policy optimization," 2018, arXiv:1805.11074. [Online]. Available: https://arxiv.org/abs/1805.11074

[8] C. Schulte, M. Lagerkvist, and G. Tack, Gecode, 2006, pp. 11–13. [Online]. Available: http://www.gecode.org

[9] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," in Proc. IEEE 4th Int. Conf. Cloud Netw. (CloudNet), Oct. 2015, pp. 171–177.

[10] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in Proc. 11th Int. Conf. Netw. Service Manage. (CNSM), Nov. 2015, pp. 50–56.

[11] H. Moens and F. De Turck, "VNF-P: A model for efficient placement of virtualized network functions," in Proc. 10th Int. Conf. Netw. Service Manage. (CNSM), Nov. 2014, pp. 418–423.

[12] A. Gupta, M. F. Habib, P. Chowdhury, M. Tornatore, and B. Mukherjee, "On

service chaining using virtual network functions in network-enabled Cloud systems," in Proc. IEEE Int. Conf. Adv. Netw. Telecommun. Syst.(ANTS), Dec. 2015, pp. 1–3.
[13] M. Shifrin, E. Biton, and O. Gurewitz, "Optimal control of VNF deployment and scheduling," in Proc. IEEE Int. Conf. Sci. Elect. Eng. (ICSEE),Nov. 2016, pp. 1–5.
[14] R. Riggio, A. Bradai, T. Rasheed, J. Schulz-Zander, S. Kuklinski, and T. Ahmed, "Virtual network functions orchestration in wireless networks," in Proc. 11th Int. Conf. Netw. Service Manage. (CNSM),Nov. 2015, pp. 108–116.
[15] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in Proc. IEEE Conf. Comput.Commun. (INFOCOM), Apr./May 2015, pp. 1346–1354.
[16] R. Mijumbi, "Placement and scheduling of functions in network function virtualization," 2015, arXiv:1512.00217. [Online]. Available: https://arxiv.org/abs/1512.00217
[17] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions," in Proc. IEEE Int. Symp. Integr. Netw. Manage., May 2015, pp. 98–106.
[18] S. Kim, S. Park, Y. Kim, S. Kim, and K. Lee, "VNF-EQ: Dynamic placement of virtual network functions for energy efficiency and QoS guarantee in NFV," Cluster Comput., vol. 20, no. 3, pp. 2107–2117,2017.
[19] R. Mijumbi, J.-L. Gorricho, J. Serrat, M. Claeys, F. De Turck, and S. Latré, "Design and evaluation of learning algorithms for dynamic resource management in virtual networks," in Proc. IEEE Netw. Oper. Manage. Symp. (NOMS), May 2014, pp. 1–9.
[20] R. Mijumbi, J.-L. Gorricho, J. Serrat, M. Claeys, J. Famaey, and F. De Turck, "Neural network-based autonomous allocation of resources in virtual networks," in Proc. Eur. Conf. Netw. Commun. (EuCNC),Jun. 2014, pp. 1–6.
[21] H. Yao, B. Zhang, P. Zhang, S. Wu, C. Jiang, and S. Guo, "RDAM: A reinforcement learning based dynamic attribute matrix representation for virtual network embedding," IEEE Trans. Emerg. Topics Comput., to be published.
[22] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in Proc. Adv. Neural Inf. Process. Syst. (NIPS), 2015, pp. 2692–2700. [23] R. S. Sutton et al., Introduction to Reinforcement Learning, vol. 135.Cambridge, MA, USA: MIT Press, 1998.
[24] D. P. Bertsekas, "Nonlinear Programming," J. Oper. Res. Soc., vol. 48,no. 3, p. 334, 1997.

[25] R. J. Williams, “Simple statistical gradient-following algorithms for Connectionist reinforcement learning,” Mach. Learn., vol. 8, nos. 3–4,pp. 229–256, 1992.
[26] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in Proc. Adv. Neural Inf. Process. Syst. (NIPS),2014, pp. 3104–3112.
[27] D. Bahdanau, K. Cho, and Y. Bengio, “Neural Machine Translation by jointly learning to align and translate,” 2014, arXiv:1409.0473. [Online].Available: https://arxiv.org/abs/1409.0473
[28] S. Kumaraswamy and M. K. Nair, “Bin packing algorithms for virtual machine placement in cloud computing: A review,” Int. J. Elect. Comput. Eng., vol. 9, no. 1, p. 512, 2019.

译文原文出处：

[1] Solozabal,,Ruben,et.al. Virtual Network Function Placement Optimization With Deep Reinforcement Learning[J].IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS,2020,38(2):292-303.