Task Scheduling of Joint Node Selection and Path Planning in Computing Power Network

Chengyong Yang , Xuanlong Ruan , Jianlin Cheng *

Posted Date: 20 May 2026

doi: 10.20944/preprints202605.1329.v1

Keywords: computing power network; task scheduling; node selection; path planning; deep reinforcement learning

 doi:10.20944/preprints202605.1329.v1

Article

Task Scheduling of Joint Node Selection and Path Planning in Computing Power Network

Chengyong Yang, Xuanlong Ruan and Jianlin Cheng *

College of Computer Science and Engineering, Guilin University of Technology, Guilin 541006, China
* Correspondence: chengjianlin@glut.edu.cn

Abstract

Cloud computing and mobile edge computing address the growing demand for computing power driven by the rise in data-intensive applications, but they are prone to creating computing silos, resulting in unbalanced resource utilization. To address this issue, the Computing Power Network (CPN) has been introduced to enable the centralized management and scheduling of resources across the entire network. However, task scheduling in the CPN requires joint selection of computation nodes and routing paths, which greatly increases the complexity of scheduling problem. In existing studies, heuristic methods are difficult to satisfy real-time requirements, whereas deep reinforcement learning methods ignore the collaborative optimization of network resources, making them difficult to adapt to complex CPN scenarios. To this end, we propose a task scheduling method for the CPN, called TS-DQNF. First, the method uses the Deep Q-Network (DQN) to determine the computation node for computation task. Then, it introduces a dynamic congestion-aware mechanism to determine the shortest routing path. Finally, it gradually obtains the optimal task scheduling scheme through multiple rounds of alternating iterations. Simulation results show that the TS-DQNF achieves good performance and good convergence performance under different scenarios and scales.

Keywords: computing power network; task scheduling; node selection; path planning; deep reinforcement learning

1. Introduction

In recent years, the rapid development of sixth-generation mobile communication (6G) has driven a surge in data-intensive applications, such as virtual reality [1], autonomous driving [2], face recognition [3], and artificial intelligence-generated content (AIGC) [4], leading to explosive growth in global data volumes. These data-intensive applications impose unprecedented requirements on system computing capability and network transmission capability. Although traditional cloud computing [5] provides massive computing resources, it often incurs unacceptable transmission delay due to the long physical distance. Mobile edge computing (MEC) [6] effectively alleviates this problem by moving computing capability closer to the network edge. However, due to the diversity of user devices (UDs) and the bottleneck of Moore?s law [7], the computing resources of edge nodes are relatively limited and sparsely distributed. This tends to form computing power silos, making it difficult to meet large-scale and complex computation demands.

To break computing power silos and achieve coordinated scheduling of resources across the entire network, Computing Power Network (CPN) [8–10] has emerged. CPN is a novel networking technology that aims to optimize the distribution of computing and networking resources among computation nodes through the network control layer. It manages distributed computing and networking resources via ubiquitous network connections, promotes the deep integration of computing power and networking, and effectively addresses the challenges of resource isolation and data gravity. As a bridge connecting UDs and computation nodes, CPN needs to coordinate networking resources

 doi:10.20944/preprints202605.1329.v1

and computing resources. Therefore, efficient task scheduling in the CPN is indispensable for fully utilizing computing and networking resources.

In the context of CPN, the task scheduling problem becomes more complex. On the one hand, multiple computation nodes that satisfy the computing requirements of a task may exist simultaneously in the network. Therefore, how to schedule the computation tasks generated by UDs to appropriate computation nodes for execution, thereby optimizing computing resources, is one of the challenges of task scheduling in the CPN. On the other hand, multiple feasible networking paths may exist when computation tasks are transmitted from UDs to computation nodes. Therefore, how to plan a path with the lowest transmission cost for each task, thereby optimizing networking resources, is another challenge of task scheduling in the CPN. These two challenges constitute the joint optimization problem that task scheduling in the CPN needs to solve, which further increases the complexity of task scheduling in the CPN. In recent years, many studies on task scheduling have mainly focused on cloud computing or MEC scenarios. Some works have adopted heuristic methods to optimize task scheduling [11–13]. However, heuristic methods require a large amount of time to search for the optimal solution, making it difficult to satisfy the real-time requirements of computation tasks. On the other hand, some works have adopted deep reinforcement learning (DRL) methods to optimize scheduling decisions and have achieved significant results [14–16]. However, these DRL-based scheduling methods generally focus on optimization on the computing side and ignore the optimization of networking resources during data transmission, making them difficult to adapt to complex CPN scenarios.

In summary, we propose a task scheduling method for the CPN, called TS-DQNF. This method integrates the Deep Q-Network (DQN) algorithm with the dynamic Floyd algorithm and aims to jointly optimize computation node selection and routing path planning for tasks, thereby improving the overall task success rate of the system and reducing the average processing delay of tasks. The main contributions of this paper are summarized as follows:

- We consider a CPN task scheduling system that integrates computing and networking resources. Based on this system, we describe the specific processes of task communication and computation, and formulate an optimization problem with the objectives of maximizing the task success rate and minimizing the average processing delay.
- To solve the joint optimization problem in the CPN, we propose the TS-DQNF method. Specifically, the TS-DQNF first uses the DQN algorithm to determine the target computation node for each task. Then, it introduces a congestion-aware mechanism to improve the Floyd algorithm and compute the shortest routing path for each task. Finally, it gradually finds an appropriate task scheduling scheme through alternating iterative optimization.
- We verify the effectiveness of the proposed TS-DQNF method through simulation experiments. The results show that the TS-DQNF achieves a higher task success rate and a lower average processing delay under different network scenarios and different scales of UDs, while also exhibiting good convergence performance.

The rest of the paper is organized as follows: Section 2 reviews the related work. Section 3 introduces the proposed CPN system and formulates the system's optimization problem. Section 4 describes the proposed TS-DQNF in detail. Section 5 presents performance evaluation and analysis. Finally, Section 6 concludes the paper.

2. Related Work

2.1. Computing Power Network

In 2019, the industry began to investigate the CPN, and the proposed concept of CPN attracted significant research attention. In September 2021, the International Telecommunication Union (ITU) formally specified CPN in the ITU-T Y.2501 recommendation [17], where its layered architecture was described in detail. In 2023, the IETF [18] established the Computing-Aware Traffic Steering (CATS) working group, promoting the standardization of traffic steering. In recent years, many CPN-related problem scenarios have been widely discussed and investigated. The authors of [8]

 doi:10.20944/preprints202605.1329.v1

described the integration of CPN with 6G services and covered studies on computing, storage, and network deployment. Tang et al. [19] proposed a CPN task scheduling framework based on service-intent awareness. By using an intent-aware mechanism and an auction-based scheduling algorithm, this framework effectively matches task requirements with available computing resources. Liu et al. [20] implemented a CPN prototype testbed based on a microservice architecture and realized several key CPN enabling technologies, including computing modeling, computing awareness, and computation offloading. Li et al. [21] integrated CPN with unmanned aerial vehicle (UAV) scenarios and investigated solutions for task offloading and resource allocation in this scenario. Zhao et al. [22] studied the collaborative optimization of computation tasks and resource allocation in distributed CPN, reducing delay and improving system resource utilization. Liu et al. [23] proposed a framework that integrates time-series feature modeling with hierarchical reinforcement learning, effectively improving system resource utilization. Yin et al. [24] conducted an in-depth analysis of current studies and characteristics of terminal CPN, proposed a functional logical architecture for terminal CPN, and discussed the key technologies for promoting the development of CPN. Feng et al. [25] proposed a computing-aware and experience-driven DRL algorithm for the joint optimization of routing and scheduling in large-scale CPN, effectively ensuring balanced resource utilization.

These studies in CPN have significantly accelerated its evolution and development, providing a solid foundation for its practical and mature application. They also offer rich theoretical and practical support for the task scheduling study in the CPN scenarios presented in this paper.

2.2. Task Scheduling

In cloud computing and MEC scenarios, task scheduling studies usually involve computation offloading and resource allocation, and their core optimization objectives typically include reducing energy consumption, minimizing delay, and improving system throughput. Some existing studies have adopted heuristic methods to solve the task scheduling problem. Emara et al. [11] proposed an adaptive improved heuristic algorithm for task scheduling in cloud computing, effectively reducing task completion time and improving system resource utilization. Talaat et al. [12] proposed a task scheduling framework based on an adaptive genetic algorithm, which effectively shortened task completion time and improved the load balancing performance of the system. Sahraei et al. [13] addressed the scheduling problem in cloud computing environments by representing tasks as genetic encodings and proposed a new genetic algorithm model, achieving significant optimization in terms of cost, completion time, and resource utilization. Zhang et al. [26] proposed a novel hybrid heuristic algorithm to solve the task scheduling problem for heterogeneous virtual machines, effectively reducing the total task completion time and load imbalance while improving system resource utilization. Malik et al. [27] proposed an improved heuristic algorithm to efficiently promote parallel task scheduling in cloud computing, completing task scheduling with an optimal solution. However, these heuristic methods usually require a large amount of time to search for the target optimization scheme, making them difficult to adapt to complex network scenarios. With the rapid development of DRL, some studies have applied DRL methods to solve scheduling problems and have achieved significant results. Zhang et al. [14] proposed a reinforcement learning-based multi-objective task scheduling framework, which effectively improves system performance and enables efficient task scheduling in cloud-edge environments. Sudheer et al. [15] proposed an adaptive task scheduler by integrating a DRL method, which can effectively reduce task completion time and energy consumption. Qi et al. [16] proposed a scalable parallel task scheduling method for the task scheduling problem in Internet of Vehicles (IoV) scenarios. Yu et al. [28] proposed a novel reinforcement learning-based multi-objective task scheduling framework, which can minimize energy consumption, reduce costs, and guarantee quality of service under different workload conditions. Tian et al. [29] proposed a DRL method to solve the dynamic task scheduling problem in flight tests, effectively reducing task delay. However, these methods focus on optimization on the computing side and ignore the optimization of networking resources during data transmission. As a result, they fail to fully address the complexity introduced by the network and are difficult to apply directly to CPN scenarios.

 doi:10.20944/preprints202605.1329.v1

In summary, this paper considers the joint optimization of computation node selection and routing path planning in task scheduling under CPN scenarios, namely the simultaneous optimization of computing resources and networking resources. Accordingly, we propose a method to solve this problem, aiming to maximize the task success rate and minimize the average processing delay of tasks.

3. System Model and Problem Formulation

As shown in Figure 1, we construct a CPN task scheduling system that integrates computing and networking resources. The system consists of UDs, CPN routers, CPN nodes, a CPN controller, base stations, and some links connecting different nodes. In this system, the scheduling process of computation tasks is described as follows. When a UD generates a computation task, the task accesses the edge ingress CPN router through the wireless link of the base station. Then, the task is forwarded and transmitted among multiple CPN routers and finally reaches the target CPN node for computation. The system state information generated during the entire process is collected by the central CPN controller. The CPN controller deploys an intelligent scheduling algorithm, comprehensively formulates the optimal scheduling strategy, and then delivers the strategy to execute task scheduling. Finally, the execution result is returned to the UD, and the system state information is updated.

Due to the complexity of the CPN topology, the uncertainty of routing paths for task forwarding, and the uncertainty regarding which CPN node should execute each task, the response delay of task execution in the system may fluctuate significantly. Therefore, it is necessary to establish an optimization strategy that integrates the selection of computing resources and networking resources, so as to reduce the overall task scheduling cost and improve the overall system performance. The main symbols used in this paper are defined in Table 1.

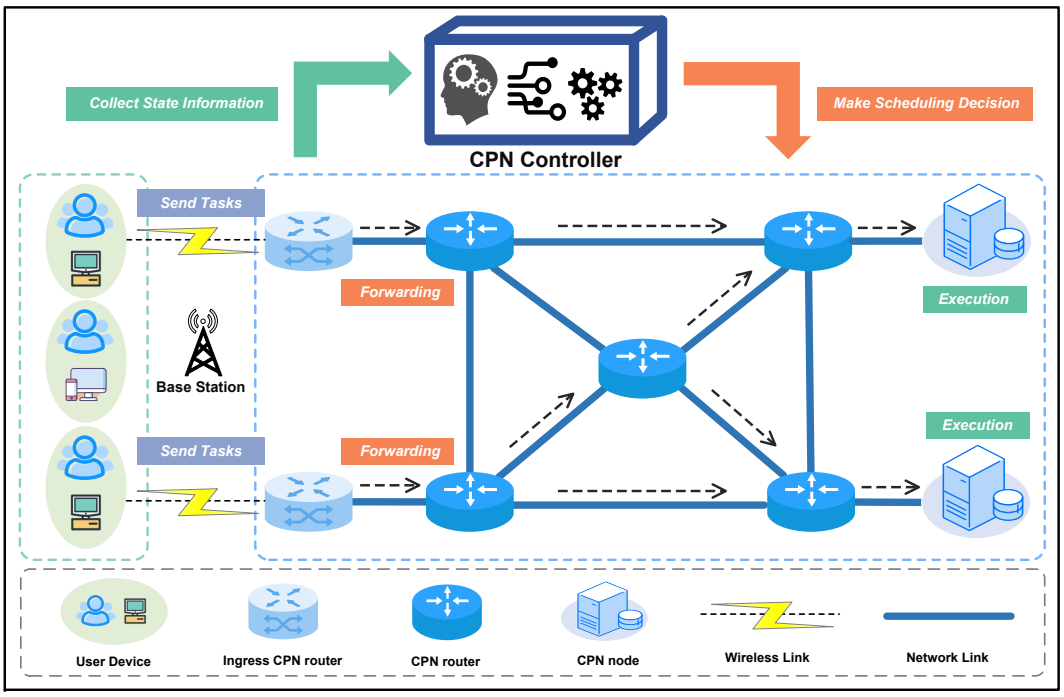


Figure 1. Proposed the CPN task scheduling system.

 doi:10.20944/preprints202605.1329.v1

Table 1. The connotation of main symbols.

Symbol	Connotation
$\mathcal{N}$	Set of UDs
$\mathcal{R}$	Set of CPN routers
$\mathcal{M}$	Set of CPN nodes
$g_n(t)$	Task generated by UD n in time slot t
$g_n^{dv}(t)$	Data volume of task $g_n(t)$
$g_n^{cv}(t)$	Computing volume of task $g_n(t)$
$g_n^{td}(t)$	Tolerable delay of task $g_n(t)$
cr_i	The i-th CPN router
f_i^{cr}	Forwarding capability of CPN router cr_i
b_i^{max}	Maximum load capacity of CPN router cr_i
$b_i(t)$	Real-time load of CPN router cr_i
$C_i(t)$	Congestion rate of CPN router cr_i
cn_i	The i-th CPN node
f_i^{cn}	Computing capacity of CPN node cn_i
$\lambda_i(t)$	Real-time load of CPN node cn_i
$\rho_i(t)$	Load rate of CPN node cn_i
F_n	Scheduling scheme of task $g_n(t)$
W	Network link transmission rate matrix
$\mathbb{H}(t)$	Task success rate
$\mathbb{D}(t)$	Task average processing delay
φ_1 & φ_2	Weighting coefficients for $\mathbb{H}(t)$ and $\mathbb{D}(t)$

3.1. CPN Model

To facilitate the discussion and analysis of the scheduling process, we divide time into fixed-duration time slots, denoted by $T = \{1, 2, ..., T\}$, where T is the total number of time slots. Let Δt indicate the duration of each time slot, and $t \in T$ indicate the index.

We define the set of UDs as $\mathcal{N} = \{1, 2, ..., N\}$, where each UD generates one computation task in each time slot t. The task generated by UD n in time slot t is denoted by $g_n(t) = \left(g_n^{dv}(t), g_n^{cv}(t), g_n^{td}(t)\right)$, where $g_n^{dv}(t)$ indicates the data volume of the task, $g_n^{cv}(t)$ indicates the computing volume of the task, and $g_n^{td}(t)$ indicates the maximum tolerable delay of the task. When the execution delay exceeds this value, the task is considered failed.

We define the set of CPN routers as $\mathcal{R} = \{1, 2, ..., R\}$, and indicate the i-th CPN router by cr_i. Each CPN router is indicated by a tuple $\left(f_i^{cr}, b_i^{\max}, b_i(t)\right)$, where f_i^{cr} indicates the forwarding capability of the CPN router, $b_i^{\max}$ indicates the maximum load capacity of the CPN router, and $b_i(t)$ indicates the real-time load of the CPN router in time slot t. To measure the congestion status of the router, we define the congestion rate of CPN router cr_i in time slot t as:

$$C_i(t) = \frac{b_i(t)}{b_i^{\max}} \tag{1}$$

We define the set of CPN nodes as $\mathcal{M} = \{1, 2, ..., M\}$, and indicate the i-th CPN node by cn_i. Each CPN node is indicated by a tuple $\left(f_i^{cn}, \lambda_i(t), \rho_i(t)\right)$, where f_i^{cn} indicates the computing capacity of CPN node cn_i, $\lambda_i(t)$ indicates the real-time load of CPN node cn_i in time slot t, and $\rho_i(t)$ indicates the load rate of CPN node cn_i in time slot t, which can be calculated as:

$$\rho_i(t) = \frac{\lambda_i(t)}{f_i^{cn}} \tag{2}$$

3.2. Communication Model

When a UD generates a task scheduling request, multiple CPN nodes in the network can provide computing power services, and multiple routing paths are available for reaching these CPN nodes.

 doi:10.20944/preprints202605.1329.v1

To clarify the scheduling process of tasks, we use the set $F = \{F_1, F_2, ..., F_n\}$ to indicate the scheduling schemes of all tasks, where F_n indicates the scheduling scheme of task $g_n(t)$, and $l_n = |F_n|$ indicates the length of this scheduling scheme. For example, suppose that $F_n = \{cr_2, cr_5, cr_7, cr_4, cn_2\}$. This indicates that task $g_n(t)$ is scheduled to CPN node cn_2 for execution, and the sequence of CPN routers it traverses is $[cr_2, cr_5, cr_7, cr_4]$.

The communication process for scheduling a task from a UD to a CPN node for execution is described as follows. First, the task is transmitted to a ingress CPN router through the wireless link of the base station. Then, the task is forwarded and transmitted among multiple CPN routers through network links. Finally, the task reaches the target CPN node for execution. The following describes these processes in detail.

3.2.1. From UDs to CPN Routers

Task $g_n(t)$ is transmitted to a ingress CPN router through a wireless link. To mitigate wireless channel interference caused by the concurrent transmission of multiple tasks, orthogonal frequency division multiplexing [30] is adopted. A unique subchannel is allocated to each task, and the total bandwidth is equally divided among all subbands. Therefore, the transmission rate of task $g_n(t)$ over the wireless link can be obtained using the Shannon formula [31], as follows:

$$v_n(t) = \frac{B^{tol}}{|N|} \cdot \log_2\left(1 + \frac{\epsilon_n(t) \cdot \eta_n(t)}{|N| \cdot \sigma_n^2(t)}\right) \tag{3}$$

where B^{tol} indicates the total bandwidth of the transmission channel, $\epsilon_n(t)$ indicates the transmission power, and $\eta_n(t)$ and $\sigma_n^2(t)$ indicate the channel gain and the power of the Gaussian white noise, respectively.

Therefore, the wireless transmission delay for task $g_n(t)$ can be calculated as:

$$D_n^{wtran}(t) = \frac{g_n^{dv}(t)}{v_n(t)} \tag{4}$$

3.2.2. From CPN routers to CPN nodes

Task $g_n(t)$ is forwarded and transmitted among multiple CPN routers through network links and finally reaches the target CPN node. We denote the transmission rate matrix of all links in the network by $W = [w_{i,j}]$, where $w_{i,j}$ indicates the link transmission rate between nodes in the network, satisfying:

$$w_{i,j} = \begin{cases} \infty, & i == j \\ Limited\ Rate, & i \neq j\ and\ (i,j) \in R, i \in R, j \in M \\ 0, & i \neq j\ and\ no\ connection \end{cases} \tag{5}$$

Considering the queuing and processing mechanism of CPN routers for tasks, the actual forwarding rate is affected by the congestion rate. Therefore, the delay generated by task $g_n(t)$ during the entire routing and forwarding process consists of two parts: the forwarding processing delay at CPN routers and the transmission delay over network links. The calculation formula is given as follows:

$$D_n^{ftran}(t) = \sum_{i=1}^{l_n-1} \left(\frac{g_n^{dv}(t)}{f_{F_n(i)}^{cr}\left(1 - C_{F_n(i)}(t)\right)} + \frac{g_n^{dv}(t)}{w_{F_n(i),F_n(i+1)}} \right) \tag{6}$$

In summary, we can calculate the delay incurred by task $g_n(t)$ throughout the entire communication process, as follows:

$$D_n^{trans}(t) = D_n^{wtran}(t) + D_n^{ftran}(t) \tag{7}$$

 doi:10.20944/preprints202605.1329.v1

3.3. Computing Model

After the communication process, task $g_n(t)$ is forwarded and transmitted among CPN routers and finally reaches the target CPN node for execution. According to the scheduling scheme F, the index of the target CPN node to which task $g_n(t)$ is scheduled is $F_n(l_n)$.

When task $g_n(t)$ reaches the target CPN node, based on the M/M/1 queuing model [32], the queuing delay of task $g_n(t)$ at the CPN node can be calculated as:

$$D_n^{que}(t) = \frac{\rho_{F_n(l_n)}(t)}{f^{cn}_{F_n(l_n)} - \lambda_{F_n(l_n)}(t)} \tag{8}$$

The delay for completing the computation of task $g_n(t)$ at the CPN node can be calculated as:

$$D_n^{com}(t) = \frac{g_n^{cv}(t)}{f^{cn}_{F_n(l_n)}} \tag{9}$$

Therefore, the total delay for completing the execution of task $g_n(t)$ at the CPN node can be obtained as follows:

$$D_n^{exe}(t) = D_n^{que}(t) + D_n^{com}(t) \tag{10}$$

3.4. Problem formulation

3.4.1. Success rate

For the overall system, the primary objective is the successful processing of computation tasks. Efficient task scheduling can maximize the task processing success rate, which is a key indicator of system performance. Therefore, we define the success rate of task processing as one of the optimization objectives. If a task is completed within its maximum tolerable delay, it is considered successful; otherwise, it is considered a failed task. Therefore, the success rate can be expressed as:

$$\mathbb{H}(t) = \frac{H^{suc}(t)}{H^{suc}(t) + H^{fail}(t)} \tag{11}$$

where $H^{suc}(t)$ and $H^{fail}(t)$ indicate the numbers of successful and failed tasks in time slot t, respectively.

3.4.2. Average Processing Delay

Delay is a key factor affecting user experience. Therefore, minimizing the delay of task processing is crucial for improving system performance. According to the above model, the total delay required to complete the scheduling of a single task $g_n(t)$ in time slot t can be obtained as follows:

$$D_n(t) = D_n^{trans}(t) + D_n^{exe}(t) \tag{12}$$

Therefore, we define the average processing delay of globally successful task processing as one of the optimization objectives, which can be expressed as:

$$\mathbb{D}(t) = \frac{1}{H^{suc}(t)} \sum_{n=1}^{N} D_n(t) \tag{13}$$

3.4.3. Optimization Problem

Based on the above optimization objectives, it is necessary to formulate a clear optimization problem to determine the optimal scheduling scheme, so as to maximize the task success rate in the

 doi:10.20944/preprints202605.1329.v1

CPN system and minimize the average processing delay of global tasks. Therefore, we define the objective function of this paper as follows:

$$\begin{aligned} &\max_{F} \frac{1}{T}\sum_{t=1}^{T}\left(\varphi_1\mathbb{H}(t)-\varphi_2\mathbb{D}(t)\right) \\ \text{s.t. } &C1:\ D_n(t)\le g_n^{td}(t),\ \forall\, n\in N \\ &C2:\ 0\le C_i(t)\le 1,\ \forall\, i\in R \\ &C3:\ 0\le \rho_i(t)\le 1,\ \forall\, i\in M \end{aligned} \tag{14}$$

where φ_1 and φ_2 are weight coefficients. $C1$ indicates that the total delay of a task cannot exceed its maximum tolerable delay. $C2$ indicates the congestion-rate constraint of CPN routers, and $C3$ indicates the load-rate constraint of CPN nodes.

4. Proposed Method

Task scheduling in the CPN is essentially a typical joint optimization problem. It needs not only to select an appropriate CPN node for each task to ensure the reasonable use of computing resources, but also to plan a suitable routing path to the selected CPN node in a complex network structure to ensure the reasonable planning of networking resources. Therefore, the two aspects need to be collaboratively optimized. Based on this idea, this paper proposes a task scheduling method that integrates DQN with the dynamic Floyd algorithm, called TS-DQNF. Specifically, the TS-DQNF first uses the DQN algorithm to determine the target CPN node for each task. Then, a congestion-aware dynamic Floyd algorithm is introduced to determine the shortest routing path from the task to the selected CPN node. Finally, through multiple rounds of alternating iterations and collaborative optimization, an appropriate task scheduling scheme is gradually obtained. By performing joint optimization, the TS-DQNF can achieve efficient CPN task scheduling, aiming to reduce the average processing delay of tasks and improve the task scheduling success rate. Figure 2 illustrates the overall execution framework of the proposed TS-DQNF.

4.1. DRL-Based Markov Decision Process for Task Scheduling

In the CPN system environment, task scheduling is a long-term process. We model the task scheduling decision-making process as a Markov decision process (MDP). Mathematically, an MDP is an idealized form of reinforcement learning [33]. The state space, action space, state transition function, and reward function of the MDP are described in detail as follows.

- **State space:** To ensure that the CPN controller has comprehensive awareness of the system, it is necessary to fully capture the state information of the system environment, enabling the controller to make informed decisions. Therefore, the system state space in time slot t can be defined as:

$$S(t)=\left\{S^{ts}(t),S^{cr}(t),S^{cn}(t)\right\} \tag{15}$$

  where $S^{ts}(t)=\left\{\left\{g_n^{dv}(t),g_n^{cv}(t),g_n^{td}(t)\right\}_{(n\in N)}\right\}$ indicates the state information of tasks, $S^{cr}(t)=\left\{\left\{f_i^{cr},b_i^{\max},b_i(t),C_i(t)\right\}_{(i\in M)}\right\}$ indicates the state information of CPN routers, $S^{cn}(t)=\left\{\left\{f_i^{cn},\lambda_i(t),\rho_i(t)\right\}_{(i\in K)}\right\}$ indicates the state information of CPN nodes.
- **Action space:** In the CPN system environment, an action refers to the scheduling decision formulated by the CPN controller, which determines where a task will finally be scheduled for execution. Therefore, the action space for task scheduling is defined as:

$$A=\{A_1,A_2,...,A_N\} \tag{16}$$

 doi:10.20944/preprints202605.1329.v1

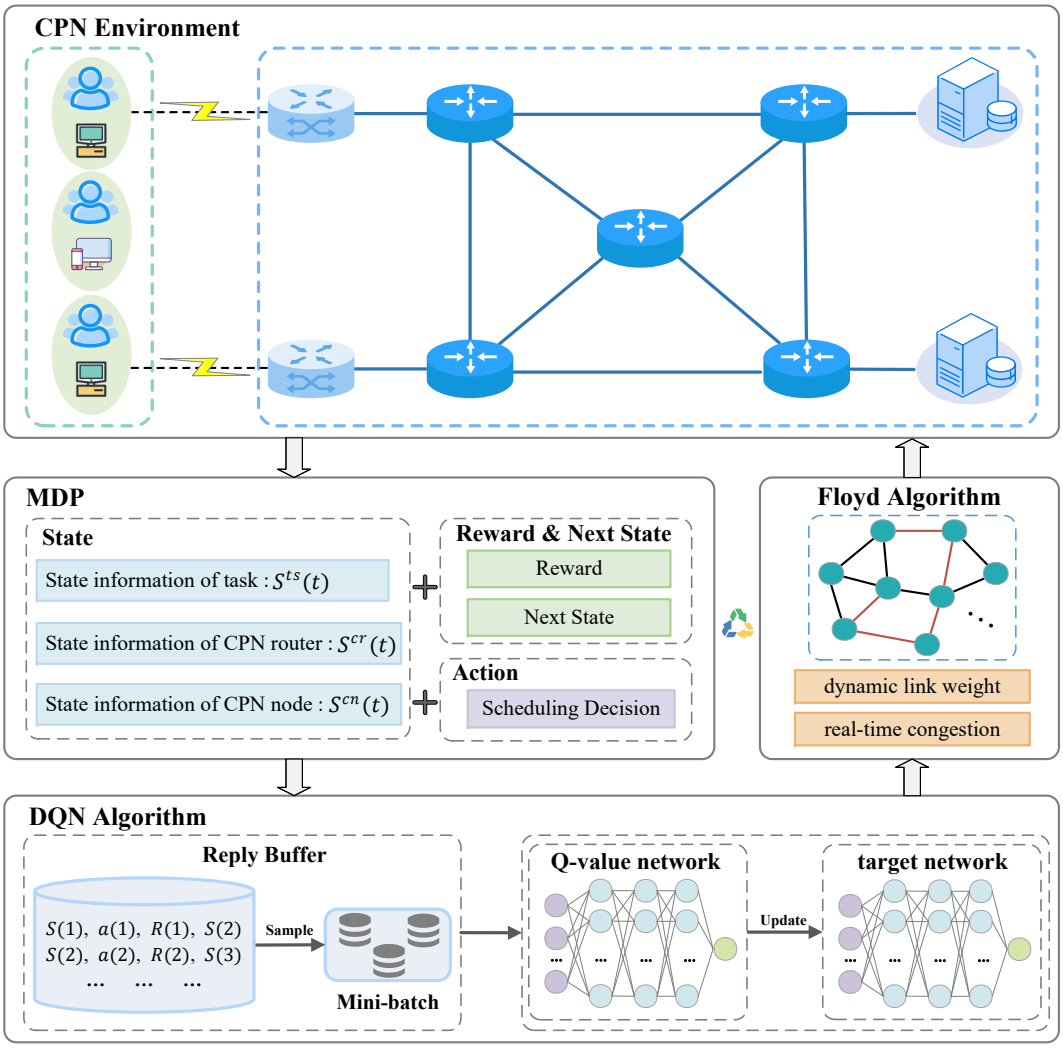


Figure 2. The overall execution framework of the proposed TS-DQNF.

where A_n indicates the action space of task $g_n(t)$, including $\{1, 2, ..., |M|\}$, and $a(t) \in A$ indicates a task scheduling action executed by the system in time slot t.

- **State-transition function:** The state-transition function models the transition from the current state $S(t)$ to the next state $S(t+1)$ after actions $a(t)$ are executed at time step t. We define the state transition function as $ST(S(t), a(t))$, which returns the next system state after executing action $a(t)$.
- **Reward function:** In DRL, the reward is the feedback from the environment after the agent executes an action, which is used to evaluate the impact of its scheduling decision on the system. As described above, this paper aims to improve the task success rate while minimizing the average processing delay of successfully executed tasks. Therefore, we define the immediate reward of the system environment in time slot t as:

$$R(t) = \begin{cases} \varphi_1 \mathbb{H}(t) - \varphi_2 \mathbb{D}(t), & success \\ -1, & fail \end{cases} \tag{17}$$

4.2. Decision Model Training of Task Scheduling Based on DQN

Based on the above MDP formulation, we adopt the DQN algorithm [34] to train the task scheduling decision model. Through continuous interaction with the system environment, an agent capable of making high-quality scheduling decisions is trained. The DQN algorithm combines Q-learning with a deep neural network (DNN), and can effectively solve complex task scheduling problems.

 doi:10.20944/preprints202605.1329.v1

It evaluates the value of each possible action under a given state by learning the state-action value function, and learns the optimal policy by maximizing the cumulative reward. By approximating the Q-value function with a DNN, DQN can handle high-dimensional state and action spaces, and improve training stability through the experience replay buffer and target network mechanisms.

Algorithm 1 presents the training process of the task scheduling decision model. First, the parameters required for training are initialized (Line 3). For each episode, the system environment is first reset, and then the agent learns through interactions with the environment. At each time step t, the algorithm first observes the environment to obtain the system state $S(t)$ (Line 7). Then, the agent selects an action $a(t)$ using the ε-greedy policy and computes the shortest routing path through Algorithm 2, thereby updating the task scheduling scheme F (Lines 8-10). Next, task scheduling is executed in the environment according to action $a(t)$ and the shortest routing path, obtaining the corresponding immediate reward R(t), while the system transfers to the next state $S(t+1)$ (Lines 11-12). Finally, the current experience tuple $< S(t), a(t), R(t), S(t+1) >$ is stored in the experience reply buffer B (Line 13). At the end of each episode, the task scheduling scheme F is reset, and the exploration rate ε is linearly decreased (Lines 15-16). When the capacity of the reply buffer is sufficiently large, the algorithm randomly samples a batch of experience data from it and calculates the loss function using the mean square error (MSE) [35]. Then, the parameters θ of the Q-value network are updated through backpropagation (Lines 17-19). The loss function is defined as follows:

$$L(\theta^t) = \left(r + \gamma \cdot \max Q(S', a'; \theta'^t) - Q(S, a; \theta^t)\right)^2 \tag{18}$$

where γ indicates the discount factor, and θ^t and θ'^t indicate the parameters of the Q-value network and the target network in time slot t, respectively. After every k episodes, the target network parameters θ' are replaced by the Q-value network parameters θ, further stabilizing the training process (Lines 20-22). After E^{max} episodes, the model is expected to converge stably and returns the Q-value network parameters θ.

Algorithm 1 Training Process of Task Scheduling Decision Model Based on DQN

1: **Input:** Task scheduling scheme F, discount factor γ, exploration rate ε, learning rate α, max episode E^{max}, and replay buffer B,
2: **Output:** The Q-value network parameters θ of the trained model
3: **Initialize:** Q-value network parameters θ, target network parameters θ'
4: **for** each episode e in E^{max} **do**
5: Reset the CPN system environment
6: **for** each step t **do**
7: Observe system state $S(t)$
8: Select action $a(t)$ using the ε-greedy policy
9: Calculate the shortest routing path according to Algorithm 2
10: Update scheduling scheme F according to $a(t)$ and shortest routing path
11: Execute task scheduling according to action $a(t)$ and shortest routing path
12: Obtain the instant reward $R(t)$ and transition to the next state $S(t+1)$
13: Store $< S(t), a^t, R(t), S(t+1) >$ into B
14: **end for**
15: Reset the task scheduling scheme F
16: Linear decrease of the exploration rate ε
17: Draw a batch of experience from B
18: Calculate the loss function $L(\theta^t)$ according to Eq. (18)
19: Update the Q-value network parameters θ using backpropagation
20: **if** e mod $k = 0$ **then**
21: Update the target network parameters $\theta' = \theta$
22: **end if**
23: **end for**
24: **Return** θ

 doi:10.20944/preprints202605.1329.v1

4.3. Dynamic Congestion-Aware Shortest Routing Path Planning

After the target CPN node is determined, we apply the Floyd algorithm [36] to plan the shortest routing forwarding path from the ingress router to the selected node for the task. Since the congestion condition of CPN routers in the network changes dynamically, while the Floyd algorithm itself lacks a mechanism for avoiding congestion or adapting to dynamic network conditions, we dynamically implement the Floyd algorithm based on real-time network congestion conditions. That is, the algorithm recalculates the shortest routing path according to the congestion rates of CPN routers in the network, rather than calculating paths only once and reusing them throughout the entire scheduling period. Specifically, at the beginning of time slot t, for any two adjacent CPN routers u and v in the network, the dynamic weight of their link is defined as:

$$l_{u,v}^{w} = \frac{1}{w_{u,v} \cdot (1 - C_u(t)) \cdot (1 - C_v(t))} \tag{19}$$

where the link weight is set to infinity if the link fails or if either of the adjacent routers has a congestion rate of 1. This weight is jointly determined by the link transmission rate and the real-time congestion rates of the routers at both ends. A more severe congestion condition leads to a larger link weight, thereby guiding the routing path planning process to avoid highly congested routers.

Based on this, this paper constructs a dynamic link weight matrix and uses it as the input of the Floyd algorithm to implement a congestion-aware shortest routing path update mechanism. Algorithm 2 describes the dynamic congestion-aware shortest routing path planning process. The inputs of the algorithm include the ingress CPN router, the target CPN node, the set of CPN routers, the link transmission rate matrix, and the real-time congestion rates of all routers in the current time slot. The output is the shortest routing path from the ingress router to the target CPN node.

First, the algorithm initializes the dynamic link weight matrix and the predecessor matrix (Line 3). Then, the algorithm traverses all router node pairs in the network to determine whether each link is reachable and whether the routers at both ends are in a fully congested state. If the link is unreachable or a fully congested node exists, the corresponding link weight is set to infinity. Otherwise, the dynamic link weight is calculated according to Eq. (19), and the predecessor information is updated (Lines 4-10). Based on this, the algorithm uses the Floyd algorithm to iteratively update the dynamic link weight matrix (Line 11). Finally, the algorithm selects the CPN router connected to the target CPN node and reconstructs the routing path by backtracking according to the predecessor matrix (Lines 12-13). By re-executing this algorithm at each task scheduling moment, routing path planning can respond to changes in the network congestion status in real time, thereby reducing task transmission delay and improving the scheduling success rate.

Algorithm 2 Dynamic Congestion-Aware Shortest Path Planning via Floyd

1: **Input:** Ingress CPN router cr_{in}, target CPN node cn_{tar}, the CPN router set $\mathcal{R}$, link transmission rate matrix W, and real-time congestion rates $\{C(t)\}_{\mathcal{R}}$
2: **Output:** The shortest routing path p
3: **Initialize:** the dynamic link weight matrix L^w and predecessor matrix Pre
4: **for** each node pair (u, v) in $\mathcal{R}$ **do**
5: **if** $w_{u,v} = 0$ or $C_u(t) = 1$ or $C_v(t) = 1$ **then**
6: Update the dynamic link weight $l_{u,v}^{w}$ to infinity
7: **else**
8: Calculate the dynamic link weight $l_{u,v}^{w}$ according to Eq. (19)
9: Add node u to predecessor matrix $Pre_{u,v}$
10: **end if**
11: **end for**
12: Apply Floyd algorithm to update L^w and Pre
13: Select the CPN router directly connected to target CPN node cn_{tar}
14: Reconstruct the routing path p according to Pre
15: **Return** p

 doi:10.20944/preprints202605.1329.v1

5. Performance Evaluation

In this section, we first introduce the simulation settings. Next, the proposed TS-DQNF is evaluated considering the following Research Questions (RQs):

- **RQ 1:** How is the convergence performance of the TS-DQNF? (Section 5.2)
- **RQ 2:** How is the performance of the TS-DQNF and other methods in different network scenarios? (Section 5.3)
- **RQ 3:** How is the effect of different UD numbers on the performance of the TS-DQNF and other methods? (Section 5.4)

5.1. Simulation Settings

We develop the proposed simulation environment using Python and are implementing the proposed TS-DQNF with the PyTorch framework. We reference to the open-source dataset Survivable Network Design Lib [37] to simulate and generate network scenarios, among which three different network scenarios are selected, namely France, India35, and Cost266. Table 3 shows the numbers of CPN nodes, CPN routers, and UDs configured in different network scenarios, and Figure 3 illustrates the specific network topologies of different network scenarios, where the blue nodes represent CPN routers and the green nodes represent CPN nodes. More detailed settings of the simulation parameters are presented in Table 2.

To verify the superiority of the proposed TS-DQNF, we compare it with the following methods:

- **PSO** [27]**:** This method is a commonly used heuristic method for solving task scheduling problems. It is adopting the particle swarm optimization algorithm to determine the target CPN node for each task and, at the same time, compute the routing path for task scheduling.
- **KDRL** [25]**:** The core of this method is a joint optimization algorithm for routing and scheduling. It is first adopting the K-shortest path (KSP) algorithm to determine a set of candidate routing paths. Then, a DRL agent is used to jointly select the target node and routing path of each task from the generated candidate set.
- **RS:** This method is adopting a random scheduling strategy, namely randomly selecting a CPN node for each task and randomly generating a routing path.
- **GS:** This method is adopting a greedy scheduling strategy. Specifically, it selects the CPN node with the lowest processing delay for each task each time, and selects the link with the lowest transmission delay each time as part of the path to generate the final routing path.

Table 2. Settings of simulation parameters [25][38][39].

Parameter	Value
Number of UDs	[10,30]
Data volume of task $g_n^{dv}(t)$	[1.0,8.0] MB
Computing volume of task $g_n^{cv}(t)$	[1.0,5.0] GCPU
Tolerable delay of task $g_n^{td}(t)$	2 s
Forwarding capability of CPN router f_i^{cr}	[75,85] MB/s
Network link transmission rate $w_{i,j}$	[400,500] MB/s
Computing capacity of CPN node f_i^{cn}	[10,15] GCPU/s
Weighting coefficients φ_1, φ_2	0.5,0.5
Discount factor	0.95
Exploration rate	0.1
Learning rate	0.0001
Replay batch size	64
Training episodes	1000

 doi:10.20944/preprints202605.1329.v1

Table 3. Configurations for different network scenarios.

Scenarios	Name	No. of CPN Routers	No. of CPN Nodes	No. of CPN UDs
1	France	21	4	20
2	India35	30	5	20
3	Cost266	30	7	20

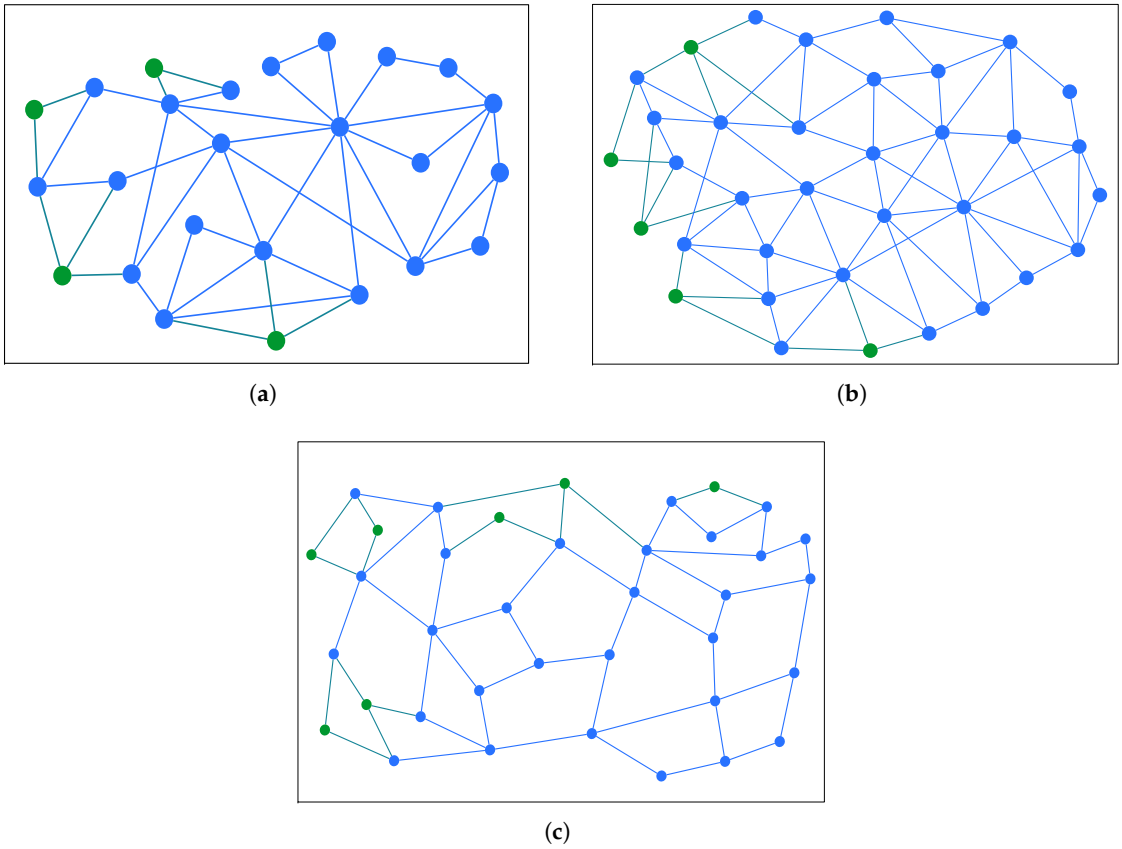


Figure 3. Topology of different network scenarios utilized for simulation: (**a**) France; (**b**) India35; (**c**) Cost266.

5.2. RQ 1: Convergence Performance of the TS-DQNF

Before the DRL algorithm is formally evaluated and applied, offline training is required to explore the convergence performance of DRL, so as to iteratively optimize and refine the decision model. Therefore, we conduct offline training of the DRL neural network in the test environment under three network scenarios. Specifically, we set different neural network hyperparameters for the proposed TS-DQNF method and conducting offline training separately. For each training process, we set 2000 episodes for offline training. Figure 4 shows the training convergence performance of the proposed TS-DQNF method under different learning rates, denoted as LR, in different network scenarios. The results indicate that different learning-rate settings are significantly affecting the training performance of the reinforcement learning agent. A higher learning rate can achieve faster convergence, whereas a lower learning rate may lead to certain fluctuations in rewards. When the learning rate is set to 0.0001, the TS-DQNF achieves the highest reward value while also showing good convergence performance and a fast convergence speed. Therefore, we set the learning rate to 0.0001 in the subsequent test evaluation.

 doi:10.20944/preprints202605.1329.v1

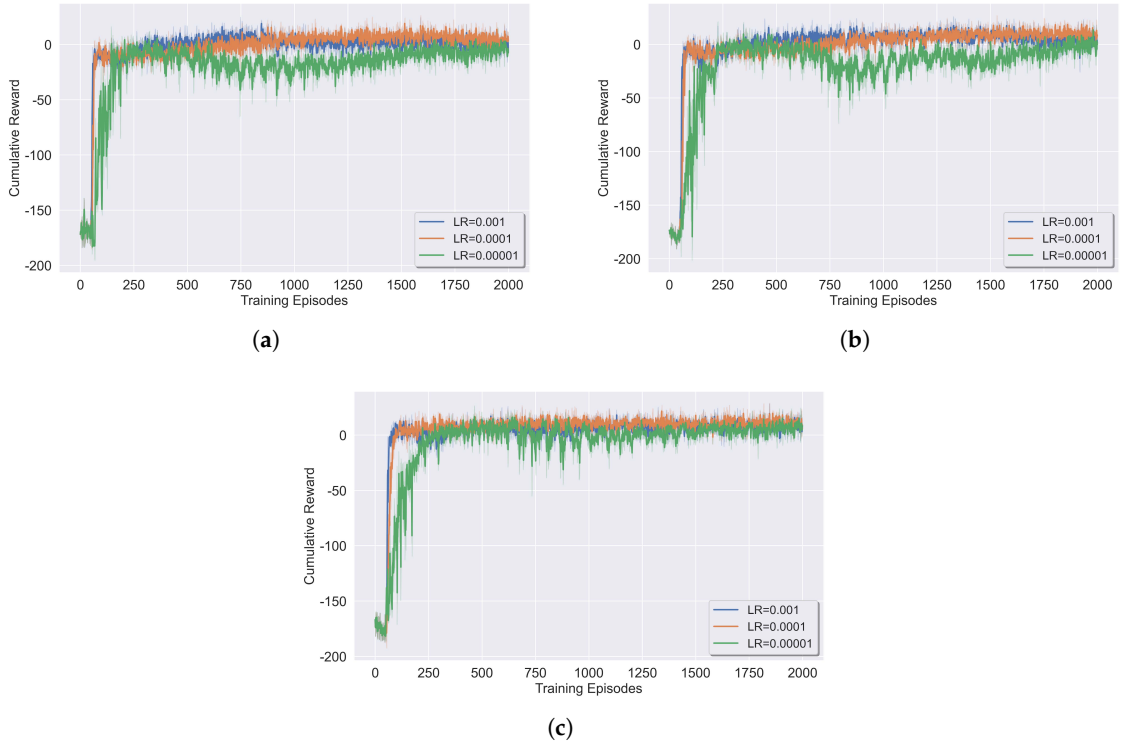


Figure 4. Training convergence performance of the proposed TS-DQNF under different learning rates in different network scenarios: (**a**) France; (**b**) India35; (**c**) Cost266.

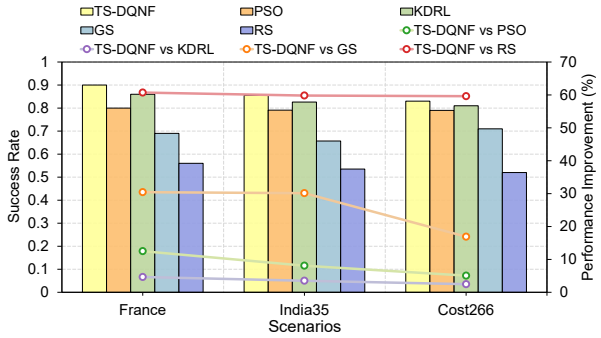
5.3. RQ2: Performance Analysis of Each Method in Different Network Scenarios

To simulate the diversity of different network scenario scales in the CPN environment, we set specific numbers of CPN nodes, CPN routers, and UDs for the three selected scenarios, as shown in Table 3. For each network scenario, we compare the TS-DQNF with other methods. Figure 5 shows the comparison between the TS-DQNF and other methods in terms of success rate and average processing delay under different network scenarios. As shown in the figure, the TS-DQNF outperforms PSO, KDRL, GS, and RS in every network scenario. Specifically, under the three network scenarios, the task success rate of the TS-DQNF improves by 5.06%-12.50%, 2.47%-4.65%, 16.91%-30.43%, and 59.61%-60.71% compared with PSO, KDRL, GS, and RS, respectively, while the average processing delay is reduced by 4.85%-6.53%, 1.92%-4.97%, 8.75%-10.57%, and 13.85%-16.87%, respectively. This is because the TS-DQNF decouples the complex joint scheduling problem and performs collaborative optimization. Specifically, the TS-DQNF employs the DQN agent to make global decisions on target CPN nodes, thereby reducing the computing cost of task scheduling. Meanwhile, it introduces the real-time congestion-aware dynamic Floyd algorithm to compute routing paths, further reducing the transmission cost of task scheduling and improving the overall optimization effect.

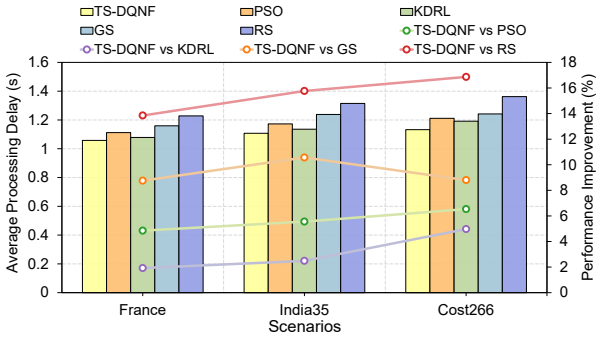
In contrast, although KDRL also employs deep reinforcement learning, it relies on the static KSP algorithm to generate the candidate set. When facing a dynamic and congested network environment, the paths in the candidate set may all be in a highly congested state, thereby increasing the transmission cost of scheduling. As a heuristic algorithm, PSO has a certain global search capability. However, as the network topology becomes more complex, its solution space grows exponentially, making it prone to falling into local optima and unable to respond to dynamic changes in the network environment in a timely manner. The GS method adopts a greedy strategy and relies only on local optimal information at the current moment. Such a shortsighted decision-making strategy can easily lead to severe network congestion when multiple tasks arrive simultaneously. The RS method makes completely blind decisions, inevitably resulting in extremely high delay and the lowest task success rate. In addition, since the India35 and Cost266 network scenarios are slightly more complex than the France network

 doi:10.20944/preprints202605.1329.v1

scenario, the overall task scheduling cost in India35 and Cost266 is slightly higher than that in the France scenario.



(**a**)



(**b**)

Figure 5. RQ2: Performance comparison of the different methods in different network scenarios: (**a**) Success Rate; (**b**) Average Processing Delay.

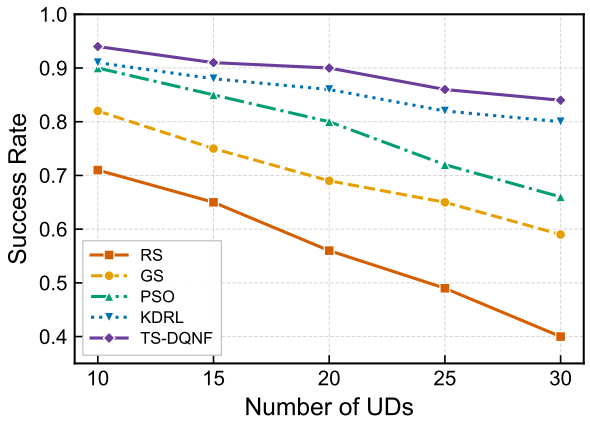
5.4. RQ3: Performance Analysis of Different UD Numbers on Each Method

To explore the performance impact of a sharp increase in the number of tasks on different methods, we test the performance of different methods with different scales of UDs in each network scenario. Specifically, we set the number of UDs to 10, 15, 20, 25, and 30, while keeping the numbers of CPN nodes and CPN routers unchanged, consistent with the settings in Table 3. Figures 6,7 and 8 show the performance trends of the TS-DQNF and other methods as the number of UDs increases under different network scenarios. As shown in the figures, when the numbers of CPN nodes and CPN routers remain unchanged, the average processing delay of each method in each network scenario exhibits an upward trend, while the success rate of each method shows a downward trend. This is because, as the number of UDs increases, the number of generated computation tasks is also increasing, leading to more intense resource competition and network congestion. Compared with other methods, the proposed TS-DQNF method achieves a higher task success rate and a lower average processing delay. Moreover, as the number of UDs continues to increase, the performance gap between the TS-DQNF and other methods is gradually widening. This is because the TS-DQNF can make better scheduling decisions under high network congestion and high CPN-node load, and can avoid congested network links to plan routing paths with the minimum transmission cost.

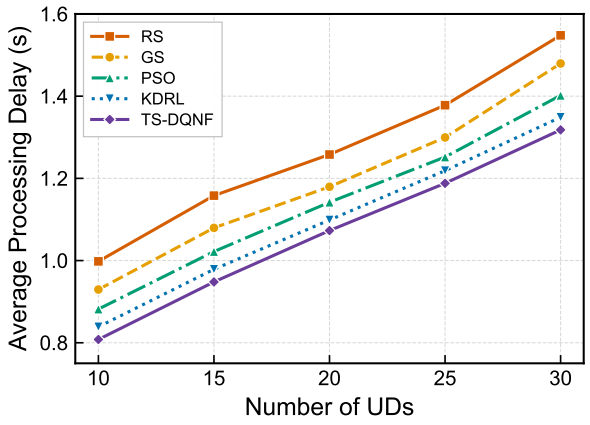
In contrast, although the task success rate of PSO approaches that of the TS-DQNF when the number of UDs is 10, it encounters higher computational complexity as the number of UDs increases, leading to a sharp increase in computing overhead. As a result, its performance gap is gradually widening, and it is difficult for PSO to find reasonable scheduling schemes for all tasks within a limited decision-making time. By comparison, both GS and RS exhibit a large performance gap compared with the TS-DQNF. The former lacks a global perspective during decision-making and often falls into local optima, while the latter shows obvious blindness in decision-making. When the number of

 doi:10.20944/preprints202605.1329.v1

tasks increases, RS completely ignores the node load status and network congestion status, thereby producing a higher task scheduling cost.



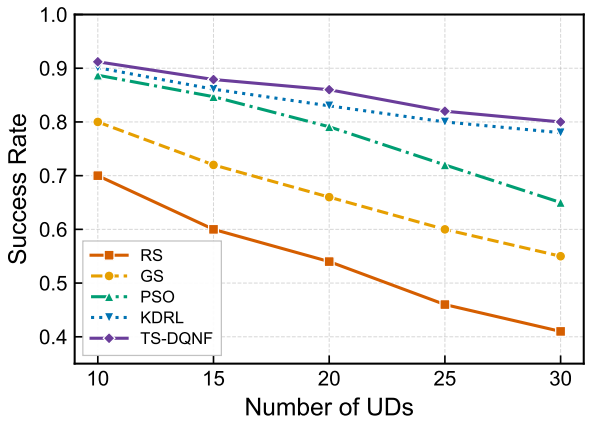
(**a**)



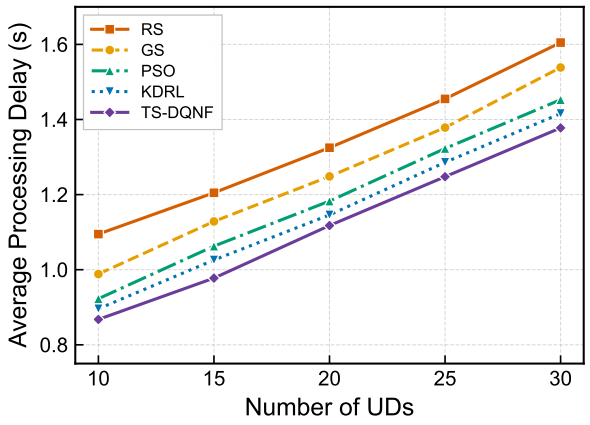
(**b**)

Figure 6. Effect of different numbers of UDs on the performance of each method in the France scenario: (**a**) Success Rate; (**b**) Average Processing Delay.

 doi:10.20944/preprints202605.1329.v1



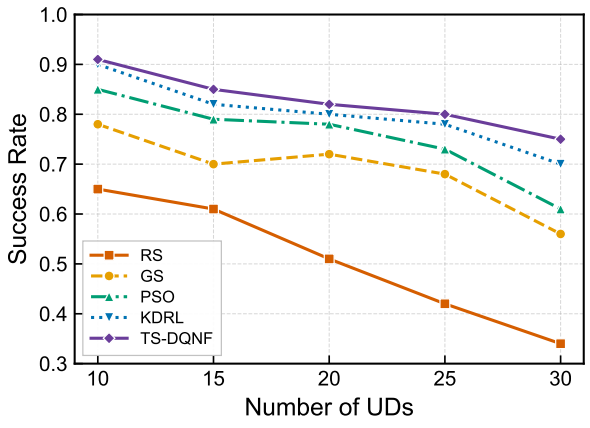
(**a**)



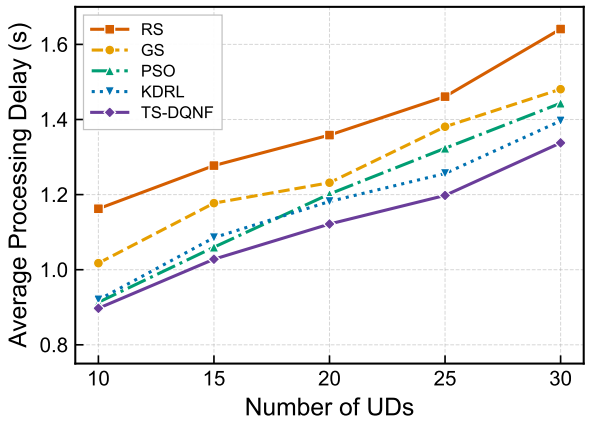
(**b**)

Figure 7. Effect of different numbers of UDs on the performance of each method in the India35 scenario: (**a**) Success Rate; (**b**) Average Processing Delay.

 doi:10.20944/preprints202605.1329.v1



(**a**)



(**b**)

Figure 8. Effect of different numbers of UDs on the performance of each method in the Cost266 scenario: (**a**) Success Rate; (**b**) Average Processing Delay.

6. Conclusion

In this paper, we propose a task scheduling method for the CPN, called TS-DQNF, aiming to improve the task scheduling success rate and reduce the average processing delay. The proposed TS-DQNF first uses the DQN algorithm to determine the target CPN node for each task. Then, considering network congestion, it introduces a congestion-aware dynamic Floyd algorithm to compute the shortest routing path for task scheduling. Finally, through multiple rounds of alternating iterative optimization, an appropriate task scheduling scheme is gradually obtained. Simulation experiments show that the proposed TS-DQNF outperforms other methods under different network scenarios and different scales of UDs, effectively improving the task success rate and reducing the average processing delay.

Author Contributions: Conceptualization, C.Y., X.R. and J.C.; methodology, X.R.; software, X.R.; validation, C.Y. and J.C.; formal analysis, C.Y.; investigation, X.R.; resources, X.R.; data curation, X.R.; writing—original draft preparation, X.R.; writing—review and editing, C.Y.; visualization, X.R.; supervision, J.C.; project administration, X.R.; funding acquisition, J.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Key R&D Projects of Guangxi Science and Technology Program(Grant No. GuikeFN2504240026), and the Key R&D Projects of Guangxi Science and Technology Program(Grant No. GuikeFN2600640290).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

 doi:10.20944/preprints202605.1329.v1

Data Availability Statement: The dataset used in this study was publicly available. The network scenario topology dataset used in the simulation environment is available from https://sndlib.put.poznan.pl/home.action.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Wang, L.; Xu, W.; Liu, Y.; Wang, M. Artificial intelligence for virtual reality: A review. *Science China Information Sciences* **2025**, *69*, 111101.
2. Badue, C.; Guidolini, R.; Carneiro, R.V.; Azevedo, P.; Cardoso, V.B.; Forechi, A.; Jesus, L.; Berriel, R.; Paixão, T.M.; Mutz, F.; De Paula Veronese, L.; Oliveira-Santos, T.; De Souza, A.F. Self-driving cars: A survey. *Expert Systems with Applications* **2021**, *165*, 113816.
3. Liu, F.; Zhao, Q.; Liu, X.; Zeng, D. Joint face alignment and 3D face reconstruction with application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2020**, *42*, 664–678.
4. Ye, J. Artificial intelligence-generated content (AIGC) in biomedical research, healthcare delivery, and clinical practices: Technologies, applications, and regulatory considerations. *Artificial Intelligence Review* **2026**, *59*, 86.
5. Wang, W.-X.; Wu, J. Survey on hypergraph algorithms: Foundations, advances, and applications in cloud computing. *Journal of Computer Science and Technology* **2026**.
6. Tesfay, C.H.; Xiang, Z.; Yang, L.; Mahmood, J.; Ren, M.; Zhang, S.; Das, A.K.; Chaudhry, S.A. Task offloading and optimization methods in UAV-enabled mobile edge computing: A comprehensive survey. *Computer Communications* **2026**, *254*, 108537.
7. Xiao, H.; Xu, C.; Ma, Y.; Yang, S.; Zhong, L.; Muntean, G.-M. Edge intelligence: A computational task offloading scheme for dependent IoT application. *IEEE Transactions on Wireless Communications* **2022**, *21*, 7222–7237.
8. Tang, X.; Cao, C.; Wang, Y.; Zhang, S.; Liu, Y.; Li, M.; He, T. Computing power network: The architecture of convergence of computing and networking towards 6G requirement. *China Communications* **2021**, *18*, 175–185.
9. Sun, Y.; Lei, B.; Liu, J.; Huang, H.; Zhang, X.; Peng, J.; Wang, W. Computing power network: A survey. *China Communications* **2024**, *21*, 109–145.
10. Sun, W.; Li, Z.; Wang, Q.; Zhang, Y. FedTAR: Task and resource-aware federated learning for wireless computing power networks. *IEEE Internet of Things Journal* **2023**, *10*, 4257–4270.
11. Emara, F.A.; Gad-Elrab, A.A.A.; Sobhi, A.; Alsharkawy, A.S.; Embabi, M.E.; El-Baky, M.A.A. Multi-objective task scheduling algorithm for load balancing in cloud computing based on improved Harris hawks optimization. *The Journal of Supercomputing* **2025**, *81*, 790.
12. Talaat, F.M.; Hamza, A.A. CloudSched-GA: An adaptive genetic optimizer for efficient and balanced task scheduling in cloud ecosystems. *Neural Computing and Applications* **2025**, *37*, 28269–28293.
13. Sahraei, S.H.; Kashani, M.M.R.; Rezazadeh, J.; Farahbakhsh, R. Efficient job scheduling in cloud computing based on genetic algorithm. *International Journal of Communication Networks and Distributed Systems* **2019**, *22*, 447–467.
14. Zhang, W.; Ou, H. Reinforcement learning based multi objective task scheduling for energy efficient and cost effective cloud edge computing. *Scientific Reports* **2025**, *15*, 41716.
15. Mangalampalli, S.; Karri, G.R.; Ratnamani, M.V.; Mohanty, S.N.; Jabr, B.A.; Ali, Y.A.; Ali, S.; Abdullaeva, B.S. Efficient deep reinforcement learning based task scheduler in multi cloud environment. *Scientific Reports* **2024**, *14*, 21850.
16. Qi, Q.; Zhang, L.; Wang, J.; Sun, H.; Zhuang, Z.; Liao, J.; Yu, F.R. Scalable parallel task scheduling for autonomous driving using multi-task deep reinforcement learning. *IEEE Transactions on Vehicular Technology* **2020**, *69*, 13861–13874.
17. ITU-T. *Computing Power Network-Framework and Architecture: Y.2501*; 2021.
18. Xia, L.; Guo, D.; Wang, Y.; Sun, D.; Zhen, W.; Jing, C. Optimal load scheduling based on mobile edge computing technology in 5G dense networking. In Proceedings of the 2022 3rd Asia Conference on Computers and Communications (ACCC), 2022; pp. 137–142.
19. Tang, Q.; Xie, R.; Feng, L.; Yu, F.R.; Chen, T.; Zhang, R.; Huang, T. SIaTS: A service intent-aware task scheduling framework for computing power networks. *IEEE Network* **2024**, *38*, 233–240.
20. Liu, J.; Sun, Y.; Su, J.; Li, Z.; Zhang, X.; Lei, B.; Wang, W. Computing power network: A testbed and applications with edge intelligence. In Proceedings of the IEEE INFOCOM 2022–IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2022; pp. 1–2.

 doi:10.20944/preprints202605.1329.v1

21. Li, J.; Zeng, Q.; Lu, L.; Zhang, E.; Xu, A.; Li, W. Task offloading and computing resource allocation of the joint UAV with computing power network. *The Journal of Supercomputing* **2025**, *81*, 884.
22. Zhao, W.; Huang, X.; Li, D.; Zhang, P.; Xie, K. Intelligent task scheduling towards distributed computing power network. In *Advanced Intelligent Computing Technology and Applications*; Huang, D.-S.; Zhang, C.; Zhang, Q.; Pan, Y., Eds.; Springer Nature Singapore: Singapore, 2025; pp. 495–506.
23. Liu, H.; Zhang, S.; Li, L.; Sun, T.; Xue, W.; Yao, X.; Xu, Y. Computing power network dynamic resource scheduling integrating time series mixing dynamic state estimation and hierarchical reinforcement learning. *Scientific Reports* **2026**, *16*, 2905.
24. Yin, M.; Gao, X.; Wu, S.; Wang, H.; Zhou, T.; Lin, W. Key technologies for terminal computing power network. In *Proceedings of the 15th International Conference on Computer Engineering and Networks*; Cheng, Z.; Liu, X.; Su, J., Eds.; Springer Nature Singapore: Singapore, 2026; pp. 627–636.
25. Feng, L.; Xie, R.; Tang, Q.; Huang, T.; Xiong, Z.; Chen, T.; Zhang, R.; Tan, S.; Fang, Z. CaRCS: Joint optimization of computing-aware routing and collaborative scheduling in computing power networks. *IEEE Network* **2025**, *39*, 270–278.
26. Zhang, Y.; Zhang, H.; Song, C. A hybrid algorithm for multi-objective task scheduling in heterogeneous cloud computing. *The Journal of Supercomputing* **2025**, *81*, 1143.
27. Malik, M.; Nandan, D.; Prabha, C.; Uddin, M.; Acharya, B.; Hu, Y.-C. A bio-inspired metaheuristic approach for cloud task scheduling using lateral hyena based particle swarm optimization. *Multimedia Tools and Applications* **2025**, *84*, 20023–20046.
28. Yu, X.; Mi, J.; Tang, L.; Long, L.; Qin, X. Dynamic multi objective task scheduling in cloud computing using reinforcement learning for energy and cost optimization. *Scientific Reports* **2025**, *15*, 45387.
29. Tian, B.; Xiao, G.; Shen, Y. A deep reinforcement learning approach for dynamic task scheduling of flight tests. *The Journal of Supercomputing* **2024**, *80*, 18761–18796.
30. Xiao, H.; Xu, C.; Ma, Y.; Yang, S.; Zhong, L.; Muntean, G.-M. Edge intelligence: A computational task offloading scheme for dependent IoT application. *IEEE Transactions on Wireless Communications* **2022**, *21*, 7222–7237.
31. Sun, Y.; Xu, J.; Cui, S. User association and resource allocation for MEC-enabled IoT networks. *IEEE Transactions on Wireless Communications* **2022**, *21*, 8051–8062.
32. Mas, L.; Vilaplana, J.; Mateo, J.; Solsona, F. A queuing theory model for fog computing. *The Journal of Supercomputing* **2022**, *78*, 11138–11155.
33. Cao, X.; Tang, G.; Guo, D.; Li, Y.; Zhang, W. Edge federation: Towards an integrated service provisioning model. *IEEE/ACM Transactions on Networking* **2020**, *28*, 1116–1129.
34. Sun, Z.; Mo, Y.; Yu, C. Graph-reinforcement-learning-based task offloading for multiaccess edge computing. *IEEE Internet of Things Journal* **2023**, *10*, 3138–3150.
35. Liu, G.; Deng, W.; Xie, X.; Huang, L.; Tang, H. Human-level control through directly trained deep spiking Q-networks. *IEEE Transactions on Cybernetics* **2023**, *53*, 7187–7198.
36. Floyd, R.W. Algorithm 97, shortest path algorithms. *Communications of the ACM* **1962**, *5*, 345.
37. Orlowski, S.; Pióro, M.; Tomaszewski, A.; Wessäly, R. SNDlib 1.0–survivable network design library. *Networks* **2010**, *55*, 276–286.
38. Nguyen, D.C.; Ding, M.; Pathirana, P.N.; Seneviratne, A.; Li, J.; Poor, H.V. Cooperative task offloading and block mining in blockchain-based edge computing with multi-agent deep reinforcement learning. *IEEE Transactions on Mobile Computing* **2023**, *22*, 2021–2037.
39. Laili, Y.; Wang, X.; Zhang, L.; Ren, L. DSAC-configured differential evolution for cloud–edge–device collaborative task scheduling. *IEEE Transactions on Industrial Informatics* **2024**, *20*, 1753–1763.