基于有向图与卷积网络强化学习的端侧协同算力资源分配方法

顾健华[1,2],冯建华[1],许辉阳[2*],刘佟佟[2],周 婷[2]
(1. 清华大学计算机科学与技术系,北京 100084;2. 中国移动通信集团终端有限公司,北京 100053)

摘 要: 随着人工智能应用场景的集中式爆发,移动应用对数据通信和计算的需求日益增长,位于远端的传统云计算处理方法难以满足快速响应的要求. 因此,整合利用海量的用户侧终端设备算力(包括计算、存储、通信等)的端侧算力网络,通过分布式协作合理地利用终端算力完成计算任务成为一种新的处理方法. 鉴于单台终端设备的资源受限,高企的通信开销限制任务协同效果,导致终端很难高效协同完成高度复杂的计算任务. 本文提出利用点对点(Device-to-Device,D2D)通信辅助终端节点协同计算,并设计了基于有向图卷积网络(Directed Graph Convolutional Network,DGCN)的协作拓扑和资源分配决策算法(Multi-Agent Soft Actor-Critic,MA-SAC),将有向无环图(Directed Acyclic Graph,DAG)任务中包含的子任务部署到多个终端进行协同计算,满足DAG子任务部署在多个不同节点间的跨节点传输需求,降低子任务间数据传输在基站侧的网络通信开销. 仿真结果显示,所提算法能够在保证业务时延要求下,降低38.2%的网络通信开销,有效提升31.9%的端侧资源利用率.

关键词: 端侧算力;终端协同;多跳D2D;端算力分配;有向图卷积网络
中图分类号: TN915.07 **文献标识码:** A **文章编号:** 0372-2112(2025)06-1771-13
电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20251106
第二十七届中国科协年会学术论文

Directed Graph and Convolutional Network Reinforcement Learning for Terminal-Side Collaborative Computing Resource Allocation Scheme

GU Jian-hua[1,2], FENG Jian-hua[1], XU Hui-yang[2*], LIU Tong-tong[2], ZHOU Ting[2]
(1. *Department of Computer Science and Technology*, *Tsinghua University*, *Beijing* 100084, *China*;
2. *China Mobile Group Device Co.*, *Ltd.*, *Beijing* 100053, *China*)

Abstract: Driven by the concentrated surge of AI application scenarios, the increasing requirements on data communication and computation in mobile applications is growing, the traditional cloud computing which relies on remote processing, often fails to meet low-latency requirements. Therefore, a new paradigm has emerged: terminal-side computing power that aggregate the vast terminal devices (including computing, storage, communication, etc) through distributed collaboration to efficiently execute computational tasks. However, constrained by the limited resource of standalone device and prohibitive communication overhead that impairs task coordination, such terminals still face significant challenges in achieving efficient collaboration for highly complex computing tasks.This paper presents device-to-device (D2D) communication assisted terminal devices collaborative computing, and a multi-agent soft actor-critic (MA-SAC) based on directed graph convolutional network (DGCN) is designed to solve this problem.The subtasks included in directed acyclic graph (DAG) tasks were deployed to multiple terminals for collaborative computing, it is introduced to cater to the exigencies of task transmission between disparate nodes within the DAG, and reduces the communication overhead when data transmission in the network. Through the simulations, the efficacy of the proposed scheme is demonstrated. The proposed scheme reduces network communication overhead by 38.2% and effectively improve resource utilization by 31.9%.

Key words: terminal-side computing power; terminal collaboration; multi hop D2D; terminal-side computing power allocation; directed graph convolutional network

收稿日期:2024-12-09;修回日期:2025-05-30;责任编辑:宋甲英
*通讯作者:许辉阳

1 引言

近年来,基于大模型的人工智能技术突飞猛进,ChatGPT、Sora等人工智能产品和应用不断涌现,对人们生产生活方式将产生深刻影响.然而,大模型的训练和推理需要巨量的算力资源,传统的云计算模式对巨大的算力和网络资源的需求力不从心,无法及时响应AI的实时性要求.随着5G网络的快速发展和广泛普及,算力从云端下沉到边缘,形成了端侧算力网络的概念(Terminal-Side Computing Force Network, TSCFN)[1,2],端侧算力网络将云的算力卸载到终端侧,并充分整合端侧的闲散算力,有效地降低了云的算力资源和网络资源的压力,克服了云计算延迟高、隐私性差、安全性低以及能耗高等局限性[3,4].端侧算力网络与AI的结合成为研究热点,但是由于资源有限,端设备很难完成高度复杂的模型推理任务.因此,通过分布式协作合理地完成终端算力资源的分配成为研究重点,其中的核心点就是如何根据任务需要高效地调度端侧算力.

针对算力的调度问题,业界已有较多的研究基础.主要思路是通过不同的优化算法来降低调度时延、降低成本和提高服务QoS水平、提高系统性能.Zeng等人[5]通过遗传甲虫调度算法进行算力调度,但是这种启发式类遗传算法属于离线优化方法,求解过程依赖于预先设计的种群进化,涉及多个策略且参数需要精心调校,容易因参数设置不当而陷入局部最优或导致搜索效率降低,当调度问题规模较大时,遗传算法的种群进化和适应度评价可能需要较多迭代,从而导致计算时间较长.此外,Zhu等人[6]提出了任务调度算法,Hu等人[7]提出了异构集群中的容器部署算法,但这些调度机制是基于标准、统一、稳定的硬件设备[8~10],对于异构硬件的设备难以统一调度.随着AI的进一步发展,也逐渐开始有研究利用终端设备来进行分布式推理、端边云协同推理,但仅限于特定的业务场景以及单一或者数量有限的终端设备[11,12].

端侧算力分布式协作的基础是有向的任务能够在端与端之间进行迁移.子任务在不同端节点迁移过程中具有先后服务顺序,形成有向无环图(Directed Acyclic Graphs, DAG).设备到设备(Device-to-Device, D2D)通信技术可以实现端到端的数据卸载,优化工作任务流的性能.基于任务迁移模型将任务迁移调度问题构造为有向无环图中的约束随机最短路径问题,可以有效降低端侧负荷,解决高网络消耗的问题[13~15].3GPP R16和R17中将D2D技术作为典型移动计算终端直接通信技术,如车联网中车车间通信[16,17].在D2D通信中,数据在源和目的端之间直接传输,不需要基站的转发,因此在端侧算力分布式协作中采用D2D进行传输能够降低空口资源占用以及由于基站中转带来的通信开销.此外,端侧算力协同并不限于两两协同,更多情况下需要引入多跳D2D技术.相比单跳D2D,多跳D2D可以支持更远距离、更复杂网络结构、更高效率的通信[18~21].通过D2D可以实现任务迁移,更加充分地利用终端的算力,然而端侧算力网络中节点复杂多样,端侧算力动态协同调度尚待研究.

同时,考虑到端侧算力不稳定性等特点,端侧资源分配是一个非线性问题.深度学习能够处理复杂的非线性问题,在图像识别、计算机视觉等领域得到了广泛应用[22,23].然而,深度学习在处理特殊的结构化数据时存在局限性.卷积神经网络(Convolutional Neural Networks, CNN)在处理网格化数据(如图像)方面表现出色,但对于处理具有稀疏性的图数据,效果受限[24].图神经网络(Graph Neural Network, GNN)能够直接在图结构数据上工作,有效处理节点和边之间的关系,但在面对需要决策和连续学习的问题时,单纯的GNN又不足以应对[25].有向图与卷积网络强化学习结合了GNN对图结构数据的处理能力和强化学习在决策方面的优势,使模型能够在理解复杂网络结构的同时,对策略开展有效的学习和进行决策[26,27].

本文提出一种全新的端侧协同算力资源动态调度分配的算法,研究内容及创新点如下.首先,将一个任务分解为由多个子任务组成的有向无环图,并引入多跳D2D机制,满足DAG子任务部署在不同节点间的跨节点传输需求.然后,以端侧资源利用率最高、任务执行时延最小为目标,设计了联合优化问题模型.最后,设计了基于有向图卷积网络(Directed Graph Convolutional Network, DGCN)的协作拓扑和资源分配决策算法(Multi-Agent Soft Actor-Critic, MA-SAC),实现了端侧算力的分布式协作,并合理地完成终端算力资源的动态调度和分配.本文所提的方法基于MA-SAC深度强化学习框架,作为一种深度强化学习算法,可以在多智能体框架中通过不断与周围环境交互学习的过程中,形成较强的在线适应性;通过熵正则化等机制鼓励探索,降低了对初始参数的敏感性,从而在复杂状态空间中更易找到全局最优解.MA-SAC是为多智能体环境设计的,能够更好地处理多主体之间的竞争与协作问题.当调度问题规模较大时,MA-SAC借助深度神经网络对策略和价值函数进行逼近,并支持多路并行计算,在处理大规模、复杂任务时具有更高的计算效率和扩展性.

2 系统模型

2.1 场景描述

本文关注的D2D通信辅助的多终端计算任务协同场景如图1所示,部分主要记号如表1所示.系统中基站集合为$\mathcal{U}$,接入基站$u\in\mathcal{U}$下的移动终端的集合为

$\mathcal{M}_u$. 终端可以分为两类:卸载请求终端和协同计算终端. 卸载请求终端有需要卸载的工作流应用程序,协同计算终端拥有空闲的计算资源来提供计算服务,本文认为卸载请求终端是计算任务的发起者,具有一定计算资源,来承担一部分初始计算任务. 因此,本文将两类终端都称为资源节点(Resource Node,RN).

表 1　记号表(部分主要记号说明)

符号	说明
$\mathcal{K},K$	服务的前后继子任务集合与集合中子任务数量
$\mathcal{O}$	系统中服务集合
$G(V,E)$	DAG任务图,V表示图的顶点,E表示图的边
$\boldsymbol{w}=\{w_i\|i\in V\}$	子任务的计算资源要求
f_n	RNn总资源量
$f_{n,i}$	RNn分配给子任务i的资源量
t_i	子任务i的计算时长
$r_{o,ij}$	子任务i和j之间的传输带宽要求
$t_{\text{start},i}$	子任务i的开始时间
$t_{\text{end},i}$	子任务i的结束时间
$P(i)$	子任务i的前继子任务构成的集合
$t_{ij,mn}$	子任务i和子任务j分别在RNm和RNn上执行
$e_{d,ij}$	从子任务i传输到子任务j的数据量大小
χ	网络波动,零均值高斯随机变量
η_M	端侧资源利用率
T_{dl}	任务执行时延
$T_{d,\mathcal{K}}$	任务的总执行时间开销
$T_{d,\max}$	系统中现有全局任务最大完成时间
$\mathcal{U},U$	基站集合与数量
$\mathcal{M}_u,M_u$	基站$u\in U$下移动终端集合与数量

本文关注的计算任务是由一系列功能组件组成的应用程序或服务,组件之间具有前后继关系,形成有向无环图结构,本文称为DAG任务. 在微服务和云原生技术的强力驱动下,这类应用的比重已形成明显的上升趋势. D2D通信允许移动终端卸载计算任务到其他协作终端,并且考虑到DAG任务具有前后继的数据依赖和逻辑顺序,因此备选终端可以选择其邻近的另一个终端作为下一跳协作终端,此过程可连续迭代,因此产生了基于多跳D2D的多终端计算任务协同. 就整体而言,卸载请求者是把DAG任务分配给邻近范围内一系列可提供计算资源的协作计算终端,由多个终端协同处理DAG任务.

蜂窝D2D中基站一般作为管理者,监测终端状态并进行无线资源协调,保障D2D通信与传统蜂窝通信的服务质量. 在多终端计算任务协同场景中,为了实现基于多跳D2D的多终端计算任务协同,每个终端可以感知邻近RN(单跳D2D节点)的信号强度,将感知到的服务节点信息提交给网络侧的管理节点(Management Node,MN),因此MN可以通过基站感知可参与协作的RN的网络状态信息. 为了获取各RN的资源状态信息,本文假定RN在上报网络状态信息的同时上报算力资源状态信息. 由于管理节点收到一定区域内RN上报的单跳范围内的RN信息,管理节点进而可以综合出多跳D2D的潜在通信拓扑. 由于该采集过程持续进行,管理节点能及时更新RN间潜在拓扑,同时优选符合一定条件的RN作为备选协作计算节点,从而提升卸载任务的可靠性.

基于上述设计,MN对维护协作RN间的协作关系和D2D拓扑非常重要. 由于蜂窝网中存在大量移动终端,管理节点如果采用集中方式部署,意味着需要收集大量终端信息,会占用回程链路资源,同时影响状态信息采集的时效性. 因此,本文提出的D2D辅助多终端计算任务协同框架下,管理功能实体位于基站的边缘云上,就近处理RN上报的资源信息,并且基站的边缘云资源也可以作为可用算力资源. 同时每个管理节点基于深度强化学习智能体构建而成,利用采集的信息和邻近管理节点提供的管理信息,完成待卸载DAG任务在多个RN的部署与资源分配决策.

具体而言,当有协作任务请求时,卸载请求者向网络管理节点提交包含了任务DAG工作流、完成时间以及预算限制等信息. 管理节点在收到卸载请求者发来的DAG任务卸载请求时,根据任务请求者提供的任务信息,并综合邻近区域一定范围内的RN资源信息以及RN间潜在拓扑,来决策DAG任务如何分配到合适的协作节点,D2D通信拓扑如何构建以及资源如何分配. 本文假设管理节点分布式部署于基站. 此外,当子任务无法被RN及时处理时,会移交给基站的边缘云计算资源处理.

2.2　DAG任务模型

如前文所述,本文关注的卸载计算任务为DAG计算任务,这类计算任务由多个可以分布式部署运行的组件构成,如图1右上角所示5个功能组件P1~P5组成的应用程序,这些功能组件按照一定的前后继顺序执行,并允许在不同节点上执行,这已成为应用体系结构发展的主要方向. 服务被假设为由具有前后继数据依赖的K个子任务组成. 基于此,我们通过一个带权DAG来模拟多任务服务,记为$G=\{V,E\}$,$V=\{1,2,\cdots,K\}$中的顶点代表子任务,而$E=\{(i,j)\|i,j\in V\}$中的边代表子任务之间的依赖关系. 此外,依赖性表明子任务只有在其前驱任务完成并接收到结果后才能执行. 子任务有计算量属性,用$w=\{w_i\|i\in V\}$表示子任务所需的CPU(Central Processing Unit)周期数. 记子任务k在RNn上执行,表示为$x_{k,n}=1$,否则为0. 对于网络中的RNn,总

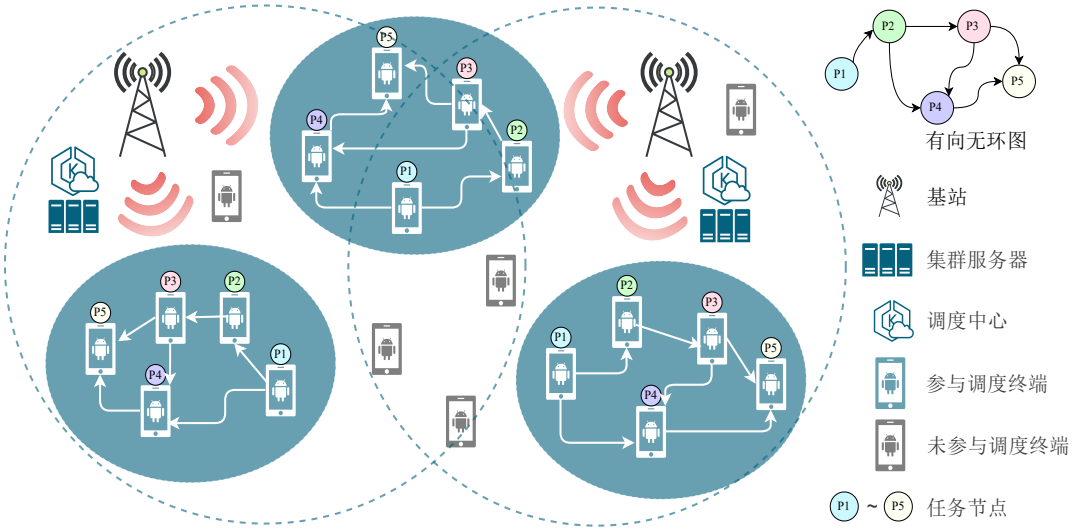


图1 基于多跳D2D的多终端计算任务协同场景示意图

资源量为$f_n\left(\text{cycle/s}\right)$,分配给子任务的固定计算能力定义为$f_{n,k}\left(\text{cycle/s}\right),k\in K$,则计算时长为

$$t_i^k=\frac{w_i}{f_{n,k}}\quad i\in V,k\in K \tag{1}$$

除了计算资源需求,节点之间的边代表任务的前后继关系,有通信量属性,表示子任务间保持顺利协同工作,需要保证节点间的传输带宽满足子任务间的传输要求$r_{i,j},\left(i,j\right)\in E$.

对于DAG任务,将子任务i的开始时间和结束时间定义为$t_{\text{start},i}$和$t_{\text{end},i}$. 分配决策向量定义为$X_k=\left\{X_{i,n}\right\}_{i\in k,n\in M_u}$其中$x_{i,n}=1$代表将子任务$i$部署到RN$n$上,否则$x_{i,n}=0$;对于全局任务$o$,部署决策变量$X=\left\{X_k\right\}_{k\in o}$. 本文中认为同一应用的子任务应该部署在不同RN,因此$\sum\limits_{i\in K}x_{i,n}=1$.

2.3 D2D协同计算模型

假设两个RN可以基于蜂窝D2D相互通信,我们分别定义RNm和RNn之间的数据传输速率和延迟(包括传播延迟、排队延迟和处理延迟)为$B_{m,n}$和$P_{m,n}$. 集合$e_{ij}\left(i,j\in E\right)$代表从子任务$i$传输到子任务$j$的数据量大小. 当子任务$i$和子任务$j$分别在RN$m$和RN$n$上执行时,从子任务$i$到子任务$j$的结果传输时间可以表示如式(2). 使用Anamuro等人[28]的D2D通信模型,该模型将修正香农容量公式与对数正态阴影传播模型相结合,如式(3). 其中,B_{eff}为带宽效率;B_w为设备通信带宽;P_m为设备通信传输功率;K_d为路径距离的损耗因子(以m为单位);N_o为噪声功率谱密度;θ_{eff}为信噪比效率;α_d为路径损耗指数,当前距离d由终端的轨迹计算得出;网络波动χ是零均值高斯随机变量. 由于DAG任务中的子任务有前后继关系,$P(i)$表示子任务i的所有前继子任务,只有当前继任务$j\in P(i)$都完成并传输结果数据到子任务i所在节点后,子任务才能开始执行. 假设子任务i部署于RNn,记RNn的计算资源总量为f_n,部署在RNn的子任务i的分配的资源量为$f_{n,i}$,则子任务i的完成时间可表示为$t_{\text{end},i}=t_{i,n}+\max\limits_{i\in p(i)}\left(t_{ji,mn}+t_{\text{end},j}\right),m=\alpha_j,n=\alpha_i$. 其中,$t_{i,n}$表示子任务$i$在RN$n$的执行时间;$t_{ji,mn}$表示前继子任务$j$完成数据传输的时间;$t_{\text{end},j}$是前继任务完成时间;假设前继任务$j$完成后将结果数据传给后继任务$i$,因此$t_{\text{end},i}$可表示子任务$i$的完成时间.

$$t_{ij,mn}=\begin{cases}0, & m,n\in K\text{ and }m=n\\ \dfrac{e_{ij}}{B_{mn}}+p_{mn}, & \text{otherwise}\end{cases} \tag{2}$$

$$B_{m,n}\left(d,\chi\right)=B_{\text{eff}}B_w\log_2\left(1+\frac{P_mK_de^{\chi}}{N_oB_w\theta_{\text{eff}}d^{\alpha_d}}\right) \tag{3}$$

根据DAG任务的前后继关系,任务的总执行时间开销可表示为$T_{d,k}=\max\limits_{i\in K}t_{\text{end},i}-t_{\text{start},1}$,表示最后一个执行完的子任务的结束时间减去第一个任务开始时间. 对全局任务而言,有最大完成时延$T_{d,\max}=\max\limits_{K\in o}T_{d,K}$. 假设RN上可预部署子任务,部署决策意味着在特定RN上激活子任务对应的服务,因此本文忽略服务部署时间. 此外,起始子任务和结束子任务可以在网络中的任何RN上执行. 端侧资源利用率可表示为$\eta_u=\sum\limits_{n\in M}\sum\limits_{V\in K}\sum\limits_{k\in V}f_{n,k}\Big/\left(\sum\limits_{n\in M}f_n\right)$.

3　问题模型

3.1　协同资源节点选择

终端设备在长时间的高负载运行过程中,容易出现崩溃、死机、卡顿等问题,终端设备的稳定性对于用户来说非常重要. 终端设备的稳定性在实际运行中往往受多个因素影响,包括负载过高、电量不足、网络抖动、离线时间等情况,因此可以根据多方面因素对节点作为潜在协同资源节点进行评估. 为简化讨论,本节根据节点的剩余资源、节点距用户的距离、节点形态是否满足用户业务需求门限进行可用节点初筛,以实现更好的DAG任务与资源节点匹配. 记RNn中,$R_{1,n}$为剩余资源,$R_{2,n}$为终端的距离,$R_{3,n}$为终端形态,用户业务i对资源的需求分别为$R_{\text{th}1,i}$,距离上界为$R_{\text{th}2,i}$,形态类别要求为$R_{\text{th}3,i}$,适配业务i的节点初选规则可以表示为

$$S_i=\left\{n\middle|\, n\in U_i,\ R_{\text{th}1,i}\geqslant R_{1,n}, R_{\text{th}2,2}\geqslant R_{2,n}, R_{\text{th}3,i}=R_{3,n}\right\} \tag{4}$$

该评分用于资源节点的初筛,以压缩求解算法的解空间,提升训练和求解效率. 本文在针对待卸载应用分析潜在D2D协同网络时,会优先进行节点选择;后续算法决策实际卸载的节点时,只选择通过初筛的节点. 对于节点RNn,获得与RNn临近的其他Q个相邻RN(如果多于Q则只取最好的Q个)$n'\in S_i$的链路状况. 在部分观察区域中,这Q个RN为D2D第一跳范围. 这也是RNn可以上报给网络侧管理节点的RN集合. 管理节点可以依据RN上报的节点集合,通过节点合并等操作,构建局部关系图. 基于算法1可以构建协同计算节点基于D2D链路的协作关系拓扑,并且由于该过程中基站可以持续获取端侧资源信息,该关系拓扑以及包含的节点信息可以持续更新.

算法1　协同计算节点选择与D2D潜在协作关系拓扑构建算法

输入:RN上报的节点信息

输出:D2D潜在协作关系拓扑

1. 管理节点收到RN$u\in U$上报的临近RN信息
2. 选择满足业务需求的Q个RN,该集合记为S_i
3. 以管理节点下所有RN为顶点,初始化关系图$G(S_i)$的邻接矩阵
4. 如果RN(i)和任意RN(j)有边,关系图$G(S_i)$邻接矩阵的(i,j)和(j,i)置为1
5. 直到处理完所有RN(i),$i\in S_i$上报的信息

3.2　优化问题构建

考虑到多跳D2D卸载的业务应在资源约束限制下,尽可能提升服务质量和改善资源利用率. 在服务质量方面,本文关注服务的资源需求应得到满足,同时最小化系统中的服务完成时间,本文用$T_{d.\max}$表示$\max\left\{T_{d,\kappa},\kappa\in o\right\}$,因此最小化系统整体任务时延可表示为$\min T_{d.\max}$;在资源利用率方面,本文关注的目标是基于D2D通信,避免蜂窝空口通信资源浪费,提升端侧资源利用率,并降低边缘云资源占用. 因此,本文以最大化端侧资源利用率η_m和最小化最大任务执行时间$T_{dl.\max}$为目标,以DAG任务的子任务部署决策和资源分配为优化变量,构建优化问题如下:

$$\left\{\min_{X,\left\{f_{n,i}\right\}_{n\in s_i,i\in\kappa,\kappa\in o}} T_{d,\max},\ \max_{X,\left\{f_{n,i}\right\}_{n\in s_i,i\in\kappa,\kappa\in o}} \eta_m\right\}$$

$$\text{s.t.}\begin{cases}\text{C1: } t_{\text{start},i}\geqslant 0\\ \text{C2: } t_{\text{end},i}=t_{\text{start},i}+t_{i,k}\\ \text{C3: } \sum\limits_{i\in\kappa} x_{n,i}=1\\ \text{C4: } t_{\text{end},i}=t_{i,k}+\max\limits_{j\in P(i)}\left(t_{ji,mn}+t_{\text{end},j}\right), m=\alpha_j, n=\alpha_i\\ \text{C5: } \sum\limits_{\kappa\in o}\sum\limits_{i} f_{n,i}^{r} x_{n,i}\leqslant f_n\\ \text{C6: } \sum\limits_{(i,j)\in\left\{\alpha_i=m,\alpha_j=n\right\}} r_{o,ij}\leqslant r_{m,n}, m,n\in U\end{cases} \tag{5}$$

其中,约束条件C1保证子任务i的开始时间$t_{\text{start},i}$应该大于或等于0;约束条件C2确保子任务的执行是原子的;C3中,$x_{n,i}=1$代表将子任务i部署到RNn上,否则$x_{n,i}=0$,因为每个子任务具有不同的特征,所以每个子任务应该被分配到不同的RN,否则它们将被合并,因此$\sum\limits_{i\in K} x_{n,i}=1$;C4中限制子任务$i$的结束时间为$t_{\text{end},i}$,在其所有前驱任务完成并接收到结果后才能执行;C5表示子任务分配资源之和不能超过节点的资源上限;C6表示子任务的通信需求不能超过节点间通信信道容量. 该问题同时考虑了相互依赖的两个目标,需要根据任务之间的依赖关系和可用的资源,动态调整任务的调度策略. 考虑到任务调度需要快速响应卸载请求者,需要快速完成卸载决策,对算法效率要求极高,因此本文采用基于深度强化学习作为算法框架,实现对本文优化问题的高效求解,并通过仿真验证所提方案有效性.

4　基于有向图与卷积网络强化学习的端侧协同算力资源分配算法

4.1　算法框架

本节介绍基于深度强化学习(Deep Reinforcement Learning,DRL)和图神经网络(GNN)的任务调度算法. 在详细介绍之前,将强化学习(Reinforcement Learning,RL)算法简要介绍如下. 在RL算法中,智能体首先获取环境状态,然后,代理根据预定义的策略选择动作. 在执行该操作后,系统将在下一个时间段中通过与环境交互得到新的状态. 然后,RL代理将重复上述操作,即根据预期奖励大小选择新动作. 对于依赖任务,智能体需

要处理任务间的依赖关系. 考虑到GNN可以有效处理图数据,我们选择有向图卷积网络(DGCN)模型来提取有向任务图中顶点之间的信息,使用图卷积网络(GCN)来提取网络节点关系图的信息,整体框架如图2所示. 根据算法的组成结构,本节将介绍该算法的主要组件. 先介绍单个SAC智能体的状态信息,然后再介绍调度算法.

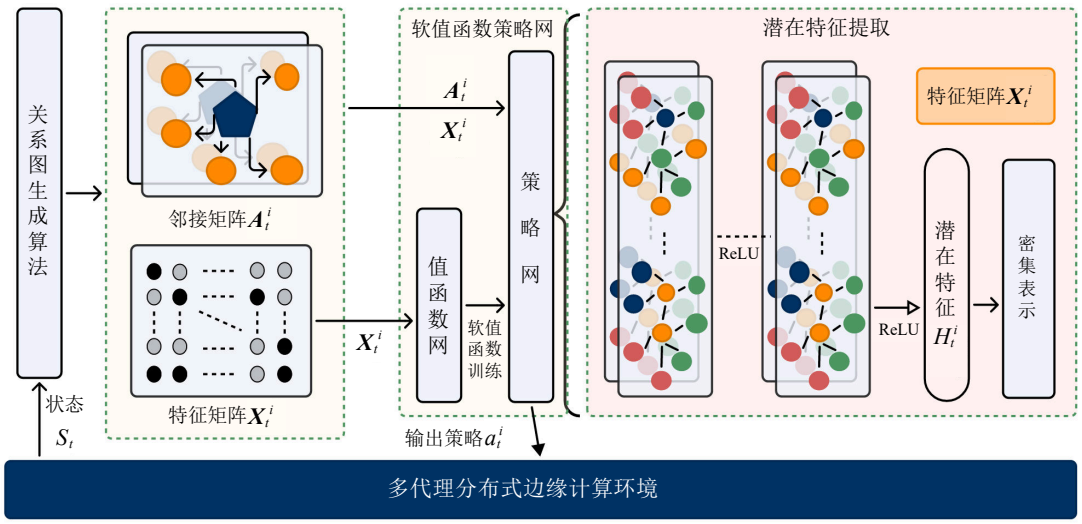


图2 基于图深度强化学习的DAG任务分配整体框架

4.2 基于GNN的状态表示

4.2.1 基于DGCN的任务图嵌入

对于一个组件化任务,节点特征包含子任务的计算任务量,边缘权重表示子任务之间的数据依赖. 由于子任务之间的依赖关系为有向的,使用了一个有向图卷积网络(DGCN)来将任务信息嵌入到一组向量当中,以更好地表示任务在调度决策时的必要信息. 用$K=G(V,E)$表示任务K的所有子任务属性和子任务之间的依赖关系. 其中,$v_i \in V$表示子任务的节点属性;$e_{ij} \in E$表示子任务之间传输的数据量. 则任务的初始特征可以表示为$X_k=[v_1; v_2; \cdots; v_k] \in \Re^{K\times d_k}$,其中$d_k$表示子任务的属性个数. 用DGCN嵌入任务信息的过程,使用一阶邻接矩阵$\boldsymbol{A}_F$,二阶入度邻接矩阵$\boldsymbol{A}_{s_{\text{in}}}$如式(6)所示,二阶出度邻接矩阵$\boldsymbol{A}_{s_{\text{out}}}$如式(7)所示,用来表示任务的有向图结构[29]. 其中$\boldsymbol{A}_F$中元素取值规则为若存在从节点i到节点j的边,则$\boldsymbol{A}_F(i,j)=e_{ij}$,否则$\boldsymbol{A}_F(i,j)=0$.

$$\boldsymbol{A}_{s_{\text{in}}}(i,j)=\sum_k\left(\boldsymbol{A}_{k,i}\boldsymbol{A}_{k,j}\Big/\sum_v\boldsymbol{A}_{k,v}\right) \tag{6}$$

$$\boldsymbol{A}_{s_{\text{out}}}(i,j)=\sum_k\left(\boldsymbol{A}_{i,k}\boldsymbol{A}_{j,k}\Big/\sum_v\boldsymbol{A}_{v,k}\right) \tag{7}$$

其中,$\boldsymbol{A}_{m,n}$表示原图邻接矩阵(m,n)处的元素. 接下来,对有向图进行三种变换:

$$\begin{aligned} Z_F&=\tilde{D}_F^{-\frac{1}{2}}\tilde{A}_F^{-\frac{1}{2}}\tilde{D}_F^{-\frac{1}{2}}X_k\Theta\\ Z_{S_{\text{in}}}&=\tilde{D}_{S_{\text{in}}}^{-\frac{1}{2}}\tilde{A}_{S_{\text{in}}}^{-\frac{1}{2}}\tilde{D}_{S_{\text{in}}}^{-\frac{1}{2}}X_k\Theta\\ Z_{S_{\text{out}}}&=\tilde{D}_{S_{\text{out}}}^{-\frac{1}{2}}\tilde{A}_{S_{\text{out}}}^{-\frac{1}{2}}\tilde{D}_{S_{\text{out}}}^{-\frac{1}{2}}X_k\Theta \end{aligned} \tag{8}$$

其中,Θ为基于参数$\{W,b\}$的线性变换操作,和GCN一样,$\tilde{A}_x=A_x+\lambda I$,$\tilde{D}_z=D_z+\lambda I$. 对于邻接矩阵$\boldsymbol{A}_x \in \mathbf{R}^{K\times K}$对应的度矩阵$\boldsymbol{D}_x$,其中$x \in \{F,S_{\text{in}},S_{\text{out}}\}$,其计算规则为$\boldsymbol{D}_x(i,j)=\hat{\sum_j}\boldsymbol{A}_x(i,j)$. 有向图的卷积层模型$\tilde{Y}=f(X,A)$如式(9)所示,其中$\alpha$和$\beta$是可学习的参数. 最后将$\tilde{Y} \in \mathbf{R}^{3K\times D}$展平,是输出的节点属性的个数,作为DGCN的输出,则该输出可表示为式(10).

$$\begin{aligned} \tilde{Y}=\text{Concat}\Big[&\text{ReLU}\left(Z_F\right),\alpha\,\text{ReLU}\left(Z_{S_{\text{in}}}\right),\\ &\beta\,\text{ReLu}\left(Z_{S_{\text{out}}}\right)\Big] \end{aligned} \tag{9}$$

$$S_{\text{DGCN}}=\text{flat}\left(\tilde{Y}\right) \tag{10}$$

4.2.2 基于GCN的D2D潜在协作关系图嵌入

对于一个无向的潜在协作关系拓扑图,节点特征包含RN的计算能力,边缘权重表示链路带宽. 将链路带宽作为邻接矩阵元素,分别使用GCN对邻域信息进行聚合,再将输出进行拼接,作为D2D潜在协作关系拓扑图的嵌入向量. 该拓扑图由资源节点以及潜在协作关系构成,其中RNn有属性f_n表示RN的计

算资源. 协同计算资源节点的初始特征可以表示为$F=\left[f_1;f_2;\cdots;f_n;\cdots\right]_{(M+MQ)d_D}$,其中$d_D$表示网络节点的属性个数,本文取值为1. 拓扑图中的边,使用一阶邻接矩阵$\boldsymbol{A}_F^{\text{band}}$和$\boldsymbol{A}_F^{\text{delay}}$来表示,取值规则为若存在一条边从节点$m$到节点$n$,或者从节点$n$到节点$m$,则$\boldsymbol{A}_F^{\text{band}}(m,n)=B_{m,n}$,$\boldsymbol{A}_F^{\text{delay}}(m,n)=p_{m,n}$,否则取0.

接下来,每一个卷积层的传播规则为$Y_{l+1}^Z=\text{ReLU}\left\{\tilde{D}^{-\frac{1}{2}}\tilde{A}^Z\tilde{D}^{-\frac{1}{2}}Y_l^Z\Theta^{(l)}\right\}$,$Z\in\{\text{band},\text{delay}\}$,其中,$\tilde{A}$和$\tilde{D}$的计算与前述DGCN相同,$Y_1^{\text{band}}=Y_1^{\text{delay}}=F$,$\Theta^{(l)}$是每一层基于$\{W,b\}$的线性变换. GCN的每一层通过邻接矩阵$\boldsymbol{A}$和特征矩阵$\boldsymbol{Y}_l$相乘得到每个节点邻居特征的汇总,然后再乘上一个参数矩阵,进行ReLU非线性变换,得到聚合邻居顶点特征的矩阵$\boldsymbol{Y}_{l+1}$. 对邻接矩阵进行归一化操作$\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$是为了信息传递过程中保持特征矩阵原有的分布,防止一些度数高和度数低的顶点在特征分布上产生较大的差异. 最终经过L层卷积,将Y_{L+1}^{band}和Y_{L+1}^{delay}进行拼接后再展平,作为GCN的输出,该输出表示为

$$S_{\text{GCN}}=\text{flat}\left(\text{Contat}\left[Y_{L+1}^{\text{band}},Y_{L+1}^{\text{delay}}\right]\right)\tag{11}$$

4.2.3　非图数据嵌入

非图结构数据$X_{\text{ng}}=\left\{N_{\text{sub}},r_{\text{tmp}},A_{\text{node}}\right\}$,其中,$N_{\text{sub}}$为当前需要调度的子任务编号;$r_{\text{tmp}}$为当前的调度结果;$A_{\text{node}}$为当前可用的节点. 使用全连接层提取非图结构数据特征,经过L层全连接,神经网络的输出如式(12)所示. 其中,$L=0$时,$X^{L+1}=X_{ng}$.

$$S_{ng}=\text{ReLU}\left(X^{L+1}\Theta^{L+1}\right)\tag{12}$$

4.3　DRL四要素

状态空间:$S_t=\left(S_{\text{DGCN}_t},S_{\text{GCN}_t},S_{\text{ng}_t}\right)$,强化学习的状态空间由DGCN、GCN以及全连接网络的输出拼接构成,S_{DGCN_t}、S_{GCN_t}、S_{ng_t}可分别由式(10)~式(12)得到.

动作空间:对于每一个子任务,决策目标是分配给子任务一个合适的节点. 因此,动作的范围是所有节点的集合,表示为$A_t\in\{n\}_{1+\varphi(Q)}=A$,其中$A_t$代表$t$时刻的动作,$A$是动作的集合. 在选择动作时,使用decaying-ε-greedy机制,以ε的概率随机选择动作,以$1-\varepsilon$的概率按照最大Q值选动作,表示为

$$A_t=\arg\max_{A_t}Q\left(S_t,A_t\right).$$

奖励:如前文所述,本文优化目标是提升端侧资源利用的同时最小化任务完成时间. 对于每一个子任务来说,给子任务分配节点资源以后,节点资源开销和链路通信开销是即时开销,而任务的总体执行时间和端侧资源利用率需要等到所有子任务都分配节点后才可以计算. 因此,将R_t分为即时奖励R_t^{in}和延迟奖励R_t^{dl}. 首先,根据当前子任务的调度决策计算即时奖励$R_t^{\text{in}}=\beta_M\eta_M+\beta_c\sum_{i\in\{5,6,7\}}l_{(C_i)}$,$l_{\{C_i\}}$表示约束$C_i$是否满足,满足取值1,否则取值0;然后,在所有子任务都分配到对应节点后,计算延迟奖励$R_t^{\text{dl}}=\beta_D T_{d,\max}$;最后,调度完成的总奖励可以表示为$R_t=R_t^{\text{in}}+R_t^{\text{dl}}$. $\beta=\left\{\beta_M,\beta_c,\beta_d\right\}$是超参数,用于调整奖励值到一个合适的范围,更好地训练神经网络. 本文采用基于模型训练的场景,由于环境计算中可以直接获取模拟终端的资源状态,并计算奖励值,因此获取资源状态开销在训练阶段可以忽略;而对于实际应用场景中,基站获取移动终端资源状态需要扩展一个监测数据维度,远小于现有协议中基站对移动终端网络状态的监测维度,但仍然会产生一定开销,限于本文关注整体框架和算法,这部分开销的具体影响将作为未来研究.

4.4　算法应用

由于系统中有U个基站对应U个管理节点,管理下属的M_u个终端,U个管理终端构成了多智能体协同系统. 本文使用多智能体强化学习框架,构建基于多智能体的资源分配算法,多智能体之间通过局部观察信息进行间接信息交互并通过全局环境反馈的奖励值促进协同. 具体而言,每个智能体仅通过自身的局部感知获取周围邻近智能体的信息,其观测区域存在重叠部分,从而能够间接“感知”到相邻智能体的状态和决策倾向. 这种隐式的信息共享方式不依赖于显式通信信道,而是通过观察环境中其他智能体的行为来推断彼此的决策偏好. 除了局部观测外,全局环境反馈的奖励信号为所有智能体提供了统一的评价标准. 通过环境奖励的正反馈机制,智能体在训练过程中能够调整策略,趋向于那些能带来较高全局回报的协同行为,从而逐步习得协作能力. 主要流程见算法2.

在接收到任务的DAG信息、网络资源和链路信息后,位于基站的智能体首先通过拓扑排序算法将该子任务添加到调度队列中. 然后,通过状态嵌入得到当前的状态S_t,通过动作选择策略选择动作A_t代理根据动作值将对应的网络节点分配给子任务,计算即时或延迟奖励,得到下一状态S_{t+1},并存储$\left[S_t,A_t,R_t,S_{t+1}\right]$到内存池中. 智能体训练之前需要先观察环境,存储数据. 当前步骤大于探索步骤时,代理会从内存池中随机选组小批数据进行训练. 训练的目的是最小化损失$L(\theta)$,$L(\theta)=E\left[\left(y(t)-Q\left(S_t,A_t;\theta\right)\right)^2\right]$,其中$Q\left(S_t,A_t;\theta\right)$是权值为$\theta$的策略网络的输出. $y(t)$为目标Q值,定义如式(13). 其中,α为学习率;γ为折扣率;θ'是目标网络的

参数. 目标网络和策略网络共享同样的网络结构. 目标网络定期复制策略网络的权值,以提供更加稳定的训练结果.

$$y(t)=E\left[\begin{matrix}(1-\alpha)Q\left(S_{t-1},A_{t-1};\ \theta'\right)\\+\alpha\left(R_{t-1}+\gamma\max Q(S_t,A_t;\ \theta')\right)\end{matrix}\middle|\ S_{t-1},A_{t-1}\right] \tag{13}$$

记多智能体 actor 网络为 $\pi=\left(\pi_u\right)_{u\in U}$,其参数为 $\theta=\left(\theta_u\right)_{u\in U}$,使用 π_{θ_u} 表示参数为 θ_u 的 actor 网络 π_u,输入状态观察,输出为动作决策;同样,使用 Q_{μ_u} 表示参数为 μ_u 的 critic 网络 θ_u,输入为状态动作对,输出为 Q 值. 根据集中式训练分布式执行框架和SAC,得到智能体 i 更新 actor 网络需要最小化目标式(14). 其中,经验回放池 D 用于存储智能体的状态 x、行为 a、$\tilde{a}=\left\{\tilde{a}_1,\tilde{a}_2,\cdots,\tilde{a}_U\right\}$ 奖励 r 和下一个状态 x',使用四元组 (x,a,r,x') 作为 D 中的一条存储,其中 $x=\left(o_1,o_2,\cdots,o_U\right)$、$a=\left(a_1,a_2,\cdots,a_U\right)$、$r=\left(r_1,r_2,\cdots,r_U\right)$、$x'=\left(o_1',o_2',\cdots,o_U'\right)$.

$$J_{\pi_u}\left(\theta_u\right)=E_{X\sim D,\tilde{a}\sim\pi_u}\left[\lambda\log\left(\pi_{\theta_u}\left(\tilde{a}_u\middle|o_u\right)\right)-Q_{\mu_u}(x,\tilde{a})\right] \tag{14}$$

每个智能体使用一个独立的 critic 网络 $Q_{\mu_u}(x,\tilde{a})$,输入为从经验回放池 D 中取样得到的所有智能体的观察 x 和动作,为了增加智能体训练的稳定性,使用当前智能体的 actor 策略产生动作 $\tilde{a}_u\sim\pi_{\theta_u}\left(\cdot\middle|o_u\right)$;输出 $Q_{\mu_u}(x,\tilde{a})$ 为智能体 u 动作 $\tilde{a}_u$ 的 Q 值. 分散执行的 actor 网络 $\pi_{\theta_u}\left(\cdot\middle|o_u\right)$ 仅利用其自身的观察 O_u 得到动作的概率分布,智能体通过对得到的概率分布采样获得具体执行的动作 $\tilde{a}_u$. 为了保证在进行采样后依然是可微的,设置 $\tilde{a}_u=\tilde{a}_{\theta_u}(o_u,\xi)$,$\xi\sim\mathrm{N}(0,1)$,对动作概率分布 $\pi_{\theta_i}\left(\cdot\middle|o_i\right)$ 进行采样操作. $J_{\pi_u}\left(\theta_u\right)$ 中的期望可以通过从经验回放池 D 采样进行近似计算,通过 SGD 算法对智能体 u 的 actor 网络进行更新,critic 网络参数 μ_u 可以通过最小化智能体 u 的贝尔曼误差 $J_B\left(\mu_u\right)$ 进行更新,如下所示:

$$J_{\mathrm{B}}\left(\mu_u\right)=\mathrm{E}_{x,a,r,x'\sim\mathrm{D}}\left[\frac{1}{2}\left(Q_{\mu_u}(x,a)-y\right)^2\right] \tag{15}$$

$$y=r_i+\gamma\mathrm{E}_{a'\sim\pi_{\bar{\theta}}}\left[Q_{\bar{\mu}_u}(x',a')-\lambda\log\left(\pi_{\hat{\theta}_u}\left(a_u'\middle|o_u'\right)\right)\right] \tag{16}$$

其中,$\bar{\mu}_u$ 为智能体 u 的目标 critic 网络 $Q_{\bar{\mu}_u}$ 参数,在经验回放池 D 中进行随机采样得到固定批量的样本对 $J_Q\left(\mu_u\right)$ 进行近似计算. 智能体 u 通过将下一个状态的观察 o_u' 输入目标 actor 网络 $\pi_{\bar{\theta}_i}$,然后根据得到的动作概率分布进行采样得到智能体的下一个动作 a_u'. 为了保证算法稳定更新,SAC 智能体也采用了评估网络和目标网络两套网络,又由于采用了 Actor-Critic 架构[30],因此智能体具有 4 个深度神经网络,分别为评估 Actor 网络和评估 Critic 网络以及目标 Actor 网络和目标 Critic 网络. 在训练过程中,只对评估网络进行训练,目标网络用于稳定深度神经网络的学习效果. 评估 Actor 网络和评估 Critic 网络参数经若干次更新后,需要对目标 Actor 网络参数 $\bar{\theta}_u$ 和目标 Critic 网络参数 $\bar{\mu}_u$ 进行软更新. 其中,τ 为控制目标网络软更新的超参数,θ_u 和 μ_u 分别是训练 Actor 网络和 Critic 网络最新网络参数.

$$\bar{\theta}_u=\tau\theta_u+(1-\tau)\bar{\theta}_u \tag{17}$$

$$\bar{\mu}_u=\tau\mu_u+(1-\tau)\bar{\mu}_u \tag{18}$$

算法2 基于MA-SAC的资源分配算法

输入:网络环境和任务信息

输出:任务调度结果

1. 初始化各智能体的 Actor 和 Critic 的评估网络和目标网络参数
2. 初始化每个智能体的经验池的大小 B_k
3. FOR $i=0$ TO $E-1$ DO
4. 重置边缘计算环境
5. FOR $u=0$ TO $U-1$ DO
6. 将 DAG 任务按照优先级排序
7. 根据算法 1 获得 PRG(u),并根据图嵌入结果,形成智能体 u 的状态 S_t^u
8. 智能体 u 选择执行的动作: $\mathrm{A}_t^u=\theta_u\left(S_t^u\right)$
9. 智能体 u 执行动作后,获得下一时刻的状态 S_{t+1}^u
10. END FOR
11. 所有智能体执行完动作,获取共同奖励 R_t
12. FOR $u=0$TO $U-1$DO
13. IF 经验池内存储的经验数量小于 B_uDO
14. 存储 $\left\{S_t,A_t,R_t,S_{t+1}\right\}$ 到智能体 u 的经验池中
15. ELSE
16. 用 $\left\{S_t,A_t,R_t,S_{t+1}\right\}$ 替换最早存储的经验
17. 随机选择大小为 B_b 的一小批量的经验样本
18. 通过式(13)更新 Critic 评估网络
19. 通过式(11)来更新 Actor 评估网络的参数
20. 根据式(15~16)更新 Actor 和 Critic 目标网络
21. END IF
22. END FOR
23. END FOR

5 仿真及实验

5.1 实验环境

本实验场景为一个移动边缘协同计算场景,包括两个边缘云和若干移动节点,计算单元的计算能力分别在[2,4] GHz 和[1,3] GHz 区间,边缘云的计算单元数是 20~40. 移动节点的峰值功耗分别设置为 15 W,空闲功耗设置为 10 W. D2D 协作跳数最多设置为 6,DAG 子任务的数量设置为 2~6. 此外,子任务的计算需求从 1×10^9~6×10^9 CPU 周期随机分布. 子任务之间的数据依赖性设置为[1,6] Mbits,其他主要参数见表 2. 仿真实

验在x86服务器上进行,基于Python和PyTorch搭建仿真环境,配置了4张RTX 3090 GPU用于智能体训练和推理. 为了验证所提方案在多种条件下的适用性,实验对比了多种边缘云和终端算力以及带宽资源配置,还对比了任务包含的子任务数、子任务计算和通信需求情况下,所提方案的性能表现.

表 2　仿真主要参数

参数	取值范围及单位
边缘云算力	[8,16] GHz
基站数量	12
基站与移动终端数量比	1∶5~1∶10
移动节点算力	[1,3] GHz
边缘云峰值功率	210 W
移动节点峰值功率	15 W
边缘云空闲功耗	185 W
移动节点空闲功耗	10 W
每个Agent梳理任务数	2~8个
DAG任务的子任务数量	2~6个
子任务计算资源需求	$[1\sim6]\times10^9$ cycles
子任务间数据依赖	[1,6] Mbits
子任务间最低带宽要求	[0.1,1] Mbps
CPU	双路Intel XEON Gold 5218R
内存	256 G
硬盘	4T SSD
GPU	RTX 3090

为了验证所提的D2D通信辅助多终端DAG任务协同机制的有效性,所提方案与没有节点选择和没有D2D通信辅助的多终端DAG任务协同机制进行了比较. 此外,在算法框架方面,所提方法还与CNN与GCN作为特征提取的两种方案进行了比较.

(1)CNN[31]来嵌入任务图和网络拓扑信息,并利用SAC框架进行强化学习训练.

(2)GCN[32]来嵌入有向任务图和网络拓扑信息,并利用SAC框架进行深度强化学习训练.

消融实验分析方案中节点选择、多跳D2D通信和任务调度算法对边缘网络算力资源协同调度的影响. 方案对比评估包括以下指标:

(1)计算任务延迟. 从任务开始到结束的总延迟,包括边缘节点和云中心的计算处理延迟以及节点之间的数据传输延迟.

(2)通信开销. 衡量多节点协同执行DAG任务时前后继任务的数据交互产生的通信资源开销,本文用多节点协同执行DAG任务期间数据传输占用的平均带宽进行度量.

(3)端侧资源利用率. 衡量是否充分利用端侧资源,本文用系统中的DAG任务占用RN资源的占用率平均值进行度量.

5.2　结果及分析

5.2.1　节点选择和D2D通信对移动边缘环境任务协同的影响

(1)节点选择机制对任务执行时间和通信开销的影响. 为了验证节点选择机制对多跳D2D任务协同方案的影响,本文比较了任务执行时间随任务数量的变化,如图3所示,图中图注的dl代表任务执行时间,bw代表通信开销,NS代表带有节点选择机制,NoNS代表未采用节点选择机制. 由图3可知,带有节点选择机制的任务调度方案,在不同子任务数量下,任务执行完成时间均比无节点选择的任务完成时间少,这是因为通过节点选择,能够降低无效节点的数量,压缩强化学习代理的解空间,从而提升有效解的质量,使用更优的节点组合和协作方案来完成任务计算,从而降低任务完成时间并降低通信开销,验证了节点选择的重要性.

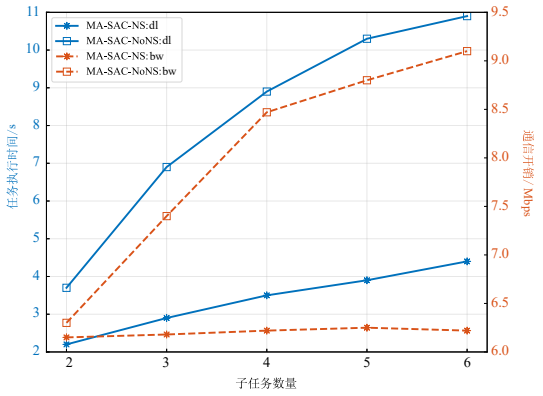


图3　任务执行时间随任务数量变化

(2)采用D2D通信对任务执行时间和通信开销的影响. 为了验证采用D2D通信对多跳D2D任务协同方案的影响,本文比较了网络通信开销随子任务间平均带宽需求的变化,如图4所示,采用D2D链路通信的MA-SAC-SL方案相比未采用D2D链路通信的MA-GSAC-NoSL方案,能有效降低网络通信开销,提升端侧任务协同效率,任务执行时间明显降低. 这是因为在采用D2D通信情况下,子任务之间可直接进行通信,避免了通过基站转发数据,从而有效降低基站空口的通信开销. 在平均带宽需求较高的情况下,采用D2D通信带来的通信开销的降低效果尤其明显.

5.2.2　不同特征嵌入方式对方案性能的影响

在前述基础上,本节验证不同特征嵌入方法对方案性能的影响.

(1)任务执行时间. 为了验证不同嵌入方案对多跳D2D任务协同方案的影响,本文比较了不同嵌入方案下任务执行时间随任务数量与子任务间平均带宽需求

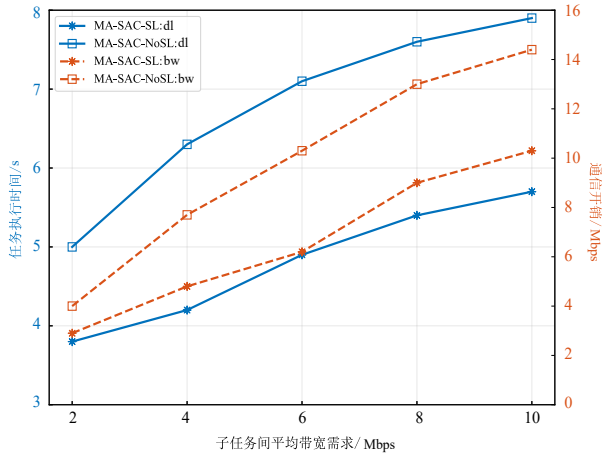
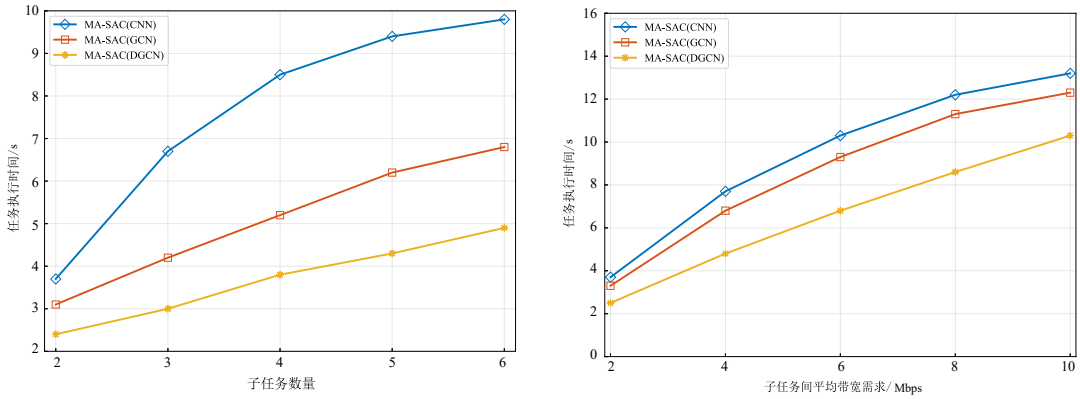


图4 通信开销随通信成本变化

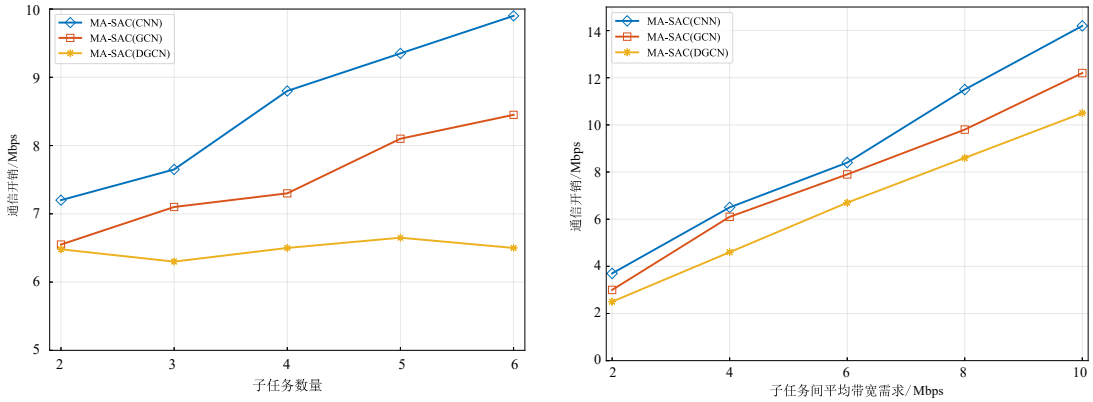
的变化情况,如图5所示.从结果可知,DGCN作为嵌入方案时,任务执行完成时间均比其他两种方案的任务完成时间少,这是因为GCN和CNN缺少图信息抽取能力,不能有效进行训练,因此产生的任务分配方案不够优秀,从而影响了任务完成时间,MA-SAC(DGCN)能有效提取任务、网络和资源约束特征,模型能够得到更有效的训练,解更优,因此任务完成时间更短.

(2)通信开销.本文还比较了不同嵌入方案下通信开销随子任务数量以及子任务间平均带宽需求的变化情况,如图6所示.从结果可知,DGCN作为嵌入方案时,通信开销均比其他两种方案的任务完成时间少,这是因为通信开销较大时,会影响任务完成时间,而本方案将任务完成时间作为优化目标之一,从前述分析已知GCN和CNN缺少图信息抽取能力,不能有效进行训练,因此产生的任务分配方案不够优秀.最终反映到通信开销方面,结果就是MA-SAC(DGCN)方案下通信开销低于另外两种对比方案,相比MA-SAC(CNN),MA-SAC(DGCN)的通信开销可降低38.2%.



(a) 任务执行时间与子任务数量的关系 (b) 任务执行时间与子任务间平均带宽需求的关系

图5 任务执行时间与子任务数量和子任务间平均带宽需求的关系



(a) 通信开销与子任务数量的关系 (b) 通信开销与子任务间平均带宽需求的关系

图6 通信开销与子任务数量和子任务间平均带宽需求的关系

(3)端侧资源利用率.端侧资源利用率提升也是本文方案关注的目标.图7评估在不同的端侧和边缘云资源比值从1:16上升到6:16情况下,端侧资源利用率的变化.从图7可知MA-SAC(DGCN)比对照方案有更

高的端侧资源利用率. 这是由于目标之一是充分利用端侧资源,但是GCN和CNN由于缺少图信息抽取能力,不能有效进行训练,因此产生的解可能无法完全满足约束要求,此时这些任务会回退到云端执行,因此造成了端侧资源利用率差异. DGCN能更有效提取任务图和网络图等图特征,因此终端资源利用率表现好于对比方案,结果显示相比MA-SAC(CNN)方案,能提升约31.9%.

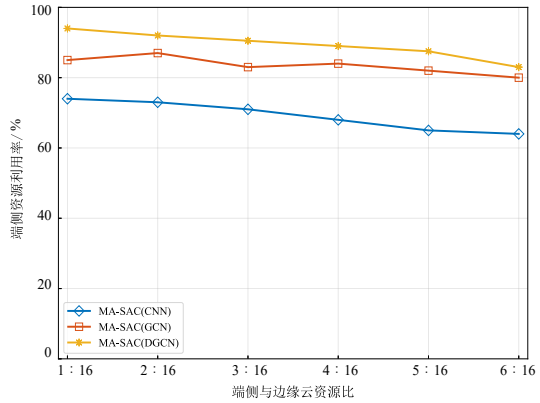


图7　不同云端资源量下端侧资源利用率

6　结论

本文提出了基于多跳D2D的DAG任务协同和分配框架,通过多跳D2D网络能够充分利用D2D链接进行直接通信,从而降低DAG子任务部署在不同节点间的任务跨节点传输难度,但同时引入了更高的D2D协作拓扑管理复杂度;本文提出了基于MA-SAC的协作拓扑和资源分配决策算法,通过将任务部署决策分配到基站为粒度的多个智能体,从而降低决策算法作用范围,同时通过多智能体协同降低由于资源分配产生的冲突,提升资源分配的效率. 仿真结果显示,本文所提方案能够在保证业务时延要求下,有效提升端侧资源利用率,降低蜂窝基站通信负载. 由于本文重点关注多跳D2D使能多终端协同处理DAG任务的总体框架和关键算法,子算法或流程如节点选择、感知机制存在优化空间,未来会针对性提出优化算法或机制,进一步优化所提方案性能;此外,本文工作可以扩展到云边端多级算力场景,通过层次化异构多智能体实现DAG任务调度,形成DAG计算任务在云边端多级算力环境下协同计算.

参考文献

[1] GU J H, FENG J H, XU H Y, et al. Research on terminal-side computing force network based on massive terminals[J]. Electronics, 2022, 11(13): 2108.

[2] LI X N, LU J G, ZHANG P M. A GNN-based routing and scheduling mechanism for multi-domain computing first network[C]//2023 IEEE 9th International Conference on Cloud Computing and Intelligent Systems (CCIS). Piscataway: IEEE, 2023: 153-163.

[3] JIN H, LUO R K, HE Q, et al. Cost-effective data placement in edge storage systems with erasure code[J]. IEEE Transactions on Services Computing, 2023, 16(2): 1039-1050.

[4] LI J C, ZHOU F Q, LI W J, et al. Componentized task scheduling in cloud-edge cooperative scenarios based on GNN-enhanced DRL[C]//NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium. Piscataway: IEEE, 2023: 1-4.

[5] ZENG Q, PENG B, LI Q, et al. Container-based task scheduling for edge computing using A multi-strategy hybrid of genetic beetles[C]//2022 IEEE 8th International Conference on Computer and Communications (ICCC). Piscataway: IEEE, 2022: 2198-2203.

[6] ZHU L L, HUANG K, HU Y F, et al. A self-adapting task scheduling algorithm for container cloud using learning automata[J]. IEEE Access, 2021, 9: 81236-81252.

[7] HU Y, DE LAAT C, ZHAO Z M. Multi-objective container deployment on heterogeneous clusters[C]//2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID). Piscataway: IEEE, 2019: 1.

[8] LAI W K, WANG Y C, WEI S C. Delay-aware container scheduling in Kubernetes[J]. IEEE Internet of Things Journal, 2023, 10(13): 11813-11824.

[9] MENG C Y, SONG S J, TONG H G, et al. DeepScaler: Holistic autoscaling for microservices based on spatiotemporal GNN with adaptive graph learning[C]//2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE). Piscataway: IEEE, 2023: 53-65.

[10] FATICANTI F, DE PELLEGRINI F, SIRACUSA D, et al. Cutting throughput with the edge: App-aware placement in fog computing[C]//2019 6th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2019 5th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom). Piscataway: IEEE, 2019: 196-203.

[11] HUANG Y K, QIAO X Q, LAI W H, et al. Enabling DNN acceleration with data and model parallelization over ubiquitous end devices[J]. IEEE Internet of Things Journal, 2022, 9(16): 15053-15065.

[12] DUAN S J, WANG D, REN J, et al. Distributed artificial intelligence empowered by end-edge-cloud computing: A

survey[J]. IEEE Communications Surveys & Tutorials, 2023, 25(1): 591-624.

[13] NING Z L, DONG P R, WANG X J, et al. Distributed and dynamic service placement in pervasive edge computing networks[J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 32(6): 1277-1292.

[14] MA L L, YI S H, CARTER N, et al. Efficient live migration of edge services leveraging container layered storage[J]. IEEE Transactions on Mobile Computing, 2019, 18(9): 2020-2033.

[15] ZHANG X F, ZHANG J, LIU Z T, et al. MDP-based task offloading for vehicular edge computing under certain and uncertain transition probabilities[J]. IEEE Transactions on Vehicular Technology, 2020, 69(3): 3296-3309.

[16] 3GPP. Proximity-based services (ProSe); Stage 2: TS 23.303 v17.7.0[S].

[17] 3GPP. Proximity based Services (ProSe) in the 5G System (5GS): TS 23.304 v17.7.0[S].

[18] DAS A, DAS N, DAS BARMAN A. Multi-hop D2D communication in cellular networks to minimize EMR[J]. IEEE Transactions on Green Communications and Networking, 2022, 6(2): 713-722.

[19] TEJA P R, MISHRA P K. Path selection and resource allocation for 5G multi-hop D2D networks[J]. Computer Communications, 2022, 195: 292-302.

[20] MUY S, LEE J R. Spectrum efficiency maximization for multi-hop D2D communication underlaying cellular networks: Machine learning-based methods[J]. Expert Systems with Applications, 2023, 213: 118167.

[21] SUN H, NARAGHI-POUR M, SHENG W X, et al. A hop-by-hop relay selection strategy in multi-hop cognitive relay networks[J]. IEEE Access, 2020, 8: 21117-21126.

[22] 叶梓峰, 王永华, 万频, 等. 基于优先记忆库结合竞争深度Q网络的动态功率控制[J]. 电讯技术, 2019, 59(10): 1132-1139.
YE Z F, WANG Y H, WAN P, et al. A dynamic power control strategy based on dueling deep Q network with prioritized experience replay[J]. Telecommunication Engineering, 2019, 59(10): 1132-1139. (in Chinese)

[23] WANG J S, LIU Y, ZHANG W T, et al. ReLFA: Resist link flooding attacks via renyi entropy and deep reinforcement learning in SDN-IoT[J]. China Communications, 2022, 19(7): 157-171.

[24] BAO R M, ZHOU Y L, JIANG W R. FL-CNN-LSTM: Indoor air quality prediction using fuzzy logic and CNN-LSTM model[C]//2022 2nd International Conference on Electrical Engineering and Control Science (IC2ECS). Piscataway: IEEE, 2022: 986-989.

[25] AWASTHI A K, GAROV A K, SHARMA M, et al. GNN model based on node classification forecasting in social network[C]//2023 International Conference on Artificial Intelligence and Smart Communication (AISC). Piscataway: IEEE, 2023: 1039-1043.

[26] FERRIOL-GALMÉS M, PAILLISSE J, SUÁREZ-VARELA J, et al. RouteNet-Fermi: Network modeling with graph neural networks[J]. IEEE/ACM Transactions on Networking, 2023, 31(6): 3080-3095.

[27] HAN X Y, XIE M X, YU K, et al. Combining graph neural network with deep reinforcement learning for resource allocation in computing force networks[J]. Frontiers of Information Technology & Electronic Engineering, 2024, 25(5): 701-712.

[28] ANAMURO C V, VARSIER N, SCHWOERER J, et al. Distance-aware relay selection in an energy-efficient discovery protocol for 5G D2D communication[J]. IEEE Transactions on Wireless Communications, 2021, 20(7): 4379-4391.

[29] 肖硕, 黄珍珍, 张国鹏, 等. 基于SAC的多智能体深度强化学习算法[J]. 电子学报, 2021, 49(9): 1675-1681.
XIAO S, HUANG Z Z, ZHANG G P, et al. Deep reinforcement learning algorithm of multi-agent based on SAC[J]. Acta Electronica Sinica, 2021, 49(9): 1675-1681. (in Chinese)

[30] 秦鹏, 王硕, 付民, 等. 基于MATD3的空地网络资源优化[J]. 中国科学: 信息科学, 2024, 54(6): 1474-1486.
QIN P, WANG S, FU M, et al. Air-ground integrated network resource optimization based on MATD3[J]. Scientia Sinica (Informationis), 2024, 54(6): 1474-1486. (in Chinese)

[31] TONG Z, DENG X M, YE F, et al. Adaptive computation offloading and resource allocation strategy in a mobile edge computing environment[J]. Information Sciences, 2020, 537: 116-131.

[32] MO C T, CHEN J H, LIAO W. Graph convolutional network augmented deep reinforcement learning for dependent task offloading in mobile edge computing[C]//2023 IEEE Wireless Communications and Networking Conference (WCNC). Piscataway: IEEE, 2023: 1-6.

作者简介

顾健华　男,1972年10月出生于安徽省马鞍山市. 现为清华大学计算机科学与技术系博士研究生. 主要研究方向为分布式计算、端侧算力网络.
E-mail: gujh19@mails.tsinghua.edu.cn

冯建华　男,1967年8月出生于山西省运城市. 现为清华大学计算机科学与技术系教授. 主要研究方向为数据库、分布式数据库和数据安全.
E-mail: fengjh@tsinghua.edu.cn

许辉阳　男,1984年9月出生于安徽省安庆市. 现为中国移动终端公司高级工程师、博士后. 主要研究方向为云终端、边缘计算、端侧算力网络、端侧AI.
E-mail: xuhuiyang@cmdc.chinamobile.com

刘佟佟　男,1982年2月出生于黑龙江省大庆市. 现为中国移动终端公司工程师. 主要研究方向主要研究方向为云终端、端侧算力网络.
E-mail: liutongtong@cmdc.chinamobile.com

周　婷　女,1988年12月出生于内蒙古自治区包头市. 现为中国移动终端公司工程师. 主要研究方向为端侧算力网络、端侧AI.
E-mail: zhouting_fx@cmdc.chinamobile.com