绿色算力网络中智能匹配任务卸载方案

兰世战[1,2]，马丽芳[3]

（1. 华南理工大学软件学院，广东 广州 510006；
2. 中国移动通信集团广西有限公司，广西 南宁 530012；
3. 广西建设职业技术学院信息工程学院，广西 南宁 530007）

摘　要：算力网络通过整合云、边、端算力资源，为数字经济提供数据感知、传输、运算等一体化服务，但其高速发展伴随亟待解决的高能耗问题。任务卸载技术通过合理分配计算任务，提升用户体验、降低传输时延并减少能耗，是一种重要的解决方案。为降低算力网络的整体能耗，实现绿色可持续发展，提出了一种基于匹配机制的智能匹配任务卸载方案。通过匹配算力网络中的任务与节点资源，减少因不合理任务卸载产生的能源消耗，提升算力网络的整体性能表现。同时采用结合强化学习与神经网络的深度学习方法，进一步优化卸载策略，显著降低算力网络能耗。仿真实验验证表明，该方法具有良好的有效性与可靠性。

关键词：算力网络；智能匹配；强化学习；深度学习

中图分类号：TP18

文献标志码：A

doi: 10.11959/j.issn.1000−0801.2025030

Intelligent matching task offloading scheme in green computing power networks

LAN Shizhan[1,2], MA Lifang[3]

1. School of Software Engineering, South China University of Technology, Guangzhou 510006, China
2. China Mobile Group Guangxi Co., Ltd., Nanning 530012, China
3. Faculty of Information Engineering, Guangxi Polytechnic of Construction, Nanning 530007, China

Abstract: By integrating cloud, edge, and device resources, the computing power networks provide integrated services such as data sensing, transmission, and computation for the digital economy. However, their rapid development is accompanied by pressing challenges of high energy consumption. The task offloading technology is an important solution that allocates computing tasks properly, improves user experience, reduces transmission latency and energy consumption, making it a crucial solution. In order to reduce the overall energy consumption of computing power networks and achieve green, sustainable development, an intelligent matching task unloading scheme based on matching mechanism was proposed. By matching tasks and node resources in the computing power network, the scheme mini-

收稿日期：2024−09−23；修回日期：2024−12−26
通信作者：兰世战，2773355@qq.com
基金项目：国家自然科学基金面上项目（No.62471208）
Foundation Item: The National Natural Science Foundation of China (No.62471208)

mized energy consumption caused by inefficient task offloading and enhances overall network performance. Furthermore, a deep learning approach combining reinforcement learning with neural networks was employed to further optimize the offloading strategy, significantly reducing network energy consumption. Simulation experiments demonstrate that the proposed method is effective and reliable.

Key words: computing power network, intelligent matching, reinforcement learning, deep learning

0　引言

随着社会经济的发展，数字经济的引擎作用日益突出，算力就是未来的生产力。到2025年，中国数字经济占总GDP比例达41.5%，且计算力指数平均每提高一个点，国内的数字经济和GDP将分别增长3.5‰和1.8‰[1-2]。数字经济的发展将推动海量数据产生，数据处理需要云、边、端协同的强大算力和广泛覆盖的网络连接。然而，对任务和资源的粗略分析导致的能源浪费，对于致力于环保的算力网络来说无疑是一种巨大的打击[3-4]。由于对算力任务和资源的粗略利用，通常无法充分利用算力资源的潜力。此外，为了提供多样化的算力服务，算力网络需要根据需求调整计算服务，而这每次都会引发资源和服务的竞争[5-7]。另外，由于当前算力网络资源有限，多个计算服务会竞争有限的资源，从而导致服务质量下降[8-10]。

采用计算卸载方案解决算力网络中任务竞争资源的问题是研究热点。文献[11]考虑了多用户的任务卸载问题，设计了一种迭代方案动态做出卸载决策，提高了执行效率并降低了时延。文献[12]采用了近似动态规划最小化随机流量下时延和能耗问题，将动态优化问题建模为无限时域下平均奖励连续时间马尔可夫决策过程模型。文献[13]考虑了任务和子任务之间的依赖性，并且研究了超密集的网络结构中任务依赖关系，提出了一种启发式的算法，在保证子任务依赖性的同时提高任务的执行效率，降低超密集网络中的任务时延。以上的算法均考虑了降低时延的目标，但是没有综合考虑资源和任务的复杂关系，也没能均衡时延和能耗的平衡关系。

文献[14]研究了能量和时延之间的均衡问题，并且设计了一种能量感知方案来联合优化通信和计算的资源配比，用一种迭代搜索算法寻找混合整数非线性问题的最优解。文献[15]提出了建立斯塔克伯格博弈来处理网络中各方的收益问题，通过多层博弈的方案降低时延和传输能耗，达到收益平衡的方案。文献[16]为了均衡时延和能耗问题，用队列的方案检测和控制网络拥塞，缓解拥塞造成的时延和能耗传输浪费，从而智能地决策处理数据服务包。以上研究虽然都考虑了能耗和时延的均衡问题，但是没有考虑算力网络中任务和资源的匹配能力，未能细化资源侧的性能表现和任务侧的任务依赖关系。

为了克服传统优化算法中难以快速解决决策优化问题的短板，一些深度学习方案得到了研究。文献[17]采用了深度学习在线卸载方案提升了决策的计算效率。文献[18]提出了一种深度强化学习框架来应用在任务卸载场景中，用于评估不同算法的性能和表现。文献[19]使用了深度学习方案解决任务卸载的安全性问题，关注了算力网络卸载问题的数据隐私和能耗表现。但是以上的方案缺少对变化环境的适应能力。

经过分析，目前算力网络还存在以下的问题。

（1）算力网络资源的粗略表示和分析会影响任务的卸载表现，进而影响整个算力网络能耗表现。在算力网络中，粗略的资源表示可能无法准确反映计算资源的实际性能和特点，而粗略的资源分析阶段，无法充分考虑资源的多维属性，如计算能力、网络带宽等，可能会导致资源分配不

合理，进而影响任务的处理效率和能耗。

（2）算力网络任务的粗略分析和粗略卸载会忽视任务间存在的依赖关系，反向影响任务本身的执行时延。忽视任务间复杂且关键的依赖关系，可能会导致任务执行顺序的混乱，甚至可能使得某些任务因等待其前置任务的结果而被迫增加处理时延。这种反向影响会显著增加系统整体的处理时延，进而降低整个算力网络的运行效率。

针对以上的问题，本文提出了绿色算力网络中的智能匹配任务卸载方案，主要的贡献有以下3点。

（1）本文提出了考虑任务依赖关系的任务侧建模方案。将一个实际应用程序分解为多个具有依赖关系的子任务，并且每个子任务的启动都严格依赖于其前置子任务的顺利完成。通过将任务的依赖关系作为分析考量因素，优化任务卸载与资源分配，以实现任务执行与资源分配的最优化协同。

（2）综合考虑资源与任务的复杂关系，建立资源和任务的匹配模型以制定任务卸载策略。通过向量的分类思想衡量任务与资源池的性能差距，并以欧几里得距离量化任务与资源之间的匹配度。通过将任务卸载问题转化为对匹配度进行优化分配的问题，基于整个有向无环图（directed acyclic graph，DAG）任务的总匹配度为任务卸载策略的制定提供优化依据。

（3）采用深度强化学习技术挖掘任务和资源的隐藏信息，适应算力网络的复杂环境。创新性地采用改进的Seq2Seq（sequence to sequence）网络模型进行深度学习过程，并结合元强化学习算法与近端策略优化，从大规模学习任务中采样训练，有效地挖掘任务和资源的隐藏信息，以显著提升算法在算力网络复杂环境中的适应性。

1 相关工作

1.1 绿色算力网络中的任务卸载

任务卸载通过将计算密集型任务从资源受限的节点转移到能力更强的节点或服务器上，实现了分布式计算资源的高效利用。广泛地研究探索了任务卸载策略，以优化各种目标，如时延[20-21]、能量使用[22]，以及时延和能量[23-24]之间的平衡。文献[20]使用了拉格朗日对偶方法获得最优任务卸载和资源分配策略，以有效地分配设备、边缘和云服务器所需的计算资源，并最小化任务处理时延。文献[21]开发了一个竞争博弈论模型，用于共享通信信道上用户之间的任务卸载，重点是减少每个人的能源使用。文献[22]将问题表述为混合整数非线性规划问题，并提出了一种双深度Q网络（double deep Q-network，DDQN）方法，以确定计算卸载和资源分配的联合策略，最大限度地降低整个移动边缘计算系统的能耗。文献[23]提出了一种时间注意力确定性策略梯度，基于深度强化学习深度确定策略梯度（deep deterministic policy gradient，DDPG）方法，解决动态移动边缘计算环境中的去中心化计算卸载和资源分配的联合优化问题，以最小化任务完成时间和能耗。文献[24]应用随机优化将随机任务卸载挑战转化为确定性问题，设计了一种在线节能卸载算法。然而，大多数研究考虑的是粗粒度的资源或任务，限制了其对绿色算力网络中遇到的各种复杂场景的适用性。

1.2 任务卸载技术

为了解决任务卸载的问题，目前已经涌现了各种各样的研究成果，包括提出了启发式算法[25]、数学优化[12]和基于强化学习的方法[18,26-27]。文献[25]将卸载问题定义为一个混合整数线性规划，并开发了一个启发式解决方案。文献[12]使用近似动态规划解决了窄带物联网（narrow band Internet of things，NB-IoT）边缘计算环境中计算

卸载和多用户调度的综合问题。由于强化学习允许代理参与环境并适应状态和奖励动态的变化，因此对卸载决策的使用进行了彻底的研究。文献[18]实现了深度强化学习来解决无线电供电移动边缘计算设置中的二进制计算卸载问题。文献[26]引入了一种基于DDPG的任务卸载策略，旨在降低时延和能量使用，同时考虑用户之间的资源竞争。文献[27]设计了一种用于能量收集环境下的计算卸载的Actor-Critic算法，目标是使成功卸载的任务数量最大化。

另外，现在也有很多联合优化任务卸载决策、频谱分配及计算资源调度问题的研究方案[28-29]。其中，文献[28]将优化问题表述为混合整数非线性规划问题，同时涉及任务卸载决策、副载波和功率分配以及计算资源分配的联合优化，并提出了一个两级交替方法框架来解决所制定的问题。文献[29]构建了联合任务卸载、功率分配和资源分配的多目标优化问题，最大限度地提高用户的卸载收益，并开发了一种高效的多目标进化算法来解决响应时间最小化、能耗最小化和成本最小化等问题。

1.3　基于DAG任务的任务卸载

最近，有多项工作涉及具有DAG依赖项的任务卸载。文献[30]为支持移动多接入边缘计算（multi-access edge computing，MEC）的物联网引入了一种基于DAG的卸载策略，处理了任务间跨不同DAG的依赖关系。文献[31]研究了具有依赖关系和服务缓存约束的任务卸载挑战，开发了一种启发式策略来缩短最大完成时间和减少能源使用。文献[32]探索了在涉及单用户和多用户的场景中卸载依赖任务，使用图注意力网络和深度强化学习来减少任务的完成时间。虽然这些研究做出了有价值的贡献，但仍然面临着对意外扰动的适应性、处理高维输入数据和计算效率方面的挑战。

2　问题模型

2.1　整体卸载建模

算力网络中任务卸载系统模型如图1所示。该模型涉及一个具有子任务依赖性的实际应用程序，即一个子任务只能在其前置子任务执行完毕后才能启动。本文利用系统状态和任务配置文件来制定任务卸载策略，因此，部分任务在本地执行，而另一部分任务则通过无线信道发送到细粒度资源池进行卸载。

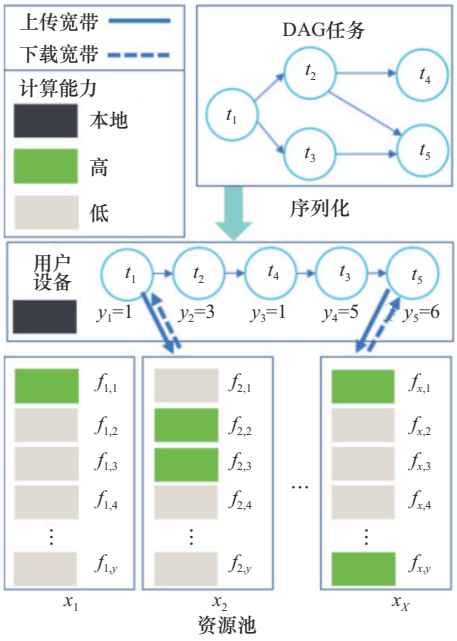


图1　算力网络中任务卸载系统模型

2.2　任务模型

参考文献[33]，本文对用户侧的任务根据其依赖性进行分析后，将其拆解为具有子任务依赖关系的DAG，用图$G=(T,\varepsilon)$表示，其中，T表示子任务的集合，t_i表示子任务，$i=1,2,\cdots,T$，T表示子任务的总数量，ε表示每一条图中边的集合，每一条边用$e(i,j)$表示，并且$e(i,j)\in\varepsilon$时表示2个任务t_i和t_j之间存在着依赖关系，任务t_j依赖t_i。为了衡量任务与资源的匹配程度，需要指定一个指标来说明任务或者资源的所属类型，

表示方式是用任务标签$Y=\{1,\cdots,y,\cdots,Y\}$指示。用$X=\{1,2,\cdots,x,\cdots,X\}$表示算力侧资源池的集合，为此，本文在考虑资源池的计算能力时，就综合考虑算力池与对应的任务的影响，不再将两者分开独立建模讨论，表示的方式为$f_x=\{f_{x,y}\}$，代表在算力池x的计算能力，集合中的每一项代表的含义是在算力池x上执行y类型任务的计算能力。建模至此可以发现，每一个算力池对于不同类型的任务均会有不同的计算能力。除此以外本文也假设任务可以在本地执行。

2.3 通信模型

将任务卸载至远程算力池的过程中，一个至关重要的环节便是节点之间高效且准确的数据传输。为了全面评估这一过程中的传输时延，需要构建一个精确的通信模型，并基于香农定理[34]来表述数据的传输速率，从而有助于降低时延。首先，数据的传输速率建模为：

$$r_{i,x}=B\cdot \mathrm{lb}\left(1+\frac{g_i\cdot p_{i,x}}{N}\right) \tag{1}$$

其中，B表示所分配的信道带宽，N表示背景噪声功率，$g_i=d^{-\alpha}$是将运行任务t_i发送到资源池x的信道增益，d表示任务t_i所在节点与资源池x之间的物理距离，α是路径损耗因子，$p_{i,x}$是将运行任务t_i发送到资源池x的信道传输功率。由此，可以根据上传信道带宽、功率等，计算t_i任务发送到资源池x的上传速率$r_{i,x}^{U}$及任务下载回本地的传输速率$r_{i,x}^{D}$。

2.4 时延模型

任务来到卸载阶段之前，将由本文的算法来决定卸载策略，对此需要建立任务的时延模型，将此过程用数学建模的方式表现如下。

首先，当任务在本地执行的时候，用$T_i^{\mathrm{UE}}=C_i/M_i f^{\mathrm{UE}}$算得本地执行的时延结果。其中，$f^{\mathrm{UE}}$表示本地的计算能力，$M_i$是$|Y|$维的单位向量用于进行计算子任务的时延，如果第$y$个元素是1，其余元素都是0，就说明这个要计算的任务时延是y类型的，其表示方式为f_y^{UE}，$y\in Y$。$f_y^{\mathrm{UE}}\in f^{\mathrm{UE}}$代表本地执行类型$y$任务的计算能力。

其次，当任务被卸载到算力池时，需要在3个阶段建模分析，分别是将任务发送到资源池、在算力资源池上执行任务、将任务执行完后返回的结果发送回本地[35]。为了得到任务在资源池的执行结果，需要具体了解任务具备的属性，本文将具体的属性代表用$F_i=(C_i,D_i^{\mathrm{S}},D_i^{\mathrm{R}},M_i)$表示。其中，$C_i$表示运行任务$t_i$所需的中央处理器（central processing unit，CPU）周期，D_i^{S}表示发送的任务的数据大小，D_i^{R}表示接收到的任务数据大小。

同时也需要表示资源池的状态，用$B_{i,x}=\{\left(r_{i,x}^{\mathrm{U}},r_{i,x}^{\mathrm{D}},f_{i,x}\right),x\in X\}$表示。其中，$r_{i,x}^{\mathrm{U}}$表示$t_i$任务发送到资源池$x$上的传输速率，$r_{i,x}^{\mathrm{D}}$表示任务下载回本地的传输速率，$f_{i,x}=\{f_{i,x,y},y\in Y\}$表示所有类型在资源池$x$上的任务计算能力，$f_{i,x,y}$表示单独分配给$y$任务的计算能力。

因此就可以计算得到将任务发送后，再经过资源池执行，然后发送回来的总等待时间。

$$T_{i,x}^{\mathrm{M}}=T_{i,x}^{\mathrm{U}}+T_{i,x}^{\mathrm{S}}+T_{i,x}^{\mathrm{D}} \tag{2}$$

其中，$T_{i,x}^{\mathrm{U}}=\dfrac{D_i^{\mathrm{S}}}{r_{i,x}^{\mathrm{U}}}$，$T_{i,x}^{\mathrm{S}}=\dfrac{C_i}{M_i f_{i,x}}$，$T_{i,x}^{\mathrm{D}}=\dfrac{D_i^{\mathrm{R}}}{r_{i,x}^{\mathrm{D}}}$，$T_{i,x}^{\mathrm{U}}$表示发送数据的时间，$T_{i,x}^{\mathrm{S}}$表示在资源池的时间，$T_{i,x}^{\mathrm{D}}$表示接收数据的时间。

2.5 调度模型

完成了对时延等待的建模，就可以获得任务的完成时间，本文把任务t_i的各类完成时间通过以下方式完成建模[36]。

$$\hat{T}_{i,x}^{\mathrm{U}}=\max\{\tilde{T}_{i,x}^{\mathrm{U}},\max_{k\in P(t_i)}\{\hat{T}_k^{\mathrm{D}},\hat{T}_k^{\mathrm{UE}}\}\}+T_{i,x}^{\mathrm{U}} \tag{3}$$

$$\hat{T}_{i,x}^{\mathrm{S}}=\max\{\tilde{T}_{i,x}^{\mathrm{S}},\max\{\hat{T}_{i,x}^{\mathrm{U}},\max_{k\in P(t_i)}\hat{T}_{i,x}^{\mathrm{S}}\}\}+T_{i,x}^{\mathrm{S}} \tag{4}$$

$$\tilde{T}_{i,x}^{\mathrm{D}}=\max\{\tilde{T}_{i,x}^{\mathrm{D}},\hat{T}_{i,x}^{\mathrm{S}}\}+T_{i,x}^{\mathrm{D}} \tag{5}$$

其中，$\hat{T}_{i,x}^{\mathrm{U}}$表示任务上传后的完成时间，$\hat{T}_{i,x}^{\mathrm{S}}$表示在资源池$x$上执行完成的时间，$\hat{T}_{i,x}^{\mathrm{D}}$表示任务完成

后发回到本地的完成时间，$\hat{T}_i^{\mathrm{UE}}$表示将任务在本地设备运行的完成时间。计算完成时间时，表达式中需要包含必要的可用时间，因此通过比较等待时间和可用时间就可以获得完成时间。可用时间的建模为$\tilde{T}_{i,x}^{\mathrm{U}}$、$\tilde{T}_{i,x}^{\mathrm{S}}$、$\tilde{T}_{i,x}^{\mathrm{D}}$和$\tilde{T}_i^{\mathrm{UE}}$，分别表示上传、资源池执行、下载和本地的可用时间，表示为：

$$\tilde{T}_{i,x}^{\mathrm{U}}=\max\{\tilde{T}_{i-1,x}^{\mathrm{U}},\hat{T}_{i-1,x}^{\mathrm{U}},x\in X\} \tag{6}$$

$$\tilde{T}_{i,x}^{\mathrm{S}}=\max\{\tilde{T}_{i-1,x}^{\mathrm{S}},\hat{T}_{i-1,x}^{\mathrm{S}},x\in X\} \tag{7}$$

$$\tilde{T}_{i,x}^{\mathrm{D}}=\max\{\tilde{T}_{i-1,x}^{\mathrm{D}},\hat{T}_{i-1,x}^{\mathrm{D}},x\in X\} \tag{8}$$

$P(\cdot)$代表映射的含义是映射原象的所有父任务组成的集合。对于在本地完成任务的执行过程里有类似的结果，$\hat{T}_i^{\mathrm{UE}}$是本地完成任务的时间，表示为：

$$\hat{T}_i^{\mathrm{UE}}=\max\{\tilde{T}_i^{\mathrm{UE}},\max_{k\in P(t_i)}\{\hat{T}_{k,x}^{\mathrm{D}},x\in X,\hat{T}_k^{\mathrm{UE}}\}\}+T_{i,x}^{\mathrm{UE}} \tag{9}$$

$$\tilde{T}_i^{\mathrm{UE}}=\max\{\tilde{T}_{i-1}^{\mathrm{UE}},\hat{T}_{i-1}^{\mathrm{UE}}\} \tag{10}$$

有了以上的建模方案后，当给定一个任务卸载的调度方案时，就可以计算出所有任务的总完成时间。

本文将一次对于整个DAG的调度计划定义为$A_{1:T}$，其中，对于单个的子任务有独立的调度动作定义为a_i，并且$a_i\in A_{1:T}$，$i=1,2,\cdots,T$。一次调度动作的取值范围是$\{0\}\cup X$，表示该任务将被指定在本地执行或者其中某个资源池上进行。当有了$A_{1:T}$时，整体DAG的完成时间表示为：

$$T_{A_{1:T}}=\max_{k=1,\cdots,T}\{\max\left\{\hat{T}_{k,x}^{\mathrm{D}},x\in X,\hat{T}_k^{\mathrm{UE}}\right\}\} \tag{11}$$

2.6　能耗模型

本节将针对任务能耗与资源池关系进行对应的建模。在算力网络中，任务在物理网络中传输不可避免地产生能量开销，本文将这部分能量表示为[37]：

$$E_{i,x}^{\mathrm{U}}=P_{i,x}^{\mathrm{U}}T_{i,x}^{\mathrm{U}} \tag{12}$$

$$E_{i,x}^{\mathrm{D}}=P_{i,x}^{\mathrm{D}}T_{i,x}^{\mathrm{D}} \tag{13}$$

其中，$P_{i,x}^{\mathrm{U}}$和$P_{i,x}^{\mathrm{D}}$分别代表任务t_i发送和下载时的网络发射功率。除了传输所消耗的能量，执行也是一大消耗能量的部分，在资源池上，本文将任务的计算能耗表示为$E_{i,x}^{\mathrm{S}}=\kappa_x^{\mathrm{M}}(M_i f_{i,x})^2C_i$，$\kappa_x^{\mathrm{M}}$是算力池的有效转换参数。所以，任务被$t_i$卸载带来的能量总耗为：

$$E_{i,x}^{\mathrm{M}}=E_{i,x}^{\mathrm{U}}+E_{i,x}^{\mathrm{S}}+E_{i,x}^{\mathrm{D}} \tag{14}$$

回到本地，本文也关注本地产生的能量消耗，因为在本地执行，将不会存在因为数据传输而产生的能耗，因此，本地的能耗就可以表示为：

$$E_i^{\mathrm{UE}}=\kappa^{\mathrm{UE}}(M_i f^{\mathrm{UE}})^2C_i \tag{15}$$

其中，κ^{UE}表示本地的有效转换参数。

计算出给定调度计划后，整个DAG中所有任务的能耗总和为：

$$E_{A_{1:T}}=\sum\nolimits_{i\in T}(1-\mathrm{sgn}(a_i))E_i^{\mathrm{UE}}+\mathrm{sgn}(a_i)E_{i,x}^{\mathrm{M}} \tag{16}$$

其中，sgn(·)是一种指示函数，表示如果元素为0，那么函数值的结果就是0，如果元素的值为正数，那么函数值就是1。

2.7　匹配度模型

本文在有了衡量任务和资源池的指标之后，就需要用一种更加精确的方式表示资源池与任务的关系，为此提出了资源池与任务匹配度的概念，并且用匹配度建模的方法完成两者关联程度的描述。本文参考了向量的分类思想，由于已经有了类型标签M_i，就用衡量资源与任务标签的向量差距，表示获得两者之间的匹配程度。本文假设存在一个L维0-1因子向量，那么可以构成的任务类型总数就可以表示为$|M_i|=2^L$。如果通过计算匹配度较高，就说明该任务在该资源池上能够获得更高的计算能力，但这并不意味着不能处理其他类型的任务，因为匹配度的高低，将表示该任务在该资源池的计算表现高低。

将任务合理地卸载到合适的资源池上，实际上也是对匹配度的合理分配。本文对向量的距离

计算方式采取欧几里得方法来进行[38]，表示为：

$$d_{i,x}=1/(1+d'_{i,x})$$

$$d'_{i,x}=\text{Euclidean}(\boldsymbol{J}_i,\boldsymbol{J}_x^M)=$$

$$\sqrt{\sum_{l=1}^{L}\{\boldsymbol{J}_i[l-1]-\boldsymbol{J}_x^M[l-1]\}^2} \quad (17)$$

其中，任务t_i的因子向量表示为$\boldsymbol{J}_i$，资源池的因子向量表示为$\boldsymbol{J}_x^{\mathrm{M}}$。本文也对本地资源赋予匹配度$d_i^{\mathrm{UE}}$，同样是经过本地的因子向量$\boldsymbol{J}^{\mathrm{UE}}$计算距离得到。

至此也可以获得整个DAG任务的总匹配度。

$$d_{A_{1:T}}=\sum_{i=1}^{T}(1-\text{sgn}(a_i))d_i^{\mathrm{UE}}+\text{sgn}(a_i)d_{i,x}^{\mathrm{M}} \quad (18)$$

3 解决方案

3.1 算力网络中卸载优化目标定义

有了对算力网络中资源侧与任务侧的综合定义和详细分析建模，总体优化目标就可以表示为：

$$(\mathrm{P})\min_{A_{1:T}}\ \alpha_1 T_{A_{1:T}}+\alpha_2 E_{A_{1:T}}-\alpha_3 d_{A_{1:T}} \quad (19)$$

本文通过调整3个参数来控制总体目标的重点是集中在匹配程度、能源消耗，还是任务时延上。该问题属于NP难问题，因此找到最优的卸载调度方案是一项非常困难的任务。

3.2 卸载过程的马尔可夫决策建模

为了解决这个优化问题，本文对其中一个任务t_i的卸载决策过程进行建模，要定义马尔可夫决策，需要指定状态、动作和奖励。为了方便定义，本文指定一个学习任务$\mathcal{L}_i$，$\mathcal{L}_i$服从$\rho(\mathcal{L})$分布。

状态$S:=\left\{s_i|s_i=\left(G(T,\varepsilon),A_{1:i}\right)\right\}$

动作$A:=\{0,1,2,\cdots,X\}$

奖励 $R:=\alpha_1 T_{A_{1:i}}+\alpha_2 E_{A_{1:i}}-\alpha_3 d_{A_{1:i}}-(\alpha_1 T_{A_{1:i-1}}+\alpha_2 E_{A_{1:i-1}}-\alpha_3 d_{A_{1:i-1}})$

其中，状态由算力网络的环境和任务的执行顺序决定，为此需要掌握整个DAG的拓扑结构及任务和算力资源池的详细属性。

对动作来说，此步骤的决策动作与调度决策动作相同。目标奖励则是对最终优化目标函数的预估负增量。

具备了马尔可夫决策过程的建模之后，调度任务的预估概率就可以获得，本文将其表示为：

$$\pi(\alpha_i|G(T,\varepsilon),A_{1:i-1}) \quad (20)$$

对整个DAG中的所有任务，应用概率链规则就可以得到整个DAG中所有任务的卸载策略概率。

$$\pi(A_{1:n}|G(T,\varepsilon))=\prod_{i=1}^{n}\pi(\alpha_i|G(T,\varepsilon),A_{1:i-1}) \quad (21)$$

式（21）表示给定DAG后采取了卸载计划$A_{1:n}$的概率。

3.3 元强化学习解决算力网络卸载策略问题

本文采用了Seq2Seq网络模型来进行深度学习过程，为了提高策略的效果，对网络结构进行了改造。在编码器的输入和输出位置各加入了一层注意力层来提高对任务的关注程度，进而发现算力网络整体环境中的隐藏细节。宽注意力机制Seq2Seq神经网络如图2所示。

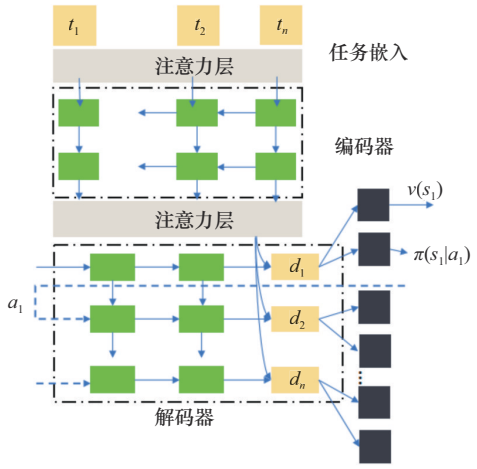


图2 宽注意力机制Seq2Seq神经网络

深度强化学习的部分采用了元强化学习算法，因为元强化学习算法能更好地适应算力网络变化。本文的训练目标为：

$$J^{\mathrm{MRL}}(\boldsymbol{\theta})=\boldsymbol{E}_{\mathcal{L}_i\sim\rho(\mathcal{L}),\tau\sim P_{\mathcal{L}_i}(\tau,\boldsymbol{\theta}'_i))}[J_{\mathcal{L}_i}^{\mathrm{PPO}}(\boldsymbol{\theta}'_i)] \quad (22)$$

其中，$J_{\mathcal{L}_i}^{\mathrm{PPO}}(\boldsymbol{\theta}_i)$是骨干算法的目标函数，本文的

骨干强化学习算法选择的是近端策略优化。$\boldsymbol{\theta}_i'=U_{\tau\sim P_{\mathcal{L}_i}(\tau,\boldsymbol{\theta}_i)}(\boldsymbol{\theta}_i,\mathcal{L}_i),\boldsymbol{\theta}_i=\boldsymbol{\theta}$，$\boldsymbol{\theta}$ 表示网络参数向量，采用梯度上升来最大化训练目标。

本文从批量大小为 u 的一批学习任务 $\mathcal{L}$ 中采样出一些学习任务，并对其进行主干强化学习训练。主干强化学习训练完成后，本文用梯度上升更新 $\boldsymbol{\theta}$，采用的是 Adam 方法[39]。

本文算法关于时间复杂性的理论分析如下。考虑最坏情况的 DAG 分布，计算 $T_{A_{1:T}}$ 的最坏时间复杂度为 $O(\mathrm{TX})$，计算 $E_{A_{1:T}}$ 的时间复杂度为 $O(\max\{\mathrm{TX},\mathrm{TXY},\mathrm{TX}\})=O(\mathrm{TXY})$，计算匹配度 $d_{A_{1:T}}$ 的时间复杂度为 $O(\mathrm{TXL})$。引入强化学习的马尔可夫决策过程的时间复杂度为 $O((X+1)^T\cdot Y^T\cdot(X+1)+K\cdot T_s\cdot S_{\mathrm{eq}})$，其中，$K$ 为迭代次数，S_{eq} 为神经网络的复杂性。元强化学习不影响总的时间复杂度，且上述涉及时间复杂度的计算均可顺序执行。据此，总的时间复杂度为 $O((X+1)^T\cdot Y^T\cdot(X+1)+K\cdot T_s\cdot S_{\mathrm{eq}})$。

4　实验和分析

4.1　实验环境仿真参数设置

本文设置本地的计算力为 109 Hz，设置了 4 种类型的标签，并且为不同匹配潜力设置了不同的计算能力。本文假设每个资源池对待相同匹配程度的任务有相同的计算能力，因此，为高匹配任务提供 400 GHz 能力，为低匹配任务提供 10 GHz 能力。本文的数据大小范围为 6～600 KB。另外，假设了 CPU 计算周期数为 100。

4.2　实验结果和数据分析

本文通过实验对比目前的主流算法有：全远端卸载[40]、全本地卸载[41]、贪心策略卸载[42]和近端策略优化卸载[43]、DDQN[22]和 DDPG[23]。

DAG 任务数量为 20 时算法的表现如图 3 所示。

从实验结果可以看出，本文的算法整体上可以明显地降低 DAG 任务的总时延表现，并且

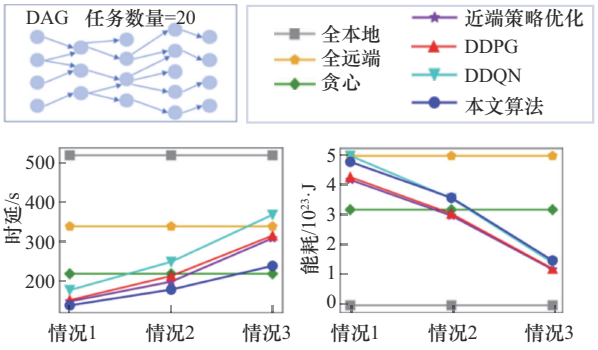


图 3　DAG 任务数量为 20 时算法的表现

维持一定的能耗降低能力。具体而言，本文关注 3 种情况。第一种，在情况 1 中设置参数 $\alpha_1=0.8$、$\alpha_2=0.1$、$\alpha_3=0.1$，可以看到在情况 1 中，本文的算法能够有效地降低时延表现，对比全卸载远端的方案，能够有效地降低时延 72.6%；第二种，在情况 2 中设置的参数配比为 0.4、0.4、0.1，本文的算法与全卸载远端的方案比，能耗降低 34.6%，时延也降低了 52.6%；第三种，情况 3 中设置的参数配比为 0.3、0.7、0.1，本文的算法比全部卸载远端的方案能耗降低了 72.9%，而且时延也降低了 25.6%。而与 DDQN 和 DDPG 相比，本文算法的能耗优势相当，但是拥有更低的时延，在情况 3 中时延分别降低了 35.0% 和 24.2%。这进一步体现了本文算法降低 DAG 任务的总时延表现的显著能力，同时能够兼顾降低能耗。

经过分析发现，产生上述实验结果的主要原因是本文每次对优化目标的重视程度不同，在情况 1 中，本文更加关注时延的表现，所以反馈到实验结果中就是本文的算法在降低时延上表现最好；在情况 2 中，本文平衡了对时延和能耗的重视程度，得到了能耗和时延表现均有所优势的卸载策略；在情况 3 中，本文提高了对于算力网络能耗的关注程度，此时本文的卸载策略将大幅降低整体的算力网络能源消耗。

在卸载结果中，能耗显著降低时，算力网络的能源消耗大幅减少，但这也可能导致任务时延

的增加，此结果符合实验预期。因为本文任务大量的计算能耗产生在算力池一侧，卸载到算力池虽然能够加速任务执行速度，降低时延，但也极大增加了能源的消耗，因此，两者此消彼长的表现是符合事实的。可以看到，即使本文的算法关注有所侧重时，依然能保证另一个指标处于较好的表现范围内，而不是完全放弃参考另一指标，只全力提升一个指标效果。

本文在当前的任务数据集上有了较好的表现后，需要进一步推广到更丰富的数据拓扑结构中，验证本文方法的有效性，因此，在重新调整了DAG任务的拓扑结构后，DAG任务数量为30时算法表现如图4所示。

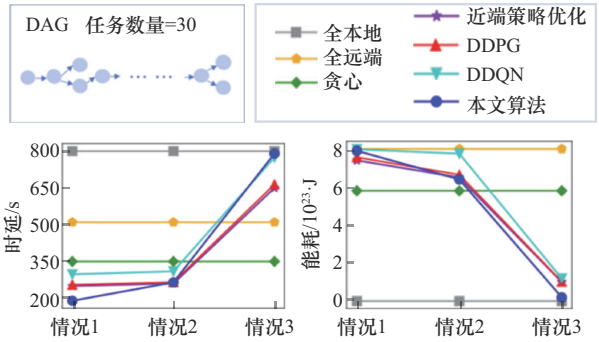
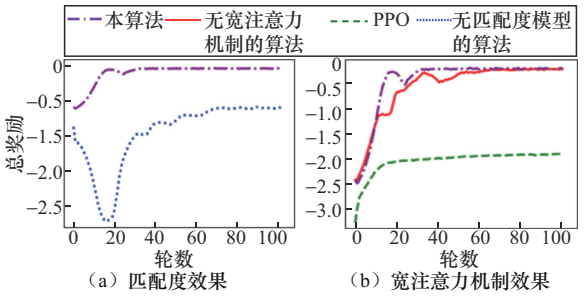


图4 DAG任务数量为30时算法表现

在调整的数据集中，本文修改了DAG拓扑结构，增加了节点的数量，使得网络结构细长。但是可以从图4中看出，尽管具体的实验结果上有数据差异，但是整体趋势上，3种情况的实验结果得到了与图3实验相同的结论，证明了本文的算法设计能够满足结构复杂多样的DAG算力任务。具体而言，相比于图3，本文的算法对比DDQN与DDPG，在具有相当时延优势的情况下，取得了更优的能耗表现。对于情况3，对比DDQN与DDPG分别降低了83.2%和80.4%。

为了说明本文设计的作用，进行了如下的实验，不同算法的学习效果比较如图5所示。

本文设置了2种实验，首先，对比有无匹配



（a）匹配度效果 （b）宽注意力机制效果

图5 不同算法的学习效果比较

度模型对本算法的影响，在删除了匹配度模型之后，从图5（a）中可以看到，本算法得到了更高的总奖励，具有更好的卸载表现，说明了本文采用的匹配机制的有效性。

另一种是去除了宽注意力机制的实验，从图5（b）中可以看到，没有宽注意力机制会使得学习的收敛速度变慢，本文的算法能够更快速地得到卸载决策收敛结果。这说明宽注意力机制能够更好地利用算力网络整体的信息，加速了学习过程。同时实验还比较了近端策略优化方法，可以看到不采用元强化学习的深度强化学习算法，在学习总奖励的表现上不如本文的算法，本文的算法能够很好地适应变化的算力网络。

5 结束语

本文针对高速发展的算力网络带来的能耗问题，提出了一种基于智能匹配机制的卸载策略，首先，分析了任务依赖和算力的综合联系，建立了合理的匹配模型，提出了宽注意力机制的Seq2Seq网络和元强化学习任务卸载方法，完成了对算力网络中任务的卸载决策，关注任务间的依赖关系和算力网络资源与任务的复杂信息，同时，能够为变化的算力网络提供可以调整关注重点角度的动态卸载方案，提高了卸载决策的效率，降低了算力网络时延和能量消耗。

参考文献：

[1] 郭琨, 康雨馨, 卓训方. 京津冀国家算力枢纽节点赋能全球数字经济标杆城市建设[J]. 大数据, 2023, 9(5): 134-139.
GUO K, KANG Y X, ZHUO X F. Beijing-Tianjin-Hebei national integrated big-data center system empowers global digital economy benchmark city construction[J]. Big Data Research, 2023, 9(5): 134-139.

[2] 段晓东, 姚惠娟, 付月霞, 等. 面向算网一体化演进的算力网络技术[J]. 电信科学, 2021, 37(10): 76-85.
DUAN X D, YAO H J, FU Y X, et al. Computing force network technologies for computing and network integration evolution[J]. Telecommunications Science, 2021, 37(10): 76-85.

[3] 罗军舟, 金嘉晖, 宋爱波, 等. 云计算: 体系架构与关键技术[J]. 通信学报, 2011, 32(7): 3-21.
LUO J Z, JIN J H, SONG A B, et al. Cloud computing: architecture and key technologies[J]. Journal on Communications, 2011, 32(7): 3-21.

[4] RAZA K, PATLE V K, ARYA S. A review on green computing for eco-friendly and sustainable IT[J]. Journal of Computational Intelligence and Electronic Systems, 2012, 1(1): 3-16.

[5] 林俊宇, 王慧强, 马春光, 等. 一种基于DAG动态重构的认知网络服务迁移方法[J]. 软件学报, 2014, 25(10): 2373-2384.
LIN J Y, WANG H Q, MA C G, et al. Service migration method for cognitive network based on DAG dynamic reconstruction[J]. Journal of Software, 2014, 25(10): 2373-2384.

[6] FENG C, HAN P C, ZHANG X, et al. Computation offloading in mobile edge computing networks: a survey[J]. Journal of Network and Computer Applications, 2022, 202: 103366.

[7] ZHANG J, GUO H Z, LIU J J, et al. Task offloading in vehicular edge computing networks: a load-balancing solution[J]. IEEE Transactions on Vehicular Technology, 2020, 69(2): 2092-2104.

[8] 崔勇, 宋健, 缪葱葱, 等. 移动云计算研究进展与趋势[J]. 计算机学报, 2017, 40(2): 273-295.
CUI Y, SONG J, MIAO C C, et al. Mobile cloud computing research progress and trends[J]. Chinese Journal of Computers, 2017, 40(2): 273-295.

[9] GUO H Z, LIU J J, REN J, et al. Intelligent task offloading in vehicular edge computing networks[J]. IEEE Wireless Communications, 2020, 27(4): 126-132.

[10] WANG Y P, LANG P, TIAN D X, et al. A game-based computation offloading method in vehicular multiaccess edge computing networks[J]. IEEE Internet of Things Journal, 2020, 7(6): 4987-4996.

[11] NING Z L, DONG P R, KONG X J, et al. A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things[J]. IEEE Internet of Things Journal, 2019, 6(3): 4804-4814.

[12] LEI L, XU H J, XIONG X, et al. Joint computation offloading and multiuser scheduling using approximate dynamic programming in NB-IoT edge computing system[J]. IEEE Internet of Things Journal, 2019, 6(3): 5345-5362.

[13] HAN Y P, ZHAO Z W, MO J W, et al. Efficient task offloading with dependency guarantees in ultra-dense edge networks[C]// Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM). Piscataway: IEEE Press, 2019: 1-6.

[14] ZHANG J, HU X P, NING Z L, et al. Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks[J]. IEEE Internet of Things Journal, 2018, 5(4): 2633-2645.

[15] YUAN X W, XIE Z D, TAN X. Computation offloading in UAV-enabled edge computing: a Stackelberg game approach [J]. Sensors, 2022, 22(10): 3854.

[16] ISMAIL A H, EL-BAHNASAWY N A, HAMED H F A. AGCM: active queue management-based green cloud model for mobile edge computing[J]. Wireless Personal Communications, 2019, 105(3): 765-785.

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

[18] HUANG L, BI S Z, ZHANG Y J A. Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks[J]. IEEE Transactions on Mobile Computing, 2020, 19(11): 2581-2593.

[19] ZHENG C, LIU S H, HUANG Y M, et al. Hybrid policy learning for energy-latency tradeoff in MEC-assisted VR video service[J]. IEEE Transactions on Vehicular Technology, 2021, 70 (9): 9006-9021.

[20] LIU F Z, HUANG J W, WANG X B. Joint task offloading and resource allocation for device-edge-cloud collaboration with subtask dependencies[J]. IEEE Transactions on Cloud Computing, 2023, 11(3): 3027-3039.

[21] MESKAR E, TODD T D, ZHAO D M, et al. Energy aware offloading for competing users on a shared communication channel[J]. IEEE Transactions on Mobile Computing, 2017, 16(1): 87-96.

[22] ZHOU H, JIANG K, LIU X X, et al. Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing[J]. IEEE Internet of Things Journal, 2022, 9(2): 1517-1530.

[23] CHEN J, XING H L, XIAO Z W, 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.

[24] CHEN Y, ZHANG N, ZHANG Y C, et al. Energy efficient dynamic offloading in mobile edge computing for Internet of Things[J]. IEEE Transactions on Cloud Computing, 2021, 9(3): 1050-1060.

[25] NING Z L, DONG P R, KONG X J, et al. A cooperative partial computation offloading scheme for mobile edge computing en-

abled Internet of Things[J]. IEEE Internet of Things Journal, 2019, 6(3): 4804-4814.

[26] QU B, BAI Y, CHU Y, et al. Resource allocation for MEC system with multi-users resource competition based on deep reinforcement learning approach[J]. Computer Networks, 2022, 215: 109181.

[27] ZHANG Z C, YU F R, FU F, et al. Joint offloading and resource allocation in mobile edge computing systems: an actor-critic approach[C]//Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM). Piscataway: IEEE Press, 2018: 1-6.

[28] TAN L, KUANG Z F, ZHAO L, et al. Energy-efficient joint task offloading and resource allocation in OFDMA-based collaborative edge computing[J]. IEEE Transactions on Wireless Communications, 2022, 21(3): 1960-1972.

[29] WANG P, LI K L, XIAO B, et al. Multiobjective optimization for joint task offloading, power assignment, and resource allocation in mobile edge computing[J]. IEEE Internet of Things Journal, 2022, 9(14): 11737-11748.

[30] ZHOU X B, GE S X, LIU P B, et al. DAG-based dependent tasks offloading in MEC-enabled IoT with soft cooperation[J]. IEEE Transactions on Mobile Computing, 2024, 23(6): 6908-6920.

[31] LV X Y, DU H W, YE Q. TBTOA: a DAG-based task offloading scheme for mobile edge computing[C]//Proceedings of the ICC 2022 - IEEE International Conference on Communications. Piscataway: IEEE Press, 2022: 4607-4612.

[32] CAO Z Q, DENG X H, YUE S, et al. Dependent task offloading in edge computing using GNN and deep reinforcement learning[J]. IEEE Internet of Things Journal, 2024, 11(12): 21632-21646.

[33] FU X, TANG B, GUO F Y, et al. Priority and dependency-based DAG tasks offloading in fog/edge collaborative environment[C]//Proceedings of the 2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD). Piscataway: IEEE Press, 2021: 440-445.

[34] 刘昌锦, 韦哲. 香农公式在扩频通信中的应用[J]. 四川兵工学报, 2013, 34(4): 80-83.
LIU C J, WEI Z. Research on application of Shannon formula in spread spectrum communication[J]. Journal of Sichuan Ordnance, 2013, 34(4): 80-83.

[35] QIN M, CHENG N, JING Z W, et al. Service-oriented energy-latency tradeoff for IoT task partial offloading in MEC-enhanced multi-RAT networks[J]. IEEE Internet of Things Journal, 2021, 8(3): 1896-1907.

[36] XU Y G, LIU L, DING Z J. DAG-aware joint task scheduling and cache management in spark clusters[C]//Proceedings of the 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS). Piscataway: IEEE Press, 2020: 378-387.

[37] HEIDARI A, ALI JABRAEIL JAMALI M, JAFARI NAVIMIPOUR N, et al. Deep Q-learning technique for offloading offline/online computation in blockchain-enabled green IoT-edge scenarios[J]. Applied Sciences, 2022, 12(16): 8232.

[38] WANG T, LIANG Y Z, YANG Y, et al. An intelligent edge-computing-based method to counter coupling problems in cyber-physical systems[J]. IEEE Network, 2020, 34(3): 16-22.

[39] KINGA D, ADAM J B. A method for stochastic optimization[C]// International conference on learning representations (ICLR). Piscataway: IEEE Press, 2015: 5-6.

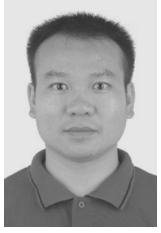
[40] MAO B M, TANG F X, KAWAMOTO Y, et al. Optimizing computation offloading in satellite-UAV-served 6G IoT: a deep learning approach[J]. IEEE Network, 2021, 35(4): 102-108.

[41] ZHAN W H, LUO C B, MIN G Y, et al. Mobility-aware multi-user offloading optimization for mobile edge computing[J]. IEEE Transactions on Vehicular Technology, 2020, 69(3): 3341-3356.

[42] ZHOU W, LIN C W, DUAN J R, et al. An optimized greedy-based task offloading method for mobile edge computing[M] Lecture Notes in Computer Science. Cham: Springer International Publishing, 2022: 494-508.

[43] CHEN G, XU X J, ZENG Q T, et al. A vehicle-assisted computation offloading algorithm based on proximal policy optimization in vehicle edge networks[J]. Mobile Networks and Applications, 2023, 28(6): 2041-2055.

[作者简介]



兰世战（1981– ），男，中国移动通信集团广西有限公司正高级工程师，主要研究方向为算力网络、移动边缘计算CDN和AI等。

马丽芳（1980– ），女，广西建设职业技术学院信息工程学院副教授，主要研究方向为计算机应用、云计算和人工智能。