

资源不确定环境下基于深度强化学习的雾计算工作流 动态调度优化方法

林 剑, 徐 占, 何以凡[†], 许晨昊, 赵治涵

(浙江财经大学 信息技术与人工智能学院, 杭州 310018)

摘 要: 针对资源不确定环境下雾计算工作流动态调度问题 (DWSPFC-RU), 构建以最小化总延迟时间为目标的数学模型, 并提出一种多树小生境遗传编程辅助的深度强化学习 (MTNGPDRL) 方法进行求解. 首先, 将 DWSPFC-RU 分解为路由决策子问题和排序决策子问题, 并据此构建路由智能体与排序智能体. 其次, 提出基于小生境策略的多树遗传编程算法 (MTNGP), 用于生成高效且多样的路由与排序规则. 在此基础上, 将 MTNGP 的规则学习能力与决斗双重深度 Q 网络相结合, 构建多树小生境遗传编程辅助的多智能体深度强化学习框架. 针对 DWSPFC-RU 问题特点, 为两类智能体设计相应的状态空间和奖励函数, 并以 MTNGP 生成的规则构建其动作集合. 最后, 基于不同规模算例进行仿真实验与对比分析, 结果验证了所提出方法在求解 DWSPFC-RU 上的有效性与鲁棒性.

关键词: 资源不确定; 雾计算; 工作流调度; 深度强化学习; 遗传编程; 小生境

中图分类号: TP181 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2025.1088

引用格式: 林剑, 徐占, 何以凡, 等. 资源不确定环境下基于深度强化学习的雾计算工作流动态调度优化方法 [J]. 控制与决策.

Deep reinforcement learning-based dynamic workflow scheduling in fog computing under resource uncertainty

LIN Jian, XU Zhan, HE Yi-fan[†], XU Chen-hao, ZHAO Zhi-han

(School of Information Technology and Artificial Intelligence, Zhejiang University of Finance & Economics, Hangzhou 310018, China)

Abstract: To address the dynamic workflow scheduling problem in fog computing under resource uncertainty (DWSPFC-RU), this paper formulates a mathematical model with the objective of minimizing total tardiness, and proposes a multi-tree niching genetic programming-assisted deep reinforcement learning (MTNGPDRL) method to solve the problem. First, DWSPFC-RU is decomposed into routing decision and sequencing decision subproblems, and the routing agent and sequencing agent are then constructed. Second, a multi-tree niching genetic programming (MTNGP) algorithm based on the niche strategy is proposed to generate efficient and diverse routing and sequencing rules. On this basis, the rule learning capability of MTNGP is integrated with the dueling double deep Q-network to construct a multi-agent deep reinforcement learning framework assisted by MTNGP. According to the characteristics of DWSPFC-RU, the state spaces and reward functions are designed for the two agents, and the action sets are constructed based on the rules generated by MTNGP. Finally, simulation experiments and comparative analysis are conducted on test instances of different scales, and the results verify the effectiveness and robustness of the proposed method in solving DWSPFC-RU.

Keywords: resource uncertainty; fog computing; workflow scheduling; deep reinforcement learning; genetic programming; niche

0 引 言

随着物联网、5G 以及人工智能技术的迅速发

展, 工业互联网^[1]、无人驾驶^[2]与远程医疗^[3]等数据

密集型智能应用不断涌现, 对计算服务的低时延、高

收稿日期: 2025-10-18; 录用日期: 2026-03-18.

基金项目: 国家自然科学基金项目 (61973267); 浙江省自然科学基金项目 (LY24F030008, LQN26F030004).

[†]通信作者. E-mail: yifanhe@zufe.edu.cn.

可靠性和能效提出了严格要求. 传统云计算虽具有强大的集中式计算与存储能力, 但远距离数据传输带来的通信时延与带宽瓶颈, 使其难以满足上述时延敏感型任务的实时处理需求^[4].

为应对这一挑战, 雾计算^[5]作为一种介于云端与边缘之间的分布式计算范式, 通过在网络边缘部署具备计算与存储能力的节点, 实现云-边-端协同, 有效降低核心网络负载、提升响应速度, 为分布式工作流动态执行提供了新型架构. 工作流调度^[6]是雾计算系统性能优化的关键, 其核心目标是在满足时序与资源约束的前提下, 为各任务分配最合适的计算节点, 以提升系统整体运行效率. 然而, 雾计算环境的异构性、动态性和分散性使得调度问题相较于传统云环境更为复杂. 在实际系统中, 算力波动、带宽变化、设备故障和任务随机到达等现象普遍存在, 导致资源状态与任务执行过程均表现出显著不确定性, 从而使得工作流调度从静态优化演变为资源不确定环境下的动态优化问题^[4, 5, 7]. 因此, 研究资源不确定环境下的雾计算工作流动态调度问题 (Dynamic Workflow Scheduling Problem in Fog Computing under Resource Uncertainty, DWSPFC-RU), 在不完全信息和动态环境中实现鲁棒、自适应且高效的调度决策, 已成为当前雾计算研究的重要课题.

近年来, 深度强化学习 (Deep Reinforcement Learning, DRL)^[8] 因具备自学习与自适应决策能力, 逐渐成为动态调度领域的研究热点^[9]. Jayanetti 等人^[10] 设计了一种具有分层动作空间的 DRL 方法, 以区分路由阶段中的云节点与边缘节点. Zhang 等人^[11] 将遗传算法用于生成工作流调度方案, 并结合深度 Q 网络进行计算资源分配. Chen 等人^[12] 采用决斗双重深度 Q 网络 (Dueling Double Deep Q-Network, D3QN) 来解决具有动态任务到达的多工作流协同调度问题. Pan 和 Wei^[13] 提出了一种基于正则化的 DRL 方法, 以实现高效的工作流调度. Wang 等人^[14] 利用聚类技术将连续动作向量映射为“就绪任务-处理器”对, 从而实现从连续动作空间到离散动作空间的转换. Jayanetti 等人^[15] 基于演员-评论家框架设计了一种多智能体深度强化学习方法, 用于多云环境下的工作流调度. He 等人^[16] 提出了一种多智能体方法来解决云工作流调度问题, 其中使用 Q 学习进行工作流内任务排序, 并采用 D3QN 进行路由决策.

尽管 DRL 已是调度优化领域的热点, 现有方法在解决工作流调度问题时仍存在明显不足. 首先, 大多数研究忽视了与路由调度同样关键的任务排序决策. 在具有严格时延约束的工作流场景中, 设备上任

务的执行顺序直接影响工作流的完成时刻. 其次, 部分研究采用端到端的调度方式, 通常直接将系统状态映射为具体的设备或任务动作输出, 这种方法虽赋予智能体充分的决策自由, 但由于受到静态神经网络结构的限制, 往往只能在特定规模的场景中有效. 一旦系统遭遇设备故障等突发事件, 预定义的映射关系难以及时调整, 甚至可能导致智能体完全失效. 为弥补端到端方法的缺陷, 部分研究引入固定数量的人工预定义规则集作为动作空间, 使智能体通过间接决策完成调度. 然而这种方法高度依赖人工设计, 具有较强的主观性, 且所构建的动作集合通常仅能覆盖有限的场景特征. 当系统规模扩大或任务特性发生变化时, 现有动作空间往往无法及时适配, 从而导致调度性能显著下降, 这一局限性在动态与不确定环境中表现得尤为突出.

为解决上述问题, 本文提出一种面向资源不确定环境的多智能体深度强化学习雾计算工作流动态调度方法. 与现有研究中主要针对确定性环境的模型不同, 本文综合考虑工作流动态到达、设备故障等不确定因素, 构建了由路由智能体和排序智能体组成的多智能体架构, 以实现资源分配与任务选择的协同优化.

此外, 为提高动作空间的适应性与策略的鲁棒性, 本文引入多树小生境遗传编程算法 (Multi-Tree Niching Genetic Programming, MTNGP), 为智能体自动设计多样化的路由和排序规则集. 在此基础上融入 D3QN 结构, 提出多树小生境遗传编程辅助的深度强化学习 (Multi-Tree Niching Genetic Programming-assisted Deep Reinforcement Learning, MTNGPDRL) 方法, 实现资源不确定环境下复杂任务的高效动态调度. 最后, 基于多种典型科学工作流构建雾计算仿真环境, 并通过多组实验对所提方法的有效性与泛化能力进行验证.

本文的主要贡献归纳如下:

- (1) 构建一种资源不确定环境下的雾计算工作流动态调度模型, 同时考虑了任务依赖、资源约束及系统不确定性;
- (2) 设计多树小生境遗传编程算法, 用于生成多样高效的调度规则, 以提升智能体的动作多样性与策略鲁棒性;
- (3) 提出多树小生境遗传编程辅助的深度强化学习框架, 实现资源分配与任务排序的协同优化;
- (4) 通过大量实验分析验证所提方法在解决不确定环境下雾计算工作流动态调度问题上的高效性、稳定性与泛化能力.

1 问题描述与模型

1.1 问题描述

在 DWSPFC-RU 中, 系统由一组用户设备 \mathcal{D}^u 、一组边缘服务器 \mathcal{D}^e 和一组云服务器 \mathcal{D}^c 构成. 每台用户设备会不定期发布工作流, 工作流任务可在本地执行, 也可卸载至边缘或云服务器执行. DWSPFC-RU 的目标是在满足任务依赖约束的前提下, 将各工作流的任务分配至适当的处理设备, 并安排其开始执行的时间, 以最小化系统中所有工作流的总延迟时间.

DWSPFC-RU 问题遵循以下假设:

假设 1 每个工作流的具体信息在其被发布之前完全未知;

假设 2 任务仅在其所有前驱任务均完成后才可就绪并等待执行;

假设 3 同一时间每个设备最多只能处理一个任务, 且每个任务只能由一个设备执行;

假设 4 任务处理过程为非抢占式, 即一旦任务进入上传、执行或下载阶段, 调度器不得主动中断或切换;

假设 5 若设备在运行过程中发生故障, 将暂时失去处理能力, 任务的上传、执行或下载进程会被迫中断, 需在系统恢复后重新开始;

假设 6 用户设备与边缘服务器受存储容量限制, 满载时无法接收新任务, 而云服务器则具备充足存储资源.

假设 7 各设备在任何时刻仅能获取本地队列与运行状态, 以及有限的邻域摘要, 不可直接获得全局队列与全局状态.

1.2 问题建模

1.2.1 工作流模型

定义 \mathcal{W} 为系统中所有工作流的集合, 每个工作流 $W_i \in \mathcal{W}$ 可以被表示为一个有向无环图 $G_i = (\mathcal{T}_i, \mathcal{E}_i)$, 其中 \mathcal{T}_i 为图中的节点集合, 表示该工作流包含的所有任务, 而图的有向边集合 \mathcal{E}_i 则表示任务之间的前驱依赖关系.

每个任务 $T_{i,j} \in \mathcal{T}_i$ 可以由一个三元组 $(Data_{i,j}^{up}, Comp_{i,j}, Data_{i,j}^{down})$ 定义, 其中 $Comp_{i,j}$ 表示完成任务 $T_{i,j}$ 所需的计算资源量 (即 CPU 周期数), $Data_{i,j}^{up}$ 和 $Data_{i,j}^{down}$ 分别表示任务的输入数据量和输出数据量. 根据前驱依赖关系, 每个任务 $T_{i,j}$ 对应前驱任务集合 $pred(T_{i,j}) \subset \mathcal{T}_i$ 和后继任务集合 $succ(T_{i,j}) \subset \mathcal{T}_i$, 仅当所有前驱任务完成时, $T_{i,j}$ 视为就绪, 进入等待调度执行状态. 无前驱任务 (即 $pred(T_{i,j}) = \emptyset$) 的

任务称为入口任务. 每个工作流 $W_i \in \mathcal{W}$ 在时刻 a_i 由用户设备 D_{u_i} 发布, 并设有相应的截止时间 DD_i . 当工作流中的所有任务均完成时, 视为该工作流完成.

1.2.2 设备模型

定义 \mathcal{D} 为系统中所有设备的集合, 包含用户设备 \mathcal{D}^u 、边缘服务器 \mathcal{D}^e 和云服务器 \mathcal{D}^c 三类设备. 设备 $D_k \in \mathcal{D}^e \cup \mathcal{D}^c \cup \mathcal{D}^u$ 的算力记为 γ_k . $T_{i,j}$ 在设备 D_k 上的理论执行时间可由式 (1) 计算.

$$\tau_{i,j,k}^{exec} = \frac{Comp_{i,j}}{\gamma_k}. \quad (1)$$

设备在运行过程中可能发生故障, 对于任意设备 D_k , 其第 l 次故障开始的时间记为 $BS_{k,l}$, 相应的修复用时记为 $bs_{k,l}$. 在故障期间, 设备无法处理任何任务, 需等待修复完成后方可恢复正常工作. ω_k 为设备的存储容量, 存储容量将满时, 设备将无法接收新任务.

1.2.3 传输模型

用户设备可将任务卸载至边缘服务器或云服务器, 记两设备 D_u 和 D_v 之间的带宽为 $B_{u,v}$. 传输模型基于香农定理, 设备 D_u 与 D_v 间的传输速率可由式 (2) 计算.

$$r_{u,v} = B_{u,v} \cdot \log_2 \left(1 + \frac{p_u \cdot g_{u,v}}{w_0 + N_0} \right). \quad (2)$$

其中, w_0 表示环境噪声, N_0 表示来自其他设备的干扰噪声, p_u 表示设备 D_u 的传输功率, $g_{u,v} = dist_{u,v}^{-\sigma}$ 表示信道增益, 其由设备间的距离 $dist_{u,v}$ 和路径损耗指数 σ 计算得到.

当任务 $T_{i,j}$ 被分配至边缘服务器或云服务器 D_v 执行时, 其输入数据的上传时间和输出数据的下载时间可由式 (3) 计算. 若 $T_{i,j}$ 在本地执行, 则上传时间和下载时间均为 0.

$$\tau_{i,j,v}^{up} = \frac{Data_{i,j}^{up}}{r_{u_i,v}}, \quad \tau_{i,j,v}^{down} = \frac{Data_{i,j}^{down}}{r_{v,u_i}}. \quad (3)$$

1.2.4 任务生命周期模型

任务 $T_{i,j}$ 处理过程中的关键时间点可表示为五元组 $(rt_{i,j}, at_{i,j}, st_{i,j}, et_{i,j}, ct_{i,j})$. 其中, $rt_{i,j}$ 表示任务的就绪时间, 即该任务具备调度条件的时刻; $at_{i,j}$ 表示输入数据到达执行设备的时刻; $st_{i,j}$ 表示任务 $T_{i,j}$ 开始执行的时刻; $et_{i,j}$ 表示任务 $T_{i,j}$ 执行完成的时刻; $ct_{i,j}$ 表示任务 $T_{i,j}$ 的输出数据返回原设备的时刻, 即该任务最终完成的时刻. 此外, 定义二元决策变量 $X_{i,j,k}$ 表示任务 $T_{i,j}$ 的设备分配关系: 当 $X_{i,j,k} = 1$ 时, 表示任务 $T_{i,j}$ 被分配至设备 D_k 执行; 当 $X_{i,j,k} = 0$ 时, 则表示该任务未分配至设备 D_k .

1.2.5 问题模型

基于上述分析, DWSPFC-RU 的数学模型可推导如下.

$$\min \sum_{i=1}^{|\mathcal{W}|} \max_{T_{i,j} \in \mathcal{W}_i} (ct_{i,j} - DD_i, 0), \quad (4)$$

$$\text{s.t.} \sum_{D_k \in \mathcal{D}} X_{i,j,k} = 1, \forall i, j, \quad (5)$$

$$X_{i,j,k} = 0, \forall D_k \in \mathcal{D}^u \setminus \{D_{u_i}\}, \quad (6)$$

$$(st_{i,j} - et_{i',j'})(et_{i,j} - st_{i',j'})X_{i,j,k}X_{i',j',k} \geq 0, \quad (7)$$

$$rt_{i,j} - \max_{j' \in \text{pred}(T_{i,j})} (ct_{i,j'}) = 0, \forall \text{pred}(T_{i,j}) \neq \emptyset, \quad (8)$$

$$rt_{i,j} - a_i = 0, \forall \text{pred}(T_{i,j}) = \emptyset, \quad (9)$$

$$st_{i,j} - rt_{i,j} - \sum_{D_k \in \mathcal{D}} \tau_{i,j,k}^{up} X_{i,j,k} \geq 0, \quad (10)$$

$$et_{i,j} - st_{i,j} - \sum_{D_k \in \mathcal{D}} \tau_{i,j,k}^{exec} X_{i,j,k} \geq 0, \quad (11)$$

$$ct_{i,j} - et_{i,j} - \sum_{D_k \in \mathcal{D}} \tau_{i,j,k}^{down} X_{i,j,k} \geq 0, \quad (12)$$

$$\omega_k - \sum_{i,j} (Data_{i,j}^{up} + Data_{i,j}^{down}) X_{i,j,k} \geq 0, \quad (13)$$

$$(at_{i,j} - \tau_{i,j,k}^{up} - (BS_{k,l} + bs_{k,l})) X_{i,j,k} \geq 0 \\ \vee (BS_{k,l} - at_{i,j}) X_{i,j,k} \geq 0, \quad (14)$$

$$(et_{i,j} - \tau_{i,j,k}^{exec} - (BS_{k,l} + bs_{k,l})) X_{i,j,k} \geq 0 \\ \vee (BS_{k,l} - et_{i,j}) X_{i,j,k} \geq 0, \quad (15)$$

$$(ct_{i,j} - \tau_{i,j,k}^{down} - (BS_{k,l} + bs_{k,l})) X_{i,j,k} \geq 0 \\ \vee (BS_{k,l} - ct_{i,j}) X_{i,j,k} \geq 0, \quad (16)$$

$$X_{i,j,k} \in \{0, 1\}, st_{i,j} \geq 0, \forall i \in \mathcal{W}, j \in T_i, k \in \mathcal{D}. \quad (17)$$

式 (4) 为 DWSPFC-RU 的优化目标, 即最小化所有工作流的总延迟时间; 式 (5) 保证每个任务必须且仅被分配给一个设备; 式 (6) 强调当任务需要在用户侧执行时, 只能分配至其发起的用户设备; 式 (7) 表示同一设备在同一时刻至多处理一个任务; 式 (8) 和式 (9) 分别定义了非入口任务和入口任务的就绪时间; 式 (10)–(12) 给出了任务上传、执行与下载三个阶段的时间约束. 式 (13) 表示边缘/终端设备的存储容量约束; 式 (14)–(16) 规定任务在上传、执行和下载阶段若遭遇设备故障, 需在设备恢复后重新开始相应阶段; 式 (17) 定义了决策变量的取值范围.

2 基于多树小生境遗传编程辅助的深度强化学习方法

2.1 方法概述

在 DWSPFC-RU 问题中, 每个任务均涉及两个

相互依赖的决策变量 $X_{i,j,k}$ 与 $st_{i,j}$, 分别对应路由决策和排序决策. 前者用于确定任务的执行设备, 后者用于确定任务在目标设备上的开始执行时间. 因此, DWSPFC-RU 可视为一个两阶段序贯决策问题.

本文提出一种多树小生境遗传编程辅助的深度强化学习方法 MTNGPDRL 以求解该问题. 该方法首先引入多树小生境遗传编程方法 MTNGP, 在小生境机制的辅助下协同演化路由规则和排序规则, 生成兼具多样性与有效性的候选规则集, 并据此构建智能体的动作空间. 在此基础上, 设计路由智能体和排序智能体, 分别负责调度过程中的路由决策与排序决策, 从而形成规则演化与强化学习决策相结合的协同优化框架.

2.2 基于多树小生境遗传编程的动作演化

MTNGP 沿袭遗传编程的基本范式, 以表达式树表示个体, 其中内部节点为算术或比较运算符 (如 +、-、 \times 、 \div 、max、min 等), 叶节点取自预设的终端特征集. 相较于经典遗传编程, MTNGP 在个体表示、遗传操作和多样性维持机制方面进行了改进. 其整体框架如图 1 所示, 具体实现过程见算法 1.

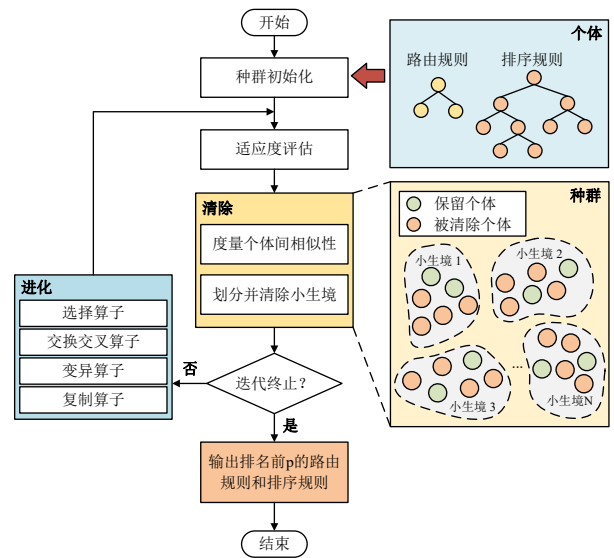


图1 MTNGP 流程

算法1 MTNGP算法

输入: 种群大小 P_{size} , 迭代次数 G , 适应度函数 f , 半径 ρ , 容量 κ , 决策场景 \mathcal{RS} 和 \mathcal{TS} , 输出个体数 p

输出: 最优个体集合 A

1 $\mathcal{R} \leftarrow$ 初始化大小为 P_{size} 的种群;

2 for $g \leftarrow 1$ to G do

3 $r^* \leftarrow \arg \min_{r \in \mathcal{R}} f(r)$;

4 foreach $r \in \mathcal{R}$ do

5 $c(r) \leftarrow BC(r, r^*, \mathcal{RS}, \mathcal{TS})$;

```

6   $\mathcal{R}' \leftarrow$  将  $\mathcal{R}$  按  $f$  升序排序;
7  for  $i \leftarrow 1$  to  $|\mathcal{R}'|$  do
8     $r \leftarrow \mathcal{R}'[i]$ ;
9    if  $f(r) < \infty$  then
10     size  $\leftarrow 1$ ;
11     for  $j \leftarrow i + 1$  to  $|\mathcal{R}'|$  do
12        $r' \leftarrow \mathcal{R}'[j]$ ;
13       if  $\|c(r) - c(r')\|_2 \leq \rho$  then
14         if size  $< \kappa$  then
15           size  $\leftarrow$  size + 1;
16         else
17            $f(r') \leftarrow \infty$ ;
18    $\mathcal{R} \leftarrow$  对  $\mathcal{R}'$  执行选择、交换交叉和变异;
19  $A \leftarrow$  从  $\mathcal{R}$  中选择最优的  $p$  个个体;
20 返回  $A$ ;

```

MTNGP 采用双树编码, 每个个体由两棵表达式树构成, 分别表示路由规则和排序规则, 并通过复制、变异和交换交叉三类遗传算子进行搜索. 两棵树共享统一的终端集和函数集, 从而保证生成规则建立在一致的状态特征与运算符基础之上.

通过引入小生境策略, MTNGP 依据个体差异性, 将种群划分为若干独立小生境, 避免个体过度集中, 从而提升群体多样性并覆盖搜索空间中的潜在优良区域. 借助这一机制, MTNGP 能够进化出一组多样且互补的调度规则.

小生境机制的核心在于个体间距离度量与种群管理策略. 在 MTNGP 中, 每个个体以表达式树而非实向量表示, 因此直接计算结构距离既困难, 又未必能够真实反映规则在决策行为上的差异.

为此, 本文提出一种基于行为表征的个体间距离度量方式. 行为表征以向量形式表示, 由路由规则与排序规则的行为表征向量联合构成. 个体 r 的行为表征由参照个体 r^* 、任务路由场景集合 \mathcal{RS} 和任务排序场景集合 \mathcal{TS} 共同决定. 为区分两类规则, 分别用下标 “R” 和 “S” 标记个体中的路由规则和排序规则.

在每一次迭代中, 从随机生成的实例中抽取 20 个路由场景和 20 个排序场景. 每个路由场景对应某台设备在面临路由决策时的观测状态, 包含该时刻所有可选设备及相关属性信息; 每个排序场景则对应某台设备的独立调度点, 包含其等待队列中的任务集合及相关属性信息. 对于每个路由场景 ds_i , 首先利用参照个体的路由规则 r_R^* 对候选设备进行优先级排序, 并标记其中最高优先级的设备 hd_i ; 随后, 将待评估个体的路由规则 r_R 应用于同一场景, 重新

生成排序, 并记录 hd_i 在该排序中的名次, 作为 r_R 行为表征向量的第 i 个元素. 通过重复该过程得到路由规则的完整行为表征.

类似地, 对于每个排序场景 ts_i , 首先利用参照个体的排序规则 r_S^* 对待排任务排序, 并标记其中最高优先级的任务 ht_i ; 随后将待评估个体的排序规则 r_S 应用于同一场景, 记录 ht_i 在排序中的名次, 作为 r_S 行为表征向量的第 i 个元素. 重复该过程即可得到排序规则的完整表征向量. 最终, 个体的行为表征向量由 r_R 和 r_S 两部分拼接而成. 完整的计算流程如算法 2 所示.

算法2 个体行为表征向量计算

```

输入: 候选规则  $r$ , 参照规则  $r^*$ , 路由决策场景集  $\mathcal{RS}$ , 排序决策场景集  $\mathcal{TS}$ 
输出: 行为表征向量  $c$ 
1   $c^R \leftarrow []$ ,  $c^S \leftarrow []$ ;
2  foreach  $ds_i \in \mathcal{RS}$  do
3     $AD_i \leftarrow$  该场景下的可用设备集合,  $hd_i \leftarrow$ 
arg max $d \in AD_i$   $r_R^*(d)$ ;
4     $AD'_i \leftarrow$  按候选规则的路由规则  $r_R$  对  $AD_i$  降序排序;
5     $c^R \leftarrow [c^R, hd_i \text{ 在 } AD'_i \text{ 中的索引}]$ ;
6  foreach  $ts_i \in \mathcal{TS}$  do
7     $T_i \leftarrow$  该场景下设备等待队列中的待排任务集合,
 $ht_i \leftarrow$  arg max $t \in T_i$   $r_S^*(t)$ ;
8     $T'_i \leftarrow$  按候选规则的排序规则  $r_S$  对  $T_i$  降序排序;
9     $c^S \leftarrow [c^S, ht_i \text{ 在 } T'_i \text{ 中的索引}]$ ;
10  $c \leftarrow [c^R, c^S]$ ;
11 返回  $c$ ;

```

在种群管理策略方面, 本文采取清除策略, 其核心思想是在每个小生境中仅保留最优的一组个体, 并淘汰同一小生境内的其余个体. 通过这种方式, 算法能够有效避免某一优良个体快速垄断种群.

清除过程由两个参数控制: 半径 ρ 用于限定小生境的边界, 即设定个体在行为表征空间中可接受的最大相似距离; 若两个个体的差异低于该阈值, 则被划分至同一小生境. 容量 κ 则规定了每个小生境中允许存活的个体数量, 用来防止相似个体过多聚集. 完成清除后, 算法继续执行标准的遗传操作; 在进化终止时, 依据适应度排名选取前 p 个个体, 作为最终的输出.

本文从设备、任务与工作流三个维度出发, 构建了包含 16 项指标的终端集, 以刻画系统运行的核心特征并为规则生成提供支撑, 具体如表 1 所示.

表1 MTNGP 终端集

终端	解释
TIQ	等待队列中的任务数
QWL	队列中的总工作量
TRWL	传输队列总负载
MRT	当前设备剩余处理时间
UT	任务的上传时间
DT	任务的下载时间
PT	任务的执行时间
TQT	队列的预计处理时间
WWT	工作流已等待的时间
TWT	任务已等待的时间
TDD	距离工作流截止时间的剩余时间
EST	工作流的预估松弛时间
RTC	工作流中剩余任务数
RTW	工作流的剩余工作量
TWL	工作流的总工作量
DTR	工作流中的预计延迟任务比例

2.3 智能体设计

为刻画多个智能体在不完全信息环境下的协同决策过程, 本文采用分布式部分可观测马尔可夫决策过程^[17]进行建模. 该框架下, 路由智能体与排序智能体基于局部观测进行决策, 并通过各自的奖励信号实现协同优化. 下面分别对两类智能体的状态、动作与奖励进行定义. 为便于描述, 预先定义若干关键符号与变量如下:

- (1) EAT_k : 设备 D_k 的最早可用时间;
- (2) WQ_k : 设备 D_k 的等待队列任务集合;
- (3) UPT_k : 当前正在上传至设备 D_k 的任务集合;

(4) $\delta_{i,j}$: 任务 $T_{i,j}$ 的完成指示变量 (完成取 1, 否则取 0);

(5) $WL_k = (\sum_{T_{i,j} \in WQ_k \cup UPT_k} Comp_{i,j}) / \gamma_k$: 设备 D_k 的工作负载;

(6) $CR_i = (\sum_{T_{i,j} \in W_i} \delta_{i,j} Comp_{i,j}) / (\sum_{T_{i,j} \in W_i} Comp_{i,j})$: 工作流 W_i 的完成率;

(7) $\tau_{i,j,k}^{norm} = (\tau_{i,j,k}^{exec} - \underline{\tau}) / (\bar{\tau} - \underline{\tau})$: 任务 $T_{i,j}$ 在设备 D_k 上的归一化执行时长, 其中 $\underline{\tau}$ 和 $\bar{\tau}$ 分别表示所有设备上任务执行时长的全局最小值和最大值;

(8) $ECT_{i,j}$: 任务 $T_{i,j}$ 的预计完成时间, 若未就绪则基于前序任务的预计完成时间和自身处理时长计算, 若正在执行则依据当前进度确定, 若已完成则等于其实际完成时间;

(9) $ACT_{i,j}$: 任务 $T_{i,j}$ 的实际完成时间;

(10) $WCT_i = \max_{T_{i,j} \in W_i} ECT_{i,j}$: 工作流 W_i 的预计完成时间;

(11) $EST_i = DD_i - WCT_i$: 工作流 W_i 的预计松弛时间;

(12) $DTR_i = |\{T_{i,j} | ECT_{i,j} > DD_i\}| / |T_i|$: 工作流 W_i 的预计任务延迟率.

2.3.1 路由智能体设计

路由智能体负责为每个就绪任务选择合适的执行设备. 在每个决策时刻 t , 将观测特征 o_t 作为神经网络的输入, 用于反映智能体在部分可观测条件下对全局状态 s_t 的局部感知. 定义 \mathcal{D}_a 为当前可用设备集合, 其中 $D_k \in \mathcal{D}_a$ 表示第 k 个可用设备. 为确定任务的最优分配目标, 需要综合考量各设备的传输能力、处理能力与实时负载. 基于此, 构建以下特征以形成路由智能体的观测空间:

(1) 当前可用设备的总数: $|\mathcal{D}_a|$;

(2) 各设备最早空闲时间 $\{EAT_k | D_k \in \mathcal{D}_a\}$ 的平均值、标准差、最大值和最小值, 即

$$\mathcal{EAT} = \{EAT_{avg}, EAT_{std}, EAT_{max}, EAT_{min}\}; \quad (18)$$

(3) 任务 $T_{i,j}$ 上传至各可用设备所需时间 $\{\tau_{i,j,k}^{up} | D_k \in \mathcal{D}_a\}$ 的均值、标准差、最大值和最小值, 即

$$\mathcal{UT} = \{UT_{avg}, UT_{std}, UT_{max}, UT_{min}\}; \quad (19)$$

(4) 任务 $T_{i,j}$ 在各可用设备上所需执行时间 $\{\tau_{i,j,k}^{exec} | D_k \in \mathcal{D}_a\}$ 的均值、标准差、最大值和最小值, 即

$$\mathcal{PT} = \{PT_{avg}, PT_{std}, PT_{max}, PT_{min}\}; \quad (20)$$

(5) 任务 $T_{i,j}$ 从各可用设备下载回原设备所需时间 $\{\tau_{i,j,k}^{down} | D_k \in \mathcal{D}_a\}$ 的均值、标准差、最大值和最小值, 即

$$\mathcal{DT} = \{DT_{avg}, DT_{std}, DT_{max}, DT_{min}\}; \quad (21)$$

(6) 各可用设备当前总负载 $\{WL_k | D_k \in \mathcal{D}_a\}$ 的均值、标准差、最大值和最小值, 即

$$\mathcal{WL} = \{WL_{avg}, WL_{std}, WL_{max}, WL_{min}\}; \quad (22)$$

基于以上 21 个特征, MTNGPDRL 路由智能体的观测空间定义如下:

$$o_t^r = \{|\mathcal{D}_a|, \mathcal{EAT}, \mathcal{UT}, \mathcal{PT}, \mathcal{DT}, \mathcal{WL}\} \quad (23)$$

若直接采用端到端方式将动作空间映射至所有设备, 可能导致在动态环境下出现不可用设备的分配风险. 为避免该问题, 路由智能体的动作空间被设计为在一组固定数量的调度策略之间进行选择. 具体来说, 该动作空间既包括 MTNGP 自动进化生成的一组多样化、可解释的规则, 也包含经典人工规则最早完成优先 (EFT) 与最短处理时间优先 (PT), 其

形式化表示为:

$$A_t^r = \{\mathbf{Rule}_1, \dots, \mathbf{Rule}_p, \text{EFT}, \text{PT}\} \quad (24)$$

在为任务 $T_{i,j}$ 做出路由决策后,路由智能体不会立即获得奖励,而是在任务完成后由系统进行回溯赋予.由于实际完成时间 $ACT_{i,j}$ 不仅受路由决策影响,还与任务规模、设备性能及瞬时系统负载等因素密切相关,若直接以其作为奖励依据,难以准确反映路由决策本身的优劣.为此,引入预计完成时间 $ECT_{i,j}$ 作为参考基准,用以表征任务在当前估计条件下的预期完成时间,从而对 $ACT_{i,j}$ 进行补正.在此基础上,若任务超过截止时间完成,则在奖励中加入与延迟时间成正比的惩罚项,其强度由系数 α 调节.路由智能体的奖励函数如式(25)所示.

$$R_t^r = ECT_{i,j} - ACT_{i,j} + \min\{\alpha(DD_i - ACT_{i,j}), 0\} \quad (25)$$

2.3.2 排序智能体设计

在路由智能体完成任务分配后,排序智能体负责确定已分配任务在目标设备上的执行优先顺序.由于动态调度过程中任务到达与完成会不断改变等待队列长度,若直接以各等待任务的详细状态作为输入,则输入维度将随队列长度变化,而全连接神经网络难以处理变长输入.为此,本文采用统计性指标对等待队列整体状态进行表征,以固定状态空间维度,并为排序智能体提供更高层次的系统状态表示.具体而言,排序智能体的状态特征构建如下:

(1) 设备 D_k 等待队列 \mathcal{WQ}_k 中的任务数量 TIQ_k ;

(2) 设备等待队列中各任务归一化执行时间 $\{\tau_{i,j,k}^{norm} \mid T_{i,j} \in \mathcal{WQ}_k\}$ 的平均值、标准差、最大值和最小值,即

$$\mathcal{ET} = \{ET_{avg}, ET_{std}, ET_{max}, ET_{min}\}; \quad (26)$$

(3) 设备等待队列中任务所属工作流完成率 $\{CR_i \mid \exists T_{i,j} \in \mathcal{WQ}_k\}$ 的平均值、标准差、最大值和最小值,即

$$\mathcal{CR} = \{CR_{avg}, CR_{std}, CR_{max}, CR_{min}\}; \quad (27)$$

(4) 设备等待队列中各任务所属工作流的估计松弛时间 $\{EST_i \mid \exists T_{i,j} \in \mathcal{WQ}_k\}$ 经max-min归一化后的平均值与标准差,即

$$\mathcal{NST} = \{NST_{avg}, NST_{std}\}; \quad (28)$$

(5) 设备等待队列中各任务所属工作流的估计延迟任务率 $\{DTR_i \mid \exists T_{i,j} \in \mathcal{WQ}_k\}$ 的平均值、标准差、最大值和最小值,即

$$DTR = \{DTR_{avg}, DTR_{std}, DTR_{max}, DTR_{min}\}; \quad (29)$$

(6) 设备 D_k 的等待队列中各个任务所对应工作流预计将发生逾期的占比,记为估计延迟工作流率 DWR_k ,其定义为:

$$DWR_k = \frac{|\{T_{i,j} \mid WCT_i > DD_i\}|}{TIQ_k}. \quad (30)$$

基于以上16个特征,排序智能体的状态特征可形式化表示为:

$$\phi_t^s = \{TIQ_k, \mathcal{ET}, \mathcal{CR}, \mathcal{NST}, DTR, DWR_k\} \quad (31)$$

由于等待队列长度动态变化,排序智能体难以直接端到端选择任务.为此,本文将MTNGP生成的规则与经典人工规则(先来先服务FCFS、最短处理时间优先SPT、完成率最高优先HCR、最短松弛时间优先MS)结合,构建排序智能体的动作空间.具体来说,排序智能体的动作空间定义为:

$$A_t^s = \{\mathbf{Rule}_1, \dots, \mathbf{Rule}_p, \text{FCFS}, \text{SPT}, \text{HCR}, \text{MS}\}. \quad (32)$$

排序智能体旨在选择最有助于降低工作流累计延迟的任务.然而,当某一任务被优先调度时,其他任务的排队时间必然延长,进而影响其所属工作流的松弛时间.本文将此类影响统一建模为松弛时间的增益或损失,并据此设计奖励函数,以引导智能体在提升整体性能的同时兼顾对其他任务的影响.

首先,定义松弛时间增益为被选中任务 $T_{i,j}$ 所属工作流 W_i 因该次决策而获得的松弛时间增加量:

$$\Delta S_1 = EST_i - EST'_i, \quad (33)$$

其中, EST'_i 为 W_i 的原始松弛时间, EST_i 为该次决策后的松弛时间.

其次,定义松弛时间损失为队列中其他任务所属工作流因排队延长而损失的松弛时间:

$$\Delta S_2 = \sum_{T_{i',j'} \in \mathcal{WQ}_k \setminus \{T_{i,j}\}} (EST'_{i'} - EST_{i'}). \quad (34)$$

于是,排序智能体该次决策对该设备整体松弛时间带来的净变化为 $\Delta S_1 - \Delta S_2$.为避免不同队列长度导致奖励值失衡,本文将其按任务数 TIQ_k 进行缩放,并在区间 $[\beta_{min}, \beta_{max}]$ 内裁剪,最终得到奖励函数:

$$R_t^s = \text{clip}\left(\frac{\Delta S_1 - \Delta S_2}{TIQ_k}, \beta_{min}, \beta_{max}\right). \quad (35)$$

2.4 网络结构与训练算法

MTNGPDRL智能体基于D3QN实现,网络均采用全连接结构.每个智能体包含一个评估网络 $Q(\theta_E)$ 和一个目标网络 $Q'(\theta_T)$,前者用于动作价值估计,后者用于提供稳定的目标Q值.两者结构一

致,目标网络参数通过软更新机制与评估网络同步.

两类智能体采用相似的网络结构,每个网络包含两个隐藏层,每层1024个节点,并使用 ReLU 作为激活函数.在输出层部分,网络采用决斗架构,因此分为两个分支:第一个分支输出一个标量,表示状态价值 $V(s)$;第二个分支通过线性层输出每个动作的动作优势值 $A(s, a)$.输入层与输出层的规模在两类智能体之间略有差异.为区分,路由智能体的网络结构以下标“ r ”表示,排序智能体的网络结构以下标“ s ”表示.具体而言,对于 $Q_r(\theta_E)$ 和 $Q'_r(\theta_T)$,输入层包含21个节点,动作优势输出维度为 $p+2$;而对于 $Q_s(\theta_E)$ 和 $Q'_s(\theta_T)$,输入层为16个节点,动作优势输出维度为 $p+4$.MTNGPDRL 的训练流程如算法3所示,首先利用 MTNGP 的协同进化生成多样化的路由与排序规则,并据此构建两类智能体的离散动作空间.由于规则在生成过程中已完成磨合,路由与排序智能体能够在该动作空间中直接开展联合训练,从而避免依赖额外的临时策略来稳定训练.

算法3 MTNGPDRL智能体训练

输入: 最大回合数 L , 每回合训练次数 N

输出: $Q_r(\theta_E)$, $Q'_r(\theta_T)$, $Q_s(\theta_E)$, $Q'_s(\theta_T)$, \mathcal{A}_r 和 \mathcal{A}_s

1 随机初始化 $Q_r(\theta_E)$, $Q'_r(\theta_T)$, $Q_s(\theta_E)$, $Q'_s(\theta_T)$ 以及经验回放缓冲区 M_r , M_s ;

2 $\mathcal{A}_r, \mathcal{A}_s \leftarrow$ MTNGP, $\pi_r \leftarrow Q_r(\theta_E)$, $\pi_s \leftarrow Q_s(\theta_E)$;

3 **for** $episode = 1$ **to** L **do**

4 随机生成一个 DWSPFC-RU 实例;

5 在策略 π_r, π_s 下进行仿真,收集经验并存入缓冲区;

6 **for** $i = 1$ **to** N **do**

7 更新 $Q_r(\theta_E)$, $Q'_r(\theta_T)$, $Q_s(\theta_E)$, $Q'_s(\theta_T)$;

8 $\pi_r \leftarrow Q_r(\theta_E)$, $\pi_s \leftarrow Q_s(\theta_E)$;

9 **返回** $Q_r(\theta_E)$, $Q'_r(\theta_T)$, $Q_s(\theta_E)$, $Q'_s(\theta_T)$, \mathcal{A}_r 和 \mathcal{A}_s ;

2.5 调度框架

MTNGPDRL 在雾计算环境中也采用分布式部署.路由智能体部署于用户设备上,负责将就绪任务卸载至合适的目标设备;排序智能体则运行在各类执行设备上,负责本地队列的调度.当任务转为就绪后,路由智能体结合当前可用设备的状态信息,从动作空间中选择调度策略,间接确定任务的分配位置.若目标设备处于空闲状态,任务立即开始执行;否则进入等待队列,当设备释放资源后,排序智能体依据本地观测,从规则集中挑选合适的排序规则,决定下一个执行任务.设备运行过程中会随机出现故障,此时相关任务进程中中断,修复之后设备重新开始运行.

3 实验分析

3.1 参数设置

在训练和测试环境中,用户设备上新工作流的到达服从参数为 λ_{new} 的泊松过程,其中 λ_{new} 设为0.1. 工作流 W_i 的截止时间 $DD_i = a_i + (WCT_i - a_i) \times DDT$, 其中 a_i 表示其到达时刻, $DDT \in [1.5, 2.5]$ 表示截止时间紧迫度.

实验采用五类经典科学工作流模型^[18],每个任务的上传和下载数据量范围为720至2880 MB,计算需求设定在 $[5 \times 10^4, 1.5 \times 10^5]$ 百万周期 (Mega Cycles) 区间内服从均匀分布.三类设备的相关信息根据表2设定的参数范围进行随机初始化.设备故障发生时间服从以故障发生率为参数的泊松过程,故障设备在持续一段修复时间后将恢复至正常可用状态.

表2 设备初始化参数

类型	用户设备	边缘服务器	云服务器
带宽 (MHz)	-	100	100
算力 (GHz)	[1.2, 2.5]	[2.5, 5]	[5, 10]
距离 (m)	-	[50, 100]	[200, 500]
传输功率 (W)	1	1	1
故障发生率	0.0001	0.0001	0.0001
修复用时 (s)	[5, 20]	[5, 20]	[5, 20]
存储容量 (TB)	1/2	8/16	-

式2所示传输模型中各参数设置为: $\sigma = 4$, $w_0 = -100$ dBm, $N_0 = -80$ dBm. 训练过程中,式25中路由智能体奖励函数的参数 α 设置为0.1,式35中排序智能体奖励的参数 β_{min} 和 β_{max} 分别设为-3和3.

在雾计算环境中,工作流到达速率和场景复杂度是决定系统负载与调度难度的关键因素.其中,工作流的到达频率主要取决于系统中用户设备的数量,较高的到达速率会显著增加系统负载;场景复杂度则由云/边缘服务器数量及工作流规模共同决定,规模越大,调度问题越复杂.基于此,本文设计了四类典型雾计算场景:低到达频率-低复杂度(LL)、高到达频率-低复杂度(HL)、低到达频率-高复杂度(LH)、高到达频率-高复杂度(HH),具体设置如表3所示.

表3 实验场景配置

场景	用户设备	边缘服务器	云服务器	工作流数量
LL	1	5	2	10
HL	3	5	2	10
LH	1	15	5	30
HH	3	15	5	30

后续实验中, 智能体将在四类场景下独立训练, 并在对应场景下进行性能评估。

3.2 实验配置

本文所提算法采用 Python 实现, 运行于配备 Intel® Core™ i5-13500H 2.6GHz 处理器、NVIDIA GeForce RTX 4060 Laptop GPU 以及 16GB 内存的计算平台。

MTNGP 所采用的函数集合为 $\{+, -, \times, \div, \max, \min\}$, 其中 “ \div ” 为保护除法, 当分母为 0 时返回 1. MTNGP 的参数配置如表 4 所示, 在进化过程中, 适应度评估阶段同时使用两个实例, 并在每一代进行刷新. 经过 50 次迭代后, 从最后一代中选取适应度最优的 4 个个体作为输出。

表4 MTNGP 参数设置

参数名	参数值
种群大小 P_{size}	1000
迭代次数 G	50
初始最小/最大深度	2 / 6
最大深度	8
交叉率	0.80
变异率	0.15
终端/非终端选择比例	10% / 90%
精英保留	10
半径 ρ	0
容量 κ	1

MTNGPDRL 智能体训练超参数设定见表 5. 训练过程中, 每个回合均在随机生成的环境中进行. 训练结束后, 在验证实例上评估各代策略, 并选取性能最优者作为最终调度策略。

表5 MTNGPDRL 智能体训练参数

参数	路由智能体	排序智能体
最大训练轮数 L	125	125
学习率 η	1×10^{-5}	1×10^{-5}
小批量样本规模 $ B $	256	256
软更新率 τ	5×10^{-3}	5×10^{-3}
折扣因子 γ	0.99	0.99
初始探索率	1	1
探索率衰减系数	5×10^{-3}	4×10^{-3}
最小探索率	0.375	0.5
每回合训练迭代次数	1	4
经验回放缓冲区容量	1,000,000	1,000,000
优化器	Adam	Adam

3.3 评价指标

后续所有实验均在独立随机生成的 50 个实例上进行评估. 为衡量算法在这些实例上的性能, 本文

采用以下两项指标进行分析。

(1) {平均总延迟 (Average Tardiness, AT)}: 用于衡量算法在多组测试实例上的整体调度效果. 设总实例数为 R , 第 i 次测试中算法产生的延迟时间为 $T_{target,i}$, 平均总延迟的计算公式如下:

$$AT_{target} = \frac{\sum_{i=1}^R T_{target,i}}{R} \quad (36)$$

(2) 标准化性能 (Normalized Performance, NP): 为减弱不同实例目标值差异的影响, 引入标准化性能指标^[19], 以某一算法或规则作为基准 (baseline), 对其他算法进行评估, NP 越大, 表示目标算法相对基准算法的性能增益越大. 算法在单个实例上的标准化性能计算公式如下:

$$NP_{target} = \frac{T_{baseline} - T_{target}}{T_{baseline}} \quad (37)$$

3.4 MTNGP 进化动作有效性验证

为验证 MTNGP 所生成调度动作的有效性, 将其自动进化得到的路由动作 (MTNGP-R1~R4) 和排序动作 (MTNGP-S1~S4) 分别与表 6 和表 7 中的人工启发式调度规则进行对比. WIQ 和 FCFS 分别作为路由和排序的标准化性能基准. 实验结果如图 2 和图 3 所示, 结果表明, MTNGP 自动进化生成的规则在四类场景中均显著优于人工规则, 证明其能够学习到更有效的调度策略。

表6 路由启发式调度规则

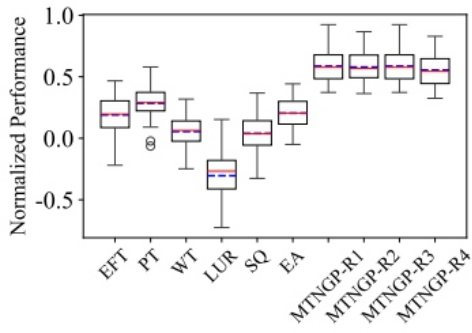
规则	含义
EFT	最早完成优先
PT	最短处理时间优先
WT	最短等待时间优先
LUR	最低利用率优先
SQ	最短等待队列优先
EA	最早空闲优先
WIQ	最小队列负载优先

表7 排序启发式调度规则

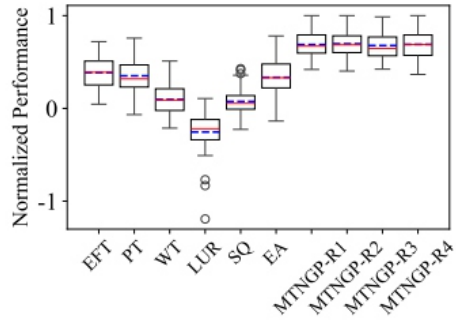
规则	含义
FCFS	先来先服务
SPT	最短处理时间优先
HCR	完成率最高优先
MS	最短松弛时间优先
HUR	上行排序值最高优先
RSR	剩余处理时间与松弛时间比值最高优先
EASPT	到达时间最早且处理时间最短优先
ATC	表观延误成本最高优先

3.5 MTNGPDRL 性能验证

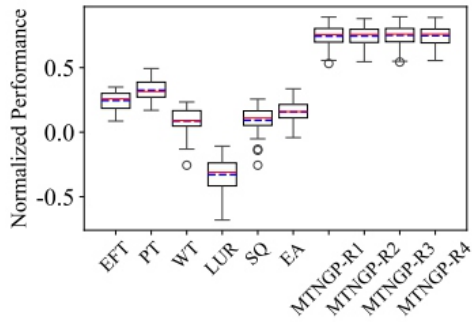
为评估 MTNGPDRL 调度性能, 本文将其与



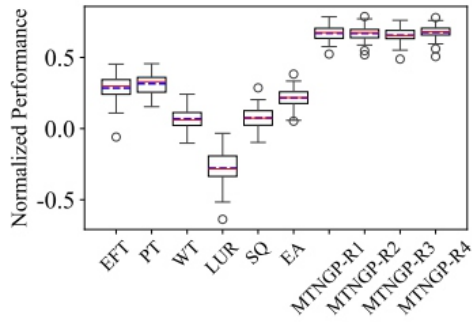
(a) LL



(b) HL



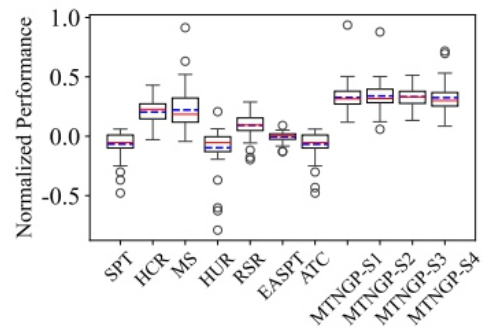
(c) LH



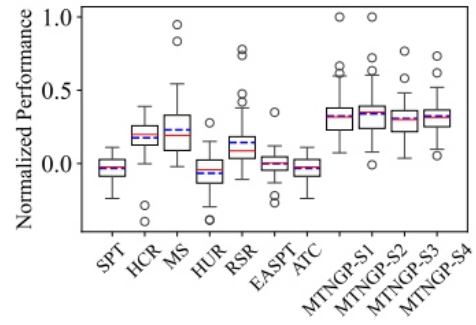
(d) HH

图2 路由进化动作有效性验证

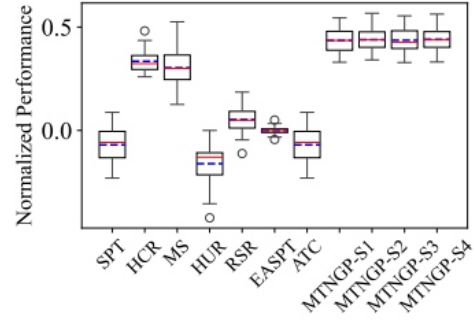
3.4 节实验中验证得到的两种最优人工路由规则 (PT 与 EFT) 及四种最优排序规则 (HCR、MS、RSR 和 EASPT) 的组合以及代表性进化方法 MTGP^[20] 进行对比. 此外, 本文还引入两种深度强化学习对比方法: 其一为基于人工规则构建排序动作空间的深度强化学习方法 (Deep Reinforcement Learning with Handcrafted Rules, DRL-HR), 其路由智能体采用端到端结构, 并通过动作掩码机制避免选择当前不可



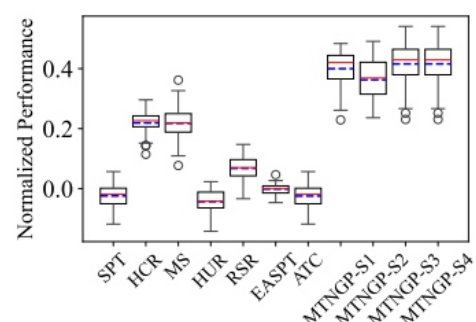
(a) LL



(b) HL



(c) LH



(d) HH

图3 排序进化动作有效性验证

用设备, 而排序智能体的动作空间由人工规则集构建而成; 其二为基于指针网络的深度强化学习方法 (Deep Reinforcement Learning with Pointer Network, DRL-PN), 其利用 Transformer 编码变长候选集合, 直接进行端到端的决策, 不依赖任何人工规则或进化规则. 标准化性能以 WIQ 和 FCFS 组合作为计算基准.

图4与表8给出了所有方法在不同场景下的标准化性能与平均总延迟。结果表明, MTNGPDRL 在所有场景中均取得最优性能; 相较于次优方法 DRL-HR, 其平均总延迟降低了 6.3%–18.9%。

3.6 泛化性能验证

为进一步验证所提方法的泛化性能, 将在四类基础场景中训练得到的模型迁移至三类扩展场景下进行测试。扩展场景1和2引入表9中任务更多、结构更复杂的工作流模型, 以评估算法在任务规模扩展下的调度性能。其中, 扩展场景1在规模A类和规模B类工作流中按约70%与30%的比例随机选取; 扩展场景2则在规模A、B、C三类工作流中按约50%、30%和20%的比例随机选取。扩展场景3进一步检验算法在资源规模扩展下的迁移与适应能力。上述三种扩展场景均与前文定义的四类调度场景(LL/HL/LH/HH)相结合, 共形成12种实验配置, 具体设置如表10所示。由于DRL-HR的全连接输出层维度固定且与设备数量绑定, 无法适应设备数量变化, 故不参与扩展场景3的对比。

表11和表12分别汇总了对应场景中的平均总延迟结果。结果显示, MTNGPDRL在扩展场景中表现显著领先于所有对比方法。与次优方法相比, MTNGPDRL的延迟降低比例约为2.2%–16.3%, 有效降低了平均总延迟, 充分证明了其能够有效应对工作流规模扩展带来的挑战。

扩展场景3中各方法的平均总延迟结果汇总于表13。结果表明, MTNGPDRL在扩展场景3中的表现仍优于所有对比方法。与次优方法DRL-PN相比, MTNGPDRL在扩展场景3中的平均总延迟降幅约为9.3%–30.3%, 证明MTNGPDRL在场景条件变化时仍能保持较好的适应能力, 并在大规模复杂环境

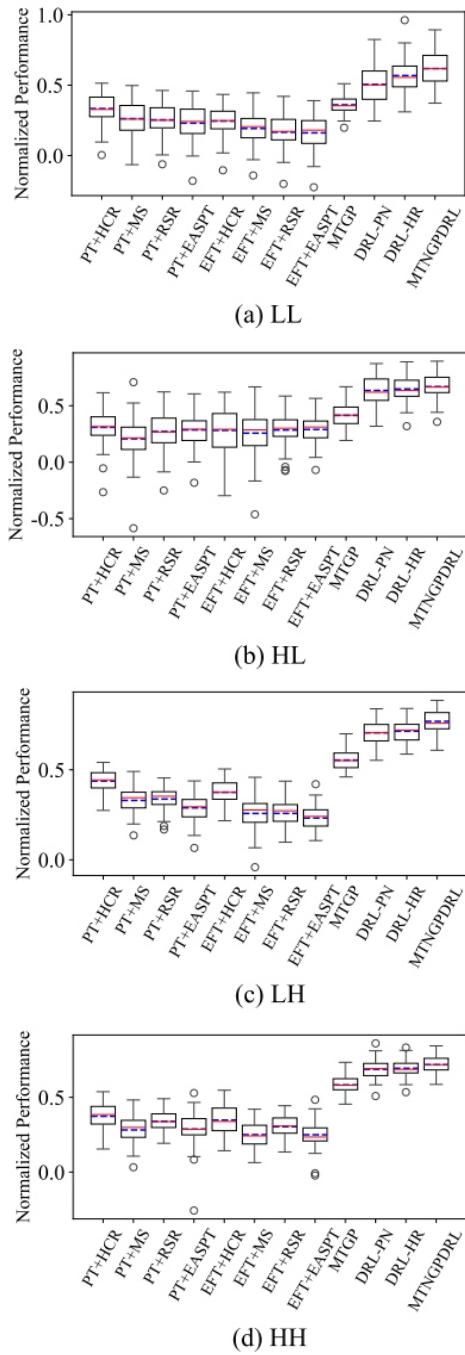


图4 MTNGPDRL 性能验证

表8 四类场景下不同算法平均总延迟结果比较

场景	PT+HCR	PT+MS	PT+RSR	PT+EASPT	EFT+HCR	EFT+MS	EFT+RSR	EFT+EASPT	MTGP	DRL-PN	DRL-HR	MTNGPDRL(↑)
LL	3432.86	3807.18	3821.20	3994.34	3849.39	4118.82	4218.04	4297.42	3291.46	2676.27	2318.53	2089.96
HL	3398.80	3872.96	3482.92	3437.95	3517.28	3661.95	3446.43	3431.55	2789.55	1826.07	1759.57	1649.16
LH	14290.25	16985.95	16812.90	18077.47	15847.76	18802.56	18791.07	19466.47	11365.56	7559.76	7323.85	5940.57
HH	16287.55	18645.96	17144.72	18510.32	16930.31	19460.68	18071.98	19446.28	10868.01	8193.92	8005.16	7384.22

表9 不同类型工作流包含任务数量

类型	规模A	规模B	规模C
Cybershake	30	50	100
Epigenomics	24	46	100
Inspiral	30	50	100
Montage	25	50	100
Sipht	30	60	100

下持续获得较优的调度性能。

4 结论

本文针对雾计算动态工作流调度中的资源不确定性构建相应的问题模型, 并提出 MTNGPDRL 方法进行求解。MTNGPDRL 将调度过程分解为路由与排序两个决策子问题, 并使用相应的智能体进行协

表10 扩展场景实验配置

类别	场景	用户设备	边缘服务器	云服务器	workflow数量	workflow类型
扩展1	LL	1	5	2	10	A/B
	HL	3	5	2	10	A/B
	LH	1	15	5	30	A/B
	HH	3	15	5	30	A/B
扩展2	LL	1	5	2	10	A/B/C
	HL	3	5	2	10	A/B/C
	LH	1	15	5	30	A/B/C
	HH	3	15	5	30	A/B/C
扩展3	LL	2	6	6	20	A
	HL	4	6	6	20	A
	LH	2	15	15	45	A
	HH	4	15	15	45	A

同优化. 通过引入 MTNGP, 自动演化生成多样高效的路由与排序规则, 有效克服了传统深度强化学习算法中动作空间固定及策略收敛不稳定的问题. 在此基础上, 构建了基于 MTNGP 和 D3QN 的多智能体深度强化学习框架. 在多种典型科学 workflow 及不同规模的雾计算场景下开展仿真实验, 结果表明所提 MTNGPDRL 方法的调度性能显著优于所有对比算法. 在大规模异构与高不确定场景中, MTNGPDRL 仍能保持较好的泛化性能.

未来研究将进一步考虑能耗约束、通信代价等因素, 以构建更贴近实际应用需求的雾计算调度优化模型; 同时, 引入故障预测、风险感知等机制, 以提升所提算法在实际雾计算应用场景中的有效性.

表11 扩展场景 1 下不同算法平均总延迟结果比较

场景	PT+HCR	PT+MS	PT+RSR	PT+EASPT	EFT+HCR	EFT+MS	EFT+RSR	EFT+EASPT	MTGP	DRL-PN	DRL-HR	MTNGPDRL(↑)
LL	4954.21	5457.93	5606.27	5944.10	5221.00	5837.82	6070.96	6095.60	4307.05	4168.63	3586.67	3353.66
HL	5066.69	5643.71	5183.63	5143.37	4830.35	5348.00	5131.47	5031.59	4084.36	3222.76	3000.11	2767.66
LH	18796.89	21762.11	21786.50	23572.39	19958.58	23711.59	23594.27	25165.81	14686.64	11770.51	11195.44	9366.39
HH	21110.70	24761.50	22805.29	24603.10	22949.37	26322.91	24172.07	26844.15	14421.62	13350.08	12721.91	11722.48

表12 扩展场景 2 下不同算法平均总延迟结果比较

场景	PT+HCR	PT+MS	PT+RSR	PT+EASPT	EFT+HCR	EFT+MS	EFT+RSR	EFT+EASPT	MTGP	DRL-PN	DRL-HR	MTNGPDRL(↑)
LL	7094.43	7749.65	8134.27	8436.17	7266.27	8326.28	8353.04	8719.31	5847.11	6376.77	5446.71	5328.47
HL	7226.15	8129.43	7385.31	7865.71	7292.75	8088.02	7412.76	7867.55	5981.72	5840.15	5279.38	4934.89
LH	24750.62	29585.87	29227.47	32385.72	26552.60	31387.66	30665.07	33744.61	20842.19	18457.64	17240.42	15262.35
HH	27580.09	32096.43	29813.16	33039.41	28300.67	32972.90	30596.09	33950.03	18671.61	20117.55	19148.95	17445.47

表13 扩展场景 3 下不同算法平均总延迟结果比较

场景	PT+HCR	PT+MS	PT+RSR	PT+EASPT	EFT+HCR	EFT+MS	EFT+RSR	EFT+EASPT	MTGP	DRL-PN	MTNGPDRL(↑)
LL	7905.91	8978.53	8847.99	8796.71	8869.44	10493.67	9971.37	9898.89	6906.97	4119.81	3019.68
HL	8463.51	9502.38	8693.35	8942.18	9387.75	10388.70	9554.03	9594.10	6557.59	3529.96	3203.33
LH	22923.07	27660.33	26256.50	28349.46	25784.04	31657.94	29942.65	31314.56	15158.04	6299.02	4393.00
HH	25501.55	29693.07	27388.70	29573.35	29004.47	33881.80	29735.51	31998.71	16393.69	8107.18	6122.19

参考文献 (References)

- [1] Chen Y R, Zhang Y, Zhuang Y B, et al. Efficient and secure blockchain consensus algorithm for heterogeneous industrial Internet of Things nodes based on double-DAG[J]. *IEEE Transactions on Industrial Informatics*, 2024, 20(4): 6300-6312.
- [2] 刘连玉, 巩在武, 张雪, 等. 应急情景下融合改进 D*Lite 算法和 DWA 算法的无人驾驶汽车路径规划[J]. *控制与决策*, 2025, 40(10): 2985-2994. (Liu L Y, Gong Z W, Zhang X, et al. Fusion of improved D*Lite algorithm and DWA algorithm for driverless vehicle path planning in emergency scenarios[J]. *Control and Decision*, 2025, 40(10): 2985-2994.)
- [3] 林雨嘉, 李金海, 宁焕生, 等. 数字孪生技术在智慧医疗与健康领域的技术、应用与挑战综述[J]. *工程科学学报*, 2025, 47(9): 1812-1824. (Lin Y J, Li J H, Ning H S, et al. Review of technology, application, and challenge of digital twins in smart healthcare[J]. *Chinese Journal of Engineering*, 2025, 47(9): 1812-1824.)
- [4] Asghari A, Sohrabi M K. Server placement in mobile cloud computing: A comprehensive survey for edge computing, fog computing and cloudlet[J]. *Computer Science Review*, 2024, 51: 100616.
- [5] Srirama S N. A decade of research in fog computing: Relevance, challenges, and future directions[J]. *Software: Practice and Experience*, 2024, 54(1): 3-23.
- [6] Hawaou K S, Kamla V C, Yassa S, et al. Industry 4.0 and industrial workflow scheduling: A survey[J]. *Journal of Industrial Information Integration*, 2024, 38: 100546.
- [7] 尹璐, 周俊龙, 孙晋, 等. 不确定性感知的边缘计算任务调度算法[J]. *控制与决策*, 2024, 39(7): 2405-2413.

- (Yin L, Zhou J L, Sun J, et al. Uncertainty-aware task scheduling algorithm in edge computing environments[J]. *Control and Decision*, 2024, 39(7): 2405-2413.)
- [8] Wang X, Wang S, Liang X X, et al. Deep reinforcement learning: A survey[J]. *IEEE Transactions on Neural Networks and Learning Systems*, 2024, 35(4): 5064-5078.
- [9] 王艳红, 付威通, 张俊, 等. 基于改进近端策略优化算法的柔性作业车间调度[J]. *控制与决策*, 2025, 40(6): 1883-1891.
(Wang Y H, Fu W T, Zhang J, et al. Flexible job-shop scheduling based on improved proximal policy optimization algorithm[J]. *Control and Decision*, 2025, 40(6): 1883-1891.)
- [10] Jayanetti A, Halgamuge S, Buyya R. Deep reinforcement learning for energy and time optimized scheduling of precedence-constrained tasks in edge — cloud computing environments[J]. *Future Generation Computer Systems*, 2022, 137: 14-30.
- [11] Zhang J W, Cheng L, Liu C, et al. Cost-aware scheduling systems for real-time workflows in cloud: An approach based on Genetic Algorithm and Deep Reinforcement Learning[J]. *Expert Systems with Applications*, 2023, 234: 120972.
- [12] Chen G X, Qi J, Sun Y, et al. A collaborative scheduling method for cloud computing heterogeneous workflows based on deep reinforcement learning[J]. *Future Generation Computer Systems*, 2023, 141: 284-297.
- [13] Pan J H, Wei Y. A deep reinforcement learning-based scheduling framework for real-time workflows in the cloud environment[J]. *Expert Systems with Applications*, 2024, 255: 124845.
- [14] Wang Z, Zhan W H, Duan H C, et al. Deep-reinforcement-learning-based continuous workflows scheduling in heterogeneous environments[J]. *IEEE Internet of Things Journal*, 2025, 12(10): 14036-14050.
- [15] Jayanetti A, Halgamuge S, Buyya R. Multi-agent deep reinforcement learning framework for renewable energy-aware workflow scheduling on distributed cloud data centers[J]. *IEEE Transactions on Parallel and Distributed Systems*, 2024, 35(4): 604-615.
- [16] He H Y, Gu Y, Hu Y, et al. Real-time workflow scheduling in hybrid clouds with privacy and security constraints: A deep reinforcement learning approach[J]. *Expert Systems with Applications*, 2025, 278: 127376.
- [17] He Y F, Xu Z, Lin J, et al. Deep reinforcement learning with evolved actions for dynamic workflow scheduling in distributed fog computing[J]. *Neurocomputing*, 2026, 677: 133115.
- [18] Li Y Z, He Y F, Lin J, et al. A reinforcement learning-based population hyper-heuristic for energy-efficient cloud workflow scheduling problem[J]. *IEEE Transactions on Services Computing*, 2025, 18(5): 2545-2558.
- [19] Liu R K, Piplani R, Toro C. Deep reinforcement learning for dynamic scheduling of a flexible job shop[J]. *International Journal of Production Research*, 2022, 60(13): 4049-4069.
- [20] Xu M, Mei Y, Zhu S Q, et al. Genetic programming for dynamic workflow scheduling in fog computing[J]. *IEEE Transactions on Services Computing*, 2023, 16(4): 2657-2671.

作者简介

林剑 (1983–), 男, 教授, 博士, 博士生导师, 主要研究方向为智能计算、调度优化, E-mail: linjian1001@126.com;

徐占 (2000–), 男, 硕士生, 主要研究方向为智能计算、深度强化学习, E-mail: dsxx4501@163.com;

何以凡 (1995–), 男, 讲师, 博士, 主要研究方向为智能计算、调度优化, E-mail: yifanhe@zufe.edu.cn;

许晨昊 (2000–), 男, 硕士生, 主要研究方向为智能计算、调度优化, E-mail: chenhao_xu@zufe.edu.cn;

赵治涵 (2001–), 男, 硕士生, 主要研究方向为智能计算、调度优化, E-mail: 241506021018@zufe.edu.cn.