基于强化学习的5G网络中可靠感知的动态服务链调度

贾俊忠†，杨磊†*，曹建农‡
†华南理工大学软件工程学院，广州，中国
‡香港理工大学计算机系，香港，中国 电子邮件: sejjz@mail.scut.edu.cn, sely@scut.edu.cn, jiannong.cao@polyu.edu.hk
*通讯作者

摘要--
作为未来5G网络的关键推动因素，服务功能链（SFC）沿着虚拟网络功能（VNF）的链条转发流量，以提供网络服务的灵活性。SFC中最重要的问题之一是部署VNF，并在通信节点之间安排到达的请求，以实现低延迟和高可靠性。现有的工作考虑了一个静态的网络，并假设所有的SFC请求是预先知道的，这是不实际的。在本文中，我们专注于动态5G网络环境，其中SFC请求随机到达，计算节点可以以时间成本重新部署所有类型的VNF。我们将支持NFV的5G网络中的SFC调度问题表述为混合整数非线性编程。其目标是使满足延迟和可靠性约束的请求数量最大化。为了解决这个问题，我们提出了一种高效的算法来决定VNF的冗余度，同时使延迟最小化。然后，我们提出了一种最先进的强化学习（RL）来学习SFC调度策略，以提高SFC请求的成功率。我们的方法的有效性通过广泛的模拟进行了评估。结果表明，我们提出的RL解决方案可以比基准提高18.7%的成功率。

Index Terms-Service function chain, 5G network, reliability, reinforcement learning.

I. 简介

最近，软件定义网络（SDN）和网络功能虚拟化（NFV）在网络架构演进中发挥了重要作用。传统上，网络功能是由专用的硬件设备（中间盒）实现的，这增加了网络服务提供商的资金和运营成本，同时也造成了*网络耦合的*问题。当一种新的服务出现时，相关的硬件设备必须按一定顺序部署和连接。这种手工操作极其耗时、耗力和容易出错，阻碍了服务添加和网络升级的操作。为了应对这一挑战，NFV将网络功能的实现从专用硬件转移到基于软件的组件，命名为虚拟网络功能（VNF）[1]。VNF是抽象的网络功能，在普通商品（如基于X86的系统）服务器上运行，没有特定的硬件设备。

在NFV框架中，几个VNF实例的有序组合包括一个服务功能链（SFC）[2]。流经不同VNF序列的流量将建立多个SFC，可以支持各种网络服务，如防火墙、负载平衡器、深度包检测（DPI）。入侵检测系统（IDS），等等。NFV实现了SFC中使用的软件实现的网络功能的虚拟化。NFV被SFC采用，以提供高效和有效的网络功能部署和协调[3]。IETF SFC工作组（RFC 7665）和开放网络基金会（ONF）提出了SFC架构规范，展示了运营商网络、移动网络和数据中心网络的使用案例。

SFC调度问题已经在许多研究中被广泛讨论，但由于5G网络的低延迟和高可靠性要求，执行SFC调度具有挑战性。为了提高网络服务的可靠性，一个SFC的执行需要有额外的冗余VNF实例。冗余实例会占用更多的计算资源，从而导致其他网络服务的等待时间延长。因此，5G环境中的SFC调度器必须更加智能，以平衡延迟和可靠性。

在本文中，我们考虑了动态5G网络环境中的在线调度，其中SFC请求是以跑步方式到达的。在这种情况下，计算节点应该切换其部署的VNF类型，以满足传入的SFC请求中的新类型VNF。然而，在一个计算节点上重新部署各种VNF并不是没有成本的；节点必须关闭以前的VNF并设置新的VNF，这就引入了额外的重新部署时间。以前提出的网络模型假设一个节点在所有的时间段内只能承载单一类型的VNF。他们没有考虑计算节 点 上VNF的重新部署。这种静态部署策略导致在动态变化的SFC请求下，网络资源的利用效率不高。此外，一些现有的研究考虑了计算节点上的重新部署，但忽略了重新部署的时间成本。如果重新部署经常发生，就会在重新部署上花费大量的时间，增加请求的延迟。因此，解决方案应该平衡延迟和重新部署的问题。

为了解决这个问题，我们提出了一种高效的方法，将冗余的确定与SFC调度分开。对于前者，我们提出了一种启发式算法来决定VNF的冗余度，同时使延迟最小化。对于后者，我们开发了一种基于策略梯度的强化学习（RL）方法，将这些VNF放在计算节点上，目的是最大限度地提高成功率。

IEEE INFOCOM 2021 - IEEE 计算机通信会议 | 978-1-6654-0325-2/21/$31.00 © 2021 IEEE | DOI: 10.1109/INFOCOM42981.2021.9488707

SFC的要求。此外，我们设计了一种新的RL行动，与现有的将强化学习应用于调度问题的研究不同。在相关的工作中，行动决定了哪个节点应该部署VNF，其中行动空间的大小被定义为计算节 点 的数量。我们方法中的RL行动只是决定是否推迟VNF的执行。因此，我们的RL模型可以扩展到任意数量的计算节点，而无需修改RL网络模型。仿真结果显示，与基准算法相比，我们提出的方法可以提高18.7%的成功率。本文的主要贡献总结如下。

- 据作者所知，本文是第一个考虑在SFC请求随机到达的动态网络环境中具有延迟约束的可靠性感知的SFC调度问题的工作。这种环境中的计算节点可以承载所有类型的VNF，在切换承载的VNF类型时，需要花费重新部署的时间。
- 本文将可靠性感知调度问题表述为混合整数非线性编程（MINLP）问题，该问题为NP-hard，显示了SFC调度问题的复杂性和找到全局最优解的困难性。
- 我们提出了一种强化学习算法来提供一个可行的调度。我们的方法的有效性是通过全面的模拟来揭示的。结果表明，所提出的方法在SFC成功率方面优于其他基准。

II. 相关的工作

近年来，对VNF放置和SFC调度已经有了深入的研究。[4]提出了一个优化模型，以尽量减少资源的使用。对于每个实验，他们随机放置VNF，并根据日/夜计划注入流量。每次改变流量计划（12小时），都要重新运行混合整数线性编程，并对网络进行重新配置。在[5]中，作者为城域网中的SFC设置提出了一种动态VNF放置算法。在每个时刻，一定数量的用户请求一个特定的SFC，VNF被放置以最小化阻塞的概率。

文献[6]提供了一种用于VNF扩展的在线算法，以动态设置数据中心网络中的网络服务。文献[7]提出了一种启发式的SFC重构方法，并将其结果与CPLEX中实现的最佳解决方案进行了比较。他们没有考虑在重构SFC时，由于QoS降低而导致的运营商收入损失所带来的重构成本。[8]提出了一种基于水平扩展技术的合并算法，其中，通过实例化/删除VNF来增加/减少专用于VNF的处理能力，而不改变分配给VNF的处理能力。

网络服务的故障可能是正常的，对应用程序的性能有巨大的影响；因此，它是SFC调度中的关键问题之一[9]。在[10]中，作者提出了一个基于SDN的复制框架，用于可靠性感知网络服务。Sherry等人[11]开发了一个基于日志的网络服务恢复模型，该模型可用于恢复失败的网络服务。Fan等人[12]和Ye等人[13]研究了可靠服务链嵌入的问题，并提出了基于专用备份和共享备份配置的不同启发式算法。文献[14]提出了一种基于二方匹配的网络服务备份的网络无关的解决方案。文献[15]讨论了动态网络中的SFC可靠性问题，以最小化总安置成本。然而，它忽略了SFC请求的延迟约束。

上述工作主要集中在静态SFC环境下，所有的请求信息都是事先知道的，这在实际网络中是不可行的。在本文中，我们考虑到在动态网络环境中，SFC请求可能是随机出现的。与上述工作不同，我们的解决方案可以适应现实世界的场景。目前的工作目标大多集中在优化网络成本或只考虑延迟。为了满足5G网络的需求，我们的目标是减少平均端到端的延迟，同时保证可靠性。我们开发了一种基于强化学习（RL）的最先进的算法，用于在低延迟和可靠性感知的5G网络中执行动态SFC调度。

III. 系统模型和问题制定

在本节中，我们将介绍ETSI定义的NFV架构和NFV框架中的SFC映射过程。接下来，我们将介绍这项工作中的网络模型。

A. NFV架构

VNFs是从传统的基于硬件的中间箱中抽象出来的软件模块，需要一些物理的重新来源（即CPU、内存、带宽等）[16]。它们可以部署在虚拟机（VM）或商品服务器上托管的容器上，以提供可扩展和弹性的网络服务。一般来说，NFV架构中有三个组成部分。服务、NFV基础设施（NFVI）以及NFV管理和协调（NFV-MANO），如图1所示。对这些组件的简要描述如下。

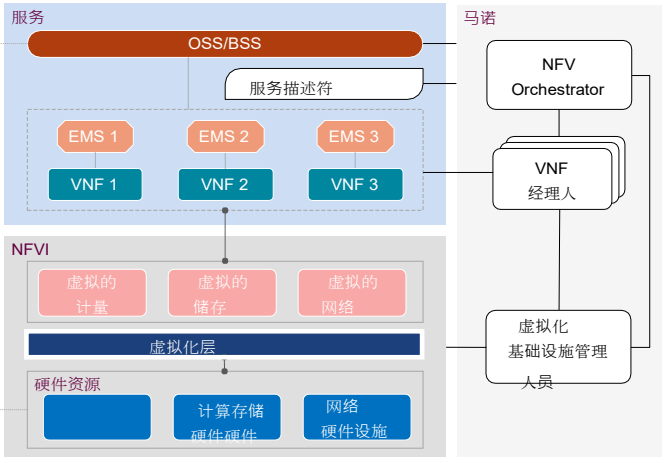


图1.ETSI NFV架构框架。

服务。一个服务是一个VNF序列，它可以在几个虚拟机（计算节点）中实现。一个VNF由一个元素管理系统（EMS）监管，负责其创建、配置、性能和观察。此外，EMS可以提供运营支持系统和业务支持系统（OSS/BSS）所需的信息，这有助于电信服务提供商部署和管理各种电信服务[17]。

NFVI。NFV基础设施是一套用于承载和连接VNF的资源。NFVI是一个数据中心，包括服务器、管理程序、操作系统、虚拟机、虚拟交换机和网络资源，它是基于广泛使用的低成本标准化计算组件。

MANO。NFV Orchestrator负责新的网络服务和VNF的入职，服务生命周期管理和全球资源管理等。VNF管理器负责监督VNF实例的生命周期管理，并协调EMS和NFVI。虚拟化基础设施管理器（VIM）控制NFVI的计算、存储和网络资源。

B. 证监会映射

一个网络服务是由几个网络功能组成的。在NFV环境中，这些功能被虚拟化为各种VNFs。SFC的映射问题旨在决定计算节点应在何处以及何时部署和处理给定的VNFs。ETSI将网络服务定义为几个有序的VNFs的序列[18]，即服务功能链（SFC）。数据包应逐一通过一组VNFs，这些VNFs在SFC中描述，以完成一个端到端的网络服务。图2显示了SFC的一个例子。它展示了一个典型的端到端网络服务 *防火墙* *负载平衡加密包* {*检查解密*。协调器负责网络服务链和VNF放置。它接受网络服务请求，将请求转化为 → 到SFC，并将VNFs映射到虚拟化层提供的虚拟机中。物理机（PM）是通过虚拟化层来提供抽象的为VNFs提供资源。

C. 网络模型

在这项工作中，我们考虑在5G网络的边缘部署一个物理网络。由于5G的高带宽，网络资源的主要限制是计算能力而不是网络传输。因此，我们假设物理网络是一个完全连接的网络。这个假设是为了抽象出网络层的路由和数据传输调度的细节，因为我们专注于应用层的SFC调度问题。在这种情况下，端到端的延迟可以被定义为处理时间和等待时间的总和。VNF的处理时间计算为：$p_f = w_f / V_k$，其中w_f是VNF f的计算负载，V_k是计算节点k的计算速度。请注意，在前辈VNF完成之前，后辈VNF不能开始执行。

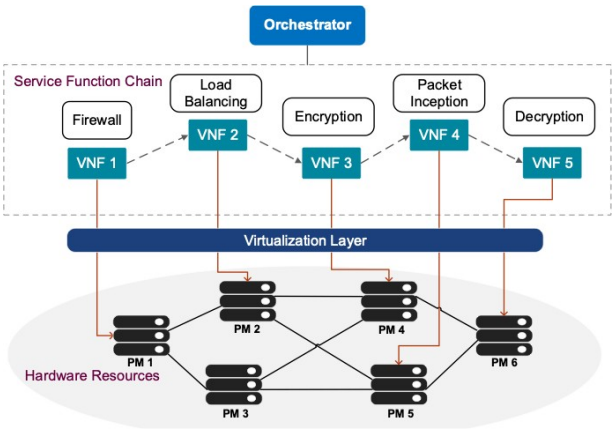


图2.NFV框架中的服务功能链映射。

可靠性是5G网络的一个关键性能指标。VNF的执行可能会因为硬件或软件故障而中断（例如，物理机的意外重启/关闭，网络断开，软件错误等）[19]。我们采用主动-主动冗余方案来提高可靠性，该方案部署了同一VNF的多个实例同时运行。根据ETSI[20]定义的可靠性模型，VNF的可靠性是指至少有一个冗余的子组件是可用的概率。我们将θ定义为计算节点的可靠性，它规定了节点上成功完成VNF的概率。那么，VNF的可靠性被定义为：。

$$r_f = 1 - \prod_{i}^{R_f} (1 - \theta) \tag{1}$$

其中，r_f 是 VNF f 的可靠性，R_f 是 f 的冗余度。关于端到端服务，端到端可靠性被计算为构成SFC的VNF的可靠性的乘积。因此，一个SFC的可靠性是。

$$R_i = \prod_{v \in s} \left[1 - \prod_{j}^{R_v} (1 - \theta) \right] \tag{2}$$

其中，v指的是SCF s中的一个VNF，而R_v是在这种情况下，实例化的冗余实例越多，SFC的可靠性就越高。然而，网络中极多的VNF实例会耗费日志队列时间，由于网络资源的限制，会导致更大的端到端延迟。因此，平衡冗余和延迟之间的权衡是很重要的。

在本文中，我们将虚拟机（VM）视为一个计算节点，它可以在物理服务器上运行。为了简单起见，我们认为每个节点一次只能承载一个VNF。当SFC请求到达时，我们的问题是决定每个VNF的冗余度，并将这些VNF实例放在节点上。一个节点可以切换其VNF的类型（即重新部署），有一个重新部署延迟Δ。此外，我们假设所有的节点具有相同的可靠性θ。为了提高一个VNF的可靠性，我们可以部署多个VNF实例，称为冗余VNF。冗余的VNFs可以运行

表一
数学表达式中的术语

条款	解释
N	NT 计算节点的集合，$k \in N$
S	ST SFC请求的集合
s_i	第i个SFC请求
s_{ij}	请求s的第j个VNF$_i$
F	S中VNF类型的集合，$f \in F$
Θ_i	s的可靠性要求$_i$
Φ_i	s的端到端截止时间$_i$
d_i	s的端到端延迟$_i$
R_i	SFC s的可靠性$_i$
B_{ij}	VNF s的冗余度$_{ij}$
w_f	VNF f的计算负荷
V_k	节点k的计算速度
θ	计算节点的可靠性
T	时间段的集合
Δ	部署一个VNF实例的延迟
R_m	最大冗余度

在不同的节点上或在同一节点上重新运行。

本文使用的符号见表一。

时间被划分为离散的时隙。设N是一个集合，代表网络中的所有计算节点。每个节点代表一个虚拟机（VM），其中节点k的计算速度为V_k。一个SFC由VNF的有序序列组成，其中一个链中的每个VNF都是不同的。彼此之间的关系。设S是一组SFC请求，其中s_{ij}是请注意，端到端是指请求*链中*的第j个VNF。

s_i的结束延迟不应超过其最后期限Φ_i。

我们系统模型中的假设如下。首先，SFC中的VNF的类型是相互不同的。第二，SFC是一个控制流，包括一连串的VNFs。在一个SFC中，一个VNF通常有非常有限的数量的数据到后续的VNF。因此，数据传输延迟可以被忽略。此外，连接计算节点的网络具有相对充足的带宽。我们假设在资源受限的环境中，计算成本和重新部署成本对性能有很大的影响，而两个连接的VNF之间的数据传输延迟被忽略了。第三，为了简单起见，我们认为计算节点的首次部署延迟可以是忽略不计。此外，每个节点一次只能承载一个VNF。最后，一个VNF的处理应该是连续的（非抢占式），不能被其他VNF打断。

D. 问题的提出

决策变量。我们使用$\hat{x}_{ij}^{\delta k}$，表示computing节点k在时隙δ开始处理VNF s_{ij}。

$$\hat{x}_{ij}^{\delta k} = \begin{cases} 1, & \text{节点}k\text{在时隙}\delta\text{开始处理}s_{ij} \\ 0, & \text{否则。} \end{cases} \tag{3}$$

我们还定义了一个辅助变量来表示VNF*的*$_{ij}$。在时隙δ对节点k进行处理。

$x^{\delta k}$和$\hat{x}^{\delta k}$之间的关系如下。

$$\frac{w_{ij}}{V_k}\hat{x}_{ij}^{\delta k} \le x_{ij}^{\delta k}。 \quad \forall i \in S, j \in s_i, k \in N。 \tag{5}$$

$$x_{ij}^{\delta k} \ge \hat{x}_{ij}^{\delta k}。$$

$$\forall i \in S, j \in s_i, k \in N, \delta, \delta' \in T: \delta \le \delta' \le \delta + \frac{w_{ij}}{V_k} - 1。 \tag{6}$$

端到端延迟约束。以下约束条件保证了一条链的端到端延迟应满足其最后期限要求。

$$d_i = \sum_{\delta}^{|T|} \delta \hat{x}_{i|s_i|}^{\delta k} - \sum_{\delta}^{|T|} \delta \hat{x}_{i1}^{\delta k} \le \Phi_i, \quad \forall i \in S, k \in N。 \tag{7}$$

可靠性约束。以下约束表明，VNF冗余实例的数量应确保SFC请求的可靠性要求。

$$R_i = \prod_{j}^{|s_i|} r_{ij} = \prod_{j}^{|s_i|} \left(1 - \prod_{k}^{|N|} (1-\theta)^{\sum_{\delta}^{|T|} \hat{x}_{ij}^{\delta k}}\right) \ge \Theta_i。 \tag{8}$$

$$\forall i \in S, j \in s_i, k \in N。$$

VNF部署延迟约束。当计算节点关闭其正在运行的VNF实例并部署不同类型的VNF实例时，有一个重新部署延迟Δ。

$$\delta \hat{x}_{ij}^{\delta k} \ge (\delta' + \Delta + 1)\, x_{ij}^{\delta' k}。$$

$$\forall \delta, \delta' \in T: \delta' \le \delta - 1,\ i, i' \in S, \tag{9}$$

$$j \in s_i,\ j' \in s_{i'} : f_{ij} = f_{i'j'}, k \in N。$$

序列顺序约束。以下约束确保请求s_i中VNF链的顺序。VNF s_{ij}应该在下一个VNF s_{ij+1}开始处理时完成。

$$\sum_{\delta}^{|T|} \delta \hat{x}_{ij}^{\delta k} + \frac{w_{ij+1}}{V_k} \le \sum_{\delta}^{|T|} \delta \hat{x}_{ij+1}^{\delta k}, \tag{10}$$

$$\forall i \in S,\ j \in s_i,\ k,\ k' \in N。$$

VNF处理约束。每个VNF *的* $_{ij}$，应该在一些计算节点中托管。以下约束确保一个VNF被实例化一次以上。

$$\sum_{k}^{|N|} \sum_{\delta}^{|T|} \hat{x}_{ij}^{\delta k} \ge 1, \quad \forall i \in S, j \in s_i, k \in N。 \tag{11}$$

在本文中，我们的目标是在给定的网络资源下，最大限度地提高成功的SFC请求的数量。如果一个请求s_i的延迟d_i不超过其最后期限Φ_i，则为成功。目标定义如下，其中I为*指标函数*。

$$\text{最大} \sum_{k}^{|N|} \mathrm{I}(d_i \le \Phi_i) \tag{12}$$

可靠性感知的服务功能链调度模型是一个混合整数非线性问题（MINLP），有一系列的约束条件，这是很复杂的问题。当我们考虑平面调度时，设置重新部署延迟x延迟约束，本文中的问题可以被简化为一个

$\delta_k = 1$，　　节点k在时隙δ处理s_{ij}　　(4)　　$\Delta=0$，忽略SFC的可靠性要求，并取消

灵活的工作车间调度问题（FJSP）。FJSP被证明是NP-hard的，因此我们的问题也是NP-hard的。因此，全局最优解不可能在多项式时间内得到，我们提出了一种基于强化学习的算法来生成一个次优解。

IV. 意识到可靠性的SFC调度的解决方案

根据[21]，基本的SFC调度问题可以被认为是一个扩展的灵活工作间调度问题（FJSP），该问题被证明是NP-hard。此外，可靠性意识到的SFC调度问题与重新部署在一个动态网络中，成本的计算要比基本的计算方法更复杂的情况。因此，要在多项式时间内找到一个最佳解决方案是非常困难的 。为了简化问题，可靠性意识的SFC调度程序被分解成两个子问题。第一个子问题是确定VNF冗余的最佳数量以保证可靠性要求。对于这个问题，我们提出了一个启发式的方法来确定冗余的最佳数量，同时避免高延迟。第二个子问题是将这些VNFs映射到适当的计算节点，目的是使成功的SFC请求的数量最大化。我们开发了一种基于RL的智能算法来处理SFC的映射问题。

A. 可靠性感知的动态SFC调度方法概述

为了解决这个问题，我们提出了一个可靠性感知动态SFC调度方法（RDSSA）。图3显示了这个方法的工作流程。它包括三个关键步骤。当有一个SFC请求时，冗余度测定被用来根据所需的SFC可靠性计算VNF的最佳冗余度。它规定了SFC中的哪个VNF应该被复制，以及应该部署多少个冗余的VNF。这些未排定的VNF在队列中等待。一个VNF的优先级是基于距离相应SFC的最后期限的剩余时间。基于规则的节点选择方法提供了一种策略，为排在队首的VNF选择一个合适的计算节点。在这个过程中，我们引入了一个名为延迟执行的机制，这意味着调度员可以延迟执行一个输入的VNF。其动机是为了减少多余的VNF所消耗的计算资源。在这种情况下，动态SFC调度的强化学习（Reinforcement Learning）会根据网络环境的变化来学习一个延迟策略。如果它决定推迟一个VNF，该VNF将被返回到队列中。否则，调度器采用基于规则的方法来为VNF分配节点。

B. 冗余度的确定

请注意，部署的冗余实例越多，SFC的可靠性就越高。然而，由于计算能力的限制，端到端的延迟将随着冗余数量的增加而增加。为了在不超过最后期限的情况下获得一个最佳的可靠性感知调度，它

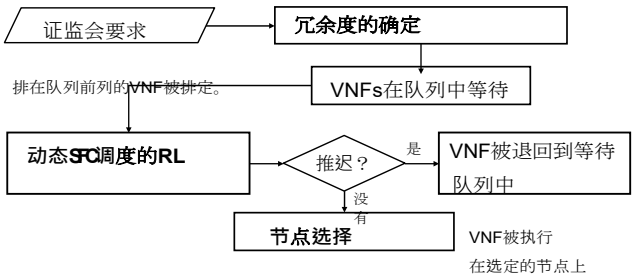


图3.拟议的RDSSA的工作流程。

需要确定冗余的最小数量，并决定SFC中哪些VNF应该被复制。

算法1：冗余度确定算法

输入：R_m, L, $S = s_i$, θ_i, Θ
输出。冗余度（s_i）：S的冗余度$_i$

```
设置一个1 ...一个L  Rm。
如果Rel({ai}) < Θ 那么
    返回不能满足给定的可靠性。
还有4个
    设置一个1 ...一个L 1;
    设置index1。
    while Rel({ai}) < Θ do
aindexaindex+1。
index (index + 1)%L。
    结束
S'    AscSortByComputationLoad( si {。}
for i1 to L do
    冗余 (s'i)          ← ai。
    结束
结束
```

我们提出了一种最佳冗余确定算法，如算法1所示。鉴于VNF的最大冗余数R_m，SFC的长度L，服务功能链S，计算节点的可靠性θ，以及SFC请求的可靠性要求Θ，该算法输出一个冗余列表A，表示SFC中每个VNF的冗余数，其中$Redundancy(s_i)$是VNF实例数s_i，$Rel(A)$计算A的可靠性。回顾一下，我们认为所有计算节点在其执行过程中有相同的可靠性θ。例如，如果有一个SFC，其中的VNF和所需的可靠性Θ如图4所示。一个计算节点的可靠性θ被设置为0.96。冗余列表A被初始化为1，然后从左到右循环递增，直到满足所需的可靠性Θ。之后，我们按计算负荷从高到低对VNF进行排序。在这种情况下，冗余VNF的数量与冗余列表中的数量相对应。算法1确保冗余列表A中的两个数字之间的差异小于1，这可以证明是在给定冗余量时的最大可靠性。

*定理1：*给定两个冗余a，b，其中a，$b \geq 2$和$a + b \leq R_m$，如果$0 < a < b < 1$，比$Rel(a, b)$最大化。

*证明。*为了用矛盾法证明它，试着假设

该陈述是错误的。设$p=1-\theta$，其中$0<\theta<1$，则$[1-(1-\theta)^a]-[1-(1-\theta)^a]=(1-p^a)-(1-p^b)$。因此。

$$(1-p^a)(1-p^b)<(1-p^{a+1})(1-p^{b-1}),-p^a(1-p)<p^{b-1}(1-p)。$$

$$-p^a<p^{b-1}。$$

$$a<b-1.$$

在这里，它与$a\ b$ 0达到了矛盾。 $-\geq$ ■

此外，我们更希望具有较低负载的VNF具有更多的冗余，因为与高负载的VNF相比，它的处理时间更短。通过这种方式，我们使用最小数量的冗余来保证可靠性要求，并减少冗余造成的额外处理时间。

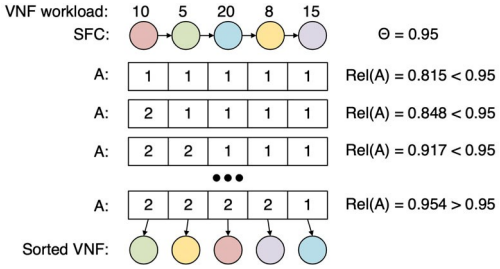


图4.一个确定冗余的例子。

C. 基于规则的节点选择方法

调度规则用于为VNF选择一个计算节点。现有的研究提出了一些精心设计的基于规则的算法。Zhang等人[22]提出了一种优先级驱动的加权算法来解决VNF放置问题，该算法可以低成本地找到一个接近最优的解决方案。在[23]中，作者开发了一种两阶段的算法来最小化VNF部署的总成本。然而，现有的SFC调度解决方案并不适用于本文的问题，因为它们考虑的是静态网络，所有的SFC请求都是事先知道的。在这种情况下，我们考虑基于规则的FJSP方法，如最早完成（EFF）和最早开始（ESF）。EFF选择能够提前完成VNF的节点。ESF则选择较早开始使用VNF的节点。这些简单的基于规则的方法只提供了可行的解决方案，我们将在后面提出强化学习的方法来提高其性能。

D. 强化学习用于动态SFC调度

我们开发了一个强化学习（RL），使用一个被称为策略网络的神经网络来动态优化SFC调度。强化学习代理输入当前的

我们的RL框架如图5所示，它包括了SFCs的状态，并输出了一个调度行动。我们的RL框架如图5所示。一个调度*代理*观察SFC的请求和计算节点*的状态*来决定一个调度行动。

在基于NFV的5G网络*环境*下的*行动*。代理人在执行行动后获得*奖励*，并更新网络参数。这里，代理使用嵌入网络将原始状态信息向量编码为RL网络的低维特征。

调度事件。我们的调度模型是任务触发的。未排定的VNF在队列中等待。RL代理

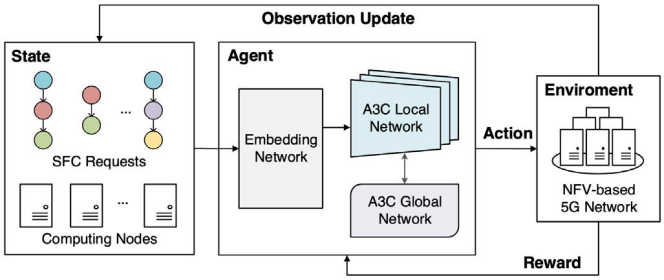


图5.拟议的强化学习框架。

当VNF等待队列不是空的时候，应该做出调度决定。该队列的行为如下。

- 队列的优先级是基于距离最后期限的剩余时间。剩余时间较少的SFC的VNF在队列中具有较高的优先级。
- 新到达的SFC的第一个VNF（连同其所有冗余）将被添加到队列中。
- 当一个VNF（及其所有冗余）完成后，其后续任务将被添加到队列中，除非前者是最后一个任务。此外，如果一个VNF的所有冗余都失败了，我们认为这个VNF的SFC被中止了，后续的VNF将不会被添加到队列中。
- 如果其中一个VNF冗余完成了，其他的将从队列中移除，如果它们已经被部署在节点上，它们的执行将被中止。
- 排在队首的VNF将被调度员（或RL代理）首先调度。

状态观察。状态是RL代理的输入。在本文中，状态包括VNF类型、链上剩余的VNF数量、链的长度、VNF计算负载、SFC的剩余长度、距离最后期限的剩余时间、节点维护的VNF类型以及每个节点的提前时间。VNF提前时间T_a 被计算为加权VNF最后期限T_w 和估计完成时间T_e之间的差，即$T_a = T_w - T_e$。估计完成时间T_e被定义为当前时间T_n、重新部署时间T_δ 和处理时间T_p之和，即$T_e = T_n + T_\delta + T_p$。VNF i在节点k的处理时间被计算为$T_p = w_{ij}/V_k$。当VNF s_{ij}的类型与节点k相同时，重新部署时间为0。否则，将有一个重新部署延迟Δ。

$$0，\text{如果VNF类型}f_i\text{与节点}k\text{的类型相同}$$
$$\Delta，\text{否则}。 \tag{13}$$

The weighted VNF deadline T_w is defined as the computation-load-weighted average value of SFC deadline:

$$T_w = \frac{\sum_{u \leq j} w_{iu}}{\sum_{v \leq |s_{ij}|} w_{iv}} \Phi_i 。 \tag{14}$$

T_w表示一个VNF的预期完成时间。如果一个VNF在其加权VNF截止日期之前完成，那么它就是一个适当的调度。否则，如果完成时间在其加权的VNF截止日期之后，则意味着该SFC在执行中滞后，最终可能超过截止日期。行动。

行动是RL代理的输出，它决定了VNF应该被部署在哪里。在这里，我们

引入一种叫做*延迟执行的*机制。当有一个VNF要部署时，调度员或RL代理可以*将*这个VNF推迟到下一个调度事件。延迟执行可以使一些多余的VNF不被首先执行，而是留在队列中等待。此时，先前启动的VNF可能会成功完成，这些推迟的VNF可以从队列中移除，减少冗余VNF占用的计算资源，并提高成功率。基于这一动机，我们在第五节B中评估了延迟执行的影响。图7中的结果显示，延迟执行对成功率有明显的影响，这支持了我们的动机。然而，固定概率的延迟执行不能适应动态的可用计算资源。因此，我们考虑开发一种方法，根据动态环境提供一个具有不同概率的延迟执行。在本文中，我们采用了与其他相关工作不同的强化学习的行动[2][24]。我们的网络输出一个*延迟率*，即输入的VNF将被延迟的概率。如果一个VNF被推迟，调度器将不分配节点来部署它，VNF将留在队列中等待下一个调度事件。否则，VNF将根据基于规则的方法被部署在一个节点上。重要的是，通过这种行动设计，我们的代理可以适应不同数量的计算节点而不改变模型。

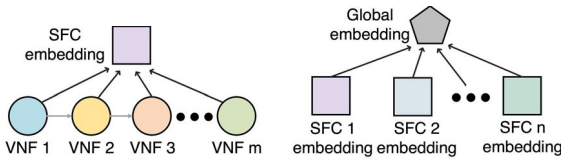
奖赏。奖励是来自环境的反馈
在一个行动被执行之后。为了指导代理，我们设计了一个基于SFC调度目标的奖励R。每个VNF

必须在其SFC截止日期前完成，以提高成功率。我们认为，如果一个链中的每个VNF都能

在其VNF最后期限T_w
之前完成，整个链条将满足最后期限；因此，我们使用VNF的提前时间作为奖励：$r = T_w - T_e$。当术语r是一个负数时，我们用负的奖励来惩罚代理，因为
的VNF的滞后完成。当术语r变成正数时，代理将由于VNF的完成时间早于其VNF的最后期限而得到奖励。请注意，在实验中，我们对大absolute值的奖励进行了约束，以防止模型因大的反馈信号而发生分歧。

E.SFC嵌入

RL代理将观察信息转换为特征，以传递给策略网络。传统上，特征向量被设计为包含所有的状态信息。然而，这种方法不能处理到达SFC的数量和SFC的长度都是任意的情况。在本文中，我们使用了一个叫做TextCNN的卷积神经网络。
[25]将状态信息编码为嵌入向量，以克服任意输入的困难并实现可扩展性。TextCNN是一种用于句子分类任务的有用方法。我们采用它作为状态信息的编码器，它可以将高维稀疏向量压缩成低维密集向量。我们的嵌入层输入SFCs信息并输出两级嵌入，如图6所示。第一层是*SFC嵌入*，如图6(a)所示，它汇总了整个SFC的信息。VNFs的链。SFC嵌入网络将SFC中的每个VNF特征转化为一个矢量。另一个层次是*全局嵌入*，它对系统中的所有SFC信息进行总结。同样，全局嵌入网络输入所有的SFC嵌入并输出一个全局总结向量。注意，SFC嵌入向量的大小为16，而全局嵌入向量的大小为64。



(a) SFC 嵌入。(b)全局嵌入。

图6.两个层次的嵌入网络。

F.培训

我们的RL代理使用政策梯度进行训练，即Asynchronous Advantage Actor-Critic（A3C）。与一些简单地基于价值迭代或策略梯度的技术不同，A3C结合了两种方法的优点，这意味着A3C预测了价值函数V和最佳策略函数π。
一个最佳行动。代理人使用A3C算法更新政策$\pi(a_i \; s_i \; ; \; \theta^{\mathrm{l}})$的参数$\theta$和价值函数$V(s_i \; | \; ; \; \theta^{\mathrm{vl}})$的参数$\theta_v$[26]，具体公式如下。

$$\mathrm{d}\theta \leftarrow \mathrm{d}\theta + \nabla_\theta \log \pi(a_i \,|s_i\,; \theta^{\mathrm{l}})(R - V(s_i\,; \theta^{\mathrm{vl}})) \quad (15)$$

$$\mathrm{d}\theta_v \leftarrow \mathrm{d}\theta_v + \frac{\partial (R - V(s_i\,; \theta^{\mathrm{vl}}))^2}{\partial \theta^{\mathrm{v}}_{\mathrm{l}}}. \quad (16)$$

这里，R是时间折算的奖励，$(R \quad V \quad (\bar{s_i} \; ; \; \theta^{\mathrm{vl}}))$估计在一个步骤中的奖励与长期奖励相比要好多少（或差多少）。$_\theta \log \pi(a_i\, s_i\, ; \theta^{\mathrm{l}})$表示增加制作的概率方向。
在状态$_i s_i$
。因此，方程（15）教导代理人选择那些可以获得高于平均奖励的行动。等式（16）旨在减少估计的奖励值和实际奖励之间的差异。A3C利用了多个代理（或工人），每个代理都有自己的本地参数和环境的副本。这些代理与他们的本地环境进行异步互动，并定期向全球网络上传更新。通过这种方式，全球网络可以更充分地探索参数空间。我们使用3个隐藏层来实现A3C网络，每层有64、32、16个隐藏单元。

V. 评价

在这一节中，我们首先介绍了我们模拟的设置。然后，我们进行模拟，评估我们提出的RDSSA解决SFC调度问题的性能。

A. 仿真设置

SFC 模 拟 器
。我们实现了一个基于Python的SFC模拟器，并使用PyTorch机器学习框架来构建和训练代理网络。所有的模拟都在英特尔酷睿i9 2.3GHz和32GB内存的机器上执行。

服 务 功 能 链 。根据[27]，在我们的模拟中，SFC的长度从2到6不等。每个VNF在100、150、180、200、250之间有自己的计算负荷。

300}，指的是计算量。证监会的可靠性要求在0.95，0.99，0.999，{ 0.9995中随机选择 } 证监会的到达遵循泊松过程，平均间隔为15个时间单位。

计算节点。计算节点的处理速度是在8、12、15、18中随机选择的。每个计算节点具有相同的可靠性θ=0.96，这表明在节点上成功执行VNF的概率。在整个VNF执行过程中，计算节点的故障可能在任何时候发生。我们假设该故障是一次性的事件，这不会影响后续任务。

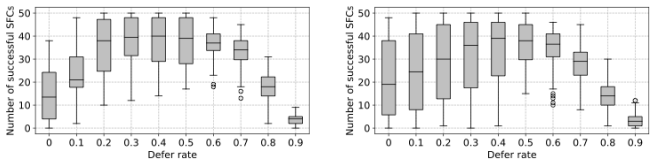
基准算法。由于与灵活的工作间调度问题的相似性，我们选择了两种稳健的基准算法（即最早完成的算法和最早开始的算法）。首先），这在FJSP中被广泛使用。此外，我们实现了一个深Q网络（DQN）模型作为比较方法。

- 最早完成优先。EFF是一种贪婪的算法，分配给能在最早时间完成的节点。
- 最早启动优先。ESF将VNF分配给能在最早时间启动VNF的com-puting节点。
- 深度Q网络。RL代理的DQN实现，只使用价值函数来做出行动。为了保证训练的收敛性，我们在这里使用与我们相同的嵌入层和奖励函数。DQN的行动遵循现有的工作[28][29]，决定应该选择哪个VNF，而不是推迟率。

此外，我们在EFF和ESF中引入了推迟率，这意味着一个VNF将以一定的概率被推迟。

B. 业绩成果

递延率的影响。我们首先评估EFF和ESF在延迟率变化时的性能。延迟率是指输入的VNF被推迟的概率。在延迟率较低的情况下，调度员更倾向于尽快为VNF分配节点。这导致大量冗余的VNF同时执行，占用了大量的计算资源。此外，在高延迟率的情况下，调度器不会首先执行多余的VNF，而是在先前启动的VNF成功完成后，将队列中多余的VNF删除。这导致冗余VNF在前一个VNF失败后被执行，增加了等待时间。图7显示了100个实验中成功的SFC的平均数量的分布。对于EFF，适当的延迟率应该在0.3到0.5之间。对于ESF来说，当延迟率在0.5左右时，成功的SFC的平均数量达到了一个峰值。这表明，当网络中存在大量冗余的VNF时，延迟率对成功率起着重要作用。然而，传统的恒定延迟率的方法不能根据动态的可用计算资源来调整延迟率，这导致了低成功率。在这种情况下，我们提出了一种基于学习的方法，可以随着网络状态的变化提供更好的延迟率。



(a) 递延 率不同的EFF 。(b) 递延率不同的ESF。图7.不同延迟率下的成功SFC数量。

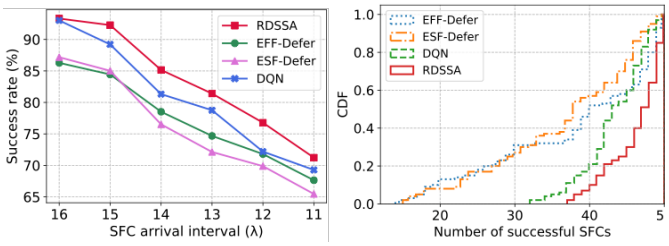


图8.不同SFC到达间隔的成功率。 图9.成功的SFC的CDF。

证监会到达时间间隔的影响。证监会到达时间间隔表示两个连续的SFC请求之间的平均时间间隔，这意味着工作到达的强度。为了研究影响我们提出的算法的成功率的SFC到达频率，我们改变了请求间隔并测试SFC的成功率。这里，重新部署的时间Δ被设定为150个时间单位。请注意，EFF-Defer（或ESF-Defer）是具有延迟执行的EFF（或ESF）方法。EFF-defer和ESF-defer的延迟率被设定为0.4和0.5 分别，基于图7所示的最大成功率。图8描述了结果，显示成功率随着小的间隔而下降。显然，相同的计算节点可以提供有限的服务能力来部署VNFs。随着时间间隔的减少，SFC请求的密度增加。在这种情况下，成功率会下降，因为有限的计算资源无法承受大的并发请求。仿真结果显示，我们提出的算法在所有测试间隔中都获得了远优于基准的性能。ESF-Defer和ESF-Defer以固定的概率推迟VNF，不能适应变化的请求间隔。DQN的作用是决定VNF在哪个节点上执行，而不推迟执行。这导致大量冗余的VNF在节点上执行，占据了计算资源，从而降低了成功率。此外，如图9所示，RDSSA比其他基准更稳定，平均成功率更高。计算节点数量的影响。我们评估了不同数量计算节点的性能，如图10所示。结果显示，所有方法的成功率都随着节点的增加而增加。当有足够多的节点时，所有的方法都能达到很高的成功率。请注意，在相同的SFC请求到达的情况下，成功率受到计算能力的影响。计算能力可以受到节点和调度策略的影响。显然，有了更多的计算节点，计算资源对于固定数量的SFC是足够的，从而导致更高的SFC成功率。RDSSA可以通过分析请求的信息来充分利用计算资源，保留所需的计算资源（某些特定的）。

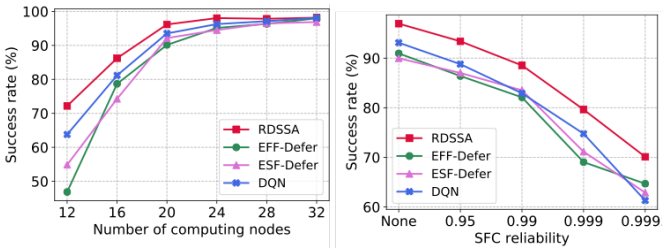


图10.不同的成功率 图11.成功率与可靠性的关系：计算节点的数量。

VNF类型的节点），并分配适当的节点，以适应请求的最后期限，以实现全球的高成功率。我们可以发现，当节点数量相对较少时，我们提出的方法可以学到更好的政策。

可靠性的影响。我们接下来要研究的是影响的所需SFC可靠性。为了获得合理的成功率，这里的到达间隔被设定为12个时间单位。一个更好的调度策略应该明智地推迟冗余的VNF；也就是说，当一个计算节点上已经有一个VNF在运行时，它的备份应该被推迟，以免占用过多的计算资源。此外，当VNF的执行失败时，冗余的VNF应该被部署并尽快完成，因为失败的VNF已经花费了时间。图11显示，我们提出的方法优于其他基准，特别是在高可靠性要求下（0.999时）。结果显示，当要求的SFC可靠性增加时，成功率下降。这是因为可靠性越高，系统中冗余的VNF就越多，导致对计算资源的竞争，最终SFC无法在截止日期内完成。

调配延迟的影响。我们进一步比较如图12所示，在我们的网络环境中，计算节点可以部署所有类型的VNF，在切换托管的VNF类型时，需要付出重新部署的时间成本。减少重新部署的时间成本是为了提高计算资源的有效利用。结果显示，成功率随着重新部署延迟的增加而下降。在较高的重新部署延迟的情况下，RL的成功率提高得更多，这意味着RL代理可以学习如何重复使用相同类型的虚拟机，避免过多的重新部署。在Δ=120时，我们的方法比基于规则的方法至少增加了18.7%的成功率。

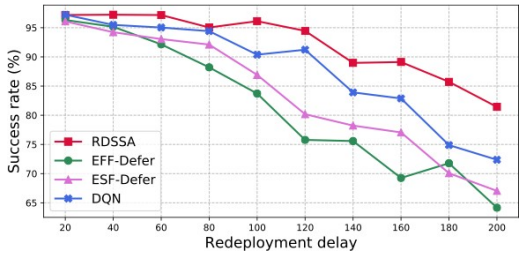


图12.不同重新部署延迟的成功率。

所提方法的收敛性。最后，我们将我们提出的方法与DQN进行比较。图13显示了学习曲线

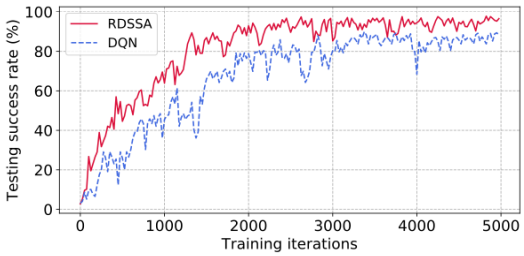


图13.学习曲线的比较。

这两个模型，每25次迭代测试模型。我们可以发现，我们的方法在训练效率方面优于DQN。RDSSA由多个独立的工作器（本地网络）组成，它们有自己的参数，使用多核CPU上并行运行的线程技术。每个工作者与环境的副本互动，并定期向全球/主网络更新梯度。因此，这些工作者可以在更短的时间内探索更大的状态动作空间。此外，与传统的基于价值迭代方法的DQN模型不同，我们使用的模型结合了价值函数$V(s)$和策略函数$\pi(s)$。代理人通过提供状态值，使用$V(s)$的输出来指导$\pi(s)$，并更新该状态下选择行动的概率。结果表明，我们的方法在最终成功率和收敛速度上都优于DQN。

VI. 结论

本文研究了基于NFV的5G网络中可靠性感知的SFC调度问题。为了实现高可靠性和低延迟之间的权衡，我们将该问题表述为一个NP-hard的MILP问题。我们的方法包括两个步骤，即冗余确定和SFC调度。对于前一个步骤，我们开发了一种高效的算法来确定最小的冗余数量，以保证可靠性而不需要太多的处理时间。在后一步骤中，我们提出了一种基于强化学习的调度算法，以及一种将特征编码为低维向量的嵌入技术。最后，仿真结果表明，所提出的方法在提高动态到达的SFC的成功率方面是有效的，特别是在高的重新部署时间。作为未来的工作，我们希望考虑一个多资源的网络环境（即带宽、CPU和内存），并在这个现实生活的模型中解决SFC调度问题。

鸣谢

这项工作得到了国家自然科学基金61972161的部分支持，广东省基础与应用基础研究基金2020A1515011496的部分支持，广东省重点领域研究发展计划2019B010154004的部分支持，以及香港研究资助局一般研究基金理大15217919和理大152133/18的部分支持。

参考文献

[1] D.Zheng, C. Peng, X. Liao, and X. Cao, "Toward optimal hybrid service function chain embedding in multiaccess edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp.6035-6045, 2020.

[2] X.Fu, F. R. Yu, J. Wang, Q. Qi, and J. Liao, "Service function chain embedding for nfv-enabled iot based on deep reinforcement learning," *IEEE Communications Magazine*, vol. 57, no. 11, pp.102-108, 2019.

[3] A.M. Medhat, T. Taleb, A. Elmangoush, G. A. Carella, S. Covaci, and T.Magedanz，"下一代网络中的服务功能链。State of the art and research challenges," *IEEE Communications Mag- azine*, vol. 55, no. 2, pp.216-223, 2016.

[4] M.R. Raza, M. Fiorani, A. Rostami, P. Ohlen, L. Wosinska, and P.Monti, "Benefits of programmability in 5g transport networks," in *2017 Optical Fiber Communications Conference and Exhibition (OFC)*, 2017, pp.

[5] L.Askari, A. Hmaity, F. Musumeci, and M. Tornatore, "Virtual-network- function placement for dynamic service chaining in metro-area net- works," in *2018 International Conference on Optical Network Design and Modeling（ONDM）*.IEEE，2018，pp.136-141。

[6] X.Wang, C. Wu, F. Le, A. Liu, Z. Li, and F. Lau, "Online vnf scaling in datacenters," in *2016 IEEE 9th International Conference on Cloud Computing（CLOUD）*.IEEE，2016年，第140-147页。

[7] T.Wen, H. Yu, G. Sun, and L. Liu, "Network function consolidation in service function chaining orchestration," in *2016 IEEE International Conference on Communications（ICC）*.IEEE，2016年，第1-6页。

[8] M.Ghaznavi, A. Khan, N. Shahriar, K. Alsubhi, R. Ahmed, and R.Boutaba，"弹性虚拟网络功能放置"，*在2015年IEEE第四届云计算国际会议（CloudNet）*上。IEEE，2015，pp.255-260。

[9] R.Yu, G. Xue, and X. Zhang, "Qos-aware and reliable traffic steering for service function chaining in mobile networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp.

[10] S.Rajagopalan, D. Williams, and H. Jamjoom, "Pico replication:一个用于中间箱的高可用性框架，"*《第四届云计算年度研讨会论文集》*，2013年，第1-15页。

[11] J.Sherry, P. X. Gao, S. Basu, A. Panda, A. Krishnamurthy, C. Maciocco, M.Manesh, J. Martins, S. Ratnasamy, L. Rizzo *et al.* , "Rollback-recovery for middleboxes," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015, pp.227-240.

[12] J.Fan, M. Jiang, O. Rottenstreich, Y. Zhao, T. Guan, R. Ramesh, S. Das, and C. Qiao, "A framework for provisioning availability of nfv in data center networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2246-2259, 2018.

[13] Z.Ye, X. Cao, J. Wang, H. Yu, and C. Qiao, "Joint topology design and mapping of service function chains for efefcient, scalable, and reliable network functions virtualization," *IEEE Network*, vol. 30, no.3, pp. 81-87, 2016.

[14] Y.Kanizo, O. Rottenstreich, I. Segall, and J. Yallouz, "优化中间箱的虚拟备份分配，" *IEEE/ACM Transactions on Networking*, vol. 25, no.5, pp. 2759-2772, 2017.

[15] M.Karimzadeh-Farshbafan, V. Shah-Mansouri, and D. Niyato, "A dynamic reliability-aware service placement for network function virtualization (nfv)," *IEEE Journal on Selected Areas in Communications*, vol.38, no.2, pp.318-333, 2019.

[16] R. Mijumbi, J. Serrat, J.-L. Gorricho, S. Latre, M. Charalambides, and D. Lopez, "Management and orchestration challenges in network functions virtualization, " *IEEE Communications Magazine*, vol. 54, no. 1, pp.

[17] J.Quittek, P. Bauskar, T. BenMeriem, A. Bennett, and M. Besson, "网络功能虚拟化（nfv）-管理和协调，" *ETSI NFV ISG, 白皮书*, 2014。

[18] G. ETSI，"网络功能虚拟化（NFV）。建筑框架，"*ETSI Gs NFV*，第2卷，第2期，第V1页，2013年。

[19] L.Qu, C. Assi, K. Shaban, and M. J. Khabbaz, "A reliability-aware network service chain provisioning with delay guarantees in nfv-enabled enterprise datacenter networks, " *IEEE Transactions on Network and Service Management*, vol. 14, no.3, pp. 554-568, 2017.

[20] N.Isg，"网络功能虚拟化（NFV）；可靠性；关于端到端可靠性的模型和特征的报告，"*ETSI GS NFV-REL*，第1卷，第1页，2016年。

[21] F. Riera, E. Escalona, J. Batalle, E. Grasa, and J. A. Garcia- Espin, "虚拟网络功能调度。概念和挑战，" in *2014 international conference on smart communications in network technologies (SaCoNeT)*.IEEE, 2014, pp.1-5.

[22] Q.Zhang, Y. Xiao, F. Liu, J. C. Lui, J. Guo, and T. Wang, "Joint optimization of chain placement and request scheduling for network function virtualization," in *2017 IEEE 37th International Conference on Distributed Computing Systems（ICDCS）*.IEEE，2017，第731-741页。

[23] Gu, J. Hu, D. Zeng, S. Guo, and H. Jin, "地理分布式数据中心的服务功能链部署和网络流调度", *IEEE网络科学与工程期刊*, 2020, doi: 10.1109/TNSE.2020.2997376.

[24] H.Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning scheduling algorithms for data processing cluster," in *Proceedings of the ACM Special Interest Group on Data Communication*, 2019, pp.270-288.

[25] Y.Kim，"用于句子分类的卷积神经网络"。*arXiv预印本arXiv:1408.5882*，2014。

[26] V.Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D.Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reincement learning," in *International conference on machine learning*, 2016, pp.1928-1937.

[27] W.Haeffner、J. Napper、M. Stiemerling、D. Lopez和J. Uttaro，"移动网络中的服务功能链用例"，IETF草案，2015。

[28] G. Li, H. Zhou, B. Feng, G. Li, and S. Yu, "Automatic selection of security service function chaining using reinforcement learning," in *2018 IEEE Globecom Workshops (GC Wkshps)*, 2018.

[29] X.Chen, Z. Li, Y. Zhang, R. Long, H. Yu, X. Du, and M. Guizani, "Reinforcement learning based qos/qo-aware service function chaining in software-driven 5g slices," *arXiv preprint arXiv:1804.02099*, 2018.