移动边缘计算中基于图到序列深度强化学习的复杂任务部署策略

陈卓[1]，操民涛[2]，周致圆[3]，黄欣[4]，李彦[2]

（1. 重庆理工大学计算机科学与工程学院，重庆 400054；2. 重庆理工大学两江人工智能学院，重庆 401135；
3. 重庆凯瑞机器人技术有限公司，重庆 400799；4. 中国移动通信集团重庆有限公司，重庆 401120）

摘 要：借助于移动边缘计算（MEC）和网络虚拟化技术，可使移动端将执行各类复杂应用所需的算力、存储和传输等资源需求就近卸载至边缘服务节点，从而获得更高效的服务体验。面向边缘服务商，研究其在进行复杂任务部署时所面临的能耗优化决策问题。首先将复杂任务部署于多个边缘服务节点的问题建模为混合整数规划（MIP）模型，然后提出了一种融合图到序列的深度强化学习（DRL）求解策略。该策略通过基于图的编码器设计提取并学习子任务间潜在的依赖关系，从而根据边缘服务节点的可用资源状态及使用率自动发现任务部署的通用模式，最终快速获得能耗优化的部署策略。在不同的网络规模中，将所提策略与具代表性的基准策略进行了全面对比。实验结果表明，所提策略在任务部署错误率、MEC 系统总功耗和算法求解效率等方面均显著优于基准策略。

关键词：移动边缘计算；任务部署；深度强化学习；图神经网络

中图分类号：TP393.0

文献标志码：A

DOI: 10.11959/j.issn.1000−436x.2024058

Graph-to-sequence deep reinforcement learning based complex task deployment strategy in MEC

CHEN Zhuo[1], CAO Mintao[2], ZHOU Zhiyuan[3], HUANG Xin[4], LI Yan[2]

1. College of Computer Science and Engineering, Chongqing University of Technology, Chongqing 400054, China
2. School of Artificial Intelligence, Chongqing University of Technology, Chongqing 401135, China
3. Chongqing CAERI Robot Technology Co., Ltd., Chongqing 400799, China
4. China Mobile Communications Corporation Chongqing Co., Ltd., Chongqing 401120, China

Abstract: With the help of mobile edge computing (MEC) and network virtualization technology, the mobile terminals can offload the computing, storage, transmission and other resource required for executing various complex applications to the edge service nodes nearby, so as to obtain more efficient service experience. For edge service providers, the optimal energy consumption decision-making problem when deploying complex tasks was comprehensively investigated. Firstly, the problem of deploying complex tasks to multiple edge service nodes was modeled as a mixed integer programming (MIP) model, and then a deep reinforcement learning (DRL) solution strategy that integrated graph to sequence was proposed. Potential dependencies between multiple subtasks through a graph-based encoder design were extracted and learned, thereby automatically discovering common patterns of task deployment based on the available resource status and utilization rate of edge service nodes, and ultimately quickly obtaining the deployment strategy with the optimal energy consumption. Compared with representative benchmark strategies in different network scales, the experimental results show that the proposed strategy is significantly superior to the benchmark strategies in terms of task deployment error ratio, total power consumption of MEC system, and algorithm solving efficiency.

Keywords: mobile edge computing, task deployment, deep reinforcement learning, graph neural network

---

收稿日期：2023−08−30；**修回日期：**2023−11−01

基金项目：国家自然科学基金资助项目（No.62071077, No.61671096）

Foundation Items: The National Natural Science Foundation of China (No.62071077, No.61671096)

0　引言

随着 5G 以及物联网的快速部署，各类网络终端设备的规模持续激增并由此涌现出丰富多样的移动业务场景和应用。受限于移动应用的服务质量、数据安全与隐私、云端网络带宽资源瓶颈等，基于云计算的移动应用服务模式很难满足用户的服务需求[1]。为应对这一挑战，以移动边缘计算（MEC, mobile edge computing）为代表的近端计算范式应运而生，其目的是在移动网络边缘、无线接入网内以及移动用户附近提供 IT 服务环境[1-3]，在为终端用户提供强大的计算能力、能源效率和存储容量的同时具备低时延和高带宽的服务特点，从而提高用户体验质量，使系统和业务的运营更具成本效益和竞争力[4-5]。在基于 MEC 的框架中，无线终端设备可以将 IT 资源敏感型任务（如多模态智能计算应用、算力敏感的区块链应用、基于数字孪生的虚实交互类应用等）卸载到边缘服务节点，从而实现更好的业务体验[6]。从边缘服务商的角度而言，在接收了用户的任务卸载请求之后，需要根据任务对资源类型、请求规模以及边缘区域中的可用资源状况进行合理的部署决策，进而选择适合的边缘服务节点执行任务。目前，学术界已经就移动任务在边云端的卸载问题进行了研究[7-11]。与此同时，为了填补端与中心之间的算力真空和支持各种泛在应用，边缘服务节点被越来越广泛地部署，其中部分部署位置甚至难以通过电网直接供电。因此，如何在保证边缘服务质量的前提下有效控制能耗已成为亟待研究解决的问题。

为应对 MEC 在服务复杂任务过程中的能耗挑战，面向边缘服务商，本文研究在多个边缘服务节点中进行优化的任务部署。首先从逻辑上将复杂任务分解为多个具有相互依赖关系的子任务[12]，并在满足任务执行所需的计算、存储和通信等 IT 资源以及边缘服务节点的资源约束前提下通过建模加以分析，然后将多个子任务之间的潜在关系视作图结构并利用图神经网络进行特征提取，再结合强化学习在自学习探索和实时决策能力强等方面所具有的优势设计求解策略，最后通过对比实验进行全面评估验证。本文的主要贡献如下。

1) 面向复杂任务在边缘端的能耗优化部署问题，本文以边缘服务节点的能耗优化为目标，建立了一个混合整数规划（MIP, mixed integer programming）模型，不仅把多维度 IT 资源的规模纳入考虑，还将 IT 资源的使用率和能耗之间的关系纳入建模分析。该模型更全面地描述了在资源受限情况下任务的部署决策对能耗开销的影响。

2) 本文提出了一种基于图到序列强化学习的求解策略，通过智能体与任务及边缘服务节点的持续交互进行学习并自动寻找部署方案。特别地，智能体由基于图神经网络的编码器和循环神经网络的解码器组成，能够在对任务之间潜在图关联结构进行特征提取的基础上完成部署决策序列的解码输出，实现了复杂任务部署决策的求解质量和求解效率之间的平衡。

3) 本文设计了详细的实验方案对所提求解策略的有效性进行评估，将所提策略与近年来具有代表性的工作，如神经组合优化（NCO, neural combinatorial optimization）算法[13-14]、首次适应（FF, first-fit）算法[15-16]、Gurobi 求解器[17]以及混合智能算法（HIA, hybrid intelligent algorithm）[18-19]进行多个指标的全面对比。结果表明，在相同的实验环境参数下，所提策略在求解有效性、求解质量和平均求解时间等方面均优于其他对比策略。

1　相关工作

在 MEC 环境中，边缘网络的拓扑结构和计算能力存在较大差异，边缘服务节点所能提供的资源相对有限。因此，MEC 环境下的调度问题具有特殊性，主要以能耗和时延为优化目标。针对 MEC 已接收的卸载任务，根据可用资源的情况可以分为单节点部署和多节点部署两大类。对于将任务分配到单节点的场景，Sun 等[7]针对多用户单边缘服务节点，引入了一个名为计算效率的指标，并提出一个结合本地计算和数据卸载的联合计算算法。优化问题的目的是使用户之间的计算效率加权和最大，然后使用迭代和梯度下降方法进行求解。Yang 等[8]研究了针对单一节点的多用户非正交多址 MEC 系统中的卸载时延最小化问题，通过采用深度 Q 网络强化学习算法，在不事先了解其他用户行动的情况下确定最佳用户组合状态，从而显著降低系统卸载时延。更进一步地，Wang 等[9]用有向无环图结构表示复杂任务，提出了一个基于深度强化学习（DRL）的 MEC 系统的卸载框架，通过使用循环神经网络来解决动态场景下的任务卸载。Li 等[10]在多用户多节点的场景下，将计算卸载和资源分配的联合优化

问题表述为一个混合整数非线性优化模型，并提出了基于遗传算法的两阶段优化算法以获得优化解。Cui 等[11]提出了一种改进的细粒度调度算法，对任务的执行位置和调度顺序进行优化决策从而减少服务时延。已有工作尚缺乏对复杂任务在边缘端进行优化的部署决策问题的探讨。

在移动端涌现出越来越多的智能化应用，这些任务的执行通常需要依赖多项服务的响应。因此可以从逻辑上将复杂任务划分为多个相关联的子任务并在边缘或云端节点上部署执行。Hu 等[12]将智能网联车产生的时间敏感任务视作相互关联的多个子任务，并提出了边缘虚拟服务单元的联合调度策略以提供低时延服务。Laroui 等[20]针对多个子任务对服务质量的差异化需求，根据边缘端的可用资源状况进行适应性的资源分配。Chen 等[21]以有向无环图结构来表示子任务之间的关联关系，并在任务卸载至边云协作的服务节点过程中将子任务之间的依赖关系纳入考虑以缩短任务的服务时间。可以看到，上述工作还没有对复杂任务的部署与边缘端的能耗控制之间的关系进行深入探讨。

目前，基于深度学习方法来求解组合优化问题已经取得了一些初步的结果[15,22-24]，但由于其主要沿用了监督学习的模式，样本的获取限制了求解问题的规模。而强化学习通过自学习的方式，让智能体不断地与环境进行交互并执行反馈动作从而搜索优化解，这使该方法所学习到的策略能够充分利用问题的结构从而获得质量更好的优化解。Liu 等[23]通过建立一个基于神经组合优化的服务框架，在具有隐私保护的条件下有效改进了旅行推销员问题的求解效率和质量。Jiang 等[24]针对多节点环境下的多用户服务问题进行建模，并基于指针网络架构进行求解。为改善边缘系统的服务能力，Li 等[15]提出需要对任务卸载、任务部署、边缘缓存和资源分配进行联合优化，并提出了基于深度 Q 学习的边缘缓存和任务部署求解策略。但对于边缘服务商在满足移动任务服务质量的同时，如何通过高效部署决策改善系统能耗的开销仍需深入研究。

2 系统框架与建模

2.1 场景描述及形式化定义

本文研究的 MEC 环境下的复杂任务部署场景如图 1 所示。多个移动设备（如智能终端、智能车或者工业现场设备等）通过物联网网关或基站就近连接多个资源异构的边缘服务节点。移动设备上运行不同的复杂任务。复杂任务可以从逻辑上分解为多个相互关联的子任务，而子任务之间的关系可以是一般化的图结构。边缘服务商通过虚拟化技术的引入，使边缘服务节点可以被实例化为多个虚拟节点（VN, virtualized node），如虚拟机（VM, virtual machine）或轻量化容器，这也进一步使边缘服务商可以灵活地决策任务的部署。边缘服务商接收到来自移动设备的任务卸载请求后，根据子任务执行所需资源、子任务之间的逻辑连接关系以及边缘服务节点当前可用的计算、内存和磁盘等 IT 资源，通过虚拟架构管理器（VIM, virtual infrastructure manager）[25-26]的控制系统实现对子任务在 VN 上的映射部署。本文以边缘服务节点总能耗最小化为优化目标，同时将 VN 中的网络链路带宽和时延约束纳入考虑。

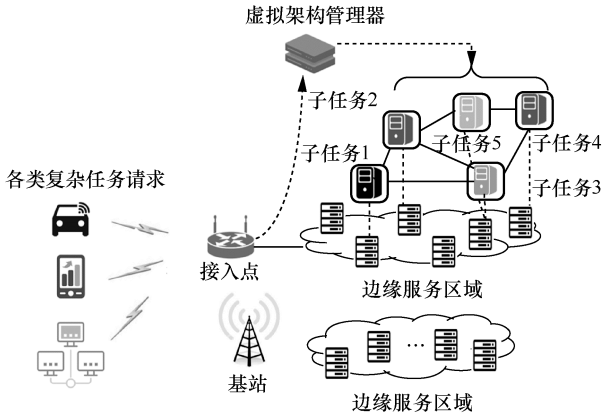


图 1 MEC 环境下的复杂任务部署场景

为了详细描述该问题，将移动设备的复杂任务表示为一个图结构 $\mathcal{G}=(\mathcal{V},\mathcal{E})$。其中，顶点集合 $\mathcal{V}$ 由 M 个子任务组成，定义为 $\mathcal{V}=\{1,2,\cdots,M\}$，每一个子任务 $v\in\mathcal{V}$ 可以用一个三元组 $v=(r_v^{\text{cpu}},r_v^{\text{mem}},r_v^{\text{disk}})$ 表示，r_v^{cpu}、r_v^{mem}、r_v^{disk} 分别代表子任务 v 执行时所需的计算资源、内存资源和磁盘资源；$\mathcal{E}$ 表示边集合，定义为 $\mathcal{E}=\{e(v,v')\,|\,v,v'\in\mathcal{V}\}$，$e(v,v')$ 表示子任务 v 和子任务 v' 之间存在依赖关系。MEC 网络由 N 个边缘服务节点组成，定义为 $\mathcal{H}=\{1,2,\cdots,N\}$，其中每个边缘服务节点 $h\in\mathcal{H}$ 可以用一个三元组 $h=(a_h^{\text{cpu}},a_h^{\text{mem}},a_h^{\text{disk}})$ 表示，a_h^{cpu}、a_h^{mem}、a_h^{disk} 分别代表边缘服务节点 h 可以提供的计算资源、内存资源和磁盘资源。因此本文的复杂任务部署问题可转

化为寻找一个优化的任务部署集合 $\mathcal{F}\in\{0,1\}^{M\times N}$，其中 $f_{vh}\in\mathcal{F}$ 表示子任务 v 是否成功部署到节点 h 上（1 表示成功部署，0 表示未成功部署）。

在边缘服务节点能耗方面，已有工作只考虑节点能耗与中央处理器（CPU, central processing unit）的利用率呈线性关系，不同的是本文从多个维度考虑节点的能耗，将内存和磁盘也纳入其中。为此，本文定义了节点负载率 F_h，为边缘服务节点中各个组件赋予了不同的权重比例，然后通过加权求和的方式得到每个节点的负载率，如式(1)所示。

$$F_h=\omega_1U_h^{\text{cpu}}+\omega_2U_h^{\text{mem}}+\omega_3U_h^{\text{disk}} \tag{1}$$

其中，U_h^{cpu}、U_h^{mem}、U_h^{disk} 分别代表边缘服务节点 h 的计算、内存、磁盘的利用率，ω_1、ω_2、ω_3 分别代表计算、内存、磁盘在边缘服务节点中的能耗占比且其和为 1。利用率定义式为

$$\begin{cases}U_h^{\text{cpu}}=\dfrac{\sum\limits_{v\in\mathcal{V}}r_v^{\text{cpu}}f_{vh}}{a_h^{\text{cpu}}}\\ U_h^{\text{mem}}=\dfrac{\sum\limits_{v\in\mathcal{V}}r_v^{\text{mem}}f_{vh}}{a_h^{\text{mem}}}\\ U_h^{\text{disk}}=\dfrac{\sum\limits_{v\in\mathcal{V}}r_v^{\text{disk}}f_{vh}}{a_h^{\text{disk}}}\end{cases} \tag{2}$$

本文将边缘服务节点满负荷时的能耗定义为 $E_h^{\max}$，边缘服务节点空闲时的能耗定义为 E_h^{idle}，可以得到所有边缘服务节点执行子任务时的总能耗 $E_{\mathcal{H}}$ 为

$$E_{\mathcal{H}}=\sum_{h\in\mathcal{H}}[E_h^{\text{idle}}+(E_h^{\max}-E_h^{\text{idle}})F_h] \tag{3}$$

此外，本文考虑移动设备与边缘服务节点之间的通信链路资源。假设移动设备中复杂任务的某一子任务部署到边缘服务节点上，则该移动设备和边缘服务节点之间存在一条通信链路 l（$1\leqslant l\leqslant N$），所有链路组成的集合定义为 L。

2.2　模型建立

本文面向复杂任务在多个边缘服务节点的能耗优化部署进行建模分析。该模型充分考虑了诸如计算（以边缘服务节点的 CPU 资源表示）、内存、磁盘资源等硬件约束，以及通信过程中的带宽和时延约束。模型目标是通过找到复杂任务所对应的多个子任务的部署策略，使 MEC 系统在服务过程中产生的总能耗最优。结合 2.1 节中的相关定义，目标函数及相关约束条件为

$$\begin{aligned}&\min_{\mathcal{F}}\sum_{h\in\mathcal{H}}\left[E_h^{\text{idle}}+\left(E_h^{\max}-E_h^{\text{idle}}\right)F_h\right]\\ \text{s.t.}\quad&\text{C}_1:\sum_{h\in\mathcal{H}}f_{vh}=1,\quad\forall v\in\mathcal{V}\\ &\text{C}_2:\sum_{v\in\mathcal{V}}r_v^{\text{cpu}}f_{vh}\leqslant a_h^{\text{cpu}},\quad\forall h\in\mathcal{H}\\ &\text{C}_3:\sum_{v\in\mathcal{V}}r_v^{\text{mem}}f_{vh}\leqslant a_h^{\text{mem}},\quad\forall h\in\mathcal{H}\\ &\text{C}_4:\sum_{v\in\mathcal{V}}r_v^{\text{disk}}f_{vh}\leqslant a_h^{\text{disk}},\quad\forall h\in\mathcal{H}\\ &\text{C}_5:\sum_{v\in\mathcal{V}}b_l^{v}f_{\text{vh}}\leqslant b_l,\quad\forall l\in L\\ &\text{C}_6:\sum_{l\in L}\sum_{v\in\mathcal{V}}d_l^{v}f_{vh}+\sum_{h\in\mathcal{H}}\sum_{v\in\mathcal{V}}d_h^{v}f_{vh}\leqslant d_{\mathcal{V}},\\ &\quad\forall l\in L,\forall h\in\mathcal{H}\end{aligned} \tag{4}$$

其中，约束条件 C_1 表明对于复杂任务 $\mathcal{V}$ 中的任意子任务 v，在同一时间只能在一个边缘服务节点上部署执行。不等式约束条件 $\text{C}_2\sim\text{C}_4$ 分别表示所有子任务在某一个边缘服务节点上使用的 CPU、内存和磁盘资源不能超过该边缘服务节点的可用资源。约束条件 C_5 表明每条链路中每个子任务所需的带宽量 b_l^v 之和不会超过该链路的最大带宽 b_l。约束条件 C_6 考虑了一个时延模型，复杂任务部署问题中的时延包括子任务传输到边缘服务节点的链路时延 d_l^v 和子任务在边缘服务节点上的执行时延 d_h^v，其时延之和不能超过服务复杂任务所允许的最大时延 $d_{\mathcal{V}}$。

本文进一步分析式(4)中定义的优化问题的 NP 性。具体来说，每个子任务被视作一个虚拟的“物品”，其所需的资源，包括 CPU、内存、磁盘资源，以及带宽和时延，对应于物品不同维度的属性。同时，边缘服务节点被视为具有有限资源能力的“容器”。而对于子任务在边缘服务节点上的优化部署问题，实际就是将“物品”优化放置于受限的“容器”之中，这可以被视作一个多维装箱问题（MDBPP, multi-dimensional bin packing problem）。式(4)所描述的优化目标则可以描述为在不超过任何“容器”不同维度容量的条件下，将所有需打包的“物品”放置到最少的“容器”中。由于 MDBPP 是一个 NP-hard 问题[27]，因此式(4)所描述的优化问题继承了 MDBPP 的 NP-hard 属性，难以在多项式时间内找到全局最优解。为

应对这一困难并寻求一种高效的解决方案，本文提出了一种基于 DRL 的求解策略。

3 融合图到序列的深度强化学习求解策略

本文设计了一个新的 DRL 框架来寻求 MEC 环境中复杂任务的部署决策解。该框架通过将图神经网络与强化学习相结合，运用神经网络表示策略和值函数，从而实现高维度且复杂问题的有效处理。为了实现多个子任务在多个边缘服务节点上的部署，提取和分析子任务之间的复杂依赖关系尤其重要，而这种关联关系可视作一种图结构。另外，图神经网络能够从非欧几里得数据中高效提取特征，适用于完成节点分类、图分类或关系预测等任务[28]。基于图神经网络在处理图结构数据方面的独特优势，本文采用图卷积提取子任务之间的依赖关系特征，使求解框架能够在不违反约束条件的前提下更准确地给出部署策略。同时，由于子任务的部署策略本质上是一个部署决策序列，因此本文基于编码–解码的基本思想构建了一个图到序列的解输出模型。图 2 描述了本文提出的融合了图到序列模型求解策略（DRL-G2S）的框架和主要组成部分。

3.1 图到序列模型

本节首先描述图 2 中智能体的设计。智能体采用一个图到序列结构的神经网络模型，它主要由图编码器和循环神经网络（RNN）解码器两部分组成。在这个神经网络模型中，一开始就给出固定的子任务依赖关系图作为模型输入进行训练是不合理的。这是因为在复杂的 MEC 环境中，子任务之间的依赖关系可能并非固定不变的。相反，它们可能需要根据任务的实际情况进行动态调整。因此，在图编码器中采用一种动态图构造方法从子任务序列中构建依赖关系，以确保神经网络模型能够适应不断变化的 MEC 环境，从而提高任务执行的效率并满足时延要求。在这种方法中，任务序列 $X=\{x_1,x_2,\cdots,x_M\}$ 经过嵌入层得到一个任务嵌入矩阵 $\boldsymbol{E}\in\mathbb{R}^{D\times M}$，其中，$M$ 表示复杂任务中的子任务数量，D 表示嵌入向量的维度。然后对该任务嵌入矩阵运用自注意力机制计算出子任务稠密邻接矩阵 $\boldsymbol{A}\in\mathbb{R}^{M\times M}$，如式(5)所示。

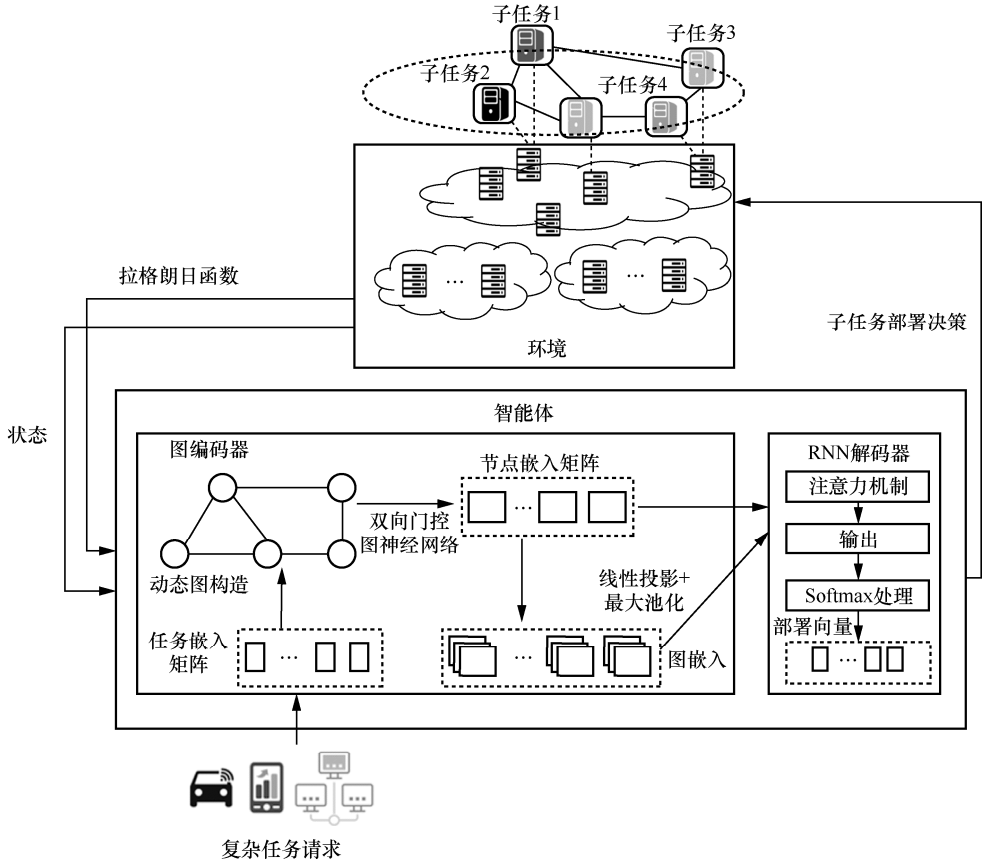


图 2 DRL-G2S 的框架和组成部分

$$\boldsymbol{A}=\mathrm{ReLU}(\boldsymbol{W}\boldsymbol{E})^{\mathrm{T}}\,\mathrm{ReLU}(\boldsymbol{W}\boldsymbol{E}) \tag{5}$$

其中，$\boldsymbol{W}\in\mathbb{R}^{F\times D}$ 是一个可训练的权重矩阵，F 是隐藏层的维度，ReLU 是一个常用的激活函数。

使用 K 最近邻（KNN, K-nearest neighbor）法的思想对稠密邻接矩阵 $\boldsymbol{A}$ 进行稀疏化处理，即每个子任务节点只保留与其依赖关系最强的 K 个子任务及注意力分数（依赖关系较弱的会被掩码处理），从而得到一个如式(6)所示的稀疏邻接矩阵 $\overline{\boldsymbol{A}}$。

$$\overline{\boldsymbol{A}}=\mathrm{KNN}(\boldsymbol{A},K) \tag{6}$$

受双向循环神经网络（BiRNN, bidirectional recurrent neural network）[29]的启发，通过对所得到的稀疏邻接矩阵 $\overline{\boldsymbol{A}}$ 及它的转置 $\overline{\boldsymbol{A}}^{\mathrm{T}}$ 分别进行 Softmax 运算，并根据它们的传入和传出方向计算出 2 个归一化邻接矩阵 $\boldsymbol{A}^{\dashv}$ 和 $\boldsymbol{A}^{\vdash}$，如式(7)所示。

$$\boldsymbol{A}^{\dashv},\boldsymbol{A}^{\vdash}=\mathrm{Softmax}\left(\left\{\overline{\boldsymbol{A}},\overline{\boldsymbol{A}}^{\mathrm{T}}\right\}\right) \tag{7}$$

其中，Softmax 函数是归一化指数函数，可将一组输入值（通常称为 logits）映射到一个概率分布并使所有输出值的和为 1。

在构造好任务图之后，本文基于双向门控图神经网络（BiGGNN）来处理任务图，以交错的方式从输入边和输出边学习子任务节点的特征表示。与传统的图神经网络（如图卷积网络和图注意力网络）相比，BiGGNN 并不依赖于卷积运算，而是利用循环神经网络实现节点表示向量的更新。通过在节点之间沿有向边进行信息传递和状态更新，可以更好地捕捉图结构中的复杂关系和依赖。另外，本文设计了双向门控递归单元，从任务嵌入矩阵 $\boldsymbol{E}$ 中提取任务序列的所有子任务节点信息作为初始节点嵌入 $\boldsymbol{h}^{(0)}$，其中每个子任务的节点嵌入为 $\boldsymbol{h}_v$。在每一跳 k，对于图中的每个节点 v 应用一个聚合函数，该函数将一组传入（或传出）的邻近节点 v' 的权重向量 $\boldsymbol{a}_{vv'}^{\dashv}$（或 $\boldsymbol{a}_{vv'}^{\vdash}$）作为输入，并输出一个向后（或向前）的聚合向量 $\boldsymbol{h}_{\mathcal{N}_{\dashv(v)}}^{(k)}$（或 $\boldsymbol{h}_{\mathcal{N}_{\vdash(v)}}^{(k)}$）。然后计算聚合的加权平均值，其中权重向量来自归一化后的邻接矩阵，其定义如式(8)所示。

$$\boldsymbol{h}_{\mathcal{N}_{\dashv(v)}}^{(k)}=\sum_{\forall v'\in\mathcal{N}_{\dashv(v)}}\boldsymbol{a}_{vv'}^{\dashv}\boldsymbol{h}_{v'}^{(k-1)},\quad \boldsymbol{h}_{\mathcal{N}_{\vdash(v)}}^{(k)}=\sum_{\forall v'\in\mathcal{N}_{\vdash(v)}}\boldsymbol{a}_{vv'}^{\vdash}\boldsymbol{h}_{v'}^{(k-1)} \tag{8}$$

本文选择在每一跳 k 融合 2 个方向上的聚合信息。如式(9)所示，Fuse 是一个自定义的融合函数，其含义为 2 个信息源的门控和。而 Fuse 函数具体如式(10)所示。

$$\boldsymbol{h}_{\mathcal{N}_{(v)}}^{(k)}=\mathrm{Fuse}\left(\boldsymbol{h}_{\mathcal{N}_{\dashv(v)}}^{(k)},\boldsymbol{h}_{\mathcal{N}_{\vdash(v)}}^{(k)}\right) \tag{9}$$

$$\mathrm{Fuse}\,(\boldsymbol{a},\boldsymbol{b})=\boldsymbol{z}\odot\boldsymbol{a}+(1-\boldsymbol{z})\odot\boldsymbol{b},\boldsymbol{z}=\sigma\left(\boldsymbol{W}_z[\boldsymbol{a};\boldsymbol{b};\boldsymbol{a}\odot\boldsymbol{b};\boldsymbol{a}-\boldsymbol{b}]+\boldsymbol{b}_z\right) \tag{10}$$

其中，$\odot$ 代表逐元素相乘，σ 代表 Sigmoid 函数，$\boldsymbol{z}$ 是门控向量。最后使用 GRU，通过合并聚合信息来更新节点嵌入。经过 k 跳的计算得到节点的最终状态嵌入 $\boldsymbol{h}_v^{(k)}$，其表达式如式(11)所示。

$$\boldsymbol{h}_v^{(k)}=\mathrm{GRU}\left(\boldsymbol{h}_v^{(k-1)},\boldsymbol{h}_{\mathcal{N}_{(v)}}^{(k)}\right) \tag{11}$$

其中，k 是一个超参数。为了计算图嵌入 $\boldsymbol{h}^{\mathcal{G}}$，本文首先对节点嵌入应用线性映射（LP, linear projection），然后对所有的节点嵌入应用最大池化（MP, max pooling），从而获得一个 F 维的向量 $\boldsymbol{h}^{\mathcal{G}}\in\mathbb{R}^F$。

在循环神经网络解码器方面，本文参考经典的序列到序列模型架构，并采用基于注意力机制的 GRU 解码器。解码器将图嵌入 $\boldsymbol{h}^{\mathcal{G}}$ 作为 GRU 初始隐藏层状态，同时利用节点嵌入 $\left\{\boldsymbol{h}_v^{(k)},\forall v\in\mathcal{G}\right\}$ 来计算注意力得分。在每一步中，解码器生成一个边缘服务节点编号，然后通过一个全连接层和 Softmax 函数输出部署决策序列的概率分布。图到序列模型算法描述如算法 1 所示。

算法 1　图到序列模型算法

输入　任务序列 $X=\{x_1,x_2,\cdots,x_M\}$，邻接点距离 K，计算跳数 k

输出　部署决策序列 $Y=\{y_1,y_2,\cdots,y_M\}$

1) 对任务序列 X 进行嵌入操作，生成任务嵌入矩阵 $\boldsymbol{E}$

2) 初始化节点嵌入 $\boldsymbol{h}^{(0)}=\mathrm{BiGRU}\left(\boldsymbol{E}\right)$ 和可训练权重矩阵 $\boldsymbol{W}$

3) 计算任务邻接矩阵 $\boldsymbol{A}$

4) 任务矩阵稀疏化处理 $\overline{\boldsymbol{A}}$

5) 归一化任务矩阵传入和传出方向 $\boldsymbol{A}^{\dashv},\boldsymbol{A}^{\vdash}$

6) for 1 to k do

　　更新传入和传出方向的节点嵌入信息 $\boldsymbol{h}_{\mathcal{N}_{\dashv(v)}}^{(k)},\boldsymbol{h}_{\mathcal{N}_{\vdash(v)}}^{(k)}$，融合节点的 2 个方向，聚合成一个节点嵌入 $\boldsymbol{h}_{\mathcal{N}_{(v)}}^{(k)}$，融合上一层节点嵌入信息 $\boldsymbol{h}_v^{(k)}$

end for

7) 计算任务图嵌入 $\boldsymbol{h}^{\mathcal{G}}=\text{MaxPool1d}\left(\text{Linear}(\boldsymbol{h}^{(k)})\right)$

8) 应用解码器对嵌入向量解码得到 $\text{logits}=\text{RNN}(\boldsymbol{h}^{(k)},\boldsymbol{h}^{\mathcal{G}})$

9) 进行归一化和抽样 $Y=\text{Sample}\left(\text{Softmax}(\text{logits})\right)$

10) 返回部署决策序列 Y

3.2 强化学习求解及策略梯度优化

在本文提出的求解策略中，强化学习模型由智能体（Agent）和环境（Environment）组成。Agent 从 Environment 获取某个状态后，利用该状态输出一个动作（Action），Action 会在环境中被执行，然后 Environment 根据 Agent 所采取的动作，输出下一个状态（State）以及当前动作所带来的奖励（Reward）。强化学习的目的就是寻找一个策略，使累计奖励（也称为回报）的期望最大化。这个策略被称为优化策略。在本文所研究的场景中，各种移动设备、边缘服务节点以及无线链路组成了强化学习的环境。本文在 3.1 节描述了 Agent 由深度神经网络模块加以实现。首先各类移动设备发送的复杂任务作为状态被 Agent 接收，然后 Agent 根据所制定的策略给出相应的部署决策作为动作，最后由环境执行该部署决策并给出一个奖励反馈。同时，移动设备发送新的任务作为下一个状态。在求解框架中，状态空间、动作空间以及奖励函数的详细定义介绍如下。

状态空间。移动设备发送的复杂任务可以表示为一个任务序列 $X=\{x_1,x_2,\cdots,x_M\}$，其中 $1\leqslant x_v\leqslant M$，$x_v$ 是某一子任务，包含 CPU、内存、磁盘资源，该序列是 MEC 环境传递给 Agent 的一个状态，所有可能的任务序列构成的空间称为状态空间，定义为 $\mathcal{X}$，即优化问题中复杂任务 $\mathcal{V}$ 的集合。

动作空间。经过 Agent 中神经网络的训练，Agent 将输出一个部署决策序列 $Y=\{y_1,\cdots,y_h,\cdots,y_N\}$，其中 $1\leqslant|h|\leqslant N$，当 $y_h=1$ 时也即 $f_{vh}=1$，表示子任务 v 在服务器 h 部署。该序列表示了每个子任务部署到边缘服务节点的状况，所有可能得到的部署决策构成的空间称为动作空间，定义为 $\mathcal{Y}$，即优化问题中有效解 $\mathcal{F}$ 的集合。

奖励函数。复杂任务部署的目标是最小化 MEC 系统的总能耗，在假设的多个边缘服务节点的环境中，每次执行任务部署决策后，环境会根据每个子任务部署到相应的边缘服务节点以及链路通信状态计算出复杂任务部署的总能耗。如果总能耗偏高，则会给出一个负反馈作为奖励；如果总能耗较低，则会给出一个正反馈作为奖励。因此，本文将部署决策序列 Y 所产生的总能耗 $E(Y)$ 作为奖励函数，如式(12)所示。

$$E(Y)=\sum_{y\in Y}\sum_{h\in\mathcal{H}}\left[E_h^{\text{idle}}+\left(E_h^{\max}-E_h^{\text{idle}}\right)F_h\right]y \tag{12}$$

在复杂任务部署问题中，可行解的组合数量极多且解的空间规模庞大。基于价值的学习方法很难学习到一个较好的结果。因此本文选择使用深度神经网络 $\pi(Y\mid X;\theta)$ 去近似策略函数，该神经网络被称为策略网络，θ 表示神经网络参数。每当观测到一个状态 X，就用策略网络计算出每个动作的概率值，然后进行随机抽样得到一个动作 Y，最后交由环境执行该动作。为了找到一个良好的策略函数，策略的质量应仅依赖于神经网络参数 θ，而不受任何时刻的状态和动作的影响。为此，本文使用策略梯度方法定义一个目标函数，该目标函数代表每个 θ 所获得的期望回报。通过不断迭代更新，任务部署模型能够适应各种情况。首先，在给定复杂任务序列的所有可能部署方案 $Y\in\pi_\theta(\cdot\mid X)$ 中，定义其产生的期望能耗 $J_\pi^E(\theta\mid X)$ 如式(13)所示，从而消除不同部署方案带来的影响。

$$J_\pi^E(\theta\mid X)=\mathbb{E}_{Y\sim\pi_\theta(\cdot\mid X)}[E(Y)] \tag{13}$$

然后，Agent 需要从所有可能的任务组合中推断出子任务部署的策略。因此，求式(13)的期望，去除不同任务组合带来的影响，得到式(14)。

$$J_\pi^E(\theta)=\mathbb{E}_{X\sim\mathcal{X}}\left[J_\pi^E(\theta\mid X)\right] \tag{14}$$

此时，策略函数只受 θ 的影响。同时还需要考虑与策略相关联的约束不满足的情况，则可表示为

$$J_\pi^C(\theta)=\mathbb{E}_{X\sim\mathcal{X}}\left[J_\pi^C(\theta\mid X)\right] \tag{15}$$

至此，式(4)所描述的优化问题则转化为式(16)所描述的寻找期望能耗最小的策略。

$$\begin{aligned}&\min_{\pi}J_\pi^E(\theta)\\ \text{s.t.}\quad &J_\pi^{C_i}(\theta)\leqslant 0\end{aligned} \tag{16}$$

其中，$J_\pi^{C_i}(\theta)$ 表示环境返回的每个不满足约束的信号。在本文的场景中，有 5 个信号，分别代表 CPU、内存、磁盘资源以及带宽和时延累计约束不满足。

考虑式(16)中描述的原始问题的优化解，即目

标函数在满足约束条件下可以获得的最小值。本文继续利用拉格朗日松弛技术，将该问题转化为一个不受约束的问题，其中不可行的解决方案受到惩罚，如式(17)所示。

$$g(\lambda)=\min_\theta J_\pi^L(\lambda,\theta)=\min_\theta\left[J_\pi^E(\theta)+\sum_i\lambda_i J_\pi^{C_i}(\theta)\right]=\min_\theta\left[J_\pi^E(\theta)+J_\pi^\xi(\theta)\right] \tag{17}$$

其中，$J_\pi^L(\lambda,\theta)$是拉格朗日目标函数；$g(\lambda)$是拉格朗日对偶函数；λ_i是拉格朗日乘子，也是惩罚系数。同时，定义$J_\pi^\xi(\theta)$为期望惩罚，其值是所有约束不满足信号的期望加权和。

对偶函数是一个凸函数，因此可以据此找到产生下界的拉格朗日系数，从而求得原始问题的最优值。拉格朗日对偶问题如式(18)所示。

$$\max_\lambda g(\lambda)=\max_\lambda\min_\theta J_\pi^L(\lambda,\theta) \tag{18}$$

本文手动选择拉格朗日乘子，由此产生的拉格朗日函数$J_\pi^L(\theta)$变成了本文需要推导策略的最终目标函数。本文希望对策略网络中的参数θ进行更新，使目标函数$J_\pi^L(\theta)$越来越小。因此本文采用蒙特卡罗策略梯度以及梯度下降更新θ。设当前策略网络的参数为θ_{now}，经过梯度下降更新，得到新的参数θ_{new}，可以表示为

$$\theta_{\text{new}}=\theta_{\text{now}}+\beta\nabla_\theta J_\pi^L(\theta) \tag{19}$$

本文使用策略梯度定理，利用对数似然法导出了拉格朗日梯度，如式(20)所示。

$$\nabla_\theta J_\pi^L=\mathbb{E}\mathbb{E}_{Y\sim\pi_\theta(\cdot|X)}\left[L(Y|X)\nabla_\theta\log\pi_\theta(Y|X)\right]$$
$$L(Y|X)=E(Y|X)+\xi(Y|X)=E(Y|X)+\sum_i\lambda_i C_i(Y|X) \tag{20}$$

其中，$L(Y|X)$定义为每次迭代得到的期望能耗惩罚，是能耗信号$E(Y|X)$与所有约束不满足信号$C(Y|X)$的和。

在实际操作中，通过连加或者定积分求出期望的计算量非常大。为此，本文使用蒙特卡罗近似方法去近似策略梯度，从状态空间$\mathcal{X}$中随机抽出B个样本$\left(X_1,X_2,\cdots,X_B\right)\sim\mathcal{X}$。同时，使用一个不依赖于动作$Y$的基线$b_{\theta_c}(X)$减少梯度的方差，加快式(21)的收敛速度。

$$\nabla_\theta J_\pi^L(\theta)\approx\frac{1}{B}\sum_{j=1}^B\left(L\left(Y_j|X_j\right)-b_{\theta_c}(X_j)\right)\cdot\nabla_\theta\log\pi_\theta\left(Y_j|X_j\right) \tag{21}$$

其中，本文使用一个只与状态X相关的状态价值网络去近似基线，网络的输入与策略网络相同，是策略梯度的一个无偏估计。神经网络参数θ_c使用随机梯度下降训练，损失函数为预测值$b_{\theta_c}(X)$和从环境中获得的实际惩罚期望的均方误差，如式(22)所示。

$$\mathcal{L}\left(\theta_c\right)=\frac{1}{B}\sum_{j=1}^B\left\|b_{\theta_c}(X_j)-L\left(Y_j|X_j\right)\right\|^2 \tag{22}$$

综上，本文提出的任务部署决策求解框架的训练算法如算法 2 所示。

算法 2　基于带基线的强化学习算法的复杂任务部署训练算法

输入　任务集合$\mathcal{X}$，训练回合数 episodes，批处理大小B

输出　权重参数θ

1) 初始化环境信息，加载边缘服务节点及链路信息

2) 初始化策略网络参数θ

3) 初始化价值网络参数θ_c

4) for $n=1$ to episodes do

$X_j\sim\text{SampleInput}\left(\mathcal{X}\right),j\in\{1,\cdots,B\}$　// 从任务集合中抽取B个

样本训练

$Y_j\sim\text{SampleSolution}\left(\pi_\theta\left(\cdot|X_j\right)\right),j\in\{1,\cdots,B\}$

//从B个样本中抽取相应的部署决策

$b_j\leftarrow b_{\theta_v}\left(X_j\right),j\in\{1,\cdots,B\}$　//计算相应的基线

$g_\theta\leftarrow\frac{1}{B}\sum_{j=1}^B\left(L\left(Y_j|X_j\right)-b_{\theta_c}\left(X_j\right)\right)\nabla_\theta\log\pi_\theta\left(Y_j|X_j\right)$

//梯度计算

$\mathcal{L}\left(\theta_c\right)\leftarrow\frac{1}{B}\sum_{j=1}^B\left\|b_{\theta_c}\left(X_j\right)-L\left(Y_j|X_j\right)\right\|^2$

//计算损失函数

$\theta\leftarrow\text{Adam}\left(\theta,g_\theta\right)$　//更新策略网络参数

$\theta_c \leftarrow \mathrm{Adam}\left(\theta_c, \nabla_{\theta_c} \mathcal{L}_c\right)$//更新价值网络参数

end for

5) 返回神经网络参数 θ

4 实验与结果分析

4.1 实验环境搭建

本文通过模拟真实 MEC 场景下的节点复杂任务请求，对所提出的求解策略进行性能评估。实验在深度学习服务器上进行，其硬件配置包括 I9 处理器（主频 3.0 GHz），双 NVIDIA GeForce RTX3090 GPU，64 GB 内存以及 2 TB 固态硬盘。同时，实验利用了 PyTorch 1.8 实现深度学习和神经网络部分，并在 Pycharm 平台上实现了系统。边缘服务节点部署在多个移动设备周围半径为 1 km 的范围内。

为模拟真实的复杂任务，本文参考文献[9]实现了一个复杂任务图结构生成器来产生任务序列。生成器的参数如下。1)任务长度：每个图中的子任务数量。本文定义了两类任务，第 1 类任务的任务长度从 12 增加到 24，第 2 类任务的任务长度从 20 增加到 32，两类任务均以步长为 2 逐渐增加。2)资源需求量：包括 CPU、内存和磁盘资源的需求量。鉴于子任务是在边缘服务节点的虚拟机或容器中执行的，本文对 MEC 物理节点资源进行抽象，将 CPU、内存、磁盘资源等物理资源转换为可管理、调度、分发的逻辑资源。其中，从集合 {1,2,4} 中随机选择 CPU 核心数量，从集合 {1,2,4,8} 中随机选择内存大小，从集合 {50,100,150,250} 中随机选择磁盘容量。3)子任务需求带宽：子任务部署到边缘服务节点上时，每个子任务执行的带宽需求量服从[10,100] 的均匀分布。4)子任务容忍时延：为了简化时延问题，部署问题中每个子任务都有一个从上传数据到执行完成的最大容忍时延，服从[1,10] 的均匀分布。

在边缘服务节点的参数设置方面，本文首先定义了小型和大型 2 种网络规模的环境，边缘服务节点的数量分别为 10 和 20。另外，实验模拟了真实环境中边缘服务节点的异构性，即节点的参数规格以及可提供的 IT 资源规模存在差异。其中，边缘服务节点的满载功耗因具体规格、配置和应用场景而异。通常边缘服务节点会根据其处理能力、内存、磁盘和其他硬件组件的需求进行设计，以确保能效比（性能与功耗之比）达到最佳。实验假设了 4 种类型的边缘服务节点，每个节点的闲置功耗为 100 W。针对大规模环境，本文选择 Type1 型 8 台，Type2 型 6 台，Type3 型 4 台，Type4 型 2 台；针对小规模环境，本文选择 Type1 型 4 台，Type2 型 3 台，Type3 型 2 台，Type4 型 1 台。MEC 环境中服务节点的参数设置如表 1 所示。

表 1 MEC 环境中服务节点的参数设置

节点类型	取值范围
MEC-Type1	CPU 核心数为 6，内存为 8 GB，磁盘为 300 GB，链路时延为 1 s，链路带宽为 100 Mbit/s，满载功耗为 300 W
MEC-Type2	CPU 核心数为 8，内存为 16 GB，磁盘为 400 GB，链路时延为 3 s，链路带宽为 400 Mbit/s，满载功耗为 400 W
MEC-Type3	CPU 核心数为 10，内存为 24 GB，磁盘为 500 GB，链路时延为 5 s，链路带宽为 500 Mbit/s，满载功耗为 500 W
MEC-Type4	CPU 核心数为 16，内存为 32 GB，磁盘为 600 GB，链路时延为 5 s，链路带宽为 1 000 Mbit/s，满载功耗为 700 W

此外，在模型训练方面，神经网络相关参数设置如表 2 所示。

表 2 神经网络相关参数设置

节点类型	取值范围
学习率（智能体）	0.001
批处理大小	128
嵌入层维度	3
图神经网络计算跳数	{1,2,3}
学习率（基线）	0.1
温度超参数	2
推理模型数	6
温度超参数抽样数量	16

4.2 对比算法介绍

实验选取了 4 种具有代表性的基准部署策略与本文所提出的策略 DRL-G2S 进行比较。基准部署策略介绍如下。

1) NCO 算法[13-14]。NCO 算法是近年来提出的基于神经网络求解组合优化问题的新策略，基于强化学习来训练神经网络以获得优化解。NCO 算法在组合优化问题上求解质量较好，但模型训练和算法执行通常需要较高的计算资源。

2) FF 算法[15-16]。FF 及其变种算法是经典的资源分配和任务调度算法，属于启发式策略，已经被广泛应用于数据中心及云计算等领域。FF 算法的核心思想是按照一定的顺序（如任务到达顺序）对任务进行处理，将每个任务分配给第一个能够满足该任务需求的资源。在实验中，FF 算法能够遍历 MEC

环境中的所有节点，探索出各种可能的部署情况。

3) Gurobi 求解器[17]。Gurobi 是一款高性能的商用数学优化求解器。无论问题求解速度还是解的质量，Gurobi 求解器都有优秀的表现，本文选择 Gurobi 求解结果作为理论最优结果。虽然 Gurobi 求解器具有良好的性能，但在 MEC 环境下用户对任务的服务响应时间要求很高，许多时候需要在秒级内给出良好的部署决策。因此，本文主要在有时间约束的前提下对 DRL-G2S 与 Gurobi 的求解质量进行对比。实验中将有效的调度策略的时间限制在 1 s 以内。

4) HIA[18-19]。HIA 将启发式算法和神经网络加以结合，通过利用启发式算法先寻找到潜在可行解来指导神经网络算法优化自身参数，以获得优化的部署决策。

4.3　评价指标

本文选取了以下几个关键指标作为评价标准，从多个角度对比了各种不同的部署策略。

1) 模型稳定性：评估模型训练过程中的历史学习曲线的收敛性。通过分析模型在训练过程中损失曲线的波动情况，来评估模型的收敛情况。

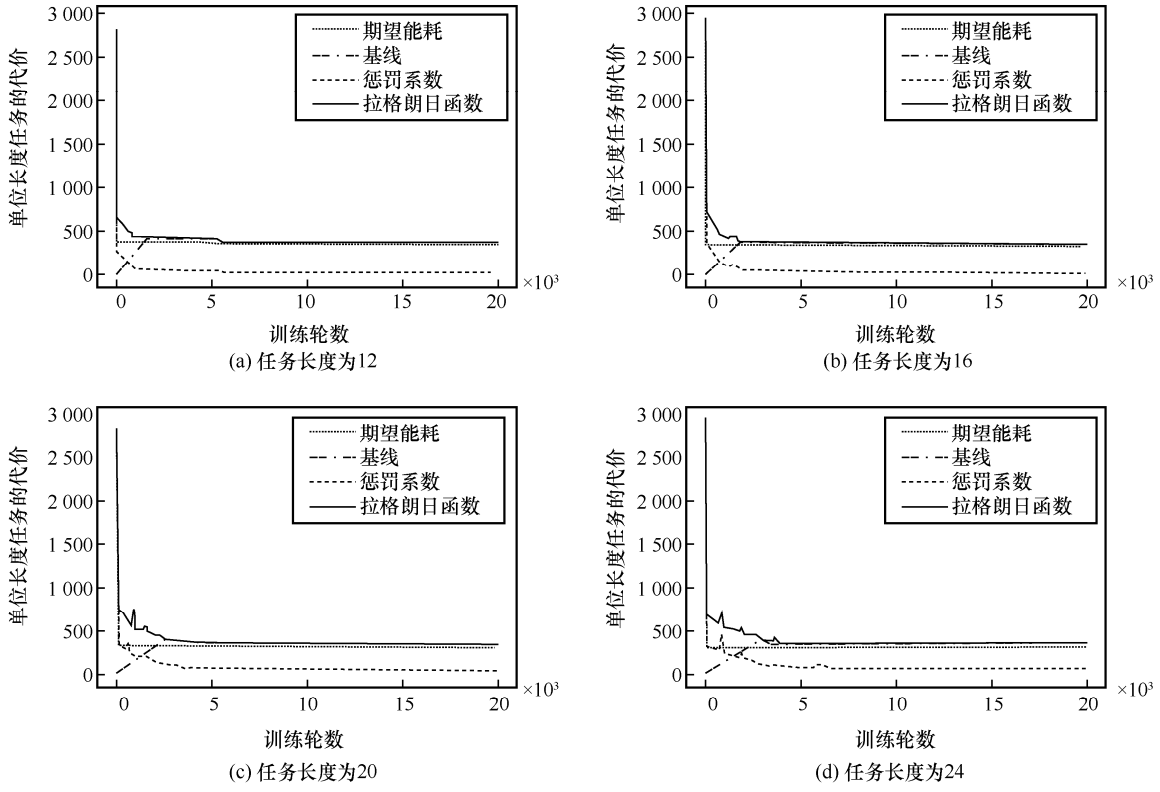
2) 部署错误率：复杂任务的多个子任务在各个不同边缘服务节点上部署所产生的期望惩罚。如果调度结束后期望惩罚为 0，则认为部署策略有效，否则部署失败。该指标能够直接评估各种部署策略的有效性。

3) 期望能耗：评估在不同部署策略下系统的能耗开销。该指标是能耗优化最重要的指标，良好的部署策略应在满足其他约束条件下，尽量降低系统整体能耗。该指标直接影响边缘服务商的运营开销。

4) 平均求解时间：评估通过不同算法获得的部署决策的实际执行时间。由于该指标直接影响提交复杂任务请求的用户体验，因此该指标是衡量部署策略质量的关键指标。

4.4　结果分析

本节将通过应用上述的几种评价指标，在小规模和大规模环境下对所提策略进行仿真，并对仿真结果进行深入分析。本文对所提出的图到序列模型的性能进行了评估。通过研究图到序列模型在不同任务长度下的学习历史，来分析模型的稳定性以及适用性。在小规模系统环境下，当任务长度分别为 12、16、20 以及 24 时，对模型进行训练的实验结果如图 3 所示。从图 3 可以看到，随着训练轮数的增加，模型的代价逐渐下降并达到收敛。由于本实验涉及不同任务长度的训练，因此本文将系统总开销除以任务长度得到单



(a) 任务长度为12　(b) 任务长度为16

(c) 任务长度为20　(d) 任务长度为24

图 3　在小规模系统环境下对模型进行训练的实验结果

位长度任务的代价，以客观评估在不同任务长度下的模型训练效果。另外，随着子任务数量的不断增加，可用资源数量将会从空间充足的状态变得越来越有限。为了更加清楚地了解相关的状态，本文引入了期望能耗 J_π^E、基线 b、惩罚系数 J_π^ξ 和拉格朗日函数 J_π^L 的近似值来进行衡量。

从单一任务长度来看，在学习开始时，智能体生成较多违反约束的随机放置序列，导致惩罚系数较高。然而，在学习过程中，智能体通过随机梯度下降不断调整神经网络参数权重值，使基线值从 0 增加并逼近拉格朗日函数，从而加速拉格朗日函数的最小化速度。在大量迭代过程后，智能体持续改进其策略，减少约束不满足情况的出现，寻找局部极小值或鞍点，直至达到最终的稳定状态。在不同任务长度下，经过 20 000 轮的迭代可以发现，当任务长度较小时，边缘服务节点所能提供的资源相对充足，与之相关的惩罚系数在训练后接近 0，模型迅速趋于稳定，可以推断出较优的调度策略。然而，随着任务长度的增加，边缘服务节点所能提供的资源逐渐有限，模型需要更长的时间才能达到稳定状态，同时约束不满足的概率增大，惩罚系数相应提高。图 4 展示了不同任务长度下 DRL-G2S 训练的损失曲线。该曲线反映了模型训练的动态趋势和收敛情况，可以发现，随着任务长度的增加，模型一开始的损失值逐渐增大且曲线的波动越发明显，同时模型收敛的训练轮数也在增加，这表明任务长度的增加提高了任务复杂度，进而导致 DRL-G2S 需要更长的训练时长。

为了验证 DRL-G2S 的有效性，本文在小规模和大规模环境下，通过应用上述的多种评价标准，将模型的结果与 FF 算法、NCO 算法、Gurobi 求解器以及 HIA 进行实时比较。针对不同任务长度，本文分别随机抽取了 1 000 个任务进行测试，根据可行解的数量来评估结果。在这个实验中，Gurobi 求解器在 2 种规模下的最大执行限制时间分别为 1 s 和 10 s。

针对部署错误率，图 5 比较了 2 种规模下 5 种算法的部署错误率。总体而言，随着任务长度的增加，不同调度策略的部署错误率逐渐上升，这是因为边缘服务节点的资源环境受到更严格的限制，有效解的空间逐渐减小。从图 5(a)可以看出，在小规模环境下，各算法的错误率差异较小，但 DRL-G2S 明显表现更优。同时，当任务长度较短时，FF 算法和 NCO 算法的部署错误率相近。当任务长度适中

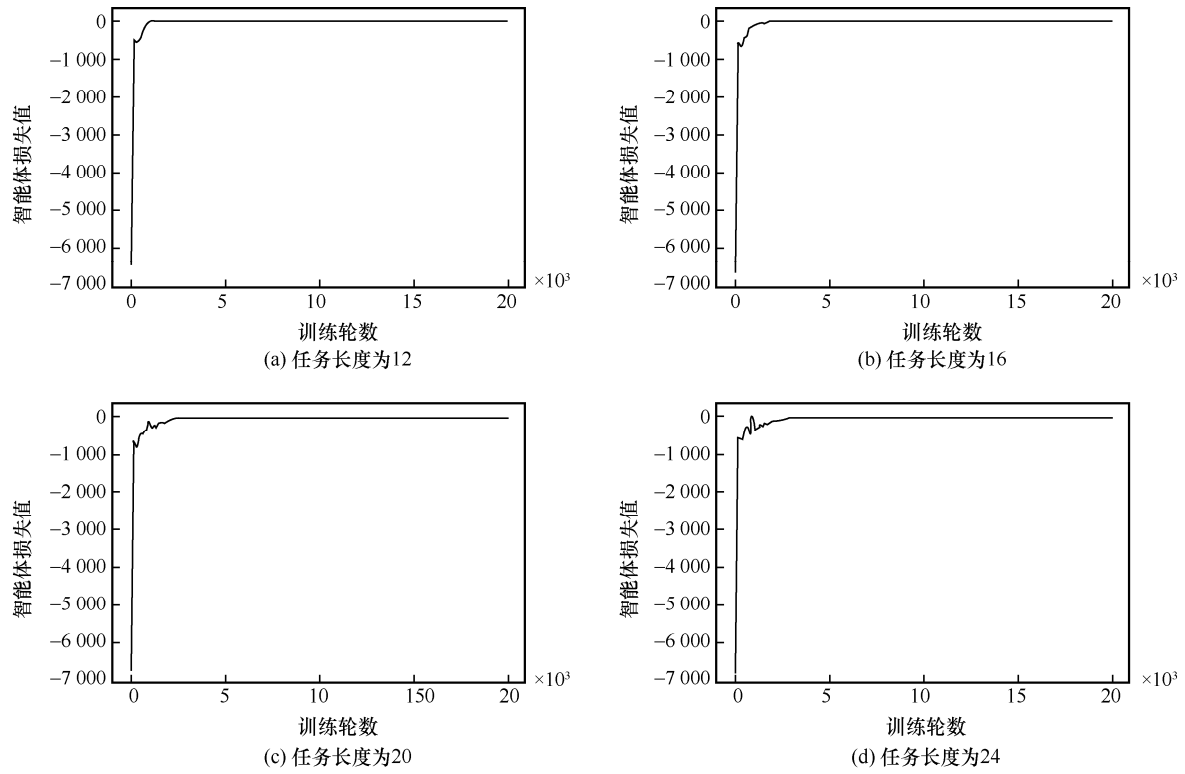
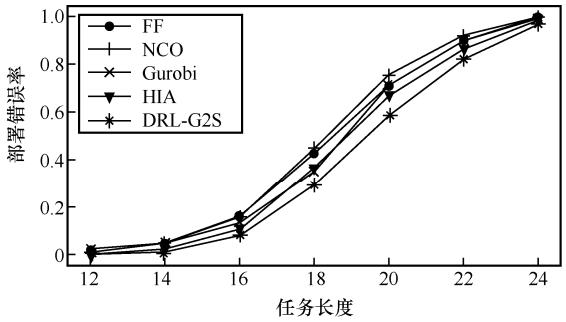
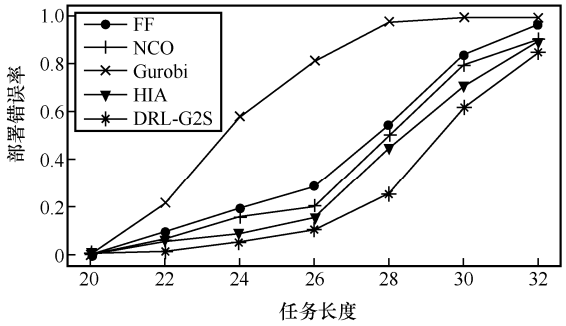


图 4　不同任务长度下 DRL-G2S 训练的损失曲线

（如任务长度为 18）时，Gurobi 求解器的部署错误率接近 HIA，相对于其他 2 种算法有 18.7%的提升，但仍高于 DRL-G2S 算法。在大规模环境下，如图 5(b)所示，Gurobi 求解器的错误率较高，这是因为求解时间限制导致其无法在短时间内获得有效解决方案。FF 算法接近 NCO 算法，但仍存在一定差距。HIA 综合了 FF 算法和 NCO 算法的特点，性能优于前两者。DRL-G2S 在大规模环境下的错误率相对较低。综上所述，无论在哪种规模的环境下，DRL-G2S 在解决方案有效性方面都表现出色。



(a) MEC服务器数量为10

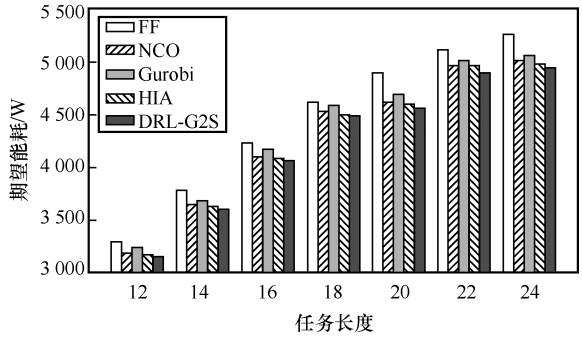


(b) MEC服务器数量为20

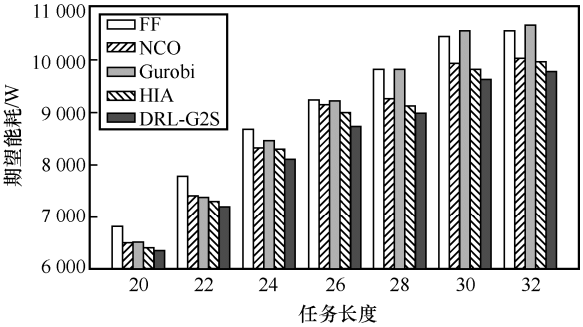
图 5　2 种规模下 5 种算法的错误率

系统能耗开支是本文重点关注的指标。为评估 5 种算法在不同网络规模以及不同任务复杂度（一个复杂任务所包括的子任务的规模及关联性）情况下的性能，在小规模和大规模的 MEC 场景下，服务节点规模分别为 10 和 20，而请求的复杂任务则分别包含 12～24 个子任务和 20～32 个子任务。

图 6 展示了 5 种算法在不同任务长度下的期望能耗对比。在图 6(a)的小规模场景下，3 种与神经网络相关的算法（NCO 算法、HIA 以及 DRL-G2S）的期望能耗接近，但 DRL-G2S 的能耗表现相对更低。值得注意的是，FF 算法的能耗始终保持在最高水平，这表明启发式算法所得解的质量不够理想，即使在解的输出时间开销较低的情况下，也不利于能耗的优化控制。在图 6(b)的大规模场景下，Gurobi 求解器在任务长度较短（即资源约束较小）的情况下表现出较低的期望能耗，但随着任务长度的增加，由于求解时间限制，其期望能耗逐渐升高，最终超过其他 4 种算法。随着网络规模的增加和任务复杂性的提高，DRL-G2S 在能效方面相对于 NCO 算法和 HIA 表现出明显的优势。这主要是因为 DRL-G2S 引入了图神经网络来提取复杂任务中多个子任务之间的关联关系，从而有助于找到更有利于能耗优化的解决方案。



(a) MEC服务器数量为10

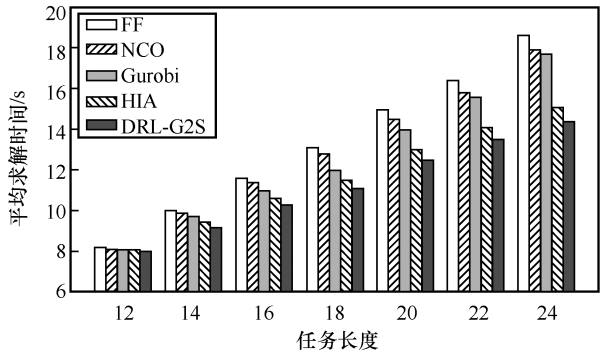


(b) MEC服务器数量为20

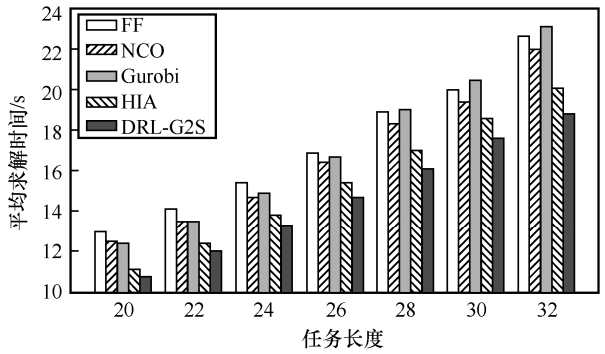
图 6　5 种算法在不同任务长度下的期望能耗对比

最后，本文在 2 种不同规模的 MEC 场景下对 5 种算法的部署策略求解时间进行了对比，如图 7 所示。在图 7(a)中，随着任务长度（即任务复杂度）的增加，NCO、HIA 以及 DRL-G2S 这 3 种基于神经网络算法的部署策略求解时间都相应增加。值得注意的是，尽管 NCO 和 HIA 均采用了强化学习方法以实现自学习搜索优化解，但由于 DRL-G2S 在智能体的设计中引入了基于图到序列的编解码设计，能够更深入地捕捉子任务之间的关系，从而使求解时间降低了 5.25%~10.37%。在图 7(b)中，Gurobi 求解器在限定时间内难以找到具有优势的部署策略，其任务平均求解时间逐渐高于其他对比算法，因此基于商用求解器的策略不适合对时间要求严格的复杂任务进行部署决策。更重要的是，随着任务长度增

加，DRL-G2S 的优势更明显，表明 DRL-G2S 更适用于解决复杂任务的部署请求。当 MEC 服务器数量增加时，算法有机会选择 IT 资源更充裕或者间距更近的服务节点部署多个子任务，因此在相同任务长度下算法获得的平均求解时间降低。例如，与小规模 MEC 场景相比，当任务长度为 20、22 和 24 时，大规模 MEC 场景下的平均求解时间分别降低了 12.23%、11.36%和 10.17%。另外，和在小规模 MEC 场景下类似，采用 DRL-G2S 仍然表现出相对更低的平均求解时间。



(a) MEC服务器数量为10



(b) MEC服务器数量为20

图 7 2 种不同规模的 MEC 场景下 5 种算法的部署策略求解时间

5 结束语

本文深入探讨了 MEC 环境中具有挑战性的多资源约束下的复杂任务部署问题，在考虑多种资源限制条件的同时，以最小化能耗为目标建立了 MIP 模型，利用图神经网络算法动态建模子任务间的关系，设计了一个融合图到序列的深度强化学习求解策略。实验结果表明，在相同的实验环境下，本文所提出的求解策略通过持续学习和主动推理部署策略，在综合任务部署错误率、MEC 系统总能耗以及算法平均求解时间等关键评价指标方面均优于各类代表性求解策略。后续研究将继续探索图结构的自动特征提取及学习的策略，并进一步研究基于图到序列的神经网络模型求解资源分配等关联问题。

参考文献：

[1] DUAN S J, WANG D, REN J, et al. Distributed artificial intelligence empowered by end-edge-cloud computing: a survey[J]. IEEE Communications Surveys & Tutorials, 2023, 25(1): 591-624.

[2] LUO R K, JIN H, HE Q, et al. Cost-effective edge server network design in mobile edge computing environment[J]. IEEE Transactions on Sustainable Computing, 2022, 7(4): 839-850.

[3] DJIGAL H, XU J, LIU L F, et al. Machine and deep learning for resource allocation in multi-access edge computing: a survey[J]. IEEE Communications Surveys & Tutorials, 2022, 24(4): 2449-2494.

[4] GU L, ZHANG W Y, WANG Z K, et al. Service management and energy scheduling toward low-carbon edge computing[J]. IEEE Transactions on Sustainable Computing, 2023, 8(1): 109-119.

[5] FENG C, HAN P C, ZHANG X, et al. Computation offloading in mobile edge computing networks: a survey[J]. Journal of Network and Computer Applications, 2022, 202(8): 1033-1046.

[6] YE Y H, SHI L Q, CHU X L, et al. Resource allocation in backscatter-assisted wireless powered MEC networks with limited MEC computation capacity[J]. IEEE Transactions on Wireless Communications, 2022, 21(12): 10678-10694.

[7] SUN H, ZHOU F, HU R Q. Joint offloading and computation energy efficiency maximization in a mobile edge computing system[J]. IEEE Transactions on Vehicular Technology, 2019, 68(3): 3052-3056.

[8] YANG G S, HOU L, HE X Y, et al. Offloading time optimization via Markov decision process in mobile-edge computing[J]. IEEE Internet of Things Journal, 2021, 8(4): 2483-2493.

[9] WANG J, HU J, MIN G, et al. Computation offloading in multi-access edge computing using a deep sequential model based on reinforcement learning[J]. IEEE Communications Magazine, 2019, 57(5): 64-69.

[10] LI H, XU H, ZHOU C, et al. Joint optimization strategy of computation offloading and resource allocation in multi-access edge computing environment[J]. IEEE Transactions on Vehicular Technology, 2020, 69(9): 10214-10226.

[11] CUI Y Y, ZHANG D G, ZHANG T, et al. A novel offloading scheduling method for mobile application in mobile edge computing[J]. Wireless Networks, 2022, 28(6): 2345-2363.

[12] HU B, YAN Z L, ZHAO M G. Workload-aware scheduling of multiple-criticality real-time applications in vehicular edge computing system[J]. IEEE Transactions on Industrial Informatics, 2023, 19(10): 10091-10101.

[13] ATTAOUI W, SABIR E, ELBIAZE H, et al. VNF and CNF placement in 5G: recent advances and future trends[J]. IEEE Transactions on Network and Service Management, 2023, 20(4): 4698-4733.

[14] CHEN Z, ZHU B W. Deep reinforcement learning based container cluster placement strategy in edge computing environment[C]// Proceedings of the 2022 IEEE Global Communications Conference. Piscataway: IEEE Press, 2022: 2212-2217.

[15] LI C L, TANG J H, MA T, et al. Load balance based workflow job scheduling algorithm in distributed cloud[J]. Journal of Network and Computer Applications, 2020, 152(1): 1025-1037.

[16] KIRAN N, LIU X L, WANG S H, et al. VNF placement and resource

allocation in SDN/NFV-enabled MEC networks[C]//Proceedings of the 2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW). Piscataway: IEEE Press, 2020: 1-6.

[17] MULEY V Y. Mathematical programming for modeling expression of a gene using gurobi optimizer to identify its transcriptional regulators[J]. Methods in Molecular Biology, 2021, 12(6): 99-113.

[18] XUE F, HAI Q R, DONG T T, et al. A deep reinforcement learning based hybrid algorithm for efficient resource scheduling in edge computing environment[J]. Information Sciences: an International Journal, 2022, 608(3): 362-374.

[19] CHEN Z, WEI P H, LI Y. Combining neural network-based method with heuristic policy for optimal task scheduling in hierarchical edge cloud[J]. Digital Communications and Networks, 2023, 9(3): 688-697.

[20] LAROUI M, KHEDHER H I, MOUNGLA H, et al. Virtual mobile edge computing based on IoT devices resources in smart cities[C]//Proceedings of the IEEE International Conference on Communications (ICC). Piscataway: IEEE Press, 2020: 1-6.

[21] CHEN L, WU J G, ZHANG J, et al. Dependency-aware computation offloading for mobile edge computing with edge-cloud cooperation[J]. IEEE Transactions on Cloud Computing, 2022, 10(4): 2451-2468.

[22] MAZYAVKINA N, SVIRIDOV S, IVANOV S, et al. Reinforcement learning for combinatorial optimization: a survey[J]. Computers & Operations Research, 2021, 134(2): 1054-1069.

[23] LIU J, LI X H, LIU X M, et al. A privacy-preserving service framework for traveling salesman problem-based neural combinatorial optimization network[J]. IEEE Transactions on Cloud Computing, 2023, 11(4): 3381-3395.

[24] JIANG Q M, ZHANG Y, YAN J Y. Neural combinatorial optimization for energy-efficient offloading in mobile edge computing[J]. IEEE Access, 2020, 8: 35077-35089.

[25] HUANG Z N, SAMAAN N, KARMOUCH A. A novel resource reliability-aware infrastructure manager for containerized network functions[C]//Proceedings of the IEEE International Conference on Communications. Piscataway: IEEE Press, 2021: 1-6.

[26] ADOGA H U, ELKHATIB Y, PEZAROS D P. On the performance benefits of heterogeneous virtual network function execution frameworks[C]//Proceedings of the 2022 IEEE 8th International Conference on Network Softwarization (NetSoft). Piscataway: IEEE Press, 2022: 109-114.

[27] MUNIEN C, EZUGWU A E. Metaheuristic algorithms for one-dimensional Bin-packing problems: a survey of recent advances and applications[J]. Journal of Intelligent Systems, 2021, 30(1): 636-663.

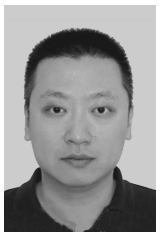
[28] RUIZ L, GAMA F, RIBEIRO A. Graph neural networks: architectures, stability, and transferability[J]. Proceedings of the IEEE, 2021, 109(5): 660-682.

[29] CHEN Y, WU L F, ZAKI M J. Reinforcement learning based graph-to-sequence model for natural question generation[J]. arXiv Preprint, arXiv: 1908.04942, 2019.

[作者简介]

陈卓（1980− ），男，重庆人，博士，重庆理工大学副教授，主要研究方向为边缘计算、物联网及区块链技术等。

操民涛（1996− ），男，安徽怀宁人，重庆理工大学硕士生，主要研究方向为边缘计算与机器学习算法。



周致圆（1981− ），男，重庆人，重庆凯瑞机器人技术有限公司高级工程师，主要研究方向为智能计算与工业互联网技术。

黄欣（1974− ），女，重庆人，中国移动通信集团重庆有限公司高级工程师，主要研究方向为 5G 网络智能化管理与控制技术。

李彦（1976− ），男，湖南岳阳人，博士，重庆理工大学教授，主要研究方向为智能网络与工业互联网技术。