Tsinghua Science and Technology
Doi: 10.26599/TST.2026.9010046

Research Article

Deadline-Aware Cooperative Scheduling of DAG Tasks in Multi-Edge Computing for Data-Intensive Real-Time Services in Digital Infrastructure

Youke Wu, Kaiyuan Zheng, Yaxin Mei, Yuzhu Liang (✉), Haiyang Huang, and Tian Wang

Abstract With the rapid growth of data volume and increasingly stringent latency requirements in data-intensive real-time services within digital infrastructure, mobile edge computing (MEC) must process complex workloads efficiently under constrained computation and bandwidth resources. In practice, many workloads can be modeled as directed acyclic graphs (DAGs), where subtasks are subject to strict precedence constraints. Meanwhile, cooperative multi-edge environments feature heterogeneous computing capabilities, dynamic queue backlogs, and stochastic arrivals, making offline or static heuristics difficult to adapt. This paper investigates the online scheduling and offloading of dependency-constrained DAG tasks in a cooperative multi-edge setting, aiming to minimize the long-term average makespan, and proves that the resulting optimization problem is NP-hard. To this end, we propose an online cooperative scheduling framework, termed DRL-TCES. Specifically, DRL-TCES leverages a Top-k sampling based Deep Q-Network (DQN) to dynamically select a cooperative edge set, and then determines the workload partition ratios by enforcing completion-time balancing across selected nodes, jointly accounting for transmission, queuing, and computation overheads to enhance parallel execution efficiency. Extensive simulations show that DRL-TCES reduces makespan by up to 63.9% and achieves a 98.5% task success rate over strong baselines, demonstrating superior scalability and robustness.

Keywords Mobile Edge Computing (MEC), DAG, Task Scheduling, Multi-edge Cooperation.

1 Introduction

With the rapid proliferation of mobile Internet and Internet-of-Things (IoT) applications, a wide range of data-intensive and latency-sensitive services have emerged, including real-time sensing, online analytics, and interactive applications [1–3]. These services constitute essential components of modern digital infrastructure, which increasingly relies on distributed computing and communication resources to ensure real-time responsiveness, service reliability, and scalable data processing. Mobile Edge Computing (MEC) alleviates end-to-end latency and reduces cloud congestion by offloading computation tasks to edge servers (ESs) close to end users, and has become a key infrastructure for supporting next-generation real-time services and computing power networks [4, 5]. However, in practical deployments, a single ES usually has limited computing, storage, and bandwidth resources, while significant heterogeneity and load imbalance exist across different edge nodes [2]. As a result, isolated edge execution often fails to consistently satisfy stringent latency requirements under high concurrency and dynamic workloads.

To overcome these limitations, multi-edge collaboration has emerged as a promising paradigm, where multiple ESs cooperatively share computation and communication workloads to enhance processing capacity, service stability, and system robustness [6–8]. By exploiting spatial resource diversity and parallel execution opportunities, collaborative MEC can significantly improve task completion efficiency and resilience, which is crucial for sustaining high-quality service delivery

• Youke Wu is with the School of Economics and Management, Wuyi University, Jiangmen 529020, Guangdong, China. E-mail: wyk@wyu.edu.cn.
• Kaiyuan Zheng, Yaxin Mei, and Yuzhu Liang are with the Institute of Artificial Intelligence and Future Networks, Beijing Normal University, Zhuhai 519087, China. E-mail: zhky@mail.bnu.edu.cn; yxmei@mail.bnu.edu.cn; yuzhuliang@bnu.edu.cn.
• Haiyang Huang is with the School of Artificial Intelligence and Computer Science, Nantong University, Nantong 226019, China. E-mail: haiyanghuang@mail.bnu.edu.cn.
• Tian Wang is with the Engineering Research Center of Cloud-Edge Intelligent Collaboration on Big Data, Ministry of Education, Beijing Normal University, Zhuhai, Guangdong, China; the College of Computer and Data Science, Fuzhou University, China; and the School of Information Engineering, Huzhou University, China. E-mail: tianwang@bnu.edu.cn.
* Corresponding author: Yuzhu Liang.
Manuscript received: 20xx-xx-xx; revised: 20xx-11-xx; accepted: 20xx-00-xx

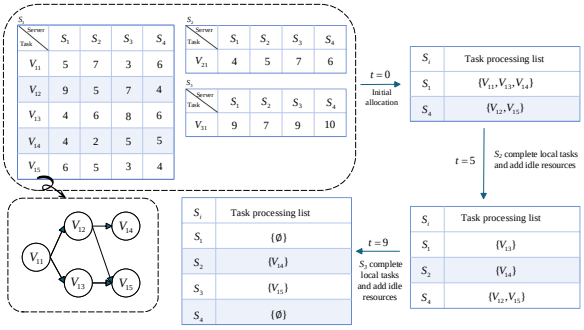

Fig. 1 An example of multi-edge cooperative DAG scheduling

in large-scale digital infrastructure. Nevertheless, effectively orchestrating collaboration among multiple ESs in realistic environments remains highly challenging, especially when task structures and system states are dynamic.

Unlike traditional independent task offloading, many real-world applications consist of multiple interdependent subtasks with strict precedence constraints [9–11]. Such dependency-aware tasks are commonly modeled as directed acyclic graphs (DAGs), where a subtask can only be executed after all its predecessors have completed [12, 13]. In a multi-edge collaborative setting, DAG dependencies introduce strong scheduling couplings: the scheduler must jointly consider (i) execution order constraints imposed by dependencies, (ii) additional latency caused by inter-edge data transmission, (iii) heterogeneous queuing and computing capabilities of ESs, and (iv) online uncertainty induced by stochastic task arrivals and time-varying system states. These factors jointly amplify critical paths in dependency chains, making makespan minimization substantially more difficult in dynamic environments [14]. Such challenges are particularly pronounced in data-intensive real-time services, where strict latency guarantees are indispensable for maintaining service quality in digital infrastructure.

Although collaborative MEC scheduling has been extensively studied, existing works still exhibit notable limitations when handling dependency-aware tasks [15]. Many approaches simplify the problem by assuming independent tasks, predictable system states, or limited collaboration scopes, such as restricting execution to a single ES or a small fixed subset of nodes [16–18]. While these assumptions reduce computational complexity, they fail to capture the combinatorial explosion and dynamic load fluctuations inherent in large-scale, dependency-aware multi-edge collaboration, leading to degraded performance under realistic workloads and undermining the scalability required by emerging digital services.

Fig. 1 illustrates an example of the multi-edge cooperative DAG scheduling process. At time $t = 0$, dependent subtasks are initially allocated across multiple ESs to minimize the overall makespan, where V_{11}, V_{13}, V_{14} are assigned to S_1 and V_{12}, V_{15} are dispatched to the idle server S_4 for parallel execution. As S_2 and S_3 complete their local workloads at $t = 5$ and $t = 9$, respectively, newly released resources are incorporated into the collaboration set, enabling dynamic reallocation of ready subtasks. This event-driven adjustment process continuously adapts scheduling decisions to real-time resource availability and effectively shortens the critical path under dependency constraints, thereby enhancing system responsiveness and utilization efficiency.

Motivated by these observations, we investigate a more practical and challenging problem: how to perform online scheduling and offloading for dependency-aware tasks in a multi-edge collaborative MEC environment with stochastic arrivals and heterogeneous resources, so as to minimize the long-term average task makespan while maximizing task success rate [19]. We formalize tasks as DAGs and formulate an online optimization problem targeting makespan minimization. Further analysis reveals that the problem is inherently computationally intractable, as it can be reduced to the classical NP-hard makespan minimization problem, which calls for efficient online decision-making mechanisms without relying on complete prior knowledge. To address this challenge, we propose *DRL-TCES*, an online multi-edge collaborative scheduling framework for dependency-aware tasks. The key idea is to leverage deep reinforcement learning to dynamically select high-quality collaborating edge sets under partial observability. Once a collaboration set is determined, subtasks are distributed among selected ESs to mitigate bottleneck effects on the critical path. In particular, we adopt a Top-k sampling–based DQN to cope with the rapidly expanding collaboration action space as the number of ESs increases, improving decision efficiency while maintaining sufficient exploration. Moreover, the learning objective explicitly focuses on controlling the maximum completion latency, ensuring that collaboration gains are effectively translated into makespan reduction for data-intensive real-time services.

The main contributions of this paper are summarized as follows:

- We formulate a dependency-aware online scheduling problem for multi-edge collaborative MEC, capturing DAG-structured tasks, stochastic arrivals, and heterogeneous edge resources, and show that the problem is NP-hard.
- We design DRL-TCES, a deep reinforcement learning–based scheduling framework that dynamically selects

collaborating edge sets and mitigates critical-path bottlenecks through makespan-oriented decision making.
- Extensive experiments demonstrate that DRL-TCES significantly outperforms representative baselines, achieving up to a 63.9% reduction in average task makespan while maintaining a task success rate of up to 98.5%, indicating strong scalability and robustness for real-time data-processing pipelines in digital infrastructure.

The remainder of this paper is organized as follows. Section II reviews related work. Section III presents the system model and problem formulation. Section IV details the design of DRL-TCES. Section V reports experimental settings and performance evaluations. Section VI concludes the paper and discusses future research directions.

2 Related Work

2.1 Dependency-Aware Task Offloading

Dependency-aware task offloading has attracted extensive attention in recent years, where applications are commonly modeled as directed acyclic graphs (DAGs) to explicitly represent precedence constraints and data dependencies among subtasks. In such formulations, vertices correspond to computational components, while directed edges capture execution ordering and intermediate data transfers, providing a structured abstraction for complex applications deployed in edge and cloud–edge systems. Compared with independent-task offloading, dependency-aware offloading requires joint consideration of execution order, inter-task synchronization, communication overhead, and heterogeneous resource availability, which substantially increases modeling complexity and decision dimensionality. Accordingly, prior studies typically aim to optimize end-to-end performance metrics such as makespan or critical-path delay, while satisfying real-time constraints and energy budgets under limited edge resources.

From the perspective of *offloading under incomplete or uncertain system information*, Dai *et al.* [20] studied dependent-task offloading in scenarios where system-side information is unavailable or partially observable. Their work focuses on decision robustness in the presence of uncertainty, revealing fundamental challenges in dependency-aware offloading without full knowledge of execution environments. From the perspective of *learning-based dependency-aware offloading*, Chen *et al.* [21] incorporated task dependency information into reinforcement learning models to guide offloading decisions in dynamic edge computing systems. To address *real-time requirements* and heterogeneous task structures, including combinations of dependent and parallel subtasks, Chen *et al.* [22] further proposed a deep reinforcement learning based strategy for cloud–edge environments that adapts to time-varying workloads. Beyond single-objective optimization, Tong *et al.* [23] investigated multi-objective dependency-aware offloading and adopted a federated DQN framework to balance competing objectives, such as latency and energy consumption, under distributed learning settings. In addition, Zhang *et al.* [24] examined dependent-task offloading in cloud–edge–device collaborative systems, taking into account practical system factors such as heterogeneous service capabilities and cross-layer coordination.

Limitations. While the above studies establish important foundations for dependency-aware task offloading, they predominantly emphasize *offloading decision policies or learning frameworks* at an abstract level. In most cases, the execution and scheduling of dependent subtasks on shared edge resources are treated implicitly or assumed to be resolved by underlying systems. However, in realistic deployments, multiple DAG-based applications may arrive concurrently and contend for the same edge resources, making *online, fine-grained scheduling* an integral part of dependency-aware offloading. The lack of explicit consideration of runtime scheduling interactions and resource contention limits the applicability of existing approaches in highly dynamic environments with fluctuating workloads and resource availability. In summary, existing dependency-aware offloading studies mainly optimize whether and where subtasks should be offloaded, but they rarely model how multiple dependent subtasks are scheduled online under shared and time-varying edge resource contention. This limits their applicability to realistic multi-task MEC systems, where fine-grained scheduling decisions directly affect end-to-end makespan.

2.2 Edge-Cooperative Dependency-Aware Offloading

More recently, edge-cooperative computing has been investigated as a complementary paradigm to further enhance the performance of dependency-aware applications by enabling *multi-edge* or *edge–cloud* collaboration [25–27]. In such systems, different stages of a DAG-based application may be distributed across multiple edge nodes, allowing workloads to be spatially decomposed and executed in parallel, thereby alleviating local resource bottlenecks and improving overall system utilization. Compared with single-edge offloading, edge-cooperative dependency-aware offloading introduces additional challenges related to inter-edge coordination, task placement consistency, and dynamic resource heterogeneity.

In vehicular edge computing environments, Zhao *et al.* proposed MESON [28], which considers both task dependencies and user mobility in offloading decisions. This line of work

emphasizes the importance of incorporating mobility dynamics into dependency-aware offloading under distributed edge infrastructures. Building upon this direction, Chen *et al.* [29] leveraged digital-twin-assisted reinforcement learning to support mobility-aware dependent task offloading, using virtual replicas of physical systems to improve decision robustness under mobility uncertainty. Beyond mobility-centric scenarios, several studies have explored *learning-based dependency-aware offloading and scheduling* in cooperative edge environments. Wang *et al.* [30] modeled dependent task offloading as a Markov decision process and employed deep reinforcement learning to minimize task completion delay, explicitly accounting for DAG precedence constraints in edge computing systems. Zhao *et al.* [31] further integrated *service caching* into dependency-aware offloading, demonstrating that joint consideration of caching and dependent task placement can reduce latency and backhaul usage in mobile edge computing.

Limitations. Despite these advances, existing studies primarily concentrate on *offloading decision optimization* for individual workflows or limited cooperative scopes, often relying on simplified execution or scheduling assumptions. In many cases, the execution of dependent subtasks is abstracted at a coarse level, and the impact of *online scheduling interactions* among multiple concurrent workflows is not explicitly modeled. As a result, the problem of *cooperative execution and scheduling of multiple dependency-constrained workflows* under strong multi-edge resource contention remains insufficiently addressed. This gap motivates the need for dependency-aware offloading frameworks that explicitly integrate online scheduling and contention-aware coordination across multiple edge nodes. Moreover, most existing cooperative approaches consider only individual workflows or simplified cooperation patterns, and thus do not fully capture the joint challenges of stochastic arrivals, heterogeneous queue backlogs, and concurrent DAG execution across multiple edge servers. As a result, the integration of dependency-aware online scheduling and multi-edge cooperative execution remains insufficiently explored.

Based on the above discussion, the main research gaps can be summarized as follows. First, existing studies predominantly focus on offloading decision policies, while explicit online scheduling under shared multi-edge resource contention is insufficiently addressed. Second, current cooperative methods usually assume simplified workflow settings or limited collaboration scopes, making them less suitable for dynamic environments with multiple concurrent DAG tasks. Third, few works jointly consider cooperative edge-set selection and workload partitioning with the explicit objective of reducing long-term makespan under stochastic arrivals and heterogeneous resources. These gaps motivate us to develop DRL-TCES, which integrates dependency-aware online scheduling, dynamic cooperative edge selection, and makespan-oriented workload allocation into a unified framework.

3 System Model

3.1 System Model

The system comprises a set of edge servers (ESs), denoted as $\mathcal{S} = \{1, 2, \cdots, S\}$, which are interconnected via a high-bandwidth wired core network. System time is discretized into equal-length time slots of duration Δt, denoted by $\mathcal{T} = \{1, \cdots, T\}$. Each ES $s \in \mathcal{S}$ possesses a heterogeneous computing capacity f_s, and the transmission rate between ES s and ES s' is given by $r_{s,s'}$. The main symbols and notations used throughout this paper are summarized in Table 1.

3.2 Task Model

The task arrival process follows a Bernoulli process. In each time slot t, a new task $j \in \mathcal{J}_t$ arrives at each ES s with probability p_{job}. Each arriving task j is represented as a DAG $\mathcal{G}_j = (\mathcal{V}_j, \mathcal{E}_j)$, where $\mathcal{V}_j = \{V_1, \ldots, V_{N_j}\}$ is the set of subtasks of task j, and $\mathcal{E}_j \subseteq \mathcal{V}_j \times \mathcal{V}_j$ defines the precedence constraints (i.e., dependencies) among these subtasks. For each subtask $v \in \mathcal{V}_j$, the attributes include the input data size d_v, the computation density ρ_v, the source ES $s_v \in \mathcal{S}$, and the arrival time T_v^{arr}. The set of all predecessors of subtask v is denoted by $Pred(v)$.

A variable $x_{v,s} \in [0, 1]$ is used to represent the fraction of subtask v allocated to ES s, with:

$$\sum_{s \in \mathcal{S}} x_{v,s} = 1, \quad \forall j, \forall v \in \mathcal{V}_j. \tag{1}$$

For each ES, it may assign local fraction subtasks to itself or receive fraction subtasks sent from other ESs. These subtasks enter the task queue and are dequeued immediately upon completion. The task queue on ES s is defined as $\mathcal{Q}_s = \{Q_1, \cdots, Q_{N'_s}\}$. The fraction subtasks set is denoted by $\mathcal{W}_v = \{W_1, \cdots, W_S\}$. Consequently, for each fraction subtask $w \in \mathcal{W}_v$, its data size is $d_w = x_{v,s} \cdot d_v$, its computational density is $\rho_w = x_{v,s} \cdot \rho_v$, its source ES is $s_w = s_v$, and its enqueue time is defined as T_{w,s_w}^{enq}.

For each fraction subtask w, the transmission time is:

$$T_{w,s}^{\text{trans}} = \frac{d_w}{r_{s_w,s}}. \tag{2}$$

The processing time for fraction subtask $w \in \mathcal{Q}_s$ on ES s is:

$$T_{w,s}^{\text{pro}} = \frac{\rho_w d_w}{f_s} = \frac{z_w}{f_s}, \tag{3}$$

Table 1 Key notations.

Symbol	Definition
$\mathcal{S}$	Set of edge servers (ESs)
S	Total number of edge servers
s, s'	Index of edge servers
$\mathcal{T}$	Set of time slots
Δt	Duration of each time slot
f_s	Computing capacity of ES s
$r_{s,s'}$	Transmission rate between ES s and ES s'
$\mathcal{J}_t$	Set of tasks arriving at time slot t
p_{job}	Task arrival probability
j	Index of task
$\mathcal{G}_j = (\mathcal{V}_j, \mathcal{E}_j)$	DAG of task j
$\mathcal{V}_j$	Set of subtasks of task j
$\mathcal{E}_j$	Precedence constraints among subtasks
N_j	Number of subtasks in task j
v, u	Index of subtasks
d_v	Input data size of subtask v
ρ_v	Computation density of subtask v
s_v	Source ES of subtask v
T_v^{arr}	Arrival time of subtask v
$Pred(v)$	Set of predecessor subtasks of v
$x_{v,s}$	Fraction of subtask v allocated to ES s
$\mathcal{Q}_s$	Task queue on ES s
$\mathcal{W}_v$	Set of fraction subtasks generated from v
w	Index of fraction subtask
d_w	Data size of fraction subtask w
ρ_w	Computation density of fraction subtask w
s_w	Source ES of fraction subtask w
$T_{w,s}^{\text{enq}}$	Enqueue time of w at ES s
$T_{w,s}^{\text{trans}}$	Transmission time of w to ES s
$T_{w,s}^{\text{pro}}$	Processing time of w on ES s
z_w	Workload of fraction subtask w
l_w	Index of fraction subtask w in queue $\mathcal{Q}_s$
$T_{w,s}^{\text{que}}$	Queueing time of w on ES s
$T_{w,s}^{\text{comp}}$	Completion time of w on ES s
T_v^{comp}	Completion time of subtask v
T_v^{ready}	Ready time of subtask v
α	Output data size scaling factor
C_j	Makespan of task j
D_j	Deadline of task j
$C_t^{\max}$	Maximum makespan among all tasks at t

where z_w is the workload of fraction subtask w. Let the index of fraction subtask w in queue $\mathcal{Q}_s$ be denoted by l_w. The first fraction subtask is the task queue is $w_1^* = \arg\min_{w' \in \mathcal{Q}_s} l_{w'}$. The queuing time for fraction subtask w is calculated as:

$$T_{w,s}^{\text{que}} = \sum_{l_{w'} < l_w, w' \in \mathcal{Q}_s} T_{w',s}^{\text{pro}} + T_{w_1^*,s}^{\text{enq}} - T_{w,s}^{\text{enq}}. \tag{4}$$

The completion time for fraction subtask q on ES s is:

$$T_{w,s}^{\text{comp}} = T_{w,s}^{\text{enq}} + T_{w,s}^{\text{pro}}. \tag{5}$$

The completion time of subtask v is the maximum value among the sum of the completion time and the transmission time for each of its fraction subtasks.

$$T_v^{\text{comp}} = \max_{w \in \mathcal{W}_v} \{T_w^{\text{comp}} + \alpha T_{w,s_w}^{\text{trans}}\}, \tag{6}$$

where $0 < \alpha < 0.2$[32] is a parameter indicating that the size of the processed result data is typically smaller than that of the original task.

Due to the dependencies, subtask v must not be processed until obtaining the results of all predecessors. We define it as the ready time T_v^{ready}

$$T_v^{\text{ready}} = \max_{u \in Pred(v)} (T_u^{\text{comp}} + \alpha \frac{d_u}{r_{s_u,s_v}}). \tag{7}$$

The formulas for T_w^{enq}, T_v^{ready}, and T_v^{comp} form a dependency cycle. When tasks arrive, the computation begins with subtasks that have zero in-degree, i.e., subtasks without predecessors, before proceeding to handle those with dependencies. Therefore, the enqueue time for fraction subtask w on ES s is:

$$T_w^{\text{enq}} = T_v^{\text{ready}} + T_{w,s_w}^{\text{trans}}. \tag{8}$$

The makespan of task j is:

$$C_j = \max_{v \in \mathcal{V}_j} T_v^{\text{comp}}. \tag{9}$$

A task j is considered failed if its makespan C_j exceeds a given threshold D_j:

$$C_j < D_j . \forall j \in \mathcal{J}_t. \tag{10}$$

The makespan for the entire set of tasks is defined as:

$$C_t^{\max} = \max_{j \in \mathcal{J}_t} C_j. \tag{11}$$

3.3 Problem Definition

The objective is to minimize the long-term average makespan of DAG tasks over discrete time slots. The problem can be formulated as:

$$\mathbf{P}: \min_{x_{v,s}} \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} C_t^{\max}, \tag{12}$$

$$\text{s.t.} \quad Eq.(1), (10). \tag{13}$$

Here, $C_t^{\max}$ denotes the maximum makespan among all tasks arriving at time slot t, which is consistent with the slot-based system model.

3.4 Problem Analysis

Theorem 1. *The proposed optimization problem* $\mathbf{P}$ *is NP-hard.*

Proof. We prove the NP-hardness of $\mathbf{P}$ via a polynomial-time reduction from the classical Multiprocessor Scheduling Problem (MSP), which is NP-hard [33].

Consider a special case of problem $\mathbf{P}$ with only two ES (S=2), no dependencies among subtasks (each task consists of a single subtask), and no data transmission (i.e., α=0).

Furthermore, restrict the allocation variable $x_{v,s}$ to be binary, i.e., $x_{v,s} \in \{0, 1\}$. This means each subtask must be processed entirely on one ES. The processing time of subtask v on ES s is then $\rho_v d_v / f_s$ if $x_{v,s} = 1$, and 0 otherwise. Since all subtasks arrive simultaneously (or in any order with the same arrival time), their sequence on each ES can be chosen arbitrarily. The makespan of the task set is the maximum completion time among all subtasks, which is exactly the makespan of scheduling these subtasks on two unrelated machines (unrelated if $f_1 \neq f_2$; identical if $f_1 = f_2$).

The problem of minimizing the makespan on two machines is NP-hard. Therefore, our problem $\mathbf{P}$ is at least as hard as this NP-hard problem, and hence $\mathbf{P}$ is NP-hard. □

4 Methods

4.1 Proposed DRL-Based Online Scheduling Method

To address the complex online optimization problem $\mathbf{P}$, a Deep Reinforcement Learning based Top-k Cooperative Edge Scheduling for Dependent Tasks (DRL-TCES) is proposed. The core insight is to decompose the joint decision into a sequential process. First, a novel Top-k DQN network dynamically selects a set of cooperative ESs for each arriving subtask. Second, the optimal allocation of subtask fractions is determined analytically by applying the principle of equalizing completion times. In the proposed framework, collaborative edge computing is implemented in a distributed manner without relying on a centralized controller. For each arriving subtask, its source edge server acts as the decision-making agent, which observes the current system state and selects a cooperative set of edge servers via the Top-k DQN policy. The selected servers then jointly execute the subtask in a parallel manner through workload partitioning.

4.2 Formulation of the DRL-TCES Components

4.2.1 State Space

The state for a subtask v arriving at time t must encapsulate both its intrinsic demand and the extrinsic system congestion. We define the state vector as:

$$\mathbf{S}_{v,t} = [d_v, \rho_v, z'_{t,1}, z'_{t,2}, \ldots, z'_{t,S}] \in \mathbb{R}^{S+2}. \tag{14}$$

where $z_{t,s}$ represents the estimated total remaining workload (in CPU cycles) in the queue of ES s at time t. It is updated as:

$$z'_{t,s} = \max\left\{ z'_{t-1,s} + \sum_{v \in \mathcal{V}_j} (x_{v,s} \cdot d_v \cdot \rho_v) - f_s \cdot \Delta t, 0 \right\}. \tag{15}$$

This state design enables the agent to perceive task requirements and system load holistically.

4.2.2 Action Space

The action for subtask v is to select a cooperative set of ESs. We define the action space as a subset of the ES set with a predefined size k:

$$\mathbf{A}_{v,t} \subset \mathcal{S}, \quad |\mathbf{A}_{v,t}| = k. \tag{16}$$

The parameter k balances parallelism and coordination overhead. In practice, k is selected as a small constant to balance collaboration gain and decision overhead. A larger k allows more ESs to participate in parallel execution and may better alleviate local bottlenecks, but it also introduces higher coordination cost, transmission overhead, and increased computational complexity in both Top-k selection and fraction allocation. Therefore, k should be determined according to the system scale, communication conditions, and latency requirements. In this work, we focus on the practically relevant setting where $k \ll S$. The network maps an input state to an advantage value $A(\mathbf{S}_{v,t}, s'; \theta)$ for every individual ES s'. The Q-value for a composite action s' (a set of k ESs) is defined as the sum of the advantage values of its members:

$$Q(\mathbf{S}_{v,t}, \mathbf{A}_{v,t}; \theta) = \sum_{s' \in \mathbf{A}_{v,t}} A(\mathbf{S}_{v,t}, s'; \theta). \tag{17}$$

The final action is derived by selecting the k ESs with the highest $A(\mathbf{S}_{v,t}, s'; \theta)$ from the DQN output.

4.2.3 Reward Function

The reward function is critical for aligning the DRL agent's goal with our objective of minimizing makespan and failure rate. We design a two-level reward:

1. Subtask Contribution Reward: If subtask v completes successfully, its reward is defined as the negative of its contribution to the prolongation of its parent task j's makespan. Let C_j^{before} and C_j^{after} be the estimated makespan of task j before and after scheduling subtask v, respectively. The contribution reward is:

$$\mathbf{r}_v^{\text{comp}} = -(C_j^{\text{after}} - C_j^{\text{before}}). \tag{18}$$

This encourages the agent to prioritize scheduling decisions that shorten the critical path of the DAG.

2. Task Failure Penalty: If task j fails (i.e., $C_j > D_j$), a significant constant penalty is applied to all its constituent subtasks to discourage actions leading to deadline violations:

$$\mathbf{r}^{\text{fail}} = -2 \cdot D_j. \tag{19}$$

The immediate reward for training the DQN agent for subtask v is:

$$\mathbf{r}_{v,t} = \mathbf{r}_v^{\text{comp}} + \mathbb{I}(C_j > D_j) \cdot \mathbf{r}^{\text{fail}}, \tag{20}$$

where $\mathbb{I}(\cdot)$ is the indicator function.

4.2.4 *Two-Stage Decision Making*

Stage 1: Top-k DQN for Multi-ES Selection. Given state $\mathbf{S}_{v,t}$, the DQN agent outputs advantage values and selects the action $\mathbf{A}_{v,t}$ as described.

Stage 2: Closed-Form Optimal Allocation of Fraction Subtasks. Given the selected ES set $\mathbf{A}_{v,t} = \{s_1, s_2, \cdots, s_k\}$, the original problem reduces to optimizing the allocation fractions $x_{v,s}$ for these k ESs. According to Eq. (11), the completion time of subtask v is determined by the maximum completion time among its fraction subtasks. To minimize this makespan, the optimal solution must equalize the completion times of all fraction subtasks assigned to the selected ESs.

Therefore, for any two selected ESs $s_1, s_2 \in \mathbf{A}_{v,t}$, with their corresponding fraction subtasks denoted as w_{s_1} and w_{s_2}, the following equality must hold:

$$T^{\text{comp}}_{w_{s_1}} + \alpha T^{\text{trans}}_{w_{s_1},s_1} = T^{\text{comp}}_{w_{s_2}} + \alpha T^{\text{trans}}_{w_{s_2},s_2}. \tag{21}$$

Expanding this using the definitions of T^{comp}_{w} and $T^{\text{trans}}_{w,s}$ leads to a linear equation in terms of x_{v,s_1} and x_{v,s_2}. By establishing $k-1$ such equations along with the workload conservation constraint $\sum_{s_i \in \mathbf{A}_{v,t}} x_{v,s_i} = 1$, we obtain a system of k linear equations.

This system can be solved efficiently via Gaussian elimination to obtain the optimal allocation x_{v,s_i}. For ESs not in $\mathbf{A}_{v,t}$, we set $x_{v,s'} = 0$.

4.3 The DRL-TCES Online Algorithm

Specifically, the source edge server of each subtask is responsible for initiating the collaborative decision process, including cooperative set selection and workload allocation. The execution of subtasks is then distributed across the selected edge servers, which process their assigned fraction subtasks independently based on local queues. This design enables scalable and fully distributed collaborative edge computing. The online operation and training procedure of DRL-TCES is summarized in Algorithm 1.

To further clarify the decision process, for each arriving subtask v at time slot t, the source edge server first constructs the state vector $S_{v,t}$ by combining task attributes and current queue workloads of all edge servers. Based on this state, the DQN outputs advantage values for all candidate edge servers, and the Top-k servers with the highest values are selected to form the cooperative set. Given this set, the workload allocation is then determined analytically by equalizing the completion times of fraction subtasks across the selected servers. This two-stage process enables efficient integration of learning-based decision making and optimization-based workload partitioning.

Algorithm 1 DRL-TCES Online Algorithm.

Input: Task arrivals DAGs $\{\mathcal{G}_j\}_{j\in\mathcal{J}_t}$, number of selected ESs k, target network update interval η, and discount factor γ.
Output: Online workload allocation decisions $\{x_{v,s}\}$.
Initialize online DQN θ_s, target DQN θ'_s, replay buffer $\mathcal{D}_s$ for each source ES s, and total step $L^{\text{step}} \leftarrow 0$;
for each time slot $t \in \mathcal{T}$ **do**
  for each arriving subtask $v \in \mathcal{V}_j$ **do**
    Agent at source ES s_v observes state $\mathbf{S}_{v,t}$;
    Select action $\mathbf{A}_{v,t}$ = Top-k ESs via ϵ-greedy policy based on $Q(\mathbf{S}_{v,t}, \cdot; \theta_{s_v})$;
    Solve the linear equation system from equalizing completion times for ESs in $\mathbf{A}_{v,t}$;
    Obtain optimal allocation fractions $\{x_{v,s_i}\}$ for $s_i \in \mathbf{A}_{v,t}$;
    Execute allocation, observe task completion/failure;
    Calculate reward $r_{v,t}$ using Eq. (20);
    Observe next state $\mathbf{S}_{v,t+1}$;
    Store transition $(\mathbf{S}_{v,t}, \mathbf{A}_{v,t}, r_{v,t}, \mathbf{S}_{v,t+1})$ in replay buffer $\mathcal{D}_{s_v}$;
  end for
  for each source ES s **do**
    Sample a random mini-batch $\{(\mathbf{S}, \mathbf{A}, \mathbf{r}, \mathbf{S}')\}$ from $\mathcal{D}_s$;
    for each transition $(\mathbf{S}, \mathbf{A}, \mathbf{r}, \mathbf{S}')$ in the mini-batch **do**
      Compute target $y = \mathbf{r} + \gamma \cdot \max_{\mathbf{A}'} Q(\mathbf{S}', \mathbf{A}'; \theta'_s)$;
      Update θ_s by minimizing loss: $L(\theta_s) = (y - Q(\mathbf{S}, \mathbf{A}; \theta_s))^2$;
    end for
    $L^{\text{step}} \leftarrow L^{\text{step}} + 1$;
  end for
  if $L^{\text{step}} \bmod \eta = 0$ **then**
    Update target network: $\theta'_s \leftarrow \theta_s$ for each source ES s;
  end if
end for

4.4 Complexity and Convergence Analysis

4.4.1 *Computational Complexity*

We analyze the computational overhead of the proposed DRL-TCES framework. Let $S = |\mathcal{S}|$ denote the total number of ESs, k refers to the size of the cooperative set, and N_t refers to the number of subtasks arriving in time slot t. For the DQN, let P represent the number of network parameters and

B the mini-batch size.

Per-Subtask Online Decision Cost: For each arriving subtask v, DRL-TCES executes a two-stage decision process: (i) selecting a cooperative set via DQN inference, and (ii) determining the optimal workload allocation via linear equation solving.

Stage 1: Top-k Cooperative Set Selection. A DQN inference requires a forward pass with complexity $O(P)$, producing advantage values for all S servers. Selecting the optimal set $\mathbf{A}_{v,t}$ corresponds to identifying the k ESs with the highest advantage values. Using a binary heap or linear-time selection algorithm, this Top-k operation incurs a cost of $O(S \log k)$. Therefore, the complexity of Stage 1 is $O(P + S \log k)$.

Stage 2: Optimal Fraction Allocation. Given the fixed set $\mathbf{A}_{v,t}$, the optimal allocation vector $\{x_{v,s}\}$ is obtained by equalizing the completion times across the selected ESs. This yields a system of k linear equations ($k-1$ balancing equations plus the workload conservation constraint $\sum_{s \in \mathbf{A}_{v,t}} x_{v,s} = 1$), which can be solved using Gaussian elimination with complexity $O(k^3)$. Since k is a small constant (e.g., $k = 3$) and $k \ll S$, this cost is negligible in practice.

Thus, the total decision complexity per subtask is

$$\mathcal{C}_{\text{decision}} = O\big(P + S \log k + k^3\big). \tag{22}$$

This decomposition avoids the combinatorial explosion of enumerating $\binom{S}{k}$ possible subsets, ensuring polynomial-time scalability with respect to the system size S.

4.4.2 *Training Complexity*

During online training (Algorithm 1), each source ES updates its DQN using a mini-batch of size B. For a single ES, computing the Temporal-Difference (TD) target requires a forward pass on the target network and a Top-k selection, resulting in a cost of $O(B(P + S \log k))$, while the backpropagation step scales as $O(BP)$. Hence, the per-ES training cost per update is $O(B(P + S \log k))$.

Aggregating across all S ESs, the total training complexity per time slot is

$$\mathcal{C}_{\text{train}}^{\text{total}} = O\big(S \cdot B(P + S \log k)\big), \tag{23}$$

which is fully parallelizable across ESs.

4.4.3 *Convergence Analysis*

The convergence behavior of DRL-TCES is motivated by the theoretical foundations of Q-learning. For finite MDPs, tabular Q-learning is known to converge to the optimal action-value function under sufficient state–action visitation and diminishing learning rates [34].

In DRL-TCES, the scheduling problem is formulated as an MDP, while DQN are employed to handle large and continuous state spaces. Although strict convergence guarantees no longer hold under function approximation, standard stabilization mechanisms-including experience replay, target networks, and ϵ-greedy exploration-are adopted to alleviate non-stationarity and training instability.

Existing analyses of Deep Q-learning, such as [35], show that under bounded rewards and sufficient approximation capacity, the learning error can be controlled and the learned Q-values converge to a neighborhood of the optimal solution. In DRL-TCES, the Top-k edge selection strategy serves as a value-based action abstraction that improves decision efficiency without altering the core learning dynamics of DQN.

Therefore, while global optimality cannot be theoretically guaranteed, DRL-TCES is expected to converge to a stable and near-optimal policy in practice.

5 Result and Discussion

5.1 Experimental Settings

We consider a collaborative edge computing system composed of multiple heterogeneous ESs. The system operates in discrete time slots, each with a duration of $\Delta t = 0.1$ s, and the total simulation horizon consists of $T = 120$ time slots. Unless otherwise specified, α is set to 0.1 following [32].

To capture heterogeneity among ESs, the computing capacities of ESs are configured in an arithmetic progression. The transmission rate between any two ESs is randomly selected within the range of 400–500 Mbits/s.

Tasks arrive at the system following a Bernoulli process with arrival probability $p_{\text{job}} = 0.4$. Each task is represented as a DAG consisting of multiple dependent subtasks, where the number of nodes in each DAG is randomly generated within the interval $[3, 5]$. The data size of each subtask is uniformly sampled around $d_{j,v} = 50$ Mbits.

During collaborative offloading, each subtask is allowed to be executed by at most k ESs. In this study, the Top-k parameter is set to $k = 3$. This choice reflects a trade-off between cooperation effectiveness and online decision complexity. A small value of k is sufficient to exploit multi-edge collaboration while keeping the scheduling overhead negligible in practice.

All agents share the same network architecture and training logic, while making decisions independently based on local observations. Experience replay is adopted during training, with a replay buffer size of 1024 and a mini-batch size of 32. The target network is synchronized every $\eta = 200$ updates. The learning rate is set to 0.001, and the discount factor is $\gamma = 0.9$. An ϵ-greedy policy is used for action selection,

where ϵ increases linearly from 0 to 0.99 with a step size of 0.001. The hyperparameters used in DRL-TCES are selected based on a combination of commonly adopted settings in prior DQN-based studies and empirical tuning for our specific multi-edge scheduling scenario [25]. We initialize these parameters according to standard practices in the literature and then perform limited adjustments to ensure stable training convergence under dynamic workloads [36]. We further observe that the performance of DRL-TCES is relatively robust to moderate variations of key hyperparameters, as long as they remain within reasonable ranges, indicating the practicality of the proposed method.

5.2 Performance Metrics

We evaluate system performance from two key perspectives: average Makespan and task success rate.

5.2.1 *Average Makespan*

The average makespan is defined as the arithmetic mean of the system makespan across all time slots, which serves to approximate the theoretical long-term time-averaged objective. This metric serves as a discrete-time approximation of the long-term time-averaged objective defined in 24.

$$\bar{C}^{\max} = \frac{1}{T} \sum_{t \in \mathcal{T}} C_t^{\max}. \tag{24}$$

5.2.2 *Task Success Rate*

For task j, if the makespan exceeds the maximum allowable delay D_j, the task is considered failed. The task success rate is defined as

$$\mathrm{SR} = 1 - \frac{\sum_{t \in \mathcal{T}} \sum_{j \in \mathcal{J}_t} \mathbb{I}(C_j > D_j)}{\sum_{t \in \mathcal{T}} |\mathcal{J}_t|}, \tag{25}$$

where $|\mathcal{J}_t|$ denotes the number of task in set $\mathcal{J}_t$.

5.3 Baseline Algorithms

To evaluate the effectiveness of the proposed approach, we compare it with the following baseline algorithms:

Random Offloading (RAND) [37]: A baseline strategy that randomly selects the execution location without considering system states.

Least-Queue-First Offloading (LQF) [38]: A greedy heuristic that offloads tasks to the ES with the smallest current queue backlog.

DRL-LAO [39]: A deep reinforcement learning–based load-aware offloading algorithm that incorporates historical edge load dynamics into the decision-making process.

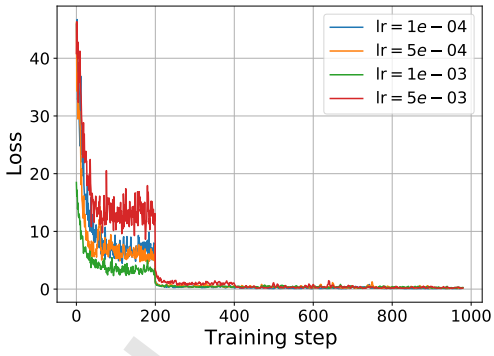

Fig. 2 Training loss under different learning rates

5.4 Experimental Results and Analysis

5.4.1 *Convergence Behavior*

Fig. 2 illustrates the training loss of DRL-TCES under different learning rates. It can be observed that, for all tested learning rates, the loss decreases rapidly during the initial training phase and gradually converges to a stable low level as training proceeds, indicating effective policy learning and stabilization. The rapid decline in the early stage suggests that the proposed state representation and reward design enable the agent to quickly capture dominant scheduling patterns and adapt to the dynamic system environment. As training continues, the progressively smoother loss trajectory reflects the gradual refinement of action-value estimation and the stabilization of decision policies through iterative interaction with the environment.

In particular, moderate learning rates (e.g., $lr = 10^{-3}$) achieve faster convergence with noticeably reduced oscillations, resulting in a more stable training process. This behavior indicates that an appropriate learning rate can effectively balance gradient update magnitude and stochastic exploration, thereby facilitating efficient propagation of temporal-difference errors without introducing excessive variance. In contrast, overly large learning rates may lead to abrupt parameter updates, amplifying the noise in gradient estimation and causing pronounced oscillations during the early training stage. The observed convergence characteristics are consistent with the expected learning dynamics of DQN-based methods equipped with experience replay and target networks, which jointly mitigate training instability caused by non-stationary data distributions and correlated samples.

5.4.2 *Impact of the Number of ESs*

Fig. 3(a) and Fig. 4(a) illustrate the impact of the number of ESs on the average makespan and task success rate, respectively. As the number of ESs increases from 4 to 8, the proposed DRL-TCES achieves a substantial reduction

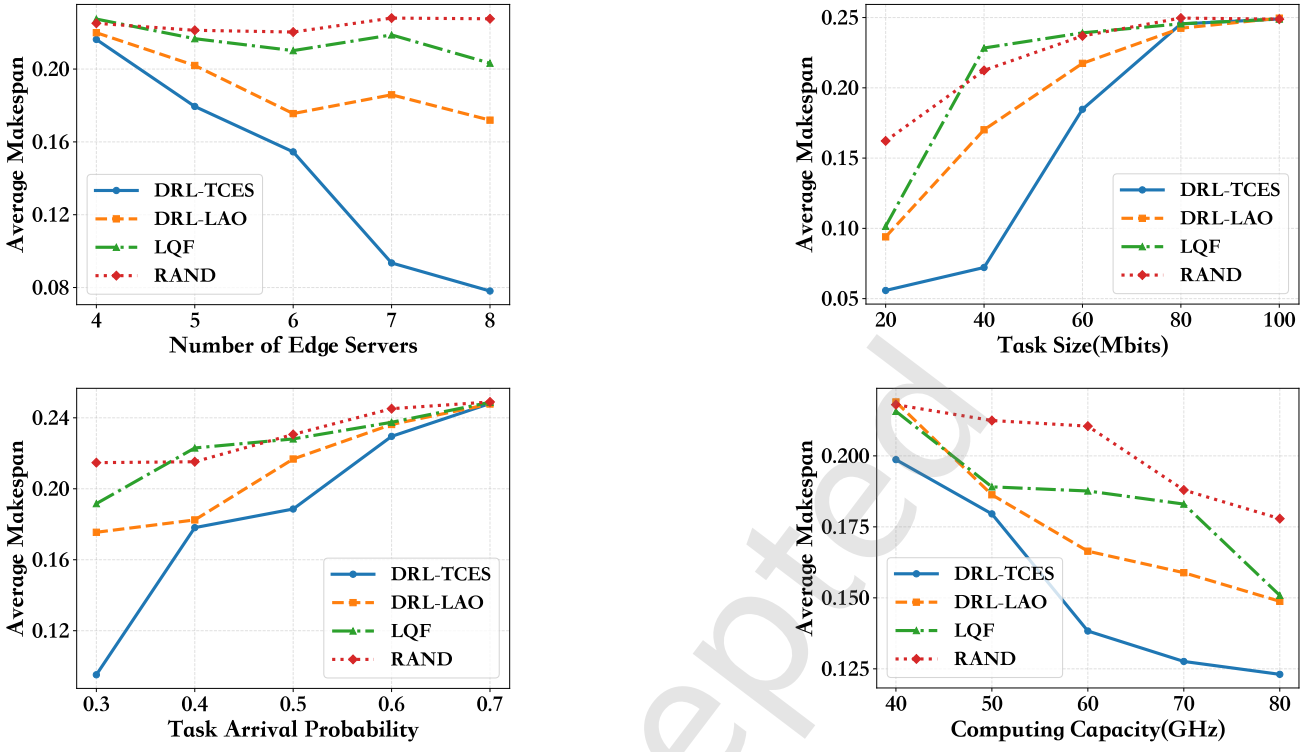

Fig. 3 Average makespan under different system parameters.

in average makespan, decreasing from 0.216 s to 0.078 s, which corresponds to a reduction of approximately 63.9%. This pronounced improvement indicates that DRL-TCES can effectively exploit the increased spatial diversity of edge resources and the expanded opportunity for parallel execution enabled by a larger ES pool. With more candidate ESs participating in cooperative execution, the scheduler gains higher flexibility in selecting complementary computing resources, thereby enabling finer-grained workload partitioning and more efficient exploitation of parallelism. In particular, the cooperative selection mechanism allows tasks to be distributed across multiple ESs in a manner that alleviates localized overload, reduces queueing pressure at individual servers, and shortens the critical execution path induced by subtask dependencies.

In comparison, DRL-LAO only reduces the completion time by 21.8% over the same range, suggesting limited scalability when the number of candidate ESs grows. This behavior implies that DRL-LAO is less effective at leveraging additional ESs once the action space expands, as its decision-making strategy cannot fully capture the combinatorial benefit brought by multi-edge cooperation. The heuristic LQF and RAND baselines exhibit only marginal improvements, indicating that simple queue-based or random strategies are insufficient to systematically utilize newly available edge resources. As the system scale increases, these heuristics fail to coordinate parallel execution effectively, leading to persistent workload imbalance and suboptimal utilization of the expanded ES pool.

From the perspective of task success rate, DRL-TCES demonstrates strong scalability with respect to the number of ESs. When the number of ESs reaches 6, DRL-TCES already achieves a success rate exceeding 98.5%, and the performance remains stable as more ESs are added, reflecting the rapid mitigation of dominant system bottlenecks. This saturation behavior suggests that once a sufficient number of cooperative ESs becomes available, the system transitions from a resource-constrained regime to a coordination-limited regime, after which additional resources yield diminishing returns. By comparison, DRL-LAO achieves only a 74.2% success rate at 8 ESs, while LQF and RAND remain below 45%, highlighting their limited ability to construct effective collaboration sets. These results indicate that the Top-k DQN-based ES selection mechanism in DRL-TCES can efficiently identify high-quality collaboration sets from a rapidly expanding candidate space, thereby improving both latency performance and execution reliability under increasing system scale.

5.4.3 Impact of Task Size

Fig. 3(b) and Fig. 4(b) show system performance under different task sizes. As the task size increases from 20 Mbits

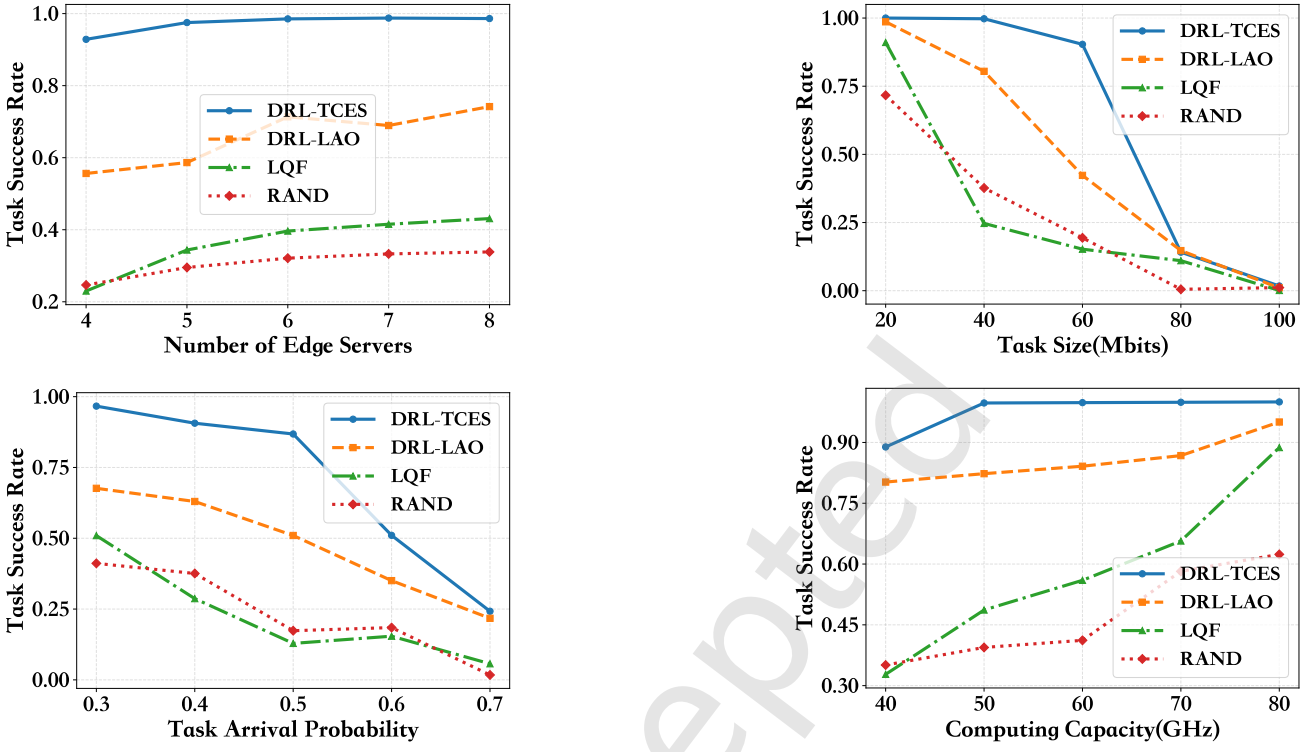

Fig. 4 Task success rate under different system parameters.

to 100 Mbits, all algorithms experience an increase in average makespan due to higher transmission and computation overhead, which is consistent with expected system behavior in edge computing environments. Larger task sizes incur longer uplink transmission delays and higher processing demands, placing additional stress on both communication and computation resources, while also intensifying queueing effects at edge servers under concurrent task arrivals. As a result, fraction subtasks are more likely to accumulate in local queues, leading to prolonged waiting times and amplified delays along dependency paths. Nevertheless, DRL-TCES consistently achieves lower completion time than all baselines across the entire task size range, reflecting its robust adaptability under varying workload intensities and its ability to effectively coordinate multi-edge resources.

In particular, at a task size of 60 Mbits, DRL-TCES achieves an average completion time of 0.185 s, which is 14.9% lower than that of DRL-LAO. This performance gap highlights the advantage of cooperative execution and adaptive ES selection in mitigating increased workload pressure, especially when both transmission and computation resources become heavily loaded. By distributing computation across multiple ESs and dynamically balancing workload allocation, DRL-TCES reduces the processing burden on individual servers and more fully exploits inter-server parallelism, thereby alleviating localized congestion. When the task size further increases to 80 Mbits, the completion time of all algorithms begins to converge, indicating that the system approaches its capacity limit and that both computation and transmission resources become progressively saturated. Under such extreme workload conditions, marginal gains from additional optimization diminish, and the system enters a regime dominated by fundamental resource constraints.

Despite this convergence in latency, DRL-TCES still exhibits a clear advantage in task success rate, reflecting its superior capability in deadline satisfaction under heavy data demands. At 80 Mbits, DRL-TCES maintains a success rate of 14.1%, whereas DRL-LAO and LQF drop to 14.7% and 11.0%, respectively, and RAND nearly fails with only 0.5% success. This reliability improvement stems from the ability of DRL-TCES to exploit cooperative parallelism and proactively avoid excessive load concentration on single ESs, thereby reducing the probability of severe queue buildup and deadline violations. Such behavior indicates that even when the system operates near saturation, DRL-TCES can sustain more stable execution dynamics and higher service reliability by leveraging coordinated multi-edge execution.

5.4.4 Impact of Task Arrival Probability

Fig. 3(c) and Fig. 4(c) evaluate the impact of task arrival probability, which serves as a key indicator of overall system

workload intensity. A higher arrival probability implies that more tasks are injected into the system within a given time window, thereby increasing competition for limited communication and computation resources at the edge and intensifying contention among concurrently arriving workflows. As the arrival probability increases from 0.3 to 0.7, the system becomes increasingly congested, leading to longer queueing delays and lower task success rates for all evaluated algorithms. This degradation trend reflects the growing contention for shared edge resources under heavy traffic conditions, where task queues accumulate rapidly and the waiting time of fraction subtasks grows sharply. Under such circumstances, even minor inefficiencies in scheduling and resource allocation can be amplified along dependency chains, causing delays to propagate through successive subtasks and ultimately extending the critical execution path of DAG-based applications.

Despite this challenging setting, DRL-TCES consistently outperforms the baselines in both latency and reliability. At an arrival probability of 0.7, DRL-TCES achieves a task success rate of 24.2%, which is 11.5% higher than DRL-LAO, 17.4% higher than LQF, and 22.5% higher than RAND. In addition to higher reliability, DRL-TCES maintains a lower average task completion time compared to the baselines across the entire arrival probability range, reflecting its enhanced capability in alleviating queue congestion under heavy workloads. This performance advantage can be attributed to the ability of DRL-TCES to dynamically distribute incoming workloads among multiple ESs and continuously adapt its cooperation strategy in response to fluctuating traffic intensity. By proactively preventing excessive load concentration on individual ESs and enabling more balanced resource utilization across the collaborative edge set, DRL-TCES effectively reduces queue buildup and limits the propagation of delays along dependency paths, thereby sustaining higher service reliability and more stable execution performance under high arrival rates.

5.4.5 Impact of Computing Capacity

Fig. 3(d) and Fig. 4(d) illustrate the effect of heterogeneous computing capacity on system performance. Increasing computing capacity directly enhances the processing capability of edge servers, thereby reducing execution latency for computation-intensive tasks and shortening the service time of queued workloads. As the computing capacity increases from 40 GHz to 80 GHz, the average completion time of DRL-TCES decreases from 0.199 s to 0.123 s, corresponding to a reduction of 38.1%. This reduction is larger than that achieved by DRL-LAO (32.1%) and LQF (30.0%), indicating that DRL-TCES can more effectively translate enhanced computing capacity into end-to-end latency improvement. The amplified performance gain further suggests that DRL-TCES is capable of fully leveraging newly available computational resources to relieve system bottlenecks, especially under conditions where multiple dependent subtasks compete for limited processing capacity.

The superior latency reduction suggests that DRL-TCES is better at adapting its ES selection and workload distribution strategy to heterogeneous processing capabilities. By preferentially assigning tasks to ESs with higher available capacity and coordinating execution across multiple servers, DRL-TCES effectively balances processing loads and alleviates localized congestion. This cooperative adaptation mechanism mitigates the formation of processing bottlenecks that typically arise in imbalanced resource configurations, thereby reducing queueing delays and stabilizing task execution. In contrast, the baselines benefit less from increased computing capacity, implying limited ability to dynamically exploit heterogeneity when resource distributions change, which results in suboptimal utilization of high-capacity ESs and persistent performance gaps.

In terms of task success rate, DRL-TCES achieves a success rate close to 100% at 80 GHz, whereas DRL-LAO achieves 95.0%, and LQF and RAND remain below 90% and 63%, respectively. This improvement in reliability indicates that higher computing capacity not only shortens execution time but also significantly reduces the likelihood of deadline violations by enabling faster completion of critical subtasks along dependency paths. By jointly optimizing ES selection and workload allocation, DRL-TCES effectively mitigates performance degradation caused by imbalance across edge servers and sustains stable task execution under diverse resource configurations, even in the presence of pronounced heterogeneity.

5.5 Discussion

The experimental results reveal several important insights into the intrinsic characteristics of dependency-aware multi-edge cooperative scheduling. From the system perspective, the consistent makespan reduction observed under increasing system scale confirms that intelligently orchestrated multi-edge collaboration can effectively exploit spatial resource diversity, thereby alleviating localized congestion and shortening critical execution paths in DAG-structured workloads. This highlights the fundamental advantage of cooperation-aware scheduling over isolated or greedy decision strategies in dynamic MEC environments.

From the algorithmic perspective, the superior robustness of DRL-TCES under heavy workloads demonstrates the ef-

fectiveness of jointly optimizing cooperative edge selection and workload partitioning. The Top-k sampling mechanism enables scalable exploration of the exponentially expanding action space, while the analytical workload allocation strategy equalizes subtask completion times across selected ESs, ensuring that parallel execution gains are fully translated into end-to-end latency reduction. This tightly coupled learning–optimization framework allows DRL-TCES to adapt rapidly to dynamic congestion patterns and heterogeneous system states.

From the perspective of resource heterogeneity, the significant performance gains achieved under diverse computing capacities indicate that DRL-TCES possesses strong adaptability to practical MEC deployments, where heterogeneous hardware platforms and fluctuating resource availability are ubiquitous. By dynamically prioritizing high-capacity ESs and maintaining balanced workload distribution, the framework effectively mitigates performance degradation caused by resource imbalance and improves overall system resilience.

From a broader perspective, the proposed DRL-TCES framework provides a generalizable paradigm for dependency-aware cooperative scheduling beyond conventional MEC systems. The integration of learning-based cooperative selection and analytical workload balancing can be extended to more complex scenarios, such as bandwidth-constrained wireless networks, mobility-driven edge environments, and partially observable systems. Incorporating predictive mechanisms, such as workload forecasting and mobility modeling, represents a promising direction for further enhancing decision stability and long-term performance.

6 Conclusion

This paper investigated the online scheduling of dependency-aware DAG tasks in cooperative multi-edge computing systems with stochastic arrivals and heterogeneous resources. We formulated a long-term makespan minimization problem under deadline constraints and proved its NP-hardness, revealing the inherent complexity of dependency-aware multi-edge scheduling. To address this challenge, we proposed DRL-TCES, which combines a Top-k DQN for cooperative edge selection with an analytical workload partitioning scheme that equalizes subtask completion times across selected edge servers. This design explicitly targets critical-path reduction in DAG execution while remaining adaptive to dynamic queue backlogs and resource fluctuations. Extensive results demonstrate that DRL-TCES consistently outperforms heuristic and learning-based baselines across a wide range of system scales and workload intensities, achieving up to 63.9% reduction in average makespan and a task success rate of up to 98.5%. These results highlight the effectiveness of jointly optimizing dependency-aware scheduling and multi-edge cooperation in dynamic MEC environments. An important direction is to extend the proposed framework to communication-constrained or partially observable settings, where inter-edge bandwidth contention and delayed state information further complicate dependency-aware cooperative scheduling.

Acknowledgment

The above work was supported in part by the Guangxi Key Research & Development Program (FN2504240036,2025FN96441087), Guangdong S&T Programme (No. 2025B0101120006), the Natural Science Foundation of Guangdong Province (2024A1515011323), the Supplemental Funds for Major Scientific Research Projects of Beijing Normal University, Zhuhai (ZHPT2023002), the Fundamental Research Funds for the Central Universities, Guangdong Province Educational Science Planning Research Project under 2025JKZG069, Higher Education Research Topics of Guangdong Association of Higher Education in the 14th Five-Year Plan under 24GYB207.

Declaration of competing interest

The authors have no competing interests to declare that are relevant to the content of this article.

Reference

[1] X. Wang and J. Ma, Cloud-network-end collaborative security for wireless networks: Architecture, mechanisms, and applications, *Tsinghua Science and Technology*, vol. 30, no. 1, pp. 1–17, 2025.

[2] H. Huang, T. Meng, and W. Jia, Joint optimization of prompt security and system performance in edge-cloud llm systems, in *IEEE INFOCOM 2025-IEEE Conference on Computer Communications*. IEEE, 2025, pp. 1–10.

[3] J. Tang, M. Xu, S. Fu, and K. Huang, A scheduling optimization technique based on reuse in spark to defend against apt attack, *Tsinghua Science and Technology*, vol. 23, no. 5, pp. 550–560, 2018.

[4] Y. Liang, G. Li, G. Zhang, J. Guo, Q. Liu, J. Zheng, and T. Wang, Latency reduction in immersive systems through request scheduling with digital twin networks in collaborative edge computing, *ACM Transactions on Sensor Networks*, 2024.

[5] H. Meng, J. Ding, H. Wang, Z. Zhang, X. Yao, and H. Ning, Blockchain enabled metaverse: Development and applications, *Tsinghua Science and Technology*, vol. 30, no. 4, pp. 1552–1582, 2025.

[6] Y. Liang, M. Yin, W. Wang, Q. Liu, L. Wang, X. Zheng, and T. Wang, Collaborative edge server placement for maximizing qos with distributed data cleaning, *IEEE Transactions on Services Computing*, pp. 1–15, 2025.

[7] L. Zeng, Q. Liu, S. Shen, and X. Liu, Improved double deep q network-based task scheduling algorithm in edge computing for makespan optimization, *Tsinghua Science and Technology*, vol. 29, no. 3, pp. 806–817, 2023.

[8] H. Zou, J. Guo, Y. Li, W. Fan, W. Su, C. Xu, Y. Liang, T. Wang, and J. Cao, Logical correction enabled collaborative person detection inference in edge networks, *IEEE Transactions on Mobile Computing*, vol. 25, no. 4, pp. 4747–4761, 2026.

[9] G. Zhang, B. Zhang, S. Peng, and C. Li, Dependency-aware joint task offloading and resource allocation in heterogeneous mobile edge computing, *IEEE Transactions on Wireless Communications*, vol. 23, no. 12, pp. 19 444–19 458, 2024.

[10] H. Du, M. Liu, N. Liu, D. Li, W. Li, and L. Xu, Scheduling of low-latency medical services in healthcare cloud with deep reinforcement learning, *Tsinghua Science and Technology*, vol. 30, no. 1, pp. 100–111, 2025.

[11] L. Liu, H. Tan, S. H.-C. Jiang, Z. Han, X.-Y. Li, and H. Huang, Dependent task placement and scheduling with function configuration in edge computing, in *Proceedings of the International Symposium on Quality of Service*, 2019, pp. 1–10.

[12] Y. Jiang, K. Di, R. Qian, X. Wu, F. Chen, P. Li, X. Fu, and Y. Jiang, Optimizing risk-aware task migration algorithm among multiplex uav groups through hybrid attention multi-agent reinforcement learning, *Tsinghua Science and Technology*, vol. 30, no. 1, pp. 318–330, 2025.

[13] X. Ma, Z. Wang, and J. Li, Uniform exploration of dynamic attributed graph via embedded community comparison, *Tsinghua Science and Technology*, 2025.

[14] X. Chen, S. Hu, C. Yu, Z. Chen, and G. Min, Real-time offloading for dependent and parallel tasks in cloud-edge environments using deep reinforcement learning, *IEEE Transactions on Parallel and Distributed Systems*, vol. 35, no. 3, pp. 391–404, 2024.

[15] Y. Huang and R. Zhou, An online framework for joint uav trajectory planning and intelligent dependent task offloading, in *2023 20th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 2023, pp. 258–266.

[16] Y. Liang, M. Yin, Y. Zhang, W. Wang, W. Jia, and T. Wang, Grouping reduces energy cost in directionally rechargeable wireless vehicular and sensor networks, *IEEE Transactions on Vehicular Technology*, vol. 72, no. 8, pp. 10 840–10 851, 2023.

[17] F. Huang, W. Wang, Q. Liu, W. Fan, W. Su, W. Jia, T. Wang, and J. Cao, Edgemanager: Online adaptive resource management for hierarchical dnn inference in collaborative edge environments, *IEEE Transactions on Mobile Computing*, vol. 25, no. 4, pp. 5571–5589, 2026.

[18] Z. Peng, N. Zou, J. Zeng, B. Li, S. Chen, T. Wang, and W. Jia, Dynamic grouping and aggregation weight optimization for hierarchical federated learning with quantization, *IEEE Transactions on Networking*, vol. 34, pp. 3824–3839, 2026.

[19] I. Rahmati, H. Shah-Mansouri, and A. Movaghar, Qeco: A qoe-oriented computation offloading algorithm based on deep reinforcement learning for mobile edge computing, *IEEE Transactions on Network Science and Engineering*, 2025.

[20] X. Dai, Z. Xiao, H. Jiang, M. Lei, G. Min, J. Liu, and S. Dustdar, Offloading dependent tasks in edge computing with unknown system-side information, *IEEE Transactions on Services Computing*, vol. 16, no. 6, pp. 4345–4359, 2023.

[21] X. Chen, J. Cao, Y. Sahni, S. Jiang, and Z. Liang, Dynamic task offloading in edge computing based on dependency-aware reinforcement learning, *IEEE Transactions on Cloud Computing*, vol. 12, no. 2, pp. 594–608, 2024.

[22] X. Chen, S. Hu, C. Yu, Z. Chen, and G. Min, Real-time offloading for dependent and parallel tasks in cloud-edge environments using deep reinforcement learning, *IEEE Transactions on Parallel and Distributed Systems*, vol. 35, no. 3, pp. 391–404, 2024.

[23] Z. Tong, J. Deng, J. Mei, Y. Zhang, and K. Li, Multi-objective dag task offloading in mec environment based on federated dqn with automated hyperparameter optimization, *IEEE Transactions on Services Computing*, vol. 17, no. 6, pp. 3999–4012, 2024.

[24] J. Zhang, J. Chen, X. Bao, C. Liu, P. Yuan, X. Zhang, and S. Wang, Dependent task offloading mechanism for cloud–edge-device collaboration, *Journal of Network and Computer Applications*, vol. 216, p. 103656, 2023.

[25] C. Xu, J. Guo, Y. Li, H. Zou, W. Jia, and T. Wang, Dynamic parallel multi-server selection and allocation in collaborative edge computing, *IEEE Transactions on Mobile Computing*, vol. 23, no. 11, pp. 10 523–10 537, 2024.

[26] T. Wang, Y. Liang, W. Jia, M. Arif, A. Liu, and M. Xie, Coupling resource management based on fog computing in smart city systems, *Journal of Network and Computer Applications*, vol. 135, pp. 11–19, 2019.

[27] H. Huang, T. Meng, J. Guo, X. Wei, and W. Jia, Seceg: A secure and efficient strategy against ddos attacks in mobile edge computing, *ACM Transactions on Sensor Networks*, vol. 20, no. 3, pp. 1–21, 2024.

[28] L. Zhao, E. Zhang, S. Wan, A. Hawbani, A. Y. Al-Dubai, G. Min, and A. Y. Zomaya, Meson: A mobility-aware dependent task offloading scheme for urban vehicular edge computing, *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 4259–4272, 2023.

[29] X. Chen, J. Cao, Y. Sahni, M. Zhang, Z. Liang, and L. Yang, Mobility-aware dependent task offloading in edge computing: a digital twin-assisted reinforcement learning approach, *IEEE Transactions on Mobile Computing*, vol. 24, no. 4, pp. 2979–2994, 2025.

[30] J. Wang, J. Hu, G. Min *et al.*, Dependent task offloading for edge computing based on deep reinforcement learning, *IEEE*

Transactions on Computers, vol. 71, no. 10, pp. 2449–2461, 2021.

[31] G. Zhao, H. Xu, Y. Zhao *et al.*, Offloading dependent tasks in mobile edge computing with service caching, in *Proceedings of IEEE INFOCOM 2020*, 2020, pp. 1997–2006.

[32] S. Liu, Y. Yu, X. Lian, Y. Feng, C. She, P. L. Yeoh, L. Guo, B. Vucetic, and Y. Li, Dependent task scheduling and offloading for minimizing deadline violation ratio in mobile edge computing networks, *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 2, pp. 538–554, 2023.

[33] Y. Zakharova and M. Sakhno, Complexity and heuristic algorithms for speed scaling scheduling of parallel jobs with energy constraint, *Journal of Computational and Applied Mathematics*, vol. 457, p. 116254, 2025.

[34] Q. He, L. Zhang, H. Fang, X. Wang, L. Ma, K. Yu, and J. Zhang, Multistage competitive opinion maximization with q-learning-based method in social networks, *IEEE transactions on neural networks and learning systems*, vol. 36, no. 4, pp. 7158–7168, 2024.

[35] T. Bozkus and U. Mitra, Multi-timescale ensemble q-learning for markov decision process policy optimization, *IEEE Transactions on Signal Processing*, vol. 72, pp. 1427–1442, 2024.

[36] C. Xu, J. Guo, Y. Liang, H. Zou, J. Zeng, H. Dai, W. Jia, J. Cao, and T. Wang, Enhancing qoe in collaborative edge systems with feedback diffusion generative scheduling, *IEEE Transactions on Mobile Computing*, vol. 24, no. 12, pp. 12 983–12 998, 2025.

[37] P. F. Moshiri, M. Simsek, and B. Kantarci, Joint optimization of completion ratio and latency of offloaded tasks with multiple priority levels in 5g edge, *IEEE Transactions on Network and Service Management*, 2025.

[38] C. Xu, J. Guo, J. Zeng, Y. Li, J. Cao, and T. Wang, Incorporating startup delay into collaborative edge computing for superior task efficiency, in *2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*. IEEE, 2024, pp. 1–10.

[39] Q. He, Q. Li, C. Zhang, X. Wang, Y. Bi, L. Zhao, A. Hawbani, and K. Yu, Task optimization allocation in vehicle based edge computing systems with deep reinforcement learning, *IEEE Transactions on Computers*, 2025.

Author biography

Youke Wu received the B.Sc. degree in information management and information system from Hunan University, Changsha, China, in 2005, and the M.Sc. and Ph.D. degrees in economics from Shenzhen University, Shenzhen, China, in 2004 and 2011, respectively. Currently, she is an Associate Professor with Wuyi University, Jiangmen, China. Her research interests include social networks, mobile computing, and economics.

Kaiyuan Zheng is currently pursuing the B.S. degree at Beijing Normal University, Zhuhai, China. His research interests include artificial intelligence and edge computing.

Yuzhu Liang received his M.S. degree from National Huaqiao University of China in 2020 and his Ph.D. degree from Beijing Normal University of China in 2025. He is currently a postdoctoral researcher at Beijing Normal University. His research interests include edge computing, mobile computing, and AI.

Haiyang Huang received his M.S. degree from the College of Computer Science and Technology, Huaqiao University, China, in 2020, and his Ph.D. degree in the School of Artificial Intelligence, Beijing Normal University, China. Currently, he is a lecturer in the School of Artificial Intelligence and Computer Science, Nantong University, China. His research interests include mobile edge computing security, resource scheduling, and edge AI.

Tian Wang received his BSc and MSc degrees in Computer Science from the Central South University in 2004 and 2007, respectively. He received his PhD degree in City University of Hong Kong in in Computer Science in 2011. Currently, he is a professor in the Institute of Artificial Intelligence and Future Networks, Beijing Normal University. His research interests include internet of things, edge computing and mobile computing. He has 27 patents and has published more than 300 papers in high-level journals and conferences. He has more than 18000 citations, according to Google Scholar. His H-index is 78.