

文章编号:1007-5321(2021)01-0007-07

DOI:10.13190/j.jbupt.2020-035

移动网络 SFC 部署与计算资源分配联合算法

张天魁¹, 王筱斐¹, 杨立伟², 杨鼎成³

(1. 北京邮电大学 信息与通信工程学院, 北京 100876; 2. 中国农业大学 信息与电气工程学院, 北京 100083;
3. 南昌大学 信息工程学院, 南昌 330031)

摘要: 在网络功能虚拟化的移动核心网中,提出了一种基于服务功能链(SFC)部署与计算资源分配联合算法. 首先考虑 SFC 中虚拟网络功能(VNF)计算资源分配对处理时延的影响,建立 SFC 部署与计算资源分配联合优化问题,实现 SFC 的部署成本和端到端时延加权的最小化. 其次,为了求解所提优化问题,利用多智能体深度确定性策略梯度算法,从 SFC 各 VNF 的历史数据中学习策略指导即时的通用服务器节点选择和计算资源分配,提出了相应的 SFC 部署与计算资源分配联合算法. 仿真结果表明,所提算法可以在保证 SFC 的服务质量需求的条件下实现部署成本和端到端时延的有效权衡.

关键词: 移动核心网; 服务功能链; 计算资源; 多智能体深度确定性策略梯度

中图分类号: TN929.53

文献标志码: A

A SFC Deployment and Computation Resource Allocation Joint Algorithm in Mobile Networks

ZHANG Tian-kui¹, WANG Xiao-fei¹, YANG Li-wei², YANG Ding-cheng³

(1. School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China;
2. College of Information and Electrical Engineering, China Agricultural University, Beijing 100083, China;
3. School of Information Engineering, Nanchang University, Nanchang 330031, China)

Abstract: In the mobile core networks, a service function chain (SFC) deployment and computation resource allocation joint algorithm for network function virtualization is proposed. Considering the impact of computation resource allocation of virtual network function(VNF) on processing delay in the SFC, a joint optimization problem of SFC deployment and computation resource allocation is established to minimize the weighted sum of deployment cost and end to end service delay of SFCs. To deal with the proposed problem, the multi-agent deep deterministic policy gradient algorithm is used to learn the strategy from the historical data of each VNF in the SFC to guide the immediate selection of general server nodes and computation resource allocation, based on which, a SFC deployment and computation resource allocation joint algorithm is proposed. Simulations show that the proposed algorithm can achieve an effective trade-off between deployment cost and end to end service delay while ensuring quality of service requirements of SFCs.

Key words: mobile core network; service function chain; computation resource; multi-agent deep deterministic policy gradient

收稿日期: 2020-03-31

基金项目: 国家自然科学基金项目(61971060)

作者简介: 张天魁(1980—), 男, 教授, E-mail: zhangtiankui@bupt.edu.cn.

为了应对未来差异化多样性的业务需求,移动网络中引入了虚拟化技术,以提高网络的灵活性和可扩展性^[1]. 基于虚拟化技术,第 5 代移动通信系统(5G, the fifth generation of mobile communications system)的核心网采用基于服务的架构(SBA, service based architecture),并将各核心网元功能虚拟化为虚拟网络功能(VNF, virtual network function). 核心网中的 VNF 包括会话管理功能(SMF, session management function)、策略控制功能(PCF, policy control function)、应用功能(AF, application function)和用户平面功能(UPF, user plane function)等^[2]. 各种类型的核心网元功能组链为服务功能链(SFC, service function chain),代表了不同需求的网络服务^[3].

在 SFC 部署的过程中,各个 VNF 需要基础设施网络为其提供物理资源,如计算、存储等,这些资源通常被假设为一个固定值^[4]. 事实上,物理网络资源分配量对服务性能有很大的影响,在通用硬件设备上所部署的 SFC 的时延性能受分配给各 VNF 计算资源量的影响,并且在达到某个计算资源分配量时,时延存在一个下限值^[5]. Alleg 等^[5]考虑了分配给 VNF 的计算资源量与其处理延迟之间的线性关系,提升了系统的服务接受率. 魏等^[6]建立了最小化 SFC 的平均链路时延和负载均衡的多目标优化,通过 Q 学习算法来解决 VNF 映射问题. 以上文献在实现 VNF 映射时均未考虑物理节点的 CPU、内存、存储等物理网络资源的分配情况. 此外,在移动网络中,无线接入网对端到端网络切片性能的影响不容忽视. 高等^[7]研究了无线接入网侧的 SFC 部署算法,最大化网络运营商收益. Zhou 等^[8]提出了无线接入网侧的网络切片资源分配算法. Tong 等^[9]则研究了端到端网络切片的 SFC 部署与资源分配联合优化问题.

在移动核心网中考虑计算资源分配量与 VNF 处理时延的线性关系,提出了一种 SFC 部署与计算资源分配联合算法. 所提算法将核心网中 SFC 部署和计算资源分配问题建模为以最小化 SFC 部署成本和端到端时延加权和为目标的最优化问题,并且利用深度强化学习领域中的多智能体深度确定性策略梯度(MADDPG, multi-agent deep deterministic policy gradient)算法进行求解. 最后给出性能仿真结果与分析.

1 系统模型

在 SBA 支持下,未来移动网络通过构建多个网络切片以服务于不同的应用场景. 网络切片可以是一条或多条 SFC,这意味着 SFC 可以单独或组合在一起提供服务. 采用 5G 核心网 SFC 模型,如图 1 所示. 整个核心网主要由 3 部分构成,包括专有网络控制功能云、用户面功能云和业务应用中心云. 其中,SFC1 为会话管理 SFC,主要包含策略选择及会话管理的网元功能,如 SMF、PCF、AF 等;SFC2 为用户面 SFC,主要包含负责数据路由和转发的多个 UPF 网元功能;SFC3 为业务应用 SFC. 物理网络可由异构的服务节点、交换节点和存储节点组网构成. 虚拟网络中的 SFC 由各个 VNF 功能按顺序组合而成. VNF 可以部署在物理网络中的任意通用服务器节点上,所部署的通用服务器节点为其分配 CPU、内存、磁盘等资源. 为了降低模型的复杂度,将物理节点上的各类资源统一看作是计算资源.

将虚拟网络运营商请求的 SFC 集合表示为 $K = \{1, \dots, k\}$, k 表示 SFC 的指示;可部署在物理网络中的不同种类的 VNF 集合为 $F = \{f_x | x = 1, \dots, m, \dots, M\}$, 其中, M 代表系统中 VNF 的总数, m 表示 VNF 的指示. 因此系统的第 k 条 SFC 的 VNF 组合可表示为 $\{f_m^k | f_m^k \in F\}$.

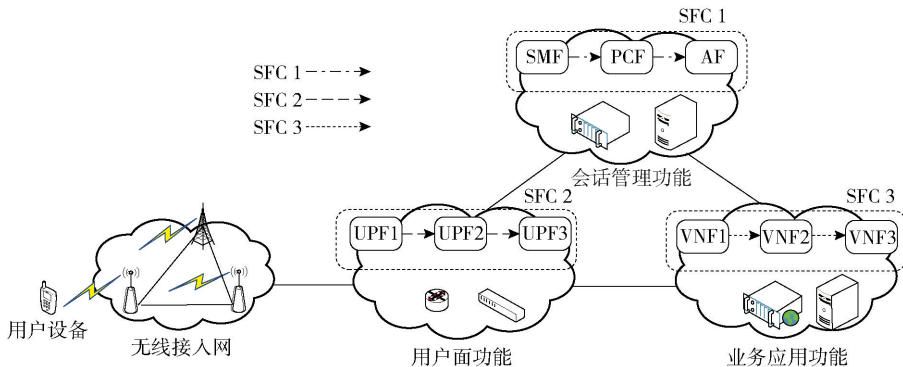


图 1 5G 核心网 SFC 模型

由于 VNF 处理时延受通用服务器节点为其分配的计算资源量影响^[10], 每个 SFC 中 VNF 所分配的计算资源量与其所需的数据处理速率呈线性关系^[8], 相应的, 第 k 条 SFC 中的第 m 个 VNF 所产生的处理时延 d_m^k 与计算资源分配量 ϕ_m^k 的函数关系表示为

$$d_m^k = a_m \phi_m^k + b_m \quad (1)$$

其中系数 a_m 和 b_m 可表示为

$$a_m = \frac{d_{\max}^m - d_{\min}^m}{\psi_{\min}^m - \psi_{\max}^m} \quad (2)$$

$$b_m = \frac{d_{\min}^m \psi_{\min}^m - d_{\max}^m \psi_{\max}^m}{\psi_{\min}^m - \psi_{\max}^m} \quad (3)$$

其中: d_{\min}^m 和 d_{\max}^m 为第 m 个 VNF 的最小和最大处理时延, ψ_{\min}^m 和 ψ_{\max}^m 分别为第 m 个 VNF 最小和最大计算资源分配量. a_m 为 ϕ_m^k 与 d_m^k 函数的斜率, 表示了计算资源分配量对处理时延的影响. $|a_m|$ 较小时, 计算资源分配对处理时延影响较小. 当分配给第 m 个 VNF 的计算资源量超过 ψ_{\max}^m 时, 达到其处理时延的下界 ψ_{\min}^m ; 当分配给第 m 个 VNF 的计算资源量少于 ψ_{\min}^m , 该 VNF 将无法执行.

2 SFC 部署与计算资源分配联合算法

2.1 优化问题

当第 m 个 VNF 映射在物理节点 n 上时, 其处理时延将受物理节点实际的 CPU 运行速度、服务器温度等因素的影响. 结合实际的经验, CPU 运行速度越快, 通用服务器节点的租用价格越昂贵. 处理器物理节点 n 上的计算资源的单价表示为 δ_n . 定义 $\eta_{m,n}^k = 1$ 表示第 k 条 SFC 的第 m 个 VNF 映射在物理节点 n 上, 否则 $\eta_{m,n}^k = 0$. SFC 端到端时延由处理时延和传输时延 2 部分组成. 第 k 条 SFC 将映射在物理节点 n 上的第 m 个 VNF 所产生的处理时延为

$$d_{m,n}^k = \rho_n d_m^k \quad (4)$$

其中 $\rho_n = 1/\delta_n$ 表示物理节点时延系数. 该时延系数的物理意义为: 物理节点 n 的计算资源单价低将导致愿意部署的 VNF 多; 相应的, 获得的计算资源越少, 则 SFC 的处理时延越高. 第 k 条 SFC 的处理时延表示为

$$D_{\text{pr}}^k = \sum_{f_m^k \in F} d_{m,n}^k \quad (5)$$

第 k 条 SFC 的传输时延表示为

$$D_{\text{tr}}^k = \sum_{f_m^k \in F} \sum_{n, n' \in N} \eta_{m,n}^k \eta_{m,n'}^k d_{n,n'} \quad (6)$$

其中: N 为处理器节点集合, 点 n' 为第 k 条 SFC 中第 m 个 VNF 映射的物理节点 n 的后续物理节点, $d_{n,n'}$ 为物理节点 n 与 n' 之间的传输时延. 因此, 第 k 条 SFC 的端到端时延可表示为

$$D^k = D_{\text{pr}}^k + D_{\text{tr}}^k \quad (7)$$

第 k 条 SFC 的部署成本可表示为

$$C^k = \sum_{f_m^k \in F} \sum_{n \in N} \eta_{m,n}^k \phi_{m,n}^k \delta_n \quad (8)$$

在该模型中, 系统的部署成本与网络时延存在相互制约的关系. 若以减少系统成本为目标, 则会导致网络时延变长; 反之, 以降低网络时延为目标, 系统成本则会增加. 从服务提供商的角度出发, 希望以合适的部署成本得到可接受的端到端时延, 实现两者的折中. 因此, 最小化系统中所有 SFC 的部署成本和端到端时延加权和优化目标表示为

$$\begin{aligned} & \min_{\eta, \phi} \sum_{k \in K} (\alpha C^k + \beta D^k) \\ & \text{s. t. } C1: \sum_{n \in N} \eta_{m,n}^k = 1 \\ & C2: \psi_{\min}^m \leq \phi_{m,n}^k \leq \psi_{\max}^m \\ & C3: \sum_{k \in K} \sum_{f_m^k \in F} \eta_{m,n}^k \phi_{m,n}^k \leq V_n^{\max}, \forall n \in N \\ & C4: D^k \leq D_{\text{lim}}^k \\ & C5: \alpha > 0, \beta > 0 \end{aligned} \quad (9)$$

其中 α 和 β 为权重系数. 约束条件 C1 表示每个 SFC 中的 VNF 只能部署在一个物理节点上, 即不允许 VNF 再拆分; 约束条件 C2 表示 VNF 应满足其最小/最大计算资源分配量限制; 约束条件 C3 表示部署在某物理节点的所有 VNF 的计算资源总分配量不得超过此节点可提供的总计算资源量; 约束条件 C4 表示 SFC 的端到端时延不得超过其可容忍的时延限制; 约束条件 C5 表示 2 个权重系数的取值均为正数.

采用深度强化学习领域中的 MADDPG 算法来解决 SFC 部署与计算资源分配联合优化问题. MADDPG 中每个智能体都有行动者和评论家 2 个网络. 行动者网络与单智能体行动者评论家算法一样. 评论家网络除了本智能体观察到的状态信息外, 还使用其他智能体的信息辅助训练. 而在测试过程中, 每个智能体只使用自己的行动者网络做动作选择, 不会引入其他智能体的信息.

2.2 算法设计

在 SFC 部署和计算资源分配模型中, 多智能体对应于各 SFC 中的 VNF, 它们通过与环境交互并采

取动作. M 个 VNF 组成的多智能体马尔可夫博弈表示为 $(S, A, r_1^t, \dots, r_M^t, p, \gamma)$, 其中, S 为状态空间, A 为动作空间, r_m^t 为第 m 个 VNF 在 t 时刻的回报函数, p 为转移概率, 当所有智能体在当前状态 $s_m^t \in S$ 下同时采取动作 $\{a_m^t \in A, 1 \leq m \leq M\}$, 并转移到一个新状态 $s_m^{t+1} \in S$, 可记为 $p(s_m^{t+1} | s_m^t, a_1^t, \dots, a_M^t)$. 在每个时刻 t , 各 SFC 中的 VNF 作为智能体观察状态空间 S 中的当前时刻状态 s^t , 接着基于策略 π , 从动作空间 A 中选择动作并执行, 即选择其部署的物理节点和计算资源分配值, 记为 a^t . 基于当前采取的动作, 状态会转移到一个新的状态 s^{t+1} , 环境反馈给智能体当前时刻的回报 r_m^t . 状态空间 S 、动作空间 A 、回报函数 r_m^t 的定义分别如下.

状态空间 根据各个物理节点的实时计算资源剩余情况为当前部署的 SFC 的各个 VNF 选择其部署的节点位置和计算资源分配值. 因此, SFC 的各个 VNF 所观察的环境状态为基础设施中所有物理节点的计算资源剩余量. 将第 n 个物理节点的计算资源剩余量记为 V_n , 某个时刻的状态可记为 $s_{k,m} = [V_1, V_2, \dots, V_N]$, 其中, N 为基础设施网络中物理节点的总数量.

动作空间 SFC 中的 VNF 需要选择物理节点和计算资源分配值, 因此每个时刻的可选动作由 2 部分组成. 由于基础设施网络中有 N 个物理节点, 动作集中的每个动作对应一个节点, 而可分配的計算资源量可离散化为 W 种可选级别, 因此动作集中的每个动作可记为 $a = \{n, \phi\}$, 其中, $n \in \{1, 2, 3, \dots, N\}$, $\phi \in \{\varphi_1, \varphi_2, \dots, \varphi_W\}$.

回报函数 对于深度强化学习, 学习过程由回报函数驱动. 每个智能体通过与环境的交互最大化其回报值. 当动作选择满足约束条件时, 将各个 VNF 的回报定义为其本身的成本和时延的加权值的负数值; 否则将回报值定义为一个负的极大值 r_{neg} . 因此, 第 m 个智能体在 t 时刻的回报函数可以表示为

$$r_m^t = \begin{cases} -\alpha C_{k,m} - \beta D_{k,m}, & \text{C1 - C4} \\ r_{\text{neg}}, & \text{otherwise} \end{cases} \quad (10)$$

其中 $C_{k,m}$, $D_{k,m}$ 分别为第 m 个 VNF 用于部署第 k 条 SFC 所产生的成本和时延.

每个智能体学习的目标是最大化自己的累积折扣回报:

$$R_m^t = \sum_{n=0}^T \gamma^n r_m^{t+n} \quad (11)$$

其中: T 为时间范围, $\gamma \in [0, 1)$ 为折扣因子.

对于具有 M 个智能体的马尔可夫博弈, 在确定性策略下, 用 $\mu = [\mu_1, \dots, \mu_M]$ 表示 M 个智能体的确定性策略向量, 则第 m 个智能体的动作—价值函数 (即 Critic) 可表示为

$$Q_m^{\mu}(s^t, a^t) = E_{s^{t+1}, r_m^t} \{ r_m^t + \gamma Q_m^{\mu}(s^{t+1}, \mu(s^{t+1})) \} \quad (12)$$

其中: $s^t = [s_1^t, \dots, s_M^t]$ 为所有智能体在 t 时刻的状态向量, $a^t = [a_1^t, \dots, a_M^t]$ 为所有智能体在 t 时刻的动作向量. 评论家网络可以使用 Q 学习中的贝尔曼方程学习, 使用贪婪策略 $\mu(s^t) = \arg \max Q(s^t, a^t)$. 将第 m 个智能体的动作—价值函数参数记为 θ_m^Q , 动作—价值函数 Q_m^{μ} 的更新策略为

$$L(\theta_m^Q) = E_{s^t, a^t, r_m^t, s^{t+1}} \{ [Q_m(s^t, a^t | \theta_m^Q) - y_m]^2 \} \quad (13)$$

其中,

$$y_m = r_m^t + Q_m[s^{t+1}, \mu(s^{t+1}) | \theta_m^Q] \quad (14)$$

根据确定性策略梯度 (DPG, deterministic policy gradient) 算法, 参数化的动作函数 $\mu(s^t | \theta^{\mu})$ 用来估计策略, 即将状态确定性地映射到特定的动作上. 利用 MADDPG 可以有效地解决 DPG 算法带来的高方差梯度估计问题. MADDPG 中的评论家网络不仅使用本智能体的动作作为网络输入, 也将其他智能体的动作作为输入, 它通过使用其他智能体的信息来评估自身选择动作的优劣, 消除了环境的不确定性. 如果所有 VNF 所采取的动作是已知的, 那么环境就是稳定的.

在 MADDPG 中, 将第 m 个智能体的确定性策略 μ_m 参数记为 θ_m^{μ} , 则第 m 个智能体的期望回报 $J_m = E[R_m^t]$ 的梯度为

$$\begin{aligned} \nabla_{\theta_m^{\mu}} J_m &\approx E_{s^t, a^t \sim G} [\nabla_{\theta_m^{\mu}} Q_m(s^t, a^t | \theta_m^Q) |_{a_m^t = \mu_m(s_m^t | \theta_m^{\mu})}] = \\ &E_{s^t, a^t \sim G} [\nabla_{a_m^t} Q_m(s^t, a^t | \theta_m^Q) |_{a_m^t = \mu_m(s_m^t)}] \nabla_{\theta_m^{\mu}} \mu_m(s_m^t | \theta_m^{\mu}) \end{aligned} \quad (15)$$

其中: G 为经验重放缓冲区, 每个样本以元组 (s^t, a^t, r^t, s^{t+1}) 的形式保存所有智能体的历史信息. 当所有的 VNF 都做出决策并执行后, 环境状态改变, 每个 VNF 都会观察到一个新状态 s^{t+1} 并获得上一时刻的回报 r^t .

在 MADDPG 中, 通过使用深度神经网络 (DNN, deep neural networks) 近似行动者网络的状态—动作映射和评论家网络的动作—价值函数避免高维空间引起的收敛问题. 行动者网络的输入为 VNF 观察

的状态,输出为其自身的动作选择;评论家网络的输入为所有 VNF 的状态和动作,输出为当前动作的 Q_m^μ . DNN 网络参数在训练中更新. 由于网络参数在更新的同时,还被用来计算目标值,这会带来一定的不稳定性,因此每个 VNF 的行动者网络和评论家网络将使用相同的 DNN 网络,即学习网络和目标网络,其中目标网络用来计算目标值 $\mu_m(s_m^t | \theta_m^\mu)$ 、 $Q_m(s^t, a^t | \theta_m^Q)$,其参数更新表示为

$$\theta^{\mu'} \leftarrow \tau \theta^{\mu'} + (1 - \tau) \theta^{\mu'}, \theta^{Q^{t+1}} \leftarrow \tau \theta^{Q^t} + (1 - \tau) \theta^{Q^{t+1}} \quad (16)$$

其中更新因子 $\tau \ll 1$.

MADDPG 算法在集中训练阶段通过与 SBA 基础网络环境交互,不断获得更新历史信息,即状态、动作与回报. 训练过程从经验重放缓冲区中随机抽取数据,用于其本身的策略更新. 由于经验重放缓冲区是一个有限的固定值,当缓冲区已满,最早的样本将被丢弃. MADDPG 是 off-policy 算法,算法每次迭代过程中,都会从缓冲区随机抽取小批量数据,从而达到抑制数据的时间相关性的目的. 在训练开始时,每个 VNF 的策略是随机初始化的,之后随着策略的更新学习能达到最大回报值的策略.

SFC 部署和计算资源分配算法的 MADDPG 训练阶段如下:

输入 虚拟网络参数、底层物理网络参数

1 随机初始化行动者和评论家网络参数向量 θ^μ 和 θ^Q

2 初始化目标行动者和目标评论家网络参数向量 θ^μ, θ^Q

3 初始化经验重放缓冲区 G

4 所有 VNF 观察初始状态 $s^0 = \{s_1^0, \dots, s_M^0\}$

5 for 每个时刻 t do

6 所有 VNF 基于当前策略选择动作向量 a^t

7 所有 VNF 执行动作向量 a^t , 获得回报向量 r^t , 观察下一状态向量 s^{t+1}

8 将元组 (s^t, a^t, r^t, s^{t+1}) 保存到经验重放缓冲区 G

9 随机在 G 中采样小批量数据

10 设置 $y_m = r_m^t + Q_m[s^{t+1}, \mu(s^{t+1}) | \theta_m^Q]$

11 根据式(12)更新评论家网络

12 根据式(15)更新行动者网络

13 更新目标网络参数向量

14 end for

输出 目标行动者网络参数向量 $\theta^{\mu'}$

在测试阶段,每个 VNF 只通过行动者网络来选

择动作,将环境状态输入行动者网络,输出为所有动作的 Q_m^μ ,选择最大 Q_m^μ 对应的动作并执行. SFC 部署和计算资源分配算法的 MADDPG 测试阶段步骤如下:

输入 虚拟网络参数、底层物理网络参数、目标行动者网络参数向量 $\theta^{\mu'}$

1 将训练好的参数向量 $\theta^{\mu'}$ 导入模型

2 所有 VNF 观察状态向量 $s^t = [s_1^t, \dots, s_M^t]$

3 所有 VNF 根据 $a^t = [a_m^t = \mu_m^t(s_m^t), m \in M]$ 物理节点和计算资源量

4 所有 VNF 执行动作 a^t , 计算系统回报

5 返回测试结果

2.3 复杂度分析

MADDPG 算法复杂度需要考虑行动者网络和批评家网络 2 部分的复杂度^[11]. 定义行动者网络中第 j 层神经元的个数为 U_j , 行动者网络的计算复杂度为 $O\left(\sum_{j=2}^{J-1} (U_{j-1}U_j + U_jU_{j+1})\right)$, 其中 J 为行动者网络的层数. 定义批评家网络中第 h 层神经元的个数为 U_h , 批评家网络的计算复杂度为

$O\left(\sum_{h=2}^{H-1} (U_{h-1}U_h + U_hU_{h+1})\right)$, 其中 H 为批评家网络的层数. 则 MADDPG 训练阶段的算法复杂度为 $O\left(\sum_{j=2}^{J-1} (U_{j-1}U_j + U_jU_{j+1}) + \sum_{h=2}^{H-1} (U_{h-1}U_h + U_hU_{h+1})\right)$; 在测试阶段的算法复杂度为

$O\left(\sum_{l=2}^{L-1} (U_{l-1}U_l + U_lU_{l+1})\right)$, 其中 L 为行动者网络层数, U_l 为第 l 层神经元的个数.

在所提算法的实际部署中,网络控制器获得不同物理节点和 SFC 的参数信息,通过离线的方式实现算法的训练与测试过程.

3 仿真结果分析

针对所提算法进行性能仿真验证. 在仿真中,将基础设施网络中的各类物理资源统一为物理节点的计算资源个数. 此外,将虚拟网络中 SFC 的各类型 VNF 设置为不同的计算资源需求和时延性能. 仿真中物理网络、虚拟网络及其他相关仿真参数如表 1 所示. 仿真采用 PyCharm 平台开发,性能验证过程中采用蒙特卡洛方法,物理节点拓扑随机产生,仿真性能结果通过 100 次随机后取平均值.

表 1 仿真参数表

参数	数值
物理节点数量/个	9
初始物理节点计算资源/个	30
物理节点计算资源单价	0.6~2.0(均匀分布)
物理链路传输时延/ms	0.1~0.5(均匀分布)
VNF 类型数量/个	8
SFC 中 VNF 节点数/个	2~4
VNF 最少计算资源需求/个	1.0~2.0(均匀分布)
VNF 最多计算资源需求/个	3.0~4.0(均匀分布)
VNF 最小处理时延/ms	0.4~0.8(均匀分布)
VNF 最大处理时延/ms	1.6~2.4(均匀分布)
SFC 最大容忍时延/ms	7.5
权重系数 α, β	1, 1
计算资源离散化级别	3
更新因子 τ	0.01

首先验证所提算法的收敛性. 图 2 所示为当 SFC 条数为 3 时, 系统总回报关于迭代次数的收敛情况. 由图 2 可知, 收敛前期波动比较大, 这是因为所提算法将所有 VNF 的全局信息引入来做集中式训练, 这需要利用所有 VNF 之间的充分合作来达到全局优化的目标. 但可以看到收敛后会维持在一个稳定的值, 这得益于 MADDPG 引入全局信息来辅助训练, 使得算法收敛后稳定性好.

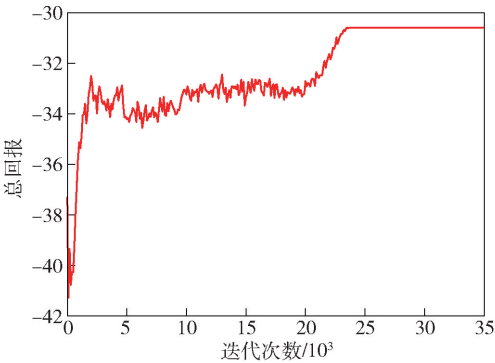


图 2 总回报收敛曲线

为了进一步验证所提算法性能, 采用以下 2 种典型算法为对比算法. ① 最小计算资源单价物理节点选择和最大计算资源分配 (MCMR, minimum cost and maximum resource): 各 SFC 中 VNF 根据最小计算资源单价选择服务器节点并选择最大的计算资源分配量. ② 随机物理节点选择和计算资源分配 (RD, random) 算法: 各 SFC 中 VNF 随机选择其映

射的服务器节点和计算资源分配量. 图 3 所示为不同算法的系统成本与时延加权值的对比结果. 结果表明, 所提 MADDPG 算法可以达到成本和时延加权值的最小化, 并且 SFC 数量越多, 所提算法在此方面的优越性越明显. MCMR 算法虽然优先选择最小单价的物理节点, 并通过最大计算资源分配量尽可能降低在此节点上的处理时延, 但不能很好地实现两者的性能折中.

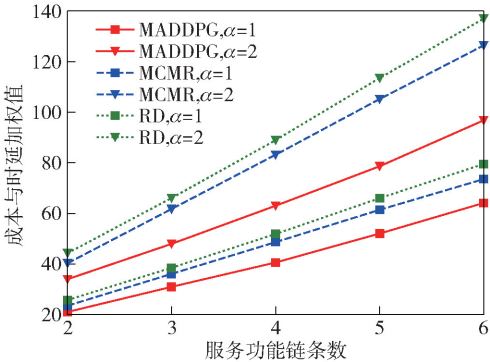


图 3 SFC 条数变化下的成本与时延加权值对比

图 4 所示为不同算法的 SFC 部署总成本对比结果. 由于对比算法不受优化目标权重系数影响, 因此无论权重系数取值为多少, MCMR 算法和 RD 算法在系统总成本方面的表现是相同的. 由图 4 可知, MADDPG 算法在降低系统总成本方面的性能最好. 这是因为 MCMR 算法虽然采取了计算资源单价优先的物理节点选择, 但由于在每个物理节点上分配最大的可选计算资源量, 从而导致了总成本的增加. RD 算法由于其选择物理节点及计算资源量的随机特性, 在降低系统总成本方面的性能表现最差. 所提算法通过物理节点和计算资源量的联合最佳选择, 有效地降低了系统总成本. 此外, 当权重系数 α 为 2 时, 即提高了系统总成本的优化权重占比, 因此所提算法进一步降低了系统总成本.

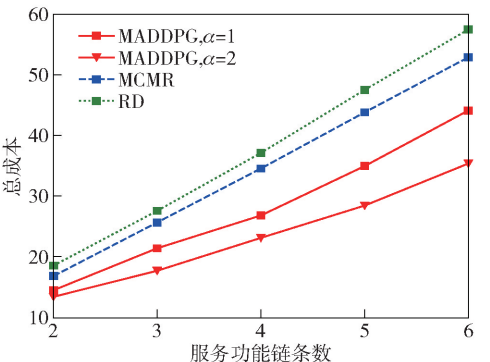


图 4 SFC 条数变化下的系统总成本对比

图 5 所示为不同算法的 SFC 总时延对比结果。当权重系数 α 为 1 时, MADDPG 算法保证了系统中所有 SFC 总时延的最小化。MCMR 算法通过在服务器节点上分配最大的计算资源量尽可能地降低在此节点上的处理时延, 但由于它倾向于选择计算资源单价较小的节点, 这意味着 MCMR 算法所选择的服务器节点处理能力较差, 从而导致了总时延的增加, 因此 MCMR 算法次于 MADDPG 算法。RD 算法在降低所有 SFC 总时延方面仍然表现最差。当降低系统总时延的权重占比, 如当权重系数 α 为 2 时, 使用 MADDPG 算法的系统中所有 SFC 的总时延会增加。

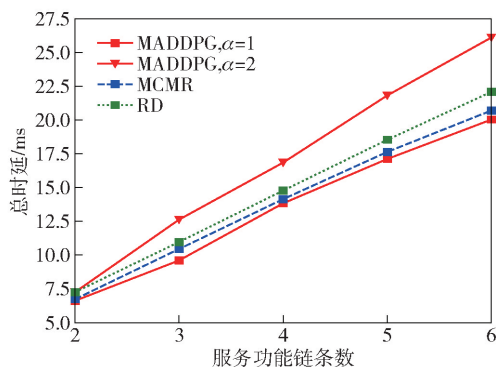


图 5 SFC 条数变化下的系统总时延对比

4 结束语

在移动核心网架构下提出了一种 SFC 部署与计算资源分配联合算法。考虑计算资源分配量与 VNF 处理时延的线性依赖关系, 建立了最小化系统成本和时延加权和的优化问题, 提出了 SFC 部署与计算资源分配联合算法。仿真结果表明, 所提算法可以更加有效地实现系统总成本和总时延两者的性能折中。

参考文献:

- [1] Goyal R, Mahajan N, Goyal T, et al. Exploration of 5G technology for cellular communication: a survey [C] // 2018 Second International Conference on Intelligent Computing and Control Systems. New York: IEEE, 2018: 330-334.
- [2] Lu Jiegang, Xiao Limin, Tian Zhigang, et al. 5G Enhanced service-based core design [C] // 2019 28th Wireless and Optical Communications Conference (WOCC). New York: IEEE, 2019: 1-5.
- [3] Medhat A M, Taleb T, Elmangoush A, et al. Service function chaining in next generation networks: state of the art and research challenges [J]. IEEE Communications Magazine, 2016, 55(2): 216-223.
- [4] Nejad M A T, Parsaeefard S, Maddah-Ali M A, et al. vSPACE: VNF simultaneous placement, admission control and embedding [J]. IEEE Journal on Selected Areas in Communications, 2018, 36(3): 542-557.
- [5] Alleg A, Ahmed T, Mosbah M, et al. Delay-aware VNF placement and chaining based on a flexible resource allocation approach [C] // 2017 13th International Conference on Network and Service Management (CNSM). New York: IEEE, 2017: 1-7.
- [6] 魏亮, 黄韬, 张娇, 等. 基于强化学习的服务链映射算法 [J]. 通信学报, 2018, 39(1): 90-100.
Wei Liang, Huang Tao, Zhang Jiao, et al. Service chain mapping algorithm based on reinforcement learning [J]. Journal on Communications, 2018, 39(1): 90-100.
- [7] 高鹏, 胡晓东, 李家兴, 等. 5G-C-RAN 中最大化效用 SFC 部署算法 [J]. 计算机工程与应用, 2019, 55(7): 108-114, 206.
Gao Peng, Hu Xiaodong, Li Jiaxing, et al. Service function chain deployment algorithm based on network utility maximization in 5G-C-RAN [J]. Computer Engineering and Applications, 2019, 55(7): 108-114, 206.
- [8] Zhou Liushan, Zhang Tiankui, Li Jing, et al. Radio resource allocation for RAN slicing in mobile networks [C] // IEEE. IEEE/CIC International Conference on Communications in China. Chongqing: IEEE, 2020: 1280-1285.
- [9] Tong Zhou, Zhang Tiankui, Zhu Yutao, et al. Communication and computation resource allocation for end-to-end slicing in mobile networks [C] // IEEE. IEEE/CIC International Conference on Communications in China. Chongqing: IEEE, 2020: 1286-1291.
- [10] Amdahl G M. Validity of the single processor approach to achieving large scale computing capabilities [C] // Proc AFIPS Spring Joint Computer Conference. Reston: AFIPS Press, 1967: 483-485.
- [11] Li Zheng, Guo Caili. Multi-agent deep reinforcement learning based spectrum allocation for D2D underlay communications [J]. IEEE Transactions on Vehicular Technology, 2020, 69(2): 1828-1840.