工业物联网中数字孪生辅助任务卸载算法

唐 伦 单贞贞* 文明艳 李 荔 陈前斌
(重庆邮电大学通信与信息工程学院 重庆 400065)
(移动通信技术重庆市重点实验室 重庆 400065)

摘 要：针对工业物联网(IIoT)设备资源有限和边缘服务器资源动态变化导致的任务协同计算效率低等问题，该文提出一种工业物联网中数字孪生(DT)辅助任务卸载算法。首先，该算法构建了云-边-端3层数字孪生辅助任务卸载框架，在所创建的数字孪生层中生成近似最佳的任务卸载策略。其次，在任务计算时间和能量的约束下，从时延的角度研究了计算卸载过程中用户关联和任务划分的联合优化问题，建立了最小化任务卸载时间和服务失败惩罚的优化模型。最后，提出一种基于深度多智能体参数化Q网络(DMAPQN)的用户关联和任务划分算法，通过每个智能体不断地探索和学习，以获取近似最佳的用户关联和任务划分策略，并将该策略下发至物理实体网络中执行。仿真结果表明，所提任务卸载算法有效降低了任务协同计算时间，同时为每个计算任务提供近似最佳的卸载策略。

关键词：工业物联网；数字孪生；边缘关联；任务划分；深度强化学习
中图分类号：TN929.5 **文献标识码**：A **文章编号**：1009-5896(2024)04-1296-10
DOI: 10.11999/JEIT230317

Digital Twin-assisted Task Offloading Algorithms for the Industrial Internet of Things

TANG Lun SHAN Zhenzhen WEN Mingyan LI Li CHEN Qianbin
(*School of Communications and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing* 400065, *China*)
(*Chongqing Key Laboratory of mobile Communications Technology, Chongqing* 400065, *China*)

Abstract: To address the low efficiency of task collaboration computation caused by limited resources of Industrial Internet of Things (IIoT) devices and dynamic changes of edge server resources, a Digital Twin (DT)-assisted task offloading algorithm is proposed for IIoT. First, the cloud-edge-end three-layer digital twin-assisted task offloading framework is constructed by the algorithm, and the approximate optimal task offloading strategy is generated in the created digital twin layer. Second, under the constraints of task computation time and energy, the joint optimization problem of user association and task partition in the computation offloading process is studied from the perspective of delay. An optimization model is established with the goal of minimizing the task offloading time and service failure penalty. Finally, a user association and task partition algorithm based on Deep Multi-Agent Parameterized Q-Network (DMAPQN) is proposed. The approximate optimal user association and task partition strategy is obtained by each intelligent agent through continuous exploration and learning, and it is issued to the physical entity network for execution. Simulation results show that the proposed task offloading algorithm effectively reduces the task collaboration computation time and provides approximate optimal offloading strategies for each computational task.

收稿日期：2023-04-26；改回日期：2024-02-28；网络出版：2024-03-08
*通信作者： 单贞贞 2804117551@qq.com
基金项目：国家自然科学基金(62071078)，重庆市教委科学技术研究项目(KJZD-M201800601)，川渝联合实施重点研发项目(2021YFQ0053)，贵州省教育厅自然科学研究项目(黔教合KY字[2021]236)
Foundation Items: The National Natural Science Foundation of China (62071078), The Science and Technology Research Program of Chongqing Municipal Education Commission (KJZD-M201800601), Sichuan and Chongqing Key R&D Projects (2021YFQ0053), The Natural Science Project of Guizhou Provincial Department of Education (QJHKYZ[2021]236)

Key words: Industrial Internet of Things (IIoT); Digital Twins (DT); Edge association; Division of tasks; Deep reinforcement learning

1　引言

随着智能机器人、增强/虚拟现实等物联网(Internet of Things, IoT)应用程序的出现，工业物联网(Industrial Internet of Things, IIoT)中对延迟的要求越来越强烈，尤其是计算密集型任务。而IIoT设备的计算能力有限，及时完成这些计算任务具有挑战性。移动边缘计算(Mobile Edge Computing, MEC)是支持延迟敏感型物联网应用的一个有前途的解决方案，在网络边缘部署MEC服务器，允许IIoT设备可将计算任务卸载至附近的微基站(Micro Base Station, MBS)和宏基站(Base Station, BS)中进行计算和处理。

数字孪生(Digital Twin, DT)作为一种6G新兴的技术，可以准确地将物理设备映射为数字孪生模型，以实时反映物理设备的状态变化。通过在DT中创建与物理网络相对应的数字孪生体，实时监控整个网络的运行状态，并向用户直接提供准确、及时的卸载决策，使得物理网络中的协同计算卸载过程更加智能高效[1–3]。文献[4]研究了将无人机作为中继的数字孪生辅助任务卸载问题，建立了总能耗的优化模型。文献[5]提出了一种在数字孪生边缘网络中进行随机计算卸载的方法，将随机计算卸载问题描述为最小化网络效率的优化问题。

然而，上述所使用的DT技术仍面临一些挑战：(1)由于DT的便利性，如何根据网络动态性智能构建DT，以及设置合理的数据偏差值，以准确获取网络的运行状态数据、构建数字模型、辅助任务卸载策略的生成。(2)在多用户多MEC服务器的场景中，每个用户的边缘关联策略对卸载过程存在着重要的影响，如何根据实际边缘服务器(Edge Services, ESs)的剩余资源状态选择最佳的ES是关键问题，而且多个MEC服务器之间是相互影响的，对于用户而言，存在着资源竞争的关系。(3)为了充分利用MBS和BS的资源，学习一种智能的任务划分策略对于提升协同计算效率是至关重要的，然而，用户不同所需求的通信和计算资源不同，且任务划分策略与用户关联策略是相互耦合的，如何联合学习近似最佳的用户关联策略和任务划分策略是当前亟需解决的主要问题。

综上，针对目前数字孪生辅助任务卸载方案中出现的问题，本文的主要贡献如下：

(1)在工业物联网场景中提出一个3层数字孪生辅助任务卸载框架，在本框架中通过构建的数字孪生体可以准确监测物理设备的运行状态信息、实时监测物理网络中的任务请求信息。

(2)在任务计算时间和能量的约束下，从时延的角度研究了计算卸载过程中用户关联和任务划分的联合优化问题，建立最小化任务卸载时间和服务失败惩罚的优化模型，并且同时考虑了独立型子任务和顺序依赖型子任务的任务类型，分别表示了不同子任务卸载时间的计算方式。

(3)考虑到优化问题同时包含离散变量和连续变量，导致动作空间较大，并且在卸载计算的过程中用户关联策略和任务划分策略是相互耦合的。因此，提出一种基于深度多智能体参数化Q网络(Deep Multi-Agent Parameterized Q-Network, DMAPQN)的用户关联和任务划分算法进行联合求解。

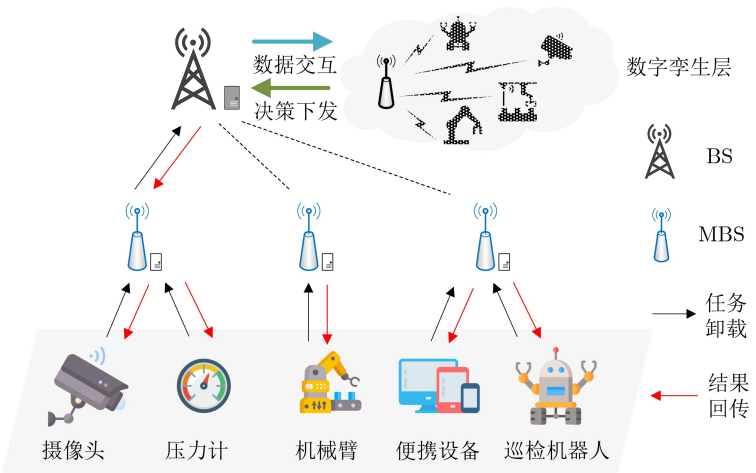

图 1 数字孪生辅助任务卸载的应用场景

2 系统模型

2.1 网络场景

本文考虑在IIoT场景中构建DT辅助任务卸载的3层网络架构，如图1所示，本架构由3层组件构成：最底层为分布式IIoT设备，中间层为有一定计算能力的MBS，最顶层为集中式的BS。假设每个IIoT设备在一个时隙n内随机生成时间敏感型计算任务，这些任务是可拆分的。考虑到IIoT设备自身的计算能力有限，本文使用MBS和BS辅助IIoT设备完成计算任务。其中，假设BS与IIoT设备之间的视距链路被阻断，若IIoT设备将计算任务卸载至BS，则需通过MBS中继至BS。因此，每个计算任务有3种完成方式，即本地计算、卸载至MBS计算和卸载至BS中计算。数字孪生层中，物理实体的数字孪生体反映了物理系统运行的动态信息。

假设有K个IIoT设备，集合表示为$\mathcal{K}=\{1,\cdots,k,\cdots,K\}$；有$U$个配有MEC服务器的MBS，集合表示为$\mathcal{U}=\{1,\cdots,u,\cdots,U\}$；一个配有MEC服务器的BS。系统分为若干个时隙，时隙集合表示为$\mathcal{N}=\{1,\cdots,n,\cdots,N\}$。假设IIoT设备$k$在时隙$n$能与$U_{k,n}$个MBS产生关联关系，但1个IIoT设备$k$只能与1个MBS相关联，故设备$k$可能的关联变量集合表示为$\varepsilon_{k,n}=\{\varepsilon_k^1,\cdots,\varepsilon_k^{u_{k,n}},\cdots,\varepsilon_k^{U_{k,n}}\},\forall u_{k,n}\in[U_{k,n}]$，其中$\varepsilon_k^{u_{k,n}}$的上标$u_{k,n}$是MBS的编号，$[U_{k,n}]$表示$U_{k,n}$个MBS的集合。当$\varepsilon_k^{u_{k,n}}=1$表示设备$k$在时隙$n$与编号为$u_{k,n}$的MBS相关联；相应地，当$\varepsilon_k^{u_{k,n}}=0$表示设备$k$在时隙$n$不与编号为$u_{k,n}$的MBS相关联。由于每个IIoT设备至多与1个MBS相关联，则将该约束公式化为

$$\sum_{u_{k,n}=1}^{U_{k,n}}\varepsilon_k^{u_{k,n}}\le 1,\forall k\in\mathcal{K} \tag{1}$$

假设IIoT设备在每个时隙最多生成1个计算任务，每个任务都可以划分为多个子任务，每个子任务的数据量大小是一定的，这些子任务可以被卸载至MBS中执行，也可以通过MBS卸载至BS中执行。当$\varepsilon_k^{u_{k,n}}=1$时，假设将IIoT设备$k$生成的一个计算任务划分给本地、MBS u和BS的任务划分比例分别用$\tau_{k,n}$,$\upsilon_{k,u,n}$和$\bar{\lambda}_{u,b,n}$表示，用于描述设备k生成计算任务输入数据的一部分(如设备生成一段视频的一个片段文件的大小)，根据定义，则有

$$\tau_{k,n}+\upsilon_{k,u,n}+\bar{\lambda}_{u,b,n}=1,\forall k\in\mathcal{K},\forall n\in\mathcal{N} \tag{2}$$

其中，$\bar{\lambda}_{u,b,n}=1-\tau_{k,n}-\upsilon_{k,u,n}$，根据实际的应用场景，任务划分比例参数$\tau_{k,n}$,$\upsilon_{k,u,n}$和$\bar{\lambda}_{u,b,n}$不能取任意小的值，故本文设置任务划分比例参数的取值集合为$\{0,0.1,\cdots,1\}$，计算时在连续定义域$[0,1]$中先获取最佳的参数值$\tau_{k,n}^*$, $\upsilon_{k,u,n}^*$和$\bar{\lambda}_{u,b,n}^*$，之后再执行四舍五入操作，选择最接近的可行值。

本文采用正交频分多址(Orthogonal Frequency Division Multiple Access, OFDMA)[6]协议，将部分计算任务尽快地卸载至MBS和BS中计算。在基于OFDMA的任务卸载中，在时隙n，IIoT设备k生成的计算任务表示为

$$Q_k[n]=\{\varsigma_k[n],D_k[n],F_k[n],T_k^{\max}[n]\},\forall k\in\mathcal{K},\forall n\in\mathcal{N} \tag{3}$$

其中，$\varsigma_k[n]$表示计算任务的类型，本文中只考虑两种任务类型，即当计算任务为独立型子任务时，$\varsigma_k[n]=1$，否则$\varsigma_k[n]=0$；$D_k[n]$表示计算任务的总位数；$F_k[n]$表示计算单位任务所需的CPU周期数；$T_k^{\max}[n]$表示任务所能容忍的最大计算时间。

2.2 数字孪生模型构建

本文考虑在BS中构建IIoT设备和MBS的DT模型。IIoT设备的DT模型是对应物理实体设备的数字化表示，根据实际的网络拓扑状态、任务请求等信息，实时地与物理IIoT设备交互更新。然而由于网络的动态变化、信道环境的不可靠等因素，所构建的DT模型不能与物理实体中的数据保持实时同步，因此会存在数据偏差。本文使用$\Delta f_k[n]$, $\Delta p_k[n]$和$\Delta b_k[n]$分别表示在时隙n中设备k可用的CPU频率偏差、发射功率偏差和带宽偏差，以描述实际的IIoT设备与其DT之间的数据偏差；则在时隙n中，IIoT设备k的DT模型表示为

$$D_k^{\text{IIoT}}[n]=\Gamma(D_{\text{IIoT}},S_{\text{IIoT}},\Delta S_{\text{IIoT}}) \tag{4}$$

其中，D_{IIoT}表示IIoT设备k中收集的数据，例如设备的CPU频率数据、发射功率数据和带宽资源数据等；S_{IIoT}表示设备的运行状态信息和信道状态数据等；ΔS_{IIoT}表示孪生设备和物理设备的交互状态数据。

类似地，在时隙n内，构建MBSu的DT模型表示为

$$D_u^{\text{MBS}}[n]=\Gamma(D_{\text{MBS}},S_{\text{MBS}},\Delta S_{\text{MBS}}) \tag{5}$$

其中，D_{MBS}表示MBSu的状态数据，如MBS的CPU频率数据；S_{MBS}是MBS的运行状态信息、连接数据和信道状态数据等；ΔS_{MBS}是数字孪生体和物理设备的交互状态数据。

2.3 任务卸载计算模型

假设IIoT设备k在时隙n内生成一个计算任务，则任务卸载计算的总时间分为本地卸载计算时间、部分传输至MBS的卸载计算时间和传输至BS的卸载计算时间，因此，本文将在以下3个模块

中分别讨论每种情形下任务卸载时间的详细建模过程。

(1)本地卸载。当计算任务部分保留至本地计算时，此时本地计算任务的数据量为$\tau_{k,n}D_k[n]$，则在时隙n内，IIoT设备执行本地计算任务所需的时间表示为[7–9]

$$T_k^{\mathrm{comp}}[n]=\frac{\tau_{k,n}D_k[n]F_k[n]}{f_k[n]} \tag{6}$$

然而由于实际的环境影响因素，DT中的孪生数据信息和实际的物理设备中的数据有偏差，假设已知CPU频率的偏差值$\Delta f_k[n]$，则本地任务计算时间的偏差为

$$\begin{aligned}\Delta T_k^{\mathrm{IIoT}}[n]&=T_{k,\mathrm{true}}^{\mathrm{comp}}[n]-T_k^{\mathrm{comp}}[n]\\&=\frac{\tau_{k,n}D_k[n]F_k[n]\Delta f_k[n]}{f_k[n](f_k[n]-\Delta f_k[n])}\end{aligned} \tag{7}$$

其中，$T_{k,\mathrm{true}}^{\mathrm{comp}}$表示本地任务计算的实际时间。因此，任务$k$的总计算时间为

$$T_k^{\mathrm{IIoT}}[n]=T_k^{\mathrm{comp}}[n]+\Delta T_k^{\mathrm{IIoT}}[n] \tag{8}$$

本地任务计算的能量消耗为

$$E_k^{\mathrm{IIoT}}[n]=\tau_{k,n}D_k[n]F_k[n]\iota_k(f_k[n]-\Delta f_k[n])^2 \tag{9}$$

其中，ι_k为IIoT设备k的有效电容系数[10]。

(2)卸载至MBS中计算。若$\varepsilon_{k,n}^u=1$，则表示任务要卸载至编号为u的MBS中计算。设在时隙n时，IIoTk到MBSu的信道增益为$G_{k,u}^{\mathrm{IIoT}}[n]$，则两者之间可达的数据传输速率为

$$R_{k,u}^{\mathrm{IIoT}}[n]=b_k\log_2\left(1+\frac{p_k[n]G_{k,u}^{\mathrm{IIoT}}[n]}{\sigma^2}\right),\quad\forall k\in\mathcal{K},\forall n\in\mathcal{N} \tag{10}$$

其中，$p_k[n]$表示设备k在时隙n时的发射功率；σ^2表示加性高斯白噪声功率[11]。则子任务卸载至MBSu的传输时间为

$$T_{k,u}^{\mathrm{trans}}[n]=\frac{(\upsilon_{k,u,n}+\bar{\lambda}_{u,b,n})D_k[n]}{R_{k,u}^{\mathrm{IIoT}}[n]},\forall k\in\mathcal{K},\forall n\in\mathcal{N} \tag{11}$$

由于DT中的数据信息与实际IIoT设备之间存在信息偏差，假设已知发射功率偏差$\Delta p_k[n]$和带宽偏差$\Delta b_k[n]$，则设备k向MBS卸载子任务的卸载时间偏差为

$$\begin{aligned}\Delta T_{k,u}^{\mathrm{trans}}[n]=&T_{k,u,t}^{\mathrm{trans}}[n]-T_{k,u}^{\mathrm{trans}}[n]\\=&\frac{(\upsilon_{k,u,n}+\bar{\lambda}_{u,b,n})D_k[n][R_{k,u}^{\mathrm{IIoT}}[n]-(b_k-\Delta b_k)\log_2(1+(p_k-\Delta p_k)G_{k,u}^{\mathrm{IIoT}}[n]/\sigma^2)]}{R_{k,u}^{\mathrm{IIoT}}[n](b_k-\Delta b_k)\log_2(1+(p_k-\Delta p_k)G_{k,u}^{\mathrm{IIoT}}[n]/\sigma^2)}\end{aligned} \tag{12}$$

其中，$T_{k,u,t}^{\mathrm{trans}}[n]$表示实际的卸载传输时间。则由设备$k$卸载至MBS的卸载时间为

$$T_k^{\mathrm{trans}}[n]=T_{k,u}^{\mathrm{trans}}[n]+\Delta T_{k,u}^{\mathrm{trans}}[n] \tag{13}$$

卸载过程中，设备k所消耗的能量为

$$E_{k,u}^{\mathrm{trans}}[n]=(p_k[n]-\Delta p_k[n])T_k^{\mathrm{trans}}[n],\forall k\in\mathcal{K},\forall n\in\mathcal{N} \tag{14}$$

MBSu处，子任务的计算时间为

$$T_u^{\mathrm{comp}}[n]=\frac{\upsilon_{k,u,n}D_k[n]F_k[n]}{f_u[n]} \tag{15}$$

同理，假设已知MBSuCPU频率的偏差值为$\Delta f_u[n]$，则MBSu处子任务计算的时间偏差为

$$\begin{aligned}\Delta T_u^{\mathrm{MBS}}[n]&=T_{u,\mathrm{true}}^{\mathrm{comp}}[n]-T_u^{\mathrm{comp}}[n]\\&=\frac{\upsilon_{k,u,n}D_k[n]F_k[n]\Delta f_u[n]}{f_u[n](f_u[n]-\Delta f_u[n])}\end{aligned} \tag{16}$$

其中，$T_{u,\mathrm{true}}^{\mathrm{comp}}$表示MBS$u$进行子任务计算的实际时间，则MBS$u$处子任务的真实计算时间为

$$T_{\mathrm{comp}}^{\mathrm{MBS}}[n]=T_u^{\mathrm{comp}}[n]+\Delta T_u^{\mathrm{MBS}}[n] \tag{17}$$

因此，MBSu处进行卸载和计算的总时间为

$$T_u^{\mathrm{MBS}}[n]=T_k^{\mathrm{trans}}[n]+T_{\mathrm{comp}}^{\mathrm{MBS}}[n] \tag{18}$$

(3)卸载至BS中计算。在时隙n内，假设MBSu到BS可达的数据传输速率为$R_{u,b}^{\mathrm{MBS}}$，则子任务从MBS卸载至BS的传输时间为

$$T_u^{\mathrm{trans}}[n]=\frac{\bar{\lambda}_{u,b,n}D_k[n]}{R_{u,b}^{\mathrm{MBS}}},\forall k\in\mathcal{K},\forall n\in\mathcal{N} \tag{19}$$

在BS处，子任务的计算时间为

$$T_{\mathrm{comp}}^{\mathrm{BS}}[n]=\frac{\bar{\lambda}_{u,b,n}D_k[n]F_k[n]}{f_b[n]} \tag{20}$$

则卸载至BS进行计算的总时间为

$$T_b^{\mathrm{BS}}[n]=T_k^{\mathrm{trans}}[n]+T_u^{\mathrm{trans}}[n]+T_{\mathrm{comp}}^{\mathrm{BS}}[n] \tag{21}$$

在通常情况下，MBS和BS的下行数据传输速率比上行链路高得多，并且输出的数据量较小。因此，本文忽略了下行传输延时，文献[12,13]中也假设如此。

本文考虑可划分任务的两种类型。对于独立型子任务，子任务可以被并行计算，因此该任务的计算时间是3处所消耗时间的最大值，则独立型子任务进行卸载计算的总时间为

$$T_k^{\mathrm{id}}[n]=\max\left\{T_k^{\mathrm{IIoT}}[n],\sum_{u=1}^{U_k}\varepsilon_{k,n}^uT_u^{\mathrm{MBS}}[n],\sum_{u=1}^{U_k}\varepsilon_{k,n}^uT_b^{\mathrm{BS}}[n]\right\} \tag{22}$$

在实际应用场景中，许多计算子任务是非独立的，计算过程是按照一定的顺序依次完成的，此类子任务的计算不可并行执行，只能串行计算，则顺序依赖型子任务进行卸载计算的总时间为

$$T_k^{\mathrm{od}}[n]=\max\left\{T_k^{\mathrm{IIoT}}[n],T_k^{\mathrm{trans}}[n]\right\}+\sum_{u=1}^{U_k}\varepsilon_{k,n}^{u}\max\left\{T_{\mathrm{comp}}^{\mathrm{MBS}}[n],T_u^{\mathrm{trans}}[n]\right\}+\sum_{u=1}^{U_k}\varepsilon_{k,n}^{u}T_{\mathrm{comp}}^{\mathrm{BS}}[n] \tag{23}$$

因此，在一段时间N内，最小化所有设备产生计算任务的平均总卸载时间的优化问题设计为

$$\begin{aligned}
&(\mathrm{P1})\ \min_{\boldsymbol{\tau},\boldsymbol{\upsilon},\boldsymbol{\varepsilon}}\frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K}\varsigma_k[n]T_k^{\mathrm{id}}[n]+(1-\varsigma_k[n])T_k^{\mathrm{od}}[n]+c_3\sum_{k=1}^{K}(T_k-T_k^{\max})\\
&\text{s.t.}\begin{cases}
\mathrm{C1}:\tau_{k,n}+\upsilon_{k,u,n}+\bar{\lambda}_{u,b,n}=1,\ \forall k\in\mathcal{K},\forall u\in[U_{k,n}]\\
\mathrm{C2}:\{\tau_{k,n},\upsilon_{k,u,n}\}\in[0,1],\ \forall k\in\mathcal{K},\forall u\in[U_{k,n}]\\
\mathrm{C3}:\{\upsilon_{k,u,n},\bar{\lambda}_{u,b,n}\}\le\varepsilon_{k,n}^{u},\ \forall k\in\mathcal{K},\forall u\in[U_{k,n}]\\
\mathrm{C4}:\sum_{u=1}^{U_{k,n}}\varepsilon_{k,n}^{u}\le 1,\ \forall k\in\mathcal{K},\forall u\in[U_{k,n}]\\
\mathrm{C5}:\sum_{k=1}^{K}\varepsilon_{k,n}^{u}\le\hbar_{\max}^{u},\ \forall k\in\mathcal{K},\forall u\in[U_{k,n}]\\
\mathrm{C6}:\{\varepsilon_{k,n}^{u}\}\in\{0,1\},\ \forall k\in\mathcal{K},\forall u\in[U_{k,n}]\\
\mathrm{C7}:\{T_k^{\mathrm{IIoT}}[n],T_u^{\mathrm{MBS}}[n],T_b^{\mathrm{BS}}[n]\}\in[0,T_k^{\max}[n]],\ \forall k\in\mathcal{K}\\
\mathrm{C8}:E_k^{\mathrm{IIoT}}[n]+E_{k,u}^{trans}[n]<E_k^{\max}[n],\ \forall k\in\mathcal{K}\\
\mathrm{C9}:\varsigma_k\in\{0,1\},\ \forall k\in\mathcal{K}\\
\mathrm{C10}:C_u\ge C_k,\ \forall k\in\mathcal{K},\forall u\in[U_{k,n}]
\end{cases}
\end{aligned} \tag{24}$$

其中，c_3表示惩罚系数，当一个任务的实际计算时间超过其所允许的最大时间时，c_3为正整数，否则$c_3=0$；$c_3\sum_{k=1}^{K}(T_k-T_k^{\max})$表示服务失败惩罚项，当实际任务计算时间$T_k$超过所约束的最大时间$T_k^{\max}$时，该项为正值加以惩罚。

约束C1表示每个任务的子任务划分比例变量之和为1；C2约束了子任务划分比例的取值范围；C3约束了IIoT设备要先与MBS相关联，才会有子任务卸载的先后顺序；C4表示每个IIoT设备最多与1个MBS相关联；约束C5表示每个MBS最多可以服务终端设备的数量$\hbar_{\max}^{u}$；约束C6表明了关联变量的取值情况；约束C7表示各个子任务的计算时延不超过每个子任务的最大容忍延时；约束C8表示终端设备进行任务计算和卸载的能耗不超过其自身剩余的能量；约束C9表示子任务的类型；约束C10表示所关联MBS的计算能力C_u应大于任务所需求的C_k。

3 基于DMAPQN的用户关联和任务划分算法

由于上述优化目标是包含离散用户关联变量和连续任务划分变量的非凸NP难优化问题，且两类变量之间具有耦合性，不能使用普通的算法直接求解。而参数化深度Q网络(Parameterized Deep Q-Networks, P-DQN)算法是独立学习范式的，各个智能体之间决策互不影响，不适用于解决上述优化问题。但DMAPQN[14]算法是P-DQN算法的多智能体形式，可通过使用多个智能体协同完成计算卸载任务。因此，本文选取既不改变原有动作空间的结构，又能维持连续和离散动作之间联系的DMAPQN算法进行求解。其中，马尔可夫决策过程(Markovian Decision Process, MDP)模型的详细构建过程如下。

3.1 MDP模型

在BS中，将每个IIoT设备对应的数字孪生体作为一个智能体，通过使用DMAPQN算法迭代学习，每个智能体只能观测到局部环境信息，所有智能体统揽当前时隙下的全局环境，多个智能体与各自的环境交互，以学习出局部最佳的用户关联策略和任务划分策略。MDP模型通过一个元组$<\mathcal{S},\mathcal{A},\mathcal{P},r>$进行描述，其中：$\mathcal{S}$表示全局环境的实际状态集合，所有智能体在当前时隙$n$所观测到的部分环境状态信息用集合$\{\boldsymbol{s}_{1,n},\cdots,\boldsymbol{s}_{k,n},\cdots,\boldsymbol{s}_{K,n}\}$表示，其中的每个集合都是全局环境状态$\mathcal{S}$的子集；$\mathcal{A}$表示智能体的动作空间集合；$r$表示环境反馈给智能体的共享奖励函数；$\mathcal{P}$表示由当前时隙智能体的状态转移概率。

针对智能体k所观测到的部分环境信息，定义

状态空间、动作空间、状态转移概率和奖励函数如下：

(1)状态空间：主要包括当前时隙接收的多个MBS的剩余计算资源集合、是否有计算任务的产生、计算任务的类型、及当前任务所能容忍的最大时间。具体地，在时隙n智能体k的状态空间$\boldsymbol{s}_{k,n}\in\mathcal{S}_n$表示为

$$\boldsymbol{s}_{k,n}=\{\mathcal{C}[n],Q_k[n]\},k\in\mathcal{K},n\in\mathcal{N} \tag{25}$$

其中，$\mathcal{C}[n]=\{C_1[n],C_2[n],\cdots,C_u[n]\}$表示智能体可选择MBS的剩余计算资源集合，$Q_k[n]$表示智能体$k$所检测到的计算任务。因此，在当前时隙全局环境状态表示为$\mathcal{S}_n=\{\boldsymbol{s}_{1,n},\cdots,\boldsymbol{s}_{k,n},\cdots,\boldsymbol{s}_{K,n}\},\mathcal{S}_n\in\mathcal{S}$。

(2)动作空间：假设智能体k在时隙n接收到$U_{k,n}$个的MBS的剩余资源信息，即存在$U_{k,n}$个MBS可与智能体k相关联，将$U_{k,n}$个MBS编号为$\{1,\cdots,u_{k,n},\cdots,U_{k,n}\}$，则对于用户关联的离散动作空间表示为$\varepsilon_{k,n}=\{\varepsilon_k^1,\cdots,\varepsilon_k^{u_{k,n}},\cdots,\varepsilon_k^{U_{k,n}}\}$，由于每个IIoT设备只能关联1个MBS，因此智能体k的离散动作与MBS的编号形成唯一映射关系$\{\varepsilon_k^{u_{k,n}}\to u_{k,n}\}$，因此，使用参数$u_{k,n}$表示其中的一个离散动作，且$u_{k,n}\in[U_{k,n}]$。则对于任意一种用户关联策略$u_{k,n}$，相应的连续动作表示为$x_{u_{k,n}}$，其中$x_{u_{k,n}}=\{\tau_{k,n},\upsilon_{k,u,n}\}$。

因此在时隙n中，智能体k的动作空间可表示为

$$\boldsymbol{a}_{k,n}=\{(u_{k,n},x_{u_{k,n}})|u_{k,n}\in[U_{k,n}],x_{u_{k,n}}\in\mathcal{X}_{k,n}\} \tag{26}$$

其中，$u_{k,n}$表示智能体k与编号为$u_{k,n}$的MBS关联；$x_{u_{k,n}}$表示对应的连续动作值；$\mathcal{X}_{k,n}$为连续动作值空间。

此外，定义当前时隙全局离散动作空间为$\mathcal{Z}_{\text{ass}}^n=\{[U_{1,n}],\cdots,[U_{k,n}],\cdots,[U_{K,n}]\}$，全局连续动作空间为$\mathcal{X}_{\text{div}}^n=\{\mathcal{X}_{1,n},\cdots,\mathcal{X}_{k,n},\cdots,\mathcal{X}_{K,n}\}$。则全局动作空间为$\mathcal{A}=\{\mathcal{Z}_{\text{ass}},\mathcal{X}_{\text{div}}\}$。

(3)状态转移概率：智能体在当前状态$\boldsymbol{s}_{k,n}$时向环境执行动作$\boldsymbol{a}_{k,n}$后，BS中的孪生环境状态会以一定的概率从当前状态$\boldsymbol{s}_{k,n}$转移到下一个状态$\boldsymbol{s}_{k,n+1}$，此概率即为状态转移概率，将其表示为$\mathcal{P}(\boldsymbol{s}_{k,n+1}|(\boldsymbol{s}_{k,n},\boldsymbol{a}_{k,n}))$。

(4)奖励函数：智能体在状态$\boldsymbol{s}_{k,n}$执行动作$\boldsymbol{a}_{k,n}$后，会生成一个用户关联和任务划分策略$\boldsymbol{\pi}_k^n$，当该策略满足P1中的约束条件C1～C10时，智能体会收到环境反馈的奖励$r_{k,n}$。而奖励函数与优化的目标函数相关，由于本文的优化目标是最小化任务卸载系统的总时间，而DRL的目标是最大化累积回报，因此奖励函数的设置应与目标函数呈负相关，故奖励函数定义为

$$\begin{aligned}r_{k,n}=&-\left(\varsigma_k[n]T_k^{\text{id}}[n]+(1-\varsigma_k[n])T_k^{\text{od}}[n]\right)\\&-\vartheta(T_k-T_k^{\max})\end{aligned} \tag{27}$$

在时隙n内，对于智能体k，将贝尔曼方程的混合动作值函数表示为

$$\begin{aligned}Q_{k,n}(\boldsymbol{s}_{k,n},u_{k,n},x_{u_{k,n}})=&\\\underset{r_{k,n},\boldsymbol{s}_{k,n+1}}{E}\Bigg[r_{k,n}+\gamma&\max_{u_{k,n}\in[U_{k,n}]}\\\sup_{x_{u_{k,n}}\in\mathcal{X}_{k,n}}&Q_{k,n}(\boldsymbol{s}_{k,n+1},u_{k,n},x_{u_{k,n}})\Bigg]\end{aligned} \tag{28}$$

其中，对离散动作$u_{k,n}$，令$x_{u_{k,n}}^*=\arg\sup_{x_{u_{k,n}}\in\mathcal{X}_{k,n}}Q_{k,n}(\boldsymbol{s}_{k,n+1},u_{k,n},x_{u_{k,n}})$，取最大的$Q_{k,n}(\boldsymbol{s}_{k,n+1},u_{k,n},x_{u_{k,n}}^*)$值。

根据文献[15]，应先固定当前状态和当前离散动作，求解出连续动作下的最佳动作值函数，再求解当前离散动作下的最佳动作值函数。故在当前网络状态$\boldsymbol{s}_{k,n}$下每个映射的离散动作$u_{k,n}$，最优的任务划分连续动作$x_{u_{k,n}}^*$可根据连续动作值函数获得，因此最佳的连续动作可表示为

$$x_{u_{k,n}}^*=\arg\sup_{x_{u_{k,n}}\in\mathcal{X}_{k,n}}Q_{k,n}(\boldsymbol{s}_{k,n+1},u_{k,n+1},x_{u_{k,n+1}}) \tag{29}$$

与DQN中类似，本文通过梯度下降最小化均方贝尔曼误差来估计$\omega_{k,n}$。具体而言，在第n步中，设$\omega_{k,n}$和$\theta_{k,n}$分别为价值网络和确定性策略网络的权重。为了结合多步算法，对于固定$t\geq1$，定义t步目标函数$y_{k,n}$为

$$\begin{aligned}y_{k,n}=&\sum_{i=0}^{t-1}\gamma^i r_{k,n+i}+\gamma^t\max_{u_{k,n}\in[U_{k,n}]}Q_{k,n}\\&\cdot(\boldsymbol{s}_{k,n+t},u_{k,n},x_{u_{k,n}}(\boldsymbol{s}_{k,n+t},\theta_{k,n});\omega_{k,n})\end{aligned} \tag{30}$$

与DQN中一样，本文对$\omega_{k,n}$也使用最小二乘损失函数。此外，当$\omega_{k,n}$固定时，网络的目标是找到一个合适的$\theta_{k,n}$，使得$Q_{k,n}(\boldsymbol{s}_{k,n},u_{k,n},x_{u_{k,n}}(\boldsymbol{s}_{k,n},\theta_{k,n});\omega_{k,n})$值最大化，因此，确定性策略网络的损失函数表示为

$$\begin{aligned}\ell_n^{\Theta}(\theta_{k,n})=-\sum_{u_{k,n}=1}^{U_{k,n}}&Q_{k,n}(\boldsymbol{s}_{k,n},u_{k,n},\\&x_{u_{k,n}}(\boldsymbol{s}_{k,n},\theta_{k,n});\omega_{k,n})\end{aligned} \tag{31}$$

价值网络的损失函数表示为

$$\ell_n^Q(\omega_{k,n})=\frac{1}{2}[Q_{k,n}(\boldsymbol{s}_{k,n},u_{k,n},x_{u_{k,n}};\omega_{k,n})-y_{k,n}]^2 \tag{32}$$

价值网络和策略网络的参数更新分别表示为

$$\omega_{k,n+1}\leftarrow\iota_{k,1}[\omega_{k,n}-\alpha_{k,1}\nabla_\omega\ell_n^Q(\omega_{k,n})]+\iota_{k,2}\omega'_{k,n} \tag{33}$$

$$\theta_{k,n+1} \leftarrow \iota_{k,1}[\theta_{k,n} - \beta_{k,2}\nabla_{\omega}\ell_n^{\Theta}(\omega_{k,n})] + \iota_{k,2}\theta'_{k,n} \quad (34)$$

其中，$\iota_{k,1}$和$\iota_{k,2}$分别表示智能体k执行参数更新过程中，本地参数和全局参数的更新权重占比；$\omega'_{k,n}$和$\theta'_{k,n}$分别表示全局混合网络下发的价值网络参数和策略网络参数，见式(38)和式(39)；$\alpha_{k,1}$和$\beta_{k,2}$分别表示本地价值网络和策略网络的学习率。

本文考虑有K个智能体之间的博弈过程，每个智能体k使用确定性策略网络$\mu_{u_{k,n}}(\theta_{k,n})$和动作值网络$Q_{k,n}(\omega_{k,n})$以学习出混合动作$(u^*_{k,n}, x^*_{u_{k,n}})$。确定性策略网络以每个智能体所观察的环境状态$\boldsymbol{s}_{k,n}$为输入，输出离散动作$\{1,\cdots,u_{k,n},\cdots,U_{k,n}\}$的最优连续参数$x_{u_{k,n}}$，之后通过动作值网络$Q_{k,n}$输出最优的混合动作，表示为

$$(u^*_{k,n}, x^*_{u_{k,n}}) = \underset{(u_{k,n},x_{u_{k,n}})}{\arg\max}\ Q_{k,n}(\boldsymbol{s}_{k,n},(u_{k,n},x_{u_{k,n}});\omega_{k,n}) \quad (35)$$

其中，$\omega_{k,n}$是在时隙n内，智能体k的动作值网络参数。

为了计算确定性策略网络的梯度，首先将每个智能体k的所有离散动作$Q_{k,n}$值相加，结果为

$$\hat{Q}_{k,n} = \sum_{u_{k,n}=1}^{U_{k,n}} Q_{k,n}(\boldsymbol{s}_{k,n},u_{k,n},x_{u_{k,n}};\omega_{k,n}),\ \forall u_{k,n}\in[U_{k,n}] \quad (36)$$

之后将所有智能体输出的$\hat{Q}_{k,n}$输入至全局混合网络中，产生$Q^{\text{sum}}_{\text{tot}}$，表示为

$$Q^{\text{sum}}_{\text{tot}} = f(\mathcal{S}_n, Q_{1,n},\cdots,Q_{k,n}\cdots,\hat{Q}_{K,n};\omega_{\text{mix}}) \quad (37)$$

之后更新所有智能体的连续动作策略μ_{u_k}，$k\in\mathcal{K}$，通过使用固定的参数$\omega_{k,n}$和ω_{mix}最大化$Q^{\text{sum}}_{\text{tot}}$，则网络参数分别表示为

$$\omega'_{k,n} \leftarrow \omega_{k,n} - \alpha_n\nabla_{\omega_{k,n}} l(\boldsymbol{\omega}_n) \quad (38)$$

$$\theta'_{k,n} \leftarrow \theta_{k,n} - \beta_n E_{\boldsymbol{s},\overline{u}_k}[\nabla_{\theta_{k,n}}\mu_{u_k}(\boldsymbol{s}_{k,n})\nabla_{x_{u_{k,n}}} Q^{\text{sum}}_{\text{tot}}(\mathcal{S}_n,\overline{u},\overline{x}_u;\boldsymbol{\omega}_n)|x_{u_k}=\mu_{u_k}(\boldsymbol{s}_{k,n})] \quad (39)$$

其中，$\boldsymbol{\omega}_n$表示时隙n的全局价值网络参数。α_n和β_n分别表示全局价值网络和策略网络的学习率。详细的算法流程如算法1所示。

4 仿真与性能分析

4.1 仿真设置

本文考虑一个500 m×500 m的区域，其中随机分布的MBS, IIoT设备均匀分布在该区域中，并参考文献[14]和文献[15]中的参数值设置。此外为了体现所提方案的优势，仿真中考虑了4种基准方案与本文所提方案进行对比。第1种方案为文献[16]中的随机执行方案；第2种方案为本地执行方案，计算任务全部由IIoT设备计算；第3种方案为文献[17]中的将计算任务全部卸载至MBS中计算；第4种方案根据文献[8]中的二分法执行任务划分策略进行计算。

4.2 仿真结果分析

图2展示了使用DMAPQN算法进行求解时不同学习率下累积奖励对比。由图2可知，不同的学习率对DMAPQN算法的收敛影响较大。学习率为0.01时，算法收敛速度最快，然而由于学习率的不恰当设置导致奖励函数的收敛值较小；学习率为0.000 5时，算法的收敛速度最慢，这是由于降低了算法的动作策略幅度；学习率为0.001时，算法取得最高的奖励值，说明智能体学习的策略效果最好，故后续仿真中设置学习率为0.001。

图3分析了有无DT辅助任务卸载对系统计算时间的影响。从图3可以看出，不论有无DT的辅助，任务计算时间都随其数据量的增加而增加。此外有DT辅助下任务的计算时间明显比无DT辅助下系统任务的计算时间低。图4分析了IIoT设备的CPU频率、发射功率和带宽数据的不同数据偏差率对任务计算时间误差的影响。从图4可以看出，随着数据偏差率的增大，对于相同数据量大小的任务，其计算时间也随之增加。由图可知将数据偏差率设置为0.000 1时的任务卸载计算时间的误差最小，故在后续仿真中，将数据偏差率设置为0.000 1。

为了评估本文所提任务卸载方案的性能，对比了4种任务卸载基准方案，结果如图5所示，本文所提方案实现了任务计算时间的最低，优于其他方案，例如当任务数据量为10 MB时，本文所提方案消耗的计算时间为330 ms，而对于随机执行方案、本地执行方案、全部卸载至MBS中方案和二分法方案所消耗的计算时间分别为596 ms, 705 ms, 617 ms和415 ms，与二分法卸载方案相比，节省了85 ms。

图6展现了每个时隙中，设备的平均最佳任务划分比例与设备数量的影响关系。由图6可见，随着IIoT设备数量的增加，分配给本地设备和BS任务数据量的比例增加，而分配给MBS的任务数据量比例降低，这是因为当业务负载增加时，MBS的卸载和计算时间会升高，反而本地设备和BS的计算时间不受业务负载的影响。因此，随着设备数量的增加，为了降低计算时间，应减少将任务分配给MBS。

图7展示了设备的平均最佳任务划分比例与任务数据量的影响关系。当任务数据量增大时，会以

算法 1　基于DMAPQN的用户关联和任务划分算法

输入：价值网络和策略网络的学习率$\{\alpha_{k,1},\alpha_n,\beta_{k,2},\beta_n\}$；探索概率$\phi$；全局学习回合数$J_{\max}$；概率分布$\psi$；一个mini-batch中小批量数据$I$；采样数据学习回合数$I_{\max}$

初始化：经验回放池Υ，初始化全局价值函数$Q_{\text{tot}}^{\text{sum}}$，随机初始化各个智能体中的网络参数，随机初始化全局网络参数θ_{tot}和ω_{tot}；

输出：用户关联和任务划分策略$\boldsymbol{\pi}^*$

for $j=1,2,\cdots,J_{\max}$ **do**
  for $n=1,2,\cdots,N$ **do**
    for $k=1,2,\cdots,K$ **do**
      根据式(29)计算连续动作$x_{u_{k,n}} \leftarrow x_{u_{k,n}}(\boldsymbol{s}_{k,n},\theta_{k,n},\theta_{\text{tot},n})$
      根据ϕ-贪婪策略选择动作$\boldsymbol{a}_{k,n}=\{u_{k,n},x_{u_{k,n}}\}$，选择原则为

$$\boldsymbol{a}_{k,n}=\begin{cases}\text{来自分布}\psi\text{ 的一个样本, 概率}\varphi\\ \{u_{k,n},x_{u_{k,n}}\},\ u_{k,n}=\arg\max\limits_{u_{k,n}\in[U_{k,n}]} Q_{k,n}(\boldsymbol{s}_{k,n},u_{k,n},x_{u_{k,n}};\omega_{k,n},\boldsymbol{\omega}_n)\text{ , 概率}1-\varphi\end{cases}$$

      执行动作$\boldsymbol{a}_{k,n}$，获取奖励$r_{k,n}$，并观察下一个环境状态$\boldsymbol{s}_{k,n+1}$
      将本次经验元组$\Upsilon_{k,n}=[\boldsymbol{s}_{k,n},\boldsymbol{a}_{k,n},r_{k,n},\boldsymbol{s}_{k,n+1}]$存储至经验回放池中
      从回放池中随机抽取经验样本$[\boldsymbol{s}_{k,i},\boldsymbol{a}_{k,i},r_{k,i},\boldsymbol{s}_{k,i+1}],i\in I$
      根据式(30)计算目标函数值$y_{k,i}$
      使用数据$\{y_{k,i},\boldsymbol{s}_{k,i},\boldsymbol{a}_{k,i}\},i\in I$计算随机梯度$\nabla_\theta \ell_n^{\Theta}(\theta_{k,n})$和$\nabla_\omega \ell_n^{Q}(\omega_{k,n})$
      根据式(31)和式(32)，计算确定性策略网络和价值网络的损失函数
      根据式(33)和式(34)，更新价值网络和策略网络的参数
      将本地价值函数$Q_{k,n}$值上传至全局混合网络
      if $i>I_{\max}$ **then**
        根据式(37)计算全局混合网络$Q_{\text{tot}}^{\text{sum}}$值
        根据式(38)和式(39)更新全局混合网络的参数
        全局混合网络将全局网络参数下发至各个智能体中
      end if
    end for
  end for
end for

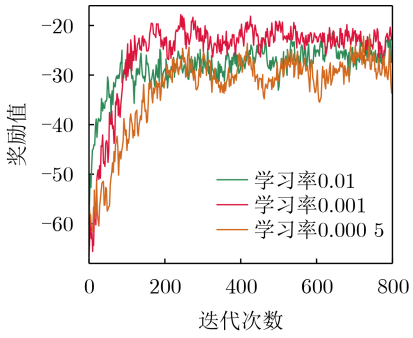

图 2 不同学习率下的累积奖励对比

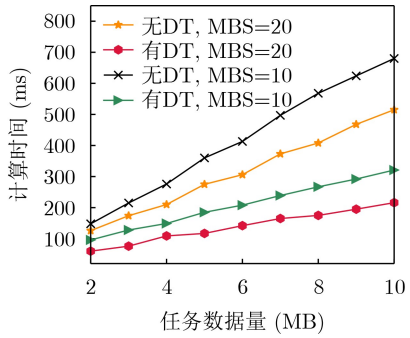

图 3 有无DT辅助的计算时间对比

较小的比例分配给本地设备和MBS，反而分配给BS的比例增大。这是因为随着任务数据量的增加，系统的总计算量增大，因此会向BS分配更多的子任务，从而可以充分利用BS中强大的计算能力以减少任务的计算时间；当任务数据量较小时，应较多地划分至MBS中计算。

5　结论

本文在云-边-端架构下的IIoT场景中，提出了DT辅助任务卸载框架，利用DT技术的优势协助完成IIoT网络中的任务卸载过程，同时使得系统中的任务卸载时间最小。首先，在数字孪生体中建立了基于任务划分和边缘关联的联合优化问题，以最小

化系统平均卸载时间和惩罚函数的加权和，作为系统协同计算卸载过程中的代价。其次，提出了一种能够同时处理连续变量和离散变量的DMAPQN算法，以在不断地迭代优化中同时学习出近似最佳的用户关联策略和任务划分策略。仿真结果表明，所提方案优于几种基准方案，证明了其在降低系统协同计算时间上的有效性。

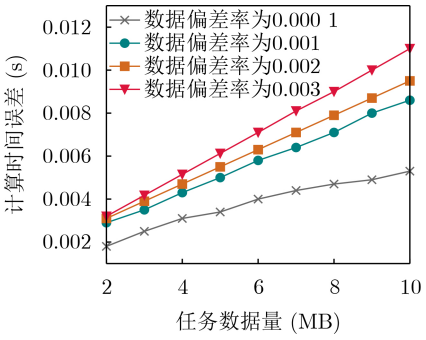

图 4 不同数据偏差率对计算时间误差的影响

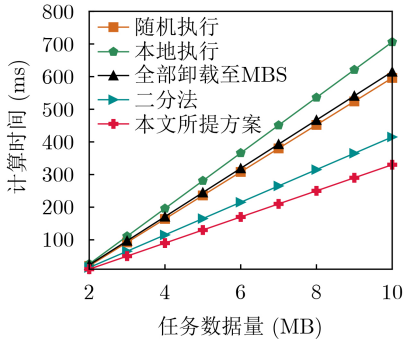

图 5 不同卸载方案下的计算时间对比

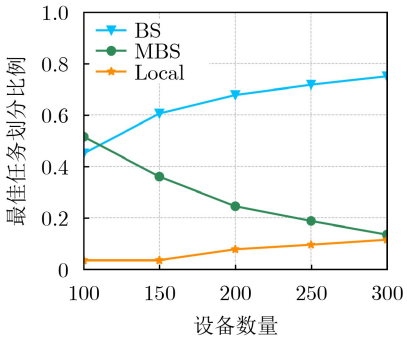

图 6 设备数量对最佳任务划分比例的影响

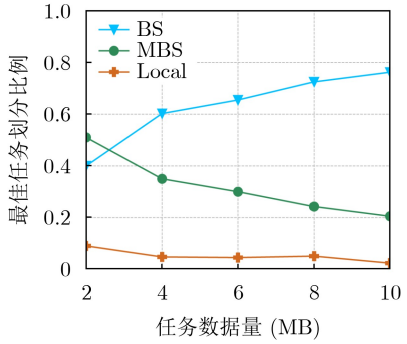

图 7 任务数据量对最佳任务划分比例的影响

参 考 文 献

[1] WU Yiwen, ZHANG Ke, and ZHANG Yan. Digital twin networks: A survey[J]. *IEEE Internet of Things Journal*, 2021, 8(18): 13789–13804. doi: 10.1109/JIOT.2021.3079510.

[2] ZHAO Liang, HAN Guangjie, LI Zhuhui, *et al*. Intelligent digital twin-based software-defined vehicular networks[J]. *IEEE Network*, 2020, 34(5): 178–184. doi: 10.1109/MNET.011.1900587.

[3] LIU Tong, TANG Lun, WANG Weili, *et al*. Digital-twin-assisted task offloading based on edge collaboration in the digital twin edge network[J]. *IEEE Internet of Things Journal*, 2022, 9(2): 1427–1444. doi: 10.1109/JIOT.2021.3086961.

[4] LI Bin, LIU Yufeng, TAN Ling, *et al*. Digital twin assisted task offloading for aerial edge computing and networks[J]. *IEEE Transactions on Vehicular Technology*, 2022, 71(10): 10863–10877. doi: 10.1109/TVT.2022.3182647.

[5] DAI Yueyue, ZHANG Ke, MAHARJAN S, *et al*. Deep reinforcement learning for stochastic computation offloading in digital twin networks[J]. *IEEE Transactions on Industrial Informatics*, 2021, 17(7): 4968–4977. doi: 10.1109/TII.2020.3016320.

[6] YE Qiaoyang, RONG Beiyu, CHEN Yudong, *et al*. User association for load balancing in heterogeneous cellular networks[J]. *IEEE Transactions on Wireless Communications*, 2013, 12(6): 2706–2716. doi: 10.1109/TWC.2013.040413.120676.

[7] DO-DUY T, VAN HUYNH D, DOBRE O A, *et al*. Digital twin-aided intelligent offloading with edge selection in mobile edge computing[J]. *IEEE Wireless Communications Letters*, 2022, 11(4): 806–810. doi: 10.1109/LWC.2022.3146207.

[8] LI Mushu, GAO Jie, ZHAO Lian, *et al*. Deep reinforcement learning for collaborative edge computing in vehicular networks[J]. *IEEE Transactions on Cognitive Communications and Networking*, 2020, 6(4): 1122–1135. doi: 10.1109/TCCN.2020.3003036.

[9] VAN HUYNH D, VAN-DINH NGUYEN, SHARMA V, *et al*. Digital twin empowered ultra-reliable and low-latency communications-based edge networks in industrial IoT environment[C]. ICC 2022 - IEEE International Conference on Communications, Seoul, Republic of, Korea, 2022: 5651–5656. doi: 10.1109/ICC45855.2022.9838860.

[10] HU Han, WANG Qun, HU R Q, *et al*. Mobility-aware offloading and resource allocation in a MEC-enabled IoT network with energy harvesting[J]. *IEEE Internet of Things*

Journal, 2021, 8(24): 17541–17556. doi: 10.1109/JIOT.2021.3081983.

[11] LI Changxiang, WANG Hong, and SONG Rongfang. Intelligent offloading for NOMA-assisted MEC via dual connectivity[J]. *IEEE Internet of Things Journal*, 2021, 8(4): 2802–2813. doi: 10.1109/JIOT.2020.3020542.

[12] HEYDARI J, GANAPATHY V, and SHAH M. Dynamic task offloading in multi-agent mobile edge computing networks[C]. 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 2019: 1–6. doi: 10.1109/GLOBECOM38437.2019.9013115.

[13] LIU Zening, YANG Yang, WANG Kunlun, *et al*. Post: Parallel offloading of splittable tasks in heterogeneous fog networks[J]. *IEEE Internet of Things Journal*, 2020, 7(4): 3170–3183. doi: 10.1109/JIOT.2020.2965566.

[14] FU Haotian, TANG Hongyao, HAO Jianye, *et al*. Deep multi-agent reinforcement learning with discrete-continuous hybrid action spaces[C]. The Twenty-Eighth International Joint Conference on Artificial Intelligence, Macao, China, 2019: 2329–2335. doi: 10.24963/IJCAI.2019/323.

[15] XIONG Jiechao, WANG Qing, YANG Zhouran, *et al*. Parametrized deep Q-networks learning: Reinforcement learning with discrete-continuous hybrid action space[EB/OL].https://arxiv.org/abs/1810.06394, 2018.

[16] SALEEM U, LIU Y, JANGSHER S, *et al*. Latency minimization for D2D-enabled partial computation offloading in mobile edge computing[J]. *IEEE Transactions on Vehicular Technology*, 2020, 69(4): 4472–4486. doi: 10.1109/TVT.2020.2978027.

[17] MOURAD A, TOUT H, WAHAB O A, *et al*. *Ad hoc* vehicular fog enabling cooperative low-latency intrusion detection[J]. *IEEE Internet of Things Journal*, 2021, 8(2): 829–843. doi: 10.1109/JIOT.2020.3008488.

唐　伦：男，教授，博士生导师，研究方向为新一代无线通信网络、异构蜂窝网络、软件定义无线网络、数字孪生边缘网络等.

单贞贞：女，硕士生，研究方向为边缘智能协同计算、联邦学习效率优化、资源协同优化等.

文明艳：女，硕士生，研究方向为移动边缘计算辅助智能驾驶技术、联邦学习效率优化等.

李　荔：女，讲师，博士生，研究方向为网络切片、机器学习等.

陈前斌：男，教授，博士生导师，研究方向为个人通信、多媒体信息处理与传输、下一代移动通信网络、异构蜂窝网络等.

责任编辑：余　蓉