dispersed resource scheduling and perception-dependent dynamic task decision making based on deep reinforcement learning

Haoran L[1]

***Abstract*— dispersed computing is a new resource-centered computing paradigm that utilizes idle resources in the network to accomplish tasks. Efficiently distributing tasks between an application's task nodes and network compute points (NCPS) is a key factor in maximizing dispersed computing performance. However, due to the heterogeneity of nodes and priority requirements of tasks, and to find suitable NCP nodes in the vast dispersed computing network, the task allocation process faces great challenges. Most current research on dispersed computing ignores the dependencies between sub-task nodes, resulting in these sub-tasks not being executed in parallel, resulting in non-trivial resource waste when allocating resources to each sub-task throughout the life cycle of the application.**

In this paper, the resource allocation problem of multi-component tasks in a network is studied, with the goal of minimizing the energy time cost (ETC) of NCP and achieving load balancing. In order to effectively solve this complex decision problem, we introduce the techniques of graph convolutional neural network (GCN) and reinforcement learning (RL), and propose a task assignment model based on incomplete preference list, through which the requirements and permissions of task nodes and NCP are quantitatively measured. Directed acyclic graph (DAG) is used to model the interdependencies between tasks, and the DAG structure is characterized by GCN. The actor-critic method is combined with the embedding layer, and the composite action actor-critic(CA2C) algorithm is used for training. In this way, task resources can be allocated more efficiently and resource waste can be reduced, thus improving the overall performance of dispersed computing systems.

I. INTRODUCTION

dispersed computing architecture is a completely dispersed and mobile computing paradigm that utilizes geographically distributed hardware to share diverse and flexible idle resources [1]. Unlike traditional edge and cloud computing environments, dispersed computing is not entirely dependent on the data center. Instead, it expands the use of wireless network devices such as servers, base stations, personal computers, smartphones, and smart devices to become more than just transmission nodes, but also network compute points (NCPS) for real-time connections. [2] dispersed computing has a unique advantage over other computing paradigms (cloud, fog, and edge), namely the flexibility to select the most convenient mobile device for computing [3][4][5], and this flexibility depends on efficient task allocation in dispersed computing.

*This work was not supported by any organization

[1]H. Kwakernaak is with Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, 7500 AE Enschede, The Netherlands h.kwakernaak at papercept.net

[2]P. Misra is with the Department of Electrical Engineering, Wright State University, Dayton, OH 45435, USA p.misra at ieee.org

However, running these computations over a dispersed network requires new resource allocation and application placement algorithms that can address several unique challenges: Constrained network connections, heterogeneous available resources, and dynamic network conditions caused by failures or device mobility between discrete nodes in a dispersed computing network are in stark contrast to servers located in well-connected data centers. Due to the highly dynamic and unstable nature of dispersed computing environments (such as the heterogeneity of computing power and communication bandwidth), the task allocation of dispersed computing faces many challenges. Among them, one of the most important challenges in task allocation is how to quickly identify and deal with instantaneous changes in the network.

With the advent of the 5G era, people are demanding higher and higher service quality. Compute-intensive applications with high computing resource requirements need to be completed in a short period of time. Until now, stream processing applications have typically been deployed in large-scale and centralized cloud environments, such as data centers. However, in addition to the growing volume of data, enhancements to end devices and the deployment of edge services are now moving computing to the edge of the network. Fog/edge computing [6] is envisioned as a promising and viable technology to ease the pressure on computing and communication. By migrating computing from the cloud to edge servers, it can provide low-latency and context-aware services for iot devices. However, in real-world scenarios [7], realistically challenging situations (e.g. traffic jams, natural disasters) are common. The fixed infrastructure used for communication or computing is often overloaded or even unavailable, making it impossible to perform computing tasks efficiently in the corresponding area.

In general, applications running on dispersed systems can be broken down into tasks with priority requirements, captured by directed acyclic graphs (DAGs). A number of research efforts [3] and industry frameworks, such as ApacheAirflow and Google Airflow Composer, have focused on the scheduling and deployment of any DAG, enabling dispersed computing systems to run generally complex applications efficiently. To evaluate a dispersed compute scheduler, impact metrics include: resource consumption, makespan, and throughput. Makespan is highly correlated with throughput. Makespan is defined as the end-to-end task DAG completion time for an input data set.

In this paper, GCN is combined with multi-agent actor Critic DRL, and a composite action space model is estab-

lished for power allocation and partial decision application of NCP. Specifically, the combination of DRL and GCN is used to derive intelligent programs to solve problems. Each agent utilizes intelligent programs for decision optimization and UL power control. Network of intelligent programs, actors and critics, with information on the current state of the environment. Previously, DRL has been used to solve the problem of partially deciding complex applications, and GCN is used to capture the characteristics of the task and the structure of the application. In this work,GCN reinforces the derived DRL model to capture the characteristics and dependencies of tasks in the application, not only to determine the decision tasks, but also to adjust the transmit power level of the NCP.

II. RELATED WORK

A. dispersed computing

Over the past few years, many researchers have delved into the problem of resource allocation in the field of distributed computing. The challenge of task offloading in cloud, edge, and fog computing environments with resource allocation requirements has attracted a lot of attention in the wireless community. However, less research has focused on dispersed computing. With the improvement of the computing power of intelligent devices, the concept of dispersed computing has gradually received attention [4][5][9].

dispersed computing [8] is considered as a computing paradigm that fully explores and utilizes idle resources in the network, and can be used as a supplement to provide services for users when edge devices are overloaded. For example, in [9], the researchers developed a run-time scheduling software for dispersed computing that is capable of deploying pipeline computing in the form of directed acyclic graphs on multiple geographically dispersed compute nodes. Ghosh et al. [10] designed a container orchestration architecture for dispersed computing that automatically and efficiently distributes tasks among a set of network compute nodes. Wu et al. [11] proposed a distributed computing offloading framework that considered user interests and network computing points, and made full use of idle and geographically dispersed computing resources through task offloading, contributing to energy saving of mobile devices. These studies take full advantage of the potential idle computing resources in the network, but have not yet well addressed the problem of incentive schemes to encourage the participation of individual devices.

B. Dependent task

In recent years, the rapid development of the Internet of Things has led to the emergence of unprecedented complex applications. These applications are usually composed of interdependent subtasks. In the case of task dependency, efficient resource scheduling and task unloading become a research focus. The above-mentioned research methods rarely make resource matching based on tasks with dependency, but this paper will add task dependency for decision-making on this basis. Rahimzadeh et al. [12] designed a polynomial-time resource allocation method for stream processing applications. Hu and Krishnamachari[13] developed a throughput optimized task scheduler to avoid the problem of low bandwidth connections when balancing the load. Zhou et al. [14] established a task resource joint management model of dispersed computing, which theoretically verified the existence and stability of the positive periodic solution of the model to ensure the efficiency and stability of the system. Yang et al. [15] proposed a maximum weight task scheduling strategy and proved its throughput optimality for dispersed computing networks. Currently, researchers are exploring various ways to address the task dependency challenge in mobile edge computing (MEC) to improve system performance, and task parallelism at the same dependency level is a potential area for future research.

C. deep reinforcement learning

However, considering the heterogeneity and dynamics of network resources, the joint problem of task division and resource allocation in multi-user multi-NCP systems may be very complicated. In addition, due to the asymmetry of network information, users cannot make the optimal unloading decision, which also promotes the research of task decision problem based on reinforcement learning. Literature [16] proposes a mobile unloading scheme based on deep reinforcement learning (DRL), which considers joint task unloading and resource allocation in a foggy computing network. Each fog node collaborates to maximize local rewards based only on local observations, so they formulate this problem as a partially observable random game and apply a deep cyclic Q network to approximate the optimal offload value function. Reference [17] studies non-orthogonal multiple access (NOMA) assisted multi-task computing offloading in multi-access MEC. Considering time-varying channel power gain, a DRL-based online algorithm is proposed to learn the optimal NOMA transmission duration to minimize the total energy consumption of local devices for task execution under delay constraints. In reference [18], a system consisting of multiple mobile devices (MD) and edge devices is studied, and the non-cooperative unloading game is considered. The goal is to design an optimal offload strategy that minimizes task drop rates and execution delays without prior knowledge of known task arrival models and channel models. Literature [19] proposes a gradient algorithm based on multi-agent depth deterministic strategy for task unloading and multi-channel access problems. Each user makes unloading decisions and decides whether the task is executed locally or unloaded to the cloud server by selecting a wireless channel. Literature [20] proposes a new task offloading algorithm based on imitation learning, in which each MD can imitate the task offloading strategy of the expert without knowing the global information. Recently, DRL algorithms based on actor-critic learning structures have also been proposed to reduce energy and time overhead, as shown in reference [21][22][23].

Taking into account the above factors and the advantages of the actor-critic network [23][24], a DRL algorithm based

on the actor-critic learning structure is adopted to optimize energy efficiency costs by taking into account energy consumption and delay. In addition, a time-slot mechanism is introduced, by setting the return time of the last executed task at the edge of each round, and adjusting the CPU frequency of the mobile device for local calculations. Each device will prepare the computing resources of the edge server for each time slot, thereby increasing the overall computing power of the system.

III. SYSTEM ARCHITECTURE

A. dispersed computing network model

Model a decentralized compute NCP network resource where the compute nodes (i.e., the NCP in the set) are the vertices and the connections (in the set) are the edges of the network graph. The computing power of each compute node including the computing power of this NCP (for example, CPU cycles per second or Hz) is expressed as; The capacity of a given resource type on NCPn is represented as the communication capability of each link including the communication capability of that link (such as link bandwidth), represented by the link. The placement of all computationally intensive dependent tasks of the application on the NCP and all task outputs on the corresponding link of the computing network is called a task assignment path. An application may have multiple paths. Consider the existence of multiple applications. Applications arrive over time and should be provided with one/more task assignment paths if they meet their QOE requirements, otherwise rejected.

Model multiple applications with multi-dependent subtasks, representing a list of all application sets as. For each application u, its task graph can be represented as a directed acyclic graph (DAG). As mentioned above, given the =(,) of an application, where is a set of tasks that represent dependencies between subtasks of the application. For each task of the application, use is used to represent the characteristics of the task. Thus, the characteristics of all tasks in the application can be represented as adjacency relationships for each task using the adjacency matrix An. The state of the task can be expressed as. The priority of the task, the amount of work, the size of the processed data, and the size of the amount of data returned will be given in the form of a tuple (,,,). The dependencies for the task graph will be given in the directed acyclic graph, where the vertices represent the information of the task and the edges between the nodes represent the amount of data passed between the tasks.

In task assignment, deal with the location of application dependent subtasks on the NCP and find a task assignment path for an application. In resource allocation, multiple applications are processed. Discover the number of task allocation paths required for each application and the amount of compute/communication resources allocated to each application to meet its QOE, taking into account its priorities.

B. communication model

In each iteration, the channel bandwidth occupied by each mobile device depends on the number of devices performed for slot t matching. Where represents the channel gain between the NTH NCP and n NCP in slot t. The mobility of the device is also considered. Assuming that the service area is fixed and the mobile device remains in it, the channel gain varies with the time slot. Under the block fading hypothesis, the channel gain remains constant for a short period of time while transmitting a computationally intensive task, but it varies for different task executions. In each slot, it will be expressed as the transmitted power of the task between the two NCP devices, and the channel noise power density. Based on Shannon's theorem, the channel transmission rate is as follows:

C. computational model

ku is an effective switching capacitor, which is based on the chip architecture. Make the task ahead. By formal expressions of preparation time and completion time, the preparation time and completion time of execution are given by (4) and (5), respectively, as follows, in (4), is the set of previous tasks whose corresponding nodes in the application DAG have edges pointing to the corresponding nodes. In addition, FT is the completion time of task u given in (5).

D. Task resource allocation preference list model

Task nodes and NCPs publish information based on their actual needs, including additional conditions with preferences. Then, based on the published information, each node can calculate the priority order of its partners, which is called the preference list. The task assignment process is about time continuity, and even if there are unfinished tasks in one batch assignment, they are automatically returned to the next batch assignment. The task assignment model needs to meet the following characteristics: 1) Task nodes can add deadlines and completion progress as requirements for each task. 2) The task node will reject NCP whose distance exceeds its allowable range and whose computing power does not meet its requirements. 3) NCP will reject task nodes whose computing power does not meet its requirements

Next, we build a preference list between the task nodes and the NCP, and then build a task assignment model based on the list. According to the preference function, the task prefers the larger the bandwidth, the more quota vectors, and the shorter the distance NCP. q is the number of tasks that NCP has accepted. NCP prefers tasks that require fewer CPU cycles.

Task preference list generation: First, each task selects NCP in the delay constraint. Then, based on the task preference function of the equation, the preference values of the selected NCP are calculated and arranged in descending order. Preference function is the key of the algorithm, it directly affects the final task assignment, and selects the top NCP in the task preference value list L as the current decision. At the same time, a record is made locally and sent to the appropriate NCP for feedback. Based on the above, the

program makes optimal task allocation decisions to minimize energy consumption. NCP preference list generation stage and match evaluation stage. The NCP preference list is generated, and the NCP will receive match requests from the program. The NCP receiving the request first eliminates the task that consumes more than its energy constraint, and then, each NCP calculates the preferred value of the remaining task according to the equation. And arrange them in descending order. Note that the preference function calculation here is an important step designed to help NCP prioritize tasks that are less computationally intensive and energy efficient.

E. problem definition

Based on a derived form expression of the completion time of the task executed by NCPn, the transmit power level of the task transmitted by NCP, the workload of the task, and the data forwarded from NCPn to other NCPs can cause significant delays in executing computationally intensive applications (generated by NCPn). On the other hand, NCPn has limited computational and energy resources for complex compute-intensive applications. These prompt a trade-off between latency and NCP energy consumption to perform compute-intensive applications. Therefore, this work aims to simultaneously minimize the weighted sum of NCP long-term delay and energy consumption, and in order to achieve load balancing, it is necessary to distribute tasks fairly among resources and avoid resource overload or idling.

DVFS(Dynamic Voltage and Frequency Scaling) is a technique used to adjust the power supply voltage and clock frequency of a processor or other digital circuit. By varying voltage and frequency, DVFS can optimize energy efficiency and performance under different workloads. The transmit power is determined by the amount of task output data, and the CPU calculation power is determined by the amount of task data.

At a given time t, let Etn represent the cumulative energy consumption of NCPn. In general, for NCPn, the energy consumption is given according to the computational energy and communication energy. On the other hand, when considering the average transmission delay in the network, it will correspond to the average execution time of the application, which is essentially the completion time of its final task. In other words, the delay in executing a compute-intensive application generated by NCPn corresponds to the completion time of the virtual exit task. The completion time of a task is the sum of its ready time and the time required to execute its workload. According to these equations, the ready time is directly related to the transmission delay between the NCP and the processor (other NCP). As a result, the completion time of an application is intricately linked to the transmission delay, making it possible to express the average transmission delay experienced in the system in terms of the average completion time of the application execution. This facilitates minimizing the energy time cost per NCP while adhering to the energy constraints of the NCP and the topology of the application generated by the NCP.

Energy consumption and time delay, energy consumption is composed of computing energy consumption and transmission energy consumption, time delay, by the application of the actual completion time and the actual preparation time subtraction, for the load can also be obtained by the following equation.

IV. METHOD

The algorithm has three main modules: 1) the actor module, which receives input parameters and generates a set of potential matching actions; 2) Identify the critic module of the optimal matching action; 3) and the policy update module, which updates the state of the system after the matching action is performed. These modules function in a sequential and iterative manner.

First, a framework based on DRL and GNN is proposed to obtain an approximate optimal matching strategy for dependent tasks in multiple applications with multiple NCP scenarios. At each decision time t, the scheduling agent observes the state of the NCP network system and the state of the current decision task S to select a matching action and receive a reward based on the custom goal. In this article, the GNN layer is used to get the embeddings of the tasks to be determined at time t, and the MLP layer is used to get the embeddings of the current NCP system. The agent uses NCP embeddings and task embeddings for the policy network, which outputs actions. Performing the action At in the current state St produces a new state St+1.

A. State embedding

The NCP environment state is based on the calculation of the task and energy overhead, and is only related to the characteristics of the NCP and the program state. To extract enough information, first embed the NCP environment state and the program information that needs to be determined separately.

NCP network embedding: Each time the information of the NCP environment is extracted, matching decisions are made on the subtasks of the current program. The NCP environment status information consists of two parts: NCP status and program status. For NCPs, CPU computing power, whether the current program is directly connected to, bandwidth, and the NCP of the task waiting queue need to make a decision. Similarly, if the task is assigned to another NCP, the channel information should be taken into account. Therefore, the current overall channel gain of the corresponding NCP also belongs to the NCP network environment state information. In addition, program information also has a significant impact on matching decisions. The maximum completion time of the task for which the current program makes a decision, the computing power of the device required by the program, the bandwidth between the current program and the corresponding NCP, and the estimated completion time of the task performed by the current program constitute the program information. The NCP information and the program information constitute the necessary information for the NCP environment. Features extracted from the NCP and

the program are fed into the MLP to learn the embeddings of the entire NCP environment.

Task embedding: Taking into account the dependencies between tasks, use GNN networks to effectively capture the overall structure of the application and better extract the information needed to make matching decisions for tasks. Based on GCN, for task embedding, embeddings learned through GNN can be considered features of tasks, and this feature extraction does not require manual feature engineering. In this section, the goal is to output a task-level embed with Fn and An as inputs. On contains all information about the application requested by the program n, and each line of On,i represents the embedded vector of the task vn,i. GNN based on the spectral domain method cannot be used to solve the node representation of a directed graph because the Laplacian matrix is an asymmetric matrix. Therefore, only GNN based on spatial domain can be used to learn the embedded vector of the task. Task embedding process, given the structure of application Gn and the characteristic Fn of all tasks, tasks can pass information between each other through interdependence.

Action space: In each decision step t, there is one and only one task that can be decided. The action At at the current time t is defined as: according to St, a matching decision needs to be made for the current decision task. Indicates that this task is executed in one of the preferences list.

B. Actor–Critic algorithm model

In this work, the ACED algorithm is used to train the model, which is the basic framework for many DRL-based algorithms, such as near-end strategy optimization. First, here's how ACED uses the actor-critic framework to handle the problem of multitasking computing decisions. The actor-critic framework consists of three parts: 1) actor-network for action selection and 2) critical-Network for evaluating state values, and 3) update module. The overall goal of the model is to select the decision with the lowest energy efficiency cost as the best strategy for matching the solution. Calculating rewards using discount factors can reduce execution losses for tasks that reach the target NCP first, and wait for tasks that are decided first but arrive later. In this case, there is little impact on the overall situation. Thus, the global impact of asynchrony can be approximately eliminated.

The purpose of the participant network is to sample an action for the task selected in the current decision step. In this model, the input of the actor network includes information about edge nodes, user profiles and tasks to be decided. Depending on the state of the environment, the actor network outputs the possibility of all actions. Sampling the action and applying the action to the NCP environment can get the state of rt and the next step st+1. The critic network is similar to the actor network in that it contains a GCN layer to capture the DAG structure of the application and the DNN to understand the NCP environment, which is then evaluated using an MLP with a node output. This work uses Tanh as the activation function. Current decision step t The state of the current task St is used as the input to the critic network and the output is the value of the current state. Next, the process of updating the network is described. This work uses the mean square objective to measure the difference between the values of state t and Rt. In contrast to other policy networks, the PPO algorithm uses clipping parameters to limit changes in each network update. It allows you to perform small-batch training on multiple epochs using the experience gathered in the current iteration.

ACED algorithm uses the Actor-Critic framework in reinforcement learning to dynamically schedule and allocate tasks and resources in the NCP environment. The Actor network is responsible for generating scheduling policies, while the Critic network evaluates the performance of the policies. By constantly adjusting the parameters of Actor and Critic in each iteration, the algorithm gradually optimizes the scheduling decision, and finally realizes the optimal task execution and resource utilization.

Update parameters in DNN by updating unload policies. An experiential replay memory is designed to store past states-action pairs. In each slot, task data collection for all devices L, channel gain sets for all devices h, and offload decisions As generated in the entire actor-critic network at the t gap, are added to memory as new training data. In the implementation, the DNN is trained only after more than half of the memory size data sample has been collected. However, new data is constantly added to the memory, and once the memory is full, the old data is replaced. Memory is used to train DNNS at update intervals. Specifically, at t time slot intervals, regular training sessions are conducted with DNNS to prevent overfitting of the model. The parameters of the DNN were then continued to be updated using the Adam algorithm, which was designed to minimize the mean cross-entropy loss function on the data sample.

Agent i observes the environment and receives its initial state oti, then obtains a task partitioning policy and a transmission power policy by entering the state into the participant network. In order to improve the efficiency of exploration, noise t is removed from the (OU) process to generate time-dependent exploration during the learning process. After that, all agents perform their corresponding task offloading actions and observe reward ut, union with the next state st+1. The proxy is then stored in a centralized replay buffer (st,at,ut,st+1). Repeat until the replay buffer is full, and then randomly sample H samples from the replay buffer to form a small batch for training the participant network, as well as the centralized criticism network.

V. CONCLUSIONS

In this paper, a task resource matching decision strategy based on deep reinforcement learning is proposed to maximize the computing power of the system. Algorithms enable all devices to compete for the computing resources available in each time slot, ensuring maximum fairness for all mobile devices. In addition, the order quantization algorithm is used to process the output parameters of DNN to speed up the convergence of the model. The experimental results

show that the energy efficiency of the system is improved effectively.

Future research could explore how to further optimize the DRL algorithm, taking into account more real-world factors (such as network latency, equipment failures, etc.), and how to verify the effectiveness of the algorithm in larger and more complex systems. In summary, the deep reinforcement learning based task resource matching decision strategy proposed in this paper shows its potential in improving the computing power and energy efficiency of the system, but further research and practice are needed to verify its applicability and stability in various practical scenarios.

REFERENCES

[1] M.R.Schurgot,M.Wang,A.E.Conway,L.G.Greenwald,and P.D.Lebling,“A dispersed computing architecture for resource-centric computation and communication,”IEEE Commun.Mag.,vol.57,no.7,pp.13–19,Jul.2019.

[2] H.Yange tal.,“Dispersed computing for tactical edge in future wars:Vision,architecture,and challenges,”Wireless Commun.Mobile Comput.,vol.2021,Jan.2021,Art.no.8899186.

[3] Knezevic A,Nguyen Q,Tran J A,Ghosh P,Annavaram M(2017)CIRCE-a runtime scheduler for DAG based dispersed computing.In: /IEEE symposium on edge computing,San Jose/Silicon Valley,CA,USA,pp1–2.

[4] Conway A E,Wang M,L juca E,Lebling PD(2020)A Dynamic Transport Overlay System for Mission Oriented Dispersed Computing Over IoBT.In:MILCOM2019–2019 IEEE military communications conference(MILCOM),Norfolk,VA,USA,pp815–820.

[5] Yang H,Li G,Sun G,etal(2021)Dispersed computing for tactical edge in future wars:vision,architecture,and challenges.Wirel Commun Mob Comput2021:8899186:1-8899186:31.

[6] J.Wang,J.Hu,G.Min,W.Zhan,A.Zomaya,andN.Georgalas,“Dependent task off loading for edge computing based on deep reinforcement learning,”IEEE Transactions on Computers,2021.

[7] J.Chen,Y.Yang,C.Wang,H.Zhang,C.Qiu,and X.Wang,“Multi task off loading strategy optimization based on directed a cyclic graphs for edge computing,”IEEE Internet of Things Journal,2021.

[8] C.-S.Yang,R.Pedarsani,and A.S.A vestimehr,"Communication-a ware scheduling of serial tasks for dispersed computing,"IEEE/ACM Trans.Net w.,vol.27,no.4,pp.1330–1343,Aug.2019.

[9] A.Knezevicetal.,"CIRCE—A runtime scheduler for DAG-based dispersed computing:Demo,"in Proc.2nd ACM/IEEE Symp.Edge Comput.(SEC),Oct.2017,pp.1–2.

[10] P.Ghosh,Q.Nguyen,and B.Krishnamachari,"Container orchestration for dispersed computing,"in Proc.5th Int.Workshop Container Technol.Container Clouds,Dec.2019,pp.19–24.

[11] H.Wuetal.,"Resolving multitask competition for constrained resources in dispersed computing:A bilateral matching game,"IEEE Internet ThingsJ.,vol.8,no.23,pp.16972–16983,Dec.2021

[12] P.Rahimzadehetal.,"SPARCLE:Stream processing applications over dispersed computing networks,"in Proc.40th IEEE Conf.Distrib.Comput.Syst.,Singapore,2020,pp.1067–1078.

[13] D.Huand B.Krishnamachari,"Throughput optimized scheduler for dispersed computing systems,"in Proc.7th IEEE Conf.Mobile Cloud Comput.Services Eng.,Newark,CA,USA,2019,pp.76–84.

[14] C.Zhou,C.Gong,H.Hui,F.Lin,andG.Zeng,"Atask-resource joint management model with intelligent control for mission-aware dispersed computing,"China Commun.,vol.18,no.10,pp.214–232,Oct.2021.

[15] C.S.Yang,R.Pedarsani,andA.S.Avestimehr,"Communication-aware scheduling of serial tasks for dispersed computing,"IEEE/ACM Trans.Netw.,vol.27,no.4,pp.1330–1343,Aug.2019.

[16] J.BaekandG.Kaddoum,"Heterogeneous task off loading and resource allocations via deep recurrent reinforcement learning in partial observablemultifog networks,"IEEE Internet ThingsJ.,vol.8,no.2,pp.1041–1056,Jan.2021.

[17] L.Qian,Y.Wu,F.Jiang,N.Yu,W.Lu,andB.Lin,"NOMA assisted multitask multi-access mobile edge computing via deep reinforcement learning for industrial Internet of Things,"IEEE Trans.Ind.In format.,vol.17,no.8,pp.5688–5698,Aug.2021.

[18] J.Heydari,V.Ganapathy,andM.Shah,"Dynamic task off loading in multi-agent mobile edge computing networks,"in Proc.IEEE Global Commun.Conf.(GLOBECOM),Waikoloa,HI,USA,Dec.2019,pp.1–6.

[19] Z.Cao,P.Zhou,R.Li,S.Huang,andD.Wu,"Multi agent deep reinforcement learning for joint multi channel access and task off loading of mobile-edge computing in industry4.0,"IEEE Internet Things J.,vol.7,no.7,pp.6201–6213,Jul.2020.

[20] X.Wang,Z.Ning,andS.Guo,"Multi-agent imitation learning for pervasive edge computing:A decentralized computation off loading algorithm,"IEEE Trans.Paralle lDistrib.Syst.,vol.32,no.2,pp.411–425,Feb.2021.

[21] J.Yan,S.Bi,andY.J.Angela Zhang,"Off loading and resource allocation with general task graph in mobile edge computing:A deep reinforcement learning approach,"IEEE Trans.Wireless Commun.,vol.19,no.8,pp.5404–5419,Aug.2020.

[22] J.Zou,T.Hao,C.Yu,andH.Jin,"A3C-DO:A regional resource scheduling framework based on deep reinforcement learning in edge scenario,"IEEE Trans.Comput.,vol.70,no.2,pp.228–239,Feb.2021.

[23] L.Ale,N.Zhang,X.Fang,X.Chen,S.Wu,and L.Li,"Delay-aware and energy-efficient computation off loading in mobile-edge computing using deep reinforcement learning,"IEEE Trans.Cogn.Commun.Netw.,vol.7,no.3,pp.881–892,Sep.2021,doi:10.1109/TCCN.2021.3

[24] X.Chen,H.Zhang,C.Wu,S.Mao,Y.Ji,and M.Bennis,"Optimized computation off loading performance in virtual edge computing systems via deep reinforcement learning,"IEEE Internet Things J.,vol.6,no.3,pp.4005–4018,Jun.2019.