移动边缘网络中基于双深度 Q 学习的高能效资源分配方法

喻鹏[1]，张俊也[1]，李文璟[1]，周凡钦[1]，丰雷[1]，付澍[2]，邱雪松[1]
（1. 北京邮电大学网络与交换技术国家重点实验室，北京 100876；2. 重庆大学微电子与通信工程学院，重庆 400044）

摘 要：为了提升移动边缘网络中系统的能量使用效率，面向多任务、多终端设备、多边缘网关、多边缘服务器共存网络架构的下行通信过程，提出了一种基于双深度 Q 学习（DDQL）的通信、计算、存储融合资源分配方法。以任务平均能耗最小化为优化目标，设置任务时延和通信、计算、存储资源限制等约束条件，构建了对应的资源分配模型。依据模型特征，基于 DDQL 框架，提出了适用于通信和计算资源智能决策、存储资源按需分配的资源分配模型和算法。仿真结果表明，所提出的基于 DDQL 资源分配方法可以有效地解决多任务资源分配问题，具有较好的收敛性和较低的时间复杂度，在保障业务服务质量的同时，相对于基于随机算法、贪心算法、粒子群优化算法、深度 Q 学习等方法，降低了至少 5%的任务平均能耗。

关键词：移动边缘网络；融合资源分配；高能效；双深度 Q 学习
中图分类号：TN929.5
文献标识码：A
doi: 10.11959/j.issn.1000−436x.2020218

Energy-efficient resource allocation method in mobile edge network based on double deep Q-learning

YU Peng[1], ZHANG Junye[1], LI Wenjing[1], ZHOU Fanqin[1], FENG Lei[1], FU Shu[2], QIU Xuesong[1]
1. State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China
2. School of Microelectronics Communication Engineering, Chongqing University, Chongqing 400044, China

Abstract: To improve the system energy efficiency in mobile edge networks, a resource allocation method based on double deep Q-learning(DDQL) for integration of communication, computing, storage resources was proposed for the downlink communication process under the network architecture of multiple tasks, end devices, edge gateways and edge servers. A resource allocation model was constructed, which took the minimization of average energy consumption of tasks as the optimization goal and set the constraints of task delay limits and communication, computing, and storage resource limits. According to the model characteristics, a suitable resource allocation model and method based on DDQL framework was proposed to make intelligent allocation decisions for communication and computing resources and allocate storage resources on demand. Simulation results show that the proposed DDQL-based solution can effectively solve the multi-task resource allocation problem with good converge and low time complexity, and it reduces the average energy consumption of tasks by at least 5% compared with the solving methods based on random algorithm, greedy algorithm, particle swarm optimization algorithm and deep Q-learning while ensuring the quality of service.

Key words: mobile edge network, integrated resource allocation, energy-efficient, DDQL

1 引言

随着移动通信网络的不断演进，超五代（B5G, beyond the 5th generation）、第六代（6G, the 6th generation）网络将带来新型业务场景，如自动驾驶、工业控制、增强/虚拟现实等，这些场景对带宽、时延、功耗、可靠性等指标提出了更高的要求[1]。对应的海量无线接入设备所需的高效快速的资源调

收稿日期：2020−06−28；**修回日期**：2020−09−19
基金项目：国家重点研发计划基金资助项目（No.2018YFE0205502）
Foundation Item: The National Key Research and Development Program of China (No.2018YFE0205502)

度，也将给网络带来巨大挑战。

为了解决上述问题，移动边缘计算（MEC, mobile edge computing）被提出。通过边缘计算，终端设备可以卸载部分或全部计算任务到基站等网络边缘节点，拓展了终端设备计算能力。相对于集中到云端的计算方法，MEC 能够有效地降低任务处理时延，减轻核心网的流量压力，保障数据私密性与安全性[2]。

未来无线网络的深度将显著拓展，从单一的信息传输到传输、存储和处理的多维同步，需要通信、计算和存储资源以及相关控制的无缝融合[3]。而基于 MEC 的移动边缘网络（MEN, mobile edge network）的核心思想也正是将网络的资源、内容和功能迁移到网络边缘，从而提升网络整体的资源调度效率，MEC 被认为是 B5G/6G 网络的重要组成部分[1]，其资源分配方法对系统的性能有着重要影响。

5G 性能相比 4G 有了大幅度提升，但是基站部署密度也进一步提升，导致 5G 网络的基站功耗为 4G 基站的 3～4 倍[4]。而未来 6G 网络将拥有超高吞吐量、超大带宽，网络节点的部署将更加密集，规模更加庞大, 将会面临更大的能耗压力。对应地，绿色节能是未来网络发展的一大需求[5]。由于基站能耗约占通信能耗的 60%～80%[6]，而边缘网络作为基站的主要部署位置，将会成为通信网络能耗产生的重要组成部分。因此，MEN 中高能效的资源分配方法具有重要的研究意义与价值。

近年来，MEC 得到了广泛关注，MEN 资源分配问题也得到了大量的研究，而多维资源联合优化是其中的研究热点。文献[7]指出基于雾计算的通信与计算融合可以有效地提升系统的性能，并概述了基于雾计算的移动通信网络的网络架构、系统容量和资源管理。文献[8]将缓存资源引入用于多媒体内容交付的移动基站中，考虑缓存与前传成本，从经济角度进行优化。文献[9]研究了服务缓存放置、计算卸载决策和系统资源分配的联合优化。这些为边缘网络的融合资源分配方法提供了参考依据。

文献[10]针对有多个能量采集设备的 MEC 系统，将最小化长期平均执行成本的联合计算卸载和动态资源分配问题描述为一个随机优化问题，提出了一种基于李雅普诺夫优化的在线算法，将原问题转化为时隙确定性问题。文献[11]针对协同多点传输设计了一种联合负载感知聚类和基于图着色的小区间资源调度的资源分配方法。为了实现泛在边缘计算，需要实现多边缘服务器的协同处理。进一步地，文献[12]提出了一种限制边缘服务器超载概率的 MEC 系统资源配置的优化方法，通过样本平均近似方法将机会约束随机规划问题转化为混合整数规划问题进行求解，实现了总通信代价最小的目标。文献[13]通过用综合成本模型描述各种静态和动态性能指标，建立了混合非线性优化的在线边缘网络资源分配模型，利用正则化技术将非凸优化问题转化为凸优化问题，模型的性能较贪心算法有大幅度提高。文献[14]研究了多信道无线干扰情况下的多用户计算卸载与资源分配策略，可以通过博弈论方法分布式地高效求解，并证明了所设计的算法可以达到纳什均衡。上述研究以数学优化方法为主，对优化问题的数学形式具有较高的要求，需要针对具体问题对模型和约束进行精心设计或者进行转化，例如求解对象维度单一、模型要求无约束或者少量线性约束、可用经典算法进行求解等，且多采用离线方式，主要适用于少量网络节点的局部网络场景，难以适用于求解变量维度和约束较为复杂的场景。

针对上述不足，面对未来网络密集化、复杂化的发展趋势，强化学习（RL, reinforcement learning）作为一种免模型的方法，可以自动通过试错进行学习，具有很强的灵活性[15]，是一种有前景的解决方案[16]，RL 方法可适用于复杂动态的 MEC 系统。文献[17]将具有间歇性和不可预测性的可再生能源作为 MEC 系统的能源，提出一种有效的基于 RL 的资源管理算法，该算法分解为离线值迭代和在线强化学习，动态地学习负载卸载和边缘服务器配置的最优策略，使系统长期成本最小化。近年来，以深度 Q 学习（DQL, deep Q-learning）为代表的深度强化学习（DRL, deep reinforcement learning）算法兴起。DRL 在高维离散或者连续空间中具有很强的决策能力，克服了 RL 方法只适用于具有低维状态和动作空间问题的不足。并且，基于图形处理单元的并行计算进一步提升了 DRL 的运行速度，使网络管理具有及时性，克服了元启发式算法、凸优化算法等传统方法的运行时间限制[18-19]。

一些研究将 DRL 算法用于解决 MEN 资源分配任务。文献[20-21]提出基于 DQL 的方案，来联合优化计算资源与网络资源。文献[22]在计算卸载与资源分配问题中，将几种 DRL 算法，包括 DQL、深度确定性策略梯度（DDPG, deep deterministic

policy gradient）和异步优势 actor-critic（A3C, asynchronous advantage actor-critic）算法，进行了对比。为了解决 DQL 存在的 Q 值过估计问题，双深度 Q 学习（DDQL, double deep Q-learning）算法被提出[23]。文献[24]提出了基于 DDQL 的算法，在不了解网络状态的情况下学习最优计算卸载策略。

然而，上述研究大多关注的是上行流量为主的应用场景，较少分析下行流量为主的应用场景。并且，很多研究只考虑了单一资源的分配问题，部分研究对通信和计算资源进行了联合优化，或者关注缓存相关策略和资源分配策略的联合优化，但是对通信、计算、存储 3 种资源进行综合考虑的研究不足。此外，高能效的资源分配机制研究主要关注了终端设备能耗，而对系统的总能耗关注不够。

针对目前研究存在的问题，本文重点关注如复杂视频处理、高清视频请求等具有大量下行数据的业务，在多任务、多终端设备、多边缘网关、多边缘服务器的 MEN 场景下，以任务平均能耗最小化为优化目标，针对每个任务选择的边缘网关，考虑边缘网关最大发射功率、边缘服务器最大计算能力和最大存储空间等资源约束，以及任务时延限制等约束，构建对边缘网关发射功率和边缘服务器计算能力和存储空间进行分配决策的优化模型。该问题是一个 NP-hard 的优化问题。

本文将构建的数学模型进行简化，提出了基于 DDQL 的求解方法，并通过实验仿真将其与基于随机算法（RA, random algorithm）、贪心算法（GA, greedy algorithm）、粒子群优化（PSO, particle swarm optimization）算法、DQL 算法的求解方法进行了对比，证明本文方法降低了至少 5%的任务平均能耗。DDQL 算法具有良好的收敛性和较低的时间复杂度，可以很好地完成 MEN 高能效资源分配任务。

2 资源分配模型构建

2.1 网络系统架构分析

面向未来 B5G/6G 网络特征，网络系统架构可分为四层，自底向上分别为终端设备（ED, end device）、边缘网关（EG, edge gateway）、边缘服务器（ES, edge server）和云中心（CC, cloud center），如图 1 所示。其中，任务由终端设备发起，边缘网关主要负责网络协议转化与数据转发，边缘服务器主要负责提供计算与存储功能，云中心在远端具有更丰富的资源。云中心是系统架构的必要组成部分，但在本文模型中，假设边缘服务器可以满足任务需求，不需要在云中心进行任务处理。

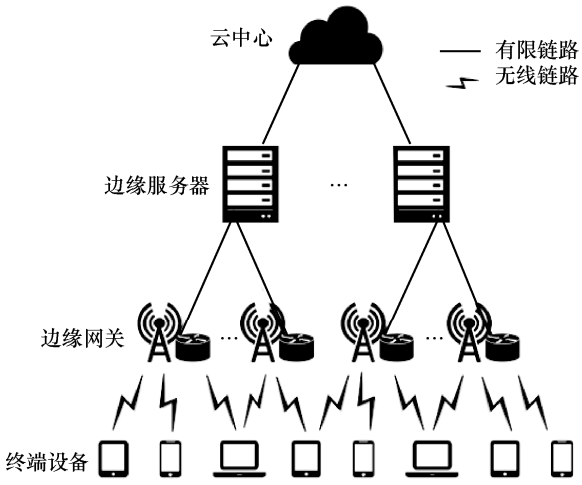


图 1 系统架构

考虑实际网络系统，边缘网关是现场级边缘计算的典型设备形态，可部署在基站侧；边缘服务器是以通用硬件为虚拟化资源的移动边缘应用平台，可部署在基带处理单元池等运营商机房中。边缘网关与边缘服务器多在网络规划时设计了其隶属关系，如多对一的关系，在网络建设时通过光纤等有线链路连接，因此可将边缘服务器与边缘网关设定为固定连接。边缘网关与终端设备通过无线信道通信，其连接关系需要在满足覆盖关系的条件下与资源分配进行联合决策。

2.2 任务模型

设终端设备的集合 $\mathcal{D}=\{1,2,\cdots,D\}$，$d\in\mathcal{D}$ 表示一个终端设备，终端设备数为 D。边缘网关的集合为 $\mathcal{G}=\{1,2,\cdots,G\}$，$g\in\mathcal{G}$ 表示一个边缘网关，边缘网关数为 G。边缘服务器的集合为 $\mathcal{S}=\{1,2,\cdots,S\}$，$s\in\mathcal{S}$ 表示一个边缘服务器，边缘服务器数为 S。

任务的集合表示为 $\mathcal{K}=\{1,2,\cdots,K\}$，$k\in\mathcal{K}$ 表示一个任务，任务数为 K。任务 k 用五元组 (d_k,l_k,b_k,c_k,T_k) 表征，其中 d_k 为发起任务 k 的终端设备，$d_k\in\mathcal{D}$，假设一个终端设备一次最多发起一个任务，l_k 为任务 k 返回终端设备的数据比特数，b_k 为任务 k 所需存储空间大小，c_k 为完成任务 k 所需中央处理器（CPU, central processing unit）时钟周期数，T_k 为完成任务 k 的时延限制。假设以上 K 个任务均为同一个时间片内发起的任务。

2.3 边缘服务器选择与能耗模型

边缘服务器的选择与能耗模型构建如下。

设 $x_{k,s}$ 表示任务 k 选择 ES s 的情况，为

$$x_{k,s}=\begin{cases}1, & \text{任务}k\text{选择了ES}\,s\\ 0, & \text{任务}k\text{没有选择ES}\,s\end{cases} \tag{1}$$

一个任务只能且必须选择一个 ES，如式(2)所示。

$$\sum_{s\in\mathcal{S}}x_{k,s}=1,\quad \forall k\in\mathcal{K} \tag{2}$$

针对存储资源，设 B_s 表示 ES s 的最大存储空间。每个 ES 中任务所占用的存储空间之和不能超过该 ES 最大存储空间，即

$$\sum_{k\in\mathcal{K}}x_{k,s}b_k\leqslant B_s,\quad \forall s\in\mathcal{S} \tag{3}$$

针对计算资源，考虑 CPU 是执行计算任务的核心设备，其性能与时钟频率有关，可采用动态电压频率调整（DVFS, dynamic voltage and frequency scaling）技术对频率进行调节，以满足任务的时延、能耗要求[25]。同一个 ES 上的不同任务可同时执行，分别获得不同的 CPU 时钟频率。F_s 表示 ES s 所能提供的最大 CPU 时钟频率。f_k 表示任务 k 所获时钟频率。每个 ES 中任务所获 CPU 时钟频率之和不能超过该 ES 能提供的最大 CPU 时钟频率，即

$$\sum_{k\in\mathcal{K}}x_{k,s}f_k\leqslant F_s,\quad \forall s\in\mathcal{S} \tag{4}$$

对每一个任务来说，其获得的 CPU 时钟频率范围有一定的限制，$F_{\min}$ 为一个任务可获得的 CPU 时钟频率的最小值，$F_{\max}$ 为一个任务可获得的 CPU 时钟频率的最大值，则有

$$F_{\min}\leqslant f_k\leqslant F_{\max},\quad \forall k\in\mathcal{K} \tag{5}$$

任务 k 的计算时延为

$$t_k^S=\frac{c_k}{f_k} \tag{6}$$

根据电路理论，动态能耗是 CPU 能耗最主要的组成部分。在本文模型中，ES 的能耗只考虑因计算产生的动态能耗，而忽略其他能耗。ES 执行任务 k 的能耗为 $\kappa c_k f_k^2$，其中，κ 为与硬件有关的常量[25]。所有 ES 执行任务的总能耗为

$$E^S=\sum_{k\in\mathcal{K}}\kappa c_k f_k^2 \tag{7}$$

2.4　边缘网关选择与能耗模型

边缘网关的选择与能耗模型如下。

$y_{k,g}$ 表示任务 k 选择 EG g 的情况，如式(8)所示。

$$y_{k,g}=\begin{cases}1, & \text{任务}k\text{选择了EG}\,g\\ 0, & \text{任务}k\text{没有选择EG}\,g\end{cases} \tag{8}$$

一个任务只能且必须选择一个 EG，如式(9)所示。

$$\sum_{g\in\mathcal{G}}y_{k,g}=1,\quad \forall k\in\mathcal{K} \tag{9}$$

ES 与 EG 通过有线链路通信，$z_{s,g}$ 表示 ES s 和 EG g 的连接关系，即

$$z_{s,g}=\begin{cases}1, & \text{ES}\,s\text{和EG}\,g\text{有链路连接}\\ 0, & \text{ES}\,s\text{和EG}\,g\text{无链路连接}\end{cases} \tag{10}$$

EG 与 ED 通过无线链路通信，$w_{g,d}$ 表示 EG g 与 ED d 的覆盖关系，如式(11)所示。

$$w_{g,d}=\begin{cases}1, & \text{ED}\,d\text{在EG}\,g\text{覆盖范围内}\\ 0, & \text{ED}\,d\text{不在EG}\,g\text{覆盖范围内}\end{cases} \tag{11}$$

任务 k 选择的 ES s 和 EG g 必须可通信，且只能选择一条路径，表示为

$$\sum_{g\in\mathcal{G}}\sum_{s\in\mathcal{S}}x_{k,s}y_{k,g}z_{s,g}=1,\quad \forall k\in\mathcal{K} \tag{12}$$

任务 k 选择的 EG g 必须能与接收任务的 ED d_k 通信，且只能选择一条路径，表示为

$$\sum_{g\in\mathcal{G}}y_{k,g}w_{g,d_k}=1,\quad \forall k\in\mathcal{K} \tag{13}$$

EG g 到 ED d 信道的带宽为 $B_{g,d}$。根据香农公式，从 EG g 到 ED d 的传输速率为

$$r_{g,d}=B_{g,d}\text{lb}(1+\delta_{g,d}) \tag{14}$$

其中，$\delta_{g,d}$ 为从 EG g 到 ED d 传输的信噪比（SNR, signal noise ratio）。$\delta_{g,d}$ 的表达式为

$$\delta_{g,d}=\frac{p_{g,d}h_{g,d}}{N_0} \tag{15}$$

其中，$p_{g,d}$ 为 EG g 到 ED d 发射功率，$h_{g,d}$ 为从 EG g 到 ED d 的路径损耗，N_0 为加性高斯白噪声谱密度。$h_{g,d}$ 的大小与 EG g 到 ED d 之间的距离 $D_{g,d}$ 有关，距离越远，路径损耗越大。

任务 k 获得的 EG 发射功率表示为 p_{g,d_k}，其范围有一定的限制，$P_{\min}$ 为最小值，$P_{\max}$ 为最大值，如式(16)所示。

$$P_{\min}\leqslant p_{g,d_k}\leqslant P_{\max},\quad \forall g\in\mathcal{G},k\in\mathcal{K} \tag{16}$$

其中，P_g 表示 EG g 所能提供的最大发射功率。一个 EG 中所有任务获得的发射功率之和不能超过该 EG 所能提供的最大发射功率，即

$$\sum_{k\in\mathcal{K}} y_{k,g} p_{g,d_k} \leqslant P_g, \quad \forall g \in \mathcal{G} \tag{17}$$

若任务 k 是从 EG g 传输到 ED d_k，则其传输时延为

$$t_{k,g,d_k} = \frac{l_k}{r_{g,d_k}} \tag{18}$$

考虑任务实际的 EG 选择情况，任务 k 从 EG 到 ED 的传输时延为

$$t_k^G = \sum_{g\in\mathcal{G}} y_{k,g} t_{k,g,d_k} \tag{19}$$

任务 k 的总时延为 ES 计算时延与从 EG 到 ED 传输时延之和，ES 与 EG 之间通过有线链路连接，传输速度很快，传输时延忽略不计，则有

$$t_k = t_k^S + t_k^G \tag{20}$$

EG g 的能耗

$$e_g = \sum_{k\in\mathcal{K}} y_{k,g} p_{g,d_k} t_{k,g,d_k} \tag{21}$$

所有 EG 的总能耗为

$$E^G = \sum_{g\in\mathcal{G}} e_g = \sum_{g\in\mathcal{G}} \sum_{k\in\mathcal{K}} y_{k,g} p_{g,d_k} t_{k,g,d_k} \tag{22}$$

2.5 能耗优化模型

在上述系统架构模型、任务模型、边缘服务器和边缘网关选择与能耗模型的基础上，考虑网络的整体能耗特征，最终的 MEN 资源分配的优化模型如式(23)所示。

$$\begin{aligned}
&\min_{\{x_{k,s}\},\{y_{k,g}\},\{z_{s,g}\},\{w_{g,d}\},\{p_{g,d_k}\},\{f_k\}} \frac{E^S + E^G}{K} \\
&\text{s.t.} \quad \mathrm{C1}: t_k \leqslant T_k, \quad \forall k \in \mathcal{K} \\
&\qquad \mathrm{C2}: \sum_{s\in\mathcal{S}} x_{k,s} = 1, \quad \forall k \in \mathcal{K} \\
&\qquad \mathrm{C3}: \sum_{g\in\mathcal{G}} y_{k,g} = 1, \quad \forall k \in \mathcal{K} \\
&\qquad \mathrm{C4}: \sum_{g\in\mathcal{G}} \sum_{s\in\mathcal{S}} x_{k,s} y_{k,g} z_{s,g} = 1, \quad \forall k \in \mathcal{K} \\
&\qquad \mathrm{C5}: \sum_{g\in\mathcal{G}} y_{k,g} w_{g,d_k} = 1, \quad \forall k \in \mathcal{K} \\
&\qquad \mathrm{C6}: \sum_{k\in\mathcal{K}} x_{k,s} b_k \leqslant B_s, \quad \forall s \in \mathcal{S} \\
&\qquad \mathrm{C7}: \sum_{k\in\mathcal{K}} x_{k,s} f_k \leqslant F_s, \quad \forall s \in \mathcal{S} \\
&\qquad \mathrm{C8}: \sum_{k\in\mathcal{K}} y_{k,g} p_{g,d_k} \leqslant P_g, \quad \forall g \in \mathcal{G} \\
&\qquad \mathrm{C9}: F_{\min} \leqslant f_k \leqslant F_{\max}, \quad \forall k \in \mathcal{K} \\
&\qquad \mathrm{C10}: P_{\min} \leqslant p_{g,d_k} \leqslant P_{\max}, \quad \forall g \in \mathcal{G}, k \in \mathcal{K}
\end{aligned} \tag{23}$$

优化目标为最小化任务平均能耗，能耗为边缘服务器计算能耗与边缘网关传输能耗之和。约束条件 C1 要求每个任务在规定时延内完成，保障用户的服务质量（QoS, quality of service）。约束条件 C2 和 C3 要求每个任务只能且必须选择一个边缘服务器和一个边缘网关。约束条件 C4 和 C5 要求每个任务选择唯一且可通信的路径。约束条件 C6~C8 分别要求满足边缘服务器的最大存储空间限制、边缘服务器的最大时钟频率限制和边缘网关的最大发射功率限制。约束条件 C9 和 C10 分别对一个任务可获得的边缘服务器时钟频率、边缘网关发射功率大小进行限制。

在该优化问题中，有六类决策变量，分别为 $x_{k,s}$、$y_{k,g}$、$z_{s,g}$、$w_{g,d}$、p_{g,d_k} 和 f_k。其中，$x_{k,s}$、$y_{k,g}$、$z_{s,g}$ 和 $w_{g,d}$ 是离散的 0-1 整数变量，p_{g,d_k} 和 f_k 是连续变量。

由于部分决策变量是离散变量，该优化问题的可行解集不是凸集，不是一个凸优化问题，无法利用凸优化问题优良的全局最优解性质，可以分析得到该问题是一个混合整数规划问题。考虑到实际工程实践的可行性，需要重点关注的不是如何精确求解最优解，而是如何高效快速地获得一个较好的可行解，结合相关工作分析，本文利用基于 DDQL 的模型来完成上述能耗优化模型的求解。

3 基于 DDQL 的模型求解方法

3.1 模型特征分析

考虑到实际网络的有线部分连接关系相对固定，因此，在任务选择 ES 与 EG 时，将 ES 与 EG 的连接关系 $z_{s,g}$ 和 EG 与 ED 的覆盖关系 $w_{g,d}$ 作为已知条件。假设每个 EG 只与一个 ES 连接，因此确定了要选择的 EG 后，只有唯一的 ES 满足 EG 与 ES 可通信的约束条件，因此，可以将 $x_{k,s}$、$y_{k,g}$ 两类决策变量合并为一类决策变量 u_k，表示任务 k 选择的 EG，选择的 ES 即为该 EG 连接的 ES。发起任务 k 的设备为 ED d_k，这个是进行资源分配前的已知条件，且每个任务只能选择一个 EG，因此任务 k 获得 EG 的发射功率 p_{g,d_k} 也可表示为 p_k。

经过简化后，关于任务 k 的决策变量有 3 个，分别为选择的 EG 的编号 u_k、任务获得 EG 发射功率 p_k、任务获得 ES 时钟频率 f_k。结合优化模型中给出的优化目标，MEN 高能效资源分配问题就是

要对每个任务的 EG 连接关系、获得 EG 发射功率、获得 ES 时钟频率进行决策，在满足时延限制、资源限制等约束条件的情况下，最小化任务平均能耗。

假设任务 k 可选择的 EG 的个数为 N_k^u，将连续变量 p_k、f_k 的可能取值离散化，假设任务 k 获得 EG 发射功率可能数值的个数为 N_k^p，获得 ES 时钟频率可能数值的个数为 N_k^f。若使用暴力搜索算法来遍历求解具有 K 个任务资源分配，其时间复杂度为 $O\left(K\prod_{k\in\mathcal{K}}(N_k^u N_k^p N_k^f)\right)$，具有指数级的时间复杂度，这是一个 NP-hard 的复杂决策优化问题，不适合大规模场景，因此需要使用智能算法在合理的时间内求次优解。

3.2　强化学习三要素定义

DRL 算法将深度学习（DL, deep learning）的强表征能力与 RL 的强决策能力相结合，并且适用于具有动态性的环境。Q 学习（Q-learning）算法是一种经典的 RL 算法，DQL 算法将 DL 方法引入 Q-learning 中，突破了 Q-learning 算法不适用于高维决策任务的局限性。DQL 算法状态空间相对容易构造，动作和奖励与网络优化的过程和目标有天然的契合度，是一种可用于网络资源分配的有效方法。但 DQL 算法中被高估的 Q 值影响了算法的性能，DDQL 算法通过分解动作选择和策略评估来克服此问题[23]。本文提出基于 DDQL 的移动边缘网络高能效资源分配方法。

RL 是智能体通过与环境交互，观察做出动作后得到的奖励，通过改变自己的行为来学习得到更多奖励的策略。RL 重要基础之一是试错的学习方式，其流程为在时刻 t，智能体从环境中观察到状态 $\boldsymbol{s}_t$，利用策略 π 选择动作 a_t。一旦该动作被执行，环境转变到下一个状态 $\boldsymbol{s}_{t+1}$，向智能体提供奖励 r_t 作为反馈。智能体的目标是学习一个可以最大化期望累积奖励的策略[25]。

在一个回合（Episode）中，从时刻 t 起，考虑无限长的时间，智能体获得的累积奖励定义为

$$G_t=\sum_{k=0}^{\infty}\gamma^k r_{t+k} \tag{24}$$

其中，$\gamma\in[0,1]$ 为折扣因子，用来削减未来奖励对现在的影响，越远的奖励作用越小。

结合本文的优化模型，对 RL 的三要素，即状态、动作和奖励进行定义。

状态：状态即为所有决策变量的组合。每个任务选择的 EG 表示为向量 $\boldsymbol{u}=[u_1,u_2,\cdots,u_K]$，每个任务获得的 EG 发射功率表示为向量 $\boldsymbol{p}=[p_1,p_2,\cdots,p_K]$，每个任务获得的 ES 时钟频率表示为向量 $\boldsymbol{f}=[f_1,f_2,\cdots,f_K]$。状态定义为 $\boldsymbol{s}=[\boldsymbol{u}\ \boldsymbol{p}\ \boldsymbol{f}]$，是一个 $3K$ 维的向量。

动作：为了使控制过程方便，每次动作只能对状态向量中的一个量进行改变，或者全都不改变，保持目前的状态。u_k^{next} 表示是否将 u_k 变为可选 EG 循环列表中的下一个 EG，p_k^{inc} 表示是否将 p_k 增加 0.1 W，p_k^{dec} 表示是否将 p_k 减小 0.1 W，f_k^{inc} 表示是否将 f_k 增加 0.1 GHz，f_k^{dec} 表示是否将 f_k 减小 0.1 GHz，j 表示是否不改变状态。p_k、f_k 本身是连续变量，本文对其进行离散化处理，p_k、f_k 增减的步长是通过仿真实验综合考虑求解结果与收敛速度后设置的。u_k^{next}、p_k^{inc}、p_k^{dec}、f_k^{inc}、f_k^{dec}、j 均为 0-1 变量，其中，1 表示是，0 表示否。每个任务的 EG 选择改变情况表示为向量 $\boldsymbol{u}^{\text{next}}=[u_1^{\text{next}},u_2^{\text{next}},\cdots,u_K^{\text{next}}]$，每个任务获得 EG 发射功率的增加情况表示为向量 $\boldsymbol{p}^{\text{inc}}=[p_1^{\text{inc}},p_2^{\text{inc}},\cdots,p_K^{\text{inc}}]$，每个任务获得 EG 发射功率的减少情况表示为向量 $\boldsymbol{p}^{\text{dec}}=[p_1^{\text{dec}},p_2^{\text{dec}},\cdots,p_K^{\text{dec}}]$，每个任务获得 ES 时钟频率的增加情况表示为向量 $\boldsymbol{f}^{\text{inc}}=[f_1^{\text{inc}},f_2^{\text{inc}},\cdots,f_K^{\text{inc}}]$，每个任务获得 ES 时钟频率的减少情况表示为向量 $\boldsymbol{f}^{\text{dec}}=[f_1^{\text{dec}},f_2^{\text{dec}},\cdots,f_K^{\text{dec}}]$。动作定义为 $\boldsymbol{a}=[\boldsymbol{u}^{\text{next}}\ \boldsymbol{p}^{\text{next}}\ \boldsymbol{p}^{\text{next}}\ \boldsymbol{f}^{\text{next}}\ \boldsymbol{f}^{\text{next}}\ j]$，是一个 $(5K+1)$ 维的独热（one-hot）编码向量，该向量只有一个元素为 1，其余均为 0，表示在 $(5K+1)$ 个可能的动作中选择一个动作。为了使表达方便，对向量形式的动作进行简化改写，记为 $\alpha\in[1,5K+1]$，表示选择执行的动作的编号，即 $\boldsymbol{a}$ 中值为 1 的元素对应的索引序号。

奖励：与式(23)模型的优化目标相对应。由于 DRL 算法要最大化累积奖励，而模型的优化目标要最小化任务平均能耗，所以将立即奖励设为优化目标的相反数，为了使奖励为正，再加上一个适当大的正数 $E_{\max}$，$E_{\max}$ 表示任务最大能耗。在状态不满足式(23)约束条件时，奖励为 0。奖励定义为

$$r=\begin{cases}E_{\max}-\dfrac{E^S+E^G}{K}, & \text{满足约束条件}\\ 0, & \text{不满足约束条件}\end{cases} \tag{25}$$

3.3 DDQL 框架构建

Q 值，即状态–动作值函数，表示在状态 $\boldsymbol{s}$ 选择动作 a，按照策略 π 执行，获得的期望累积回报，定义为

$$Q_\pi(\boldsymbol{s},a)=\mathbb{E}[G_t \mid \boldsymbol{s}_t=\boldsymbol{s},a_t=a] \tag{26}$$

Q-learning 算法需要将每个状态–动作对的 Q 值以表格形式存储，当状态或动作空间过大时，便无法存储。

DQL 算法通过深度神经网络（DNN, deep neural network）来逼近最优策略对应 Q 值，表示为 $Q_*(\boldsymbol{s},a)$ [26]，如式(27)所示。

$$Q(\boldsymbol{s},a;\boldsymbol{\theta})\approx Q_*(\boldsymbol{s},a) \tag{27}$$

其中，参数 $\boldsymbol{\theta}$ 代表神经网络的权重，在迭代中通过调整参数 $\boldsymbol{\theta}$ 来训练神经网络。将用来估计值函数的神经网络称为 Q 网络（Q-network）。

本文使用的 DNN 为多层前馈神经网络（FNN, feedforward neural network），神经元分层排列，相邻两层的神经元之间全连接，通过反向传播来调整参数。DNN 以状态为输入，输出所有可能的动作对应的 Q 值。DNN 使用 ReLU 函数作为激活函数，ReLU 函数定义为

$$\mathrm{ReLU}(x)=\max(0,x) \tag{28}$$

DQL 算法中，使用 2 个结构相同的 DNN。其中，当前 Q 网络为 ϕ，参数为 $\boldsymbol{\theta}$，用于评估当前状态动作对的 Q 值；目标 Q 网络为 $\hat{\phi}$，参数为 $\boldsymbol{\theta}^-$，用于产生目标 Q 值。

误差函数为均方误差形式，定义[27]为

$$L^{\mathrm{DQL}}(\boldsymbol{\theta})=\mathbb{E}\left[(y^{\mathrm{DQL}}-Q(\boldsymbol{s},a;\boldsymbol{\theta}))^2\right] \tag{29}$$

其中，y^{DQL} 为目标 Q 值[27]，为

$$y^{\mathrm{DQL}}=r+\gamma\max_{a'}Q(\boldsymbol{s}',a';\boldsymbol{\theta}^-) \tag{30}$$

其中，$\boldsymbol{s}'$ 为在状态 $\boldsymbol{s}$ 执行动作 a 后的下一个状态，a' 为状态 $\boldsymbol{s}'$ 下可选择的动作。

DQL 算法引入固定 Q 目标机制，使用 2 个 DNN 的原因是，如果使用同一个 DNN 计算误差并更新参数，根据不断变化的 Q 值更新网络，容易导致训练过程不稳定。因此，使用 2 个结构相同的 DNN，当前 Q 网络每步都通过随机梯度下降的方法进行更新，降低误差；目标 Q 网络每隔一定的步数更新一次，赋值为和当前 Q 网络相同的参数。

DQL 算法中还使用了经验回放机制。将在每个时刻 t 下，智能体获得的经验 $e_t=(\boldsymbol{s}_t,a_t,r_t,\boldsymbol{s}_{t+1})$ 存入回放记忆单元中。回放记忆单元的容量有一定的限制，存满后，存入新的经验时会随机替换掉旧的经验。训练时，每次从回放记忆单元中随机采样，用小批量的样本对网络进行训练，更新网络参数。

但 DQL 根据式(30)计算目标 Q 值时，每次都选择下一个状态中最大的 Q 值，且选择和评价动作都基于目标 Q 网络 $\hat{\phi}$ 的参数 $\boldsymbol{\theta}^-$，这会使 Q 值被高估。

DDQL 算法针对上述问题进行改进。在 DDQL 算法中，Q 网络 ϕ 中的参数 $\boldsymbol{\theta}$ 用来选择 Q 值最大的动作，目标 Q 网络 $\hat{\phi}$ 的参数为 $\boldsymbol{\theta}^-$ 用来评估最优动作的 Q 值，将动作选择和策略评估分开。目标 Q 值[23]为

$$y^{\mathrm{DDQL}}=r+\gamma Q(\boldsymbol{s}',\arg\max_a Q(\boldsymbol{s}',a;\boldsymbol{\theta});\boldsymbol{\theta}^-) \tag{31}$$

误差函数定义为

$$L^{\mathrm{DDQL}}(\boldsymbol{\theta})=\mathbb{E}\left[(y^{\mathrm{DDQL}}-Q(\boldsymbol{s},a;\boldsymbol{\theta}))^2\right] \tag{32}$$

DDQL 算法的其他方面与 DQL 一致，其算法框架如图 2 所示。

DDQL 算法分为离线训练和在线运行 2 个阶段。其中，离线训练阶段需要进行许多回合，对 Q 网络进行训练，在选择动作的时候使用的是 ε-贪心策略，如算法 1 所示。ε-贪心策略是指，对于探索利用率 $\varepsilon\in[0,1]$，以 ε 的概率随机选择动作，以 $(1-\varepsilon)$ 的概率选择 Q 值最大的动作。在在线运行阶段，为了减少运行时间，提升收敛速度，不对 Q 网络参数进行更新，采用贪心策略选择 Q 值最大的动作[21]，如算法 2 所示。

算法 1 DDQL 训练阶段流程

输入 系统环境参数、任务参数和 DDQL 算法参数

输出 当前 Q 网络参数 $\boldsymbol{\theta}$

1) 初始化回放记忆单元

2) 用随机参数 $\boldsymbol{\theta}$ 初始化当前 Q 网络 ϕ，用参数 $\boldsymbol{\theta}^-=\boldsymbol{\theta}$ 初始化目标 Q 网络 $\hat{\phi}$

3) for episode=1:MaxEpisode

4) 初始化状态 $\boldsymbol{s}_1$，即初始化每个任务的 EG 选择情况、EG 获得发射功率和 ES 获得时钟频率

5) for t=1:MaxStep

6) 用 ε-贪心策略选择动作 a_t 并执行，改变某一任务的 EG 选择情况、EG 发射功率或 ES 时钟频率，或者均不改变

7) 观察下一状态 $\boldsymbol{s}_{t+1}$

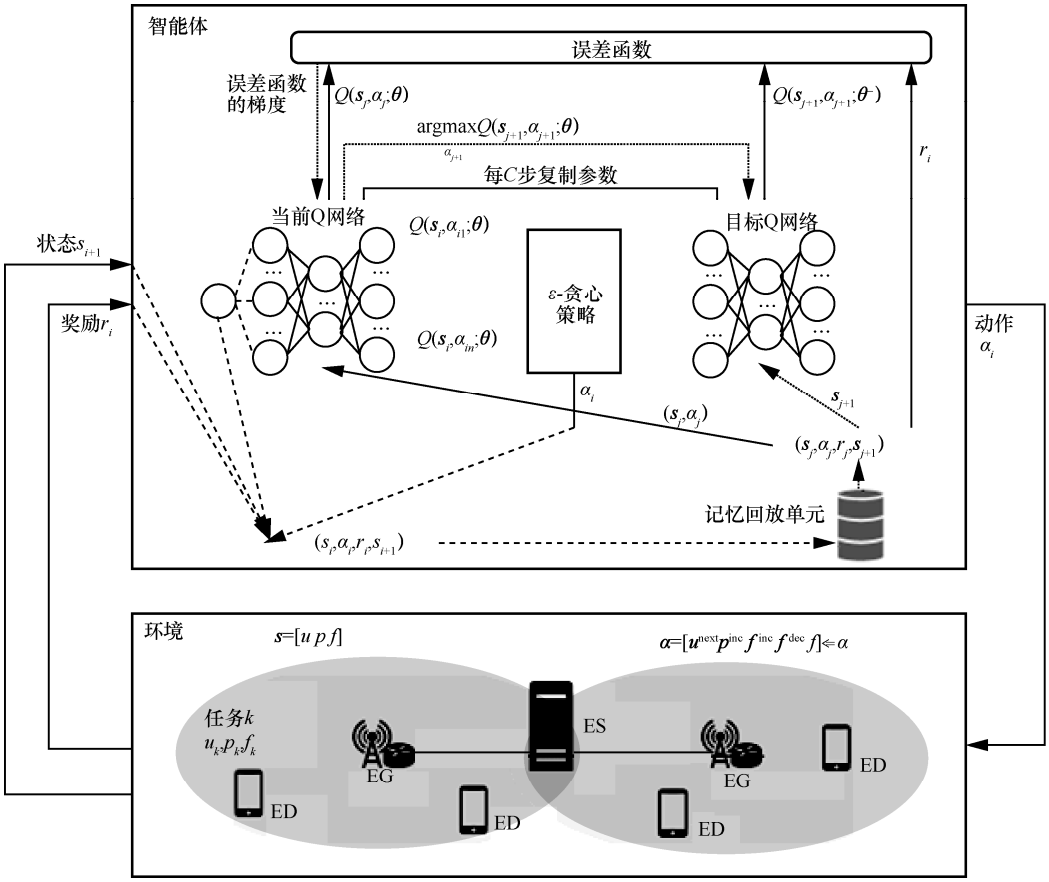


图 2　DDQL 算法框架

8)　判断状态 $\boldsymbol{s}_{t+1}$ 是否满足约束条件，计算相应任务平均能耗和奖励 r_t

9)　将状态转化 $(\boldsymbol{s}_t,a_t,r_t,\boldsymbol{s}_{t+1})$ 存入回放记忆单元

10)　从回放记忆单元中随机抽取一批状态转化

11)　对其中状态转化 $(\boldsymbol{s}_j,a_j,r_j,\boldsymbol{s}_{j+1})$ 根据式(31)计算目标 Q 值，根据式(32)计算误差，更新参数 $\boldsymbol{\theta}$

12)　每隔 C 步更新目标 Q 网络，即 $\boldsymbol{\theta}^-=\boldsymbol{\theta}$

13)　end for

14) end for

算法 2　DDQL 在线运行阶段流程

输入　系统环境参数、任务参数、DDQL 算法参数和当前 Q 网络参数 $\boldsymbol{\theta}$、当前状态 $\boldsymbol{s}_1$

输出　最终状态 $\boldsymbol{s}_{\text{MaxStep}+1}$

1) for t=1:MaxStep

2)　选择动作 $a_t=\arg\max_a Q(\boldsymbol{s}_t,a;\boldsymbol{\theta})$ 并执行

3)　观察下一状态 $\boldsymbol{s}_{t+1}$

4)　判断状态 $\boldsymbol{s}_{t+1}$ 是否满足约束条件，计算相应任务平均能耗和奖励 r_t

5) end for

3.4　对比算法介绍

为验证本文提出的基于 DDQL 的求解方法的效果，将 RA、GA、PSO 算法、DQL 算法作为对比算法。以下对几种对比算法进行简要介绍。

1)　RA：随机选择 EG 与资源进行分配，直到满足约束条件为止。

2)　GA：贪心策略是给每个任务分配尽量小的 EG 发射功率和 ES 时钟频率。首先给每个任务分配 $P_{\min}$ 的 EG 发射功率和 $F_{\min}$ 的 ES 时钟频率，若无法满足约束条件，再依次给每个任务按照与 DDQL 算法相同的步长增加分配的资源，直到满足约束条件为止。

3) PSO 算法：PSO 算法是一种模拟鸟类行为的群体智能优化算法。首先初始化一群例子，粒子具有位置、速度和适应度特征，每个粒子的位置代表一个可能的解。在每次迭代中，粒子通过个体极值

Pbest 和群体极值 Gbest 更新自身速度，通过速度改变位置，重新计算适应度，并更新 Pbest 和 Gbest。

每个粒子的位置即为 DDQL 算法中定义的状态 $\boldsymbol{s}$，共 N 维，$N=3K$。因此，对粒子的位置、速度等进行如下定义。

第 i 个粒子的位置表示为 $\boldsymbol{x}_i=[x_i^1,x_i^2,\cdots,x_i^N]$，第 i 个粒子的速度表示为 $\boldsymbol{v}_i=[v_i^1,v_i^2,\cdots,v_i^N]$，第 i 个粒子的最优位置表示为 $\boldsymbol{\varphi}_i=[\varphi_i^1,\varphi_i^2,\cdots,\varphi_i^N]$，所有粒子的最优位置表示为 $\boldsymbol{\varphi}_g=[\varphi_g^1,\varphi_g^2,\cdots,\varphi_g^N]$。其中，$i\in[1,M]$ 表示粒子编号，M 为粒子数量。粒子速度更新式为

$$v_i^n=\omega v_i^n+c_1r_1(\varphi_i^n-x_i^n)+c_2r_2(\varphi_g^n-x_i^n) \tag{33}$$

其中，$n\in[1,N]$ 代表维度编号；ω 为惯性因子，其取值范围为非负；c_1,c_2 为加速常数，前者为每个粒子的个体学习因子，后者为社会学习因子，取值范围均为非负；r_1,r_2 为 2 个 $[0,1]$ 内的随机数。

之后，检查每个粒子每一维度的速度，若超出 $[v_{\min},v_{\max}]$ 的范围，则对速度进行修正。位置更新式为

$$x_i^n=x_i^n+v_i^n \tag{34}$$

x_i^n 的取值范围是 $[x_{\min}^n,x_{\max}^n]$。同样需要对每个粒子每一维度的位置进行范围检查，对超出范围的进行修正。

适应度函数是评价粒子位置的指标，最优位置是适应度最大的位置。适应度的定义与 DDQL 算法中的奖励相同，即式(25)。

4) DQL 算法：已在 3.3 节中进行介绍，在此不再赘述。

3.5 时间复杂度分析

针对本文方法和对比算法的时间复杂度分析如下。

RA。设找到可行解需要的迭代步数为 T^{RA}，则 RA 的时间复杂度为 $O(T^{\mathrm{RA}}K)$。由于 RA 是随机进行资源分配，T^{RA} 的随机性也较大。

GA。设找到可行解需要的迭代步数为 T^{GA}，则 GA 的时间复杂度为 $O(T^{\mathrm{GA}}K)$。在任务数较小、资源不紧缺的情况下，T^{GA} 一般也较小，随着任务数的增多，需要更多的迭代次数以找到满足约束的可行解。

PSO 算法。设迭代总步数为 T^{PSO}，则 PSO 算法的时间复杂度为 $O(T^{\mathrm{PSO}}MK)$。

DDQL 算法。训练阶段的时间复杂度需要考虑训练 Q 网络的时间复杂度和训练 Q 网络的次数两部分。在训练 Q 网络的过程中，需要对每相邻两层神经元之间的连接权重进行更新，设 Q 网络的层数为 nl，第 i 层中神经元的个数为 n_i，每次训练中的迭代次数为 T^{udp}，则训练一次 Q 网络的时间复杂度为 $O\left(T^{\mathrm{udp}}K\left(\sum_{i=1}^{nl-1}n_in_{i+1}\right)\right)$。记回合数为 T^{Epi}，每回合中步数为 T^{Step}，则训练 Q 网络的次数为 $T^{\mathrm{Epi}}T^{\mathrm{Step}}$，因此，DDQL 训练阶段的时间复杂度 $O\left(T^{\mathrm{Epi}}T^{\mathrm{Step}}T^{\mathrm{udp}}K\left(\sum_{i=1}^{nl-1}n_in_{i+1}\right)\right)$。使用早停、随机失活等技巧来优化神经网络训练，会对时间复杂度产生一定影响，因此以上结果为近似结果。DDQL 算法运行阶段的时间复杂度为 $O(T^{\mathrm{Step}}K)$。DDQL 算法在线训练阶段的时间复杂度较高，但将 Q 网络训练好后，运行阶段不需要更新 Q 网络且只需进行一个回合，时间复杂度低，运行时间短，可以满足实时网络条件下对在线决策时间的要求。因此，本文在对比不同算法的时间复杂度时，使用运行阶段的时间复杂度。

DQL 算法。算法的时间复杂度与 DDQL 算法相同。

综上所述，各算法的时间复杂度如表 1 所示。

表 1 算法时间复杂度

算法	时间复杂度
暴力搜索	$O\left(K\prod_{k\in\mathcal{K}}(N_k^uN_k^pN_k^f{}_k)\right)$
RA	$O(T^{\mathrm{RA}}K)$
GA	$O(T^{\mathrm{GA}}K)$
PSO	$O(T^{\mathrm{PSO}}MK)$
DDQL/DQL	$O(T^{\mathrm{Step}}K)$

在忽略任务数对迭代步数影响的情况下，RA、GA、PSO、DDQL、DQL 等算法的时间复杂度和任务数 K 成线性关系，相比于具有指数级时间复杂度的暴力搜索算法，时间复杂度显著下降。

4 仿真实验

本章对提出的基于 DDQL 的资源分配方法进行仿真实验。首先，对仿真场景和仿真参数进行说明；然后，展示仿真结果，并对其进行分析。

4.1 仿真场景与仿真参数设置

在仿真实验中，考虑多边缘服务器、多边缘网关、多终端设备的仿真场景，如图 3 所示。考虑 900 m×900 m 的网络覆盖范围，其中包含 4 个边

缘服务器，11 个边缘网关，以及若干终端设备，其数量可设定，位置随机。边缘网关部署在基站侧，每个基站的覆盖范围的半径为 200 m，图 3 中以维诺图的形式表示基站的覆盖范围。边缘服务器与边缘网关连接关系固定，边缘网关与终端设备的连接关系需要后续通过算法进行决策。

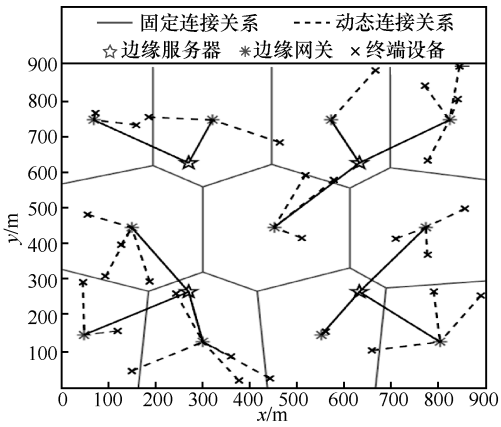


图 3　仿真场景

根据文献[28-31]设置默认情况下的系统参数，如表 2 所示。假设每个 ED 发起一个任务，其余的任务相关的参数随机生成，在所给范围内均匀分布。

表 2　　系统参数设置

参数	数值
带宽 $B_{g,d}$ / MHz	10
路径损耗 $h_{g,d}$ / dB	15.3+37.6lg($D_{g,d}$)
噪声功率谱密度 N_0 /(dBm·Hz^{-1})	−100
EG　g 最大发射功率 P_g /W	10
ES　s 最大时钟频率 F_s / GHz	20
ES　s 存储空间 B_s / TB	1
计算能耗参数 κ	10^{-27}
每个任务最大发射功率 $P_{\max}$ / W	1
每个任务最小发射功率 $P_{\min}$ / W	0.1
每个任务最大时钟频率 $F_{\max}$ / GHz	2
每个任务最小时钟频率 $F_{\min}$ / GHz	0.5
任务 k 返回数据量 l_k /MB	[1,10]
任务 k 所需存储空间 b_k /MB	[10,1 024]
任务 k 所需计算周期数 c_k /Mcycle	[10,500]
任务 k 最大时延 T_k /s	[0.1,1]
任务最大能耗 $E_{\max}$ / J	1

根据文献[32]设置 DDQL 和 DQL 算法参数，如表 3 所示。表 4 为 PSO 算法参数设置。

表 3　　DDQL 和 DQL 算法参数设置

参数	数值
折扣因子 γ	0.9
探索利用率 ε 最大值	0.9
探索利用率 ε 最小值	0.1
探索利用率 ε 减小速率/步	0.01
回放记忆单元容量/个	500
抽样大小/个	32
DNN 各层神经元个数/个	$3K,64,32,128,5K+1$

表 4　　PSO 算法参数设置

参数	数值
粒子群规模 M	50
个体经验学习因子 c_1	1
社会经验学习因子 c_2	1
惯性因子 ω	0.6
最大速度 $v_{\max}$	2
最小速度 $v_{\min}$	−2

4.2　仿真结果与分析

本文通过 MATLAB 建立数值仿真环境评估所提算法的性能。

DDQL 和 DQL 算法需要在实际运行前进行 Q 网络的训练。图 4 为 2 种算法在训练过程中的 Q 值变化情况。Q 值起初都在 0 附近，随着回合数增加，Q 值先逐渐增加，而后趋于稳定。DDQL 算法的 Q 值在 100 回合左右收敛，DQL 算法的 Q 值在 300 回合左右收敛。相比于 DQL 算法，DDQL 算法在训练阶段具有更快的收敛速度。并且，DQL 算法的 Q 值明显大于 DDQL 算法的 Q 值，反映出 DQL 算法存在 Q 值过估计的问题。

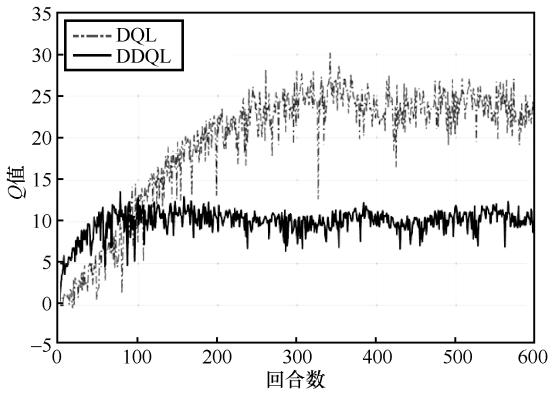


图 4　DDQL 算法和 DQL 算法训练阶段 Q 值变化情况

接下来对算法在在线运行阶段的性能进行

评估与分析。

图 5 为不同算法在不同任务数下的收敛步数对比。其中，GA 的收敛步数是指在找到可行解之前的迭代次数，找到可行解后算法停止。PSO 算法、DQL 算法、DDQL 算法的收敛步数是指结果趋于稳定前经过的迭代次数。GA、PSO 算法在任务数为 10 时，收敛步数很少，但随着任务数增加，GA 的收敛步数迅速增加，而 PSO 的收敛步数也在任务数大于 60 之后明显增加。这是因为随着任务数增加，资源逐渐紧张，需要更多的步数来搜索可行解并优化至收敛。相比之下，在任务数少时，DDQL 算法和 DQL 算法的收敛步数略多于 GA 与 PSO，但随着任务数增加，DDQL 算法和 DQL 算法的收敛步数也基本稳定，在状态与动作维度较高的情况下也显示出了良好的收敛性。并且，DDQL 算法的收敛步数总体上少于 DQL 算法。

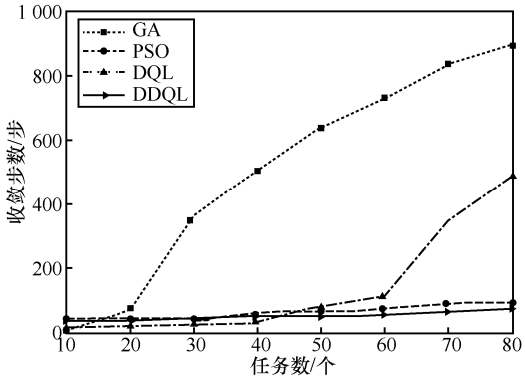


图 5 不同算法收敛步数对比

图 6~图 8 是任务数为 50 时，PSO、DQL 和 DDQL 算法运行阶段的变化情况。图 6 为任务平均能耗变化情况。PSO 虽然收敛速度快，在 10 步就收敛，但是过早地陷入了局部最优解，最终任务平均能耗为 0.356 J。由于 PSO 算法会维护历史群体最优值，所以迭代过程中，任务平均能耗只会单调减少，不会出现起伏波动。DQL 算法的收敛步数略多，在 38 步收敛，最终任务平均能耗为 0.321 J。DDQL 算法的收敛步数在 PSO 算法和 DQL 算法之间，DDQL 算法在第 21 步收敛，最终任务平均能耗为 0.291 J，比 PSO 算法少 18.3%，比 DQL 算法少 9.4%。

图 7 为任务平均获得的 EG 发射功率与 ES 时钟频率变化情况，其收敛情况与图 6 相吻合。最终，在 PSO、DQL 和 DDQL 算法下，任务平均获得的 EG 发射功率分别为 0.52 W、0.64 W 和 0.59 W，ES 时钟频率分别为 1.15 GHz、1.20 GHz 和 1.09 GHz。

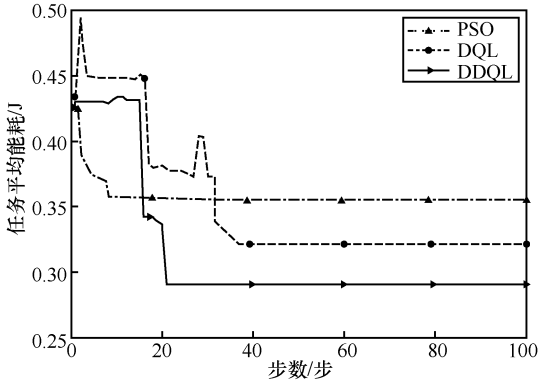
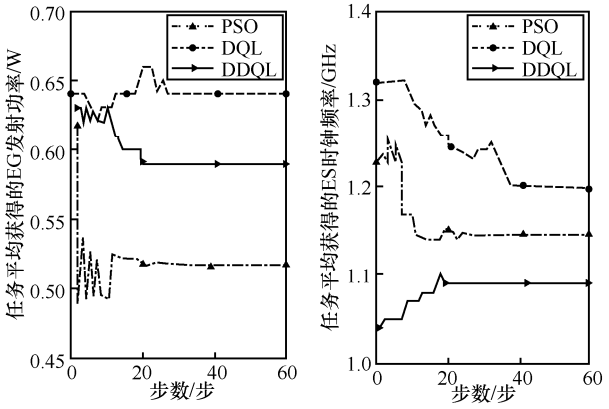


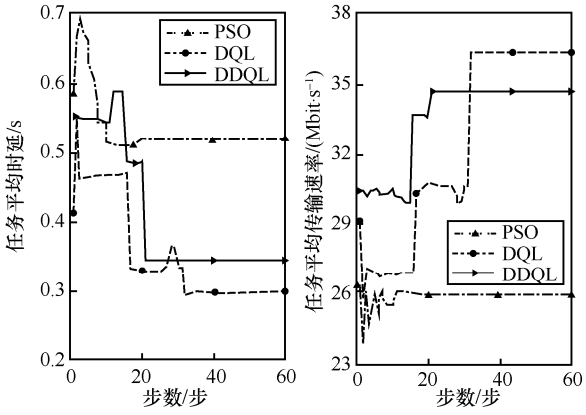
图 6 任务平均能耗变化情况



(a) 任务平均 EG 发射功率变化情况 (b) 任务平均 ES 时钟频率变化情况

图 7 资源分配变化情况

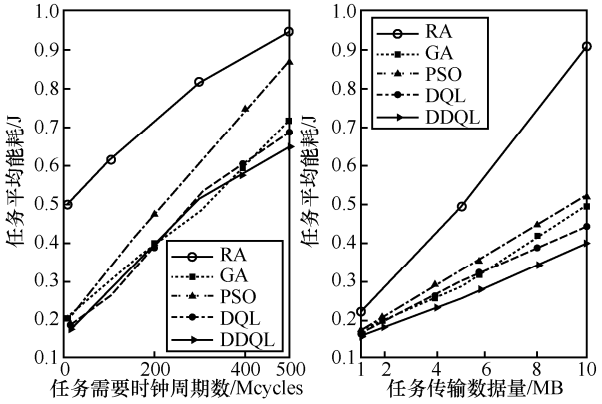
图 8 为任务平均时延与传输速率变化情况。3 种算法经过迭代优化，在任务平均能耗减小的同时，任务平均时延减少，任务平均传输速率增加，提升了用户 QoS，系统获得了更好的性能。但 DQL 用多于 DDQL 算法的任务平均能耗，获得了更低的任务平均时延和更高的传输速率，反映了能耗与性能存在一定的折中关系。



(a) 时延变化情况 (b)传输速率变化情况

图 8 任务平均时延与传输速率变化情况

图 9 为任务数为 50 时，任务平均能耗与任务需要的 CPU 时钟周期数和任务传输数据量的关系图。每个算法在每个测试任务数据量下进行 100 组实验，对结果取平均值。基于 DDQL 的算法比基于 RA、GA、PSO 和 DQL 的算法分别降低了 46.0%、10.2%、18.6%和 5.4%的任务平均能耗。基于 DDQL 的资源分配方法能有效降低任务平均能耗。



(a) 时钟周期数关系　　(b)与传输数据量关系

图 9　任务平均能耗与任务数据量关系

图 10 为任务平均能耗随任务数变化情况。由图 9 可以看出，任务需要的 CPU 时钟周期数和任务传输数据量对任务平均能耗影响较大，因此，在进行任务平均能耗与任务数关系的仿真实验时，将任务需要的 CPU 时钟周期数均设为 300 Mcycle，任务传输数据量均设为 6 MB。在每个测试任务数下，进行 100 组实验，对结果取平均值。由图 10 可以看出，随着任务数增加，任务平均能耗也增加，但增长幅度较小。不同的算法对最终的任务平均能耗影响较大。基于 DDQL 的算法比基于 RA、GA、PSO 和 DQL 的算法分别降低了 65.0%、21.5%、37.4%和 5.0%的任务平均能耗。

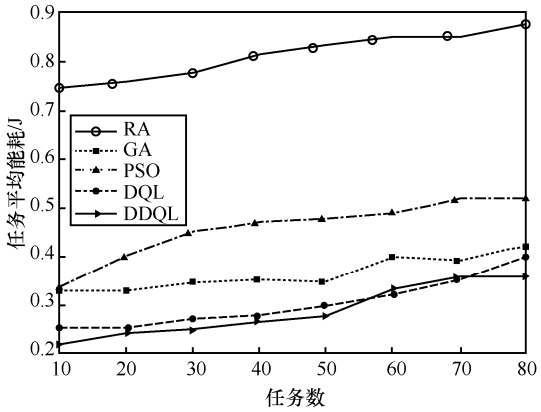


图 10　任务平均能耗与任务数关系

综上，本文通过仿真实验，验证了提出的基于 DDQL 的求解方法对解决多任务资源分配问题的有效性。训练过程与运行结果能够收敛，在训练阶段具有比 DQL 算法更快的收敛速度；在运行阶段，当任务数较多时，相比于 GA、PSO 算法，DDQL 算法收敛步数优势明显。运行中，DDQL 算法在降低任务平均能耗的同时，也能对任务平均时延与传输速率进行一定程度的优化。相比基于 RA、GA、PSO 算法、DQL 算法的方法，基于 DDQL 算法的边缘网络资源分配方法能有效降低任务平均能耗。

5　结束语

本文对移动边缘网络资源分配方法进行研究。考虑任务完成时延限制和通信、计算、存储资源限制等约束条件，建立任务平均能耗最小化的资源分配模型，并提出基于 DDQL 的求解方法，相比基于 RA、GA、PSO、DQL 的多种求解方法，降低了至少 5%的任务平均能耗。本文提出的算法为移动边缘网络中低能耗资源分配方法提供了一种有借鉴意义的参考。

本文还存在一些不足之处，需进一步改进与优化。例如，在优化模型上，需要考虑在云中心、边缘节点、终端设备协同配合的场景下，对计算卸载位置、各类资源分配等进行联合决策与优化；同时考虑上下行流量的传输过程，建立更通用的模型。在算法优化上，可考虑使用能直接对连续动作空间进行优化的方法，来避免动作步长对结果产生的影响。例如，目前奖励设置采用的是约束判别方法，后续需要考虑更为高级的处理方法，如将约束叠加至目标中。此外，目前 DDQL 算法的超参数靠人工设定，后续需要研究算法的加速机制和参数自适应设置方式，并探讨将算法用于实际系统中的可行性。

参考文献：

[1] 陈亮, 余少华. 6G 移动通信发展趋势初探（特邀）[J]. 光通信研究, 2019, 45(4): 1-8.
CHEN L, YU S H. Preliminary Study on the Trend of 6G Mobile Communication (special invitation)[J]. Study on Optical Communications, 2019, 45(4): 1-8.

[2] 田辉, 范绍帅, 吕昕晨, 等.面向 5G 需求的移动边缘计算[J]. 北京邮电大学学报,2017,40(2):1-10.
TIAN H, FAN S S, LYU X C, et al. Mobile edge computing for 5G requirements[J]. Journal of Beijing University of Posts and Telecommunications,2017,40(2):1-10.

[3] STRINATI E C, BARBAROSSA S, GONZALEZ-JIMENEZ J L, et al. 6G: the next frontier: from holographic messaging to artificial intelligence using subterahertz and visible light communication[J]. IEEE Vehicular Technology Magazine, 2019, 14(3): 42-50.

[4] 毕成. 5G 无线网络节能技术研究[J].电信工程技术与标准化, 2020, 33(8): 1-5.
BI C. Research on energy saving technology of 5G wireless network[J]. Telecom Engineering Technics and Standardization, 2020, 33(8):1-5.

[5] 赵亚军, 郁光辉, 徐汉青. 6G 移动通信网络:愿景、挑战与关键技术[J]. 中国科学:信息科学, 2019，8(49): 963-987.
ZHAO Y J, YU G H, XU H Q. 6G mobile communication networks: vision, challenges, and key technologies[J]. Scientia Sinica (Informationis), 2019, 8(49): 963-987.

[6] CAI S, CHE Y, DUAN J, et al. Green 5G heterogeneous networks through dynamic small-cell operation[J]. IEEE Journal on Selected Areas in Communications, 2016, 34(5): 1103-1115.

[7] ZHOU Y, TIAN L, LIU L, et al. Fog computing enabled future mobile communication networks: a convergence of communication and computing[J]. IEEE Communications Magazine, 2019, 57(5): 20-27.

[8] LIU L, ZHOU Y, YUAN J, et al. Economically optimal MS association for multimedia content delivery in cache-enabled heterogeneous cloud radio access networks[J]. IEEE Journal on Selected Areas in Communications, 2019, 37(7): 1584-1593.

[9] BI S, HUANG L, ZHANG Y J A. Joint optimization of service caching placement and computation offloading in mobile edge computing systems[J]. IEEE Transactions on Wireless Communications, 2020, 19(7): 4947-4963.

[10] MAO Y, ZHANG J, LETAIEF K B. Dynamic computation offloading for mobile-edge computing with energy harvesting devices[J]. IEEE Journal on Selected Areas in Communications, 2016, 34(12): 3590-3605.

[11] LIU L, ZHOU Y, GARCIA V, et al. Load aware joint CoMP clustering and inter-cell resource scheduling in heterogeneous ultra dense cellular networks[J]. IEEE Transactions on Vehicular Technology, 2017, 67(3): 2741-2755.

[12] BADRI H, BAHREINI T, GROSU D, et al. Risk-based optimization of resource provisioning in mobile edge computing[C]//2018 IEEE/ACM Symposium on Edge Computing (SEC). Piscataway: IEEE Press, 2018: 328-330.

[13] WANG L, JIAO L, LI J, et al. Moera: mobility-agnostic online resource allocation for edge computing[J]. IEEE Transactions on Mobile Computing, 2019, 18(8):1843-1856.

[14] CHEN X, JIAO L, LI W, et al. Efficient multi-user computation offloading for mobile-edge cloud computing[J]. IEEE/ACM Transactions on Networking, 2015, 24(5): 2795-2808.

[15] ZENG D, GU L, PAN S, et al. Resource management at the network edge: a deep reinforcement learning approach[J]. IEEE Network, 2019, 33(3): 26-33.

[16] 梁应敞, 谭俊杰, NIYATO D. 智能无线通信技术研究概况[J].通信学报, 2020, 41(7): 1-17.
LIANG Y C, TAN J J, NIYATO D. Overview on intelligent wireless communication technology[J]. Journal on Communications, 2020, 41(7): 1-17.

[17] XU J, CHEN L, REN S. Online learning for offloading and autoscaling in energy harvesting mobile edge computing[J]. IEEE Transactions on Cognitive Communications and Networking, 2017, 3(3): 361-373.

[18] ZHANG C, PATRAS P, HADDADI H. Deep learning in mobile and wireless networking: a survey[J]. IEEE Communications Surveys & Tutorials, 2019, 21(3): 2224-2287.

[19] ZHAO N, LIANG Y C, NIYATO D, et al. Deep reinforcement learning for user association and resource allocation in heterogeneous cellular networks[J]. IEEE Transactions on Wireless Communications, 2019, 18(11): 5141-5152.

[20] LI J, GAO H, LYU T, et al. Deep reinforcement learning based computation offloading and resource allocation for MEC[C]//2018 IEEE Wireless Communications and Networking Conference. Piscataway: IEEE Press, 2018: 1-6.

[21] WANG J, ZHAO L, LIU J, et al. Smart resource allocation for mobile edge computing: a deep reinforcement learning approach[J]. IEEE Transactions on Emerging Topics in Computing, 2019, doi: 10.1109/TETC.2019.2902661.

[22] LI Y, QI F, WANG Z, et al. Distributed edge computing offloading algorithm based on deep reinforcement learning[J]. IEEE Access, 2020, 8: 85204-85215.

[23] VAN H H, GUEZ A, SILVER D. Deep reinforcement learning with double Q-learning[C]//Proceedings of the AAAI Conference on Artificial Intelligence. Palo Alto: AAAI Press, 2016: 2094-2100.

[24] CHEN X, ZHANG H, WU C, et al. Optimized computation of-floading performance in virtual edge computing systems via deep reinforcement learning[J]. IEEE Internet of Things Journal, 2018, 6(3), 4005-4018.

[25] MAO Y, YOU C, ZHANG J, et al. A survey on mobile edge computing: the communication perspective[J]. IEEE Communications Surveys & Tutorials, 2017, 19(4): 2322-2358.

[26] 刘全, 翟建伟, 章宗长, 等. 深度强化学习综述[J]. 计算机学报, 2018, 41(1): 1-27.
LIU Q, ZHAI J W, ZHANG Z Z, et al. A survey on deep reinforcement learning[J]. Chinese Journal of Computers, 2018, 41(1): 1-27.

[27] MNIH V , KAVUKCUOGLU K , SILVER D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529-533.

[28] CAO X, WANG F, XU J, et al. Joint computation and communication cooperation for energy-efficient mobile edge computing[J]. IEEE Internet of Things Journal, 2018, 6(3): 4188-4200.

[29] TRAN T X, POMPILI D. Joint task offloading and resource allocation for multi-server mobile-edge computing networks[J]. IEEE Transactions on Vehicular Technology, 2018, 68(1): 856-868.

[30] YANG X, YU P, FENG L, et al. A deep reinforcement learning based mechanism for cell outage compensation in 5G UDN[C]//2019 IFIP/IEEE Symposium on Integrated Network and Service Management. Piscataway: IEEE Press, 2019: 476-481.

[31] WANG S, PAN C, YIN C. Joint heterogeneous tasks offloading and

resource allocation in mobile edge computing systems[C]//2018 10th International Conference on Wireless Communications and Signal Processing. Piscataway: IEEE Press, 2018: 1-6.

[32] ARULKUMARAN K, DEISENROTH M P, BRUNDAGE M, et al. Deep reinforcement learning: a brief survey[J]. IEEE Signal Processing Magazine, 2017, 34(6): 26-38.

[作者简介]

喻鹏（1986− ），男，湖北随州人，博士，北京邮电大学副教授、博士生导师，主要研究方向为 5G/6G 网络智能管控等。

张俊也（1998− ），女，山西太原人，北京邮电大学硕士生，主要研究方向为 5G/6G 网络智能管控、移动边缘计算等。

李文璟（1973− ），女，山西太谷人，博士，北京邮电大学教授、博士生导师，主要研究方向为无线网络管理和自组织网络等。

周凡钦（1988− ），男，河南浑池人，博士，北京邮电大学在站博士后，主要研究方向为无线异构网络的资源管理和负载均衡等。

丰雷（1987− ），男，北京人，北京邮电大学副教授、硕士生导师，主要研究方向为无线网络和智能电网的资源管理等。

付澍（1985− ），男，贵州贵阳人，博士，重庆大学副教授、硕士生导师，主要研究方向为星地通信、NOMA、物联网、网络一体化等。

邱雪松（1973− ），男，江西上饶人，博士，北京邮电大学教授、博士生导师，主要研究方向为网络管理与通信软件等。