DOI:10.19652/j.cnki.femt.2305499

基于用户意愿度D2D协助的工业物联网资源分配*

邓集检[1] 张月霞[1,2]
(1.北京信息科技大学信息与通信工程学院 北京 100101;
2.北京信息科技大学信息与通信系统信息产业部重点实验室 北京 100101)

摘 要:针对终端用户产生计算任务大小动态变化以及在工业物联网场景下业务的低时延、低能耗需求,提出了一种基于用户意愿度的D2D(device to device)协助的工业物联网资源分配模型。首先在用户层,每隔时隙t,由概率分布函数更新用户成为资源给予端的意愿度,在移动边缘计算(MEC)服务器层,使MEC具有决策功能,能对终端上传任务做出判断,寻找出合适的MEC处理;其次基于K-means聚类算法,将终端产生的任务匹配到对应的层进行处理;最后在资源分配阶段,为解决Q-learning里Q表难以实时更新的问题,提出N-DQN算法,使用双层神经网络相互拟合。仿真表明所提策略较传统方法,系统能耗降低约10%,系统时延降低约12%。

关键词:意愿度;工业物联网;边缘计算;K-means;资源分配
中图分类号:TN922.1 **文献标识码**:A **国家标准学科分类代码**:510

Resource allocation of industrial internet of things based on user willingness D2D assistance

Deng Jijian[1] Zhang Yuexia[1,2]
(1. School of Information and Communication Engineering, Beijing University of Information Science and Technology, Beijing 100101, China; 2. Key Laboratory of Information and Communication Systems, Ministry of Information Industry, Beijing University of Information Science and Technology, Beijing 100101, China)

Abstract: In view of the dynamic changes in the size of computing tasks generated by end users and the low-latency and low-energy consumption requirements of services in industrial IoT scenarios, a D2D-assisted industrial IoT resource allocation model based on user willingness is proposed. First, at the user layer, every time slot t, the probability distribution function is used to update the user's willingness to become a resource giver. At the mobile edge computing (MEC) server layer, the MEC is given a decision-making function that can make judgments on terminal upload tasks and find the appropriate solution. MEC processing; secondly, based on the K-means clustering algorithm, the tasks generated by the terminal are matched to the corresponding layer for processing; finally, in the resource allocation stage, in order to solve the problem that the Q table in Q-learning is difficult to update in real time, N-DQN is proposed algorithms, fit each other using two-layer neural networks. Simulation shows that the proposed strategy reduces system energy consumption by 10% and system delay by 12% compared with traditional methods.

Keywords: willingness; industrial internet of things; edge computing; K-means; resource allocation

0 引 言

随着工业物联网(industrial internet of things,IIoT)的发展,大量计算密集型应用(如智慧工厂里面的机械臂、叉车等)获得了广泛运用,为满足用户满意度,提高生产效率,这些设备往往要求低时延[1]。同时,当前IIoT设备自身计算资源有限,所产生的计算任务并不能完全自身处理。为解决该问题,当前已有大量的研究全部卸载到云和移动边缘计算(mobile edge computing,MEC)服务器层。然而,拥有丰富计算资源的云层离用户较远,用户传输时

收稿日期:2023-09-01
* **基金项目**:国家重点研发计划子课题(2020YFC1511704)

延会较大[2]。

MEC作为更贴近于终端IIoT用户的微云，它不仅在网络边缘为IIoT设备提供计算服务，减轻IIoT设备大量增加带来的影响，还可以进一步降低IIoT设备应用卸载的时延[3]。另外，IIoT用户端在某时隙下，可能部分IIoT用户处于计算资源过剩状态，通过D2D(device to device)卸载和本地计算联合使得IIoT用户在保证真实性前提下能降低计算任务时延[4]。文献[5]将D2D卸载计算与MEC架构进行结合，并考虑了计算资源和通信资源的约束，但是该方法并不适用于计算任务大小动态变化的IIoT场景，文献[6-7]提出了联合移动边缘计算和云计算的方案，该方案考虑了移动边缘计算和云计算共同协作，使用了Q-learning进行无线资源分配，该方法在IIoT场景下，面对数目庞大的计算任务产生，往往不能实时更新Q表，导致决策时延增大。文献[8]使用D2D作为资源分配的中继协助方式，提出的模型考虑了在多用户场景下，上行链路通信拥塞、D2D复用通信链路之间的干扰和上传云计算往返时延等因素的卸载方案。

为更好的贴近IIoT场景，文献[9]考虑终端用户的移动性和资源计算节点的可迁移性，提出可迁移的移动计算模型，使用马尔科夫链进行决策，对单设备的计算卸载和迁移决策使用深度Q网络(deep Q network，DQN)进行优化。文献[10]提出一种基于遗传算法的最优卸载决策及资源分配，但是该算法编程实现比较复杂，计算成本高。文献[11]提出了一种基于精度感知机器学习的IIoT联合任务卸载和资源分配方案，旨在最小化受任务卸载、边缘服务器和云服务器上的机器学习模型(machine learning，ML)模型的推理精度影响的长期平均系统成本。在IIoT场景下，由于系统高度复杂，基于操作和网络图的环境是时变的，并且可能无法获得所需的信息，文献[12]提出一种基于深度Q网络的方案，以解决基于网络图的IIoT系统中的带宽利用率和能源效率问题。文献[13]提出了一种新的多任务多服务场景联合资源管理方案，该方案由多个传感器、云服务器和配备边缘服务器的基站组成，旨在最小化系统时延与能耗总成本。

综上所述，当前大部分研究并未考虑到IIoT场景下计算任务大小的动态变化，也没有考虑到D2D设备成为计算资源给予端的意愿度，传统方法并不能适应IIoT用户产生计算任务大小动态变化情况。

本文在IIoT用户端，通过正态概率分布函数将IIoT用户分为计算资源给予端以及计算资源请求端，该方法能动态的，实时的表述某时隙下IIoT用户的计算资源剩余状态。在基站范围内，所有MEC均具有决策功能，称之为边缘决策MEC，具有判断从IIoT用户端传上来的计算任务能否在对应的MEC上进行处理，离基站最近的MEC称之为中心决策MEC，上面存储着每隔时隙t，其他边缘决策MEC的状态，如计算资源剩余值等。在资源分配阶段使用了双层Q网络互相拟合，弥补了传统Q-learning实时更新Q表的缺陷，最终达到整个系统能耗与时延最优化。

1 系统模型

1.1 资源分配系统模型

本文所提的资源分配系统模型中，如图1所示，假设在基站覆盖范围内一共有$s+u$个用户端，用户端通过动态概率分布函数，被分为请求计算资源用户端(client requesting computing resources，Cr)，记为$User=\{Cr_1,\cdots,Cr_{i-1},Cr_i,\cdots,Cr_u\}$及给予计算资源用户端(giving computing resources to clients，Gr)，记为$Ser=\{Gr_1,\cdots,Gr_{j-1},Gr_j,\cdots,Gr_s\}$。在边缘服务器层有$m$个边缘决策MEC，记为$Mec=\{MEC_1,\cdots,MEC_n,\cdots,MEC_m\}$，其主要功能是判断用户端上传的计算任务是否超过MEC自身的计算资源，若超出，将任务传至中心决策MEC，它能将上传任务卸载到有能力处理的边缘决策MEC上，确保时延与能耗最低；当计算任务过大无法通过D2D协助以及MEC卸载时，可通过无线传输至云端处理。

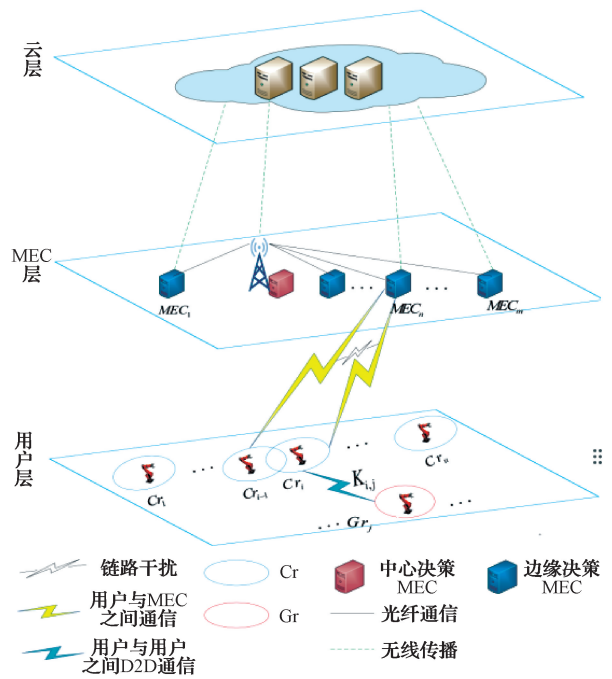

图1 基于用户意愿度的D2D协助的工业物联网资源分配模型

1.2 用户端分类与D2D协助的计算分析

图1中无论是Cr或者Gr均具有自私性，而且用户端计算任务大小是动态变化的，因此使用动态概率分布函数来表示某时刻下某用户端的成为Gr的意愿度：

$$Int=\lambda\frac{1}{\sqrt{2\pi}\sigma_1}e^{-\frac{x^2}{2\sigma_1^2}} \tag{1}$$

式中：λ 是归一化参数，确保 $Int \in (0,1)$；σ_1 表示基站覆盖范围内 IIoT 用户产生计算任务的平均大小；x 表示 IIoT 用户计算任务大小动态变化，如果 x 较大，$Int \in (0,0.3)$，表示该用户成为 Gr 意愿度低。对所有 IIoT 用户进行意愿度计算，降序排列，依次选择成为 Gr：

$$Gr_{i,j} = \text{argmax}\{Int\} \forall U_i \in U, \forall S_j \in S \tag{2}$$

假设第 i 个 Cr 在时隙 Δt 下有计算任务 $C_{task} = \{D_i, T_i\}$，D_i 表示完成计算任务所需要的 CPU 数，T_i 表示完成任务最大容忍时延。设系统内有 w 个正交信道，信道集合为 $W = \{1,2,\cdots,w\}$，每个信道带宽为 B，定义信道分配因子为：

$$w_w, k_{i,j}(t) \in \{0,1\}, \forall w \in W, \forall k_{i,j} \tag{3}$$

其中，$K_{i,j}$ 表示第 i 个 Cr 与第 j 个 Gr 之间进行 D2D 卸载[5]，考虑在第 n 个边缘决策 MEC 范围内的 IIoT 用户上传信号链路之间产生的干扰：

$$I_{w,k_{i,n}}(t) = \sum_{k_{i',n}} \sum_{w} w_w, k_{i',n}(t) p_w, k_{i',n}(t) g k_{i',n}, k_{i,n} + \sum_{n' \neq n} \sum_{k_{i,n'}} \sum_{w} w_w, k_{i,n'}(t) p_w, k_{i,n'}(t) g k_{i,n'}, k_{i,n} \tag{4}$$

式中：$p_w, k_{i',n}(t)$ 与 $p_w, k_{i,n'}(t)$ 表示用户设备里相对应的 Cr 到的发射功率[14]；$g k_{i',n}, k_{i,n}$ 表示其用户设备 $k_{i',n}$ 到当前用户设备 $k_{i,n}$ 的信道增益。

1.3 用户端分析

1)本地卸载

设定每个 Cr 计算量大小均为 D_{Cr}，计算能力均为 f_i^l，单位 CPU 时钟周期能耗均为 δ_i^l。分别计算出第 i 个 Cr 在本地处理计算任务的时延 T_i^l 以及能耗 E_i^l：

$$T_i^l = \frac{D_{Cr}}{f_i^l} \tag{5}$$

$$E_i^l = D_{Cr} \delta_i^l \tag{6}$$

2)Cr 卸载到 Gr

从第 i 个 Cr 剩余的计算任务卸载到第 j 个 Gr 之间传输速率为：

$$R_{i,j} = B \log_2 \left(1 + \frac{P_i g_{k_{i,j}}}{\sigma^2 + I_{w,k_{i,j}}(t)}\right) \tag{7}$$

式中：σ^2 表示信道噪声功率；$g_{k_{i,j}}$ 表示 i 与 j 之间通信的信道增益；$\forall U_i \in U, \forall S_j \in S$。传输时延与能耗为：

$$T_{i,j}^{tra} = \frac{D_i - D_{cr}}{R_{i,j}} \tag{8}$$

$$E_{i,j}^{tra} = P_i T_{i,j} \tag{9}$$

计算处理时延与能耗：

$$T_{i,j}^{pro} = \frac{D_i - D_{cr}}{f_j^l} \tag{10}$$

$$E_{i,j}^{pro} = (D_i - D_{cr}) \delta_j^l \tag{11}$$

式中：f_j^l 表示 Gr 计算能力；δ_j^l 表示 Gr 在单个 CPU 周期内的能耗。所以在用户层上总时延为：

$$T_i^u = T_i^l + T_{i,j}^{tra} + T_{i,j}^{pro} \tag{12}$$

总能耗为：

$$E_i^u = E_i^l + E_{i,j}^{tra} + E_{i,j}^{pro} \tag{13}$$

1.4 MEC 层分析

如果第 i 个 Cr 用户是在服务范围内的第 n 个 MEC 执行计算任务 C_{task}，首先该 IIoT 用户将计算任务传到 MEC 上，MEC 收到卸载请求后做决策，判断自身计算资源是否超过请求，若满足，则该 MEC 处理即可；若不满足，将任务传至中心决策 MEC，它上面有一个账本，里面记录着每隔时隙 Δt 更新基站覆盖范围内其他边缘决策 MEC 状态，即实时可用计算资源大小 D_{req}，使用复杂度低的 KNN 搜寻算法，来选取其他最优卸载边缘决策 MEC。

$$upload = \sqrt{(D_i - D_{req})^2} \tag{14}$$

当 upload 值越接近于 0，说明对应的边缘决策 MEC 成为卸载对象可能越大，中心决策通过光纤传播卸载到它上面进行处理。

1)计算任务在服务范围内 MEC 上处理

当计算任务上传至服务范围内的边缘决策 MEC，该 MEC 能够处理时，第 i 个 IIoT 用户传输至第 n 个边缘决策服务器上传输速率记为：

$$R_{i,n}^{tran} = \text{Blog}_2 \left(1 + \frac{P_i g_{kn}}{\sigma^2 + I_{w,k_{i,n}}(t)}\right) \tag{15}$$

式中：$I_{w,k_{i,n}}(t)$ 表示在第 n 个边缘决策服务器范围内，其他 IIoT 用户对 i 到 j 传输产生的干扰；g_{kn} 表示信道增益。第 i 个 IIoT 用户传输至第 n 个边缘决策服务器上传输时间与传输能耗分别记为：

$$T_{i,n}^{tra} = \frac{D_i}{R_{i,j}^{tran}} \tag{16}$$

$$E_{i,n}^{tra} = P_i T_{i,n}^{tra} \tag{17}$$

在第 n 个边缘决策服务器上计算处理时间与能耗分别记为：

$$T_{i,n}^{pro} = \frac{D_i}{f_n} \tag{18}$$

$$E_{i,n}^{pro} = D_i \delta_n \tag{19}$$

式中：f_n 表示第 n 个边缘决策服务器的单位时间内计算能力；δ_n 表示第 n 个边缘决策服务器的单位 CPU 时钟周期内能耗。在处理完成后，数据较小，指令回传，该过程的能耗与时延可以忽略。因此在第 n 个边缘决策服务器上处理时延与能耗分别记为：

$$T_n = T_{i,n}^{tra} + T_{i,n}^{pro} \tag{20}$$

$$E_n = E_{i,n}^{tra} + E_{i,n}^{pro} \tag{21}$$

2)计算任务需通过中心决策服务器进行调度

当第 i 个 IIoT 用户在 t 时刻下传输至服务范围内第 n 个 MEC 上处理，该时刻下第 n 个 MEC 剩余计算资源无法处理该计算任务，中心决策服务器在账本上计算式(14)，使用 KNN 搜寻最优卸载对象的 MEC，不妨记为 $M_d, M_d \in M$。从中心决策器到第 d 个 MEC 采用的是光纤传输。式(22)、(23)分别表示第 n 个 MEC

传输至第 d 个 MEC 的传输能耗，以及在第 d 个 MEC 上处理能耗。

$$E_{n,d}^{tra}=P_n\frac{D_i}{V_{of}}+P_{base}\frac{D_i}{V_{of}} \tag{22}$$

$$E_{n,d}^{pro}=D_i\delta_d \tag{23}$$

计算任务传通过中心决策服务器进行调度的时延与能耗分别记为 $T_{n\to d}$ 和 $E_{n\to d}$：

$$T_{n\to d}=T_{i,n}^{tra}+2\frac{D_i}{V_{of}}+\frac{D_i}{f_d} \tag{24}$$

$$E_{n\to d}=E_{i,n}^{tra}+E_{n,d}^{tra}+E_{n,d}^{pro} \tag{25}$$

假设计算任务在传输中损失忽略不计，P_n 表示第 n 个 MEC 发射功率，P_{base} 表示中心决策 MEC 发射功率，V_{of} 表示光纤传输速率，δ_d 表示第 d 个边缘决策服务器的单位 CPU 时钟周期内能耗。f_d 表示第 d 个边缘决策服务器的单位时间内计算能力。

1.5 云层分析

若第 i 个 IIoT 终端用户的计算任务较大，无法通过本地计算或 MEC 来处理，则只有上传到云层。式(26)表示上传云端传输时间与在云端处理计算任务时间，式(27)表示上传云端的能耗与在云端处理的能耗。

$$T_c=\frac{D_i}{R_c}+\frac{D_i}{f_c} \tag{26}$$

$$E_c=P_i\frac{D_i}{R_c}+D_i\delta_c \tag{27}$$

式中：f_c 表示云端单位时间内计算能力；δ_c 表示云端服务器的单位 CPU 时钟周期内能耗。

1.6 问题建模

将系统的任务卸载和资源分配建模为优化问题，最终目标是将整个系统的能耗和时延最小化，即最小化系统总成本。在任务的最大容忍延迟 T_i 和边缘服务器的计算资源 F 的限制下，该问题建模为：

$$\omega_i=\varphi\sum_{i=1}^{u}[\beta_1T_i^u+\beta_2(T_n\oplus T_{n\to d})+\beta_3T_c]+$$

$$0.1\xi\sum_{i=1}^{u}[\beta_1E_i^u+\beta_2(E_n\oplus E_{n\to d})+\beta_3E_c]$$

$$\min\omega_i$$

$$C1:T_i^u\leqslant T_i$$

$$C2:T_n\oplus T_{n\to d}\leqslant T_i$$

$$C3:T_c\leqslant T_i$$

$$C4:\beta_1+\beta_2+\beta_3=1;\beta_1\in\{0,1\},\beta_2\in\{0,1\},\beta_3\in\{0,1\} \tag{28}$$

$$C5:0\leqslant f_i^m\leqslant\beta_2F,\forall U_i\in U$$

$$C6:\sum_{i=1}^{u}\beta_2f_i^m\leqslant F,\forall U_i\in U$$

$$C7:P_i\leqslant P_i^{\max}$$

$$C8:\varphi+\xi=1$$

$$C9:f_j^{Gr}\geqslant D_i-D_{Cr},\forall S_j\in S$$

$$C10:z_d\geqslant D_i$$

式(28)为确保时延与能耗在一个量值上，取系统能耗值的 1/10，$T_n\oplus T_{n\to d}$ 表示若需要中心决策 MEC 调度，则两者相加，若不需要中心决策，则后者为 0。$\beta=[\beta_1,\beta_2,\beta_3]$ 是卸载决策向量，β_1 表示本地卸载与 D2D 协助卸载，β_2 表示 MEC 卸载，β_3 表示云端卸载；f 是 MEC 服务器的计算资源分配向量，表示为 $f=[f_1^m,\cdots,f_i^m,\cdots,f_u^m]$；C1 ～ C3 表示无论卸载到哪个平台，其处理时延都要小于任务最大容忍时延，C4 表示只允许卸载到一个平台处理，C5 表示 MEC 分配的计算资源要小于本身所具有的的计算资源，C6 表示所有在 MEC 上卸载计算的资源总和小于 MEC 总计算资源的和，C7 表示用户发射功率小于最大发射功率。C8 表示权重调节系数，C9 表示在卸载对象 Gr 上的计算资源应大于计算任务在本地计算处理后所剩计算资源，C10 表示经中心决策 MEC 调度后，卸载第 d 个边缘决策 MEC 所剩计算资源应大于计算任务大小。

为求解式(28)最优化问题，先要匹配到对应平台，即卸载决策 β，分别对应本地 CPU、MEC、cloud[15]；其次是给 MEC 上处理的任务资源分配的量，即资源分配向量 f。式(28)是目标函数是非凸的，鉴于 IIoT 用户基数的庞大，使用传统的非凸优化转凸优化问题很难解决。因此本文使用 K-means 方法进行卸载决策，使用改进的 Q-learning 算法进行资源分配。

2 多平台卸载决策

任务卸载及资源分配的智能算法首先确定计算任务卸载到对应平台，借鉴强化学习里的 K-means 聚类方法，将计算任务与相应的平台进行聚类[16]。

$$D=\sqrt{(T_i-\overline{T})^2+(B_i-\overline{B})^2} \tag{29}$$

式中：T_i、$\overline{T}$、B、$\overline{B}$ 分别表示第 i 个 Cr 计算任务的最大延迟、各平台处理计算任务的平均延迟、第 i 个 Cr 用户计算任务的大小、各平台允许计算任务的平均大小。如果 D 的大小越接近 0，则说明越越有可能在对应的平台执行任务。K-means 聚类算法的优点是计算复杂度低，聚类效果明显，能快速对应计算任务卸载平台，而且对于移动的终端适应性较高[17]。

3 多平台智能资源分配算法

3.1 状态空间

为联合优化卸载决策与卸载策略以最优化目标函数，定义整个系统的状态为 $\boldsymbol{S}$ 包括卸载决策向量 $\boldsymbol{\beta}$、卸载策略向量 f 以及 MEC 上剩余计算资源向量 $\boldsymbol{Z}$，其中 $\boldsymbol{Z}=[z_1,z_2,\cdots,z_i,\cdots,z_m]$，$z_i$ 表示第 i 台 MEC 服务器上所剩的计算资源：

$$z_i=f_i^m-\sum_{i=1}^{u}f_i^{consume} \tag{30}$$

$$\boldsymbol{S}=[\boldsymbol{\beta},\boldsymbol{f},\boldsymbol{Z}] \tag{31}$$

对于每个 IIoT 用户的状态包括计算任务的基本信息

与上传到MEC的传输速率，可表示为$O_{n,t}=\{B_i,D_i,T_i,R_{i,j}^{tra},R_{i,m}^{tra},R_c^{tra}\}$，$B_i$表示在第$i$个Cr用户，$D_i$在时隙$t$下计算任务的数据大小，$T_i$为时隙$t$下完成计算任务的最大容忍时延，以及传输至各平台速率。

3.2 动作空间

在时隙t，第i个Cr用户动作包含计算卸载决策和资源分配决策分别为$\phi_i=\{\beta_1,\beta_2,\beta_3\}$，$\theta_i=\{f_i^m,f_i^c\}$，$f_i^m$表示第$i$个Cr向MEC请求的计算资源，$f_i^c$表示第$i$个Cr向cloud请求的计算资源，动作记为：

$$\boldsymbol{A}=[\phi_i,\theta_i] \tag{32}$$

3.3 奖励函数

因为该研究的目标是使系统能效最小，奖励越大，所以设定奖励函数为：

$$R_i(\boldsymbol{S},\boldsymbol{A})=\frac{1}{\omega_i} \tag{33}$$

整个系统的累计奖励为：

$$J^{\pi}(s)=\lim_{N\to k}E\left(\sum_{i=1}^{u}R_i(\boldsymbol{S},\boldsymbol{A})\right) \tag{34}$$

资源最优分配策略：

$$\pi^*=\operatorname{argmax}J(s,\pi) \tag{35}$$

3.4 改进的Q-learning算法(N-DQN)

Q-learning是一种异策略的时序差分学习方法，IIoT用户在状态$\boldsymbol{S}$下，选择动作$\boldsymbol{A}$，获得奖励$R_i(\boldsymbol{S},\boldsymbol{A})$，与SARSA算法不同，Q学习算法不通过策略函数$\pi$来选下一步的动作$\boldsymbol{A}$，而是直接选择最优的Q函数，所以在IIoT用户进行资源分配时，实时更新Q表并不现实，因为随着终端用户增多，根据排列组合原理，状态与动作的组合出现的可能性将大大增加，有限的Q表将难以进行对应Q值的存储[18]。本文中N-DQN(图2)利用双网络结构和经验回放方法结合贪婪算法做出智能决策，系统构建两个结构相同的神经网络，预测网络$Q(s,a;\theta)$，根据用户在t时刻状态对$\{s_t,a_t\}$得到的回报函数$R_i(\boldsymbol{S},\boldsymbol{A})$，预测$t+1$时刻状态$s_{t+1}$，并更新Q矩阵：

$$Q(s,a;\theta)=(1-\rho)Q^*(s_t,a_t)+\rho[R_i(\boldsymbol{S},\boldsymbol{A})+\theta\max_{a_{t+1}}Q^*(s_{t+1},a_{t+1})] \tag{36}$$

为了预测网络$Q(s,a;\theta)$，系统对用户状态随机抽取U个，其中第U个样本记为(s_t,a_t,R_t,s_{t+1})，因此训练目标函数为：

$$y_u=R_t(\boldsymbol{S},\boldsymbol{A})+\gamma\operatorname*{argmax}_{a'\in A}Q^*(s_{t+1},a';\theta^-) \tag{37}$$

式中：γ为回报值的折扣。预测网络参数θ更新的均方差损失函数表示为：

$$L(\theta)=\frac{1}{U}\sum_{u=1}^{U}[y_u-Q(s_t,a_t;\theta)]^2 \tag{38}$$

其中，θ的更新方式采用梯度下降法：

$$\theta=\theta-v\nabla_\theta L(\theta) \tag{39}$$

式中：v是更新步长。目标网络参数在每隔固定采样周期，做如下更新：

$$\theta^-=\theta \tag{40}$$

目标网络$Q^*(s,a;\theta^-)$：

$$Q^*(s,a;\theta^-)=E\{R_{t+1}+\theta\max_{a'}Q^*(s_{t+1},a')\mid s_t=s,a_t=a'\} \tag{41}$$

其中，θ和θ^-分别表示两个神经网络的参数，为了在兼顾决策效率的同时遍历所有的(s,a)，在迭代过程中本文采用ζ贪婪算法[19]来选择行为：

$$a_t=\begin{cases}\operatorname{argmax}Q^*(s,a), & 1-\zeta\\ a_t, & \text{其他},\zeta\end{cases} \tag{42}$$

预测网络Q网络与目标Q^*网络进行拟合：

$$Q(s,a;\theta)\approx Q^*(s,a;\theta^-) \tag{43}$$

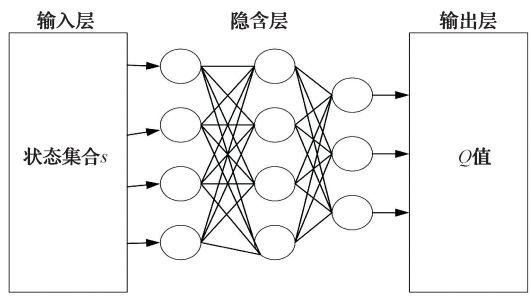

图2 N-DQN算法

4 仿真

通过仿真测试来验证所提出的模型以及算法的可行性，并参考文献[20]的仿真参数搭建仿真平台，使用java语言进行模型仿真，以及python语言里面的matplotlib库进行数据绘图。本文假设所提出的模型中基站覆盖半径为1 km，边缘决策MEC个数$m=4$，计算资源给予端用户数为$u_{Gr}=4$，计算资源请求端用户数为$s_{Cr}=6$，神经网络采用Sigmoid作为激活函数，每层64个神经元，设定时隙长度为1 s，Statesiz为8，Outputsize为4。为进一步体现本文模型的性能提升，将本文模型与文献[21]模型进行时延性能及能耗性能的分析对比，如图3所示。从图3可以看出，使用K-means聚类算法，在终端用户较多的情况下，只需知道某时刻下终端用户的计算任务大小和最大容忍时延，就能够快速将该终端用户的计算任务匹配到对应的处理平台[22]。

本文提出的N-DQN算法进行了性能分析，如图4所示，随着训练次数的增加，均方差损失函数基本降至0.1附近，这表示训练的神经网络与实际的神经网络拟合比较好；在学习率为0.01时，系统模型所获得奖励值最大，也侧面证明了本文所提的N-DQN方法在资源分配的可行性。

不同方法的系统时延与能耗对比如图5所示，可以看出，随着终端用户数的逐渐增加，本文方法与完全上传MEC和完全上传云端进行对比就会发现，无论是在能耗还是时延维度，比其他两种方法增长趋势都慢，本文方法

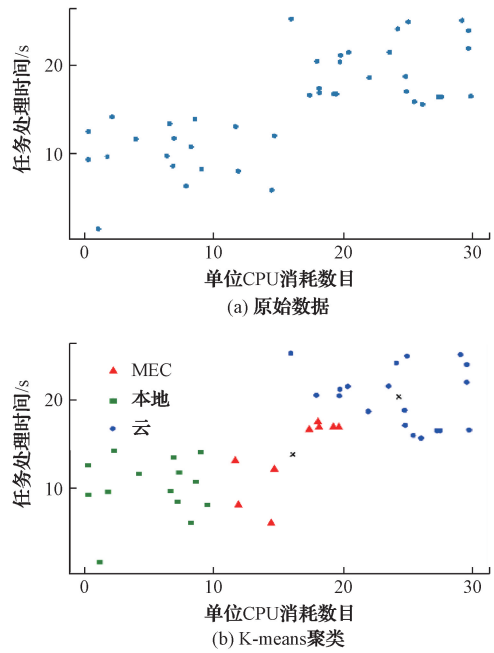

图 3　原始计算任务与 K-means 聚类后对比

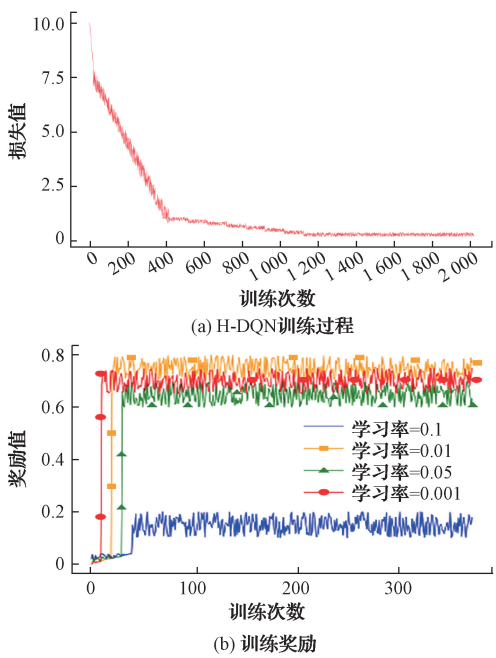

图 4　N-DQN 训练次数与损失值和奖励值的仿真

较完全上传云端时延降低了近 40%，较完全上传 MEC 降低了 26%；较完全上传云端能耗降低了 36%，较完全上传 MEC 降低了 23%。

N-DQN 传统 Q-learning 时延与能耗对比如图 6 所示，可以看出，N-DQN 较传统 Q-learning 算法在系统时延降低 12%，能耗降低 10%左右，主要原因是传统 Q-learning 在用户数量增多时，难以实时更新 Q 表，计算量增大，

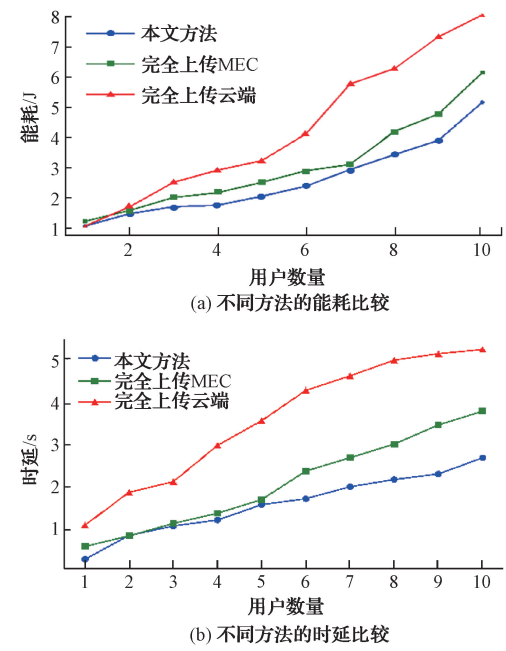

图 5　不同方法下系统时延与能耗对比

从而能耗与时延均有所增加，而 N-DQN 采用神经网络预测与拟合，有效避免这一问题。

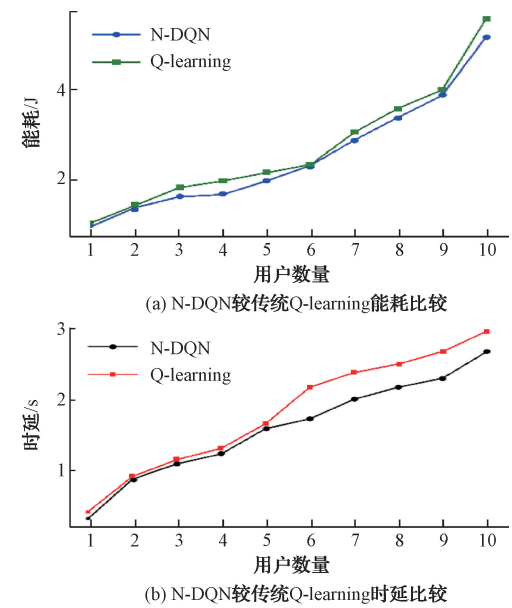

图 6　N-DQN 与传统 Q-learning 时延与能耗对比

为了检验该策略在基站不同覆盖范围内是否依旧有效，设计多组基站不通覆盖范围进行多次实验验证，该组实验中，用户数量设定为 10，结果如图 7 所示，可以明显看到，随着基站覆盖范围的增大，综合系统时延和能耗的目标函数值均是增加的，但本文方法较传统方法增长更缓慢，这说明系统能耗与时延有所降低。

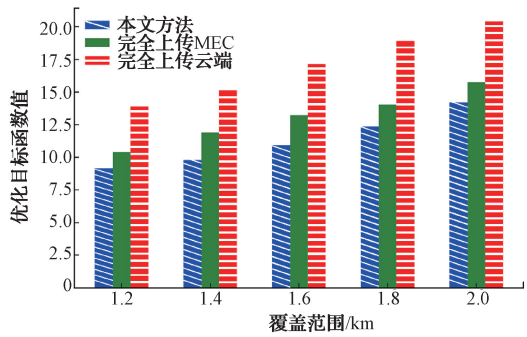

图 7　不同基站覆盖范围内优化函数目标值

5　结　论

本文提出了基于用户意愿度的 D2D 协助的工业物联网资源分配模型，在保证最大容忍时延前提下，先将终端用户动态划分为计算资源给予端和计算资源请求端，边缘服务器具备决策功能，能将终端上传任务传至合适的 MEC 处理；针对终端计算任务大小的动态变化，使用 K-means 聚类算能够较快的将计算任务匹配到相应的层进行处理。为解决传统 Q-learning 实时更新 Q 表的难度，使用双层神经网络进行拟合，均方差损失函数曲线证明了拟合效果。仿真结果表明，本文所提策略能有效降低系统能耗和计算任务处理时延。

参 考 文 献

[1] CHI Y, DONG Y, WANG Z J, et al. Knowledge-based fault diagnosis in industrial internet of things: A survey[J]. IEEE Internet of Things Journal, 2022, 9(15): 12886-12900.

[2] AAZAM M, UL ISLAM S, LONE S T, et al. Cloud of things (CoT): Cloud-fog-IoT task offloading for sustainable internet of things[J]. IEEE Transactions on Sustainable Computing, 2020, 7(1): 87-98.

[3] 熊泽凯，王素红，王靖君，等. 移动边缘计算中服务功能链的自适应优化部署策略[J]. 电讯技术，2023，63(11)：1678-1686.

[4] DAI X, XIAO Z, JIANG H, et al. Task co-offloading for d2d-assisted mobile edge computing in industrial internet of things[J]. IEEE Transactions on Industrial Informatics, 2022, 19(1): 480-490.

[5] 陈前斌，谭颀，贺兰钦，等. 云雾混合网络下基于多智能体架构的资源分配及卸载决策研究[J]. 电子与信息学报，2021，43(9)：2654-2662.

[6] 王汝言，梁颖杰，崔亚平. 车辆网络多平台卸载智能资源分配算法[J]. 电子与信息学报，2020，42(1)：263-270.

[7] LIN C C, DENG D J, YAO C C. Resource allocation in vehicular cloud computing systems with heterogeneous vehicles and roadside units[J]. IEEE Internet of Things Journal, 2017, 5(5): 3692-3700.

[8] 赵临东，庄文芹，陈建新，等. 异构蜂窝网络中分层任务卸载：建模与优化[J]. 通信学报，2020，41(4)：34-44.

[9] TANG Z, ZHOU X, ZHANG F, et al. Migration modeling and learning algorithms for containers in fog computing [J]. IEEE Transactions on Services Computing, 2018, 12(5): 712-725.

[10] 姜宇晴，李曦，纪红，等. 无人机辅助车联网中基于车辆动态聚类的协作卸载策略[J]. 移动通信，2022，46(11)：27-33.

[11] FAN W, LI S, LIU J, et al. Joint task offloading and resource allocation for accuracy-aware machine-learning-based IIoT applications[J]. IEEE Internet of Things Journal, 2022, 10(4): 3305-3321.

[12] LIANG F, YU W, LIU X, et al. Toward deep Q-network-based resource allocation in industrial internet of things [J]. IEEE Internet of Things Journal, 2021, 9(12): 9138-9150.

[13] FAN W, CHEN Z, HAO Z, et al. DNN deployment, task offloading, and resource allocation for joint task inference in IIoT[J]. IEEE Transactions on Industrial Informatics, 2022, 19(2): 1634-1646.

[14] 刘子怡，李君，李正权. 多用户蜂窝网络中基于深度强化学习的功率分配[J]. 国外电子测量技术，2023，42(3)：30-35.

[15] 赵尚维康. 工业物联网中基于边缘计算和强化学习的计算卸载方法研究[D]. 南京：南京邮电大学，2022.

[16] 刘兴鑫，李君，李正权. SWIPT-D2D 通信中基于深度强化学习的资源分配[J/OL]. 电讯技术，1-8[2024-01-30]. https://doi.org/10.20079/j.issn.1001-893x.230202003.

[17] 盛煜，朱正伟，朱晨阳，等. 基于深度强化学习的多目标边缘任务调度研究[J]. 电子测量技术，2023，46(8)：74-81.

[18] 姜宇晴，李曦，纪红，等. 无人机辅助车联网中基于车辆动态聚类的协作卸载策略[J]. 移动通信，2022，46(11)：27-33.

[19] 王路鑫. 移动边缘计算低时延场景下计算卸载与资源分配策略研究[D]. 杭州：杭州电子科技大学.

[20] YI C, HUANG S, CAI J. Joint resource allocation

for device-to-device communication assisted fog computing [J]. IEEE Transactions on Mobile Computing, 2019, 20(3): 1076-1091.

[21] NATH S, WU J. Deep reinforcement learning for dynamic computation offloading and resource allocation in cache-assisted mobile edge computing systems[J]. Intelligent and Converged Networks, 2020, 1(2): 181-198.

[22] 张玉琴,梁莉,张建亮,等. 基于改进 K-means++和 DBSCAN 的大数据聚类方法[J]. 国外电子测量技术, 2022,41(9): 40-46.

作 者 简 介

邓集检,硕士研究生,主要研究方向为无线资源管理。
E-mail:dengjijian2021@qq.com

张月霞(通信作者),博士,教授,硕士生导师,主要研究方向为无线协作通信技术、超宽带技术和无线定位技术。
E-mail:zhyx-bupt@163.com