算力网络中依赖约束任务的分布式协同分配算法

张满钧，王颖，喻鹏，邱雪松，郭少勇
（北京邮电大学网络与交换技术全国重点实验室，北京 100876）

摘要：算力网络中任务分配面临任务依赖复杂、资源多维异构且分布不均，以及大规模节点难以全局协同的三重挑战。传统的单一智能体算法难以应对广域环境的动态资源协同，且往往依赖全局状态信息，现有的多智能体强化学习算法又缺乏对任务依赖和网络拓扑的针对性建模，导致全局协作效率低下。针对上述问题，提出了算力网络中依赖约束任务的分布式协同分配算法，先通过层索引计算、三角依赖解耦与资源阈值约束合并处理复杂的子任务依赖，再采用关系图注意力网络（RGAT）表征多维资源关系与拓扑特征，结合多智能体软演员–评论家（MASAC）算法实现分布式优化。实验结果表明，在多个场景下，所提算法相较于基线算法，在任务完成时间和能效方面均有显著提升 。

关键词：算力网络；任务重构；关系图注意力网络；多智能体强化学习；任务分配
中图分类号：TP309
文献标志码：A
doi：10.11959/j.issn.1000−436x.TXXB250697

Distributed collaborative allocation algorithm of dependency-constrained tasks in computing power networks

Zhang Manjun, Wang Ying, Yu Peng, Qiu Xuesong, Guo Shaoyong
State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China

Abstract: Task allocation in computing power networks faces challenges such as complex task dependencies, multi-dimensional and heterogeneous resources with uneven distribution, and difficulty in global collaboration among large-scale nodes. Traditional single agent algorithms were difficult to cope with the dynamic resource collaboration issues in wide area environments, and often relied on global state information, while existing multi-agent reinforcement learning algorithms lacked targeted modeling of task dependencies and network topology, resulting in low global collaboration efficiency. To address these issues, a distributed collaborative allocation algorithms of dependency-constrained tasks in computing power networks was proposed. First, complex subtask dependencies were processed through layering, triangular dependency decoupling and multi-resource threshold merging. Then, relation-graph attention network (RGAT) was adopted to encode multi-resource relations and topology features, combined with multi-agent soft actor-critic (MASAC) for distributed optimization. Experimental results show that in multiple scenarios, the proposed algorithms significantly optimizes task completion time and energy efficiency compared to baseline algorithms.

Key words: computing power network, task reconfiguration, relation-graph attention network, multi-agent reinforcement learning, task allocation

收稿日期：2026–01–04；**修回日期：**2026–03–26
通信作者：王颖，wangy@bupt.edu.cn
基金项目：智能电网国家科技重大专项（2030）（No.2025ZD0804700）
Foundation Item：Smart Grid-National Science and Technology (2030) Major Project (No.2025ZD0804700)

0　引言

随着计算应用与服务日趋复杂，资源密集型、时延敏感型应用的算力需求快速增长，传统集中式计算模式已难以适配资源有限且动态多变的场景[1]。为实现算力的按需获取与协同供给，算力网络（computing power network, CPN）在多域范围内统一编排计算、内存与存储资源[2]，构建跨区域、大规模的算力基础设施[3]。然而，多维异构资源和广域分布节点也给任务分配带来了新的挑战。实际任务通常由多个存在依赖约束的子任务构成，节点地理分散会导致跨域传输时延与能耗显著上升；若不能在调度前有效降低依赖结构带来的通信与等待开销，将难以实现算力网络的按需协同。同时，计算、内存、存储等多维资源分布不均增加了联合调度复杂度，而广域环境下节点海量分布且动态变化，又使全局状态难以实时获取。

基于上述场景特征，算力网络中的任务分配需要同时解决3类关键问题：一是降低跨域传输时延与能耗，并在不破坏关键依赖的前提下简化复杂依赖结构；二是刻画计算、内存与存储等多维资源之间的关联关系，以提升任务与资源的匹配精度；三是在局部观测和通信受限条件下实现分布式协同优化。

围绕上述问题，现有研究主要从任务重构、关系建模与任务分配展开。任务重构虽可降低依赖复杂度，但往往未与资源分布不均的广域环境耦合考虑，也未兼顾复杂有向无环图（directed acyclic graph, DAG）任务的跨层依赖解耦与资源阈值约束合并[4-6]。关系建模方面，图卷积网络（graph convolutional network, GCN）[7]和图注意力网络（graph attention network, GAT）[8]能够提取结构化特征，但更侧重静态结构建模，难以精准刻画计算、内存、存储等多维资源的动态关联。文献[9]提出了一种知识图谱推理框架，显式建模关系间交互，但未结合算力网络中的多维资源关系与动态拓扑特征。任务分配方面，早期单智能体深度强化学习算法在一定程度上缓解了高维决策问题，但集中式决策依赖全局状态，通信与计算开销随网络规模显著上升[10-11]。多智能体深度强化学习算法如多智能体深度确定性策略梯度（MADDPG）算法[12]适用于多节点任务分配，但通常假设静态网络拓扑，难以适应动态节点变化，且未显式建模节点间关系。多智能体软演员–评论家（MASAC）虽通过最大熵强化学习增强探索稳定性[13]，却忽视了任务依赖与拓扑约束，难以充分利用任务间的并行潜力。因此，目前仍缺乏一种能够兼顾任务复杂、资源异构与全局协同的算力网络任务分配算法。

针对上述挑战，本文提出一种融合依赖感知任务重构与关系图注意力网络（RGAT）的多智能体协作强化学习算法，主要贡献如下。

（1）设计多目标优化模型，将算力网络中的任务分配问题形式化为分散部分可观测马尔可夫决策过程（decentralized partially observable Markov decision process, Dec-POMDP），将任务完成时间与能效联合优化。

（2）提出依赖感知任务重构算法，降低DAG复杂度。

（3）提出基于关系图注意力网络的多智能体协同分配算法，实现局部观测条件下的高效协同决策。

（4）通过仿真实验验证所提算法在不同任务量、算力节点数、子任务数场景下的性能优势。

1　系统模型

1.1　算力网络模型

算力网络示例如图1所示，无向图 $\mathcal{G}_{\text{net}}=(V_{\text{net}},E_{\text{net}})$ 表示网络拓扑，包括算力节点 $V=\{v_1,v_2,\cdots,v_N\}$ 和有线链路 $E=\{(i,j)|v_i,v_j\in V_{\text{net}}\}$。

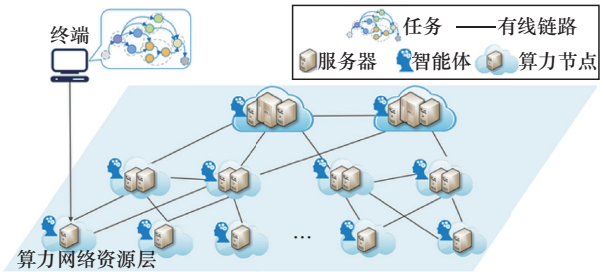


图1　算力网络示例

每一个 v_n 可以表示为一个三元组。

$$v_n=\{F_n^{\text{avail}}(t),M_n^{\text{avail}}(t),S_n^{\text{avail}}(t)\}\in V \tag{1}$$

将时间划分为等长时隙 $t\in\{1,2,\cdots,T\}$，每个时隙长度为 Δt。元组中 $F_n^{\text{avail}}(t)$、$M_n^{\text{avail}}(t)$、$S_n^{\text{avail}}(t)$ 分别表示节点在时隙 t 的计算资源、内存资源和存储容量。链路 $(i,j)\in E$ 的带宽为 $B_{ij}(t)$。

每个算力节点被视作一个智能体，负责管理邻

近节点并生成调度决策，智能体集合用$\mathcal{I}=\{1,2,\cdots,I\}$表示。任务在时隙$t$到达边缘节点后，由关联智能体决定在本地执行或调度至相邻节点。

1.2　任务模型

一个到达任务可能包含多个子任务，子任务之间可能存在依赖关系。如图2所示，一个到达任务$\tau(t)$的子任务构成一个有向无环图$\mathcal{G}_{\text{task}}=(V_{\text{task}},E_{\text{task}})$，子任务集合$\wp(t)=\{p_1(t),p_2(t),\cdots,p_m(t)\}$，其中$p_k(t)$表示任务$\tau(t)$的第$k$个子任务，任务$\tau(t)$的总截止时间为$D_{\text{task}}$。

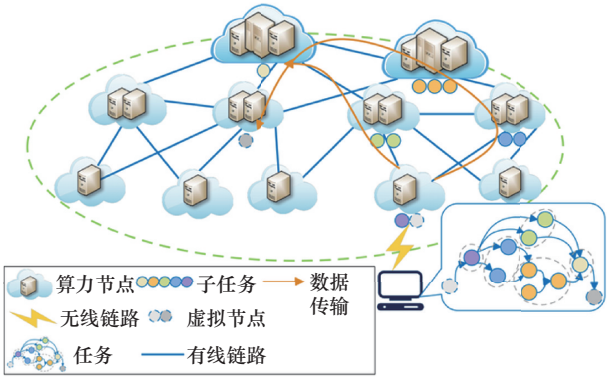


图2　DAG任务模型

任务$\tau(t)$的子任务$p_k(t)$属性集合表示为$\text{Attr}(p_k(t))=\{f_k(t),m_k(t),s_k(t),d_k(t)\}$，分别对应计算资源、内存需求、存储需求和输入数据量，其截止时间$t_k^{\text{ddl}}(t)$，由任务总截止时间D_{task}分配得到。假设$v_{\text{dst}}\in\{v_1,v_2,\cdots,v_N\}$，当任务$\tau(t)$的子任务$p_k(t)$被分配到节点$v_{\text{dst}}$执行时，节点分配给子任务的计算资源表示为$f_{k,\text{dst}}^{\text{alloc}}(t)$，节点分配给子任务的内存资源表示为$m_{k,\text{dst}}^{\text{alloc}}(t)$，节点为子任务分配的存储资源表示为$s_{k,\text{dst}}^{\text{alloc}}(t)$。

节点v_{dst}的可用资源随时间变化可表示为：

$$F_{\text{dst}}^{\text{avail}}(t+1)=F_{\text{dst}}^{\text{avail}}(t)-\sum_{k=1}^{m}f_{k,\text{dst}}^{\text{alloc}}(t)+\Delta F_{\text{dst}}(t)\quad(2)$$

$$M_{\text{dst}}^{\text{avail}}(t+1)=M_{\text{dst}}^{\text{avail}}(t)-\sum_{k=1}^{m}m_{k,\text{dst}}^{\text{alloc}}(t)+\Delta M_{\text{dst}}(t)\quad(3)$$

$$S_{\text{dst}}^{\text{avail}}(t+1)=S_{\text{dst}}^{\text{avail}}(t)-\sum_{k=1}^{m}s_{k,\text{dst}}^{\text{alloc}}(t)+\Delta S_{\text{dst}}(t)\quad(4)$$

其中，$F_{\text{dst}}^{\text{avail}}(t)$、$M_{\text{dst}}^{\text{avail}}(t)$、$S_{\text{dst}}^{\text{avail}}(t)$表示节点在上一时隙的可用资源，$\Delta F_{\text{dst}}(t)$、$\Delta M_{\text{dst}}(t)$、$\Delta S_{\text{dst}}(t)$表示资源回收或外部补充。

1.3　时延与能耗模型

（1）传输时延

对于子任务$p_k(t)$，其数据量为$d_k(t)$，从源节点v_{src}到目标节点v_{dst}的路径为$P=\{v_1,v_2,\cdots,v_u\}$，其中$v_1=v_{\text{src}},v_u=v_{\text{dst}}$，则子任务从节点$v_{\text{src}}$到节点$v_{\text{dst}}$的传输时延可表示为[14]：

$$T_k(P,d_k(t))=d_k(t)\cdot\sum_{s=1}^{u-1}\frac{1}{B_{s,\,s+1}(t)}+\sum_{s=1}^{u-1}D_{s,s+1}\quad(5)$$

采用链路无拥塞的传输时延模型，即链路带宽$B_{s,\,s+1}$视为在时隙内可用且不随业务队列动态变化。链路拥塞引起的排队时延与动态带宽分配不在式(5)中显式建模。在拥塞场景下，$B_{s,\,s+1}$为链路的有效带宽，$D_{s,s+1}$为固定传输时延。

（2）计算时延

子任务$p_k(t)$在节点v_{dst}上的计算时延可表示为：

$$T_{k,\text{dst}}^{\text{com}}(t)=\frac{f_k(t)}{f_{k,\text{dst}}^{\text{alloc}}(t)\cdot\eta_{\text{dst}}}\quad(6)$$

其中，η_{dst}表示节点v_{dst}的计算效率因子，如GPU对矩阵运算的效率$\eta_{\text{dst}}=0.9$。

（3）总时延

用二元组$x_{k,\text{dst}}(t)\in\{0,1\}$表示子任务$p_k(t)$在时隙$t$的分配决策，若$x_{k,\text{dst}}(t)=1$，则子任务$p_i(t)$分配到节点$v_{\text{dst}}$；若$x_{k,\text{dst}}(t)=0$，则不分配。一个子任务$p_k(t)$只能被分配给一个节点，因此有约束：

$$\sum_{\text{dst}=1}^{N}x_{k,\text{dst}}(t)=1,\forall p_k(t)\in\wp(t)\quad(7)$$

子任务$p_k(t)$的总时延由通信时延、计算时延和依赖任务等待时延组成。

$$T_k^{\text{tot}}(t)=T_k(P,d_k(t))+T_{k,\text{dst}}^{\text{com}}(t)+\max_{p_j\in\text{Pre}(p_k)}T_j^{\text{tot}}(t)\quad(8)$$

其中，子任务p_j是p_k的前序子任务。p_k必须等待所有前序子任务全部完成后才能执行，$T_j^{\text{tot}}(t)$表示前序子任务完成的最大时间，可看作等待时延。

任务总执行时间为所有子任务完成时间的最大值，反映了系统处理依赖感知任务的总体时延。

$$T(t)=\max_{p_k(t)\in\wp(t)}\left(x_{k,\text{dst}}(t)\cdot T_k^{\text{tot}}(t)\right)\quad(9)$$

（4）能耗

任务总能耗为所有子任务的计算能耗与通信能耗之和[15]。

$$E(t)=\sum_{k=1}^{m}x_{k,\mathrm{dst}}(t)\Big(\omega_{\mathrm{dst}}(t)\cdot f_{k,\mathrm{dst}}^{\mathrm{alloc}}(t)\cdot T_{k,\mathrm{dst}}^{\mathrm{com}}(t)+P_{\mathrm{dst}}^{\mathrm{tx}}(t)\cdot T_k\big(P,d_k(t)\big)\Big) \tag{10}$$

其中，$\omega_{\mathrm{dst}}(t)$是节点v_{dst}在时隙t的单位计算能耗，$P_{\mathrm{dst}}^{\mathrm{tx}}(t)$是节点$v_{\mathrm{dst}}$在时隙$t$的通信功率，随链路带宽$B_{ij}$动态调整。

能效是单位能耗处理的有效任务量，表示为[16]：

$$U(t)=\frac{\sum_{k=1}^{m}d_k(t)}{E(t)} \tag{11}$$

1.4 问题公式化

本文的优化目标是联合最小化任务完成的时间和最大化能源效率，相应的优化问题$\mathcal{P}1$表述如下。

$$\mathcal{P}1:\min_{\boldsymbol{X}(t),\boldsymbol{F}(t),\boldsymbol{M}(t),\boldsymbol{S}(t)}\frac{1}{T}\cdot\sum_{t=1}^{T}\left(\eta_T(t)\cdot T(t)+\eta_E(t)\cdot\frac{1}{U(t)}\right) \tag{12}$$

s.t. 式(2)~式(4)、式(7)

$$T_j^{\mathrm{tot}}(t)\leqslant T_k^{\mathrm{tot}}(t)-T_{k,\mathrm{dst}}^{\mathrm{com}}(t),\forall p_j\in\mathrm{Pre}\big(p_k\big) \tag{12a}$$

$$x_{k,\mathrm{dst}}(t)\cdot T_k^{\mathrm{tot}}(t)\leqslant t_k^{\mathrm{ddl}}(t),\forall k,\mathrm{dst},t \tag{12b}$$

$$\sum_{k=1}^{m}x_{k,\mathrm{dst}}(t)\cdot f_{k,\mathrm{dst}}^{\mathrm{alloc}}(t)\leqslant F_{\mathrm{dst}}^{\mathrm{avail}}(t),\forall\mathrm{dst},t \tag{12c}$$

$$\sum_{k=1}^{m}x_{k,\mathrm{dst}}(t)\cdot m_{k,\mathrm{dst}}^{\mathrm{alloc}}(t)\leqslant M_{\mathrm{dst}}^{\mathrm{avail}}(t),\forall\mathrm{dst},t \tag{12d}$$

$$\sum_{k=1}^{m}x_{k,\mathrm{dst}}(t)\cdot s_{k,\mathrm{dst}}^{\mathrm{alloc}}(t)\leqslant S_{\mathrm{dst}}^{\mathrm{avail}}(t),\forall\mathrm{dst},t \tag{12e}$$

其中，$\eta_T(t)$和$\eta_E(t)$分别为任务完成时间和能源效率的动态权重系数，且$\eta_T(t)+\eta_E(t)=1$，$\eta_T(t),\eta_E(t)\in(0,1)$，定义时延压力与能效压力为$u_T(t)=\mathrm{clip}\left(\frac{T(t)}{T_{\max}},0,1\right)$、$u_E(t)=\mathrm{clip}\left(\frac{\left(\frac{1}{U(t)}\right)}{U_{\max}^{-1}},0,1\right)$，$T_{\max}$与$U_{\max}$为归一化常数。进一步采用Softmax得到动态权重$\eta_T(t)=\frac{\exp(\lambda u_T(t))}{\exp(\lambda u_T(t))+\exp(\lambda u_E(t))}$、$\eta_E(t)=1-\eta_T(t)$，其中，$\lambda$为敏感度系数。当系统时延压力增大时，$\eta_T(t)$自动增大以强调时延目标；当能效压力增大时，$\eta_E(t)$自动增大以强调能效目标。

式(12a)表示子任务的依赖关系约束，即前序任务必须在当前任务开始执行前完成；式(12b)表示子任务要在其截止时间前完成；式(12c)~式(12e)确保节点分配给某个子任务的计算、内存和存储资源不超过该节点的可用资源。

2 依赖感知任务重构算法

2.1 算法概述

为降低复杂DAG任务的调度难度，本文提出依赖感知任务重构算法。在保持关键依赖关系的前提下，通过层索引计算、三角依赖解耦及阈值约束合并，将原始DAG转化为仅包含相邻层依赖的简化结构，从而减少子任务数量及跨节点通信开销[17]。

算法1 依赖感知任务重构算法

输入 任务$\tau(t)$原始任务图$\mathcal{G}_{\mathrm{task}}=\big(V_{\mathrm{task}},E_{\mathrm{task}}\big)$，阈值向量$\boldsymbol{\sigma}=\big(\sigma_f,\sigma_m,\sigma_s\big)$，任务总截止时间$D_{\mathrm{task}}$。

输出 重构图$\tilde{G}=(\tilde{V},\tilde{E}),\widetilde{\mathrm{Attr}}=\{\tilde{f}_k,\tilde{m}_k,\tilde{s}_k,\tilde{d}_k,t_k^{\mathrm{ddl}},\lambda_k\}$。

1) 初始化所有子任务的层索引$\lambda_k=0$，$\lambda_1=1$
2) 基于G_{task}前驱关系计算各子任务层索引λ_k
3) $\hat{G}=\big(\hat{V},\hat{E}\big)\leftarrow\mathcal{G}_{\mathrm{task}}$
4) 从$\hat{G}$中识别所有非相邻层边集合$\hat{E}_1$
5) while $\hat{E}_1\neq\varnothing$ do
6) 选择具有最小$\hat{\lambda}_v$的非相邻层边$e(u,v)$
7) 若有多个，则选取最小$\hat{\lambda}_u$对应的边
8) 找到层$\hat{\lambda}_u$与$\hat{\lambda}_v$之间的候选边集合$\hat{E}_2$
9) 对$\hat{E}_2$中子任务顺序合并，得到图$G_{\mathrm{task}}^{\mathrm{tmp}}$
10) 若$G_{\mathrm{task}}^{\mathrm{tmp}}$可使非相邻层边数减少
11) 则用其更新$\hat{G}$
12) 重新计算$\hat{G}$中各子任务层索引，更新$\hat{E}_1$
13) end while
14) $\tilde{G}=(\tilde{V},\tilde{E})\leftarrow\hat{G}$
15) repeat
16) 从$\tilde{G}$中选最小资源需求score_u的子任务u
17) 构造其前置、后继和并行的候选子任务集合
18) 按$\mathrm{score}_v=\frac{1}{3}\frac{f_v}{\sigma_f}+\frac{1}{3}\frac{m_v}{\sigma_m}+\frac{1}{3}\frac{s_v}{\sigma_s}$和合并类型标志flag升序排序$\tilde{V}_1$
19) 依次选择候选子任务v
20) 若顺序合并或并行合并后满足$\tilde{f}_w<\sigma_f,\tilde{m}_w<$

$\sigma_m,\tilde{s}_w < \sigma_s$，则更新$\tilde{G}$及$\widetilde{\text{Attr}}$

21) until不存在满足阈值约束的可合并子任务

22) 在$\tilde{G}$中计算从v_{start}到v_{end}的关键路径P_{critical}及其总时延T_{critical}

23) 按$t_k^{\text{ddl}} = D_{\text{task}} \times \dfrac{T_{\text{path}(\text{start}\to k)}}{T_{\text{critical}}}$分配各子任务局部截止时间，其中，$T_{\text{path}(\text{start}\to k)}$为$v_{\text{start}}$到$k$的所有路径中最长的那条

24) 输出重构图$\tilde{G}$及优化后的子任务属性$\widetilde{\text{Attr}}$

2.2　层索引计算

首先定义分层任务和子任务层索引。对于具有一个开始子任务和一个结束子任务的多组件任务，可以将多个子任务聚类到一个层中，以构建原始任务DAG的分层形式，使分层任务满足以下属性。

（1）保持分层任务的连接性，即子任务之间的边，与原始任务保持不变。

（2）所有层都是根据不同层中子任务之间的边顺序组织的。

（3）任务的边仅存在于从前层中的子任务到后层中的子任务，不存在从后层到前层的边。

对于子任务$k \in V_{\text{task}}$，其所属层的索引λ_k定义为：

$$\lambda_k = \begin{cases} 1, \text{Pre}(k) = \varnothing \\ \max \lambda_p \mid p \in \{\text{Pre}(k)+1\}, \text{Pre}(k) \neq \varnothing \end{cases} \tag{13}$$

其中，$\text{Pre}(k)$表示k的所有直接前置子任务集合，层索引反映了子任务在依赖链中的拓扑度。

如算法1前两行所示，首先依据任务图中的前驱依赖关系计算各子任务的层索引λ_k。对于起始子任务，令其层索引为1，对于其余子任务，其层索引由所有前驱子任务层索引的最大值加1确定。为保证层索引自前向后逐层生成，采用基于广度优先搜索（breadth-first search, BFS）的方式递推计算各子任务层索引。

2.3　三角依赖解耦

定义非相邻层边，对于边$e(u,v) \in E_{\text{task}}$，若$|\lambda_v - \lambda_u| \geqslant 2$，则称为非相邻层边。非相邻层边的存在是产生三角依赖的根源。

定义纯顺序分层任务，用任务图$\hat{G} = (\hat{V},\hat{E})$表示一个纯顺序分层的任务，当且仅当：$\hat{G}$为有向无环图，且已被构造为分层任务；任意边$e(u,v) \in \hat{E}$，有$|\lambda_v - \lambda_u| = 1$，即仅相邻层间存在边，不存在超过两层的边。

纯顺序分层任务可能与原始任务DAG不同，子任务的数量减少，任务边仅存在于相邻层之间，如图3(e)所示。

三角依赖解耦的任务DAG由仅包含顺序依赖关系和并行依赖关系的子任务组成。与原始任务DAG相比，解耦后的任务DAG没有非相邻层边，这将降低任务分配的复杂度。

如算法1第3)~第11)行所示，首先在原始任务图中识别所有非相邻层边。对于任意非相邻层边$e(u,v)$，若其连接的两个子任务位于非相邻层，则说明任务图中存在跨层依赖关系。为消除此类依赖，在λ_u与λ_v之间搜索候选边集合，并通过顺序合并中间相关子任务，优先选择能够减少非相邻层边数量的方案更新任务图。重复该过程，直至图中不存在非相邻层边，从而得到纯顺序分层任务图。

三角依赖示例如图3所示。图3(c)中的边$e(2,5)$为一条非相邻层边。为消除子任务2、3和5之间的三角依赖，将子任务3和5合并，并更新得到图3(d)。在此基础上，继续递归处理边$e(2,4)$和$e(4,5)$。对于合并后的子任务，按照算法选择所需计算资源较小的结果，最终得到图3(e)所示的更新图。

2.4　阈值约束合并

在完成三角依赖解耦后，为进一步降低任务图规模，本文在资源阈值约束下继续对子任务进行合并。子任务合并有顺序合并和并行合并两种方式。两个子任务的合并不会改变原始子任务的原子属性，但这两个子任务在同一算力节点上执行计算操作，可以减少子任务和边的数量。为了防止子任务的无约束合并，并最终形成具有大量资源需求的子任务，子任务在合并过程中受到阈值$\boldsymbol{\sigma}$的限制。

定义阈值$\boldsymbol{\sigma} = (\sigma_f,\sigma_m,\sigma_s)$，约定重构过程中子任务的资源需求。

（1）σ_f：单个子任务的最大计算需求，不超过单节点计算容量。

（2）σ_m：单个子任务的最大内存需求，不超过单节点内存容量。

（3）σ_s：单个子任务的最大存储需求，不超过

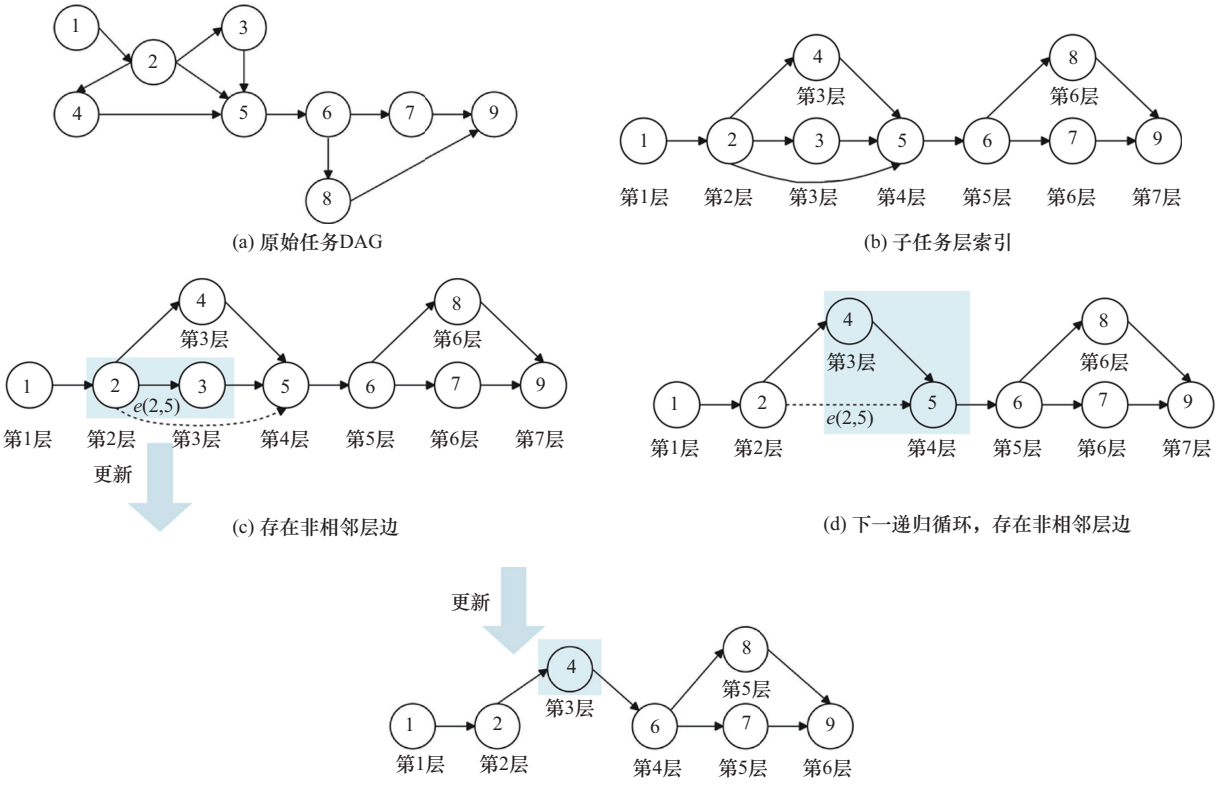


图3　三角依赖示例

单节点存储容量。

顺序合并首先找到要合并的顺序子任务u和v，并删除子任务之间的边$e(u,v)$进行合并。其后，合并连续的子任务，并将合并后的子任务重新索引为u。最后，因为资源通常是峰值占用容量，顺序合并时不重叠执行，所以资源需求取最大值，子任务的资源需求$\tilde{f}_w$、$\tilde{m}_w$和$\tilde{s}_w$分别更新为$\max(f_u,f_v)$，$\max(m_u,m_v)$和$\max(s_u,s_v)$，输入数据量$\tilde{d}_w$更新为d_u+d_v。因为顺序合并可以将两个子任务合并为一个子任务，所以原始的两个子任务可以被视为放置在同一个算力节点上，传输时延为0。例如，任务$p_k(t)$中的子任务$u=2$是子任务$v=3$的前置任务。从DAG中删除$e(2,3)$，然后合并子任务$u=2$和$v=3$，并将合并后的子任务重新索引为$w=2$，$\lambda_w=\lambda_u$。最后，更新子任务的资源请求，即$\tilde{f}_2\leftarrow\max(f_2,f_3)$、$\tilde{m}_2\leftarrow\max(m_2,m_3)$、$\tilde{s}_2\leftarrow\max(s_2,s_3)$和输入数据量$\tilde{d}_2\leftarrow d_2+d_3$，如图4(a)所示。

并行合并首先要识别同一层中的并行子任务u和v。删除并行子任务与其前一个子任务之间的边$e(s,u)$和$e(s,v)$，以及并行子任务与其后继子任务之间的边$e(u,m)$和$e(v,m)$，以方便合并。合并并行子任务，并将合并的子任务重新索引为w，$\lambda_w=\lambda_u$，同时引入新的边$e(s,w)$和$e(w,m)$。最后，更新资源需求和输入数据量：$\tilde{f}_w\leftarrow f_u+f_v$、$\tilde{m}_w\leftarrow m_u+m_v$、$\tilde{s}_w\leftarrow s_u+s_v$和$\tilde{d}_w\leftarrow d_u+d_v$。

并行合并如图4(b)所示，在同一层中找到子任务$u=2$和$v=3$，即$\lambda_2=\lambda_3=2$。移除边$e(1,2)$、$e(1,3)$、$e(2,4)$和$e(3,4)$。合并并行子任务$u=2$和$v=3$，将合并后的子任务重新索引为2，并添加新的边$e(1,2)$和$e(2,4)$。最后，因为并行执行需要同时占用资源，所以资源需求求和：$\tilde{f}_2\leftarrow f_2+f_3$、$\tilde{m}_2\leftarrow m_2+m_3$、$\tilde{s}_2\leftarrow s_2+s_3$和$\tilde{d}_2\leftarrow d_2+d_3$。

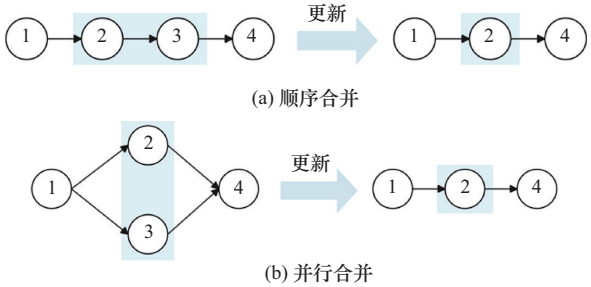


图4　顺序合并和并行合并示例

子任务合并如算法 1 第 13)~第 17)行所示，首先选择资源需求最小的子任务 u 作为当前待合并对象，并构造其前置、后继及并行候选子任务集合。随后，依据 $\text{score}_v = \frac{1}{3}\frac{f_v}{\sigma_f} + \frac{1}{3}\frac{m_v}{\sigma_m} + \frac{1}{3}\frac{s_v}{\sigma_s}$ 以及合并类型标志（flag）对候选子任务进行排序，优先尝试资源增量较小的合并操作。若候选子任务 v 为 u 的前置或后继子任务，则执行顺序合并；若 v 为 u 的并行子任务，则执行并行合并。每次合并后更新子任务属性及任务图结构，只有当合并后新子任务的计算资源、缓存资源和存储资源需求均满足阈值约束时，才接受该次合并；否则转向下一个候选子任务。重复上述过程，直至不存在可行合并对象。

2.5　子任务截止时间分配

如算法 1 最后两行所示，在重构图 $\tilde{G}$ 上计算从起始子任务 v_{start} 到结束子任务 v_{end} 的路径集合中，时延+传输时延最大的路径 P_{critical} 及其总时延 T_{critical}，并根据子任务所在路径时延占关键路径时延的比例，为各子任务分配局部截止时间 t_k^{ddl}。

$$t_k^{\text{ddl}} = D_{\text{task}} \times \frac{T_{\text{path(start}\to k)}}{T_{\text{critical}}} \tag{14}$$

3　基于 RGAT 的多智能体协同分配算法

3.1　算法概述

针对算力网络中多维资源关系复杂和大规模分布式协作问题，本文提出一种基于 RGAT 的多智能体协同分配算法，结合 RGAT 与 MASAC 实现高效分布式决策。前者通过为计算、内存、存储等资源分配独立注意力权重，并结合消息传递更新节点隐藏状态，提升智能体对资源可用性和任务适配性的判断能力；后者基于最大熵强化学习，增强策略探索能力与稳定性，避免陷入局部最优。由于注意力权重计算依赖节点间相对关系特征而非全局固定索引，所提算法能够较好适应动态环境下的异构资源协同。整体采用“离线训练–在线分布式推理”[18]两阶段模式，兼顾训练充分性与在线实时性。

基于 RGAT 的多智能体协同分配算法框架如图 5 所示，包括任务重构、关系图表征和多智能体决策 3 个阶段。首先，利用依赖感知任务重构算法对原始 DAG 进行层索引计算、三角依赖解耦和阈值约束合并，输出重构子任务集合及其局部截止时间，以降低依赖复杂度和跨节点传输开销。随后，各智能体基于本地资源状态、队列状态及 k 跳邻域拓扑构建关系特征图，通过 RGAT 生成节点表征，并结合双向长短期记忆（long short-term memory, LSTM）网络、近似离散化采样（Gumbel-Softmax）和表征融合模块得到综合环境表征。最后，将其输入 MASAC 框架，在离线训练与在线分布式推理模式下完成任务分配与资源分配决策，实现动态环境下的高效协同调度。

3.2　关系图表征生成算法

（1） 关系特征图

本节构建关系特征图表征多智能体环境，通过 RGAT 进行特征嵌入，捕获智能体相邻算力节点的资源状态与拓扑关系，帮助智能体进行交互捕捉及算力资源分配策略学习。

构建算力网络的关系特征图为图 $G(t) = <V,E,R>$，其中，V 表示所有算力节点集合，

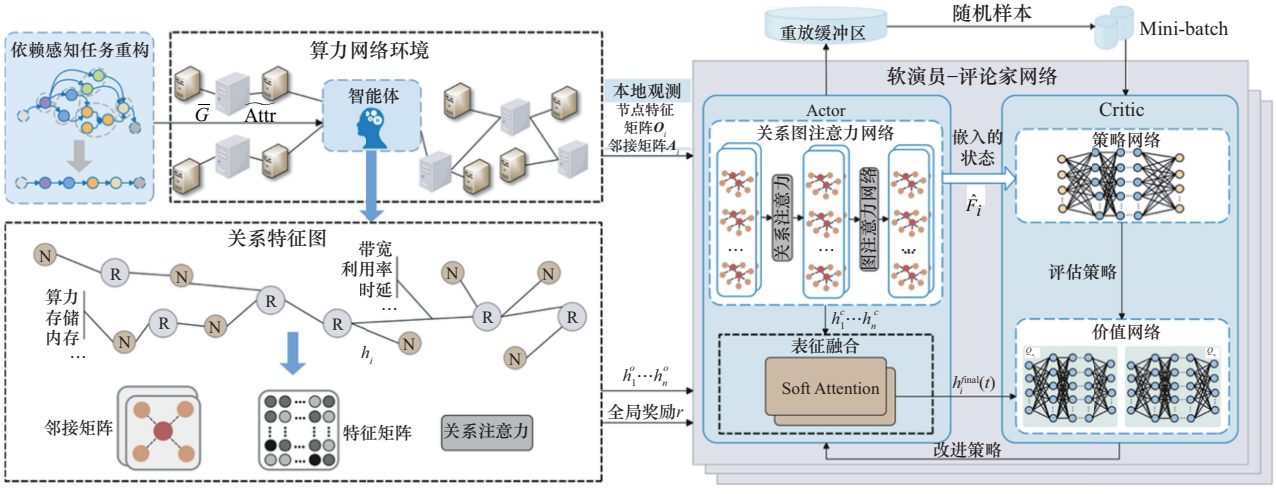


图 5　基于 RGAT 的多智能体协同分配算法框架

$(v_i,v_j)\in V$，每个节点关联特征向量$\boldsymbol{h}_i(t)$。每条边$\left(v_i,v_j\right)\in E$表示节点$v_i$与$v_j$的连接。定义多维资源关系类型$R=\{r_{\text{comp}},r_{\text{mem}},r_{\text{sto}},r_{\text{topo}}\}$，其中$\{r_{\text{comp}},r_{\text{mem}},r_{\text{sto}},r_{\text{topo}}\}$分别表示计算资源、内存资源、存储资源和网络拓扑关系。每个节点v_i的特征向量$\boldsymbol{h}_i(t)$包含动态资源状态和拓扑信息。

$$\boldsymbol{h}_i(t)=\left[F_i^{\text{avail}}(t),M_i^{\text{avail}}(t),S_i^{\text{avail}}(t),q_i(t),B_{ij}(t),D_{ij}\right] \tag{15}$$

其中，$q_i(t)$是节点v_i在时隙t的任务队列状态，由算法1输出的重构图与子任务属性构造，在时隙t根据重构图得到当前可调度子任务集合$\text{Ready}(t)$，并对$\left\{\tilde{x}_i|i\in\text{Ready}(t)\right\}$进行编码得到队列状态$q_i(t)$，从而将依赖结构与局部截止时间等信息注入节点特征。

节点i的特征通过关系特定的线性变换映射到嵌入空间。

$$\boldsymbol{h}_i^r(t)=\boldsymbol{W}_r\cdot\boldsymbol{h}_i(t) \tag{16}$$

其中，$\boldsymbol{W}_r\in\mathbb{R}^{d_{\text{hidden}}\times d_{\text{in}}}$为关系$r$的权重矩阵。

（2）关系图注意力网络

将原始局部观测状态o_i转化为节点特征矩阵$\boldsymbol{O}_i$，$\boldsymbol{A}_i$表示为邻接矩阵。嵌入过程遵循消息传递网络框架。对于$\forall v_k,v_j\in V$和$\forall\left(v_k,v_j\right)\in E$，图$G_i$中消息传递的更新规则为：

$$f_{i,k}^{l_{\text{net}}}=\phi_i^{l_{\text{net}}}\left(f_{i,k}^{l_{\text{net}}-1},\sum_{r=1}^{R}\sum_{j\in\mathcal{N}_r(k)}\alpha_{kj}^r\cdot\psi_i^{l_{\text{net}}}\left(f_{i,j}^{l_{\text{net}}-1},e_{k,j}^r\right)\right) \tag{17}$$

其中，$f_{i,k}^l$表示图G_i的第l_{net}层网络中节点k的隐藏状态；$\mathcal{N}_r(k)$表示节点k在关系类型r下的邻居集合；$\alpha_{kj}^r\in[0,1]$为注意力权重；$l_{\text{net}}\in\mathcal{L}$为RGAT的层数，第$l_{\text{net}}$层的输出表示为$F_i=[f_{i,1}^l,f_{i,2}^l,\cdots,f_{i,|V_i|}^l]$，它整合了与智能体$i$的局部观测信息的潜在特征；$\psi_i^{l_{\text{net}}}(\cdot)$表示关系特定的特征变换函数；$\phi_i^{l_{\text{net}}}(\cdot)$将局部隐藏状态与聚合特征相结合。经过图嵌入模块后，智能体i基于局部观测图G_i的嵌入状态$\hat{F}_i=\left[\boldsymbol{O}_i,F_i\right]$对任务$p_k(t)$进行协同分配。为将可变规模的节点嵌入用于策略网络输入，对$F_i(t)$进行池化得到固定维度资源表征。

$$h_i^o(t)=\text{Pool}(F_i(t)) \tag{18}$$

RGAT先通过双向LSTM评估智能体交互重要性。

$$h_{i,j}(t)=f\left(\text{BiLSTM}\left(\left[h_i^o(t)\|h_j^o(t)\right]\right)\right) \tag{19}$$

其中，h_i^o和h_j^o分别表示智能体i和智能体j的观测表征，$f(\cdot)$是前馈神经网络层，$\boldsymbol{h}_{i,j}$是二元类别向量。进一步结合Gumbel-Softmax生成可微分相关性权重，并对邻居智能体表征进行加权聚合，得到智能体i的交互表征$h_i^c(t)$。该过程能够在保持梯度可传播的同时，实现对关键邻居信息的有效筛选与聚合。

（3）表征融合

表征融合模块对资源特征表征$h_i^o(t)$、智能体交互表征$h_i^c(t)$和智能体上下文表征h_i^k进行联合建模，以同时保留资源关系信息、邻居交互信息和智能体自身状态信息。其中，上下文表征h_i^k由节点原始特征向量$\boldsymbol{h}_i(t)$经过前馈网络编码得到。

随后，采用自注意力机制对$h_i^o(t)$、$h_i^c(t)$和h_i^k进行加权融合，并结合残差连接与层归一化得到综合环境表征$h_i^{\text{final}}(t)$。其中，$h_i^{\text{final}}(t)$作为后续MA-SAC中Actor和Critic的输入表征，用于支持局部观测条件下的分布式任务分配与资源分配决策。关系图表征生成算法的伪代码如算法2所示。

算法2 关系图表征生成算法

输入 $\boldsymbol{O}_i,\boldsymbol{A}_i,\phi_i^{l_{\text{net}}},\psi_i^{l_{\text{net}}},\alpha_{kj}^r$,BiLSTM, $f(\cdot)$, Gumbel-Softmax, F_2, h_i^k, LayerNorm

输出 智能体i的综合环境表征h_i^{final}

1) for each 智能体 i do
2) 基于局部观测状态构建关系特征图G_i，并生成节点特征矩阵$\boldsymbol{O}_i$和邻接矩阵$\boldsymbol{A}_i$
3) $F_i(t)\leftarrow\text{RGAT}\left(\boldsymbol{O}_i(t),\boldsymbol{A}_i(t)\right)$
4) $h_i^o(t)\leftarrow\text{Pool}(F_i(t))$
5) for each 智能体 $j\neq i$ do
6) $h_{i,j}(t)\leftarrow f\left(\text{BiLSTM}\left(\left[h_i^o(t)\|h_j^o(t)\right]\right)\right)$
7) end for
8) 依据相关性权重聚合邻居智能体表征，得到交互表征$h_i^c(t)$
9) 结合上下文表征h_i^k，对$h_i^o(t)$和$h_i^c(t)$进行融合，得到$h_i^{\text{final}}(t)$
10) end for
11) return $h_i^{\text{final}}(t)$

3.3　基于MASAC的任务分配算法

算法1在任务到达时由任务入口节点对应的智能体执行一次，生成重构图、重构后的子任务属性及局部截止时间，并将重构结果同步给参与该任务协同调度的相关智能体。分布式执行阶段，各智能体不再重复从原始DAG运行算法1，而是依据共享的重构结果以及前驱任务完成反馈，更新当前可调度子任务集合。

算法2为在线局部表征生成模块，在每个决策时隙由各智能体基于本地资源状态、任务队列状态以及k-跳邻域拓扑与资源信息独立运行，输出综合环境表征，作为算法3中MASAC的Actor和Critic输入。同一时隙内，同一子任务仅由当前持有该子任务的唯一智能体调用本地Actor输出最终分配动作，其他智能体仅提供邻域状态信息和前驱完成反馈，不参与重复决策。RGAT负责状态表征，根据智能体i的本地观测、任务队列状态以及k跳邻域资源、拓扑信息构建关系特征图，并生成环境表征；MASAC负责决策优化，其中Actor网络以环境表征为输入，输出当前负责子任务的目标执行节点及资源分配动作，Critic网络仅在训练阶段用于联合状态–动作价值评估与参数更新。因此，在在线分布式执行阶段，具体子任务的分配结果由对应决策主体的Actor网络输出，RGAT本身不直接生成分配动作。

在算力网络中，广泛分布的智能体受通信与计算能力限制，难以获取全局信息，需基于局部观测决策，因此任务分配问题被建模为Dec-POMDP，以捕捉分布式智能体的局部观测与协作需求。Dec-POMDP模型表示为元组$<I,S,O,A,P,R,\delta>$，其中，I为智能体集合，S为状态空间，O为观测空间，A为动作空间。在执行动作后，转移到新状态的概率记为P，获得的奖励为R，折扣因子为δ。

（1）状态空间。在时隙t，每个智能体i的状态为：

$$s(t)=\left(F_i^{\text{avail}}(t),M_i^{\text{avail}}(t),S_i^{\text{avail}}(t),B_{ij}(t),D_{ij},q_i(t),t_k^{\text{ddl}}(t),F_i(t)\right) \tag{20}$$

（2）动作空间。在每个时隙t，每个智能体需要决定当前子任务$p_k(t)$分配到哪个算力节点上，并为其分配计算$f_{k,i}^{\text{alloc}}(t)$、内存$m_{k,i}^{\text{alloc}}(t)$、存储资源$s_{k,i}^{\text{alloc}}(t)$。每个智能体的动作可表示为：

$$a(t)=\left(x_{k,i}(t),f_{k,i}^{\text{alloc}}(t),m_{k,i}^{\text{alloc}}(t),s_{k,i}^{\text{alloc}}(t)\right) \tag{21}$$

（3）观测空间。智能体i在时隙t的局部观测$o_i(t)$为该节点可用计算$F_i^{\text{avail}}(t)$、内存$M_i^{\text{avail}}(t)$、存储资源$S_i^{\text{avail}}(t)$、本节点维护的待调度任务队列状态$q_i(t)$、k-跳邻域内节点与链路的资源和拓扑状态$B_{ij}(t),D_{ij}$、由算法1生成并同步得到的重构图$\tilde{G}$、子任务属性$\widetilde{\text{Attr}}$及局部截止时间信息$t_k^{\text{ddl}}(t)$。在线分布式执行不要求智能体获得全网实时状态，全局信息仅体现在离线训练过程中。

（4）奖励。根据$\mathcal{P}1$，优化目标是最小化任务执行时间和最大化能效，在奖励函数中加入对能耗的惩罚$\rho_i(t)$，具体设计为：

$$r_i(t)=\frac{h_1}{T(t)}+h_2\cdot U(t)+h_3\rho_i(t) \tag{22}$$

其中，常系数h_1和h_2用于平衡任务完成时间和能效惩罚之间的影响。此外，系数h_3可以在实际中调整，以满足能耗约束。例如，如果能耗大于系统要求的最高能耗，会增加相应的惩罚系数h_3，导致奖励减少。因此，智能体有动力进一步探索和决策满足约束条件的策略，以追求最大的收益。具体地，时隙t内的惩罚函数为：

$$\rho_i(t)=\begin{cases}0, & E(t)\leqslant E_{\max}\\ E_{\max}-E(t), & E(t)>E_{\max}\end{cases} \tag{23}$$

由于所有智能体都以学习如何最大化累积收益为目标，因此$\mathcal{P}1$可以转化为：

$$\mathcal{P}2:\ \max r=\sum_{i}^{I}\sum_{t=0}^{\infty}\delta(t)r_i(t),\forall i\in I \tag{24}$$

本文算法采用了离线策略的演员–评论家框架。以最大化累积收益和策略最大熵为目标，激励智能体探索最优策略，避免陷入局部最优。

在每个时隙t，首先依据算法1输出的重构图$\tilde{G}$与子任务属性$\widetilde{\text{Attr}}$得到当前可调度子任务集合：

$$\text{Ready}(t)=\left\{k\in\tilde{V}\,\middle|\,\text{Done}(k)=0,\ \forall p\in\text{Pre}(k),\text{Done}(p)=1\right\} \tag{25}$$

编码成各智能体的任务队列状态。随后调用算法2生成综合环境表征$h_i^{\text{final}}(t)$，并将其作为MASAC的Actor和Critic输入以输出分配动作。通过采样小批量数据训练学习分配策略，算法最优随机策略π_i^*为期望累积奖励与熵的最大化。

（1）策略网络与动作生成

基于RGAT提取的表征$h_i^{\text{final}}(t)$，策略网络$\pi_i(\cdot|h_i^{\text{final}}(t))$生成智能体$i$的动作$a(t)$。动作空间包含离

散和连续两个部分，$x_{k,i}(t)\in\{0,1\}$决定是否将子任务p_k分配到算力节点i，$\{f_{k,i}^{\text{alloc}}(t),m_{k,i}^{\text{alloc}}(t),s_{k,i}^{\text{alloc}}(t)\}$决定为选定节点分配的计算、内存和存储资源量。

策略网络采用混合输出层：分类头通过Softmax函数输出节点选择概率分布，回归头输出资源分配量的高斯分布参数。训练时，从分布中采样动作$a_i(t)\sim\pi_i\left(\cdot|h_i^{\text{final}}(t)\right)$；推理时，选择概率最大的节点和期望资源分配量。

（2）最大熵强化学习目标

为激励智能体探索多样化的分配策略并避免陷入局部最优，本文采用最大熵强化学习框架。智能体i的优化目标不仅最大化累积收益，还最大化策略熵。

$$\pi_i^*=\arg\max_{\pi_i}\mathbb{E}_{s_i(t),a_i(t)}\sum_{t=0}^{\infty}\delta(t)\left(r_i(t+1)+\alpha_iH\left(\pi_i\left(\cdot|h_i^{\text{final}}(t)\right)\right)\right)\tag{26}$$

其中，熵项α_i是一个权重参数，它决定了策略熵在优化目标中的重要性。熵项定义为：

$$\mathcal{H}(\pi_i(\cdot|F_i(t)))=-\lg\left(\pi_i|h_i^{\text{final}}(t)\right)\tag{27}$$

其中，$F_i(t)$是RGAT从状态$s(t)$中提取的表征。

（3）策略评估

为降低Q值高估带来的影响，每个智能体维护两个Q网络$Q_{i,k}$及其目标网络。训练阶段，从经验回放缓冲区中采样小批量数据，利用双Q网络对联合状态–动作价值进行评估，并交替更新Critic参数、策略网络参数[19]和温度系数α_i。其中，Critic网络通过最小化软贝尔曼残差进行更新，Actor网络以最大化Q值与策略熵加权之和为目标进行更新，目标网络采用软更新方式进行参数同步。

通过交替进行策略评估和策略改进，算法迭代优化各智能体的策略网络和Q网络，直至收敛到最优策略π_i^*。基于MASAC的任务分配算法的伪代码如算法3所示。

算法3 基于MASAC的任务分配算法

输入 智能体I、重构图$\tilde{G}=(\tilde{V},\tilde{E})$、子任务属性$\widetilde{\text{Attr}}$、智能体输出观测表征$h_i^{\text{final}}(t)$、最大迭代次数$T_{\max}$、折扣因子$\delta$、批处理大小$b$、目标更新率$\tau$、学习率$\text{lr}_{\text{actor}}$、$\text{lr}_{\text{critic}}$、$\text{lr}_{\alpha}$和关系类型$R=\{r_{\text{com}},r_{\text{mem}},r_{\text{sto}},r_{\text{topo}}\}$

输出 优化后的策略$\{\pi_i^*\}_{i\in\mathcal{I}}$

1) 初始化每个智能体$i\in\mathcal{I}$的策略网络$\pi_i(\phi_i)$、双Q网络$Q_{i,k}(\theta_{i,k})$、目标网络、经验回放缓冲区$\mathcal{D}_i$、温度系数α_i及RGAT参数
2) for epoch = 1到$T_{\max}$ do
3) 根据$\tilde{G}$和$\widetilde{\text{Attr}}$初始化观测$o_i(t)$
4) 计算Ready(t)
5) for t = 0到$T-1$ do
6) for each智能体$i\in\mathcal{I}$ do
7) 基于$\tilde{G}$, $\widetilde{\text{Attr}}$, Ready(t), $o_i(t)$构建$\boldsymbol{O}_i$, $\boldsymbol{A}_i$，并调用算法2得到$h_i^{\text{final}}(t)$
8) 由策略网络采样动作$a_i(t)\sim\pi_i(\cdot|h_i^{\text{final}}(t))$
9) end for
10) 执行联合动作$\{a_i(t)\}_{i\in I}$，获得奖励$r_i(t)$、下一观测$o_i(t+1)$和终止标志Done
11) 更新Ready$(t+1)$
12) $\{h_i^{\text{final}}(t),a_i(t),r_i(t),o_i(t+1)\}$存入$\mathcal{D}_i$
13) if达到训练间隔且$|\mathcal{D}_i|\geqslant b$ then
14) 从$\mathcal{D}_i$中采样批次$\mathcal{B}_i$，更新双Q网络、策略网络π_i和温度系数α_i，并按τ软更新目标网络
15) end if
16) end for
17) end for
18) 在线执行时，根据$\tilde{G}$、$\widetilde{\text{Attr}}$和Ready(t)构造$o_i(t)$，调用算法2得到$h_i^{\text{final}}(t)$
19) 由对应决策主体采用确定性策略输出动作并执行联合动作，更新Ready$(t+1)$
20) Return优化后的策略$\{\pi_i^*\}_{i\in I}$

3.4 复杂度分析

（1）计算复杂度

算法1的复杂度主要由单个到达任务的DAG规模决定，而不随全网智能体数直接线性增长。设任务原始DAG包含$|V|$个子任务和$|E|$条依赖边，算法1的层索引计算复杂度为$O(|V|+|E|)$，非相邻层边识别复杂度为$O(|E|)$。三角依赖解耦与阈值约束合并涉及递归图更新、候选合并搜索与资源阈值判定，其总体复杂度为与$|V|$和$|E|$有关的多项式开销。故算法1对单个任务的总计算复杂度可记为$O(|V||E|+|V|^2)$。

算法2和算法3的计算开销主要集中在关系图表征生成和MASAC策略推理上。对智能体i，设

其K跳局部观测子图包含算力节点数$|V_i^{(K)}|$、边数$|E_i^{(K)}|$，关系类型数为$|R|$，嵌入维度为d，RGAT层数为l_{net}。在每一层中，关系特定线性变换与注意力计算/聚合主要沿边进行，RGAT总复杂度为$O\left(l_{\text{net}} \cdot |R| \cdot (|V_i^{(K)}|d^2 + |E_i^{(K)}|d)\right)$。由于$|V_i^{(K)}|$通常受限于局部通信跳数，该复杂度与算力节点总数解耦，因此具有良好的可扩展性。Actor网络的推理仅涉及矩阵乘法，网络隐藏层维度为d_{hidden}，复杂度为$O\left(d_{\text{hidden}}^2\right)$。因此，单步决策的总计算复杂度由RGAT主导，近似写为$O\left(l_{\text{net}} \cdot |R| \cdot (|V_i^{(K)}|d^2 + |E_i^{(K)}|d)\right)$，全网$I$个智能体的计算复杂度为$O\left(\sum_{i=1}^{I} l_{\text{net}} \cdot |R| \cdot (|V_i^{(K)}|d^2 + |E_i^{(K)}|d)\right)$。

（2）通信复杂度

对于单个到达任务，算法1的通信开销主要来自重构结果同步。设重构图包含$|V_r|$个重构子任务和$|E_r|$条依赖边，则单次同步的数据规模与重构图描述长度线性相关，可记为$O(|V_r| + |E_r|)$。若需向I个参与协同调度的智能体分别发送，则全网通信开销为$O(I(|V_r| + |E_r|))$。

在分布式执行阶段，智能体仅与K跳范围内的邻居节点交换资源状态信息。每个时隙t，智能体i需要接收邻居的状态向量维度为d_{in}，通信数据量为$O\left(|V_i^{(K)}| \cdot d_{\text{in}}\right)$，全网通信复杂度为$O\left(\sum_{i=1}^{I} |V_i^{(K)}| \cdot d_{\text{in}}\right)$。相比集中式算法需要收集全部节点的状态，本文算法的通信开销显著降低，仅与局部邻居数量线性相关，理论上适合大规模算力网络的部署。

4　仿真分析

4.1　仿真设置

本节设计消融实验与对比实验：消融实验对比移除重构模块、三角依赖解耦或子任务合并的变体算法，对比实验从任务完成时间和能效维度展开分析。训练以回合为单位迭代，单回合包含$T = 200$个离散时隙。最大训练回合数设置为300，经验回放池容量为1×10^6，批量大小为512。本文算法约在50个回合后收敛。为增强结果可重复性，所有训练与测试结果均在5个不同随机种子下独立重复实验取平均值。仿真基于PyTorch 2.5.1实现，算力网络拓扑建模为无向连通图$G = (V,E)$，其中$|V| = 50$。训练和测试均采用小世界网络模型生成固定拓扑：首先构造每个节点与环形邻域内6个近邻相连的规则网络，再以重连概率$p_r = 0.1$对边进行随机重连，直至图保持连通。在分布式决策过程中，智能体基于k跳邻域信息构造局部子图，并以邻接矩阵$\boldsymbol{A}_i(t)$作为关系特征图输入。固定传输时延设为$D_{ij} = [5,50]$ ms[10]，节点可用计算资源范围为[0.1,4] GHz。其余算力网络与RGAT相关参数如表1所示[20-22]。

表1　　算力网络仿真参数

参数	取值
智能体数量I/个	50
子任务k的计算需求$f_k(t)$/GHz	[1.0, 3.0]
子任务k的内存需求$m_k(t)$/MB	[100, 500]
子任务k的存储需求$s_k(t)$/GB	[0.5, 2.0]
子任务k的输入数据大小$d_k(t)$/MB	[10, 110]
节点i的可用计算资源$F_i^{\text{avail}}(t)$/GHz	[0.1, 4]
节点i的可用内存资源$M_i^{\text{avail}}(t)$/GB	[1, 8]
节点i的可用存储资源$S_i^{\text{avail}}(t)$/GB	[20, 80]
节点i和节点j之间的链路带宽$B_{ij}(t)$/(Mbit·s^{-1})	[200, 3 000]
节点i和节点j之间的传输时延$D_{ij}(t)$/ms	[5, 50]
单位计算能耗ω_{dst}/(J·FLOP^{-1})	7×10^{-5}
任务总截止时间D_{task}/ms	500
合并计算需求阈值σ_f/GHz	2.0
合并内存需求阈值σ_m/MB	300
合并存储需求阈值σ_s/GB	1.0
敏感度系数λ	5
最大迭代次数$T_{\max}$	10^6
批量大小b	512
折扣因子δ	0.95
网络的学习率lr	0.005
温度参数的学习率lr_α	0.001
重放缓冲区大小$\mathcal{D}$	10^6
RGAT网络中的层数l_{net}	3
隐藏层维度d_{hidden}	256
关系类型集R	r_{comp},r_{mem},r_{sto},r_{topo}
关系特定权重矩阵的维度$\boldsymbol{W}_r$	256×128
关系特定注意力向量的维度$\boldsymbol{a}_r$	256

4.2　收敛性分析

图6为不同网络规模下算力节点数对所提算法性能的影响。由图6可以看出，最优观测范围是存在的，并且随着算力节点数的不同而变化。通过预实

验比较不同观测视野与学习率设置，综合考虑收敛速度与策略性能，后续仿真中将算力节点数和局部观测范围分别固定为50和3，学习率固定为0.005。

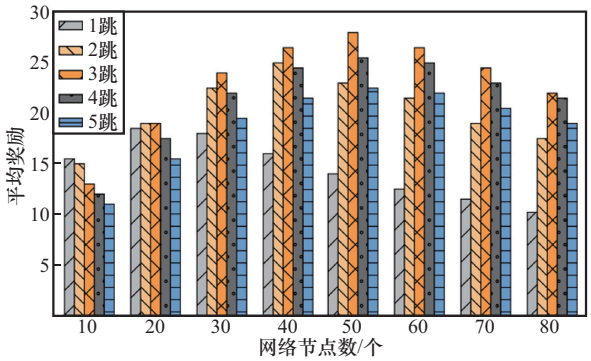


图6　不同网络规模下算力节点数对所提算法性能的影响

4.3　消融实验

4.3.1　依赖感知任务重构算法消融实验

将本文算法作为基准组，通过开关重构模块、三角依赖解耦、子任务合并，构建变体方案进行对比。

w/o重构模块：移除所有重构功能，直接对原始DAG任务进行分配。

w/o三角依赖解耦：保留层索引和子任务合并，不处理跨层依赖。

w/o子任务合并：保留层索引和三角依赖解耦，不执行顺序/并行子任务合并。

（1）子任务数的影响

子任务数对不同算法的性能影响如图7所示。由图7可知，当固定任务量为60 MB时，随着子任务数增加，各算法变体的完成时间呈上升趋势，本文算法性能始终最优：任务完成时间较w/o重构模块、w/o三角依赖解耦、w/o子任务合并分别平均降低63.5%、38.77%、31.23%，这是因为重构模块通过三角依赖解耦降低跨层等待时间，通过阈值约束合并减少子任务与跨节点传输次数。本文算法能效分别为3类算法的1.3倍、1.2倍、1.14倍，得益于重构模块通过合并减少传输能耗、解耦降低调度冲突，且子任务数≥6时差距显著扩大，表明重构模块更适配复杂依赖任务。

（2）任务量大小的影响

任务量大小对不同算法的性能影响如图8所示。由图8可知，当固定子任务数为4时，任务量增大使传输压力上升，重构模块的“子任务合并”功能价值凸显。本文算法的任务完成时间较w/o重构模块平均降低45.54%，因为子任务合并能显著减少跨节点传输次数，缩短总传输时延；本文算法的能效始终高于各变体，且任务量越大优势越明显，证明重构模块更适用于高数据量场景。

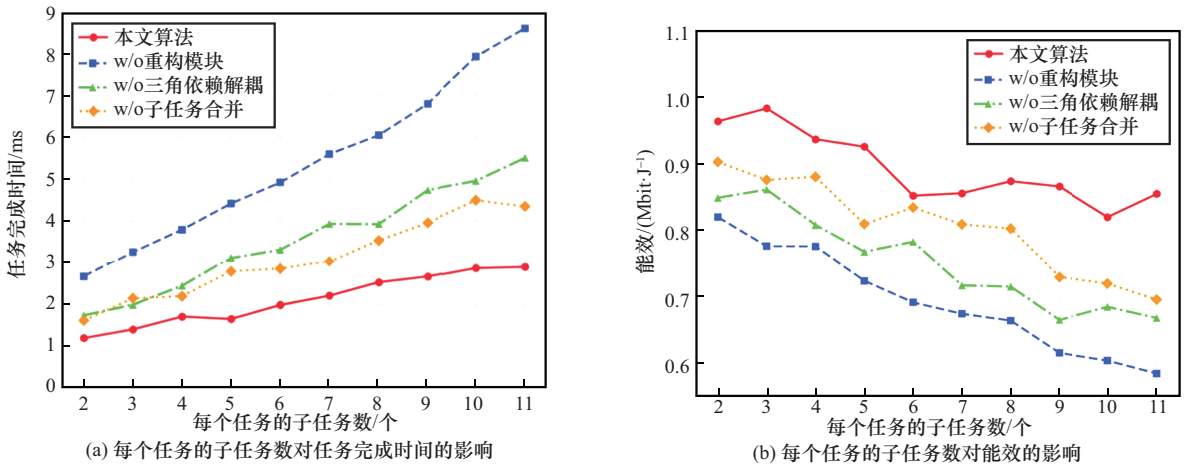
4.3.2　关系图表征模块消融实验

将本文算法作为基准组，通过与w/o RGAT模块、用GAT和GCN替换RGAT模块，构建变体方案进行对比。

w/o RGAT：移除RGAT图神经网络模块，仅利用多层感知机（multilayer perceptron, MLP）独立提取各节点的局部特征，并通过平均池化操作聚合生成状态表征。

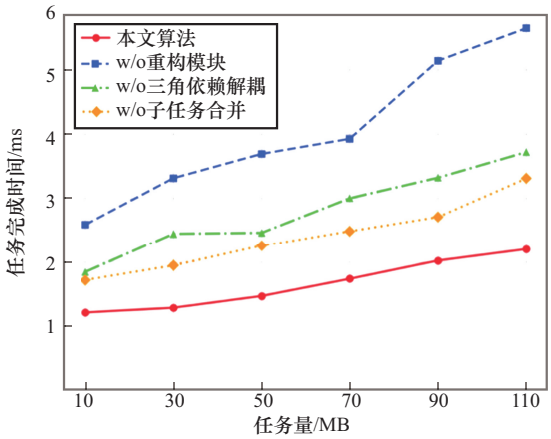
用GAT替换RGAT（W/GAT）：邻接仍用$A_i(t)$，但不区分关系类型。

用GCN替换RGAT（W/GCN）：用标准GCN卷积聚合，对邻域节点分配固定的归一化权重。

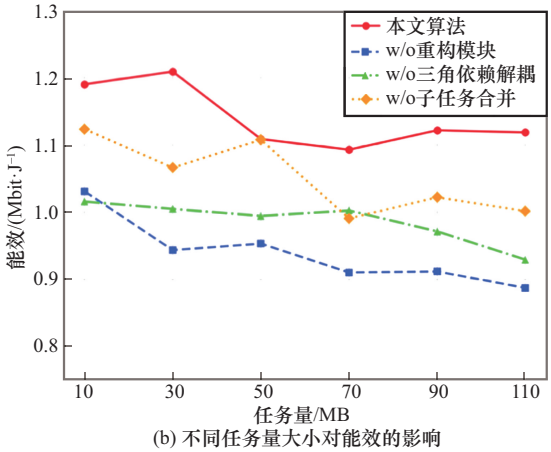


(a) 每个任务的子任务数对任务完成时间的影响　　(b) 每个任务的子任务数对能效的影响

图7　子任务数对不同算法的性能影响



(a) 不同任务量大小对任务完成时间的影响



(b) 不同任务量大小对能效的影响

图8　任务量大小对不同算法的性能影响

如表2所示，在保持任务重构模块一致的条件下，关系图表征模块的消融结果验证了图结构表征与关系感知建模的必要性。与w/o RGAT相比，本文采用RGAT的完整方案在任务完成时间和能效上分别提升约14.3%和15.2%，说明消息传递式邻居聚合能够有效增强智能体对邻居资源和链路状态的感知，从而减少不必要的跨节点传输与资源冲突。用GAT或GCN替换RGAT后性能介于两者之间：GAT与GCN虽能利用拓扑邻居信息，但由于未区分计算、内存、存储、拓扑等多维关系，其资源匹配精度与调度稳定性弱于关系感知的RGAT，因此本文算法相较GAT和GCN在3项指标上仍取得5%~9%的增益。

表2　关系表征模块消融性能表现

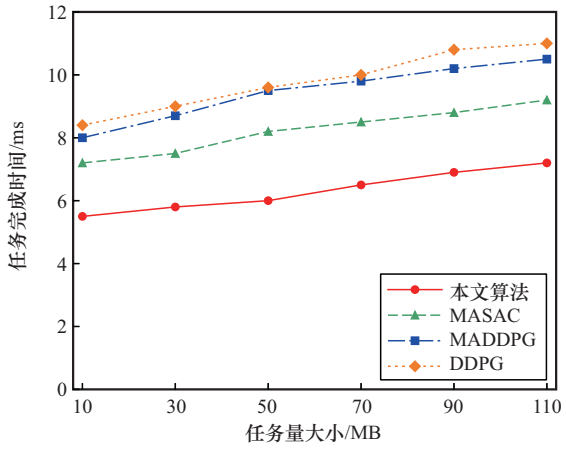
算法	任务完成时间/ms	能效/(Mbit·J^{-1})
本文算法	128.4	0.812
w/ GAT	135.9	0.772
w/ GCN	141.7	0.748
w/o RGAT	149.8	0.705

4.4　对比实验

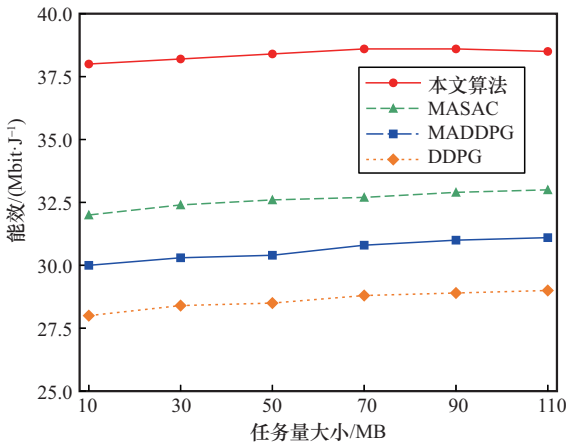
为了验证本文算法的性能，选取MASAC、MADDPG[23]、DDPG[24]3种基线算法对比，所有算法均集成相同的依赖感知任务重构模块，差异仅在于强化学习框架。

（1）任务量大小的影响

不同任务量大小对不同算法的性能影响如图9所示。由图9可知，在固定算力节点数为50个、最大可用算力为4.0 GHz及子任务数为4的条件下，随着任务数据量从10 MB增至110 MB，所有算法的任务完成时间呈线性增长，本文算法通过关系图表征提升任务和资源匹配精度，并通过协作学习降低跨节点传输与冲突，从而在全区间保持更低的完成时间，能效优势也随任务量增长保持稳定。任务完成时间较MASAC、MADDPG和DDPG分别降低24.37%、33.47%和35.38%，能效分别达到MASAC、MADDPG和DDPG的1.15倍、1.24倍和1.33倍。



(a) 不同任务量大小对任务完成时间的影响

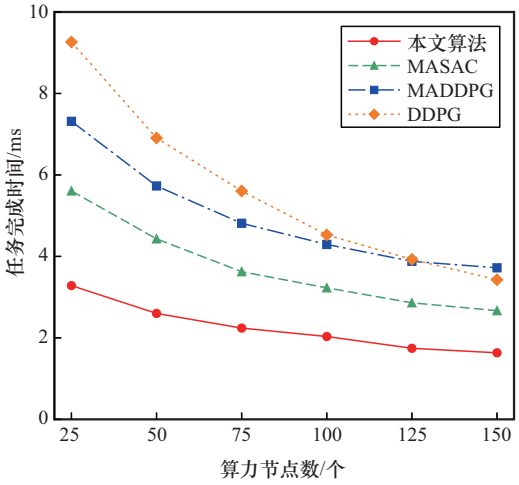


(b) 不同任务量大小对能效的影响

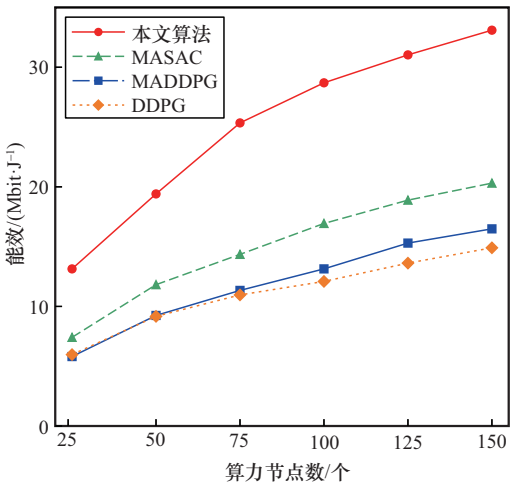
图9　不同任务量大小对不同算法的性能影响

（2）算力节点数的影响

不同算力节点数对不同算法的性能影响如图10所示。由图10(a)可知，在固定最大可用算力为4.0 GHz、子任务数为4、任务量大小为60 MB的条件下，随着算力节点数的增加，任务完成时间显著减少。这是因为算力节点数的增加丰富了边缘算力节点的可用计算资源，提升了任务处理效率。更多任务能够在时延更低的边缘节点执行，减少了任务在网络中的传输与处理耗时。由图10(b)可以看出，算法能效显著提升，本文算法的能效分别为MASAC的1.63倍、MADDPG的1.93倍、DDPG的2.21倍，且在算力节点数大于100个时曲线趋缓，这是因为资源逐步冗余使边际收益递减。



(a) 不同算力节点数对任务完成时间的影响

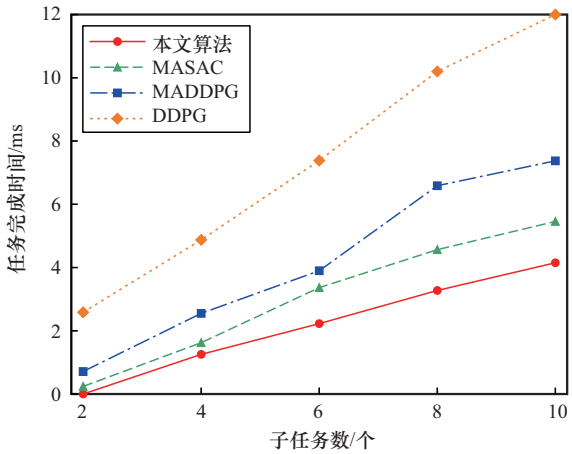


(b) 不同算力节点数对能效的影响

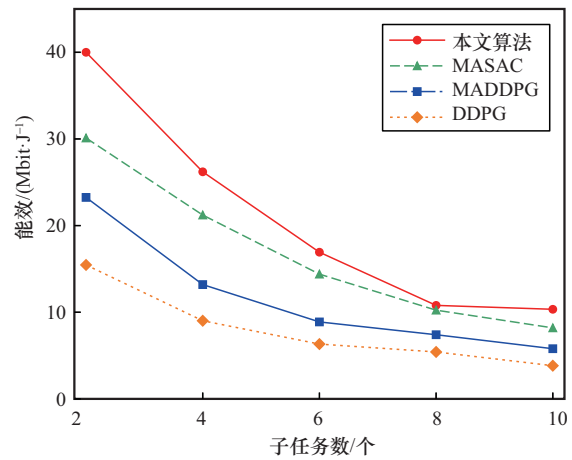
图10　不同算力节点数对不同算法的性能影响

（3）子任务数的影响

不同子任务数对不同算法性能的对比如图11所示。由图11可知，在固定最大可用算力为4.0 GHz、算力节点数为50个、任务量大小为60 MB、子任务数从2增加到6时，所有算法的任务完成时间均上升。本文算法通过更优的任务分配策略，保持了较低的增长率。本文算法通过关系图表征与协作分配减少子任务无效迁移与冲突，在子任务数大于5时优势更明显，这是因为子任务数增加会导致任务依赖关系更复杂、候选分配组合更多，同时跨节点迁移和资源冲突风险上升。本文算法中RGAT能够对计算、内存、存储和拓扑等多维关系进行关系感知建模，提升资源匹配精度；MASAC在较大决策空间下保持较强的探索能力与策略稳定性，从而减少无效迁移与冲突。因此，本文算法在任务完成时间和能效上均表现出明显优势，任务完成时间平均分别比MASAC、MADDPG和DDPG低17.29%、38.41%、57.56%，能效分别高33.04%、87.13%、143.89%。



(a) 不同子任务数对任务完成时间的影响



(b) 不同子任务数对能效的影响

图11　不同子任务数对不同算法性能的对比

（4）收敛性比较

不同算法的收敛性如图 12 所示。随着样本量增长，样本复杂度增加，两种算法的性能均有所提升。此外，本文算法在 50 个训练轮次左右收敛，显著快于 MASAC 的 120 个训练轮次，而且最终获得的奖励显著高于 MASAC，说明了引入 RGAT 的优势。

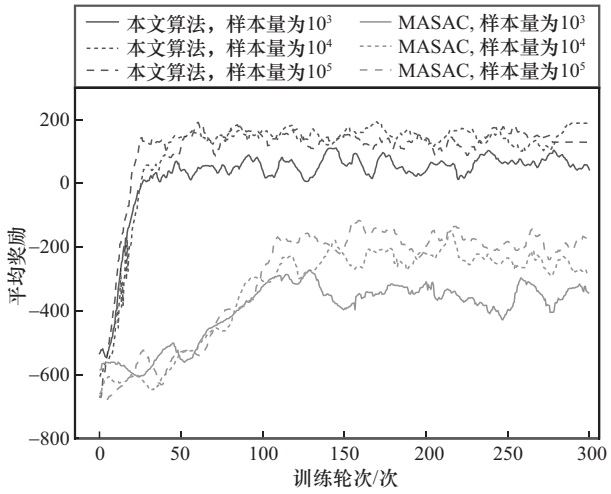


图 12　不同算法的收敛性

5　结束语

本文针对算力网络中任务依赖复杂、资源多维异构与全局协同难的挑战，提出算力网络中依赖约束任务的分布式协同分配算法。通过任务重构降低依赖复杂度，并结合 RGAT 与 MASAC 实现分布式协同决策。仿真实验验证了本文算法的有效性，消融实验表明，任务重构模块可降低任务完成时间和提升能效；通过与 MASAC、MADDPG 和 DDPG 基线算法对比，本文算法在不同网络场景下均展现出显著优势。在不同任务量、算力节点数、子任务数场景下，本文算法相较于基线算法，任务完成时间平均降低 28.6%，能效平均提升 32.5%。特别是在资源受限、任务复杂度高以及网络规模大的情况下，性能优势更突出，证明了本文算法在处理复杂异构环境中的优越性。

参考文献：

[1] Mao Y Y, You C S, Zhang J, et al. A survey on mobile edge computing: the communication perspective[J]. IEEE Communications Surveys & Tutorials, 2017, 19(4): 2322-2358.

[2] 贾庆民, 胡玉姣, 张华宇, 等. 确定性算力网络研究[J]. 通信学报, 2022, 43(10): 55-64.
Jia Q M, Hu Y J, Zhang H Y, et al. Research on deterministic computing power network[J]. Journal on Communications, 2022, 43(10): 55-64.

[3] 中国联通. 中国联通算力网络白皮书[R]. 2019.
China Unicom. China Unicom computing power network white paper[R]. 2019.

[4] Zou W H, Zhang Z S, Wang N N, et al. ReflexPilot: startup-aware dependent task scheduling based on deep reinforcement learning for edge-cloud collaborative computing[J]. IEEE Transactions on Cloud Computing, 2025, 13(2): 641-654.

[5] Fan Q, Ansari N. Application aware workload allocation for edge computing-based IoT[J]. IEEE Internet of Things Journal, 2018, 5(3): 2146-2153.

[6] Pompili D, Hajisami A, Viswanathan H. Dynamic provisioning and allocation in cloud radio access networks (C-RANs)[J]. Ad Hoc Networks, 2015, 30: 128-143.

[7] Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks[PP]. V4. (2017-02-22) [2026-01-04]. arXiv: arXiv.1609.02907.

[8] Veličković P, Cucurull G, Casanova A, et al. Graph attention networks [PP]. V3. (2018-02-04)[2026-01-04]. arXiv: arXiv.1710.10903.

[9] Tian X, Meng Y. Relgraph: a multi-relational graph neural network framework for knowledge graph reasoning based on relation graph[J]. Applied Sciences, 2024, 14(7): 3122.

[10] Jiang F B, Dong L, Wang K Z, et al. Distributed resource scheduling for large-scale MEC systems: a multiagent ensemble deep reinforcement learning with imitation acceleration[J]. IEEE Internet of Things Journal, 2022, 9(9): 6597-6610.

[11] Lu H F, Gu C H, Luo F, et al. Optimization of lightweight task offloading strategy for mobile edge computing based on deep reinforcement learning[J]. Future Generation Computer Systems, 2020, 102: 847-861.

[12] Lowe R, Wu Y, Tamar A, et al. Multi-agent actor-critic for mixed cooperative-competitive environments[PP]. V4. (2020-03-14)[2026-01-04]. arXiv: arXiv.1706.02275.

[13] Chen Z Y, Zhang S C, Tang Y, et al. Optimization of computing power network routing strategies based on multi-agent soft actor-critic[C]// Proceedings of the 2024 9th International Conference on Intelligent Computing and Signal Processing (ICSP). Piscataway: IEEE Press, 2024: 426-430.

[14] Zhang J W, Chen S H, Wang X D, et al. Dynamic reservation of edge servers via deep reinforcement learning for connected vehicles[J]. IEEE Transactions on Mobile Computing, 2023, 22(5): 2661-2674.

[15] Yu C M, Ku M L, Wang L C, et al. BRATRA: balanced routing algorithm with transmission range adjustment for energy efficiency and utilization balance in WSNs[J]. IEEE Internet of Things Journal, 2023, 10 (2): 1096-1111.

[16] Chen Y, Xu J J, Wu Y, et al. Dynamic task offloading and resource allocation for NOMA-aided mobile edge computing: an energy efficient design[J]. IEEE Transactions on Services Computing, 2024, 17(4): 1492-1503.

[17] Liao H L, Li X Y, Guo D K, et al. Dependency-aware application assigning and scheduling in edge computing[J]. IEEE Internet of Things Journal, 2022, 9(6): 4451-4463.

[18] Sohaib M, Jeon S W, Yu W. Hybrid online-offline learning for task offloading in mobile edge computing systems[J]. IEEE Transactions

on Wireless Communications, 2024, 23(7): 6873-6888.
[19] Shi J M, Du J, Wang J J, et al. Priority-aware task offloading in vehicular fog computing based on deep reinforcement learning[J]. IEEE Transactions on Vehicular Technology, 2020, 69(12): 16067-16081.
[20] Tang Q Q, Xie R C, Yu F R, et al. Collective deep reinforcement learning for intelligence sharing in the Internet of intelligence-empowered edge computing[J]. IEEE Transactions on Mobile Computing, 2023, 22 (11): 6327-6342.
[21] Huang X Y, He L J, Chen X, et al. Revenue and energy efficiency-driven delay-constrained computing task offloading and resource allocation in a vehicular edge computing network: a deep reinforcement learning approach[J]. IEEE Internet of Things Journal, 2022, 9(11): 8852-8868.
[22] Tang Q Q, Xie R C, Yu F R, et al. Decentralized computation offloading in IoT fog computing system with energy harvesting: a dec-POMDP approach[J]. IEEE Internet of Things Journal, 2020, 7(6): 4898-4911.
[23] Gao A, Zhang S, Hu Y S, et al. Game-combined multi-agent DRL for tasks offloading in wireless powered MEC networks[J]. IEEE Transactions on Vehicular Technology, 2023, 72(7): 9131-9144.
[24] Liu Q Q, Zhang H X, Zhang X, et al. Joint service caching, communication and computing resource allocation in collaborative MEC systems: a DRL-based two-timescale approach[J]. IEEE Transactions on Wireless Communications, 2024, 23(10): 15493-15506.

[作者简介]

张满钧（1999– ），女，北京邮电大学博士生，主要研究方向为算力网络、资源调度。

王颖（1976– ），女，博士，北京邮电大学副教授、博士生导师，主要研究方向为网络管理与通信软件、软件化网络、算力网络、确定性网络等。

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

邱雪松（1973– ），男，博士，北京邮电大学教授、博士生导师，主要研究方向为网络与业务管理、物联网与区块链。

郭少勇（1985– ），男，博士，北京邮电大学教授，主要研究方向为物联网与区块链。