计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.231100170

基于深度强化学习的云边协同任务迁移与资源再分配优化研究

陈 娟[1] 王 阳[1] 吴宗玲[2] 陈 鹏[1] 张逢春[1] 郝俊峰[1]
1 西华大学计算机与软件工程学院 成都 610039
2 西南交通大学信息科学与技术学院 成都 611756
(chenjuan@mail.xhu.edu.cn)

摘 要 文中研究了一个由多个边缘服务器和云服务器组成的异构云边环境,每个节点都具有计算、存储和通信能力。由于异构云边环境的不确定性和动态性,需要进行动态调度以优化资源和任务分配。传统的深度学习框架只从输入的任务数据中提取潜在的特征,大多数忽略了云边环境的网络结构信息。为了解决这个问题,文中基于 Actor-Critic 框架,利用柔性动作-评价(Soft Actor-Critic,SAC)经验训练的自进化能力和图卷积网络(Graph Convolutional Networks,GCN)中基于图的关系推演能力,提出了一种集中式的 SAC-GCN 算法。提出的 SAC-GCN 通过捕获动态的任务信息和异构的节点信息,采用自适应的损失函数来提供有效的调度策略,以适应不同的任务迁移需求。文中采用来源于真实世界的 Bit-brain 数据集,并通过 Cloud-Sim 进行大量模拟。实验结果表明,与现有算法相比,提出的 SAC-GCN 可以减少 4.81%系统能耗,缩短 3.46%任务响应时间和 2.73%的任务迁移时间,以及减小 1.5%的任务 SLA 违规率。

关键词:云计算;边缘计算;深度强化学习;动态调度;柔性动作-评价;图卷积
中图分类号 TP18

Cloud-Edge Collaborative Task Transfer and Resource Reallocation Optimization Based on Deep Reinforcement Learning

CHEN Juan[1],WANG Yang[1],WU Zongling[2],CHEN Peng[1],ZHANG Fengchun[1] and HAO Junfeng[1]
1 School of Computer and Software Engineering,Xihua University,Chengdu 610039,China
2 School of Information Science and Technology,Southwest Jiaotong University,Chengdu 611756,China

Abstract In this paper,we have investigated a heterogeneous cloud-edge environment consisting of multiple edge servers and cloud servers,where each node has computation,storage and communication capabilities. Due to the uncertainty and dynamics of the heterogeneous cloud edge environment,dynamic scheduling is required to optimize resource and task allocation. The traditional deep learning framework only extract the potential features from the input task data,mostly ignoring the network structure information characteristics of the cloud-edge environment. To solve this problem,this paper proposes a distributed SAC-GCN algorithm based on the Actor-Critic framework,using the self-evolutionary ability of the experience training of soft actor-critic(SAC) and the graph-based relationship inference ability of graph convolutional networks(GCN). The proposed SAC-GCN employs an adaptive loss function to provide effective scheduling strategies for different task migration requirements by capturing dynamic task information and heterogeneous node resource information. In this paper,we utilize the Bit-brain dataset sourced from the real world,and carries out a large number of simulations through Cloud-Sim. Experimental results show that compared with the existing algorithms,the proposed SAC-GCN can reduce the system energy consumption by 4.81%,shorten the task response time by 3.46% and the task migration time by 2.73%,and reduce the task SLA violation rate by 1.5%.

Keywords Cloud computing,Edge computing,Deep reinforcement learning,Dynamic scheduling,Flexible action-evaluation,Graph convolution

1 引言

近年来随着信息时代的到来和互联网的快速发展,用户对计算机的计算能力[1]、内存大小、网络带宽[2]、能耗管理和服务质量[3]等提出了更高的要求。并且对于拥有较多复杂多变的任务要求,仅靠固有的计算资源已经非常难以实现。随着个人计算机的普及造就了 PC 虚拟化技术的发展[4],也正是基于此,为云计算平台[5]的孕育、诞生和成长提供了支撑,

基金项目:国家自然科学基金(62376043);四川省科技计划(2020JDRC0067,2023JDRC0087);四川省自然科学基金(2024NSFTD008,2022NSFSC0556);教育部春晖计划(HZKY20220578)
This work was supported by the National Natural Science Foundation of China (62376043),Science and Technology Program of Sichuan Province (2020JDRC0067,2023JDRC0087),Natural Science Foundation of Sichuan Province of China (2024NSFTD008,2022NSFSC0556) and Chunhui Program of Ministry of Education of China(HZKY20220578).
通信作者:陈鹏(chenpeng@mail.xhu.edu.cn)

利用云计算平台的资源，能更好满足用户的需求。但只利用云计算平台的资源、虽然能带来丰富的资源、强大的计算能力，且能同时处理多个任务，但依然会存在计算延迟、拥塞、低可靠性、安全攻击等问题。为了解决传统数据处理方式下时延高、数据实时分析能力匮乏等弊端，边缘计算技术[6]应运而生，通过边缘计算[7]的分布式平台，为用户提供就近的边缘智能服务，从而降低延迟，提供更能实时处理数据的能力、更短的响应时间。因此，针对计算密集型和时延敏感型的任务，合理将边缘节点和云节点进行协同[8]计算更有研究价值。

在当前的边缘云环境中，不同节点在容量、速度、响应时间和能量消耗方面都有显著的差异，同时考虑到用户在云边环境中任务请求的随机性和计算节点资源的异构性，使得任务的调度[9]优化仍然面临很多挑战。传统方法（如轮询调度算法[10]和启发式[11-12]）已不再适用于解决边缘云中动态的任务迁移与资源分配优化问题。这些方法的搜索空间呈指数级增长，设备计算负担重，特别是在大规模场景下，无法满足多接入边缘计算（Muti-access Edge Computing，MEC）场景下的实时性决策要求。深度强化学习（Deep Reinforcement Learning，DRL）的出现为解决复杂环境下的决策问题提供了一种新的途径。智能体通过与环境进行不断的交互，并结合深度神经网络（Deep Neural Network，DNN）的高维感知能力与强化学习（Reinforcement Learning，RL）的决策能力，其可以自动学习多种状态下应当选取的最优策略对应的动作组合，从而最大化所得到的累积奖励[13]。

近年来，DRL 已经在求解边缘云环境下一些复杂决策问题领域受到大量的关注，但研究者们还没有完全发掘其应用潜力。Tuli 等[14]于 2022 年提出了 A3C-R2N2（Asynchronous-advantage-actor-critic and Residual Recurrent Neural Network）模型，该模型通过捕获工作负载的动态和资源特征，缓解了异构云边环境下的任务迁移和资源分配问题。然而，该模型中 A3C 仍然存在训练速度慢、训练不稳定的现象。此外，R2N2 仅能够提取任务的时域依赖性特征，忽略了整个网络的拓扑结构对系统性能的影响。

因此，研究一种自适应、高效的动态调度策略具有重要意义。本文针对云边混合计算环境中的不确定性、动态性和复杂性，为了有效解决动态云边混合环境下的任务迁移与资源分配问题，提出了一种结合基于最大熵框架[15]的策略梯度强化学习（Soft Actor Critic，SAC）和图卷积网络（GCN）[16]的优化算法，即 SAC-GCN，以最小化任务的平均能耗、响应时间、任务迁移时间和服务等级协议（SLA）[17]的违约比例。本文的主要贡献如下：

1）提出的 SAC-GCN 模型中，SAC 通过策略函数输出动作的最大熵，使模型的探索更加均匀，能够在变化的随机环境中更具鲁棒性。另外，GCN 能够快速有效地提取边缘云网络的结构特征和不同特征之间的高度非线性模式，从而提高模型的学习速度。

2）设计的自适应损失函数使得 SAC-GCN 模型可以动态调整调度模型，以适应任务的不同需求。

3）实验结果表明，与目前较为主流的方法相比，提出的 SAC-GCN 能够显著降低任务的平均能耗、响应时间、任务迁移时间和 SLA 违约比例。

2 相关工作

云任务调度算法通常来说可以分为 3 类：经典的任务调度算法、启发式智能任务调度算法以及基于机器学习的任务调度算法。

经典的任务调度算法如轮询调度算法[9]、最大最小调度算法[18]等适用范围广泛、逻辑简单，在某些场景下可以获得较好的性能，但是因为其针对的任务类型和使用场景比较单一，不能适用于灵活多变的任务场景和优化特定的参数。

启发式任务调度算法，如粒子群优化算法[19]、遗传算法和模拟退火算法[12]等，虽然可以针对特定的目标进行优化，且在一般情况下都表现得很好，但是算法实现比较复杂，而且不适合低延迟要求的在线任务调度。边缘计算中，减少延迟一直是一个重要的优化目标。

常见的基于机器学习的调度算法有基于深度学习、基于强化学习以及结合了二者的基于深度强化学习的算法。这类算法不仅能优化不同的参数，而且自适应能力强，能够学习不同参数之间复杂的依赖关系，还能满足在线任务的调度高效需求，因此在云计算环境中得到了广泛的应用。例如，Xue 等[20]构建异构测控网络资源调度模型，并利用改进的 DQN 算法求解了网络资源调度问题。文献[21]针对云边协同计算中存在依赖关系的任务调度问题，并为了解决移动设备资源受限问题，提出了一种基于强化学习的无模型方法，有效地缩短了应用程序的执行时间。文献[22]提出了一种时间注意力确定性策略梯度，相比原始 DDPG 实现了更快更稳定的收敛。但是，这些都没有考虑边缘-云环境中设备的网络结构信息，因此 Chen 等[23]引入了图卷积网络（GCN），提出了一种无模型的 GCN-DDPG，实现了在长期平均加权成本方面的提升。文献[14]研究了动态的边缘云计算环境下任务卸载和资源分配问题，通过分散训练 A3C-R2N2，来提升设备的能耗、相应时间和违约率。

3 系统模型与问题定义

3.1 网络模型

如图 1 所示，该架构模型中采用“云-边-端”的计算系统结构，具体如下。

第一层是用户端，用户会产生各类应用请求，并将应用以任务的形式暂存在用户端的任务接收器上，等待进一步响应。这里设置的用户端的任务包含 CPU、RAM、带宽和磁盘需求以及其预期的完成时间或截止时间。关于任务模型的描述，详见 3.2 节。

第二层是边缘端，靠近数据源。边缘计算节点连接用户端，提供计算和存储功能。边缘节点之间可以互相通信，我们在边缘端放置 DRL 模型代理服务器，该服务器上的资源管理系统包括：资源监控服务、深度强化模型、限制条件模型、分配和迁移模型以及任务接收器。以上各模块的功能以及交互流程在将 4.2 节进行详细介绍。DRL 模型代理服务器可以单独地在边缘节点上运行，对用户的任务计算请求做出及时的响应。

第三层是云端，一般而言，云计算中心的计算和存储能力强于边缘节点，提供高聚合度的集中计算、存储等服务。在云端我们主要对 DRL 模型进行更新，通过资源监控服务与深度

强化模型的交互来对任务进行合理的迁移和资源分配工作，以便更好地维持服务质量。

如图 1 所示，位于第二层边缘端和第三层云端的主机都分配了任务，本文中任务的分配和迁移主要集中在边缘端和云端，我们通过在边缘端放置 DRL 模型代理服务器来调整任务是否需要迁移。例如，当任务 2 和任务 5 所分配的主机与 DRL 模型的策略不同时，按最新策略将任务 2 迁移至主机 1，将任务 5 迁移至主机 n。

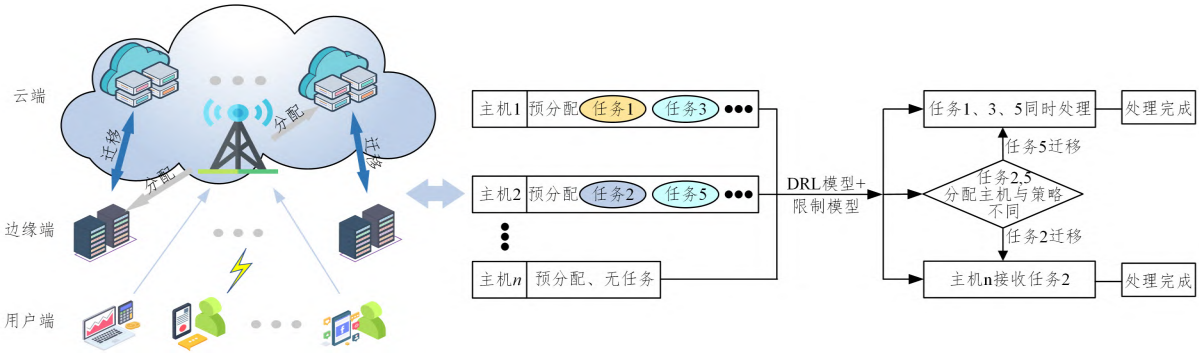

图 1　系统模型

Fig. 1　System model

3.2　任务模型

根据物联网设备的实际用户需求和移动性，任务序列是动态变化的。因此，我们划分了若干个调度间隔，每个调度间隔的持续时间相同，但是每个调度间隔所接收的任务是不同的。

图 2 给出了第 $i-1$ 调度间隔到第 i 调度间隔中的任务变化。设 $T^{T_{i-1}}$ 为第 $i-1$ 调度应该执行的总任务。由于任务的计算和带宽需随着时间而变化，即任务存在当前调度间隔没有被执行完成的风险。$T^{l_{i-1}}$ 表示第 $i-1$ 调度间隔已经完成的任务，因此第 $i-1$ 调度间隔剩余任务 $T^{r_i}=T^{T_{i-1}}-T^{l_{i-1}}$。在第 i 个时间段加载的新任务用 T^{a_i} 表示，上一调度间隔 $i-1$ 剩余的未完成任务用 T^{r_i} 表示，对于第 i 个调度间隔的任务，我们用 $T^{T_i}=\{t\mid T^{r_i}\cup T^{a_i}\}$ 表示。本文中符号的意义如表 1 所列。

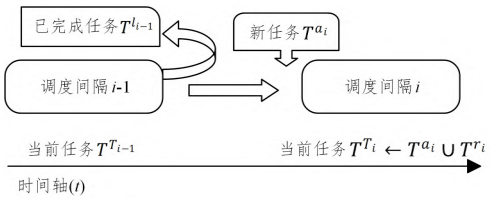

图 2　任务加载

Fig. 2　Task loading

表 1　符号表

Table 1　Symbols

符号	意义
T	任务调度间隔的集合
T^{a_i}	第 i 个调度阶段的新任务
T^{r_i}	来自第 $i-1$ 阶段的未完成的任务
T^{m_i}	第 i 个调度阶段的迁移的任务
T^{l_i}	第 i 个调度阶段的完成的任务
$T_j^{T_i}$	第 i 个调度阶段任务中的第 j 个任务
$T_j^{a_i}$	第 i 个调度阶段新任务中的第 j 个任务
$T_j^{r_i}$	来自第 $i-1$ 阶段未完成的任务中的第 j 个任务
H_n	主机集群中第 n 个主机
FV_i^s	第 i 个调度阶段 s 集合对应的特征向量
$Action_i^{DRL}$	第 i 个调度阶段调度器直接输出的调度策略
$Loss_i^{DRL}$	第 i 个调度阶段的损失函数
H^k	某任务的主机偏好排行第 k 位的主机

3.3　问题形成

优化目标是在降低能量损耗的同时，缩短在各个时间段的任务响应时间和任务迁移时间，减小 SLA 违约比例，因此本文定义了优化函数。

在优化目标中我们取分母为每一轮次的最大值，随着实验的进行，不断地调整整个系统模型中的能耗、响应时间、迁移时间和 SLA 违约率的最大值，因此该优化函数能不断地适应实验环境，从而更好地感应优化函数的变化。

平均能耗(AEC)为基础设施(包括所有边缘和云主机)的平均能耗，$\int_{t_i}^{t_{i+1}}P_h(t)\,\mathrm{d}t$ 为该调度间隔中活动主机的能耗，$t_i\to t_{i+1}$ 为一个调度间隔，$P^{\max}$ 为所有任务调度间隔 T 中的最大能量损耗。

$$AEC_i=\frac{\sum_{h\in Host}\int_{t_i}^{t_{i+1}}P_h(t)\,\mathrm{d}t}{P^{\max}} \tag{1}$$

平均响应时间(ART)为该调度间隔中完成任务 T^{l_i} 的平均响应时间；$ResponseTime(t)$ 为任务响应时间，即主机(处理该任务的主机)的响应时间。$RT^{\max}$ 为所有任务调度间隔 T 中的最大响应时间。

$$ART_i=\frac{\sum_{t\in T^{l_i}}ResponseTime(t)}{RT^{\max}} \tag{2}$$

平均迁移时间(AMT)为该调度间隔中所有任务 T^{T_i} 的平均迁移时间，$MigrationTime(t)$ 为任务的迁移的时间，$MT^{\max}$ 为所有任务调度间隔 T 中的最大迁移时间。

$$AMT_i=\frac{\sum_{t\in T^{l_i}}MigrationTime(t)}{MT^{\max}} \tag{3}$$

平均 SLA 违约比率为调度区间内完成的任务 SLA 违规的平均比例，如式(4)所示，任务 T 的 $SLA(t)$ 在文献[17]中定义，它的两个指标的乘积如下：1)每个活动主机的 SLA 违反时间；2)迁移导致的性能下降。因此：

$$SLAV_i=\frac{\sum_{t\in T^{l_i}}SLA(t)}{SLA^{\max}} \tag{4}$$

其中，$SLA^{\max}$ 为所有任务调度间隔 T 中的最大违约比例。

本文动态任务迁移与资源分配的优化目标为最小化边缘

云系统中的平均任务完成能耗、平均任务响应时间、平均任务迁移时间以及平均任务 SLA 违约率，表达式为：

$$Loss_i^{\mathrm{DRL}}=\alpha\cdot \mathrm{AEC}_i+\beta\cdot \mathrm{ART}_i+\gamma\cdot \mathrm{AMT}_i+\delta\cdot \mathrm{SLAV}_i \tag{5}$$

其中，$\alpha,\beta,\gamma,\delta\geqslant 0$，$\alpha+\beta+\gamma+\delta=1$。

基于不同的用户 QoS 要求和应用程序预先设置好不同的超参数值，以供神经网络进行学习。对于能耗敏感的应用，将 α 设置为 1，其余为零。对于响应时间要求较高的应用，将 β 设置为 1，其余为零。类似地，对于不同的应用程序，需要一组不同的权重参数值。

本文第 4 章将此优化目标与提出的深度强化模型的损失函数相结合，设置自适应损失函数，以此来动态调整调度模型从而适应任务的不同需求。

4 SAC-GCN 模型

4.1 SAC 模型关键要素

4.1.1 状态

对于调度器模型的状态输入，将其定义为 $state_i$，由 $(FV_i^{T_i},FV_i^{AM_i},FV_i^{Host})$ 组成。

$FV_i^{T_i}$ 为第 i 调度间隔的任务 T^{T_i} 特征。任务 T^{T_i} 由新任务 T^{a_i} 和正在执行的任务 T^{r_i} 组成，记 $T^{T_i}=(T^{a_i},T^{r_i})^{\mathrm{T}}$。其中 $T^{a_i}=(T_1^{a_i},T_2^{a_i},\cdots,T_r^{a_i})^{\mathrm{T}}$，$r$ 为新任务数，$T_i^{r_i}=(T_1^{r_i},T_2^{r_i},\cdots,T_q^{r_i})^{\mathrm{T}}$，$q$ 为正在执行的任务数，对于 $T_i^{a_i}$ 第 i 个任务 $T_i^{a_i}=(f_i^{T1},f_i^{T2},\cdots,f_i^{Ts})$ 和 $T_j^{r_i}$ 第 j 个任务 $T_j^{a_i}=(f_j^{T1},f_j^{T2},\cdots,f_j^{Ts})$，皆由 s 个特征项组成，各项参数为 CPU 的要求、RAM 的大小需求以及任务分配的主机索引（采用 one-hot 向量）等，这里的主机索引为上一个调度间隔的动作策略，并作为下一个调度间隔下任务是否需要迁移的参考标记。

$FV_i^{AM_i}$ 为边缘云环境各台主机关系的邻接矩阵，对于边缘云环境各台主机关系的邻接矩阵，当任务可以分配到该主机时，记为 1，否则为 0。

FV_i^{Host} 为第 i 个调度时间段各台主机的资源利用情况的特征矩阵。

$$FV_i^{\mathrm{Host}}=\begin{matrix}H_1\\H_2\\\vdots\\H_n\end{matrix}\begin{bmatrix}f_1^{H1}&f_1^{H2}&\cdots&f_1^{Hm}\\f_2^{H1}&f_2^{H2}&\cdots&f_2^{Hm}\\\vdots&\vdots&\vdots&\vdots\\f_n^{H1}&f_n^{H2}&\cdots&f_n^{Hm}\end{bmatrix}$$

其中，f_a^{Hb} 表示编号为 a 的主机的第 b 个特征值。

$FV_i^{AM_i}$ 用于 Actor 网络中的邻接矩阵，FV_i^{Host} 和 $FV_i^{T_i}$ 拼接成神经网络的输入数据，通过第 i 个调度时间段开始时的预分配，将各个任务分配到相应的主机上，用 FV_i^{Input} 表示。

$$FV_i^{\mathrm{Input}}=\begin{matrix}H_1\\H_2\\\vdots\\H_n\end{matrix}\begin{bmatrix}f_1^{H1}&f_1^{H2}&\cdots&f_1^{Hm}&f_1^{T1}&f_1^{T2}&\cdots&f_1^{Ts}\\f_2^{H1}&f_2^{H2}&\cdots&f_2^{Hm}&f_3^{T1}&f_3^{T2}&\cdots&f_3^{Ts}\\\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots\\f_n^{H1}&f_n^{H2}&\cdots&f_n^{Hm}&f_{q+r}^{T1}&f_{q+r}^{T2}&\cdots&f_{q+r}^{Ts}\end{bmatrix}$$

其中，$q+r\leqslant n$，r 为新任务数，q 为正在执行的任务数，不足 n 个任务，以零作为填充。由于边缘云设置中的任务加入了计算能力、内存大小和输入输出的限制，以上参数对于调度决策至关重要。对于不同的主机，拥有不同的主机参数，同时在任务的分配上，允许将多个任务放置在同一台主机上，以确保较低的能耗（使没有任务的主机进行休眠）。对于具有较高 I/O 容量的主机可以允许快速完成 I/O 密集型的任务，并防止违反 SLA。

4.1.2 动作

在第 i 个时间间隔开始处，该模型为新任务 $T_i^{a_i}$ 提供主机分配，基于上一时间段的任务分配图 FV_{i-1}^{map}，对任务进行主机预分配。主机分配必须要在能满足任务需求的同时，不超过主机可行性限制。该模型在通过 $state_i$ 后输出一个 $Action_i^{DRL}$（即 FV_i^{map}），对于 $Action_i^{DRL}$，如果当前任务分配的主机不是策略 FV_i^{map} 对应的最佳主机，则将该任务进行迁移。我们将需要进行迁移的任务记为 $T_i^{m_i}$。

FV_i^{map} 为尺寸大小为 $n\times n$ 的任务分配图，其中 n 表示为总主机的个数，H_a^b 表示编号为 $T_a^{T_i}$ 任务主机偏好排行第 b 位的主机。

$$FV_i^{map}=\begin{matrix}T_1^{T_i}\\T_2^{T_i}\\\vdots\\T_n^{T_i}\end{matrix}\begin{bmatrix}H_1^1&H_1^2&\cdots&H_1^n\\H_2^1&H_2^2&\cdots&H_2^n\\\vdots&\vdots&\vdots&\vdots\\H_n^1&H_n^2&\cdots&H_n^n\end{bmatrix}$$

4.1.3 奖励

动作带来的开销越小，奖励函数就会越大，由于为迁移任务进行主机分配时，会出现主机无法满足任务需求的情况，因此加入了一个限制条件模型，对 $Action_i^{DRL}$ 进行约束，从而使得任务能正常进行处理。因此最终的 $Action_i$ 为当任务 $T_j^{T_i}$ 需要迁移至主机 H_j^k 时，首先需要满足 $T_i^{T_i}\in T_i^{r_i}\cup T_i^{a_i}$，并且主机 H_j^k 能够满足任务所需的要求，并保证任务 $T_i^{T_i}$ 的任务分配图在主机 H_j^k 以前的主机被合理利用。

此外，为了使得任务分配图能够更好地降低能耗的损失以及迁移时间，本文加入了迁移惩罚机制，将其定义为 $penalty_i$。

$$penalty_i=\frac{\sum_{t\in T^{m_i}} k-1\mid H^k=Action_i^{DRL}(t)}{\mid T_i^{m_i}\mid} \tag{6}$$

因此，对于输出的 $Action_i^{DRL}$，本文需要通过限制条件模型和迁移惩罚机制来进一步优化输出结果。对于神经网络模型的输出是无约束的，因此如果在最终奖励函数 $Reward_i$ 中加入式(6)来更新神经网络的参数，不仅能使 $Reward_i$ 最大化，同时能更好地提取环境中的特征。因此，定义神经网络的奖励如式(7)所示：

$$Reward_i=-(Loss_i^{DRL}+Penalty_i) \tag{7}$$

4.2 使用 SAC-GCN 模型动态调度

图 3 给出了提出的 SAC-GCN 与边缘云网络交互工作的流程。在每个调度间隔开始时：1）通过任务接收器接收新的任务，包括任务参数，如计算、带宽和 SLA 要求；2）对于新的任务，分配和迁移服务通过 DRL 模型输出的任务偏好图通知用户/物联网设备将其请求直接发送到为此任务安排的相应的边缘/云设备；3）通过限制条件模型，利用任务分配图和限制条件对需要迁移的任务和目标主机进行记录；4）利用迁移服务对任务进行迁移。

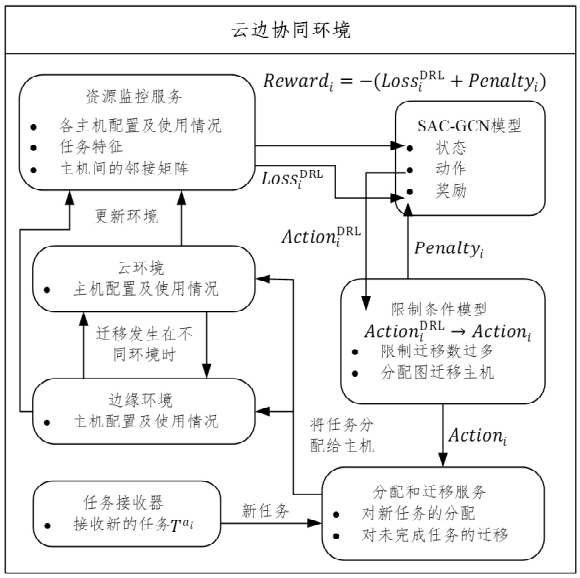

图 3　SAC-GCN 与边缘云网络交互工作的流程

Fig. 3　Workflow of SAC-GCN interacting with edge-cloud network

4.2.1　神经网络结构

本文使用 GCN 网络来近似在每个间隔 SI_i 从 $State_i$ 到 $Action_i^{DRL}$ 的转换过程。图神经网络是一种多层神经网络，利用谱聚类的思想，经传统的离散卷积应用在图结构数据上。神经网络结构使用 GCN 网络的优点是，它能够提取全局图结构数据结构特征，在对特征学习时能够综合考虑来自邻居实体的所有信息。SAC-GCN 的模型架构如图 4 所示。

图结构可以用 $G=(V,E)$ 表示，其中节点 $v\in V$，边 $e\in E$。节点初始特征 $\boldsymbol{X}=\boldsymbol{H}^0\in\boldsymbol{R}^{d_0}$，其中 d_0 代表节点的特征维度。模型中节点 v 的第 l 层的隐藏向量可表示为 $\boldsymbol{H}^{(l+1)}\in\boldsymbol{R}^{d_{l+1}}$，其中 d_{l+1} 代表第 $l+1$ 层的向量的维度。当附近主机资源足够并未过载时，此时该任务所在的主机与附近主机之间的连接关系即为 1，反之为 0，于是 50 个主机构成 $50*50$ 向量的邻接矩阵 $\boldsymbol{A}$。$\tilde{\boldsymbol{A}}$ 是有自连的邻接矩阵，即 $\tilde{\boldsymbol{A}}=\boldsymbol{A}+\boldsymbol{I}$，$\boldsymbol{I}$ 是单位矩阵。$\tilde{\boldsymbol{D}}$ 是 $\tilde{\boldsymbol{A}}$ 的度矩阵，即 $\tilde{\boldsymbol{D}}_{ii}=\Sigma_j\tilde{\boldsymbol{A}}_{ij}$，$W^l$ 为第 l 层训练得到的权重参数矩阵，σ 是非线性激活函数。

GCN 学习方式遵循逐层学习，所有节点同步更新。每层学习过程可分解为两个步骤：传递和聚合。其过程如式(8)所示：

$$H^{(l+1)}=\sigma(\tilde{D}^{-1}\tilde{A}H^{(l)}W^{(l)}) \tag{8}$$

GCN 网络由 2 个卷积层连接而成，输入为一个二维的 $50*66$ 向量的主机特征 FV_i^{Host}，接着经过两层卷积层后，将输出特征与任务特征 $FV_i^{T_i}$ 拼接并一维化，再通过带有残差连接的三层全连接，然后输出一个二维的 $50*50$ 向量。这意味着该模型最多可以管理 50 个任务。最后，在第二个维度上应用 softmax，根据输出值的大小可以直接对主机偏好值进行排序。

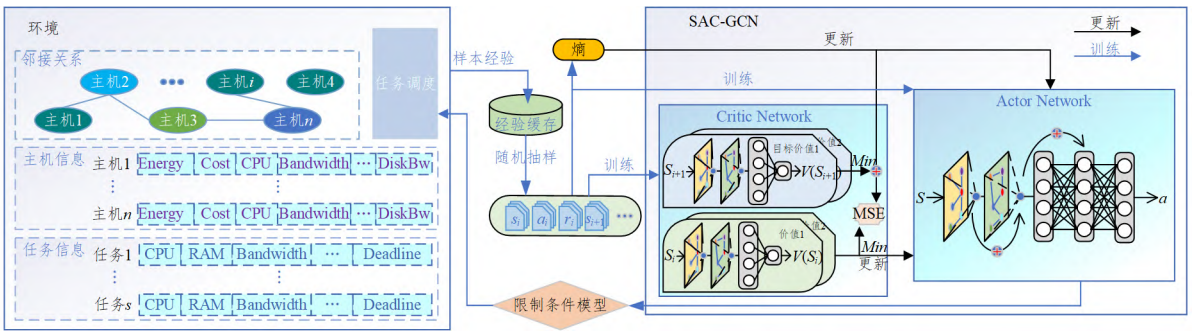

图 4　基于 SAC-GCN 模型的架构图

Fig. 4　Architecture diagram based on SAC-GCN

4.2.2　Soft Actor-Critic 算法

SAC 是基于最大熵(Maximum Entropy)这一思想发展的强化学习算法。SAC 在优化策略以获取更高累计回报的同时，也会最大化策略的熵[24]。

在实验过程中，本文选择构造一个 GCN 网络来近似 SAC 算法中的策略函数 $\pi_\theta(a_i\mid s_i)$，以生成智能体行动的策略。网络的输入是当前环境的所有状态信息，包括环境中的任务参数、主机参数和邻接矩阵，输出是一个二维的 $50*50$ 的矩阵，即每一个任务选择的主机偏好值排序，并从中选择最优主机。

在实验中，为了评判策略函数的优劣，对智能体行动的策略分别构造了两个当前价值网络 Q_{φ_1} 和 Q_{φ_2}，两个目标价值网络 $Q_{\varphi_1'}$ 和 $Q_{\varphi_2'}$。

Soft Q 当前网络的损失函数如下：

$$J_Q(\varphi)=E_{(s_i,a_i)\sim D}[Q_\varphi(s_i,a_i)-(r_i+\omega V(s_{i+1}))]^2 \tag{9}$$

4.2.3　策略学习

为了学习 SAC-GCN 算法框架中的神经网络的权值和偏差，我们使用了奖励为 $Reward_i$(即 r_i)的反向传播算法。调度开始时，SAC-GCN 模型接受来自资源监控服务的信息进行处理，将其作为智能体的当前状态 $state_i$(即 s_i)传递到 Actor 网络。Actor 网络输出的是一个二维的 50×50 向量，表示 50 个任务分别对应的 50 个主机的偏好值排序。通过限制条件模型将输出 $Action_i^{DRL}$ 转换为 $Action_i$(即 a_i)并在每个调度间隔发送到分配和迁移服务。分配和迁移服务接受通过限制模型处理的新任务分配和上一阶段未完成任务的迁移。因此对于每个调度间隔，SAC-GCN 模型都有一个正向传递。对于反向传播，为了充分利用与环境交互的经验信息，本文采用对经验缓冲区进行随机抽样的方法，得到批次大小为 B 的三元组 $\{s_i,a_i,Reward_i,s_{i+1}\}$，用于网络参数的更新。Soft Q 网络 Q_{φ_1} 和 Q_{φ_2} 以 s_i 和 a_i 为网络的输入，计算出当前价值网络输出值 $Q_{\varphi_1}(s_i,a_i)$ 和 $Q_{\varphi_2}(s_i,a_i)$，Actor 网络接收 s_{i+1}，然后输出策略 a_{i+1} 并计算该策略的熵，$Q_{\varphi_1'}$ 和 $Q_{\varphi_2'}$ 以 s_{i+1} 和 a_{i+1} 为输入，计算目标价值 $Q_{\varphi_1'}(s_{i+1},a_{i+1})$ 和 $Q_{\varphi_2'}(s_{i+1},a_{i+1})$，以此来更新网络 π_θ，Q_{φ_1}，Q_{φ_2}，$Q_{\varphi_1'}$ 和 $Q_{\varphi_2'}$ 的参数。在实验中，Critic 网

络的损失函数为：

$$J_Q(\varphi)=E_{(s_t,a_t)\sim D}\left[\frac{1}{2}(Q_\varphi(s_t,a_t)-y)^2\right] \tag{10}$$

其中，$y=Reward_i+\omega E_{a_{i+1}\sim\pi_\theta}[Q_{\varphi'}(s_{i+1},a_{i+1})-\lambda\log(\pi_\theta(a_{i+1}|s_{i+1}))]$，$D$ 为经验缓存池。φ' 是目标 critic 网络 $Q_{\varphi_1'}$ 和 $Q_{\varphi_2'}$ 的最小值。λ 为温度熵系数，根据环境自动调节。更新 Actor 网络需要最小化目标 $J_\pi(\theta)$：

$$J_\pi(\theta)=E_{(s_i,a_i)\sim D}[E_{a_i\sim\pi_\theta}[\lambda\log(\pi_\theta(a_i|s_i))-Q_\varphi(s_i,a_i)]] \tag{11}$$

其中，φ 是当前 critic 网络 Q_{φ_1} 和 Q_{φ_2} 的最小值。对于目标 critic 网络 $Q_{\varphi_1'}$ 和 $Q_{\varphi_2'}$ 的软更新，我们使用 3 个调度间隔进行参数更新。每 3 个调度间隔之后，使用当前 critic 网络的最新参数进行软更新，用于保证算法的稳定进行。

$$\varphi_1'=\tau\varphi_1+(1-\tau)\varphi_1' \tag{12}$$

$$\varphi_2'=\tau\varphi_2+(1-\tau)\varphi_2' \tag{13}$$

其中，τ 为控制目标网络软更新的超参数。最后，将上一个调度间隔的状态 s_{i+1} 作为下一调度间隔的状态 s_i。实验设置的调度间隔为 5 min，并且每模拟 5 min 进行一次软更新。

算法 1 总结了具有反向传播的模型更新和调度方法。为了确定每个调度间隔的最佳调度决策，本文迭代地对间隔状态进行预处理，并发送到具有损失和惩罚的 SAC-GCN 模型，以更新网络参数。这使得模型能够实时地适应环境、用户和应用程序的特定需求。

算法 1　SAC-GCN 算法

输入：云环境当前状态 $(FV_i^{T_i},FV_i^{AM_i},FV_i^{Host})$

输出：任务分配图 FV_i^{map}

1. 设置调度间隔数为 N；
2. 设置软更新间隔次数 M；
3. 设置批次大小为 B；
4. 随机初始化参数向量 $\theta,\varphi_1,\varphi_2,\varphi_1',\varphi_2'$；
5. for 调度间隔 i=1 to N do：
6. 　发送 $State_i$ 到 SAC-GCN 模型；
7. 　SAC-GCN 模型输出 $Action_i^{DRL}$；
8. 　网络 Q_{φ_1} 和 Q_{φ_2} 计算输出值 $Q_{\varphi_1}(s_i)$ 和 $Q_{\varphi_2}(s_i)$；
9. 　目标网络 $Q_{\varphi_1'}$ 和 $Q_{\varphi_2'}$ 计算输出值 $Q_{\varphi_1'}(s_i)$ 和 $Q_{\varphi_1'}(s_i)$；
10. 　反向传播更新网络参数：

$$\varphi_1\leftarrow\varphi_1-\lambda_Q\hat{\nabla}_{\varphi_1}J_Q(\varphi_1)$$

$$\varphi_2\leftarrow\varphi_2-\lambda_Q\hat{\nabla}_{\varphi_2}J_Q(\varphi_2)$$

$$\theta\leftarrow\theta-\lambda_\pi\hat{\nabla}_\theta J_\pi(\theta)$$

11. 　if i % M==0 then：
12. 　软更新目标网络参数：

$$\varphi_1'=\tau\varphi_1+(1-\tau)\varphi_1'$$

$$\varphi_2'=\tau\varphi_2+(1-\tau)\varphi_2'$$

13. 　end if
14. end for
15. 限制条件模型转换 $Action_i^{DRL}$ 为 $Action_i$；
16. 资源监控服务输出 $Penalty_i$；
17. 基于 $Action_i$ 分配新任务和迁移存在的任务；
18. 在边缘云基础设施中执行间隔 SI_i 的任务；
19. end for

在算法训练阶段，调度模拟开始时初始化一个经验回放缓存，用于存储动作与环境之间的交互效果。在训练过程中不断采用贪心算法选择最优动作组合，对神经网络进行训练。

5　仿真实验

本章将描述实验装置、评价指标、数据集，并给出了本文模型与几种基线算法的详细比较结果。

5.1　实验环境及配置

为了评估提出的基于 SAC-GCN 算法的调度框架，本文使用了 CloudSim4.0[25] 的模拟环境，并对实验环境进行了完善。实验间隔的大小为 5 min，与其他工作中的文献[26-28]相同，以便于与基线算法进行公平的比较。在本实验中设置了 50 个边缘云节点，用于模拟现实环境，并设置了最大任务数 50。其中边缘云节点的主机种类为 4 类，其中 2 类为边缘端主机，剩余 2 类为云端主机。为了减小计算复杂度，考虑边缘节点的响应时间为 1 ms，云节点的响应时间为 10 ms，实验中主机参数如表 2 所列。以此在相同的环境下，对比不同的任务调度算法。具体各算法的超参数设置如表 3 所列。

基于 Cloud-Sim 中的主机和任务监控服务计算损失函数。惩罚由限制条件模型计算，并发送到深度强化学习模块进行模型参数更新。下文将更详细地描述数据集、任务生成-评估和持续时间的实现、主机的配置和用于评估的指标。

通过比较模型中不同的神经网络，分析图神经网络在边缘环境中移动设备构成的拓扑结构下适用的优越性。

实验平台的硬件配置具体包括：CPU 为双路 E5-2689 处理器 16 核 32 线程，RAM 为 64 GB，2 张 GTX1070 显卡。使用的操作系统为 Ubuntu，还包括一些软件环境：Python3.9.13，深度学习框架 Pytorch，集成开发环境 JetBrains IDEA 2022。

表 2　主机配置

Table 2　Host configuration

	Name	Processor	Core	RAM/GB	MIPS	SPEC Power(Watts) for different CPU percentage usages										
						0	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Edge Layer	Hitachi HA 8000	Intel i3 3.0 GHz	2	8	1800	24.3	30.4	33.7	36.6	39.6	42.2	45.6	51.8	55.7	60.8	63.2
Edge Layer	DEPO Race X340H	Intel i5 3.2 GHz	4	16	2000	83.2	88.2	94.3	101	107	112	117	120	124	128	131
Cloud Layer	Dell PowerEdge R820	Intel Xeon 2.6 GHz	32	48	2000	110	149	167	188	218	237	268	307	358	414	446
Cloud Layer	Dell PowerEdge C6320	Intel Xeon 2.3 GHz	64	64	2660	210	371	449	522	589	647	705	802	924	1071	1229

表 3　各算法超参数设置

Table 3　Hyperparameter settings of each algorithm

超参数	A3C-R2N2	SAC-GCN
能耗缩放因子 α	0.25	0.25
响应时间缩放因子 β	0.25	0.25
迁移时间缩放因子 γ	0.25	0.25
SLV 违约缩放因子 δ	0.25	0.25
批次大小 B	—	32
折扣因子 ω	0.95	0.95
策略网络学习率	0.001～0.01	0.001
价值网络 1 学习率	—	0.001
价值网络 2 学习率	—	0.001
软更新因子 τ	—	0.05
间隔大小/min	5	5
任务个数	50	50
主机个数	50	50
Episode 大小	12	—
训练调度间隔数 N	1 440	1 440
测试调度间隔数 n	288	288

5.2　数据集

在模拟环境中，首先在数据中心创建主机，然后数据代理将创建的任务与创建的虚拟机进行绑定，最后将虚拟机分配给主机。任务处理完成时，先在虚拟机上销毁云任务，然后再丢弃虚拟机。其中动态工作负载是开源 Bit-brain 数据集[29]生成的。

为了实验环境更加接近真实世界的基础设施环境，提高实验的准确性，我们选择了 Bit-brain 数据集。因为其拥有在 Bit-brain 基础设施上托管的业务关键型工作负载的资源消耗指标的真实痕迹，这些数据包括了托管在两种不同类型机器上的超过 1 000 个虚拟机工作负载的日志。数据集中包括每个调度间隔的主机和任务特征，如任务所需的 RAM 大小、请求的 CPU 核数量、所需 MIPS 的大小、对应主机的 CPU 使用情况等。本实验利用这些任务来模拟实验环境，通过判断任务的 RAM 大小和对 CPU 的需求等参数，来对任务进行主机分配。确保任务能在目标主机上正常运行的同时，由于在该数据集上任务的各个参数在每个调度间隔中都在不停变化，因此会产生不断变化的能耗、响应时间、迁移时间和 SLA 违约率等。通过 SAC-GCN 调度器对各个优化指标的学习，来寻找较好的分配决策。

5.3　评价指标

为了评估提出的基于 SAC-GCN 的调度器的有效性，本文考虑了以下指标：平均系统能耗、平均任务响应时间、平均任务迁移时间、平均任务 SLA 违约率。

5.4　系统超参数的灵敏度分析

本文首先分析了优化目标中超参数(α，β)对模型学习的敏感性。通过设置不同的比例组合，并基于 Bit-brain 数据集进行了 5 天的模拟时间训练各个模型，并使用了一组与训练数据集不相交的工作负载测试了 24 h，调度间隔为 5 min。

如图 5 所示，为减小实验复杂度，在该实验设置中 $\alpha+\beta=1$，其余参数 γ，δ 均为 0。优化目标中的超参数会影响调度算法的性能。当能耗所占比例 α 在优化目标中越大时，SAC-GCN 模型对其优化效果越明显，相应能量损耗越小，这是因为模型为降低能耗，选择了较为集中的动作输出，将任务集中放置在具有较多资源的主机上，因此响应时间也相应增加。

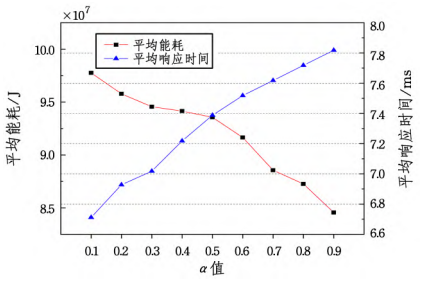

图 5　参数灵敏度分析

Fig. 5　Parameter sensitivity analysis

如图 6 所示，调度器对任务的分配大致较为聚集，这是因为 SAC-GCN 模型对任务调度的同时，需要考虑各个不同任务之间的动作组合，同时在环境初始化时，任务大多随机分配在编号较小的主机上，因此调度器通过学习这些节点信息和任务的特征，将随机到来的任务合理地分配到这些主机上，以确保降低云边环境下的整体能耗、响应时间等，以达到对能耗、响应时间等指标的优化。

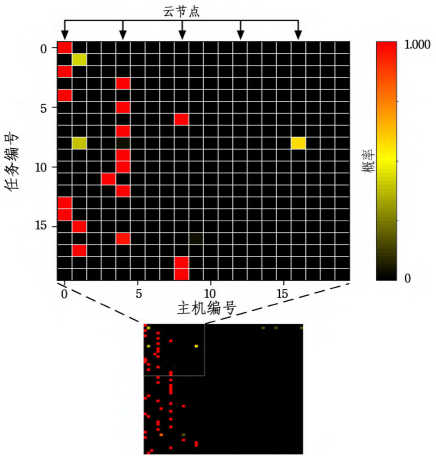

图 6　SAC-GCN 任务偏好图

Fig. 6　SAC-GCN task preference map

5.5　训练收敛对比分析

图 7 给出了提出的 SAC-GCN 与 A3C-R2N2[14]的环境奖励对比。在整个训练模型的实验中，最后 60 个调度间隔 A3C-R2N2 的平均奖励为－1.14，SAC-GCN 模型的平均奖励为－0.98，相比前者，奖励值增加了近 14.4%。这是因为一方面，相比 R2N2 模型，GCN 能够更好地提取主机节点特征，因此在由主机与主机构成的拓扑结构下，可以更好地感知和适应动态变化的环境，从而做出更优秀的调度决策。另一方面，相对于 A3C 算法只给策略网络增加了熵正则，SAC 给价值网络也增加了熵正则，使得模型更具鲁棒性，从而允许模型在网络、工作负载和设备特性的快速变化情况下更加稳定。

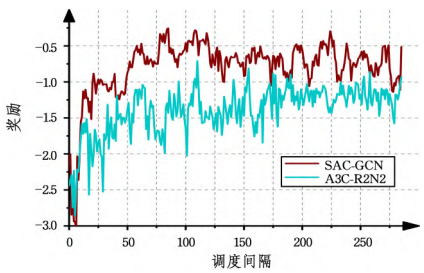

图 7　SAC-GCN 与 A3C-R2N2 的奖励对比

Fig. 7　Comparison of SAC-GCN and A3C-R2N2 rewards

5.6 算法性能对比分析

为了评估提出的 SAC-GCN 算法的有效性，本节对比了以下 6 种算法。

1）LR-MMT[26]：基于本地回归（LR）和最小迁移时间（MMT）的启发式方法，用于过载检测和任务选择并动态安排工作负载。局部回归（LR）和最小迁移时间（MMT）启发式算法。

2）DDQN：标准的基于深度 Q 学习的强化学习方法，我们实现了优化的 Double DQN。

3）REINFORCE[30]：基于政策梯度的采用全连接神经网络的强化学习方法。

4）A3C-R2N2[14]：异构随机云边环境下使用 A3C 和残差递归神经网络进行动态调度。

5）SAC：采用两层全连接替换 GCN 的优化算法。

6）SAC-GCN：本文提出的优化算法。

通过比较模型中不同的神经网络，分析图神经网络在边缘环境中移动设备构成的拓扑结构下适用的优越性。

图 8 给出了 6 种调度算法下任务迁移的能耗对比，可以看出 REINFORCE 模型和 A3C-R2N2 模型的能源消耗较低，而 SAC-GCN 的能耗最低，分别比 REINFORCE 模型和 A3C-R2N2 模型低 9.03%和 4.81%，SAC-GCN 模型相比 SAC 模型降低了 3.13%。其主要原因是 SAC-GCN 模型利用 GCN 可以快速适应任务工作负载，从而能够将一个急需资源的任务调度到一个功能和资源更强大的主机上。此外，损失函数中所有边缘噪声节点的平均能量消耗（AEC）参数的存在迫使该模型做出节能调度决策。因此，在模拟时活动的主机最少，更多的主机处于备用状态，从而节省能量。

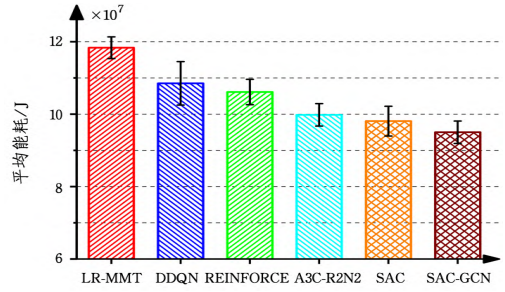

图 8 6 种调度算法下任务迁移的能耗对比

Fig. 8 Comparison of energy consumption for task migration of six scheduling algorithms

图 9 给出了 6 种调度算法下平均响应时间的对比。可以看出，在所有的调度策略中，SAC-GCN 提供的平均响应时间最短，比 A3C-R2N2 模型短 3.46%，比 SAC 模型短了 2.38%。同时，本文还对比了 6 种调度算法下的平均任务迁移时间，即调度时间内所有完成任务的迁移时间，如图 10 所示。SAC-GCN 模型的迁移时间最短，比 A3C-R2N2 模型短 2.73%，比 SAC 模型短了 3.86%，这是因为 GCN 更好地聚合了其他节点信息，使得算法更快地寻找到合理的分配方法。而 A3C-R2N2 通过联系前后调度的规律，因此都相比 SAC 模型有较短的迁移时间。对于时间关键的任务，SAC-GCN 模型能够将其分配给功能更加强大的主机，从而避免后期的迁移，以达到节省迁移时间的目的。因此，SAC-GCN 模型能够缩短任务的平均响应时间以及平均任务迁移时间。

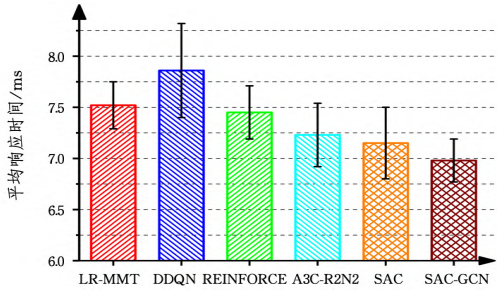

图 9 6 种调度算法下的平均响应时间对比

Fig. 9 Comparison of average response time of six scheduling algorithms

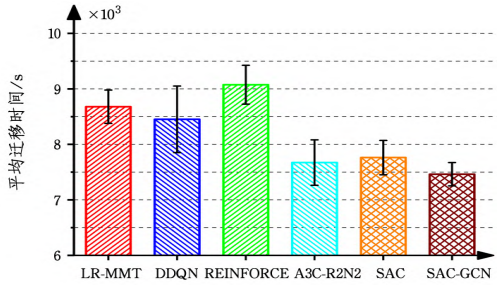

图 10 6 种调度算法下模拟时间内的平均迁移时间

Fig. 10 Average migration time during simulation time of six scheduling algorithms

此外，我们还考虑了模拟时间内总的 SLA 违约率。如图 11 所示，SAC-GCN 模型任务的 SLA 违约率最低，比 A3C-R2N2 模型低了 1.5%，比 SAC 模型低了 2.4%。但从实验结果数据波动来看，SAC-GCN 模型具有较大的波动，这是因为对于多个动作组合的 Q 值学习是十分困难的，神经网络未能较好地拟合各个动作的分数，因此导致实验结果的波动较大。

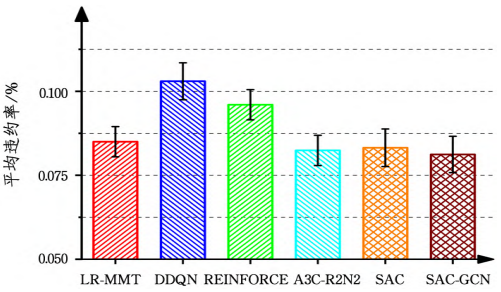

图 11 6 种调度算法下 SLA 的违约率对比

Fig. 11 Comparison of SLA default rate of six scheduling algorithms

结束语 本文研究了边缘云环境中的任务迁移与资源分配问题。本文考虑了边缘设备和云设备的资源和网络在移动时产生的差异性，以及随机工作负载和动态环境的影响，优化目标为在降低能量损耗的同时，缩短在各个时间段的任务响应时间。本文提出了基于 SAC-GCN 端到端实时任务调度算法，该算法中 GCN 能够更好地学习边缘设备间的网络特征关系，能更快地更新网络权重，而 SAC 则通过引入熵和软更新，能够在随机动态变化的环境中更具探索性和鲁棒性，通过考虑主机和云任务较为关键的重要参数来做出适用于随机动态环境的调度策略。本文在基于真实世界的 Bit-brain 数据

集上使用 Cloud-Sim 进行了大量模拟实验。实验结果表明,相比 A3C-R2N2 算法,提出的 SAC-GCN 可以减少4.81%的能源消耗,缩短 3.46%的响应时间和 2.73%的迁移时间,减小 1.5%的 SLA 违规率。

参 考 文 献

[1] MAHMUD R,SRIRAMA S,RAMAMOHANARAO K,et al. Quality of experience(QoE)-aware placement of applications in fog computing environments[J]. Journal of Parallel and Distributed Computing,2019,132:190-203.

[2] GAO H,XU Y,YIN Y,et al. Context-aware QoS prediction with neural collaborative filtering for Internet-of-Things services[J]. IEEE Internet of Things Journal,2020,7(5):4532-4542.

[3] JIA M,LIANG W,XU Z,et al. Qos-aware cloudlet load balancing in wireless metropolitan area networks[J]. IEEE Transactions on Cloud Computing,2020,8(2):623-634.

[4] SHEN H,CHEN L. A resource-efficient predictive resource Provision-ing System in Cloud Systems[J]. IEEE Transactions on Parallel and Distributed Systems,2022,33(12):3886-3900.

[5] HADDADI M,BAHNES N. Well-known open-source cloud computing platforms[C]// Proceed-ings of the International Conference on Infor-mation Systems and Advanced Technologies. Tebessa,Algeria,2021:1-6.

[6] GU L,ZHANG W,WANG Z,et al. Service Management and Energy Scheduling Toward Low-Carbon Edge Computing[J]. IEEE Transactions on Sustainable Computing,2023,8(1):109-119.

[7] HARJULA E,ARTEMENKO A,FORSSTRÖM S. Edge computing for in-dustrial IoT:Challenges and Solutions[M]// Wireless Networks and Industrial IoT. Cham:Springer,2021:225-240.

[8] CHEN Y,LIU B,LIN W,et al. Survey of Cloud-edge Collaboration[J]. Computer Science,2021,48(3):259-268.

[9] CAI X,GENG S,WU D,et al. A multicloud-model-based many-objective intelligent algorithm for efficient task scheduling in internet of things[J]. IEEE Internet of Things Journal,2021,8(12):9645-9653.

[10] ALSULAMI A,AL-HAIJA Q,THANOONM,et al. Performance evaluation of dynamic round robin algorithms for CPU scheduling[C]// Proceedings of the SoutheastCon. Huntsville,AL,USA,2019:1-5.

[11] LIU S,WANG N. Collaborative Optimization Scheduling of Cloud Service Resources Based on Improved Genetic Algorithm[J]. IEEE Access,2020,8:150878-150890.

[12] YUAN H,WANG Y. A Hybrid Schedule Technology Based on Genetic Algorithm and Simulated Annealing for Time-Triggered Ethernet[C]// Proceedings of the IEEE 2nd International Conference on Information Communication and Software Engineering. Chongqing,China,2022:151-155.

[13] JIA J,WANG W. Review of reinforcement learning research[C]// Proceedings of the 35th Youth Academic Annual Conference of Chinese Association of Automation. Zhanjiang,China,2020:186-191.

[14] TULI S,ILAGER S,RAMAMOHANARAOK,et al. Dynamic scheduling for stochastic edge-cloud computing environments using a3c learning and residual recurrent neural networks[J]. IEEE Transactions on Mobile Computing,2022,21(3):940-954.

[15] TUOMAS H,AURICK Z,PIETER A,et al. Soft actor-critic:off-policy maximum entropy deep reinforcement learning with a stochastic actor[C]// Proceedings of the 35th International Conference on Machine Learning. Stockholmsmässan,Stockholm Sweden,2018:1861-1870.

[16] XU B,CEN K,HUANG J,et al. Survey on graph convolutional network[J]. Chinese Journal of Computers,2020,43(5):755-780.

[17] BELOGLAZOV A,BUYYA R. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data-centers[J]. Concurrency and Computation:Practice and Experience,2011,24(13):1397-1420.

[18] ALIYAH Z,PAMBUDHI R,AHNAFI H,et al. Comparison of earliest deadline first and rate monotonic scheduling in polling server[C]// Proceedings of the 8th International Conference on Information and Communication Technology. Yogyakarta,Indonesia,2020:1-4.

[19] GAO M,ZHU Y,SUN J. The multi-objective cloud tasks scheduling based on hybrid particle swarm optimization[C]// Proceedings of the Eighth International Conference on Advanced Cloud and Big Data. Taiyuan,China,2020:1-5.

[20] XUE N,DING D,FAN Y,et al. Research on Joint Scheduling Method of Heterogeneous TT&C Network Resources Based on Improved DQN Algorithm[C]// Proceedings of the 2nd International Conference on Information Technology,Big Data and Artificial Intelligence. Chongqing,China,2021:1009-1014.

[21] HU S,SONG R,CHEN X,et al. Dependency-Task Scheduling in Cloud-Edge Collaborative Computing Based on Reinforcement Learning[J]. Computer Science,2023,50(11A):220900076-8.

[22] CHEN J,XING H,XIAO Z,et al. A DRL agent for jointly optimizing computation offloading and resource allocation in MEC[J]. IEEE Internet of Things Journal,2021,8(24):17508-17524.

[23] CHEN J,WU Z. Dynamic computation offloading with energy harvesting devices:a graph-based deep reinforcement learning approach[J]. IEEE Communications Letters,2021,25(9):2968-2972.

[24] HAARNOJA T,TANG H,ABBEEL P,et al. Reinforcement learning with deep energy-based policies[C]// Proceedings of the 34th International Conference on Machine Learning. Sydney NSW Australia,2017:1352-1361.

[25] SUNDAS A,PANDA S. An introduction of cloudsim simulation tool for modelling and scheduling[C]// Proceedings of the International Conference on Emerging Smart Computing and Informatics. Pune,India,2020:263-268.

[26] SUPREETH S,PATIL K,PATIL S,et al. Comparative approach for VM scheduling using modified particle swarm optimization and genetic algorithm in cloud computing[C]// Proceedings of the IEEE International Conference on Data Science and

Information System. Hassan, India, 2022: 1-6

[27] CHENG M, LI J, NAZARIAN S D R. L-cloud: deep reinforcement learning-based resource pro-visioning and task scheduling for cloud service providers[C]// Proceedings of the 23rd Asia and South Pacific Design Automation Conference. Jeju, Korea (South), 2018: 129-134.

[28] BASU D, WANG X, HONG Y, et al. Learn-as-you-go with megh: efficient live migration of virtual machines[J]. IEEE Transactions on Parallel and Distributed Systems, 2019, 30(8): 1786-1801.

[29] SHEN S, VAN V, IOSUP A. Statistical characterization of business-critical workloads hosted in cloud datacenters[C]// Proceedings of the 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. Shenzhen, China, 2015: 465-474.

[30] MAOH, ALIZADEH M, MENACHE I, et al. Resource management with deep reinforce-ment learning[C]// Proceedings of the 15th ACM Work-shop on Hot Topics in Networks. Atlanta GA USA, 2016: 50-56.

CHEN Juan, born in 1985, Ph. D, associate professor. Her main research interests include cloud computing, mobile-edge computing, and deep reinforcement learning.

CHEN Peng, born in 1979, Ph. D. His main research interests include cloud computing, machine learning, and deep learning.