Online Decentralized Task Allocation Optimization for Edge Collaborative Networks

Yaqiang Zhang*†‡, Ruyang Li*† Yaqian Zhao*†, Rengang Li*†, Xuelei Li*†, Tuo Li*†
*Inspur (Beijing) Electronic Information Industry Co., Ltd, Beijing 100085, China
†Inspur Electronic Information Industry Co., Ltd, Jinan 250101, China
‡Shandong Massive Information Technology Research Institute, Jinan 250101, China
{zhangyaqiang, liruyang, zhaoyaqian}@inspur.com, lirengang.hsslab@gmail.com, {lixuelei, lituo}@inspur.com

2022 IEEE Symposium on Computers and Communications (ISCC) | 978-1-6654-9792-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISCC55528.2022.9912855

***Abstract*—In centralized task allocation strategies, real-time status information needs to be collected from distributed edge nodes. Therefore, the overloaded transmission on backbone network appears and leads to devastating decrease in the performance of centralized strategies. To address this issue, this paper proposes a multi-agent deep reinforcement learning based online decentralized task allocation mechanism, where each edge node makes task allocation decisions based on local network-state information. A centralized-training distributed-execution method is adopted to decrease data transmission load, and a value decomposition-based technique is applied at training stage for improving long-term performance of task allocation in edge collaborative networks. Extensive experiments are conducted, and evaluation results demonstrate that our mechanism outperforms other three baseline algorithms in reducing the long-term average system delay and improving request completion rate.**

***Index Terms*—Task Allocation; Multi-agent System; Online Decentralized Optimization; Value Decomposition; Edge Collaboration Networks.**

I. Introduction

With the development of new generation communication technologies, exponential growth of intelligent devices are connected, which aggravates the congestion of backbone network. Edge computing emerges to handle the relatively complex applications (also known as user requests) through providing computation, storage and content resources at the edge of network [1], [2]. For instance, Unmanned Aerial Vehicles(UAVs) are often deployed in harsh environments to collect real-time images in industrial Internet of Things (IIoT) scenarios [3], [4]. The images data uploaded from UAVs to the cloud will produce high latency and increase network load. While in edge computing, the data to be processed can be sent to edge nodes with much lower latency, without the need to go through the backbone network. Edge computing has many advantages over cloud computing, but there exist some problems to address. On one hand, as the number of edge computing nodes increases, it is challenging to effectively manage and utilize resources. On the other hand, the complex structure and concurrent nature of user requests, in which request can be represented as Directed Acyclic Graphs (DAG), also greatly increase the workload of edge networks. In addition, more requests are time sensitive, and aging data and processing cannot support the real-time user requests.

To address the above challenges, studies have explored resource allocation in edge networks based on centralized and decentralized strategies [5], [6]. In centralized strategy, requests are usually allocated to edge nodes based on the overall status information and task load information of the edge network. However, collecting and uploading network-state information to the central decision making server may increase the load on backbone network and cause outdated information problem in a distributed edge environment. In decentralized strategies, edge nodes have their own allocation policies. Compared with the centralized strategy, only local network-state information is needed to help allocating decisions, which effectively reduces transmission burden on backbone network. But this also makes it more difficult to improve the overall performance of edge network as the lack of information exchange among edge nodes. Therefore, restricted information sharing based on direct or indirect interaction among edge nodes to improve the efficiency of decentralized allocation decisions is a major approach.

In this paper, we focus on the task allocation optimization of concurrent DAG-based user requests in edge networks. An online decentralized task allocation mechanism is proposed and discussed in the following sections. Different from the static optimization mechanism where allocation decisions for each of tasks in a user request are usually generated at one time, the online task allocation mechanism is adopted, where the executable parts of user requests are scheduled. The main contributions of this paper are summarized as follows:

- We establish an online decentralized task allocation system, which enables each edge node to make task allocation decisions based on their network status information, and forms a cooperative relationship among edge nodes. A multi-objective optimization function is built to guide the task allocation mechanism to minimize the long-term average delay of edge networks and maximize the completion rate of user requests.
- We propose a multi-agent deep reinforcement learning-based mechanism for online decentralized task allocation optimization problem. The proposed mechanism runs in a Centralized-Training Distributed-Execution (CTDE) manner. In the centralized-training stage, value decomposition technology is adopted for improving long-term performance of task allocation decisions, while the implicit cooperation among edge nodes is realized in the

978-1-6654-9792-3/22/$31.00 ©2022 IEEE

distributed-execution stage.

II. System Model

A. Network Model

In this paper, the discussed edge network consists of N edge nodes, which is denoted as $\mathcal{N} = \{1, 2, ..., n, ..., N\}$. Each edge node is attached to a base station and use the communication network infrastructure between the base stations to connect with others. A discrete time system based on segmentation is established, and each time slot t has equal time duration. For edge node n, the CPU frequency at t is denoted as $\mathcal{F}_n^t$ and the symbol $\mathcal{C}_{nn'}^t$ denotes the network transmission capacity from n to n'.

B. User Request Model

Users within the network region can access edge nodes through wireless communication and in each time slot t, user requests are generated randomly. Different from an atomic request that can not be divided into smaller sub-tasks, a user request discussed in this paper consists of a set of V tasks and can be formulated as a DAG:

$$ur = G(\mathcal{V}, \mathcal{E}) \tag{1}$$

where $\mathcal{V} = \{1, 2, ..., v, ..., V\}$ denotes the set of tasks in user request ur and $\mathcal{E}$ represents the set of relations between tasks. For task v in the request, according to $\mathcal{E}$, if another task is associated with this task and is at the front end of the relation, it is represented as the parent node task of v. We use $\mathcal{L}_v = \{1, 2, ..., l, ..., L\}$ to represent the parent node task set of task v. For any task l in $\mathcal{L}_v$, D_l^v represents the intermediate result data generated by task l and required for the processing of task v. Naturally, tasks other than the start point need the result of their parent node tasks before they can be processed. TC_l^v represents the corresponding time constraint between two tasks. Time constraints reflect that if a subsequent task cannot start execution within the time constraints placed on it by the corresponding parent node task, it is considered outdated.

C. Task Allocation Model

In this paper, we assume that all tasks of a user request are uploaded and processing at edge networks, while the final results are sent back to the user over the communication network. Tasks in a DAG-based user request are executed concurrently or sequentially by edge nodes following the logical structure. For example, some structurally parallel tasks with a same parent node task can be processed by different edge nodes concurrently to reduce latency of task processing. In order to determine where the task to be processing, a global queue U_t at time slot t for the edge network is maintained to manage tasks from user requests that are ready to be scheduled. In each time slot, tasks are queued in order of their ready time point and followed by the First-In First-Out (FIFO) rule. There exist two situations for a task being queued: 1) The task has no parent node tasks in its request structure, i.e., it is the starting point; 2) All the parent node tasks of the task have been processed. Next, representation models of different stages of task allocation are introduced.

1) Task waiting stage: The queue-based request scheduling problem is addressed, where tasks are allocated to edge nodes in turns. If task v is currently being scheduled, the symbol $a_n^t = v$ means that edge node n will be scheduled to handle task v at time t's queue U_t. As mentioned above, all parent node tasks of v have been done, and we use $\{PE_l^v\}_{l \in \mathcal{L}_v}$ to represent the processing locations of these tasks. $PE_l^v = n'$ means that parent node task l is processed at edge node n'. We take it as an example to describe the total delay generated for the task v to be processed at edge node n.

2) Task transmission stage: Then the corresponding input data from its parents tasks transferred to the edge node n from their processing edge nodes according to $\{PE_l^v\}_{l \in \mathcal{L}_v}$. Taking parent node task l for example, the transmission time is represented as follows:

$$t_{trans}^{n'n} = hold_{trs}^{n'n} + t_{trs}^{n'n} \tag{2}$$

where $hold_{trs}^{n'n}$ represents the transmission channel waiting time from edge node n' to n and $t_{trs}^{n'n}$ represents the corresponding data transmission time. We give the formula for calculating $t_{trs}^{n'n}$ from n' to n as follows:

$$t_{trs}^{n'n} = \frac{D_l^v}{\mathcal{C}_{n'n}^t} \tag{3}$$

where $\mathcal{C}_{n'n}^t$ denote the network transmission capacity and D_l^v represents the input data size.

3) Task process stage: The amount of input data of the task determines the latency caused by the task being processed at an edge node. Assume that $\mathcal{F}_n^t$ denotes the CPU frequency of the edge node n at the current time, the corresponding calculation delay formulae are given below:

$$t_{cpu}^n = \frac{k \times D_{total}^v}{\mathcal{F}_n^t} \tag{4}$$

where k is the cycle number required to process a unit of data and D_{total}^v denotes the total amount of data entered by task v.

The delay in task processing stage for task v at edge node n is:

$$t_{proc}^v = hold_{cpu}^n + t_{cpu}^n \tag{5}$$

where $hold_{cpu}^n$ denotes the processor task queue wait time. The $hold_{cpu}^n = 0$ if the CPU is free before the end of data transmission, else $hold_{cpu}^n$ is the time between the end time of data transmission and the most recent free time point of CPU.

4) Total delay: The total delay for processing task v at time t can be represented as follows:

$$t_n^v = t_{wait}^v + Latest\{t_{trans}^{n'n}\}_{n' \in \{PE_l^v\}_{l \in \mathcal{L}_v}} + t_{proc}^v \tag{6}$$

where $Latest\{t_{trans}^{n'n}\}_{n' \in \{PE_l^v\}_{l \in \mathcal{L}_v}}$ represents the latest time to complete the data transmission of its parent node tasks.

At time t, the total delay generated by tasks allocation in edge networks can be expressed as follows:

$$DL_{total}^t = \sum_{n=1}^{N} t_n^v \tag{7}$$

where N is the number of edge nodes.

III. Online Decentralized Task Allocation

A. Problem Formulation

In last section, we discuss the delays that occur at different stages of the task scheduling process. In order to optimize the long-term average delay of the system and improve the completion rate of user requests, we establish the corresponding multi-objective optimization function of the proposed task allocation optimizing problem as follows:

$$\textbf{P1:} \begin{cases} \min\limits_{T\to\infty} & \frac{1}{T}\sum_{t=0}^{T} DL_{total}^{t} \\ \\ \max\limits_{T\to\infty} & \eta \end{cases} \tag{8}$$

where T represents the maximum time slot. The first target function proposes to minimize the long term system delay and the second is to maximize the proportion η of tasks that satisfy their time constraints. For dynamic optimization problems with multiple objective functions, it is difficult to obtain effective solutions based on traditional approaches. Therefore, this paper uses Reinforcement Learning(RL) to solve dynamic optimization problems and obtains effective decentralized task allocation strategies based on policy learning.

B. Multi-Agent Deep Reinforcement Learning

In recent years, due to the emergence of distributed application scenarios, decentralized decision-making system have been proved to be effective, among which the idea of multi-agent reinforcement learning has been widely used [7]. An independent multi-agent method directly applies single-agent RL algorithm to multi-agent environment, in which each agent does not communicate with others and takes the part of environment observation as its state. In this kind of works, the non-stationary problem of environment caused by local observation is difficult to solve. As an improvement, the communication-based approach is proposed, in which each agent can observe not only their own environmental state, but also the environmental information of other agents through information exchange, so as to increase the observation ability for the environment and improve the performance of decision-making. The biggest issue is that it needs to obtain the environmental state information of other agents, so there will be a large communication cost, and it is difficult to effectively implement in some real-time application scenarios. A combination of above methods called Centralized-Training Distributed-Execution (CTDE) is proposed, in which the training process takes the observation of all agents as input, while only the local environmental information observed by each agent is needed during the decision-making. The advantage of CTDE model is that it does not need explicit information transmission among agents, but realizes the cooperation among agents by means of centralized-training, which greatly reduces the communication overhead and improves scalability for distributed decision-making problems. In this section, a CTDE-based multi-agent reinforcement learning method for online task allocation problem is proposed.

C. Reinforcement Learning Environments Description

We define each edge node as an agent in sense of multi-agent reinforcement learning. The key elements for description of multi-agent system are shown as follows:

1) Agent: In this paper, each edge node is treated as a different agent. For each agent in $\mathcal{N}$, its own status information can be timely monitored and obtained, and the network link state information with other agents as well as the global task queue information can be obtained. Then the agent makes task allocation decision for one in the queue U_t according to its observation h_n^t at time slot t.

2) System State: The joint system states observed by each agent at time slot t can be represented as:

$$S_t = \{h_1^t, h_2^t, ..., h_N^t\} \tag{9}$$

$$h_n^t = \{\mathcal{F}_n^t, \{\mathcal{C}_{nn'}^t\}_{n'\in\mathcal{N}}, U_t\}, n \in \mathcal{N} \tag{10}$$

where $\mathcal{F}_n^t$ denotes the CPU frequency of edge node n at time t and $\mathcal{C}_{nn'}^t$ indicates the network transmission capacity from edge node n to n'. Besides, U_t saves the tasks information that are ready for execution and will be updated in the beginning of each time slot. At the same time, the observed system states by each agent are updated.

3) Actions: An action mentioned in this paper is the task allocation decision according to the system states at time t. The joint action of each agent is represented as:

$$A_t = \{a_1^t, a_2^t, ..., a_N^t\} \tag{11}$$

where a_n^t denotes the task in U_t that is assigned to EN_n at time t. At each decision phase, the first N tasks are scheduled in priority and each edge node could just choose one task in U_t.

4) Reward: Reward reflects the quality of joint actions in current time slot, and always keeps pace with the optimization of objective function. Based on the ultimate goal of minimizing the long-term average delay in edge network and improving the QoE of users, we define the reward R at time t as:

$$\begin{aligned} R_t &= r_1^t + r_2^t + ... + r_N^t + k_t \times pr \\ &= (DL_1^t + DL_2^t + ... + DL_N^t) + k_t \times pr \\ &= DL_{total}^t + k_t \times pr \end{aligned} \tag{12}$$

where r_n^t denotes the reward of each agent at time slot t, pr is a punishment reward counted when a task delay is higher than its time constraints and k is the count number of pr in time t. The reward R_t represents the delay caused by edge nodes at time slot t, and it is part of the objective function. In addition, if all tasks are completed in their time constraints, the part of $k_t \times pr$ is zero, otherwise the reward would be a large value in this setting.

D. Value Decomposition-based Multi-Agent DQN (VD-MADQN)

The structure of the proposed CTDE-based multi-agent deep reinforcement learning system is shown in Fig. 1. We in this paper apply the famous Deep Q-Network(DQN) [7].

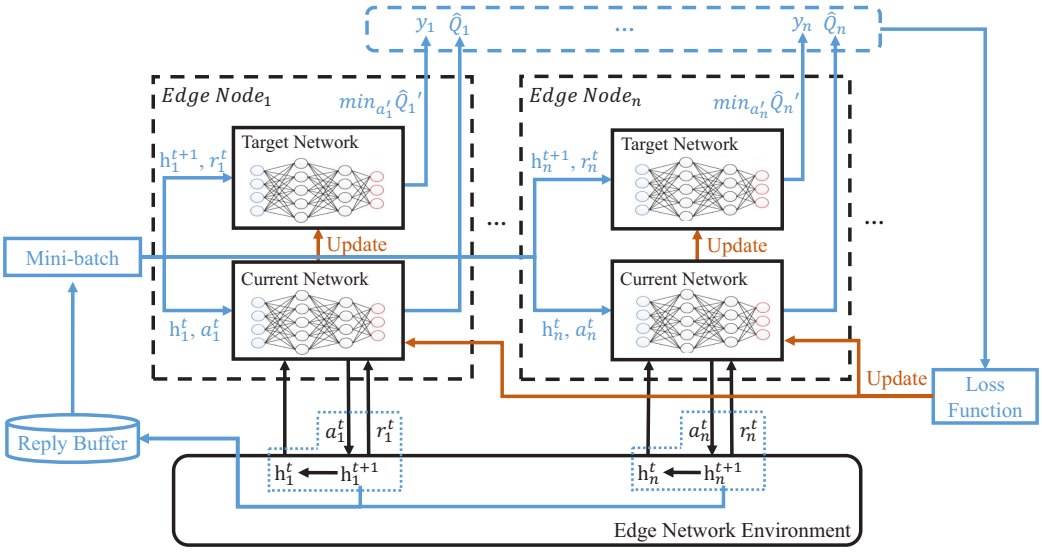

Fig. 1. CTDE-based multi-agent deep reinforcement learning structure.

In the centralized training stage, while each agent would obtain its observation as the local state information, the task scheduling strategy learning is carried out through information sharing among agents in this stage to achieve the goal of global optimization. In the decentralized execution stage, the agent does not need to share information, but can schedule tasks in the queue U_t based on local observation. The solid blue line represents the data exchange between the agent and the training server during centralized training, and the blue wire frame represents the components located on the training server. Black dotted boxes represent each edge node, the agent. For $EdgeNode_n(Agent_n)$, the decision module consists of two deep neural networks, the target network and the current network. The two networks have the same network structure and initialization network parameters ω_n and ω_n'. At time slot t, the observed local state information h_n^t is entered into the current network of $Agent_n$, and the network outputs the current action a_n^t and interacts with the environment. Then $Agent_n$ gains immediate rewards r_n^t and observes local environmental changes as h_n^{t+1}. So far, agents have only interacted with the environment and output decision actions based on their own observations. In order to utilize the cooperation between edge nodes to achieve global objective optimization, information sharing between agents is proposed with the idea of Value Decomposition(VD) manner in the centralized training stage.

The basic idea of VD is to decompose the team value function into agent-wise value functions as proposed in [10]. That is, the overall strategy of the system can be evaluated by the sum of the benefits achieved by each agent strategy. Based on the above assumptions, the information sharing among agents is achieved. The Reply Buffer is built in the training server to store information generated when an agent interacts with the environment. As shown in the blue dotted box at the bottom of the figure, the agent transmits the corresponding data to the training server. Then when the number of records in the reply buffer exceeds a certain size, the training server randomly samples a batch of data from the reply buffer and send them to each agent. Among them, h_n^t and a_n^t are sent to current network of $Agent_n$, while h_n^{t+1} and r_n^t are sent to the target network. The output of two networks are target Q-value y_n and current Q-value $\hat{Q}_n$, and then transmit to the training server. Based on the data, the team Q-value can be represented as the approximate sum of Q-value of each agent:

$$\begin{aligned} Q(S_t, A_t) =& Q_1(h_1^t, a_1^t) + Q_2(h_2^t, a_2^t) + ... + Q_N(h_N^t, a_N^t) \\ \approx& \hat{Q}_1(h_1^t, a_1^t, \omega_1) + \hat{Q}_2(h_2^t, a_2^t, \omega_2) + ... \\ &+ \hat{Q}_N(h_N^t, a_N^t, \omega_N) \end{aligned} \tag{13}$$

where $Q_1(h_n^t, a_n^t)$ denotes the real Q-value of $Agent_n$ under h_n^t and a_n^t. The team Q-value function is replaced by a linear sum of individual Q-values, which means each agent would greedily choose the action to minimize the current team Q-value. The target team Q-value is calculated as the sum of each agent:

$$Y_m = \sum_{n=1}^{N} y_{m,n} \tag{14}$$

where the target Q-value of each agent can be calculated by target network $\hat{Q}_n'(\omega_n')$ of each agent as follows:

$$y_{m,n} = r_{m,n}^t + \gamma min_{a'} \hat{Q}_n'(h_{m,n}^{t+1}, a_{m,n}', \omega_n') \tag{15}$$

Then the team loss function L is calculated as follows:

$$L = \frac{1}{\mathcal{M}} \sum_{m=1}^{\mathcal{M}} (Y_m - Q(S_m, A_m))^2 \tag{16}$$

and the parameter ω of current network in each agent is updated based on the loss function using gradient descent technology. Every once in a while, the target network shares parameter updates of the current network.

Once the centralized training stage is completed, the trained scheduling decision networks for each agent are deployed in each edge node. At the distributed execution stage, each agent does not need to exchange information with other agents, and the scheduling decision of tasks are generated in each agent by considering the local observation as the input of the scheduling decision network. The advantage of the proposed mechanism is that the sum of the local reward of each agent can be used to replace the rewards of the whole system, and the sum processing can be performed in a linear or nonlinear way. In this way, each agent can realize the optimization of team strategy and payoff by obtaining the optimization of its own strategy and payoff, thus avoiding the lazy agent problem caused by the agent's inability to determine whether the team reward is positively related to its decision, and the increase in the size of optimization problem.

Algorithm 1 VD-MADQN training procedures

1: Initialize the network parameters of $\hat{Q}_n$ and $\hat{Q}'_n$ of each agent n. Initialize ϵ, γ, c, $\mathcal{M}$, D.
2: **for** e = 1 to $episode$ **do**
3: **for** s = 1 to $step$ **do**
4: For each agent n, takes current state h_N^t as the input of $\hat{Q}_n$ and select an output action a_n according to ϵ-greedy strategy;
5: Execute each action a_n, get the reward r_n^t and store $\{(h_n^t, a_n, r_n^t, h_n^{t\prime})\}$ in D;
6: Randomly sample $\mathcal{M}$ records from D when $|D| \geq \mathcal{M}$;
7: Calculate Equations (13) and (14);
8: Update Q_n of each agent by minimizing the loss function in equation (16);
9: every c times: $\omega'_n = \omega_n$.
10: **end for**
11: **end for**

The overall centralized training procedures of the proposed mechanism is shown in Algorithm 1. At the initialization phase, the current network parameter ω_n of each agent n is randomly set and ω'_n of target network is the same as ω_n. Initialize reply buffer D, batch size m, a frequency of parameter copy c, the initialized ϵ and γ. (Lines 1). For each agent, the observed system state h_n^t at step s is put into $\hat{Q}_n$ and an action with the minimal Q-value is chosen according to ϵ-greedy strategy. Then execute each a_n, and the system transfers to a new state $h_n^{t\prime}$ while an immediately reward is calculated by r_n^t. Each tuple of $(h_n^t, a_n, r_n^t, h_n^{t\prime})$ is stored to D (Lines 4-5). While the number of elements in D is larger than the batch size m, it randomly sample m records from D and calculate the team Q-value $Q(S_t, A_t)$ and target team value Y_i according to Equations (13) and (14) (Lines 7). Then it updates parameters ω_n for each agent according to Equation (16), in which a mini-batch SGD is applied (Lines 8). Every c times it replaces the ω'_n with ω_n.

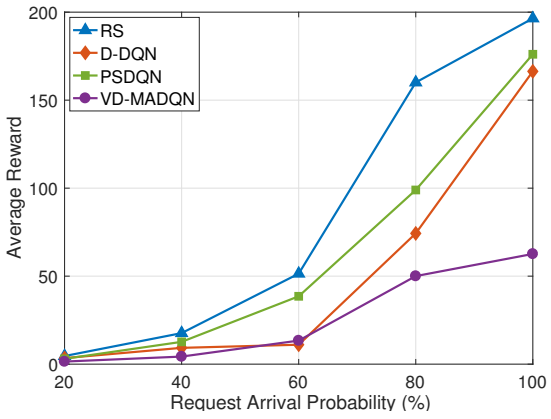

Fig. 2. The long-term average reward under different request arrival probability.

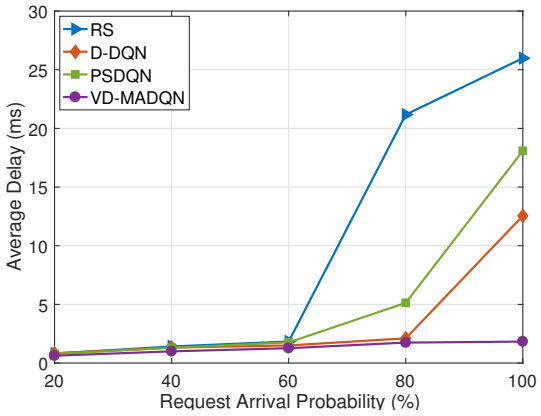

Fig. 3. The long-term average delay under different request arrival probability.

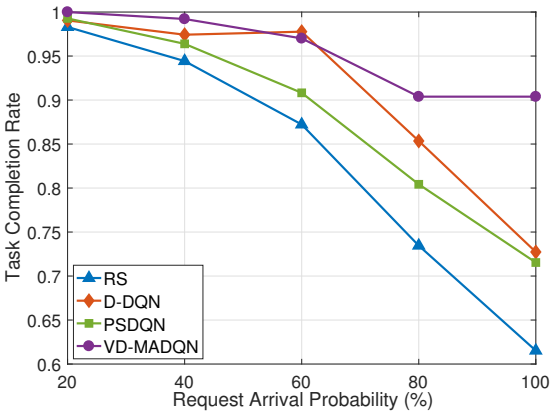

Fig. 4. The task completion rate under different request arrival probability.

IV. Implementation and Evaluation

A. Experimental Settings

In this section, we conducted a numerical simulation to investigate the performance of the proposed mechanism and a prototype was implemented by a Python program. To simulate

the real world scenario, we adopt a popular open source data set EUA [8], which includes the public base station and user information in a certain area. We use part of data to construct an edge network system, including 11 edge nodes and multiple mobile users. The CPU frequency of different edge node is set from 1GHz to 4GHz, and the average channel data transmission capacity among edge nodes is set to 1GBytes per second. User Requests are generated based on a face recognition model provided by [9], where each request is composed by multiple dependent tasks with specified logical relations. And the data size of each task range from 2MB to 3MB. The user request arrival probability is set between 20% and 100%, where 100% means that a user request must be generated at a time slot. Based on the above assumptions, we also compare the proposed mechanism with others based on distributed and centralized decision-making. A brief introduction of above mentioned mechanisms are listed below:

- **VD-MADQN:** The proposed Value Decomposition-based Multi-Agent DQN mechanism for tasks allocation following the CTDE manner.
- **D-DQN:** A Distributed DQN-based mechanism where each edge node runs a separate decision network and can only output decisions based on partially observations about the environment [11].
- **PSDQN:** A Parameter Sharing DQN-based allocation mechanism mentioned in [12]. All edge nodes share a common decision network and take their local observations as system inputs in training stage. The reward includes the total revenue generated by all edge nodes at each decision step.
- **RS:** Selects an available scheduling decision based on Random Strategy in each steps.

B. Experimental Results

Evaluation results show that the proposed mechanism performance better than other mechanisms under different settings. For each mechanisms, we set the maximum number of training episodes for each mechanism to 1000 and the maximum number of iteration steps per episode to 200. Results in Fig. 2 to 4 show the trend comparison of the system performance of each mechanism with different probability of request generation, where the number of edge nodes is set to 11. In Fig. 2, with the increase of the request arrival probability, the average system reward curves show a growing trend, especially when the load rate is greater than 60%, the curves have significant rise. According to Fig. 3 and Fig. 4, when other curves increase significantly after a certain request arrival probability, the proposed mechanism can keep a relative stable average system delay under different system loads, and its task completion rate is significantly higher than other mechanisms, especially at high loads. It can be analyzed from the results that the learning-based mechanisms performance better than that of the baseline algorithm in most cases, due to the learning of environment state, accompanied by policies optimization. While in the proposed mechanism, at the policy learning phase, each agent can sense the influence of their own decision-making on group benefits, so as to learn better policies. Therefore, compared with the fully distributed multi-agent allocation mechanism and common decision network-based mechanism, the value decomposition-based mechanism can choose better task allocation actions under distributed environment, to reduce the long-term average system delay and increase the average completion rate of tasks.

V. Conclusion

This paper proposes a multi-agent deep reinforcement learning based online decentralized task allocation mechanism in edge collaborative networks. An intelligent decentralized allocation decision strategy is adopted, where a CTDE-based framework is built for each edge node. At centralized-training stage, value decomposition technique is adopted to realize implicit cooperative decision policy training among edge nodes. At distributed-execution stage, local network-status information can be used to generate task allocation decisions that can meet the global optimization objective. Results show that the proposed mechanism can significantly improve the task allocation performance of edge collaborative networks.

Acknowledgment

This work was supported by Shandong Provincial Natural Science Foundation under Grant ZR2021QF104.

References

[1] Y. Liu, M. Peng, G. Shou, Y. Chen, and S. Chen, "Towards edge intelligence: Multi-access edge computing for 5g and internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6722–6747, 2020.

[2] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 99, pp. 869–904, 2020.

[3] X. Hu, Y. Guo, J. Hao and C. Li, "Co-deployment of UAV and Ground Base Station for Data Collection in Wireless Sensor Networks," in *2020 IEEE International Conference on Communications (ICC)*, 2020, pp.1–6.

[4] W. Guo, "Partially Explainable Big Data Driven Deep Reinforcement Learning for Green 5G UAV," in *2020 IEEE International Conference on Communications (ICC)*, 2020, pp.1–7.

[5] A. Feriani and E. Hossain, "Single and multi-agent deep reinforcement learning for ai-enabled wireless networks: A tutorial," *IEEE Communications Surveys and Tutorials*, vol. 23, no. 2, pp. 1226–1252, 2021.

[6] Y. Zhan, S. Guo, P. Li, and J. Zhang, "A deep reinforcement learning based offloading game in edge computing," *IEEE Transactions on Computers*, vol. 69, no. 6, pp. 883–893, 2020.

[7] R. Sutton and A. Barto, *Reinforcement Learning:An Introduction*, 1998.

[8] P. Lai, Q. He, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang "Optimal Edge User Allocation in Edge Computing with Variable Sized Vector Bin Packing," in *16th International Conference on Service-Oriented Computing (ICSOC2018)*, 2018, pp. 230–245.

[9] P. Hu, H. Ning, Z. Wang, T. Qiu, H. Song, Y. Wang, and X. Yao "Security and Privacy Preservation Scheme of Face Identification and Resolution Framework Using Fog Computing in Internet of Things," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1143–1155, 2017.

[10] S. Peter, L. Guy, G. Audrunas, .et al, Value-decomposition networks for cooperative multi-agent learning, CoRR abs/1706.05296 (2017).

[11] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, R. Vicente, Multiagent cooperation and competition with deep reinforcement learning, Plos One 12 (4) (2017) 1–15.

[12] K. G. Jayesh, E. Maxim, J. K. Mykel, Cooperative multi-agent control using deep reinforcement learning, in: Autonomous Agents and Multiagent Systems - AAMAS 2017 Workshops, Best Papers, São Paulo, Brazil, May 8-12, 2017, Revised Selected Papers, Vol. 10642, 2017, pp. 66–83.