Profit-Aware Computing Server Clustering and Task Scheduling in the Computing Power Network

Xiaoyao Huang[†§], Remington R. Liu[§], Jie Wu[¶], and Baoxian Zhang[‡]
[†]Cloud Computing Research Institute, China Telecom, P.R.China
[§]China Telecom Research Institute, P.R.China
[¶]Department of Computer and Information Sciences, Temple University, USA
[‡]Research Center of Ubiquitous Sensor Networks, University of Chinese Academy of Sciences, P.R.China
Email: huangxy32@chinatelecom.cn, remington.liu@outlook.com, jiewu@temple.edu, bxzhang@ucas.ac.cn

Abstract—**Computing power network has emerged as an attractive technology to tackle the increasing demand for computational resources in cloud networking. In this paper, we study the problem of optimizing the formation of clusters of computing servers/nodes and also the task scheduling considering the games between the platform and computing nodes to maximize the platform profit. We formulate this problem as an integer programming problem. We propose a deep reinforcement-learning based server clustering and auction-based task scheduling algorithm working at different time scales to solve the problem. The deep reinforcement learning-based server clustering algorithm works at a large time scale to optimize the sizes and compositions of different clusters based on temporal and spatial distribution of the tasks and also their characteristics. The auction-based task scheduling algorithm works at a small time scale to match the tasks with the clusters while satisfying the QoS requirements of tasks so as to maximize the profit of the platform. Extensive simulations are conducted to evaluate the performance of proposed algorithm and the results show its high performance.**

Index Terms—**Computing power network, server clustering, task scheduling**

I. Introduction

The rapid development of artificial intelligence (AI) technology especially the large language models (LLMs) has led to an increasing demand for computility [1]. However, computing resources are expensive and limited, making it challenging to support the explosive computation-intensive tasks with various quality of service (QoS) requirements [2], [3]. As a response to these challenges, the computing power network (CPN) has emerged as a key technology in next-generation networking that manages and allocates integrated network and computing resources from different sources flexibly to satisfy the increasing computation demands with improved system resource utilization performance and also satisfactory quality of services [4], [5].

In a large-scale CPN system, computing resources exhibit different processing capabilities, different geographical locations, and also different computing costs. Meanwhile, the computational capacity of a single computing node is often insufficient for some AI tasks with large computational power demands. In addition, the rapid development of high-speed network technology makes it possible for task training and inference over long distances. However, the latency caused by the distances among computing nodes in a CPN leads to difficulties in scheduling computing resources across the entire system to meet the delay requirements of parallel computing. As a result, optimized computing nodes clustering within given latency circle to meet the requirements of large computational tasks, and scheduling the tasks to clusters with different prices to maximize the platform profit has been an active research direction, which is the primary focus of our work in this paper.

In this paper, we study the problem of joint optimization of computing node clustering and task scheduling considering the games between the platform and the clusters for maximized platform profit. We formulate this problem as an integer programming, which is known to be NP-hard. To address this issue, we propose a deep reinforcement learning based server clustering and auction based task scheduling algorithm working at different time scales. The deep reinforcement learning (DRL) based server clustering algorithm works at a large time scale to optimize the sizes and compositions of different clusters based on the temporal and spatial distribution of tasks and also their characteristics. The auction based task scheduling algorithm works at a small time scale to match the tasks with the clusters while satisfying the tasks' QoS requirements so as to maximize the profit of the platform. Extensive simulations are conducted to evaluate the performance of the proposed algorithm and the results show its high performance.

The reminder of this paper is organized as follows. In Section II, we briefly review the related work. In Section III, we describe the framework of the system and related models and formulate the problem under study. In Section IV, we present the detailed design of the proposed algorithms. Comprehensive simulations are conducted to evaluate performance in Section V. In Section VI, we conclude this paper and give some potential directions for future work.

II. Related Work

In this section, we present a brief review of existing work related to resource management as well as economic models and pricing strategies in the area of CPNs.

Significant existing research has been focused on optimizing the allocation and scheduling of computational and network resources in CPNs to meet certain performance and latency requirements. In [6], the authors proposed a novel knowledge graph representation for the architecture of a computing power network which can automatically execute scheduled tasks using a knowledge-driven approach based on the constructed knowledge graph. In [7], the authors proposed a computation-driven clustering strategy to optimize network bandwidth resource and minimize waiting delays at the central server for federated learning in CPN. In [8], the authors proposed an in-network pooling framework using a DRL scheme to optimize idle computing and caching resources in computing power networks for 5G/6G applications. In [9], the authors proposed a comprehensive evaluation model for assessing the matching degree between supply and demand of computing power network services based on multi-dimensional data. In [10], the authors introduced a service intent-aware task scheduling framework for CPNs, leveraging intent-based networking principles to improve the task scheduling performance by considering the underlying service intent of applications and integrating resource orchestration and network control. In [11], the authors proposed a framework enabling the adaptability for computing-power users, the flexibility for networking, and the profitability for computing-power providers by formulating these three aspects into objective functions and employed the greedy algorithm to address the optimization problems.

Research on economic models and pricing strategies focuses on efficient resource allocation and incentivizing the provision of CPN services. In [12], the authors proposed a secure and decentralized federated learning approach based on blockchain, using a proof-of-accuracy consensus scheme and an evolutionary game incentive scheme to enhance the efficiency and security of federated learning in CPNs. In [13], the authors proposed a digital twin transfer scheme based on Shapley value and double auction scheme to provide efficient computing services. In [14], the authors proposed an incentive mechanism based on the Stackelberg game framework and a joint optimization algorithm to improve CPNs' energy efficiency and performance. In [15], the authors proposed a reputation-enhanced resource trading framework for CPN integrated with blockchain technology, using a decentralized reputation model and incentives to ensure dependable and equitable computing services.

However, all above work optimizes the task scheduling without considering the computing nodes clustering for serving large tasks with high processing requirement under reasonable small intra-cluster latency. In this paper, we shall optimize the formation of computing node clusters and the task scheduling considering the gaming between the service platform and computing clusters for maximized platform profit. The main contribution of this paper is technical mechanism innovation rather than in-depth economic analysis.

III. System Model and Problem Formulation

In this section, we first describe the architecture of a CPN system, and then introduce the computing model and utilities of different parties in the system. Finally, we formulate the problem under study.

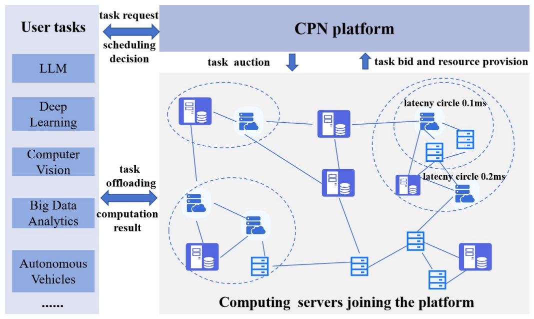

Fig. 1. The system architecture.

A. Network Model

Fig. 1 shows the CPN system under study in this paper, which consists of a CPN platform, computing servers located in different geographical locations with varying computational capabilities, and various tasks from user side, which have dynamic spatiotemporal characteristics and varying computational demands. Users subscribe computing services from the platform on a paid basis and offload tasks to the servers according to the platform's scheduling decisions. The computing servers (denoted by $\mathcal{S}$) join the system to provide resource and earn revenue via task processing. The platform works to schedule the tasks to proper participating servers while satisfying the tasks' QoS requirements to maximize its own profit. Consider some tasks might be too large for a single computing node to handle on its own, and further the tasks are not allowed to be split across distantly located nodes for collaborative processing due to the computation synchronization requirement among nodes for enabling such collaborative processing. For this concern, the platform can choose to form clusters of servers that are closely located with low enough latency for meeting the synchronization requirement of processing large tasks, thereby accommodating more task requests and thus increased profits.

The system works at two time scales: Optimized cluster formation at a large time scale (called cycle, denoted by $t \in \mathcal{T}$) and task scheduling a small time scale (called slot, denoted by $\delta \in \Delta$). Tasks are allowed to be scheduled across slots. A server $i \in \mathcal{S}$ can be represented by a tuple $\langle f_i, \boldsymbol{L}_i \rangle$, where f_i and $\boldsymbol{L}_i$ represent the computing power and the location of server i, respectively. We use binary variable $x_{i,j}$ to indicate whether servers i and j belong to the same cluster where $x_{i,j} = 1$ means yes. A cluster can be denoted as $\mathcal{C}_k^t \in \mathcal{C}^t$ where $\mathcal{C}^t$ is the set of clusters in time cycle t. It is worth noting that when all clusters have been formed, each independent node having not joined any cluster is also treated as a cluster for convenience. So we have $\sum \mathcal{C}_k^t = \mathcal{S}$. The set of tasks arrived at time slot δ is denoted by $\mathcal{R}^\delta$. Each task $r_n \in \mathcal{R}^\delta$ can be represented as a tuple $\langle l_n, m_n, \tau_n, \boldsymbol{O}_n \rangle$, where l_n represents its computation workload, m_n denotes its data size, τ_n refers to its delay requirement, and $\boldsymbol{O}_n$ refers to the location of task r_n.

B. Computing Model

The total computing capacity of a cluster $\mathcal{C}_k^t$ can be calculated as $f_{\mathcal{C}_k^t} = \sum_{i\in\mathcal{C}_k^t} f_i$. Only for the clusters formed by spatially close enough computing servers, the latency among nodes in such clusters is negligible and the computing synchronization requirement is respected in this case. As a result, the size of a cluster can be limited by restricting the maximum distance between servers in the cluster, which can be denoted by $R_{\mathcal{C}_k^t} = \max\limits_{p,q\in\mathcal{C}_k^t} \|\boldsymbol{L}_p - \boldsymbol{L}_q\|_2$ and $R_{\mathcal{C}_k^t} \leqslant R_{\max}$, where $R_{\max}$ is the maximum distance threshold determined by the computing synchronization requirement.

Tasks processing at each cluster works in a first-in-first-out manner. The computing delay $t_{n,k}^{proc}$ of a task r_n processed by cluster $\mathcal{C}_k^t$ can be calculated as

$$t_{n,k}^{proc} = \frac{l_n}{f_{\mathcal{C}_k^t}}. \tag{1}$$

Since each cluster $\mathcal{C}_k^t$ has a service queue with certain unprocessed workload $L_{n,k}$ before task r_n arrives, which can be obtained by counting the currently unexecuted tasks. the queueing delay of task r_n is

$$t_{n,k}^{wait} = \frac{L_{n,k}}{f_{\mathcal{C}_k^t}}. \tag{2}$$

The transmission delay $t_{n,k}^{trans}$ is computed as:

$$t_{n,k}^{trans} = \sum_{s\in P_{n,k}} \frac{m_n}{v_s} + \sum_{s\in P_{n,k}} \frac{d_s}{V_s}, \tag{3}$$

where the first term on the right side denotes the transmission delay,with v_s representing the bandwidth of the link s for the corresponding network path $P_{n,k}$ from task r_n to cluster $\mathcal{C}_k^t$. The second term is for the propagation delay, where d_s is the length of link s, and V_s stands for the propagation speed, which may vary depending on the medium (e.g., fiber optic, copper). The delay of a task includes communication delay, queueing delay, and computing delay. Therefore, the total delay of task r_n offloaded to cluster $\mathcal{C}_k^t$ is computed as follows.

$$t_{n,k}^{total} = t_{n,k}^{trans} + t_{n,k}^{wait} + t_{n,k}^{proc}. \tag{4}$$

C. Utility of Different Parties

The utility of platform: The platform provides computing services to users and charges a certain fee. The fee p_n for processing task r_n can be calculated as [16]:

$$p_n = p(1 + \frac{\mu}{\tau_n})l_n, \tag{5}$$

where p represents the basic unit price and μ is a coefficient. The platform has to pay the server clusters with assigned tasks for processing. We use binary variable $y_{n,k}$ to denote the scheduling decision of task r_n. If $y_{n,k} = 1$, it indicates that task r_n is allocated to cluster $\mathcal{C}_k^t$ for processing. The corresponding payment to the cluster $\mathcal{C}_k^t$ for processing of task r_n is denoted by $g_{\mathcal{C}_k^t,n}$. The utility of the platform can then be computed as:

$$U_c = \sum_{\delta\in\Delta} \sum_{r_n\in\mathcal{R}^\delta} \left(p_n - \sum_{\mathcal{C}_k^t\in\mathcal{C}} y_{n,k} g_{\mathcal{C}_k^t,n} \right). \tag{6}$$

The utility of a user: Users subscribe the task computing service from the platform and obtain the task result after a task is completed. It is worth noting that if computing resources cannot meet a task's delay requirement, the user will abandon the task offloading. The value of a task r_n is represented by v_n. Therefore, the utility of a user for the offloading of task r_n is

$$U_n = \sum_{\mathcal{C}_k^t\in\mathcal{C}} (y_{n,k} v_n - p_n). \tag{7}$$

The utility of a cluster: A task scheduled to a cluster will be decomposed onto all the servers in the cluster in proportion to the computing power of each server node. As a result, the cost of cluster k for processing task r_n includes the total cost of all the servers in the cluster $i \in \mathcal{C}_k^t$ including energy consumption cost and computing resource occupation cost.

$$c_{n,k} = \sum_{i\in\mathcal{C}_k^t} \left(\varepsilon_e l_{n,i} f_i^2 + \varepsilon_t \frac{l_{n,i}}{f_i} \right), \tag{8}$$

where $l_{n,i}$ represents the load assigned to server i and $l_n = \sum_{i\in\mathcal{C}_k^t} l_{n,i}$. ε_e and ε_t are the economic costs per unit energy consumption and time, respectively. Therefore, the utility of a cluster k is thus

$$U_k = \sum_{\delta\in\Delta} \sum_{r_n\in\mathcal{R}^\delta} y_{n,k} \left(g_{\mathcal{C}_k^t,n} - c_{n,k} \right) \frac{l_{n,i}}{l_n}, \tag{9}$$

and thus the utility of a server $i \in \mathcal{C}_k^t$ is $U_i = \sum_{\delta\in\Delta} \sum_{r_n\in\mathcal{R}^\delta} y_{n,k} (g_{\mathcal{C}_k^t,n} \frac{l_{n,i}}{l_n} - (\varepsilon_e l_{n,i} f_i^2 + \varepsilon_t \frac{l_{n,i}}{f_i}))$.

D. Problem Formulation

In this paper, the objective is to maximize the total profit of the CPN platform, which equals the total income charged from the users for providing task offloading services minus the payment made to the computing servers for task processing, while satisfying the delay requirements of tasks and rationality of all parties. In addition, only the severs should within the distance constraint can be formed in to a cluster. Accordingly, the CPN platform profit maximization problem is formulated as **P1**.

In **P1**, constraint (10a) ensures that a task can only be assigned to one cluster. Constraint (10b) ensures that only those tasks whose delay requirements can be satisfied will be assigned. Constraint (10c) limits the size of a cluster. Constraints (10d) and constraint (10e) indicate that the optimization variables are 0-1 integer variables. Constraint (10f) ensures the rationality of all parties. The above problem is an integer programming problem, which is NP-hard. Due to the huge and complex optimization space due to the large amount of tasks with spatiotemporal dynamics and heterogeneous resources, in the next section, we shall propose a DRL-based server Clustering Algorithm and an auction-based Task scheduling algorithm (DCAT) to address this problem.

$$\textbf{P1:} \max_{\boldsymbol{x},\boldsymbol{y}} \sum_{t\in\mathcal{T}}\sum_{\delta\in\Delta}\sum_{r_n\in\mathcal{R}^\delta}\left(p_n - \sum_{\mathcal{C}_k^t\in\mathcal{C}} y_{n,k} g_{c_k^t,n}\right) \tag{10}$$

$$\text{s.t.} \sum_{\mathcal{C}_k^t\in\mathcal{C}^t} y_{n,k} = 1, \forall r_n \in \mathcal{R}^\delta \tag{10a}$$

$$\sum_{\mathcal{C}_k^t\in\mathcal{C}^t} y_{n,k} t_{n,k}^{total} \leqslant \tau_n, \forall r_n \in \mathcal{R}^\delta \tag{10b}$$

$$\max_{p,q\in\mathcal{C}_k^t} \|\boldsymbol{L}_p - \boldsymbol{L}_q\|_2 \leqslant R_{\max}, \forall \mathcal{C}_k^t \in \mathcal{C}^t \tag{10c}$$

$$y_{n,k} \in \{0,1\}, \forall r_n \in \mathcal{R}^\delta, \mathcal{C}_k^t \in \mathcal{C}^t \tag{10d}$$

$$x_{i,j} \in \{0,1\}, \forall i,j \in \mathcal{S} \tag{10e}$$

$$U_c \geqslant 0, U_n \geqslant 0, U_i \geqslant 0, \forall r_n \in \mathcal{R}^\delta, i \in \mathcal{C}_k^t \in \mathcal{C}^t \tag{10f}$$

IV. Proposed DCAT Algorithm

In this section, we propose the DCAT algorithm to maximize the platform profit by optimizing the decision of cluster formation and task scheduling. DCAT works at two different time scales considering the system efficiency. At the large time scale, DCAT first makes the clustering decision including the sizes of formed clusters as well as their member nodes based on the learning of the spatiotemporal distribution of tasks and also their characteristics. At the small time scale, DCAT schedules the tasks arrived at a time slot to proper clusters by auctions while considering their delay requirements. In this section, we first model the Markov decision process of the clustering based profit maximization problem, and then propose the deep reinforcement learning-based method to solve the clustering problem and design auction-based method to solve the scheduling problem. Finally, the detailed procedure of the DCAT algorithm is presented.

A. Markov Decision Process

A Markov decision process is composed of a quintuple $\langle\mathcal{S},\mathcal{A},\mathcal{R},\mathcal{P},\xi\rangle$, where $\mathcal{S}$ is the state set, $\mathcal{A}$ is the action set, $\mathcal{R}$ is the reward function, $\mathcal{P}$ is the state transition function[1], and ξ is the discount factor. The Markov decision process is modeled as follows.

- *State Set* $\mathcal{S}$: The CPN needs to determine which servers for cluster formations based on the computing power of each server and the spatial and temporal distribution of task arrivals and servers. Therefore, the status set is defined as the computing power of each server, the task arrival rate of each area, and the task workload, data size, as well as deadline, that is, $\mathcal{S} \triangleq \langle s_t\rangle = \langle \boldsymbol{\lambda}, \boldsymbol{l}, \boldsymbol{m}, \boldsymbol{\tau}\rangle$.
- *Action Set* $\mathcal{A}$: The action set is the clustering decision of CPN, that is, $\mathcal{A} \triangleq \langle a_t\rangle = \langle \boldsymbol{x}\rangle$.

[1]Since this is a model-free process, there is no closed-form expression for the probability transfer function.

- *Reward Function* $\mathcal{R}(s_t,a_t)$: The reward function is the objective function of the platform profit maximization problem, that is, $\mathcal{R}(s_t,a_t) = \sum_{\delta\in\Delta}\sum_{r_n\in\mathcal{R}^\delta}\left(p_n - \sum_{\mathcal{C}_k^t\in\mathcal{C}} y_{n,k} g_{c_k^t,n}\right)$.

B. A2C-based Server Clustering Algorithm

To solve the platform profit maximization problem effectively, DCAT adopts reinforcement learning method based on Advantage Actor Critic (A2C). The actor network of A2C is used to learn the clustering strategy $\pi(s)$ of the platform, and the critic network is used to learn the value function $V_\pi(s_t)$ that is the expected return under the action of policy π starting from state s_t and can be calculated by

$$V_\pi(s_t) = \mathbb{E}[r_{t+1} + \xi r_{t+2} + \xi^2 r_{t+3} + \ldots | s_t]. \tag{11}$$

$Q_\pi(s_t,a_t)$ is the agent starting from state s_t, first executing action a_t, and then following the expected return of strategy π and can be calculated by

$$Q_\pi(s_t,a_t) = \mathbb{E}[r_{t+1} + \xi r_{t+2} + \xi^2 r_{t+3} + \ldots | s_t, a_t]. \tag{12}$$

The policy gradient method used in the A2C algorithm is as follows.

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) Q_{\mathbf{w}}(s,a)\right], \tag{13}$$

where $\nabla_\theta J(\theta)$ represents the policy gradient and $\pi_\theta(s,a)$ represents the probability of choosing a in state s. For the value function update of the Critic network, we can use the TD error ϑ to calculate the error between the current state-action value and the next moment state-action value.

$$\vartheta = r_{t+1} + \xi Q_w(s_{t+1},a_{t+1}) - Q_w(s_t,a_t). \tag{14}$$

The formula for updating the actor network based on the Q value function is $\theta = \theta + \alpha\nabla_\theta \log \pi_\theta(s_t,a_t) Q_w(s_t,a_t)$, and the formula for updating the Q function approximator network is $w = w + \beta\vartheta\nabla_w Q_w(s_t,a_t)$. The detailed procedure is shown in **Algorithm 1**. First, initialize the parameters of the value network $Q_w(s_t,a_t)$ and policy network $\pi_\theta(s,a|\theta)$ of deep reinforcement learning (line 1). Then enter the main training process. In each round of training, the policy network outputs a clustering strategy a_t based on the state observation results (line 3). During the execution of the clustering strategy (from line 4 to line 8), at each small time scale, **Algorithm 2** is executed to assign tasks to a cluster (line 6). The agent then receives a reward $r_{t,n}^\delta$ based on the task assignment result. When the large time scale ends, the state of the system at the next time s_{t+1} and the total reward r_t during this round of time will be observed (line 9). Then the network parameters are updated according to the returned value (from line 10 to line 12).

C. Auction-based Task Assignment Algorithm

After server clustering is completed, an effective mechanism is needed to enable tasks to be assigned to proper clusters while satisfying the rationalities of all parties. For this purpose, we design a task allocation mechanism based on a progressive

Algorithm 1: A2C-based Server Clustering Algorithm.

```
1  Initialize $Q$ function approximator $Q_w(s,a)$, initialize policy approximator $\pi_\theta(s,a|\theta)$;
2  for $t$ = 0,1,2,... do
3      Sampling clustering strategy $a_t \sim \pi_\theta(s_t,a_t|\theta)$ and make $a_t$ satisfy the constraints, and then execute it;
4      for $\delta$ = 0,1,2,... do
5          for task $r_n \in \mathcal{R}^\delta$ do
6              Algorithm 2 is executed to assign the task to a certain cluster and reward $r^\delta_{t,n}$ is returned;
7          end
8      end
9      Calculate $r_t = \sum_\delta \sum_{r_n \in \mathcal{R}^\delta} r^\delta_{t,n}$ and observe $s_{t+1}$;
10     $\delta \leftarrow r_{t+1} + \gamma Q_w(s_{t+1},a_{t+1}) - Q_w(s_t,a_t)$;
11     $w \leftarrow w + \beta\delta\nabla_w Q_w(s_t,a_t)$;
12     $\theta \leftarrow \theta + \alpha\nabla_\theta \log \pi_\theta(s_t,a_t) Q_w(s_t,a_t)$;
13     $t \leftarrow t+1$;
14 end
```

Algorithm 2: REA Algorithm.

```
   Input: Cluster set $\mathcal{C}^t$, task $r_n$, current workload of each server, the bid reduction coefficient $\gamma$.
   Output: The income after task assignment.
1  $k \leftarrow 0$, $p_n^{(k)} \leftarrow p_n$;
2  According to the cluster set $\mathcal{C}^t$ and the running status and workload of each server, candidates that meet the offloading task conditions are obtained and a candidate set $\mathcal{C}_a^{(k,t)}$ is formed;
3  while $k \leqslant K$ do
4      if $\mathcal{C}_a^{(k,t)} = \emptyset$ then
5          Task $r_n$'s assignment failed;
6          return 0;
7      end
8      if $|\mathcal{C}_a^{(k,t)}| = 1$ then
9          Assign task $r_n$ to the server $\mathcal{C}_i^t \in \mathcal{C}_a^{(k,t)}$;
10         return $p_n - p_n^{(k)}$;
11     end
12     $k \leftarrow k+1$;
13     $p_n^{(k)} \leftarrow \gamma p_n^{(k-1)}$;
14     The clusters that will continue to participate in the auction form a new candidate set $\mathcal{C}_a^{(k,t)}$;
15 end
```

offer reduction strategy. In this strategy, the CPN platform acts as the decision-maker, and each cluster is a potential service provider. The platform initially offers the task r_n at a starting price $p_n^{(0)} = p_n$. The offer price is then progressively reduced at each step, with the k-th offer being $p_n^{(k)} = \gamma p_n^{(k-1)}$, where γ is the reduction coefficient and $0 < \gamma < 1$. The process continues until the price reaches a level where a single cluster is willing to accept the task.

Whenever the CPN platform makes an offer, each cluster determines whether it is willing to process the task at that price. If a cluster finds the offer acceptable, it remains in the selection process; otherwise, it exits. The task is then assigned to the cluster that accepts the offer when it is the only remaining participant. We call this strategy as Reduced-Price Equilibrium Allocation (REA) strategy. The detailed procedure is shown in **Algorithm 2**. First, set the number of iterations k to zero and give an initial bid p_k (line 1). Then generate a candidate set $\mathcal{C}_a^{(k,t)}$ according to the constraints (line 2). In each round of bidding (from line 3 to line 15), first determine whether the task allocation fails (line 5) or is successful (line 9). If the exit condition is not triggered, the bid will be lowered (line 13) and a new round of bidding $p_n^{(k)} \leftarrow \gamma p_n^{(k-1)}$ will be carried out.

In order to analyze whether REA satisfies incentive compatibility, we first give the definition of incentive compatibility.

***Definition** 1 (Incentive Compatibility, IC): IC requires that each participant (i.e., cluster) can obtain the maximum utility (reward) when honestly reporting its actual ability or willingness to accept the task, and will not obtain higher benefits by misrepresenting or withdrawing from the auction.*

For REA strategy, we give the following theorem.

***Theorem** 1: REA satisfies IC.*

Proof of Theorem 1: The cost for each cluster $\mathcal{C}_k^t$ to process task r_n is $c_{n,k}$, which is internal information private to the cluster. $p_n^{(k)}$ is the bid for the current task, which decreases gradually in REA. The profit of the cluster after accepting the task is the current bid minus the cost of processing the task.

In REA, the bids $p_n^{(k)}$ are gradually reduced until only one cluster accepts the task. The goal of a cluster is to decide whether to accept the task by evaluating its utility function. Assume that the cost reported by $\mathcal{C}_k^t$ is $\tilde{c}_{n,k}$ (i.e., a false cost). Then there are three cases:

- Case 1 ($\tilde{c}_{n,k} < c_{n,k}$): When $p_n^{(k)} \geq \tilde{c}_{n,k}$, the cluster may accept the task, but because its actual cost $c_{n,k}$ is higher, it may eventually lead to negative utility $U_i(p_n^{(k)}, c_{n,k}) < 0$. This means that the cluster will lose money after accepting the task.
- Case 2 ($\tilde{c}_{n,k} > c_{n,k}$): Since $p_n^{(k)}$ will gradually decrease, it is possible that another cluster accepts the task because of its lower cost, and $\mathcal{C}_k^t$ loses the opportunity to accept the task. In this way, the utility of cluster C_i is zero, although it could have obtained positive utility $U_i(p_n^{(k)}, c_{n,k}) > 0$ by honestly reporting the actual cost.
- Case 3 ($\tilde{c}_{n,k} = c_{n,k}$): When $p_n^{(k)} \geq c_{n,k}$, the cluster accepts the task and obtains positive utility $U_i(p_n^{(k)}, c_{n,k}) \geq 0$. If $p_n^{(k)} < c_{n,k}$, the cluster exits the auction with zero utility but avoids potential negative utility.

Through the above analysis, we can conclude that under the REA mechanism, it is the optimal strategy for cluster C_i to honestly report its true cost $g_{c_k^t,n}$, because false reporting of costs will lead to negative utility or missed profit opportunities. Therefore, the REA strategy is incentive-compatible, and each cluster has no motivation to manipulate or falsely report its capabilities or costs. ■

Through Theorem 1, we know that REA strategy ensures the fairness and efficiency of the system in the task allocation process, and also provides theoretical support for the effective allocation of tasks.

***Theorem** 2: If* $\gamma > 1 - \frac{\min_{p \in \mathcal{C}^t \setminus q}\{c_{n,p} - \min_{q \in \mathcal{C}^t} c_{n,q}\}}{p_n^{(0)}}$,

Algorithm 3: Procedure of DCAT Algorithm.

```
1 Train the initial model according to Algorithm 1 and deploy
    at the platform;
2 for $t$ = 0,1,2,... do
3     Obtain the clustering decision $\boldsymbol{x}$ based on the
        observation according to Algorithm 1;
4     for $\delta$ = 0,1,2,... do
5         Execute Algorithm 2 for task scheduling.
6     end
7 end
```

then the task will be assigned and the difference between the bid and the minimum cost is no greater than $\min_{p\in\mathcal{C}^t\setminus q}\{c_{n,p}-\min_{q\in\mathcal{C}^t} c_{n,q}\}$.

Proof of Theorem 2: Assume that $\mathcal{C}^t$ contains N clusters, and the task processing cost of each cluster $i\in\mathcal{C}^t$ is $c_{n,i}$. Let the server with the lowest cost be 1, that is, $c_{n,1}=\min_{q\in\mathcal{C}^t}\{c_{n,q}\}$ and let $c_{n,2}$ be the cost of the second-lowest-cost server, that is, $c_{n,2}=\min_{p\in\mathcal{C}^t\setminus q}\{c_{n,p}\}$.

According to the algorithm description, the REA algorithm starts with an initial price of $p_n^{(0)}=p_n$ and decreases the price by γ in each iteration, that is, $p_n^{(k)}=\gamma p_n^{(k-1)}$, where $0<\gamma<1$. So, we have

$$p_n^{(0)}-p_n^{(1)}>p_n^{(1)}-p_n^{(2)}>\cdots>p_n^{(k)}-p_n^{(k+1)}. \quad (15)$$

That is,

$$p_n^{(0)}(1-\gamma)>\cdots>p_n^{(k)}(1-\gamma)=\gamma^k p_n^{(0)}-\gamma^{k+1}p_n^{(0)}. \quad (16)$$

Assume that at step k, $\gamma^k p_n^{(0)}<c_{n,2}$, then as long as we make $\gamma^{k+1}p_n^{(0)}>c_{n,1}$, we can ensure the task will be assigned. So, we can get

$$c_{n,2}-c_{n,1}>\gamma^k p_n^{(0)}-\gamma^{k+1}p_n^{(0)}>\gamma^k p_n^{(0)}-c_{n,1}. \quad (17)$$

That is $\gamma>1-\frac{\min_{p\in\mathcal{C}^t\setminus q}\{c_{n,p}-\min_{q\in\mathcal{C}^t} c_{n,q}\}}{p_n^{(0)}}$, and the minimum cost is no greater than $\min_{p\in\mathcal{C}^t\setminus q}\{c_{n,p}-\min_{q\in\mathcal{C}^t} c_{n,q}\}$. The theorem is proved. ■

Theorem 2 shows that as long as a suitable bid reduction coefficient γ is selected, it can be guaranteed that the task will be assigned, and the difference between the assigned income and the minimum cost also has a certain upper limit.

D. Procedure of DCAT Algorithm

The procedure of DCAT is shown in **Algorithm 3**. DCAT works at two different time scales considering the system efficiency. At the large time scale (from line 2 to line 7), DCAT calls Algorithm 1 to form and update computing server clusters based on the spatial and temporal distribution of servers and task arrivals. At the small time scale (from line 4 to line 6), DCAT calls Algorithm 2 to schedule the tasks to most profitable clusters based on the output of Algorithm 1. In return, Algorithm 1 observes the results of Algorithm 2 for optimized clustering decisions.

TABLE I
PARAMETER SETTINGS.

Parm	Values	Unit	Parm	Values	Unit
$\lvert\mathcal{S}\rvert$	20+i10, $i\in[0,4]$	-	f_i	[200, 300]	GHz
l_n	[500, 900]	G	m_n	20	Mb
τ_n	[2, 3]	s	$R_{\max}$	100+i50,$i\in[0,4]$	km
v_s	[100, 500]	Mb/s	$P_{n,k}$	[1, 24]	-
V	3×10^8	m/s	μ	2	-
p	10	-	γ	0.8	-
ϵ_e	10^{-28}	-	ϵ_t	20	-
λ	20, 30, 40, 50	-	ξ	0.99	-

V. Performance Evaluation

A. Simulation Setup

We implemented the proposed algorithm using PyTorch 2.1 and CUDA 12.1, leveraging gym [17] as the environment framework. The server locations and task arrival patterns follow the dataset [18] with computing resources data published by Huawei Group , and we have made adjustments to meet the system's requirements for parameter completeness and internal consistency. The main parameter settings are shown in Table I. For comparison purpose, for the server clustering problem, the following baseline algorithms are simulated:

- **Random Alg. (Rand)**: Servers are randomly divided into clusters, and each cluster satisfies the constraints. (*This baseline is used to illustrate the results of no optimization.*)
- **K-Means based Alg. (KM)**: The number of clusters n is given in advance, and the K-Means algorithm is executed for clustering according to the geographic locations of the servers and delete illegal connections between servers according to the constraints. (*This baseline is used to illustrate the results of optimization based on geographic location information only.*)

For the task allocation problem, the following baseline algorithm is simulated.

- **One-off Bid REA (OB-REA)**: The CPN platform only bids once and then assigns the task to a random cluster in the set of qualified candidates. (*This baseline is used to illustrate the impact of multiple bids and one-off bid on results.*)

Therefore, DCAT will be compared with DCAT-OB-REA, Rand-REA, Rand-OB-REA, KM-REA, and KM-OB-REA.

B. Convergence

First, we verified the convergence of the proposed algorithm DCAT. We show the change in reward value with training in Fig. 2. As can be seen from Fig. 2, the reward gradually increases as training proceeds. Until approximately after the 600th epoch of training, the reward value fluctuates around a certain value. This shows that the proposed algorithm has learned a stable strategy and can converge.

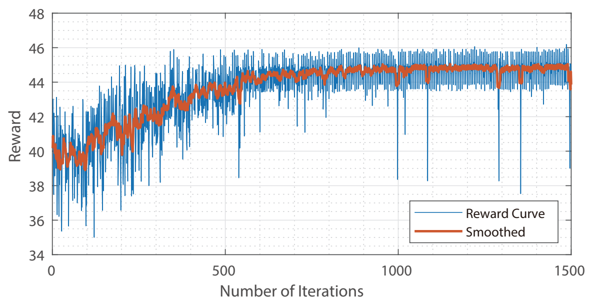

Fig. 2. Convergence of reward value during training.

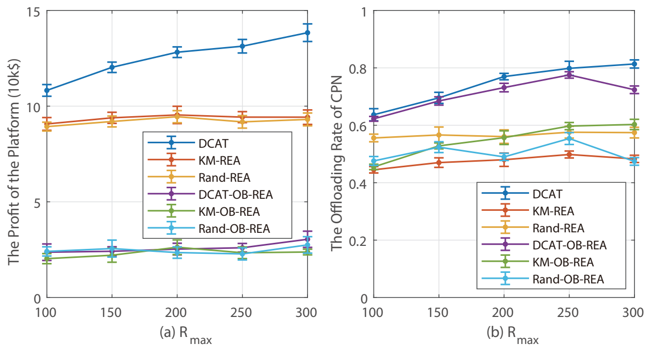

Fig. 4. Impact of cluster size on the profit and offloading rate.

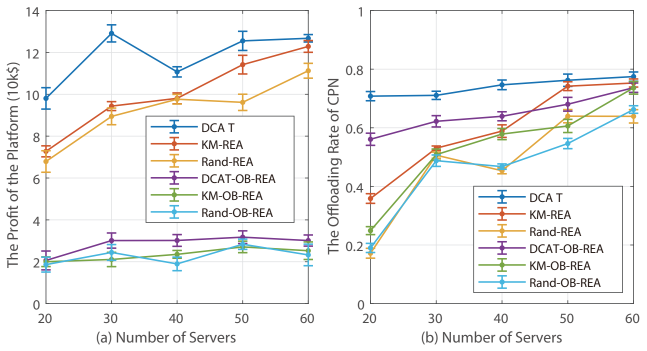

Fig. 3. Impact of number of servers on the profit and offloading rate.

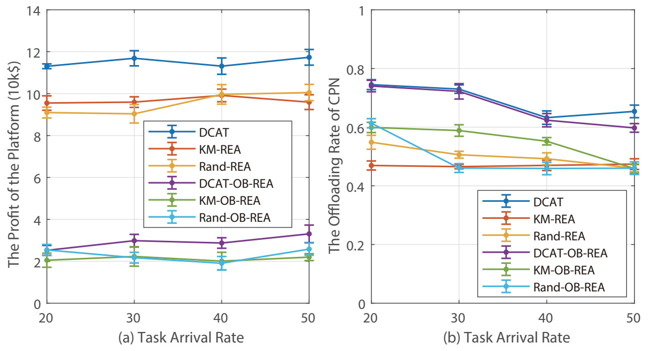

Fig. 5. Impact of task arrival rate on the profit and offloading rate.

C. *Impact of Number of Servers*

Here, we compare the performance of different algorithms as the number of servers changes. We increase the number of servers while keeping the task arrival rate fixed. The platform profit and the task offloading rate are shown in Fig. 3(a) and Fig. 3(b), respectively. In these two figures, it can be seen that as the number of servers increases, both the platform profit and task offloading rate increase. As the number of servers increases, the computing resources in the network also increase, making it possible to offload more tasks. However, since DCAT can not only achieve higher profit by approaching the bidder's private rational bid through a multiple bidding mechanism, but also cluster servers through a profit-aware mechanism to further increase profit. At the same time, the performance of the proposed algorithm is better than the baseline algorithms. For platform profit, DCAT can be 17.51% higher than the best baseline algorithm KM-REA. For task offloading rate, DCAT can be 20.47% higher than the best baseline algorithm KM-OB-REA.

The results show that the expansion of computing resources is an important and efficient solution to improve the profitability of the platform. Therefore, it is of great significance for the service providers of CPN to incentive more third-party computing resources to join the platform. The design of incentive mechanisms in CPN is indeed a significant area of technological research and can be furter studied. In practical implementation, economic models, game theory, blockchain technology, and other methods can be used to design incentive mechanisms to achieve these goals.

D. *Impact of the Size of Cluster*

Here, we evaluate the impact of cluster size on the platform profit and task offloading rate. The results are shown in Fig. 4(a) and Fig. 4(b), respectively. In these two figures, we can see that as the cluster size increases, both the platform profit and task offloading rate increase. At the same time, the performance of the proposed algorithm is better than the baseline algorithms. As the size of the cluster increases, the ability to handle tasks also increases, so the more tasks that can be offloaded, the greater the benefit. For the platform profit, DCAT can be 33.69% higher than the best baseline algorithm KM-REA. For task offloading rate, DCAT can be 19.47% higher than the best baseline algorithm KM-OB-REA. This is because as the cluster scale expands, the task processing capabilities of the platform are significantly enhanced, which leads to a double improvement in load-bearing rate and revenue.

The results show that the improvement of the cluster scale can significantly improve the overall operating capabilities and profitability of the platform, and the expansion of the cluster scale is a key factor in improving the overall performance of the platform. Therefore, it is important to continue to develop high-speed transmission technology in industries to reduce transmission delay and bring the innovation of the CPN.

E. *Impact of Task Arrival Rate*

We compared the performance of different algorithms under different workload by changing the task arrival rate. The results are shown in Fig. 5(a) and Fig. 5(b), respectively, where it can be seen that as task arrival rate increases, the platform profit and task offloading rate will decrease. It is also seen that the performance of the proposed algorithm is better than the baseline algorithms. It can be seen that REA can enable the two parties to interact multiple times and thus approach the optimal value of the implicit bid, which can obtain higher platform benefits compared to one bid. Moreover, compared with the geography-based clustering mechanism, the profit-aware clustering mechanism enables service providers to make clustering decisions based on their own profits, thereby further improving profit on the basis of improving the task offloading rate. For platform profit, DCAT can be 19.09% higher than the best baseline algorithm KM-REA. For task offloading rate, DCAT can be 14.06% higher than the best baseline algorithm KM-OB-REA.

As the task arrival rate increases, the platform's revenue and task load rate will show a downward trend. This is because a higher task arrival rate will put greater pressure on the platform's computing resources, resulting in an inability to undertake all tasks. The DCAT algorithm combines the two key links of task allocation and cluster formation to maintain good performance under different task arrival rates.

VI. Conclusion

In this paper, we focused on a novel scenario where the insufficiency of single-point computing can be compensated by forming clusters of computing nodes that satisfy latency requirements to meet the demands of high-computation-intensive applications in a computing power network. In this scenario, we studied the problem of joint optimization of cluster formation and task scheduling for maximized platform profit. We formulated this problem as an integer programming. We proposed a deep reinforcement learning based server clustering and auction-based task scheduling algorithm to solve the problem. Extensive simulation results demonstrate the high efficiency of the proposed algorithm as compared with baseline algorithms.

Based on the findings and discussions presented in this paper, several promising future directions can be proposed to further advance this research area and explore its potential applications more comprehensively. Firstly, introducing an additional layer of game theory within each cluster could significantly enhance the dynamics of resource allocation and competition for task processing. This would allow individual nodes or entities to strategically compete for tasks based on their computational capabilities and profit motives, leading to a more efficient and economically driven system. Secondly, incorporating blockchain technology into the trading process would provide a robust framework for ensuring the integrity and immutability of transactions. Blockchain's decentralized nature and cryptographic security features make it an ideal solution for establishing trust among participating nodes. Finally, there is a need for further research to develop a more accurate and comprehensive model for assessing the heterogeneity of computational resources. Heterogeneous computing involves utilizing diverse types of processors and architectures to optimize performance and efficiency. A refined model that takes into account the specific characteristics and capabilities of different computing nodes would enable a more precise measurement of their contributions to task processing.

References

[1] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-lm: Training multi-billion parameter language models using model parallelism," *arXiv preprint arXiv:1909.08053*, 2019.

[2] X.-F. Liu, Z.-H. Zhan, and J. Zhang, "Resource-aware distributed differential evolution for training expensive neural-network-based controller in power electronic circuit," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 11, pp. 6286–6296, 2021.

[3] A. Bouguettaya, A. Kechıda, and A. M. Taberkıt, "A survey on lightweight cnn-based object detection algorithms for platforms with limited computational resources," *Int'l Journal of Informatics and Applied Mathematics*, vol. 2, no. 2, pp. 28–44, 2019.

[4] X. Tang, C. Cao, Y. Wang, S. Zhang, Y. Liu, M. Li, and T. He, "Computing power network: The architecture of convergence of computing and networking towards 6G requirement," *China Communications*, pp. 175–185, 2021.

[5] J. Liu, Y. Sun, J. Su, Z. Li, X. Zhang, B. Lei, and W. Wang, "Computing power network: A testbed and applications with edge intelligence," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2022, pp. 1–2.

[6] Y. Bi, Y. Long, Y. Jin, S. Zheng, H. Liu, and H. Wang, "Automatic scheduling technology of computing power network driven by knowledge graph," in *2022 Int'l Conf. on Service Science (ICSS)*, 2022, pp. 154–160.

[7] M. Wei, Q. Zhao, B. Lei, Y. Cai, Y. Zhang, X. Zhang, and W. Wang, "Fedact: an adaptive chained training approach for federated learning in computing power networks," *Digital Communications and Networks*, early access, 2024.

[8] Z. Di, T. Luo, C. Qiu, C. Zhang, Z. Liu, X. Wang, and J. Jiang, "In-network pooling: contribution-aware allocation optimization for computing power network in B5G/6G era," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 3, pp. 1190–1202, 2022.

[9] H. Li, Y. Zhu, J. Wen, and Z. Guo, "A comprehensive evaluation model for the matching degree between supply and demand of the computing power network services," in *IEEE 2nd Int'l Conf. on Electronic Technology, Communication and Information (ICETCI)*, 2022, pp. 270–274.

[10] Q. Tang, R. Xie, L. Feng, F. R. Yu, T. Chen, R. Zhang, and T. Huang, "Siats: A service intent-aware task scheduling framework for computing power networks," *IEEE Network*, vol. 38, no. 4, pp. 233–240, 2024.

[11] X. Wang, X. Ren, C. Qiu, Y. Cao, T. Taleb, and V. C. M. Leung, "Net-in-ai: A computing-power networking framework with adaptability, flexibility, and profitability for ubiquitous ai," *IEEE Network*, vol. 35, no. 1, pp. 280–288, 2021.

[12] P. Wang, W. Sun, H. Zhang, W. Ma, and Y. Zhang, "Distributed and secure federated learning for wireless computing power networks," *IEEE Transactions on Vehicular Technology*, pp. 9381–9393, 2023.

[13] Y. Zhang, H. Zhang, Y. Lu, W. Sun, L. Wei, Y. Zhang, and B. Wang, "Adaptive digital twin placement and transfer in wireless computing power network," *IEEE Internet of Things Journal*, vol. 11, no. 6, pp. 10 924–10 936, 2024.

[14] X. Lin, R. Wu, H. Mei, and K. Yang, "A game incentive mechanism for energy efficient federated learning in computing power networks," *Digital Communications and Networks*, early access, 2023.

[15] L. Lin, J. Wu, Z. Zhou, J. Zhao, P. Li, and J. Xiong, "Computing power networking meets blockchain: A reputation-enhanced trading framework for decentralized iot cloud services," *IEEE Internet of Things Journal*, early access, 2024.

[16] X. Huang, G. Ji, B. Zhang, and C. Li, "Platform profit maximization in d2d collaboration based multi-access edge computing," *IEEE Transactions on Wireless Communications*, pp. 4282–4295, 2023.

[17] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[18] J. Shi, K. Fu, Q. Chen, C. Yang, P. Huang, M. Zhou, J. Zhao, C. Chen, and M. Guo, "Characterizing and orchestrating vm reservation in geo-distributed clouds to improve the resource efficiency," in *Proceedings of the 13th Symposium on Cloud Computing*, 2022, pp. 94–109.