SDWN 中基于多智能体图强化学习的多对多通信路由方法

文 鹏[1],叶 苗[1,2,3*],王 勇[4],何 倩[4],仇洪冰[1,3]

(1. 桂林电子科技大学信息与通信学院,广西桂林 541004;2. 桂林电子科技大学广西无线宽带通信与信号处理重点实验室,广西桂林 541004;3. 桂林电子科技大学认知无线电与信息处理省部共建教育部重点实验室,广西桂林 541004;4. 桂林电子科技大学计算机与信息安全学院,广西桂林 541004)

摘 要: 多对多通信路由问题是NP(Nondeterministic Polynomial time)难的组合优化问题,构建出高效的多对多通信路由路径还需及时获取全局网络状态信息以适应网络状态高度动态变化的特点. 本文在软件定义无线网络(Software-Defined Wireless Networks,SDWN)场景中针对现有数据驱动的多智能体深度强化学习方法存在计算和部署成本高、难以适应非欧结构特点的网络拓扑的问题,并且训练过程中无效动作过多会增加存储空间和时间开销以及收敛速度慢,本文设计了一种SDN控制平面和数据平面进行协同感知与智能决策的新框架,并针对多对多通信路由问题设计了一种两阶段的多智能体路由方法(基于智能节点部署策略的多智能体图强化学习方法:MAGDS-M2M). 为了降低在每个节点上都部署智能体所带来的计算和部署成本,设计了一种基于Q-学习的智能节点部署算法来确定需要部署智能体的网络节点;在完成多智能体部署后,在Actor-Critic(AC)框架下设计了一种基于多智能体图强化学习的多对多路由决策方法,基于图卷积网络(Graph Convolutional Networks,GCN)和图神经网络(Graph Neural Networks,GNN)重新设计Actor和Critic网络,解决了现有多智能体强化学习方法中卷积神经网络(Convolutional Neural Networks,CNN)对拓扑结构数据适应能力比较弱的问题;此外,为解决Actor网络固定长度的动作空间在训练过程中产生大量无效动作的问题,设计了一种新的动作空间局部观测方法. 实验结果表明所提出的方法相比于基准实验降低了29.33%任务完成时延,并且验证了可以通过调节参数使任务完成的时延和各节点累计能耗标准差之间达到平衡. 本文所做工作源代码已提交至开源平台https://github.com/GuetYe/MAGDS-M2M.

关键词: 多对多通信;智能节点部署;多智能体图强化学习;动作空间局部观测方法;软件定义无线网络

基金项目: 国家自然科学基金(No.62161006,No.62372353);广西无线宽带通信与信号处理重点实验室基金(桂科AD25069102);广西研究生教育创新计划基金(No.YCBZ2023134);认知无线电与信息处理教育部重点实验室主任基金(No.CRKL220103)

中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112(2025)06-1885-21

电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20240980

A Multi-Agent Graph Reinforcement Learning Method for Many-to-Many Communication Routing in SDWN

WEN Peng[1], YE Miao[1,2,3*], WANG Yong[4], HE Qian[4], QIU Hong-bing[1,3]

(1. *School of Information and Communication, Guilin University of Electronic Technology, Guilin, Guangxi* 541004, *China*; 2. *Guangxi Wireless Broadband Communication and Signal Processing Key Laboratory, Guilin University of Electronic Technology, Guilin, Guangxi* 541004, *China*; 3. *Key Laboratory of Cognitive Radio and Information Processing, Ministry of Education, Guilin University of Electronic Technology, Guilin, Guangxi* 541004, *China*; 4. *School of Computer and Information Security, Guilin University of Electronic Technology, Guilin, Guangxi* 541004, *China*)

Abstract: The many-to-many communication routing problem is an NP(Nondeterministic Polynomial time)-hard combinatorial optimization problem. Constructing efficient many-to-many communication routing paths requires timely acquisition of global network state information to adapt to the highly dynamic nature of network states. In this paper, within the context of software-defined wireless networks (SDWN), we address the issues present in existing data-driven multi-

收稿日期:2024-10-29;修回日期:2025-05-05;责任编辑:覃怀银

*通讯作者:叶苗

agent deep reinforcement learning methods, such as high computational and deployment costs, difficulty in adapting to the non-Euclidean characteristics of network topologies, excessive invalid actions during training leading to increased storage and time overheads, and slow convergence rates. This paper designs a new framework for collaborative sensing and intelligent decision-making between the SDN control plane and data plane and proposes a two-stage multi-agent routing method (Multi-Agent Graph deep reinforcement learning method based on intelligent node Deployment Strategy, MAGDS-M2M) to address the multi-to-multi communication routing problem. To reduce the computational and deployment costs associated with deploying agents on every node, a Q-learning-based intelligent node deployment algorithm is designed to determine the network nodes where agents need to be deployed. After completing the multi-agent deployment, a multi-to-multi routing decision method based on multi-agent graph reinforcement learning is developed within the actor-critic (AC) framework. This method redesigns the actor and critic networks using graph convolutional networks (GCN) and graph neural networks (GNN), addressing the weak adaptability of convolutional neural networks (CNN) to topological structure data in existing multi-agent reinforcement learning approaches. Additionally, to solve the issue of generating a large number of invalid actions during training caused by the fixed-length action space of the Actor network, a new local observation method for the action space is proposed. Experimental results demonstrate that the proposed method reduces task completion delay by 29.33% compared to benchmark experiments and verifies that by adjusting parameters, a balance can be achieved between task completion delay and the standard deviation of cumulative energy consumption across nodes. The source code developed in this work has been submitted to the open-source platform at https://github.com/GuetYe/MAGDS-M2M.

Key words: many-to-many communication; intelligent node deployment; multi-agent graph reinforcement learning; action space local observation method; soft-ware-defined wireless networks

Foundation Item(s): National Natural Science Foundation of China (No.62161006, No.62372353); The Project of Guangxi Wireless Broadband Communication and Signal Processing Key Laboratory (No.AD25069102); Innovation Project of Guangxi Graduate Education (No.YCBZ2023134); Key Laboratory of Cognitive Radio and Information Processing, Ministry of Education (Guilin University of Electronic Technology) (No.CRKL220103)

1 引言

随着无线网络技术的快速发展和普及,无线网络用户的数量和数据流量急剧增加,通信模式涵盖了一对一、一对多和多对多通信的方式.多对多通信也称为多源多目标通信或多源多汇通信,指的是多个源点同时向多个目的点发送数据[1],这种通信方式可以通过并行数据交换提高网络的资源利用效率和降低通信的延迟,从而满足用户服务质量的要求,有文献证明了这样的多对多通信路由规划问题实质上是数学优化理论当中的一个NP(Nondeterministic Polynomial time)难组合优化问题[2,3].多对多通信广泛应用于信息中心网络(Information-Centric Networking,ICN)[4]、卫星网络[5]、无线传感器网络(Wireless Sensor Network,WSN)[6]等多个领域,比如在移动边缘计算(Mobile Edge Computing,MEC)中,谷歌设计的视频流媒体平台Anvato通过多对多通信实现在线视频编辑和广告插入[7];在大数据新兴的分布式机器学习方式中也利用类似多对多通信模式来促进分布式模型训练[8]效率的提升.

近年来许多研究学者密切关注和开展对多对多通信模式下路由规划策略的研究.很多经典的MUSTER(MUlti-Source multi-sink Trees for Energy-efficient Routing)协议[9]、MMForests[10](Multi-source Multicast Forests)和基于支持向量机(Support Vector Machine,SVM)子树VNF(Virtual Network Function)实例合并的启发式算法[11]等多对多通信的路由决策方法考虑通过独立地为每个源生成树,再将它们合并成森林来实现.这类基于局部信息考虑的算法思路简单且容易实现,然而,由于忽略了树与树之间的协作与竞争,它们往往只能得到局部最优解.通过获取网络的全局状态信息可以实现集中式求解多对多通信路由决策问题.Mottola等人[9]也指出可以通过集中计算方式获得同时从多个源节点到多个汇点的最优路由策略;Jain等人[12]也说明如果可以由一个集中的中央实体对各节点的数据包传输进行精细路由决策,可以显著提高吞吐量.这些方法都依赖于全局网络的状态信息,然而,以上应用场景的网络状态信息都是高速动态变化的,传统无线网络管理方式难以及时获取全局状态信息,这给求解出满足网络服务质量(Quality of Service,QoS)的多对多通信路由规划带来了挑战.

软件定义网络(Software Defined Network,SDN)架构将控制平面与数据平面分离,它可以实时获取全局网络状态信息[13],提高对网络状态变化的感知能力,而软件定义无线网络(Software Defined Wireless Network,SDWN)架构[14]将SDN思想应用于无线网络,能够在控制层设计合理的路由策略,因此能够有效适应多对多无线通信应用场景的需求.

文献[3,11]对多对多通信这样一个NP难组合优

化问题进行了求解方法上的讨论. Ren等人[11]对VNF中的多源多播问题建立了一种生成服务功能树的优化模型,并设计了基于Viterbi算法[15]的近似求解算法MDNM(Multi-source Delay-aware NFV-enabled Multicasting),这种方法类似于动态规划方法,由于偏向静态场景下对问题的讨论,因此对高速动态变化网络场景下的效果有限;Guo等人[3]在SDN中提出了P-MCF(Prior approaches for Forest with the Minimum Cost)和E-MCF(Efficient routing Forest with the Minimum Cost)2种近似算法求解多源多播的路由策略. P-MCF简单易行,但效果不如E-MCF,而E-MCF却难以识别共享节点. 这些方法可以获取比较好的多对多通信路由决策,却很难适应网络状态信息高度动态变化的特点. 因此,设计出能够适应网络状态信息高速动态变化特性以满足多对多通信路径规划的业务需求,显得尤为重要.

相比以上经典的多对多通信问题求解方法,强化学习是一种数据驱动的人工智能方法,可以很好地适应网络状态信息高度动态变化的特点. 文献[16,17]中详细讨论了对单播、组播路由路径规划问题的强化学习求解方法. 文献[18]还使用了多智能体强化学习方式得到了更好的性能. 多智能体强化学习作为一种高级强化学习方式,它利用分布式智能体之间的相互协作,可以更好地适应网络动态状态的动态变化. 文献[19]还设计了一种基于策略价值(Actor-Critic,AC)框架的多智能体强化学习的方法,其中多智能体使用确定性策略梯度(Multi-Agent Deep Deterministic Policy Gradient,MADDPG)来建立不断变化的通信网络拓扑动态协调图(Dynamic Coordination Graph,DCG),智能体之间通过相互协调合作方式来完成对最优路由方案的求解,然而,由于设计的MADDPG方法在构建Actor和Critic神经网络结构时采用了CNN(Convolutional Neural Network)方法,难以适应具有非欧结构特点[20]的通信网络拓扑结构数据,从而导致模型训练时存储空间开销的增加. 此外,由于多智能体强化学习中Actor网络的输出常被设置为指定长度,需要设计一些筛选或惩罚机制来减少多智能体在学习过程中产生的无效动作,这样会增加模型训练的时间开销,降低算法收敛速度. 因此,使用多智能体深度强化学习方法解决路由规划问题还需要设计合理的深度网络结构,降低模型训练时的时间和空间开销,提高算法的收敛速度. 目前也逐渐开始有文献将多智能体强化学习应用到多对多通信问题的研究中,并取得了不错的效果. Chen等人[21]提出了一种无模型多智能体元近端策略优化(Multi-Agent meta-Proximal Policy Optimization,meta-MAPPO)的强化学习方法,用于解决高度动态变化的环境下的多对多通信路由决策问题,该方法需要在所有网络节点上部署智能体,而网络节点规模扩大则带来了路由策略求解的空间和时间复杂度以及部署成本的增加. 从这些最近研究工作可以看出,有效地运用数据驱动方式的多智能体强化学习方法来解决多对多通信问题,还需要考虑智能体训练的计算和部署成本问题.

通过对以上现有方法的分析,本文在SDWN架构下为多对多通信问题设计了一种基于智能节点部署的多智能体图强化学习方法(Multi-Agent Graph reinforcement learning method based on an intelligent node Deployment Strategy in Many-to-Many communication,MAGDS-M2M). 该方法包含2个阶段:第一阶段为确定部署智能体的网络节点位置设计了一种基于Q-学习算法,优化部署智能体的网络节点数量,从而降低已有文献方法对所有节点部署智能体带来的计算和部署成本;第二阶段,在确定多智能体的部署策略后,针对多对多通信路由问题,在AC框架下设计了一种集中式训练分布式执行的多智能体深度强化学习方法,该方法为了减少模型训练的存储空间开销分别使用GCN(Graph Convolutional Network)和GNN(Graph Neural Network)重新设计Actor和Critic的网络结构,此外,在Actor网络生成动作的过程中,设计了一种局部观测方法来避免无效动作的生成,从而降低模型训练的时间开销并加快收敛速度.

本文的主要研究贡献如下:

(1)相比在现有软件定义网络中数据驱动方式的智能架构体系,仅仅是在控制平面设计和实现了具备智能路由决策功能的强化学习方法,本文设计了一种SDN控制平面和数据平面进行协同感知与智能决策的新框架,并针对多对多通信路由问题设计了基于Q-学习和多智能体图强化学习的两阶段多智能体路由方法.

(2)相比现有求解多对多通信问题的多智能体强化学习方法将所有节点进行智能体部署的方式,本文设计了一种基于Q-学习的智能体部署策略,不仅具备适应网络状态信息高度变化的特点,还能降低部署成本和后续路径规划的计算开销.

(3)针对多对多通信路由具体问题特点,在AC框架下设计的集中式训练分布式执行的多智能体深度强化学习方法采用GCN和GNN分别设计了Actor和Critic网络结构,能有效减少模型训练的存储空间开销,并通过在Actor网络生成动作的过程中设计一种局部观测方法可以避免无效动作的生成,降低模型训练的时间开销并加快收敛速度.

2　相关工作

现有多对多通信路由规划问题相关的求解方法大

致可分为传统优化方法、群智能优化方法以及人工智能优化方法,本节将分别对它们进行介绍.

(1)传统优化方法. Mottola等人[9]为WSN设计了一种针对多对多通信的MUSTER路由协议. 该算法从独立构建的树开始. 它利用1-hop邻域内的路径信息,随着相邻节点同时引导不同树的流量,以完全去中心化的方式逐步完成树的形成. Tan等人[22]提出一种针对多汇点WSN的分布式流量平衡路由算法,该方法通过检测跳数、单跳邻居的数据包数量和双跳邻居的最小数据包数量,为每个节点构建梯度场,从而将节点上的数据包转发给最低梯度的邻居. Chen等人[10]为多流多源多播路由问题设计了一个最大化路径带宽的启发式算法(MMForests),该算法通过修改Dijkstra从目的节点开始对源节点进行最大带宽路径检索,为每个源和目的节点构建最大带宽路径,然后将同源的路径合并成树,再将非同源的树合并成森林. 这些方法进行路由决策的时候通常只考虑局部的网络状态信息建立路径或树,然后将它们进行合并,简单易行,但由于缺乏对全局网络信息统筹规划,获得的多对多通信路由规划方案效果有限.

(2)群智能优化方法. 目前现有使用群智能算法解决通信路由问题的文献多数集中在对单播或多播路由问题的讨论. Sun等人[23]提出了一种基于遗传算法(Genetic Algorithm,GA)多约束QoS组播路由算法(Multiple Constrained QoS Multicast Routing Algorithm,MCQMRA),适用于互联网、移动网络或其他高性能网络,能够在参数不确定的网络环境中以可扩展和灵活的方式提供满足QoS的路径. Li等人[24]在自组织网络中提出了一种结合了进化算法(Evolutionary Algorithm,EA)的快速全局搜索能力和鲁棒性,以及蚁群算法(Ant Colony Algorithm,ACA)的信息素反馈机制,以解决QoS多重约束组播路由问题的进化和蚁群混合优化算法EA-ACA-QMRA(QoS Multicast Routing Algorithm). Mann等人[25]在能量受限的无线传感器网络(Wireless Sensor Networks,WSNs)环境中提出了一种基于群体智能的能量高效分层路由协议(BeeSwarm),以优化数据包传递、能量消耗和吞吐量,并延长了网络寿命. Pan等人[26]为解决现有群智能算法精度低、扩展性差、缺乏多样性和分布不均等问题,提出了一种量子二进制人工蜂群算法(Quantum Binary Artificial Bee Colony algorithm,QBABC),有效地增强了VANET (Vehicle Ad hoc NETwork)中的可靠路由连接. 借鉴传统优化方法思路,可以先通过这些方法生成部分路由,然后再合并成多对多通信的路由. 然而由于群智能方法计算量比较大,收敛速度有限,难以适应具有高度动态变化网络状态的多对多通信路由问题.

(3)人工智能优化方法. Boyan等人[27]首次尝试利用Q-学习进行数据包路由优化,其结果在平均数据包传递时间上优于最短路径算法. 近年来,Casas-Velasco等人[16]在SDN中提出了一种基于Q-学习的智能路由算法. 随着深度强化学习的发展,Yao等人[28]提出的基于深度Q网络的能效路由算法,有效解决了软件定义数据中心网络中的能量消耗和负载均衡问题. Lu等人[29]提出了一种基于深度确定性策略梯度(Deep Deterministic Policy Gradient,DDPG)的子流自适应多路径路由算法,有效适应了数据中心网络中的动态流量变化,降低了延迟,提高了传输成功率并增加了吞吐量. Zhou等人[30]提出了一种基于临近策略优化(Proximal Policy Optimization,PPO)的QoS感知路由优化机制(PPO-based QoS-aware Routing Optimization Mechanism,PQROM),具有良好的收敛性和稳定性,且在训练时间、超参数调整和硬件消耗方面均得到提升. 然而,这些方法只适用于单播或组播等路由问题,应用于多对多通信,还必须要考虑不同的源会话间的合作. Qiu等人[31]提出了一种基于多智能体强化学习的地理位置路由协议QLGR-S,通过将每个节点视为智能体,利用本地信息评估其邻居节点的价值,该协议有效降低了数据包丢失率和路由成本,并减少了路由盲区的发生. Okine等人[18]提出了一种针对战术无线传感器网络(Tactical Wireless Sensor Networks,T-WSNs)的分布式多智能体深度强化学习(Multi-Agent Deep Reinforcement Learning,MADRL)路由决策方案. 该方案旨在应对链路层的干扰攻击,优化数据包路由,以满足严格的延迟和能量要求. 多智能体强化学习能够适应不同源通信的协作与竞争,目前在多对多通信问题上的应用研究正受到越来越多的关注. Chen等人[21]将多对多通信的路由决策问题建模为一个多智能体部分可观察马尔可夫决策过程(Markovian Decision Process,MDP),提出了一种基于多智能体强化学习的求解方案. 然而,上述多智能体强化学习方法都需要在每个节点上部署智能体,随着网络节点规模增大,相应的计算成本会显著增大.

基于以上现有方法的讨论,可以看出多智能体强化学习方法被认为是解决多对多通信更好的方法,然而这方面的研究还在起步阶段,存在很多不足之处. 本文在SDWN架构下为多对多通信问题设计了一种考虑智能体部署的多智能体深度强化学习求解方法(MAGDS-M2M).

3 多对多通信问题描述与建模

本文将SDWN中的多对多通信路由决策问题表示如下:假设将SDWN网络建模为一个无向图$G=(N,E)$,其中N表示有限个数的网络节点集合,E表示有限条数

的链路集合,用$n=|N|$表示网络节点的个数,$s_i \in N$表示网络中的编号为i的节点,用$e_{ij} \in E, s_i, s_j \in N, i \neq j$表示网络节点$s_i$与网络节点$s_j$之间存在的链路,用矩阵$\boldsymbol{A}=[a_{ij}] \in \{0,1\}^{n \times n}$表示网络中网络节点连接方式,即图$G$的邻接矩阵,其中$a_{ij}$满足:

$$a_{ij}=\begin{cases}1, & e_{ij} \in E \\ 0, & e_{ij} \notin E\end{cases} \tag{1}$$

假设SDWN中发送数据的源节点集合表示为$N_s=\{s_i\} \subseteq N$,用$n_s=|N_s|$表示所有源节点的个数,将接收数据的目的节点集合表示为$N_d=\{s_i\} \subseteq N$,用$n_d=|N_d|$表示目标节点的个数,将多对多通信任务用流量矩阵$\boldsymbol{M}=[m_{ij}] \in \mathbb{N}^{n_s \times n_d}$来表示[21],其中$\mathbb{N}$表示自然数集合,元素$m_{ij}$表示从源节点$s_i$发往目标节点$s_j$的数据包数量,则可以用$m^p=\boldsymbol{I}^T\boldsymbol{M}\boldsymbol{I}$表示任务$\boldsymbol{M}$中包含数据包的数量,这里的$\boldsymbol{I}$是相应维度的全1列向量.

依据和文献[21]一致的假设和表示方式,在这里假设SDWN中数据包的传输遵循如下的规则:在每个时间周期T_s内生成一次多对多通信任务流量矩阵$\boldsymbol{M}$,源节点在此周期T_s内进行传输,并要求数据传输在周期T_s内完成,否则将会丢弃未传输完成的数据包. 假设在周期T_s内多对多通信任务的数据包个数是有限的,存在上界κ,即满足$m^p<\kappa$.

数据包在经过网络节点缓存区时采用先进先出(First-In First-Out,FIFO)的机制,网络节点缓冲区大小相同且均为κ,由前面$m^p<\kappa$的假设可知数据包不会因为网络节点缓冲区溢出丢失. 如图1所示,假设网络节点发送一次数据包是在单位时间t_s内完成,t表示在时间周期T_s中的第t个发送数据包的时间步,下一次发送的时间步记为$t+1$,他们之间间隔一个单位时间t_s. 此外,假设网络节点s_i在单位时间t_s内发送的最多数据包个数为c_i.

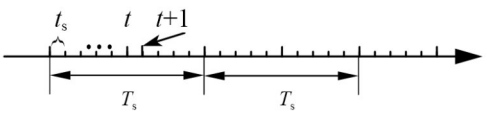


图1　数据包传输时间步

由以上假设可知,数据包到达目标节点后将被视为完成传输,不再计入数据包队列长度,即网络节点缓冲数据包个数,当多对多通信任务$\boldsymbol{M}$完成时可以认为网络中各个节点的队列长度为0,如果用ζ表示每个周期T_s中从数据开始传输到任务完成的时间,为了保证数据包不会因为超时而被丢弃,则会有$\zeta<T_s$成立. 由此可见网络节点的数据包队列长度是一个非常重要的网络状态指标,参考类似文献[22]的做法,假设用向量$\boldsymbol{p}(t)=[p_i(t)] \in \mathbb{N}^n$表示在周期$T_s$中第$t$个时间步各个网络节点数据包队列长度,其中$p_i(t)$表示网络节点$s_i$在时间步$t$数据包队列长度. 假设用$v_{ki}^{in}(t) \in \mathbb{N}$表示网络节点$s_i$在周期$T_s$中第$t$个时间步接收到来自另一网络节点$s_k$数据包数,用$v_{ij}^{out}(t) \in \mathbb{N}$表示网络节点$s_i$在周期$T_s$中第$t$个时间步发往网络节点$s_j$的数据包数,用$N_i^e$表示图$G$中网络节点$s_i$的所有邻居节点集合,则周期$T_s$中第$t$个时间步网络节点$s_i$接收的数据包个数$v_{\bullet i}^{in}(t) \in \mathbb{N}$为

$$v_{\bullet i}^{in}(t)=\sum_{s_k \in N_i^e} v_{ki}^{in}(t) \tag{2}$$

类似地,网络节点s_i在时间步t发送的数据包个数$v_{i\bullet}^{out}(t) \in \mathbb{N}$为

$$v_{i\bullet}^{out}(t)=\sum_{s_j \in N_i^e} v_{ij}^{out}(t) \tag{3}$$

根据上述假设规则,网络节点s_i在时间步t处理数据包个数$v_{i\bullet}^{out}(t)$应满足:

$$\begin{cases} v_{i\bullet}^{out}(t) \leqslant c_i, & p_i(t)>0 \\ v_{i\bullet}^{out}(t)=0, & p_i(t)=0 \end{cases} \tag{4}$$

其中,第一个式子表示网络节点s_i在时间步t存在数据包,发送的数据包个数不超过c_i;第二个式子表示网络节点s_i没有转发数据包. 因此时间步t从网络节点s_i发出的数据包个数满足如下约束:

$$0 \leqslant \sum_{s_j \in N_i^e} v_{ij}^{out}(t) \leqslant c_i \tag{5}$$

显然,对于非目标节点网络节点s_i的数据包队列长度有如下式子成立:

$$p_i(t+1)=p_i(t)+v_{\bullet i}^{in}(t)-v_{i\bullet}^{out}(t), s_i \notin N_d \tag{6}$$

对于目标网络节点s_i而言,时间步t到达的数据包括两部分,一部分是不以s_i为目标节点的数据包数$\bar{v}_{\bullet i}^{in}(t)$,另一部分以$s_i$为目标节点的数据包数$\overline{v}_{\bullet i}^{in}(t)$,则有

$$v_{\bullet i}^{in}(t)=\bar{v}_{\bullet i}^{in}(t)+\overline{v}_{\bullet i}^{in}(t) \tag{7}$$

由于到达目标的数据包将不再计入数据包队列长度,所以目标网络节点s_i中的数据包队列长度满足:

$$p_i(t+1)=p_i(t)+\bar{v}_{\bullet i}^{in}(t)-v_{\bullet i}^{out}(t), s_i \in N_d \tag{8}$$

特别地,对初始$t=0$时刻有

$$p_i(0)=\begin{cases}\sum_{s_j \in N_d} m_{ij}, & s_i \in N_s \\ 0, & s_i \notin N_s\end{cases} \tag{9}$$

其中,第一部分表示源节点的初始时间步数据包数等于需要从它传输到所有目标节点的数据包数量之和,第二部分表示非源节点上初始时间步是没有需要传输的数据包. 令$p_0=[p_i(0)] \in \mathbb{N}^n$表示网络节点上初始时间步的数据包队列长度.

多对多通信任务$\boldsymbol{M}$完成所用时间ζ就可以表示为

$$\zeta=\inf\{t>0, \boldsymbol{p}(t)=\boldsymbol{0}, p(0)=p_0\} \tag{10}$$

其中,$\boldsymbol{0}$是全0的列向量,$\inf\{t>0, \boldsymbol{p}(t)=\boldsymbol{0}, p(0)=p_0\}$表示

数据包从开始传输到网络节点中数据包队列长度为0所用的时间.

此外,网络节点能耗是描述网络性能的又一个重要指标,通常用节点累计能耗的均衡性来表示.假设用$e^{\mathrm{c}}(t)=[e_i^{\mathrm{c}}(t)]\in\mathbb{R}_+^n$表示在周期$T_{\mathrm{s}}$中第$t$个时间步时各网络节点的累计能耗,其中$e_i^{\mathrm{c}}(t)$表示网络节点$i$在周期$T_{\mathrm{s}}$中第$t$个时间步累计能耗,$\mathbb{R}$是实数集,$\mathbb{R}_+$表示非负实数集.本文这里讨论的网络节点能耗主要用于数据包的发送和接收.假设网络节点s_i发送和接收一个数据包的能耗分别表示为e_i^{t}和e_i^{r},则网络节点s_i在时间步t累计能耗可以表示为

$$e_i^{\mathrm{c}}(t)=\sum_{\tau=0}^{t}\left(v_{i\bullet}^{\mathrm{out}}(\tau)e_i^{\mathrm{t}}+v_{\bullet i}^{\mathrm{in}}(\tau)e_i^{\mathrm{r}}\right) \tag{11}$$

从式(2)~式(11)可以看出给定多对多任务$\boldsymbol{M}$的情况下,网络中各节点上的数据包队列长度只与变量$v_{ij}^{\mathrm{in}}(t)$和$v_{ij}^{\mathrm{out}}(t)$有关,但由于在时间步t内从网络节点s_i发往s_j的数据包数等于网络节点s_j收到来自网络节点s_i发来的数据包数,即$v_{ij}^{\mathrm{in}}(t)=v_{ij}^{\mathrm{out}}(t)$. 所以多对多通信的路由规划问题可以看成只是变量$v_{ij}^{\mathrm{out}}(t)$的决策优化问题.为了表述方便,结合式(10)和式(11),令

$$\begin{aligned}&f_1\left(v_{ij}^{\mathrm{out}}(t)\right)=\zeta=\inf\left\{t>0,\boldsymbol{p}(t)=\boldsymbol{0},p(0)=p_0\right\}\\&f_2\left(v_{ij}^{\mathrm{out}}(t)\right)=\mathrm{std}\left(\boldsymbol{e}^{\mathrm{c}}(\zeta)\right)\end{aligned} \tag{12}$$

其中,$\mathrm{std}\left(\boldsymbol{e}^{\mathrm{c}}(\zeta)\right)$表示向量$\boldsymbol{e}^{\mathrm{c}}(\zeta)$的标准差.

综合以上描述,对本文所讨论的多对多通信路由问题可以建立如下优化模型:在保证流量矩阵$\boldsymbol{M}$对应的任务尽快完成的同时,让每个网络节点上的累计能耗尽可能均衡.其中,针对流量矩阵$\boldsymbol{M}$对应的任务尽快完成的目标可以表示如下:

$$\min f_1\left(v_{ij}^{\mathrm{out}}(t)\right) \tag{13}$$

针对每个网络节点上的累计能耗尽可能均衡的目标可以描述如下:

$$\min f_2\left(v_{ij}^{\mathrm{out}}(t)\right) \tag{14}$$

由此得到本文对多对多通信路由决策问题建立的如下优化模型:

$$\begin{aligned}&\min\ \left[f_1\left(v_{ij}^{\mathrm{out}}(t),f_2\left(v_{ij}^{\mathrm{out}}(t)\right)\right)\right]\\&\text{s.t.}\quad 0\leqslant\sum_{s_j\in N_i^{\mathrm{s}}}v_{ij}^{\mathrm{out}}(t)\leqslant c_i,s_i\in N,\\&\qquad v_{ij}^{\mathrm{out}}(t)\in\mathbb{N},e_{ij}\in E,s_i,s_j\in N\end{aligned} \tag{15}$$

该优化模型是一个双目标离散优化问题,决策变量是$v_{ij}^{\mathrm{out}}(t)$,通过线性加权转换成如下单目标优化模型:

$$\begin{aligned}&\min f\left(v_{ij}^{\mathrm{out}}(t)\right)=\beta*f_1\left(v_{ij}^{\mathrm{out}}(t)\right)+(1-\beta)*f_2\left(v_{ij}^{\mathrm{out}}(t)\right)\\&\text{s.t.}\quad 0\leqslant\sum_{s_j\in N_i^{\mathrm{s}}}v_{ij}^{\mathrm{out}}(t)\leqslant c_i,s_i\in N,\\&\qquad v_{ij}^{\mathrm{out}}(t)\in\mathbb{N},e_{ij}\in E,s_i,s_j\in N\end{aligned} \tag{16}$$

其中,β是目标函数$f_1\left(v_{ij}^{\mathrm{out}}(t)\right)$和$f_2\left(v_{ij}^{\mathrm{out}}(t)\right)$的加权系数.第6.3节实验结果表明在实际应用中,可以通过调节β满足对不同目标函数的需求.

针对以上建立的多对多通信路由决策优化模型,为使求解得到的多对多通信路由规划能很好地适应网络状态高度动态变化的特点,本文首先设计了基于软件定义网络技术获取全局状态信息的系统架构,然后设计了基于多智能体图强化学习的智能求解方法.

4 SDWN中多对多通信路由系统架构

在本文多对多通信两阶段智能路由决策算法中,知识平面主要包括2个模块:智能求解方法中的强化学习模块和第二阶段的经验池模块.强化学习模块的主要功能是搭建强化学习的学习环境,将控制平面获得的全局和局部网络状态信息转化为状态信息,第一阶段设计的Q-学习与网络环境不断交互,让智能体朝着预想的方向学习;第二阶段设计多智能体图强化学习,先将控制平面获得的全局和局部信息转化为相应的形式存入经验池模块中,多智能体再从经验池中异步地获取数据进行训练,从而让多个智能体朝着最优的路由决策方向学习.当算法模型训练收敛后,给出当前网络状态下的网络参数,然后将参数下发给相应的智能AP节点,智能AP实时为处理的数据包指定转发策略.在上述的多智能体图强化学习中,经验池的数据包括由控制平面收集到的全局和局部状态,各智能体采用的动作以及相应的奖励和采取动作后的下一个状态,这些信息被整合成适当的数据类型,方便智能体采样训练.

设计的SDWN架构下对多对多路由调度问题的两阶段多智能求解方法的系统架构如图2所示,包括数据平面、控制平面和知识平面3个部分,其中流程步骤如下:①通过软件定义网络SDN架构中的Request和Packet_in消息分别获取全局和局部网络状态信息;②将采集的相关信息转化为便于智能体训练的数据类型,即图结构数据,将全局信息输入到第一阶段Q-学习中用于训练过程,将测量信息输入到第二阶段的多智能体图强化学习中,并将整合后的多智能体决策信息和相应的奖励存入经验池;③完成第一阶段训练所得的智能体信息传递;④模型部署模块将训练所得的智能体信息转化为数据平面AP能实现的部署指令,并将部署指令传递到数据平面;⑤完成第二阶段训练所得

智能体参数传递;⑥参数更新模块将训练所得的参数结果转化为智能AP能识别的参数类型,进一步参数传输给智能AP节点;⑦完成第一阶段的智能体训练,智能体直接根据控制平面通过北向接口传输上来的网络全局状态信息训练更新智能体,再将训练生成的智能体部署策略传输通过控制平面传递给数据平面;⑧完成第二阶段的多智能体的训练,多个智能体从经验池中小批量采样进行训练更新,再将训练生成的模型参数通过控制平面的参数更新模块将数据传输给数据平面;⑨两阶段训练结束后,数据平面智能AP的模型可以直接通过获取局部信息进行决策,此时不必再使用全局信息.

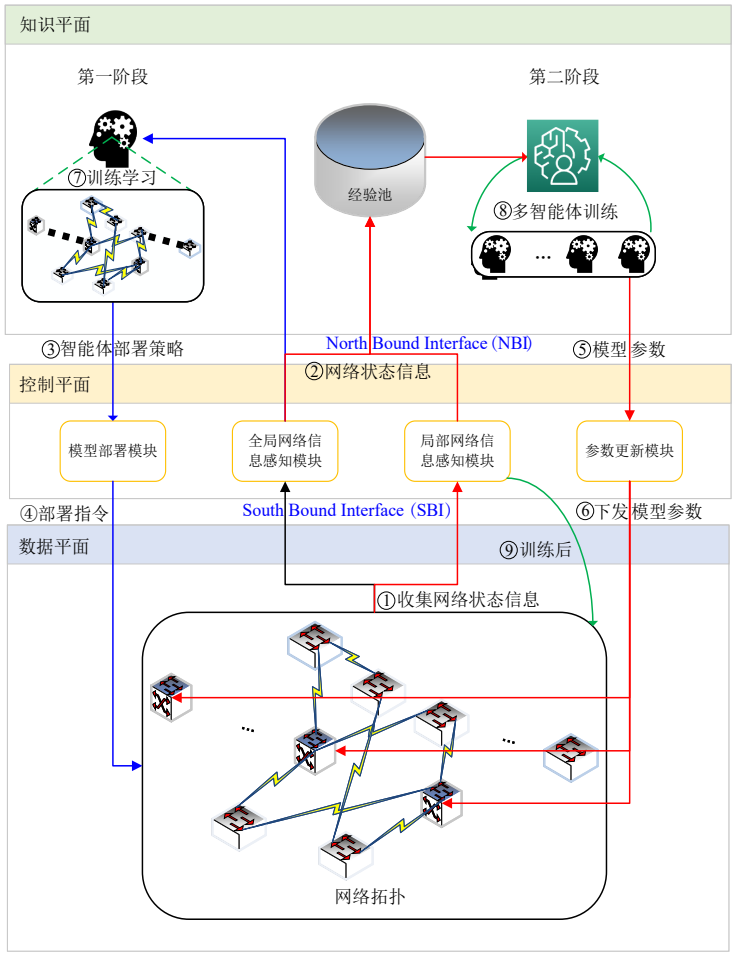


图2　SDWN中多对多通信路由结构

4.1　数据平面

数据平面根据流量矩阵定义的通信任务转发网络中的数据包,主要由无线接入节点(Access Point,AP)和站点(STAtion,STA)组成,整个网络中的AP组成无线多跳网络[32],AP节点即前面介绍的网络节点,在这些节点上需要考虑是否部署智能体,以方便规划多对多通信的路由决策,没有部署智能体的称为普通AP节点,反之称为智能AP节点.普通AP节点根据现有的最短路径算法获取数据包传输的最佳路径,智能AP节点通过智能体获取数据包传输的最佳路径.每个AP节点周期性与控制平面进行交互,将本地的数据包队列长度等信息传递给控制平面.

4.2　控制平面

控制平面负责收集和分析网络状态信息,制定决策,并将命令和策略下发到数据平面中的AP节点,通过北向接口与知识平面交互,方便知识平面策略的下发和部署.控制平面由全局网络信息感知模块、局部网络信息感知模块、模型部署模块和参数更新模块4个部分构成.

(1)全局信息感知模块.主要负责定期获取网络全局状态信息,包括每个AP节点上待处理数据包的队列长度以及累计能量使用情况,其中累计能量使用情况是通过节点中数据包变化间接计算获得的.具体地,全局信息感知模块定时给数据平面发送Request请求,假

设时隙time_t测量得某AP节点自身总接收的数据包数为pr_t,总发送的数据包数为pr_t,则该AP节点待处理的数据包数可以通过下式计算得到:

$$p_t=\text{pr}_t-\text{pt}_t \tag{17}$$

相应的累计能耗可以通过如下方式计算获取:

$$e_t=\text{pr}_t*e^{\text{r}}+\text{pt}_t*e^{\text{t}} \tag{18}$$

其中,e^{r}和e^{t}分别为AP节点接收和传输单位数据包所需的能耗.

(2)局部信息感知模块. 实时获取数据包所在前一个AP节点,当前数据包所在的位置和数据对应的目标节点位置等信息. 训练阶段智能AP将这些信息连同数据包本身通过Packet_in消息的形式上报给局部信息感知模块,局部信息感知模块进行解析并转化为图结构数据存入经验池,方便智能体获取这些信息进行训练.

(3)模型部署模块. 主要负责为数据平面配置相应的功能,分别为智能AP节点和非智能AP节点配置相应的参数信息. 第一阶段中,智能体训练后,根据状态信息将反馈智能AP和非智能AP的部署结果,模型部署模块根据这个结果为不同的AP发送相应的配置命令.

(4)参数更新模块. 主要负责将训练好的参数发送给智能AP节点. 第二阶段中,智能体训练后,各个智能体将自己的模型参数信息传递给参数更新模块,参数更新模块将这些参数传递给相应的智能AP,智能AP在收到这些参数后更新自身的参数,并通过新的参数控制数据包的传递.

4.3 知识平面

知识平面能够将强化学习智能体决策行为和推理过程集成到SDN网络架构中,提供描述、学习和智能化等功能,从而支持路由决策过程. 具体来说,除了完成模型的训练外,知识平面还可以整合来自控制平面的全局和局部状态信息,基于设计的强化学习算法,知识平面从这些状态信息中学习到网络动作行为,进一步将其转化为AP节点选择和多对多路由两方面的知识,然后智能体根据这些知识对当前网络信息做出相应的决策.

基于以上对知识平面、控制平面和数据平面中每个模块组成及功能的设计,设计的两阶段智能求解方法的训练过程可以简单描述如下:第一阶段中智能体与数据平面进行多次交互之后确定智能AP节点的部署,每次交互数据平面需要模拟完成一次流量矩阵$\boldsymbol{M}$对应的通信任务;而第二阶段的多个智能体在第一阶段智能体完成训练并部署智能体后开始训练,同样需要与数据平面进行多次交互. 不同的是,有了经验池后,智能体可以在数据采集的同时进行更新,数据平面智能AP节点的模型可以在完成一次流量矩阵$\boldsymbol{M}$对应的通信任务后再更新. 具体关于本文针对多对多路由优化问题设计的两阶段智能求解方法,包括对AP网络节点进行智能体部署的强化学习算法和确定多对多通信路由规划方案的多智能体图强化学习算法,相应的智能体状态空间、动作空间和奖励函数的设计将在本文的第5节进行详细描述.

5 两阶段智能多对多通信路由求解方法

考虑到优化模型中多对多路由决策问题的决策变量$v_{ij}^{\text{out}}(t)$具有高维、离散和时变等特点,属于NP难优化问题[33]. 传统的求解算法很难在多项式时间内完成该问题的求解,而且很难适应它高度动态变化的网络状态,对此本文设计两阶段求解方法MAGDS-M2M.

5.1 第一阶段:智能节点部署算法

如果对所有AP节点都部署智能体,随着AP节点规模增大会给路由策略求解的空间和时间复杂度以及部署成本的增加,因此考虑选取部分AP节点进行智能的部署,这就需要设计进行智能体部署的AP节点选择方案. 基于传统图论方法的节点选择方法很难适应网络状态高度动态变化的特性,因此本文设计了一种基于Q-学习的智能AP节点部署方法,设计过程具体描述如下.

首先,考虑到所有AP节点被划分为智能AP节点和普通AP节点2种情况:

$$N=N_{\text{im}}\bigcup N_{\text{un}},\ N_{\text{im}}\bigcap N_{\text{un}}=\varnothing \tag{19}$$

其中,N_{im}表示智能AP节点集合,用$n_{\text{im}}=\left|N_{\text{im}}\right|$表示智能AP节点的个数,用$n_{\text{un}}=\left|N_{\text{un}}\right|$表示普通AP节点的个数. 如果网络节点$s_i\in N_{\text{im}}$,它将使用智能体来确定$v_{ij}^{\text{out}}(t)$是否将数据包传输给下一个节点.

为了找到适宜智能AP节点,本文将智能AP节点集合N_{im}作为状态,所有可能的智能AP节点集合构成状态空间$S^{\text{no}}=\{S^{\text{no}}\}$,其中$S^{\text{no}}\subset N$表示智能AP节点的集合,用$n_{\text{sno}}=\left|S^{\text{no}}\right|$表示所有智能AP节点集合的个数. 因为节点要么是智能AP节点,要么是普通AP节点,所以$n_{\text{sno}}=2^n$.

对于智能AP节点集合,一步动作可以是删除一个节点也可以是添加一个节点,其中删除一个节点是将选定的节点划分到普通AP节点集合中;添加一个节点是选定的节点添加到智能AP节点集合中. 本文用一个$2n$维的独热向量空间$A^{\text{no}}=\{a_l^{\text{no}}\},l\in\{1,2,\cdots,2n\}$,$a_l^{\text{no}}=[a_{li}^{\text{no}}]\in\{0,1\}^{2n}$,且$a_{li}^{\text{no}}$满足:

$$a_{li}^{\text{no}}=\begin{cases}1, & i=l\\ 0, & \text{other}\end{cases} \tag{20}$$

则A^{no}的大小为$n_{\text{ano}}=\left|A^{\text{no}}\right|=2n$,需要注意的是,在确定的状态$S^{\text{no}}$下,如果$l\leqslant n$且$s_l\notin S^{\text{no}}$,$a_l^{\text{no}}$表示将$s_l$加入智能AP节点集合;当$l>n$时,记$l_n=\text{mod}(l,n)$,这里

$\mathrm{mod}(l,n)$表示l模n的运算,如果$s_{l_n}\in S^{\mathrm{no}}$,$a_l^{\mathrm{no}}$表示将$s_{l_n}$从智能AP节点集合中删除.

其他情况下的a_l^{no}均是无效动作. 这种表示法不是最简洁的,但可以避免出现不同状态下动作的物理含义发生改变的问题.

在多对多通信任务$\boldsymbol{M}$下,用D^{p}表示数据包集合,数据包个数满足$m^{\mathrm{p}}=\boldsymbol{I}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{I}=\left|D^{\mathrm{p}}\right|$,假设用$D_i^{\mathrm{p}}=\{d_{ij}^{\mathrm{p}}(t)\}\subset D^{\mathrm{p}}$表示在网络节点$s_i$上处理的数据包集合,其中$d_{ij}^{\mathrm{p}}(t)$表示时间步$t$网络节点$s_i$处理的第$j$个数据包,根据网络节点在一个时间步处理数据包的能力可知$j\leqslant c_i$.

定义数据包到其目的节点的映射des: $D^{\mathrm{p}}\mapsto N_{\mathrm{d}}$,这里的数据包传输算法分为2种:一种是随机简单路径传输算法,按照数据包到目的节点的所有简单路径,随机将数据包传输给下一个节点;另一种是随机最短路径传输算法,即按照数据包到目的节点的所有最短路径,随机将数据包传输给下一个节点. 用fph: $D_i^{\mathrm{p}}\mapsto\{\breve{N}_i^{\mathrm{e}}\}$表示根据随机简单路径传输算法数据包传输到下一个节点集合的映射,其中$\breve{N}_i^{\mathrm{e}}\subset N_i^{\mathrm{e}}$是网络节点$s_i$邻居节点的子集,表示数据包可以传递的下一个节点的集合;类似地,用sph: $D_i^{\mathrm{p}}\mapsto\{\breve{N}_i^{\mathrm{e}}\}$表示根据随机最短路径传输算法数据包传输到下一个节点集合的映射. 此外,为了讨论回路的产生,用映射pre: $D_i^{\mathrm{p}}\mapsto N_i^{\mathrm{e}}$表示数据包到其所经过的前一个节点的映射,显然数据包经过的前一个节点也是所在节点的邻居节点.

选择智能AP节点的目的是帮助这些节点更好地决策,需要定义时间步t智能AP节点s_i处理的数据包$d_{ij}^{\mathrm{p}}(t)$的奖励,根据数据包的不同,处理数据包的奖励包含下面2个部分:

$$r_{\mathrm{f}}^{\mathrm{no}}(d_{ij}^{\mathrm{p}}(t))=\begin{cases}1, & \left|\mathrm{fph}\left(d_{ij}^{\mathrm{p}}(t)\right)\Big/\mathrm{pre}\left(d_{ij}^{\mathrm{p}}(t)\right)\right|>1\\ 0, & \text{other}\end{cases}\tag{21}$$

$$r_{\mathrm{s}}^{\mathrm{no}}(d_{ij}^{\mathrm{p}}(t))=\begin{cases}1, & \left|\mathrm{sph}\left(d_{ij}^{\mathrm{p}}(t)\right)\Big/\mathrm{pre}\left(d_{ij}^{\mathrm{p}}(t)\right)\right|>1\\ 0, & \text{other}\end{cases}\tag{22}$$

其中,$\mathrm{fph}\left(d_{ij}^{\mathrm{p}}(t)\right)/\mathrm{pre}\left(d_{ij}^{\mathrm{p}}(t)\right)$和$\mathrm{sph}\left(d_{ij}^{\mathrm{p}}(t)\right)/\mathrm{pre}\left(d_{ij}^{\mathrm{p}}(t)\right)$表示在数据包可能的下一个节点中去掉它的前驱节点,以避免回路的产生. 传统的AP节点总是根据某个指标为来自相同端口具有相同目标的数据包给定下一跳,并且确定后不再改变,而智能节点可以根据当前的状态为数据包进行不同选择,这样的选择需要满足下一跳的选择不是唯一才有意义,即除了数据包之前所在的节点(保证不形成回路),其他选择的可能性大于1,数学表达为$\left|\mathrm{fph}\left(d_{ij}^{\mathrm{p}}(t)\right)/\mathrm{pre}\left(d_{ij}^{\mathrm{p}}(t)\right)\right|>1$和$\left|\mathrm{sph}\left(d_{ij}^{\mathrm{p}}(t)\right)/\mathrm{pre}\left(d_{ij}^{\mathrm{p}}(t)\right)\right|>1$,否则的话,部署智能体将和普通节点的效果一样. 式(21)、式(22)设计的原则如下:如果通过智能体选择有意义奖励1,否则奖励0. 这样的设计为第二阶段的多智能体的协作提供了可能,能保证第二阶段中智能体设置的有效性. $r_{\mathrm{f}}^{\mathrm{no}}\left(d_{ij}^{\mathrm{p}}(t)\right)$、$r_{\mathrm{s}}^{\mathrm{no}}\left(d_{ij}^{\mathrm{p}}(t)\right)$分别表示通过随机简单路径传输算法和随机最短路径传输算法在时间步t智能AP节点s_i处理的数据包$d_{ij}^{\mathrm{p}}(t)$的奖励,进一步加权获得时间步t智能AP节点s_i处理的数据包$d_{ij}^{\mathrm{p}}(t)$的奖励:

$$r^{\mathrm{no}}\left(d_{ij}^{\mathrm{p}}(t)\right)=\beta^{\mathrm{no}}*r_{\mathrm{f}}^{\mathrm{no}}\left(d_{ij}^{\mathrm{p}}(t)\right)+\left(1-\beta^{\mathrm{no}}\right)*r_{\mathrm{s}}^{\mathrm{no}}\left(d_{ij}^{\mathrm{p}}(t)\right)\tag{23}$$

其中,β^{no}表示2种传输算法的权重系数.

确定智能AP节点集合后获取的奖励函数R^{no}可以表示为

$$R^{\mathrm{no}}=\sum_{t=1}^{\zeta}\sum_{s_i\in N_{\mathrm{im}}}\sum_{j=1}^{c_i}r^{\mathrm{no}}\left(d_{ij}^{\mathrm{p}}(t)\right)\tag{24}$$

即所有重要节点在任务完成前产生的奖励之和,可以看出,奖励函数$R^{\mathrm{no}}\in\mathbb{R}^+$,奖励空间$R^{\mathrm{no}}=\left\{R^{\mathrm{no}}\right\}\subset\mathbb{R}^+$.

用$S^{\mathrm{no}}(k)$、$a_l^{\mathrm{no}}(k)$和$R^{\mathrm{no}}(k)$分别表示第k步的状态,动作和奖励值. 根据上述S^{no}、a_l^{no}定义很容易获取k+1步的状态$S^{\mathrm{no}}(k+1)$,即在集合$S^{\mathrm{no}}(k)$中添加或者删除某个节点. 据此,本文利用Q-学习算法求解最优的动作,用$Q_i^{\mathrm{no}}\left(S^{\mathrm{no}}(k),a_l^{\mathrm{no}}(k)\right)$表示第$i$次迭代在二元组$(S^{\mathrm{no}}(k),a_l^{\mathrm{no}}(k))$上的动作值函数,它的迭代更新过程为

$$\begin{aligned}Q_{i+1}^{\mathrm{no}}\left(S^{\mathrm{no}}(k),a_l^{\mathrm{no}}(k)\right)=&Q_i^{\mathrm{no}}\left(S^{\mathrm{no}}(k),a_l^{\mathrm{no}}(k)\right)+\alpha^{\mathrm{no}}\Big[R^{\mathrm{no}}(k)\\&+\gamma^{\mathrm{no}}\max_{a_l^{\mathrm{no}}}\left(Q_i^{\mathrm{no}}\left(S^{\mathrm{no}}(k+1),a_l^{\mathrm{no}}\right)\right.\\&\left.-Q_i^{\mathrm{no}}\left(S^{\mathrm{no}}(k),a_l^{\mathrm{no}}(k)\right)\right)\Big]\end{aligned}\tag{25}$$

其中,$\max_{a_l^{\mathrm{no}}}\left(Q_i^{\mathrm{no}}(S^{\mathrm{no}}(k+1),a_l^{\mathrm{no}}\right)$表示在状态$S^{\mathrm{no}}(k+1)$下,对所有可能的$a_l^{\mathrm{no}}(k+1)$取最大值. $0\leqslant\alpha^{\mathrm{no}}<1$是学习率,$0<\gamma^{\mathrm{no}}<1$是折扣因子,用于缩小未来估计的影响. 类似于文献[34]中的定理2可以保证这里设计的$Q_i^{\mathrm{no}}\left(S^{\mathrm{no}}(k),a_l^{\mathrm{no}}(k)\right)$的收敛的.

综合以上分析,设计的是智能AP节点部署算法如算法1所示. 其中第1行中$Q_0^{\mathrm{no}}(S^{\mathrm{no}},a_l^{\mathrm{no}})$表示所有可能状态和动作对应的动作值函数值. 第3~4行中初始化状态$S^{\mathrm{no}}(0)=\varnothing$,表示初始智能AP节点为空集,然后逐渐增加,直到智能AP节点数达到需要的个数n_{t}时,循环停止. 并进行下一轮的迭代. 第6行中根据当前的Q^{no}表,使用ϵ贪婪算法获取当前的$a_l^{\mathrm{no}}(k)$,具体公式表达为

$$a_l^{\mathrm{no}}(k)\leftarrow\begin{cases}\arg\max_{a_l^{\mathrm{no}}}\left(Q^{\mathrm{no}}(S^{\mathrm{no}}(k),a_l^{\mathrm{no}})\right), & r_{\mathrm{m}}\geqslant\epsilon^{\mathrm{no}}\\ \text{random action}, & r_{\mathrm{m}}<\epsilon^{\mathrm{no}}\end{cases}\tag{26}$$

其中,$0<\epsilon^{\mathrm{no}}<1$称为探索概率,$r_{\mathrm{m}}$是服从[0, 1]上的均匀分布随机数. 第7行中根动作$a_l^{\mathrm{no}}(k)$获取新的状态$S^{\mathrm{no}}(k+1)$,其具体的操作是根据动作$a_l^{\mathrm{no}}(k)$判断在智能

AP节点集合中添加还是删除节点，处理后的集合记为$S^{\mathrm{no}}(k+1)$.

算法1 智能AP节点部署算法

输入：多对多任务$\boldsymbol{M}$,学习率α^{no},折扣因子γ^{no},训练周期数E_1,重要节点需求个数n_{r}

输出：最佳Q^{no}表

1. 初始化$Q_0^{\mathrm{no}}(S^{\mathrm{no}},a_I^{\mathrm{no}})$
2. **for** 序列$e=1\rightarrow E_1$ **do**
3. 初始化状态$S^{\mathrm{no}}(0)=\varnothing$,时间步$k=0$
4. **while** $\left|S^{\mathrm{no}}(k)\right|<n_{\mathrm{r}}$ **do**
5. 根据当前的智能节点进行数据传输,根据式(21)~式(24)计算奖励$R^{\mathrm{no}}(k)$
6. 根据当前Q^{no}表,使用ϵ贪婪算法获取当前状态动作$\boldsymbol{a}_I^{\mathrm{no}}(k)$
7. 根据动作$a_I^{\mathrm{no}}(k)$和环境获取新的状态$S^{\mathrm{no}}(k+1)$
8. 根据式(25)更新Q^{no}表
9. $k=k+1$
10. **end while**
11. **end for**

在算法1中，外层循环的次数为E_1，内层循环的取决于智能AP节点个数n_{r}，需要遍历$\left|S^{\mathrm{no}}(k)\right|<n_r$的所有情况，最坏情况下，每次都要遍历所有未选择的节点$n-\left|S^{\mathrm{no}}(k)\right|$，内层循环的时间开销近似为$n_{\mathrm{r}}\times n$，所以算法1的计算开销近似为$O(E_1\times n_{\mathrm{r}}\times n)$. 内存上，需要维护状态-动作的$Q$表，算法1的内存开销近似为$O(n_{\mathrm{ano}}\times 2n)\approx O(n\times 2^n)$. 根据这个分析，使用Q-学习方法进行节点选择时，受到节点规模的限制可能变得收敛困难，在后面的实验中，本文就n=15的情况，初始设置$\epsilon^{\mathrm{no}}=0.99$，然后每隔50个迭代次数减半，直到探索率达到0.01后，保持不变，这样做增加了训练开始阶段智能体对未知环境的探索效率，训练过程中逐渐减少探索率，能在一定程度上提高收敛速率.

根据式(19)中的划分，本文将在智能AP节点集合N_{im}中的网络节点上部署智能体，使用智能体进行决策. 在普通AP节点集合N_{un}的网络节点上使用经典的随机最短路径传输算法. 所有智能AP节点上部署的智能体构成了多智能体结构.

5.2 第二阶段：多智能体图强化学习算法

在通过第一阶段智能体训练后，依据得到的智能AP节点部署策略在相应的AP网络节点上完成智能体的部署. 然而要确定多对多通信的路由决策方案，还需要对这些部署的智能体进行训练. 考虑到多对多通信中网络状态的高度动态变化特性，本文为多对多通信的路由决策设计了一种基于AC框架的多智能体图强化学习算法. 该算法为了减少模型训练的开销分别使用GCN和GNN设计了Actor和Critic的网络结构；此外，为降低模型训练的时间开销，该算法在Actor网络生成动作的过程中设计了一种局部观测方法来避免无效动作的生成，从而可以提高收敛速度，这保证了算法的可行性，设计过程具体描述如下.

首先需要注意的是，在优化模型(16)中，网络节点i要在周期T_{s}时间步t向网络节点s_j传输的数据包数$v_{ij}^{\mathrm{out}}(t)$，取决于网络节点s_i怎么处理这段时间中的每一个数据包. 为了帮助网络节点优化处理方案，在智能AP节点上部署智能体，用$I=\{I_1,I_2,\cdots,I_{n_{\mathrm{im}}}\}$表示智能体集合，其中$I_i$表示编号为$i$的智能体. 用cur: $I\mapsto N$表示智能体到所在节点编号的映射，用$\iota_i=\mathrm{cur}(I_i)$表示智能体$I_i$所在网络节点编号. 下面分别从状态空间、动作空间和奖励函数等方面介绍多智能体模型.

5.2.1 状态空间

假设节点上的数据包队列长度$p(t)$和累计使用的能量$e^{\mathrm{c}}(t)$在周期T_{s}时间步t到t+1内是不发生变化，所有智能体共同使用，称它们为全局变量. 用$S=\{S(t)\}$表示全局状态空间，其中：

$$S(t)=\left[p(t),e^{\mathrm{c}}(t)\right] \tag{27}$$

由于数据包到达智能体的顺序是随机的，所以训练时仅需考虑第一个数据包$d_{\iota_i 1}^{\mathrm{p}}(t)$即可，后面的数据用相同的模型做决策即可. 为了推导方便，后文将其简记为数据包$d_{\iota_i}^{\mathrm{p}}(t)$.

分别用$c_{\iota_i}^{\mathrm{p}}(t)=\left[c_{\iota_i k}^{\mathrm{p}}(t)\right]\in\{0,1\}^n$,$c_{\iota_i}^{\mathrm{c}}(t)=\left[c_{\iota_i k}^{\mathrm{c}}(t)\right]\in\{0,1\}^n$和$c_{\iota_i}^{\mathrm{d}}(t)=\left[c_{\iota_i k}^{\mathrm{d}}(t)\right]\in\{0,1\}^n$表示数据包$d_{\iota_i}^{\mathrm{p}}(t)$所在的前一个节点，当前数据包所在的节点和数据对应的目标节点的位置编码. 其中$c_{\iota_i k}^{\mathrm{p}}(t),c_{\iota_i k}^{\mathrm{c}}(t)$和$c_{\iota_i k}^{\mathrm{d}}(t)$分别满足：

$$c_{\iota_i k}^{\mathrm{p}}(t)=\begin{cases}1, & s_k=\mathrm{pre}\left(d_{\iota_i}^{\mathrm{p}}(t)\right)\\ 0, & \text{other}\end{cases} \tag{28}$$

$$c_{\iota_i k}^{\mathrm{c}}(t)=\begin{cases}1, & k=\iota_i\\ 0, & \text{other}\end{cases} \tag{29}$$

$$c_{\iota_i k}^{\mathrm{d}}(t)=\begin{cases}1, & s_k=\mathrm{des}\left(d_{\iota_i}^{\mathrm{p}}(t)\right)\\ 0, & \text{other}\end{cases} \tag{30}$$

由于$c_{\iota_i}^{\mathrm{p}}(t),c_{\iota_i}^{\mathrm{c}}(t)$和$c_{\iota_i}^{\mathrm{d}}(t)$这3个变量是数据包$d_{\iota_i}^{\mathrm{p}}(t)$独有的，故称它们为数据包$d_{\iota_i}^{\mathrm{p}}(t)$的局部变量.

用o_i: $\mathbb{R}^n\mapsto\mathbb{R}^n$表示智能体$I_i$对全局变量$S(t)$的局部观测映射. 通过局部映射，所有智能体状态空间可以表示为$S_1,S_2,\cdots,S_{n_{\mathrm{im}}}$，其中$S_i=\{S_i(t)\}$表示智能体$I_i$的状态空间，$S_i(t)$表示智能体$I_i$在处理数据包$d_{\iota_i}^{\mathrm{p}}(t)$时的状态，它可以表示为

$$\begin{aligned}S_i(t)=&\left[o_i\left(p(t)\right),o_i\left(e^{\mathrm{c}}(t)\right)c_{\iota_i}^{\mathrm{p}}(t),c_{\iota_i}^{\mathrm{c}}(t),c_{\iota_i}^{\mathrm{d}}(t)\right]\\ &\in\mathbb{R}^{n\times 2}\times\{0,1\}^{n\times 3}\end{aligned} \tag{31}$$

其中，$o_i(p(t))$和$o_i(e^{\mathrm{c}}(t))$分别表示智能体I_i对数据包数和累计能耗2个全局变量的局部观测，网络节点实际上是通过图卷积神经网络完成的.

5.2.2　动作空间

智能体的动作空间分别表示为$A_1, A_2, \cdots, A_{n_{\mathrm{im}}}$，其中$A_i=\{\boldsymbol{a}_{\iota_i}^{\mathrm{c}}(t)\}$表示智能体$I_i$的动作空间，$\boldsymbol{a}_{\iota_i}^{\mathrm{c}}(t)=[a_{\iota_i k}^{\mathrm{c}}]\in\{0,1\}^{\left|N_{\iota_i}^{\mathrm{c}}\right|+1}$，且$\boldsymbol{a}_{\iota_i}^{\mathrm{c}}(t)\boldsymbol{I}=1$，其中$\left|N_{\iota_i}^{\mathrm{c}}\right|$表示智能体$I_i$所在节点$\iota_i$的邻居节点个数，除邻居节点外，智能体可能决定将数据包在本地节点进行缓存，所以需要在邻居节点个数上加1. 换句话说，$\boldsymbol{a}_{\iota_i}^{\mathrm{c}}(t)$是独热编码向量，用于表示智能体$I_i$将数据包$d_{\iota_i}^{\mathrm{p}}(t)$传递给相应的邻居节点或者缓存在当前节点，可得状态空间A_i的大小为$|A_i|=\left|N_{\iota_i}^{\mathrm{c}}\right|+1$.

5.2.3　奖励函数

用$\mathrm{env}: S\times A_1\times\cdots\times A_{n_{\mathrm{im}}}\mapsto S$表示环境在各智能体的作用下环境的全局状态变化. 在此基础上，给定环境的初始状态$S(0)$，可以利用Actor网络对环境进行采样，获取样本序列：

$$\begin{aligned}\mathrm{Tr}_{1:T}=S(0),a_1^{\mathrm{c}}(0),\cdots,a_{n_{\mathrm{im}}}^{\mathrm{c}}(0),\cdots,\\ S(T-1),a_1^{\mathrm{c}}(T-1),\cdots,a_{n_{\mathrm{im}}}^{\mathrm{c}}(T-1),S(T)\end{aligned}\tag{32}$$

上述序列中状态$S(t)$仅仅表示全局状态，针对不同的智能体I_i需要进行相应的局部观测. 此外，结束时间步T有2种可能：一种是网络在时间步T时所有数据包都到达了相应的目标节点；另一种是到达最大采样时间步T_{m}，即$T=T_{\mathrm{m}}$.

针对样本序列$\mathrm{Tr}_{1:T}$的片段，$S(t),a_1^{\mathrm{c}}(t),\cdots,a_{n_{\mathrm{im}}}^{\mathrm{c}}(t),S(t+1)$，其中：

$$S(t+1)=\mathrm{env}\left(S(t),a_1^{\mathrm{c}}(t),\cdots,a_{n_{\mathrm{im}}}^{\mathrm{c}}(t)\right)\tag{33}$$

通过环境获得，类似于式(27)有：

$$S(t+1)=\left[p(t+1),e^{\mathrm{c}}(t+1)\right]\tag{34}$$

针对目标函数$f_1(v_{\iota_i j}^{\mathrm{out}}(t))$，设计智能体$I_i$的单步奖励函数为

$$-R_1^i(t)=\frac{\mathrm{sum}(p(t))+\mathrm{sum}(p(t+1))}{2\kappa}\in[0,1]\tag{35}$$

注意到这个奖励与智能体编号无关，即每个智能体在同一时间步处理数据包获得的奖励是相同的，区别在于智能体是否处理数据包. 另外，参考了文献[35]中的表示方法，在奖励函数前面添加负号，表明这是由奖励形式给出的成本. 注意到目标函数$f_1\left(v_{\iota_i j}^{\mathrm{out}}(t)\right)$是期望所有AP节点上的数据包队列尽快减少，对所有智能体而言，一次决策后剩余的数据包数越少越好，所以式(35)将决策前后数据包数的平均值作为奖励，除以κ是为了将其统一到[0, 1]内. 从这个奖励可以看出，随着网络中数据包的逐渐减少，单步处理数据的成本也逐渐减小.

针对目标函数$f_2(v_{ij}^{\mathrm{out}}(t))$，设计单步奖励为

$$\begin{aligned}-R_2^i(t)=&\frac{\mathrm{std}\left(e^{\mathrm{c}}(t)\right)}{2\left(\max\left(e^{\mathrm{c}}(t)\right)-\min\left(e^{\mathrm{c}}(t)\right)\right)}\\&+\frac{\mathrm{std}\left(e^{\mathrm{c}}(t+1)\right)}{2\left(\max\left(e^{\mathrm{c}}(t+1)\right)-\min\left(e^{\mathrm{c}}(t+1)\right)\right)}\in[0,1]\end{aligned}\tag{36}$$

其中，$\max(\cdot),\min(\cdot),\mathrm{std}(\cdot)$分别表示对应向量的最大值，最小值和标准差. 注意到目标函数$f_2\left(v_{ij}^{\mathrm{out}}(t)\right)$是期望决策后每个AP节点累计能耗尽可能均衡，对所有智能体而言，一次决策后所有AP节点上累计能耗的标准差越小越好，所以式(36)将决策前后累计能耗标准差平均值作为奖励，分别除以$\max\left(e^{\mathrm{c}}(t)\right)-\min\left(e^{\mathrm{c}}(t)\right)$和$\left(\max\left(e^{\mathrm{c}}(t+1)\right)-\min\left(e^{\mathrm{c}}(t+1)\right)\right)$是为了将其归一化到[0, 1]内. 从这个奖励可以看出，如果各节点的累计能耗差异越大，单步处理数据的成本也会越大.

奖励函数$R_i(t)$除了包含$R_1^i(t)$和$R_2^i(t)$外，为了避免回路出现，本文还做了回路惩罚，具体表达式如下：

$$\begin{aligned}-R_i(t)=&-\beta*R_1^i(t)-(1-\beta)*R_2^i(t)\\&+C_{\mathrm{h}}*I\left(\mathrm{oa}_i^{-1}\left(\boldsymbol{a}_{\iota_i}^{\mathrm{c}}(t)\right)=\boldsymbol{c}_{\iota_i}^{\mathrm{p}}(t)\right)\end{aligned}\tag{37}$$

其中，β与式(16)中保持一致，C_{h}是一个大于0的常数，$I(\cdot)$是示性函数，即

$$I\left(\mathrm{oa}_i^{-1}\left(\boldsymbol{a}_{\iota_i}^{\mathrm{c}}(t)\right)=c_{\iota_i}^{\mathrm{p}}(t)\right)=\begin{cases}1, & \mathrm{oa}_i^{-1}\left(\boldsymbol{a}_{\iota_i}^{\mathrm{c}}(t)\right)=c_{\iota_i}^{\mathrm{p}}(t)\\0, & \mathrm{other}\end{cases}\tag{38}$$

MAGDS-M2M模型训练过程中将序列$S(t),a_1^{\mathrm{c}}(t),\cdots,a_{n_{\mathrm{im}}}^{\mathrm{c}}(t),S(t+1)$加上奖励函数值存入经验池$\Re$中，即存入经验的池$\Re$的样本如下：$S(t),a_1^{\mathrm{c}}(t),\cdots,a_{n_{\mathrm{im}}}^{\mathrm{c}}(t),R_1(t),\cdots,R_{n_{\mathrm{im}}}(t),S(t+1)$.

5.2.4　Actor与Critic的网络结构设计和动作空间中局部观测方法的设计

由于设计强化学习算法是在AC框架下实现的，要通过智能体间的合作与竞争有效地解决多对多通信的路由决策问题，离不开Actor和Critic网络结构的合理设计. 此外，为了避免无效动作产生，本节还针对动作空间设计了局部观测方法.

首先，为了适应网络拓扑数据，本文采用图卷积神经网络作为Actor网络，对于智能体I_i，它可以表示为

$$\mathrm{GCN}_{\theta_i}: S_i\mapsto[0,1]^n\tag{39}$$

其中，θ_i表示智能体I_i的Actor网络可学习参数. 特别地，对于数据包$d_{\iota_i}^{\mathrm{p}}(t)$，可以通过$\mathrm{GCN}_{\theta_i}$获取智能体$I_i$对

它的处理动作，具体表示为

$$\begin{aligned}&\tilde{a}^{\mathrm{c}}_{\iota_i}(t)=\mathrm{GCN}_{\theta_i}\left(S_i(t),A\right),\\&\boldsymbol{a}^{\mathrm{c}}_{\iota_i}(t)=\mathrm{gbs}_i\left(\mathrm{oa}_i\left(\tilde{a}^{\mathrm{c}}_{\iota_i}(t)\right)\right)\end{aligned}\tag{40}$$

其中，$\tilde{a}^{\mathrm{c}}_{\iota_i}(t)\in[0,1]^n$ 是图神经网络的输出结果，$\mathrm{oa}_i:[0,1]^n\mapsto[0,1]^{\left|N^{\mathrm{c}}_{\iota_i}\right|+1}$ 是对动作空间的局部观测映射，用 $\boldsymbol{H}^i=[h^i_{kl}]\in\{0,1\}^{\left(\left|N^{\mathrm{c}}_{\iota_i}\right|+1\right)\times n}$ 表示智能体 I_i 的局部观测矩阵，这里的 k 表示节点 ι_i 的邻居节点在 N 中出现的顺序编号，它在对应 N 中的位置用 ς_k 表示，其中 h^i_{kl} 满足

$$h^i_{kl}=\begin{cases}1, & l=\varsigma_k\\0, & \text{other}\end{cases}\tag{41}$$

这里的动作空间的局部映射可以表示为

$$\mathrm{oa}_i\left(\tilde{a}^{\mathrm{c}}_{\iota_i}(t)\right)=\tilde{a}^{\mathrm{c}}_{\iota_i}(t)*\boldsymbol{H}^{i\mathrm{T}}\tag{42}$$

其中，$\boldsymbol{H}^{i\mathrm{T}}$ 表示矩阵 $\boldsymbol{H}^i$ 的转置. $\mathrm{gbs}_i:[0,1]^{\left|N^{\mathrm{c}}_{\iota_i}\right|+1}\mapsto\{0,1\}^{\left|N^{\mathrm{c}}_{\iota_i}\right|+1}$ 为文献[36]中提出的gumbel softmax采样映射方式，本文设计的映射方式 $\mathrm{oa}_i(\bullet)$ 和文献[36]中的gumbel softmax采样映射方式 $\mathrm{gbs}_i(\bullet)$ 均满足梯度可回传的性质，它们的链式梯度均为1. 此外，这里动作空间的局部观测方法避免了无效动作的生成，解决了固定长度动作的收敛问题.

相对于Actor网络，Critic网络需要拟合中心化的动作值函数，图卷积对函数的拟合能力很有限，故除了图卷积层外还有全连接层，这样的网络称为图神经网络. 对于智能体 I_i，用 $L^{\mathrm{s}}_i=\{l^{\mathrm{s}}_i(t)\}$ 表示它的局部状态空间，其中

$$l^{\mathrm{s}}_i(t)=\left[c^{\mathrm{p}}_{\iota_i}(t),c^{\mathrm{c}}_{\iota_i}(t),c^{\mathrm{d}}_{\iota_i}(t),\mathrm{oa}^{-1}_i\left(\boldsymbol{a}^{\mathrm{c}}_{\iota_i}(t)\right)\right]\tag{43}$$

其中，$\mathrm{oa}^{-1}_i:\{0,1\}^{\left|N^{\mathrm{c}}_{\iota_i}\right|+1}\mapsto[0,1]^n$ 表示动作空间局部观测的逆映射，其具体计算为

$$\mathrm{oa}^{-1}_i\left(\boldsymbol{a}^{\mathrm{c}}_{\iota_i}(t)\right)=\boldsymbol{a}^{\mathrm{c}}_{\iota_i}(t)*\boldsymbol{H}^i\tag{44}$$

根据映射 $\mathrm{oa}_i(\cdot)$ 的定义，$\mathrm{oa}^{-1}_i(\cdot)$ 同样满足梯度可回传的性质，且链式梯度也为1，$l^{\mathrm{s}}_i(t)$ 表示智能体 I_i 对数据包 $d^{\mathrm{p}}_{\iota_i}(t)$ 的局部观测状态. 智能体 I_i 的Critic网络可以表示为

$$\mathrm{GNN}_{\phi_i}:S\times L^{\mathrm{s}}_1\times\cdots\times L^{\mathrm{s}}_{n_{\mathrm{im}}}\mapsto\mathbb{R}\tag{45}$$

其中，ϕ_i 是智能体 I_i 的Critic网络参数，GNN_{ϕ_i} 的目的是学习中心化的动作值函数的值，所以它的输出是一个实数. 智能体 I_i 处理数据包 $d^{\mathrm{p}}_{\iota_i}(t)$ 时

$$\begin{aligned}&\tilde{a}^{\mathrm{c}}_{\iota_i}(t)=\mathrm{GCN}_{\theta_i}\left(S_i(t),A\right),\forall I_i\in I,\\&a^{\mathrm{c}}_{\iota_k}(t)=\mathrm{gbs}_k\left(\mathrm{oa}_k\left(\tilde{a}^{\mathrm{c}}_{\iota_k}(t)\right)\right),I_k\in I,k=i,\\&a^{\mathrm{c}}_{\iota_k}(t)=\mathrm{sfm}_k\left(\mathrm{oa}_k\left(\tilde{a}^{\mathrm{c}}_{\iota_k}(t)\right)\right),I_k\in I,k\neq i,\\&l^{\mathrm{s}}_i(t)=\left[c^{\mathrm{p}}_{\iota_i}(t),c^{\mathrm{c}}_{\iota_i}(t),c^{\mathrm{d}}_{\iota_i}(t),\mathrm{oa}^{-1}_i\left(\boldsymbol{a}^{\mathrm{c}}_{\iota_i}(t)\right)\right],\forall I_i\in I\end{aligned}\tag{46}$$

对于智能体 I_i 有

$$\begin{aligned}&Q_i\left(S(t),l^{\mathrm{s}}_1(t),\cdots,l^{\mathrm{s}}_{n_{\mathrm{im}}}(t)\right)\\&\quad=\mathrm{GNN}_{\phi_i}\left(\left[S(t),l^{\mathrm{s}}_1(t),\cdots,l^{\mathrm{s}}_{n_{\mathrm{im}}}(t)\right],A\right)\end{aligned}\tag{47}$$

其中，$\mathrm{sfm}_k:[0,1]^{\left|N^{\mathrm{c}}_{\iota_i}\right|+1}\mapsto\{0,1\}^{\left|N^{\mathrm{c}}_{\iota_i}\right|+1}$ 是softmax映射，和gumbel softmax映射不一样，它不考虑梯度可回传，$Q_i\left(S(t),l^{\mathrm{s}}_1(t),\cdots,l^{\mathrm{s}}_{n_{\mathrm{im}}}(t)\right)$ 是智能体 I_i 的中心化的动作值函数.

为了解决过高估计的问题，引入目标网络的设计，将智能体 I_i 的Actor目标网络表示如下：

$$\mathrm{GCN}_{\theta^-_i}:S_i\mapsto[0,1]^n\tag{48}$$

其中，θ^-_i 是智能体 I_i 的Actor目标网络的参数，它与Actor网络的参数仅仅数值上不同，结构上一致.

类似地，将智能体 I_i 的Critic目标网络表示为

$$\mathrm{GNN}_{\phi^-_i}:S\times L^{\mathrm{s}}_1\times\cdots\times L^{\mathrm{s}}_{n_{\mathrm{im}}}\mapsto\mathbb{R}\tag{49}$$

其中，ϕ^-_i 是智能体 I_i 的Critic目标网络的参数. 它和Critic网络也仅仅是参数不同.

对于数据包 $d^{\mathrm{p}}_{\iota_i}(t+1)$，类似于 $Q_i\left(S(t),l^{\mathrm{s}}_1(t),\cdots,l^{\mathrm{s}}_{n_{\mathrm{im}}}(t)\right)$ 的计算有

$$\begin{aligned}&S_i(t+1)=\Big[o_i\left(p(t+1)\right),o_i\left(e^{\mathrm{c}}(t+1)\right),c^{\mathrm{p}}_{\iota_i}(t+1),\\&\qquad c^{\mathrm{c}}_{\iota_i}(t+1),c^{\mathrm{d}}_{\iota_i}(t+1)\Big]\in\mathbb{R}^{n\times2}\times\{0,1\}^{n\times3},\\&\tilde{a}^{\mathrm{c}^-}_{\iota_i}(t+1)=\mathrm{GCN}_{\theta^-_i}\left(S_i(t+1),A\right),\forall I_i\in I,\\&a^{\mathrm{c}^-}_{\iota_k}(t+1)=\mathrm{gbs}_k\left(\mathrm{oa}_k\left(\tilde{a}^{\mathrm{c}^-}_{\iota_k}(t+1)\right)\right),I_k\in I,k=i,\\&a^{\mathrm{c}^-}_{\iota_k}(t+1)=\mathrm{sfm}_k\left(\mathrm{oa}_k\left(\tilde{a}^{\mathrm{c}^-}_{\iota_k}(t+1)\right)\right),I_k\in I,k\neq i,\\&l^{\mathrm{s}^-}_i(t+1)=\Big[c^{\mathrm{p}}_{\iota_i}(t+1),c^{\mathrm{c}}_{\iota_i}(t+1),c^{\mathrm{d}}_{\iota_i}(t+1),\\&\qquad\mathrm{oa}^{-1}_i\left(a^{\mathrm{c}^-}_{\iota_i}(t+1)\right)\Big],\forall I_i\in I\end{aligned}\tag{50}$$

智能体 I_i 目标网络表示的中心化的动作值函数为

$$\begin{aligned}&\mathrm{Q}_i\left(S(t),l^{\mathrm{s}^-}_1(t),\cdots,l^{\mathrm{s}^-}_{n_{\mathrm{im}}}(t)\right)\\&\leftarrow\mathrm{GNN}_{\phi_i}\left(\left[S(t+1),l^{\mathrm{s}^-}_1(t+1),\cdots,l^{\mathrm{s}^-}_{n_{\mathrm{im}}}(t+1)\right],A\right)\end{aligned}\tag{51}$$

基于以上设计，智能体间使用CTDE的多智能训练体范式，具体多智能体间的协作与竞争数据流向如图3所示.

Actor和Critic的网络结构具体设计如图4所示，①~④是Actor网络部分，其整体上是一个图卷积网络. ①包含了全局状态和局部状态，虚线边框表示全局状态，分别是节点上的数据包队列长度 $p(t)$ 和节点上累计能耗 $e^{\mathrm{c}}(t)$；②是Actor网络的第一个卷积层；③是Actor网络的第二个卷积层，需要注意在智能数据平面这些卷积层不需要直接获取全局状态，因为每个节点只用到有限的局部节点的信息，可以通过获取邻居节点的

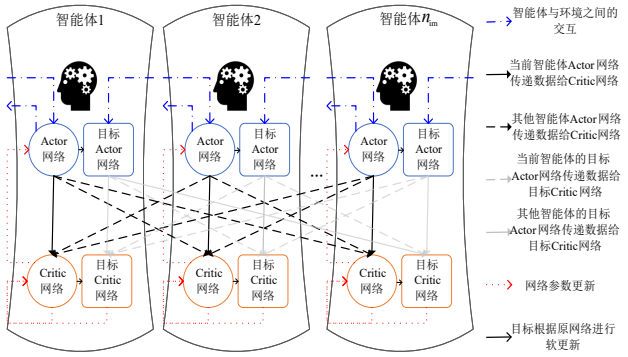


图3　多智能体间的数据流向

信息完成;④说明卷积后的结果是一张图.

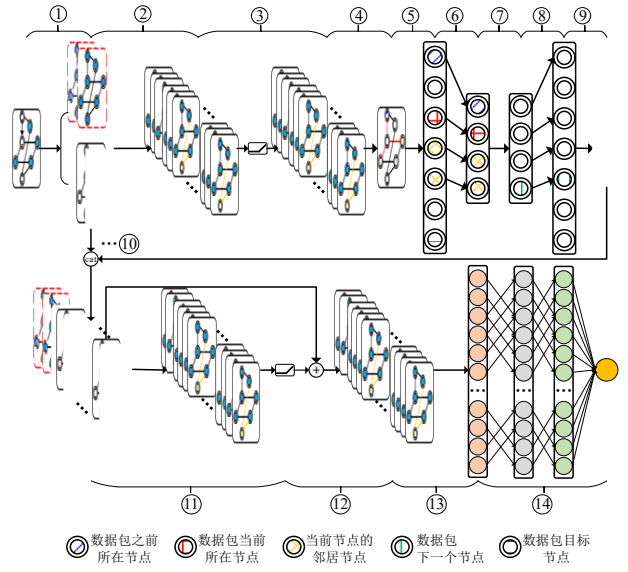


图4　Actor网络和Critic网络设计

图4中⑤~⑨实现了动作空间的局部观测,首先⑤将图按节点顺序进行展开,⑥通过映射 $\mathrm{oa}_i(\cdot)$ 实现对邻居节点的局部观测,⑦通过映射 $\mathrm{gbs}_i(\cdot)$ 获取对应的动作编码,然后⑧通过映射 $\mathrm{oa}_i^{-1}(\cdot)$ 将动作编码映射到所有节点,最后⑨按节点还原图结构. 这里需要注意的是,⑧中的动作如果指向数据包之前所在的节点,计算奖励时将给予惩罚 C_h.

需要注意的是,图4中仅画出了一个智能体的情况,⑩处省略号指的是省略了其他智能体的数据交互,除全局状态外,其他智能体将提供它们的局部观测以及通过映射 $\mathrm{sfm}_k(\cdot)$ 获得的动作,这样Critic的输入通道数为 $4n_{\mathrm{im}}+2$,其中包含2个全局状态通道,每个智能体提供3个局部状态通道和1个动作通道. ⑪~⑭表示Critic网络的实现,其整体上是一个图神经网络,其中⑪是一个卷积层;⑫将卷积层生成的特征和原特征进行1:1求和,获得新的特征;⑬将通道上的特征进行求和,方便设计线性层;⑭是线性层,最终获得一个值,用来拟合中心化的动作值函数值.

5.2.5　Actor与Critic网络的参数更新过程设计

在上一节网络结构设计下,Actor与Critic网络的参数更新过程会发生相应的变化,因此这里需要对参数的更新过程进行相应的说明. 具体地,智能体从经验池 $\Re$ 中进行小批量 b_s 采样,用 $\left[S(t),a_1^{\mathrm{c}}(t),R_1(t),\cdots,a_{n_{\mathrm{im}}}^{\mathrm{c}}(t),R_{n_{\mathrm{im}}}(t),S(t+1)\right]^{b_s}$ 表示批量为 b_s 的样本,用 $S\sim \mathrm{P}^{\boldsymbol{\mu}}$ 表示样本状态 S 的分布,在原网络和目标网络下分别用 $\boldsymbol{\mu}=[\mu_1,\mu_2,\cdots,\mu_{n_{\mathrm{im}}}]$ 和 $\boldsymbol{\mu}'=[\mu_1',\mu_2',\cdots,\mu_{n_{\mathrm{im}}}']$ 表示样本中 S 到动作函数的执行,将式(46)和式(50)中的动作计算过程分别表示为 $a_i=\mu_i(S)$ 和 $a_i'=\mu_i'(S)$. 在此表示下,批量为 b_s 样本的中心化动作值函数表示为 $Q_i^{\boldsymbol{\mu}}(S,a_1,\cdots,a_{n_{\mathrm{im}}})$,目标网络表示下的批量样本的中心化动作值函数表示为 $Q_i^{\boldsymbol{\mu}'}(S,a_1',\cdots,a_{n_{\mathrm{im}}}')$,智能体 I_i 的奖励表示为 $R_i(S)$.

用 $\varepsilon_i(S;\theta_1,\cdots,\theta_{n_{\mathrm{im}}},\phi_i)$ 表示智能体 I_i 在批量样本下Critic网络的损失函数,根据时序差分的方法[37],可以将其表示如下:

$$\begin{aligned}&\varepsilon_i(S;\theta_1,\cdots,\theta_{n_{\mathrm{im}}},\phi_i)\\&=\mathbb{E}_{S\sim \mathrm{P}^{\boldsymbol{\mu}}}\Big[\Big(Q_i(S,a_1,\cdots,a_{n_{\mathrm{im}}})-\big(R_i(S)\\&\quad+\gamma^{*}Q_i^{\boldsymbol{\mu}'}(S,a_1',\cdots,a_{n_{\mathrm{im}}}')\big)\Big)^2\Big|_{a_i=\mu_i(S),a_i'=\mu_i'(S)}\Big]\end{aligned}\tag{52}$$

其中,$0<\gamma<1$ 是衰减因子,它体现的是后面中心化的动作值函数对当前的影响.

$\varepsilon_i(S;\theta_1,\cdots,\theta_{n_{\mathrm{im}}},\phi_i)$ 关于 ϕ_i 的梯度更新为

$$\begin{aligned}&\nabla_{\phi_i}\varepsilon_i(S;\theta_1,\cdots,\theta_{n_{\mathrm{im}}},\phi_i)\\&=\nabla_{\phi_i}\mathbb{E}_{\boldsymbol{S}\sim \mathrm{P}^{\boldsymbol{\mu}}}\Big[\Big(Q_i(S,a_1,\cdots,a_{n_{\mathrm{im}}})-\big(R_i(S)\\&\quad+\gamma^{*}Q_i^{\boldsymbol{\mu}'}(S,a_1',\cdots,a_{n_{\mathrm{im}}}')\big)\Big)^2\Big|_{\boldsymbol{a}_i=\mu_i(\boldsymbol{S}),\boldsymbol{a}_i'=\mu_i'(\boldsymbol{S})}\Big]\\&=\mathbb{E}_{\boldsymbol{S}\sim \mathrm{P}^{\boldsymbol{\mu}}}\Big[2^{*}\Big(Q_i(S,a_1,\cdots,a_{n_{\mathrm{im}}})\\&\quad-\big(R_i(S)+\gamma^{*}Q_i^{\boldsymbol{\mu}'}(S,a_1',\cdots,a_{n_{\mathrm{im}}}')\big)\Big)\\&\quad^{*}\nabla_{\phi_i}Q_i\Big(S,a_1,\cdots,a_{n_{\mathrm{im}}}\Big)\Big|_{a_i=\mu_i(S),a_i'=\mu_i'(S)}\Big]\\&=\mathbb{E}_{\boldsymbol{S}\sim \mathrm{P}^{\boldsymbol{\mu}}}\Big[2^{*}\Big(Q_i(S,a_1,\cdots,a_{n_{\mathrm{im}}})-\big(R_i(S)\\&\quad+\gamma^{*}Q_i^{\boldsymbol{\mu}'}(S,a_1',\cdots,a_{n_{\mathrm{im}}}')\big)\Big)^{*}\nabla_{\phi_i}\mathrm{GNN}_{\phi_i}\big([S,l_1^{\mathrm{s}},\\&\quad\cdots,l_{\mathrm{n_{im}}}^{\mathrm{s}}],A\big)\Big|_{a_i=\mu_i(S),a_i'=\mu_i'(S)}\Big]\end{aligned}\tag{53}$$

其中,$\mathrm{GNN}_{\phi_i}\Big(\Big[S,l_1^{\mathrm{s}},\cdots,l_{n_{\mathrm{im}}}^{\mathrm{s}}\Big],A\Big)$ 是 $\mathrm{GNN}_{\phi_i}\Big(\Big[S(t),l_1^{\mathrm{s}}(t),\cdots,l_{n_{\mathrm{im}}}^{\mathrm{s}}(t)\Big],A\Big)$ 在批量样本下的表示.

用 $\eta_i(S;\theta_1,\cdots,\theta_{n_{\mathrm{im}}},\phi_i)$ 表示智能体 I_i 在批量样本下

Actor网络的损失函数，和文献[38]使用正则化处理方式类似，对$\eta_i(S;\theta_1,\cdots,\theta_{n_{\text{im}}},\phi_i)$的设计为如下方式：

$$\begin{aligned}&\eta_i(S;\theta_1,\cdots,\theta_{n_{\text{im}}},\phi_i)\\&=\mathbb{E}_{S\sim\mathrm{P}^{\boldsymbol{\mu}}}\left[\left(Q_i^{\boldsymbol{\mu}}\left(S,a_1,\cdots,a_{n_{\text{im}}}\right)+\lambda\left\|a_i\right\|^2\right)\Big|_{a_i=\mu_i(S)}\right]\end{aligned}\tag{54}$$

其中，$\lambda\left\|a_i\right\|^2$项是动作$a_i$的正则化，$\lambda>0$是正则化系数.

根据映射$\text{oa}_i,\text{oa}_i^{-1},\text{gbs}_i$保持梯度可传递且链式梯度为1，根据链式法则可以计算$\eta_i(S;\theta_1,\cdots,\theta_{n_{\text{im}}},\phi_i)$关于$\theta_i$的梯度为

$$\begin{aligned}&\nabla_{\theta_i}\eta_i(S;\theta_1,\cdots,\theta_{n_{\text{im}}},\phi_i)\\&=\nabla_{\theta_i}\mathbb{E}_{S\sim\mathrm{P}^{\boldsymbol{\mu}}}\left[\left(Q_i^{\boldsymbol{\mu}}(S,a_1,\cdots,a_{n_{\text{im}}})+\lambda\left\|a_i\right\|^2\right)\Big|_{a_i=\mu_i(S)}\right]\\&=\mathbb{E}_{S\sim\mathrm{P}^{\boldsymbol{\mu}}}\left[\left(\nabla_{a_i}Q_i^{\boldsymbol{\mu}}(S,a_1,\cdots,a_{n_{\text{im}}})\right.\right.\\&\quad\left.\left.+2*\lambda\left\|a_i\right\|\right)\nabla_{\theta_i}a_i\Big|_{a_i=\mu_i(S)}\right]\\&=\mathbb{E}_{S\sim\mathrm{P}^{\boldsymbol{\mu}}}\left[\left(\nabla_{a_i}\text{GNN}_{\phi_i}\left(\left[S,l_1^{\text{s}},\cdots,l_{n_{\text{im}}}^{\text{s}}\right],A\right)+2*\lambda\left\|a_i\right\|\right)\right.\\&\quad\left.*\nabla_{\theta_i}a_i\Big|_{a_i=\mu_i(S)}\right]\\&=\mathbb{E}_{S\sim\mathrm{P}^{\boldsymbol{\mu}}}\left[\left(\nabla_{a_i}\text{GNN}_{\phi_i}\left(\left[S,l_1^{\text{s}},\cdots,l_{n_{\text{im}}}^{\text{s}}\right],A\right)+2*\lambda\left\|a_i\right\|\right)\right.\\&\quad\left.*\nabla_{\tilde{a}_i}\text{oa}_i^{-1}\left(\text{gbs}_i\left(\text{oa}_i(\tilde{a}_i)\right)\right)*\nabla_{\theta_i}\tilde{a}_i\Big|_{a_i=\mu_i(S)}\right]\\&=\mathbb{E}_{S\sim\mathrm{P}^{\boldsymbol{\mu}}}\left[\left(\nabla_{a_i}\text{GNN}_{\phi_i}\left(\left[S,l_1^{\text{s}},\cdots,l_{n_{\text{im}}}^{\text{s}}\right],A\right)+2*\lambda\left\|a_i\right\|\right)\right.\\&\quad\left.*\nabla_{\theta_i}\text{GCN}_{\theta_i}(S,A)\Big|_{a_i=\mu_i(S)}\right]\end{aligned}\tag{55}$$

其中，$\tilde{a}_i=\text{GCN}_{\theta_i}(S,A)$是$\text{GCN}_{\theta_i}\left(S_i(t),A\right)$在批量样本下的表示，记

$$\begin{aligned}\delta:=&\mathbb{E}_{S\sim\mathrm{P}^{\boldsymbol{\mu}}}\left[Q_i(S,a_1,\cdots,a_{n_{\text{im}}})-\left(R_i(S)\right.\right.\\&\left.\left.+\gamma*Q_i^{\boldsymbol{\mu}'}(S,a_1',\cdots,a_{n_{\text{im}}}')\right)\Big|_{a_i=\mu_i(S),a_i'=\mu_i'(S)}\right]\end{aligned}\tag{56}$$

为此，Critic网络参数ϕ_i和Actor网络参数θ_i的更新表达式为

$$\phi_i\leftarrow\phi_i+\alpha_{\phi_i}*\delta*\nabla_{\phi_i}\text{GNN}_{\phi_i}\left(\left[S,l_1^{\text{s}},\cdots,l_{n_{\text{im}}}^{\text{s}}\right],A\right)\tag{57}$$

$$\begin{aligned}\theta_i\leftarrow\theta_i+\alpha_{\theta_i}*\mathbb{E}_{S\sim\mathrm{P}^{\boldsymbol{\mu}}}&\left[\left(\nabla_{a_i}\text{GNN}_{\phi_i}\left(\left[S,l_1^{\text{s}},\cdots,l_{n_{\text{im}}}^{\text{s}}\right],A\right)\right.\right.\\&\left.\left.+\lambda\left\|a_i\right\|\right)*\nabla_{\theta_i}\text{GCN}_{\theta_i}\left(S,\Big|_{a_i=\mu_i(S)}A\right)\right]\end{aligned}\tag{58}$$

其中，α_{ϕ_i}和α_{θ_i}分别是ϕ_i和θ_i的学习率，它们是超参数，λ也是超参数，在它们的作用下，更新过程可以不考虑常数.

用$\phi_i(k),\theta_i(k)$和$\phi_i^-(k),\theta_i^-(k)$分别表示原网络和目标网络Critic和Actor网络迭代k步的参数，本文这里采用了软更新方式[39]对它们进行迭代

$$\phi_i^-(k)=\begin{cases}\tau*\phi_i(k)+(1-\tau)*\phi_i^-(k), & k\bmod k_{\text{u}}=0\\\phi_i^-(k), & \text{other}\end{cases}\tag{59}$$

$$\theta_i^-(k)=\begin{cases}\tau*\theta_i(k)+(1-\tau)*\theta_i^-(k), & k\bmod k_{\text{u}}=0\\\theta_i^-(k), & \text{other}\end{cases}\tag{60}$$

其中，$0<\tau<1$是正数，k_{u}表示软更新的时间步数，mod表示除余运算. 它们在k=0时的网络初始化一样，即

$$\phi_i^-(0)=\phi_i(0),\theta_i^-(0)=\theta_i(0)\tag{61}$$

通过以上的推导和分析过程可知，本文设计的算法Actor和Critic参数优化过程满足测度梯度定理[40]要求，和Actor-Critic算法[41]一样能够保证其收敛性. 本文后续还将通过实验及其结果来验证设计算法的收敛性.

综上所述，多智能体图强化学习算法的实现过程如算法2所示. 为了保证样本的充分性，第9行中“经验池中样本足够”表明需要在经验池中样本达到一定数据量后再进行训练. 此外，第9行中的条件$t\bmod T_{\text{up}}=0$表明采样和训练过程可以异步进行，即允许采T_{up}次采样再训练一次模型，这样可以平衡采样时间和训练的速度. 第10~11行做智能体参数更新时，实际上是对每个智能体都要进行更新的，其中一个智能体更新时，其余的智能体不变，等到所有智能体更新完再进入下一个时间步.

算法2 多智能体图强化学习算法

输入：多对多任务$\boldsymbol{M}$，智能节点集合N_{im}，Actor网络学习率α_{θ_i}，Critic网络学习率α_{ϕ_i}，折扣因子γ，训练周期数E_2，软更新参数τ，奖励权重α

输出：最佳Actor网络参数θ_i

1. 分别用随机生成网络参数θ_i和ϕ_i初始化智能体I_i的Actor网络参数和Critic网络参数
2. 复制I_i的初始化参数给目标网络，$\theta_i^-\leftarrow\theta_i,\phi_i^-=\phi_i$
3. 初始化经验池$\Re$
4. **for** 序列$e=1\rightarrow E_2$ **do**
5. 获取环境初始状态$S(0)$
6. **for** 时间步$t=1\rightarrow T$ **do**
7. 将智能体的Actor网络参数传输到智能数据平面，智能数据平面根据当前的观测和ϵ-贪婪算法给出动作$\boldsymbol{a}_{I_i}^{\text{c}}(t)$
8. 通过对智能体数据平面的观测获取样本片段$S(t),a_1^{\text{c}}(t),R_1(t),\cdots,a_{n_{\text{im}}}^{\text{c}}(t),R_{n_{\text{im}}}(t),S(t+1)$，并存储到经验池$\Re$中
9. **if** 经验池中的样本足够且$t\bmod T_{\text{up}}=0$ **then**
10. 从经验池中随机采样b_{s}批量样本$\left[S(t),a_1^{\text{c}}(t),R_1(t),\cdots,a_{N_{\text{im}}}^{\text{c}}(t),R_{N_{\text{im}}}(t),S(t+1)\right]^{b_{\text{s}}}$
11. 根据式(46)~式(58)更新智能体Actor网络和Critic网络的参数θ_i,ϕ_i
12. 根据式(59)~式(60)更新智能体Actor网络和Critic网络的参数θ_i^-和ϕ_i^-
13. **end if**
14. **end for**
15. **end for**

在算法2中,外层循环的次数为E_2,内层循环的次数为T,在内层循环中需要在一些时间步采样批量为b_s的样本进行模型训练,每次训练需要遍历所有的智能体,所以还会受到智能体数量n_r的影响,其时间开销可以近似为$O(E_2\times T\times b_s\times n_r)$,其中智能体个数由第一阶段Q-学习输出的结果确定,训练周期数、训练时间步和样本批量大小由实验进行合适的调整可以确保更好的收敛速率,虽然训练时间较长,但依据强化学习的原理可知,推理决策的时间代价并不受训练过程的影响,如图4所示,在每次做决策的过程中每个智能AP节点仅仅需要线性执行①~⑨步,都是简单的线性运算,时间复杂度为$O(1)$,这为算法的应用提供了可行性. 内存上,算法需要维护经验池$\Re$,其中存储样本片段,样本片段中内存占用最大的是状态,根据图的表示,用n_e表示拓扑中边的条数,每条边用一个二维向量表示,所以每个状态内存开销为$2n_e$,假设经验池的最大容量为$C_{\Re}$,则其内存开销近似为$O(C_{\Re}\times n_e)$. 注意到算法2的时间开销与智能体个数有关,随着智能体的增加,时间开销会增大,所以进行智能AP的选择可以降低时间开销;此外如果不使用图强化学习,状态表示的内存开销为n^2,算法2内存开销将近似为$O(C_{\Re}\times n^2)$,所以使用图强化学习可以降低内存开销.

下面将通过具体的实验说明上述算法的可行性和适用性.

6　实验分析

根据上述模型框架及算法设计进行相应实验,下面将从实验环境,参数设置和实验结果3个部分进行实验分析.

6.1　实验环境

本文实验硬件平台为:Intel(R) Xeon(R) Gold 5218 CPU @ 2.30 GHz、NVIDIA GeForce RTX3090 GPU,软件环境为:Ubuntu 18.04、Python 3.9、torch 1.13.0+cu117. 后续实验中使用的网络拓扑如图5所示,该拓扑来自文献[42]的传感器多跳网络节点拓扑,其中AP节点被分为3种类型:数据入口AP为源节点;中继AP节点负责数据包的传输;数据中心AP为目标节点. 需要说明的是,这里的最短路径算法仅考虑路径跳数因素而忽略距离等因素,但从以上设计方案推导的过程可知并不局限于此.

6.2　参数设置

为了便于描述,用$\ell(\cdot)$表示集合的序列表示,即在集合的基础上添加元素的顺序. 本文设置节点序列为$\ell(N)$=[14, 12, 13, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 15],源节点序列为$\ell(N_s)$=[14, 12, 13],目标节点序列为$\ell(N_d)$=[10, 11, 15],相应的任务矩阵设置为

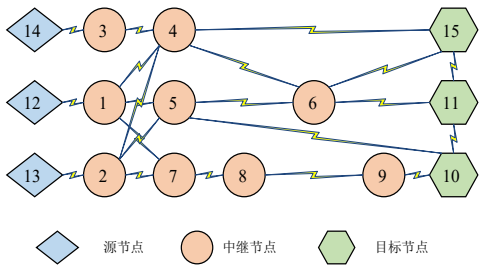


图5　网络拓扑

$$\boldsymbol{M}=\begin{bmatrix}40 & 30 & 30\\ 30 & 40 & 30\\ 30 & 30 & 40\end{bmatrix} \tag{62}$$

根据$\boldsymbol{M}=\{m_{ij}\}\in\mathbb{N}^{n_s\times n_d}$的定义,$m_{ij}$表示源序列$\ell(N_s)$中第$i$个节点发往目的节点序列$\ell(N_d)$第$j$个节点的数据包数,比如$m_{11}=40$表示从源节点14发往目的节点10的数据包数为40个. 值得注意的是文献[21]中通过微调的方式(即元学习方式)以适应不同$\boldsymbol{M}$,本文这里可以同样采用这种方式.

为了避免问题的平凡性,为网络节点赋予不同的单位时间数据包处理能力

$$c_i=\begin{cases}3, & s_i=1,2,12,13\\ 1, & \text{other}\end{cases} \tag{63}$$

即编号为1,2,12,13这4个网络节点在单位时间内处理数据能力是其他网络节点的3倍.

表1中列出了实验中其余主要参数,这里的数学符号对应前文中的相关推导中的符号,取值范围是该参数理论上的取值范围,实验过程中的调节范围指的是做实验的时候调到的数据,部分参数只有一个值,是因为这个参数相对作用不是很大,没有对它进行调试. 另外需要说明的是,数据包传输和接收能耗用的单位是unit,这是因为实验中这两个量是通过预设值计算得到的. 在下面实验结果的分析中,如果不单独说明的参数和表1中的参数保持一致.

6.3　实验结果

本文的实验包括2个部分. 第一个部分是智能AP节点选择实验,负责选择出需要智能体辅助决策的节点. 以下实验结果的多对多通信任务矩阵由式(62)给出.

图6是使用Q-学习选择智能AP节点的实验训练奖励曲线. 该结果使用参数$\alpha^{no}=0.01, E_1=600, n_r=5$. 图6中曲线是训练过程中的实验的奖励值,星点是探索发生的点,除星点外可以明显看到曲线逐渐增加,这说明了算法的有效性. 大约在200代之后虽然还有波动但已经趋于平稳,可见节点选择已经开始收敛. 虽然图中的奖励曲线在收敛后仍然在波动,但是它确定的智能AP节点基本是固定的,这为第二阶段智能体实时求解

表1 实验中主要参数的取值范围及实验中的调节范围

参数的含义	符号	范围	实验调节范围
数据包传输能耗	e_i^{t}	$(0,\infty]$	10 unit
数据包接收能耗	e_i^{r}	$(0,\infty]$	3 unit
目标函数权衡系数	β	(0, 1]	$\{1, 0.8, 0.6, 0.4, 0.2\}$
算法1学习率	α^{no}	[0, 1)	$\{1, 0.1, 0.01\}\times 10^{-1}$
算法1折扣因子	γ^{no}	(0, 1)	0.9
算法1权衡系数	β^{no}	[0, 1]	0.01
算法1探索概率	ϵ^{no}	(0, 1)	0.1
算法1周期数	E_1	$\mathbb{N}$	$\{3, 6\}\times 10^2$
算法2智能AP个数	n_{r}	$\mathbb{N}$	$\{0, 1, \cdots, 10\}$
算法2 Actor网络学习率	α_{θ_i}	[0, 1)	$\{2, 0.2, 0.02, 0.002\}\times 10^{-1}$
算法2 Critic网络学习率	α_{ϕ_i}	[0, 1)	$\{2, 0.2, 0.02\}\times 10^{-4}$
算法2折扣因子	γ	(0, 1)	0.95
算法2批量大小	b_{s}	$\mathbb{N}$	$\{2, 4, 8, 16, 32, 64\}$
算法2周期数	E_2	$\mathbb{N}$	$\{3, 5, 10, 15\}\times 10^2$
算法2时间步周期	T	$\mathbb{N}$	$\{3, 5\}\times 10^2$
算法2回路惩罚	C_{h}	$(0,\infty]$	$\{0.1, 1, 10\}$

提供了可能. 另外,随着实验代数的增加,星点逐渐变得稀疏,这是因为本文采用了探索率衰减操作,即在表1中算法1探索概率的基础上,每隔50代衰减一半. 这个衰减操作在一定程度上可以提高算法的收敛性,但可以更好地提高探索概率,对于智能AP节点选择这样需要较高探索率的实验是非常必要的.

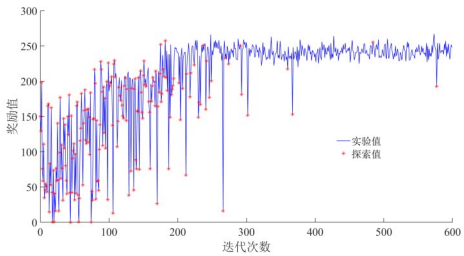


图6 智能AP节点选择训练奖励曲线

表2是智能AP节点选择的结果,这里参数的设置为 $\alpha^{\mathrm{no}}=0.01, E_1=600, n_r$ 分别取1~10,即表中智能AP节点个数的设置. 选择结果和奖励值是指定智能AP节点个数后选择出的智能AP节点集合和对应的奖励值. 从表2中数据来看,该算法确定的智能AP节点集合并不是相互包含的,奖励值也不是单调递增的,这一结果是因为智能AP节点和普通AP节点上的下一跳的选择不一样导致的,具体来说,在智能AP节点上以 β^{no} 的概率使用随机简单路径确定下一跳,以 $1-\beta^{\mathrm{no}}$ 的概率随机最短路径确定下一跳. 而在普通AP节点上以概率1使用随机最短路径选择下一跳,这样做的原因是尽可能模拟多智能体的下一跳选择.

在表2中可以看到,智能AP节点个数为5的时候选择的结果 $N_{\mathrm{im}}=\{1,2,4,5,6\}$ 涵盖了前面4种情况下的选择结果,且从智能AP节点个数从1~5的奖励值一直是单调递增的,从智能AP节点个数为5之后开始波动,再次到达更好的结果时对应的奖励值为238.5,比智能AP节点个数为5的时候的237.9仅仅多了0.06,但节点个数却增加了2个,节点个数成本比较大. 所以本文在图5的拓扑和式(62)的任务下,选择智能AP节点个数设置为5,选择结果为 $N_{\mathrm{im}}=\{1,2,4,5,6\}$,下面将以这个结果确定智能体的部署.

表2 智能AP节点选择结果

智能AP节点个数	选择结果	奖励值
1	{4}	109.5
2	{1,4}	153.4
3	{1,4,5}	189.0
4	{1,2,4,6}	200.0
5	{1,2,4,5,6}	237.9
6	{1,2,4,5,11,15}	220.4
7	{1,2,4,5,6,10,11}	238.5
8	{1,2,4,5,6,10,11,15}	239.0
9	{1,2,4,5,6,10,11,13,15}	236.0
10	{1,2,4,5,6,8,10,11,12,15}	238.1

另一部分是通过多智能体图强化学习实现路由的优化. 通过在智能AP节点上部署智能体,由智能体辅助决定网络节点将数据包发往哪一个下一跳. 对于普通AP节点上的网络节点,直接使用随机最短路径算法将数据包传递给下一跳.

为了说明本文设计方法能有效性,本文设计了基准实验,即由MMForests[10]改编而来的SPR-M2M(Shortest Path Routing in Many-to-Many communication)算法,该算法在每个源点和终点对间使用Dijkstra算法获取最短跳数多对多通信路径算法,实验结果表明该算法完成任务 $\boldsymbol{M}$ 的时延为225 slot,完成任务时各节点上累计能耗标准差为740.757. 下面从2个场景说明所提出的方法能有效地降低多对多通信任务完成的时延,并且可以通过调节参数使任务完成的时延和各节点累计能耗标准差之间达到平衡.

情况1:不考虑能耗影响的实验结果. 图7为不考虑能耗下多智能体图强化学习训练过程中的奖励结果,即目标函数的权衡系数 β=1. 智能体Actor网络的学习率均为 $\alpha_{\theta_i}=0.002$,智能体Critic网络的学习率为 $\alpha_{\phi_i}=0.000\,02$,批量更新参数 b_{s}=8,周期数 E_2=1 500,回路惩罚 C_{h}=1.

图7中智能体1~5分别对应智能AP节点集合 $N_{\mathrm{im}}=\{1,2,4,5,6\}$,即将智能体1~5分别部署到网络节点 s_1, s_2, s_4, s_5, s_6 上. 图中奖励均小于0,其绝对值表示在任务 $\boldsymbol{M}$ 下智能体处理数据付出的成本总和,其中一代就表示

一次任务 $\boldsymbol{M}$ 完成的成本,从图7中可以看出,随着迭代的代数增加,各智能体的奖励趋势逐渐增加并趋于平稳,说明了算法的有效性和收敛性. 另外,注意到约600代智能体1,2,4,5的奖励函数有所下降,而智能体3的奖励出现了增加,表明智能体之间进行了相互协调.

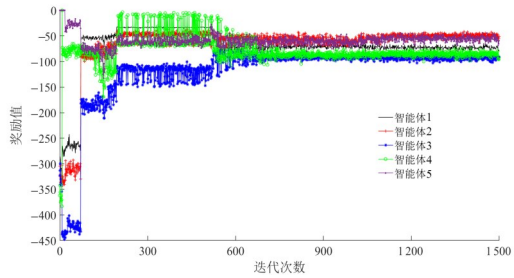


图7　多智能体强化学习训练奖励曲线

图8是情况1下训练过程中生成的模型完成任务时间的性能曲线. 曲线表示训练过程中每隔5代生成模型完成任务的时延,直线是SPR-M2M完成任务的时延225 slot. 从图8中可以看出,随着训练不断进行,MAGDS-M2M完成任务的时间越来越小,且逐渐超过SPR-M2M,这说明了本文所提方法的有效性.

在图8中可以看出模型在605代的时候取到最优的效果,完成任务的时间仅仅需要173 slot,比SPR-M2M降低了23.11%. 将605代的模型记为MAGDS-M2M-v605,下面从完成任务的数据包路径和关键节点数据包个数变化情况进一步比较MAGDS-M2M-v605和SPR-M2M算法的差异.

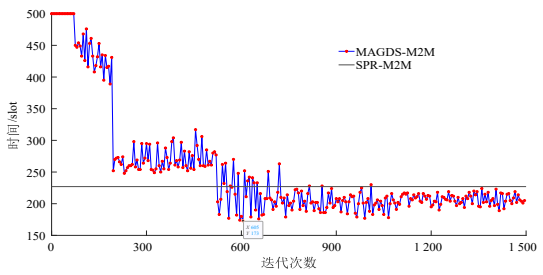
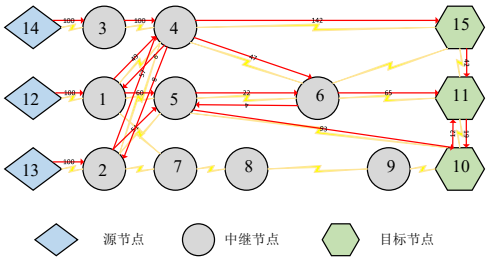
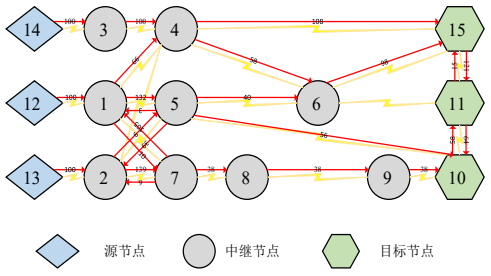


图8　模型迭代完成任务时间的变化情况

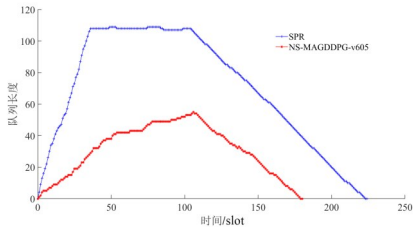
从图9中可以看到,相比于SPR-M2M,MAGDS-M2M-v605更加充分地利用了网络,考虑到了网络中所有路径. 此外,从图9(b)中可以看出,数据包交互比较频繁的节点是4、5、7号节点,图9(c)~图9(e)显示,在SPR-M2M中,节点4长时间处于拥塞状态,即数据包队列长度长时间居高不下,节点5部分时间处于闲置,节点7一直处于闲置,这导致很大的资源浪费. 相比之下,MAGDS-M2M-v605有效地解决了这个问题,图9(c)显示节点4中数据包队列长度的最大值从100左右变成了60左右,明显降低了拥塞情况,同样图9(d)显示节点5中数据包队列长度最大值从70左右变为40左右,图9(e)显示节点7从完全闲置的状态变为数据包队列长度最大值到80左右,但仍然有闲置时间,这说明节点7还不是阻碍任务完成速度的关键节点. 另外MAGDS-M2M-v605明显缩短了任务完成的时间,这说明MAGDS-M2M-v605充分利用节点7有效地分担了节点4和节点5的负担,从而提高了任务完成的速度.



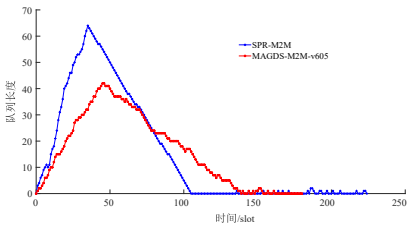
(a) SPR-M2M在应用过程中数据包的传输情况



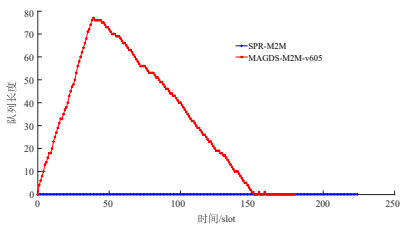
(b) MAGDS-M2M-v605在应用过程中数据包的传输情况



(c) 节点4数据包队列长度变化情况



(d) 节点5数据包队列长度变化情况



(e) 节点7数据包队列变化情况

图9　SPR-M2M和MAGDS-M2M中数据包转移的对比

上述情况将目标函数的权衡系数设置为 β=1,即不考虑节点上能耗平衡问题. 为了考虑完成任务时延和各节点能耗标准差之间的平衡,本文讨论了下面的情况.

情况2:考虑能耗影响的实验结果. 该情况下将目

标函数的权衡系数分别设置为β=1,0.8,0.6,0.4,0.2这5种情况,其余参数保持不变,观察完成任务时延和累计能耗的标准差性能变化.

为了更好地比较完成任务时延和各节点上累计能耗的标准差,将SPR-M2M的完成任务时间表示为T_{sp}=225 slot,各节点上累计能耗的标准差表示为V_{sp}=740.757;将不同β下完成任务时延和各节点上累计能耗的标准差分别表示为T_β和V_β,则不同β下完成任务时延和各节点上累计能耗的标准差根据SPR-M2M进行标准化后的目标函数为

$$z_\beta=-\left[\beta*\left(\frac{T_\beta-T_{sp}}{T_{sp}}\right)+(1-\beta)*\left(\frac{V_\beta-V_{sp}}{V_{sp}}\right)\right] \quad (64)$$

在这个定义下,SPR-M2M的目标函数值为0,其余方法的目标函数大于0,说明其性能优于SPR-M2M,否则说明性能不如SPR-M2M.

图10显示了不同β下各模型的目标函数曲线,图中显示不同β下目标函数的变化趋势基本相同,可以根据具体的需要调节β获取相应的模型,再根据相应的模型进行路由决策.

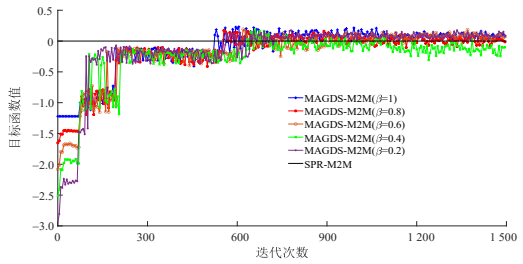


图10 模型的目标函数曲线

表3中列出了不同β下目标函数的最优值及其对应的模型代数、完成任务的时延和各节点的上累计能耗的标准差. 表3中数据显示,除了β=1没考虑累计能耗的影响外,β=0.2时目标函数最优模型取得最小的完成任务时延,但同时其各节点上累计能耗的标准差也是最大的;β=0.4时目标函数最优模型取得最小各节点上累计能耗的标准差,但同时其完成任务的时延却是最大的;β=0.6时目标函数取得最大值,相对于前面两种情况其完成任务时延和各节点上累计能耗的标准差都比较均衡. 这说明实际中可以通过调节β值获取满足不同需求的模型,从而获得多对多通信相应的路由策略.

情况3:为了验证智能体之间存在"干扰"和"博弈"的问题,本文设计了不同智能AP数量的对比,在情况1的参数基础上,分别取n_r=3,4,5,6,7,15这6种情况进行实验,其中n_r=3,4,5,6,7是根据表2进行智能体选择的相应个数的智能体,n_r=15是将所有节点作为智能体,类似于文献[21]中的MAPPO算法. 图11是这6种情况下模型迭代完成任务时间的变化情况.

表3 模型的目标函数最优结果

β值	z_β最优值(迭代次数)	相应的T_β	相应的V_β
1	**0.231(605)**	173	673.391
0.8	0.170(555)	188	599.692
0.6	0.192(620)	181	602.867
0.4	0.186(660)	194	**578.769**
0.2	0.177(1 335)	**164**	627.355

注:加粗字体表示最优结果.

图11显示智能体个数为3时虽然整体比4个智能体的情况要好,但4个智能体能达到的最佳效果要优于3个智能体的情况,而5个智能体时明显优于3个或4个智能体,更好的是6个智能体,6个智能体最低任务完成时间可以达到159 slot,比5个智能体最佳的173 slot还降低了8.09%,比SPR方法的225 slot更是降低了29.33%,这可以说明随着有效智能体的增多,多智能体能更好起到配合的效果. 但是,随着智能体数量进一步增加,比如n_r=7的情况,智能体开始变得不稳定,这主要是因为随着智能体的增加,智能体间彼此"干扰"和"博弈"的现象加剧,尤其是无效智能体,即有无智能体决策不发生变化. 比如说10号节点,虽然它有3个端口,但无论数据包从哪个节点过来,它的传输路径基本都是固定的,如果数据包的目的节点不是10,最佳下一跳基本都是11号节点,所以在它上面部署的智能体没有起到应有的作用,反而会"干扰"其他智能体决策,从而出现了不稳定的现象. 全部节点都作为智能体的方法n_r=15甚至变得更差,更是验证了该想法. 由此可以看出当智能体数量较少时,增加智能体可以获得更好的效果,但随着智能体进一步增多,不仅成本开销会增加,而且稳定性和收敛性都会变差.

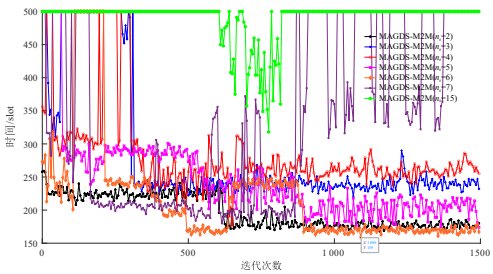


图11 不同智能体数量下模型迭代完成任务时间的变化情况

情况4:为了验证所提智能AP选择算法的有效性,本文设计了不同智能AP选择算法的对比实验,在n_r=4智能AP节点为[1,2,4,6]的基础上随机添加一个AP,比如3,10,8,7,15,和使用Q-学习选择的AP节点5进行对比,如图12所示. 图12显示,随机选择节点下的迭代完成任务的时间收敛结果基本都要高于使用Q-学习选择节点5的情况. 此外,随机选择节点可达到的最优任

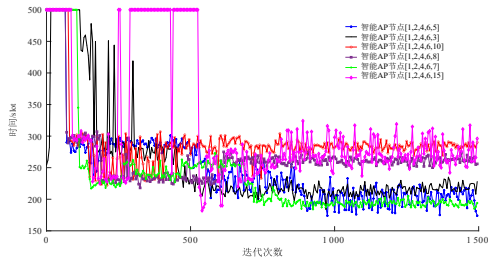


图 12　不同智能 AP 选择下模型迭代完成任务时间的变化情况

务完成时间都低于 Q-学习选择的节点 5,这可以说明本文所提的 Q-学习方法可以正确引导智能节点的选择.

考虑到之前的在智能 AP 节点数为[1,2,4,6]的基础上随机添加一个 AP 节点的实验方案,虽然以随机选择方式开展了实验,但得到的每组智能 AP 节点集合之间相似度很大,这样用以评估本文设计的智能 AP 节点选择方式效果的说服力还不够. 对此还补充了直接随机选择 5 次 AP 节点组的方案,每次选择 5 个 AP 节点,这样选择的每组 AP 节点之间差异会比较大,将它们和使用 Q-学习选择的[1,2,4,5,6]作实验对比. 如图 13 所示,智能 AP 节点[1,2,8,10,11]和[2,4,6,11,15]两组实验虽然具有明显的收敛趋势,但它们的任务完成时间比较高,甚至基本超过了 SPR-M2M 的任务完成时间的 225 slot,说明这样的智能 AP 的调度没有起到应有的作用;更有甚者,智能 AP 节点[4,5,6,8,10]的任务完成时间一直在波动,说明这种智能 AP 节点的选择反而会带来相反的效果,原因在于智能体之间没有很好合作,反而出现了相互间的干扰;智能 AP 节点[3,5,10,11,15]的任务完成时间收敛,但仅仅是在 SPR-M2M 的任务完成时间 225 slot 上下波动,没有很好地体现出智能体合作决策的优越性;智能 AP 节点[2,4,6,11,15]出现了优于[1,2,4,5,6]的结果,但不稳定,随着训练进行,任务完成时间有增加的趋势. 相比之下,说明了本文所设计的 Q-学习选择算法相比于随机选择算法虽然不一定是最优的,但接近最优并且相对稳定. 从以上两种实验方式(在已有节点的基础上添加节点和直接随机选择)及其实验对比结果可以看出,本文所设计的 Q-学习选择算法是优于对比方法的.

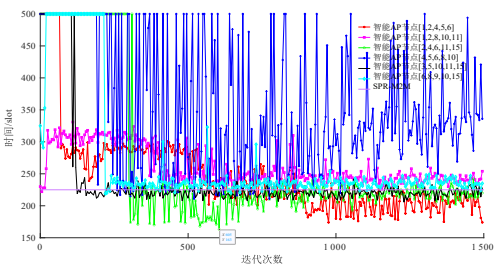


图 13　随机选择智能体模型的任务完成时间随迭代次数的变化情况

7　结论

为解决现有的路由算法在多对多通信中的局限性,本文提出了一种两阶段的智能路由算法 MAGDS-M2M. 首先通过智能 AP 节点选择算法选择适合部署智能体的网络节点,然后利用多智能体图强化学习算法训练智能体并实现多对多通信路由决策.

MAGDS-M2M 算法仅需要在部分 AP 节点上部署智能体,减少了计算和部署成本. 另一方面,MAGDS-M2M 算法采用图神经网络,更加适应网络状态信息,减少了模型训练时存储的空间开销. 此外,MAGDS-M2M 算法针对 Actor 网络的输出设计了局部观测方法,有效避免了无效动作的产生,降低了模型训练的时间开销,提升了算法收敛速度. 实验结果显示 MAGDS-M2M 算法在完成时间和能量平衡上均优于基准方法,这说明 MAGDS-M2M 有效性和适用性. 类似 SDWN 单域中无法适应网络规模增加需求的问题,随着节点数进一步的增加,通常的做法是在 SDWN 架构下采取多域分层强化学习的方式,而在多域场景下的工作属于另一个应用场景,需要对多对多路由通信问题做扩展性的进一步研究,这也是下一步需要进行的工作.

参考文献

[1] DE MORAES R M, SADJADPOUR H R, GARCIA-LUNA-ACEVES J J. Many-to-many communication for mobile ad hoc networks[J]. IEEE Transactions on Wireless Communications, 2009, 8(5): 2388-2399.

[2] XIONG S G, LI J Z. Optimizing Many-to-Many Data Aggregation in Wireless Sensor Networks[M]//Advances in Data and Web Management. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009: 550-555.

[3] GUO D K, TENG X Q, HU Z Y, et al. Source selection problem in multi-source multi-destination multicasting[J]. Computer Networks, 2017, 127: 43-55.

[4] ARSHAD S, AZAM M A, AHMED S H, et al. Towards information-centric networking (ICN) naming for internet of things (IoT): The case of smart campus[C]//Proceedings of the International Conference on Future Networks and Distributed Systems. New York: ACM, 2017.

[5] LIU R Z, ZHU Y T, ZHANG Y, et al. Resource mobility aware hybrid task planning in space information networks[J]. Journal of Communications and Information Networks, 2019, 4(4): 107-116.

[6] MARTINEZ G, LI S F, ZHOU C. Multi-commodity online maximum lifetime utility routing for energy-harvesting wireless sensor networks[C]//2014 IEEE Global Communications Conference. Piscataway: IEEE, 2014: 106-111.

[7] BELEY O, CHAPLYHA V. A management of cloud services in social-economic systems[C]// Proceedings of the 20th International Conference on Information Technology for PracticeInformation. Athens: Panhellenic Conference on Informatics, 2017: 33-45.

[8] MI H B, XU K L, FENG D W, et al. Collaborative deep learning across multiple data centers[J]. Science China Information Sciences, 2020, 63(8): 11432.

[9] MOTTOLA L, PICCO G P. MUSTER: Adaptive energy-aware multisink routing in wireless sensor networks[J]. IEEE Transactions on Mobile Computing, 2011, 10(12): 1694-1709.

[10] CHEN Y R, RADHAKRISHNAN S, DHALL S, et al. On multi-stream multi-source multicast routing[J]. Computer Networks, 2013, 57(15): 2916-2930.

[11] REN C, CHEN X X, XIANG H Y, et al. On efficient delay-aware multisource multicasting in NFV-enabled softwarized networks[J]. IEEE Transactions on Network and Service Management, 2022, 19(3): 3371-3386.

[12] JAIN K, PADHYE J, PADMANABHAN V N, et al. Impact of interference on multi-hop wireless network performance[J]. Wireless Networks, 2005, 11(4): 471-487.

[13] HE D J, CHAN S, GUIZANI M. Securing software defined wireless networks[J]. IEEE Communications Magazine, 2016, 54(1): 20-25.

[14] HU H L, CHEN H H, MUELLER P, et al. Software defined wireless networks (SDWN): Part 1 [guest editorial] [J]. IEEE Communications Magazine, 2015, 53(11): 108-109.

[15] FORNEY G D. The viterbi algorithm[J]. Proceedings of the IEEE, 1973, 61(3): 268-278.

[16] CASAS-VELASCO D M, RENDON O M C, FONSECA N L S DA. Intelligent routing based on reinforcement learning for software-defined networking[J]. IEEE Transactions on Network and Service Management, 2021, 18(1): 870-881.

[17] YE M, ZHAO C W, WEN P, et al. DHRL-FNMR: An intelligent multicast routing approach based on deep hierarchical reinforcement learning in SDN[J]. IEEE Transactions on Network and Service Management, 2024, 21(5): 5733-5755.

[18] OKINE A A, ADAM N, NAEEM F, et al. Multi-agent deep reinforcement learning for packet routing in tactical mobile sensor networks[J]. IEEE Transactions on Network and Service Management, 2024, 21(2): 2155-2169.

[19] ALAM M Z, KHAN K S, JAMALIPOUR A. Multiagent best routing in high-mobility digital-twin-driven Internet of vehicles (IoV) [J]. IEEE Internet of Things Journal, 2024, 11(8): 13708-13721.

[20] ZAFEIRIOU S, BRONSTEIN M, COHEN T, et al. Guest editorial: Non-euclidean machine learning[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2022, 44(2): 723-726.

[21] CHEN L, HU B, GUAN Z H, et al. Multiagent meta-reinforcement learning for adaptive multipath routing optimization[J]. IEEE Transactions on Neural Networks and Learning Systems, 2022, 33(10): 5374-5386.

[22] TAN D D, KIM D S. Dynamic traffic-aware routing algorithm for multi-sink wireless sensor networks[J]. Wireless Networks, 2014, 20(6): 1239-1250.

[23] SUN B L, LI L Y. Optimizing on multiple constrained QoS multicast routing algorithms based on GA[J]. Journal of Systems Engineering and Electronics, 2004, 15(4): 677-683.

[24] LI W, LI K S, HUANG Y, et al. A EA- and ACA-based QoS multicast routing algorithm with multiple constraints for ad hoc networks[J]. Soft Computing, 2017, 21(19): 5717-5727.

[25] MANN P S, SINGH S. Energy-efficient hierarchical routing for wireless sensor networks: A swarm intelligence approach[J]. Wireless Personal Communications, 2017, 92(2): 785-805.

[26] PAN X Q, PENG D L, LI S M. Quantum binary improved artificial bee colony algorithm to solve the spanning tree construction problem in vehicular ad hoc network[J]. IEEE Internet of Things Journal, 2024, 11(22): 36014-36029.

[27] BOYAN J, LITTMAN M. Packet routing in dynamically changing networks: A reinforcement learning approach[J]. Advances in Neural Information Processing Systems, 1993, 6: 671-678.

[28] YAO Z, WANG Y, QIU X S. DQN-based energy-efficient routing algorithm in software-defined data centers[J]. International Journal of Distributed Sensor Networks, 2020, 16(6): 15501477209.

[29] LU Y, CHEN Y H, XU X, et al. A sub-flow adaptive multipath routing algorithm for data centre network[J]. International Journal of Computational Intelligence Systems, 2023, 16(1): 25.

[30] ZHOU W, JIANG X, GUO B L, et al. PQROM: To optimize software defined network QoS-aware routing with proximal policy optimization[J]. Journal of Intelligent & Fuzzy Systems, 42(4): 3605-3614.

[31] QIU X, XIE Y, WANG Y, et al. QLGR: A Q-learning-

based geographic FANET routing algorithm based on multi-agent reinforcement learning[J]. KSII Transactions on Internet and Information Systems, 2021, 15(11): 4244-4274.

[32] ABDOLLAHI M, NI W, ABOLHASAN M, et al. Software-defined networking-based adaptive routing for multi-hop multi-frequency wireless mesh[J]. IEEE Transactions on Vehicular Technology, 2021, 70(12): 13073-13086.

[33] TRIMPONIAS G, XIAO Y, WU X R, et al. Node-constrained traffic engineering: Theory and applications[J]. IEEE/ACM Transactions on Networking, 2019, 27(4): 1344-1358.

[34] WEI Q L, LEWIS F L, SUN Q Y, et al. Discrete-time deterministic Q-learning: A novel convergence analysis[J]. IEEE Transactions on Cybernetics, 2017, 47(5): 1224-1237.

[35] SILVER D, LEVER G, HEESS N, et al. Deterministic policy gradient algorithms[J]. 31st International Conference on Machine Learning, ICML 2014, 2014, 1: 605-619.

[36] JANG E, GU S, POOLE B. Categorical reparameterization with gumbel-Softmax[C]//International Conference on Learning Representations. Washington DC: ICLR, 2022.

[37] DEGRIS T, WHITE M, SUTTON R S. Off-Policy Actor-Critic[C]//International Conference on Machine Learning. New York: ICML, 2012.

[38] LI L T, LI D Z, SONG T H, et al. Actor-critic learning control based on Q-regularized temporal-difference prediction with gradient correction[J]. IEEE Transactions on Neural Networks and Learning Systems, 2018, 29(12): 5899-5909.

[39] HAARNOJA T, ZHOU A, ABBEEL P, et al. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor[EB/OL]. (2018-08-08) [2025-05-12]. https://arxiv.org/abs/1801.01290v2.

[40] SUTTON R S, MCALLESTER D, SINGH S, et al. Policy gradient methods for reinforcement learning with function approximation[J]. Advances in neural information processing systems, 1999, 12: 06643.

[41] BHATNAGAR S, SUTTON R S, GHAVAMZADEH M, et al. Natural actor-critic algorithms[J]. Automatica, 2009, 45(11): 2471-2482.

[42] CHOI S P M, YEUNG D Y. Predictive Q-routing: A memory-based reinforcement learning approach to adaptive traffic control[C]//Advances in Neural Information Processing Systems, San Diego: NIPS, 1995, 8: 945-951.

作者简介

文　鹏　男,1994年生,贵州毕节人. 现为桂林电子大学信息与通信学院博士. 主要研究方向为软件定义网络、强化学习和随机优化与应用等.

E-mail: 22021101006@mails.guet.edu.cn

叶　苗　男,1977年生,广西桂林人. 现为桂林电子科技大学信息与通信学院教授、博士生导师. 主要研究方向为边缘存储与云存储、软件定义网络、无线传感网络、模式识别与机器学习等.

E-mail: yemiao@guet.edu.cn

王　勇　男,1964年生,四川成都人. 现为桂林电子科技大学计算机与信息安全学院教授、博士生导师. 主要研究方向为云计算、网络流量分析与信息安全等. 中国电子学会会员编号:E1900136115.

E-mail: ywang@guet.edu.cn

何　倩　男,1979年生,湖南郴州人. 现为桂林电子科技大学计算机与信息安全学院教授、博士生导师. 主要研究方向为模式识别、机器学习、软件定义网络与传感器网络等. 中国电子学会会员编号:E190021935S.

E-mail: heqian@guet.edu.cn

仇洪冰　男,1963年生,江苏如皋人. 现为桂林电子科技大学信息与通信学院教授、博士生导师. 主要研究方向为宽带无线通信、通信信号处理、辐射源定位等.

E-mail: qiuhb@guet.edu.cn