Task Partitioning and Scheduling Based on Stochastic Policy Gradient in Mobile Crowdsensing

Tianjing Wang, *Member, IEEE*, Yu Zhang, Hang Shen, *Member, IEEE*, and Guangwei Bai

***Abstract*—Deep reinforcement learning (DRL) has become prevalent for decision-making task assignments in mobile crowdsensing (MCS). However, when facing sensing scenarios with varying numbers of workers or task attributes, existing DRL-based task assignment schemes fail to generate matching policies continuously and are susceptible to environmental fluctuations. To overcome these issues, a twin-delayed deep stochastic policy gradient (TDDS) approach is presented for balanced and low-latency MCS task decomposition and parallel subtask allocation. A masked attention mechanism is incorporated into the policy network to enable TDDS to adapt to task-attribute and subtask variations. To enhance environmental adaptability, an off-policy DRL algorithm incorporating experience replay is developed to eliminate sample correlation during training. Gumbel-Softmax sampling is integrated into the twin-delayed deep deterministic policy gradient (TD3) to support discrete action space decisions and a customized reward strategy to reduce task completion delay and balance workloads. Extensive simulation results confirm that the proposed scheme outperforms mainstream DRL baselines in terms of environmental adaptability, task completion delay, and workload balancing.**

***Index Terms*—Attention mechanism, Gumbel-Softmax sampling, mobile crowdsensing (MCS), parallel subtask allocation, task partition.**

I. Introduction

Mobile crowdsensing (MCS) technology [1] has become prevalent in recent years due to the rapid proliferation of intelligent mobile devices with computing, perception, storage, and communication capabilities. Unlike traditional sensor networks, MCS leverages intelligent devices carried by mobile users as basic sensing units and forms working groups to collaboratively complete large-scale data sensing. MCS has been applied to various fields such as object tracking [2], environmental monitoring [3], and smart cities [4].

An MCS system must recruit a large number of workers to complete the sensing tasks continuously submitted by the platform [5], [6]. How task allocation is performed to optimize the system's sensing performance is crucial. Task allocation can be offline or online [7]. The former follows a predetermined plan for task allocation, which cannot adjust the allocation strategy in real-time to adapt to environmental dynamics. The latter can handle task allocation flexibly under dynamic environments. Song et al. [8] established an online multiskill task allocation model and integrated a greedy algorithm to match dynamic tasks with specific skill workers. Schmitz and Lykourentzou [9] designed an online-optimized greedy algorithm for reliable task allocation when the budget and quality cycle change. However, these works focused on allocation for simple tasks. The emergence of various MCS applications such as ride-hailing (e.g., DiDi[1]), on-demand delivery (e.g., Ele.me[2]), and live map (e.g., Waze[3]) has made it inevitable for MCS systems to support continuous allocation of complex tasks.

Unlike simple tasks that a worker can complete independently, a complex task requires division into multiple subtasks. These subtasks must then be allocated to several workers for collaborative completion. In this scenario, any delay in one subtask can potentially affect the timely completion of the entire task. Researchers have explored reinforcement learning (RL) and deep RL (DRL) approaches to decide on multitask allocation. Xu et al. [10] utilized RL to optimize the allocation strategy of parallel subtasks. However, RL-based methods face dimension explosion with the increase of parallel subtasks. DRL-based schemes can handle high-dimensional action spaces for parallel subtask allocation. Xu and Song [11] designed a multiagent DRL algorithm to train a local model for each worker and obtain parallel allocation actions through multiagent cooperation. Ding et al. [12] improved proximal policy optimization (PPO), solving the unreasonable action-matching problem caused by spatiotemporal complexity by dynamically matching tasks, workers, and workplaces. However, most existing DRL-based task allocations directly or indirectly assume that the number of tasks or workers is static within a period, reducing the scheme's usability.

A. Challenging Issues and Related Works

For DRL-based MCS task allocation in a time-varying environment, many challenges remain.

Manuscript received 30 January 2024; revised 28 April 2024; accepted 5 May 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 61502230 and Grant 61501224, in part by the Natural Science Foundation of Jiangsu Province under Grant BK20201357, and in part by the Six Talent Peaks Project in Jiangsu Province under Grant RJFW-020. *(Corresponding author: Hang Shen.)*

The authors are with the College of Computer and Information Engineering (College of Artificial Intelligence), Nanjing Tech University, Nanjing 211816, China (e-mail: wangtianjing@njtech.edu.cn; 202161120001@njtech.edu.cn; hshen@njtech.edu.cn; bai@njtech.edu.cn)

Digital Object Identifier 10.1109/TCSS.2024.3398430

[1]https://www.didiglobal.com/
[2]https://www.ele.me
[3]https://www.waze.com

1) Dynamics of Parallel Subtasks and Workers: The number of parallel subtasks divided from a complex task is not constant due to the tasks' heterogeneity. Xie et al. [13] presented a multistage complex task decomposition framework that dynamically divides tasks according to knowledge-intensive types and assigns subtasks to suitable service providers. Liu and Zhao [14] developed a multiattribute E-CARGO task assignment model based on adaptive heterogeneous residual networks, considering the heterogeneous workers and tasks. However, these solutions assume the number of workers is fixed to reduce model complexity. Sun et al. [15] pointed out that online task allocation in MCS is dynamic and uncertain. They designed a spatial perception multiagent Q-learning algorithm for dynamic spatial task allocation. Liu et al. [16] proposed a distributed execution framework based on DRL to provide reliable and accurate sensing services when the number of tasks and vehicles changes. In [17], a DRL-based algorithm was designed for scheduling workflow on small time scales for task offloading in space-air-ground integrated vehicular networks. However, these methods deal with simple tasks rather than parallel subtasks. Designing a DRL model for dynamic task division and worker selection is challenging.

2) Diversity of Task Allocation Environments: Differentiated tasks require MCS systems to provide differentiated system services. Several studies have proposed multitask allocation based on DRL to address the challenges of multitask concurrency, task and worker heterogeneity, and participant preference changes. Hang et al. [18] proposed a multiagent DRL-based multitask allocation to provide differentiated sensing responses. Considering the complicated and dynamic environment of vehicular computing, Qi et al. presented a DRL-based parallel task scheduling approach [19], where the output branches of multitask learning are fine-matched to parallel scheduling. Zhao et al. [20] designed a similarity function on the task transfer graph to promote the allocation of personalized multitasks. The advantage of these on-policy algorithms is that parallel subpolicies can output personalized allocation decisions, but the samples they master have a strong correlation, leading to a weak model generalization and difficulty in service quality guaranteeing. Unlike on-policy algorithms that periodically abandon samples interacting with the environment, off-policy algorithms design an experience replay pool that stores diversified samples to enhance the model's adaptability to differentiated tasks. A double deep Q-network with a priority experience replay pool is studied in [21], planning a travel path that meets the requirements of each mobile user. Existing deep deterministic policy gradient (DDPG) [22] and twin delayed deep deterministic policy gradient (TD3) [23] can also solve task scheduling, but they operate in continuous action spaces. This is because the random sampling is not derivable in discrete action spaces, and the model cannot be trained using backpropagation. Therefore, exploring an off-policy algorithm in discrete action spaces is necessary for environment diversity to ensure service stability.

3) Long-Term Balanced Scheduling in Continuous Task Allocation: Load balancing is important to consider in achieving long-term optimized task scheduling. Several studies have proposed DRL-based task allocation, considering the long-term utility of workers and requesters. Zhao et al. [24] proposed a discrete threshold task allocation algorithm based on policy gradient considering long-term utility, significantly improving the long-term continuous task allocation utility. In [25], a multiagent DRL solution was proposed to generate a multitask allocation strategy that considers the long-term interests of workers and requesters. This scheme designs a reward function that considers local and global returns to balance short-term and long-term benefits and achieve a long-term equilibrium task completion rate. Ma et al. [26] proposed a real-time task dynamic scheduling model based on centralized learning, which makes more accurate continuous task scheduling decisions by analyzing the processor load of workers. The system's load balance is better than random task allocation methods, improving CPU utilization and service quality. However, these methods do not incorporate balance indicators into the DRL reward function, making it difficult for the model to learn to schedule experience that satisfies long-term load balancing.

B. Contributions and Organization

In response to the above issues, we proposed a twin delayed deep stochastic policy gradient (TDDS) approach for long-term balanced and low-latency task allocation via dynamic partitioning and scheduling. The main contributions include the following.

1) We construct a scalable policy network consisting of two shared linear layers to extract state features of subtasks and workers, along with a masked attention mechanism to match subtasks and workers. This network can independently infer an optimal subpolicy for each subtask, with enhanced robustness of task allocation.
2) An off-policy algorithm based on TD3 is designed, which uses Gumbel-Softmax sampling to enable TD3 to output allocation decisions of parallel subtasks in discrete action spaces. The rich samples in the experience replay pool can enhance model generalization to adapt to heterogeneous MCS environments.
3) We develop an appropriate reward function considering completion delay and load balancing. This encourages the model to learn from allocation experiences that optimize both indicators simultaneously, ensuring the long-term stability of task scheduling. Simulation results demonstrated that the proposed approach's task completion delay and environmental adaptability are better than typical DRL-based baselines.

The rest of this article is organized as follows. Section II presents the system model for MCS task partition and parallel subtask allocation. Section III proposes a parallel subtask allocation scheme based on TDDS. Section IV analyzes the evaluation results under simulation experiments. Finally, we summarize the research work in Section V. The main notations and variables are listed in Table I.

II. System Model

This section begins with an overview of MCS task partitioning and continuous subtask assignment. Then, the task partitioning and assignment are transformed into a long-term optimization problem.

TABLE I
MAIN NOTATIONS AND VARIABLES

Symbols	Definition
$a_{t,i,m,n}$	Allocation strategy for task i in time window t
$\mathcal{I}_t$	Set of tasks in time window t
$\mathcal{L}_{i,n}$	Set of incomplete subtasks for worker n upon receiving $\mathcal{U}_{i,n}$
M	Maximum number of task subdivision
$\mathcal{M}_i$	Set of subtasks in state i
$\mathcal{M}_{t,i}/M_{t,i}$	Set/Num. of subtasks for task i in time window t
N	Maximum number of workers
$\mathcal{N}_i$	Set of workers in state i
$\mathcal{N}_t/N_t$	Set/Num. of workers in time window t
$\mathcal{T}/T$	Set/Num. of time windows
$\mathcal{U}_{i,n}/U_{i,n}$	Set/Num. of task i'subtasks allocated to worker n
$u_{i,n}[j]$	The jth subtask executed in $\mathcal{U}_{i,n}$
$w_{i,n}[j]$	Delay from receiving $\mathcal{U}_{i,n}$ to the start of transmission of $u_{i,n}[j]$
$z^{\text{sen}}_{i,n}/z^{\text{tra}}_{i,n}$	Sensing/transmission time for worker n from receiving $\mathcal{U}_{i,n}$ to completing $\mathcal{L}_{i,n}$
$z^{\text{sen}}_{t,i,n}/z^{\text{tra}}_{t,i,n}$	The value of $z^{\text{sen}}_{i,n}/z^{\text{tra}}_{i,n}$ in time window t

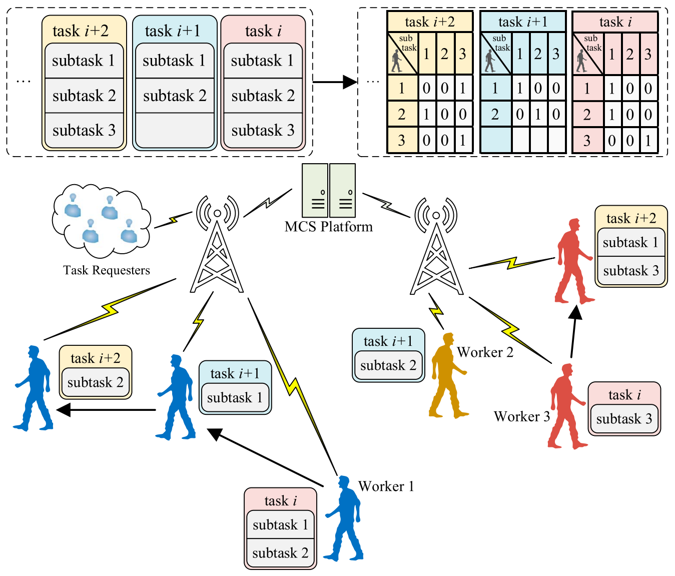

Fig. 1. Consecutive MCS task allocation.

A. System Overview

Fig. 1 illustrates an MCS system comprising a control platform, task requesters, and workers. As a dispatch center, the control platform connects the task requesters and workers via base stations. Requesters create tasks that require environmental sensing, and workers with different sensing and computing abilities cooperate to complete them. It assigns a continuous stream of tasks to a group of workers following the first-in-first-out (FIFO) rule. Each task is divided into parallel subtasks, and the control platform employs DRL to select the most suitable workers to complete these subtasks, taking into account the resource competition among the subtasks.

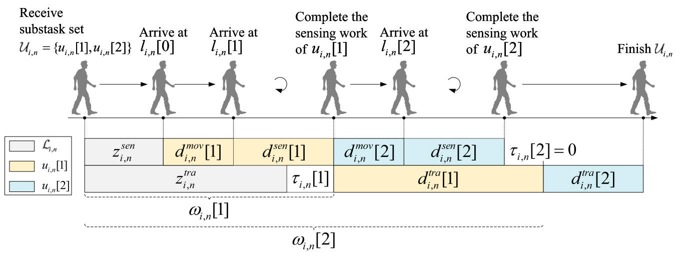

Fig. 2. Completion delay of two subtasks under FIFO.

B. Task Completion Latency Model

We now explain the allocation and execution of parallel subtasks and model the completion latency. Let $\mathcal{U}_{i,n}$ denote the set of subtasks for task i allocated to worker n, and $U_{i,n}$ indicate the total count of these subtasks. A worker movement minimization method [27] is used to determine the execution order of subtasks in $\mathcal{U}_{i,n}$. Let $u_{i,n}[j]$ represent the jth subtask executed in $\mathcal{U}_{i,n}$, and $o_{i,n}[j]$ represent the size of $u_{i,n}[j]$. $l_{i,n}[j]$ refers to the location of $u_{i,n}[j]$, which specifies the initial location of worker n prior to commencing $\mathcal{U}_{i,n}$, with $l_{i,n}[0] = l_{i-1,n}[U_{i-1,n}]$. It is assumed that the movement, sensing, and transmission rates of worker n are v^{mov}_n, v^{sen}_n, and v^{tra}_n, respectively, and their latency expressions are represented as $d^{\text{mov}}_{i,n}[j]$, $d^{\text{sen}}_{i,n}[j]$, and $d^{\text{tra}}_{i,n}[j]$

$$\begin{cases} d^{\text{mov}}_{i,n}[j] = |l_{i,n}[j] - l_{i,n}[j-1]|/v^{\text{mov}}_n \\ d^{\text{sen}}_{i,n}[j] = o_{i,n}[j]/v^{\text{sen}}_n \\ d^{\text{tra}}_{i,n}[j] = o_{i,n}[j]/v^{\text{tra}}_n. \end{cases} \tag{1}$$

Fig. 2 explains the process of worker n executing the subtasks in $\mathcal{U}_{i,n}$ when $\mathcal{U}_{i,n}$ contains two subtasks, and this can be extended to the general case. Assume $\mathcal{L}_{i,n}$ is a set of subtasks that worker n has not yet after $\mathcal{U}_{i,n}$ arrives, with the remaining sensing and transmission delays being $z^{\text{sen}}_{i,n}$ and $z^{\text{tra}}_{i,n}$, respectively. After completing the sensing of $\mathcal{L}_{i,n}$, worker n moves to $l_{i,n}[1]$ to start executing subtasks in $\mathcal{U}_{i,n}$ in sequence. Note that the delivery of a subtask must wait until the sensing of this subtask is completed and the transmission of all other subtasks before this subtask is finished before it can be executed. We define $w_{i,n}[j]$ as the delay from receiving $\mathcal{U}_{i,n}$ to the start of transmission of $u_{i,n}[j]$. The recursive expression for $w_{i,n}[j]$ is given by below equation (2), shown at the bottom of the page.

$$w_{i,n}[j] = \begin{cases} \max\{z^{\text{sen}}_{i,n} + d^{\text{mov}}_{i,n}[1] + d^{\text{sen}}_{i,n}[1], z^{\text{tra}}_{i,n}\}, & \text{if } j = 1 \\ \max\left\{ z^{\text{sen}}_{i,n} + \sum_{j'=1}^{j} (d^{\text{mov}}_{i,n}[j'] + d^{\text{sen}}_{i,n}[j']), w_{i,n}[j-1] + d^{\text{tra}}_{i,n}[j-1] \right\}, & \text{otherwise.} \end{cases} \tag{2}$$

Based on $w_{i,n}[j]$, the latency of completing all subtasks in $\mathcal{U}_{i,n}$ is $w_{i,n}[U_{i,n}] + d^{\text{tra}}_{i,n}[U_{i,n}]$. Since parallel subtasks are allocated to multiple workers for processing, the completion latency of task i is

$$d_i = \max_n \{w_{i,n}[U_{i,n}] + d^{\text{tra}}_{i,n}[U_{i,n}]\}. \tag{3}$$

C. Problem Formulation

In time window t, the group of workers is denoted as $\mathcal{N}_t$, with N_t being its cardinality. A good allocation strategy should make the sensing queue lengths of workers similar to the balanced network load. We relabel $z^{\text{sen}}_{i,n}$ in the following equation as $z^{\text{sen}}_{t,i,n}$, and define the balance index of task allocation as

$$\varphi_{t,i} \triangleq \sqrt{\frac{1}{N_t}\sum_{n\in\mathcal{N}_t}\left(z^{\text{sen}}_{t,i,n} - \frac{1}{N_t}\sum_{n\in\mathcal{N}_t} z^{\text{sen}}_{t,i,n}\right)^2}. \tag{4}$$

The smaller $\varphi_{t,i}$ is, the higher the balance of task allocation.

To evaluate the long-term task allocation in a time-varying environment, the set of time windows is defined as $\mathcal{T}$, with T being its cardinality. The set of tasks in window $t \in \mathcal{T}$ is denoted as $\mathcal{I}_t$, with I_t as its cardinality, and the set of subtasks into which task i is divided is $\mathcal{M}_{t,i}$. Let $a_{t,i,m,n} = 1$ indicate that subtask $m \in \mathcal{M}_{t,i}$ is assigned to worker n; otherwise, $a_{t,i,m,n} = 0$. Assuming that the completion delay of task i in window t is $d_{t,i}$, the task partitioning and continuous parallel subtask assignment are transformed into the following long-term optimization problem

$$\mathcal{P}1: \min \lim_{T\to\infty} \frac{1}{T}\sum_{t\in\mathcal{T}} (\zeta\mathbb{E}(d_{t,i}) + (1-\zeta)\mathbb{E}(\varphi_{t,i}))$$

$$\text{s.t.} \begin{cases} \displaystyle\sum_{n\in\mathcal{N}_t} a_{t,i,m,n} = 1, \quad \forall t\in\mathcal{T}, i\in\mathcal{I}_t, m\in\mathcal{M}_{t,i}, n\in\mathcal{N}_t & \text{(5a)} \\ a_{t,i,m,n}\in\{0,1\}, \quad \forall t\in\mathcal{T}, i\in\mathcal{I}_t, m\in\mathcal{M}_{t,i}, n\in\mathcal{N}_t & \text{(5b)} \end{cases}$$

where ζ is a weight parameter. Constraint (5a) states that one subtask is assigned to only one worker, and (5b) consists of the 0-1 decision of subtask assignment.

III. Proposed Solution

The DRL approach that supports continuous parallel subtask allocation is used to solve problem $\mathcal{P}1$. The MCS platform is abstracted as an agent, interacting with the environment at discrete time steps. During the ith interaction with the environment, the agent obtains an action a_i according to the environment state s_i and policy π_ϕ. Then, the environment transitions to the next state s_{i+1} according to the action a_i, and returns a reward r_i. The state space, action space, and reward are described as follows.

1) *State Space.* To unify input tensors' dimensions, the number of parallel subtasks is set not to exceed M. When the number is less than M, the state of all missing subtasks is filled with 0. Similarly, When the number of workers is less than N, the state of all missing workers is also filled with 0. The state of subtask m is represented by $\hat{s}_{i,m} = (l_{i,m}, o_{i,m})$, where $l_{i,m}$ and $o_{i,m}$ represent the location and size of subtask m, respectively. The state of task i is represented by the combination of the states of M subtasks

$$s^{\text{task}}_i = (\hat{s}_{i,1}, \ldots, \hat{s}_{i,M}). \tag{6}$$

The state of worker n is represented by $\tilde{s}_{i,n} = (v^{\text{mov}}_n, v^{\text{sen}}_n, v^{\text{tra}}_n, l_{i,n}[0], z^{\text{sen}}_{i,n}, z^{\text{tra}}_{i,n})$. Similarly, the state of N workers is represented by

$$s^{\text{worker}}_i = (\tilde{s}_{i,1}, \ldots, \tilde{s}_{i,N}). \tag{7}$$

Finally, the system state is composed by concatenating the task state and the worker state

$$s_i = (s^{\text{task}}_i, s^{\text{worker}}_i). \tag{8}$$

2) *Action Space.* Let $\mathcal{M}_i$ and $\mathcal{N}_i$ denote the sets of subtasks and workers at state s_i, respectively. The action space dimension of assigning $\mathcal{M}_i$ to $\mathcal{N}_i$ is at most N^M. If the number of output layer neurons of the policy network of DRL is set to N^M, the high-dimensional action space will make the learning difficult to converge. For this reason, the allocation decision of M subtasks is decomposed into M subdecisions, and the number of output layer neurons of the policy network of DRL is reduced to $M \cdot N$. For each subdecision, the action space is $\{1, 2, \ldots, N\}$, where n means that a certain subtask is assigned to worker n. The actions corresponding to the missing subtasks are ignored if the number of subtasks is less than M. Thus, the output action a_i represents assigning the M subtasks of task i to the N workers.

3) *Reward Function.* During DRL training, we give an immediate reward value $r_i = r(s_i, a_i)$ that evaluates the merits of the selected action. The goal of task allocation is to minimize the task completion delay and balance variance, while the goal of DRL is to maximize the long-term reward, so the reward function is defined as

$$r_i = \frac{\sigma_1 - (\zeta d_i + (1-\zeta)\varphi_i)}{\sigma_2} \tag{9}$$

where φ_i represents the balance index of task i allocation, and the parameters σ_1 and σ_2 are used to control the range of d_i and φ_i for the sake of DRL training.

The agent interacts with the environment and generates a sampling trajectory $\varsigma = \{s_1, a_1, r_1, \ldots, s_i, a_i, r_i, \ldots\}$ by policy π based on (8) and (9). The optimal allocation policy can be solved by maximizing the expected return of the sampling trajectory, expressed as

$$\pi^* = \max_\pi \mathcal{J}(\pi) = \mathbb{E}_{\varsigma\sim\pi}\left(\sum_{i\geq 0}\gamma^i r_{i+1}\right) \tag{10}$$

where γ is a discount factor.

Most studies use DRL of on-policy strategies such as asynchronous advantage actor–critic (A3C) [28] and PPO in task allocation [29]. Although on-policy strategies are suitable for environments where data is continuously generated, they are

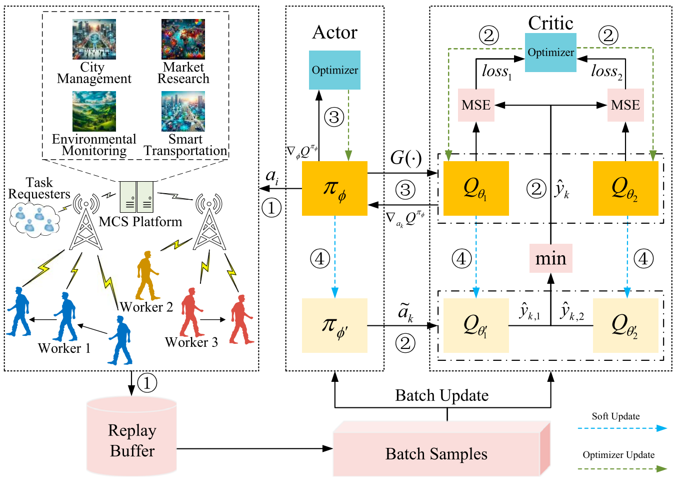

Fig. 3. TDDS model structure.

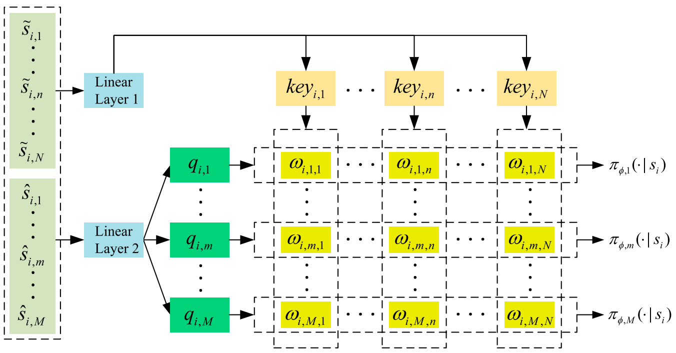

Fig. 4. Policy network for multiple subtasks.

susceptible to noise and may quickly need to remember previously learned information. The off-policy strategy shows more significant advantages in diverse environments by utilizing an experience pool to store and reuse past data, as it can learn from historical data with enhanced adaptability. TD3 is a scheduling algorithm used in continuous control. It adopts an off-policy strategy and can effectively alleviate the overestimation and high variance of the expected long-term return of a state or state-action pairs. This motivates us to apply it to parallel subtask allocation. However, converting TD3 from continuous to discrete control remains challenging.

To address this issue, we construct a twin delayed deep stochastic policy gradient (TDDS) model based on TD3, which uses two critic networks Q_{θ_1} and Q_{θ_2} and two target critic networks $Q_{\theta'_1}$ and $Q_{\theta'_2}$ with multilayer perceptron (MLP) architectures, as shown in Fig. 3. Moreover, we created an experience replay pool to enable TDDS to store samples collected by interacting with the environment.

A. Policy Network Design

Considering the inconsistent action space dimensions due to dynamic task divisions, we design the policy network π_ϕ and the target policy network $\pi_{\phi'}$ in Fig. 4 with a linear layer and an attention aggregation layer. The policy network takes the sampled state s_i as input and passes M subtask states $(\hat{s}_{i,1},\dots,\hat{s}_{i,M})$ through linear layer 1 to obtain M queries $\{q_{i,m}\}$ of dimension D. Similarly, it passes N worker states $(\tilde{s}_{i,1},\dots,\tilde{s}_{i,N})$ through linear layer 2 to obtain N keys $\text{key}_{i,n}$ of dimension D. The attention score for $q_{i,m}$ and $\text{key}_{i,n}$ is calculated as

$$\omega_{i,m,n} = \begin{cases} \dfrac{q_{i,m}\cdot \text{key}_{i,n}}{\sqrt{D}}, & \text{if } m\in\mathcal{M}_i, n\in\mathcal{N}_i \\ -\infty, & \text{otherwise.}\end{cases} \quad (11)$$

The attention weight of query m selecting key n is determined as

$$\alpha_{i,m,n} = \frac{\exp(\omega_{i,m,n})}{\sum_{n\in\mathcal{N}}\exp(\omega_{i,m,n})}. \quad (12)$$

The larger the value of $\alpha_{i,m,n}$, the higher the matching degree between subtask m and worker n. Denote the action distribution of subpolicy m as $\pi_{\phi,m}(\cdot|s_i) = (\alpha_{i,m,1},\dots,\alpha_{i,m,N})$, then the output of policy network π_ϕ is M subpolicies paired with M subtasks $\{\pi_{\phi,m}(\cdot|s_i)\}$. The attention aggregation layer can perceive the resource competition among parallel subtasks and learn how to map the state of associating one subtask with N workers to the subpolicy. Benefiting from the masked attention mechanism, the missing subtasks or workers used for padding do not affect policy network update [30].

B. Gumbel-Softmax Sampling

TDDS adapts TD3, which originally operates in the continuous action space, to work in the discrete action space by applying Gumbel-Softmax sampling. Suppose the action probability vector $p_{i,m} = (p_{i,m,1},\dots,p_{i,m,N})$ output by the subpolicy network m under the state s_i satisfies $\sum_{n\in\mathcal{N}} p_{i,m,n} = 1$. The common Gumbel-Max [31] is used to sample the discrete probability distribution $p_{i,m}$ and one-hot vector encoding is used to represent the sampled action as

$$F(p_{i,m}) = \text{one\_hot}(\arg\max_n (g_{i,m,n} + \log p_{i,m,n})) \quad (13)$$

where $g_{i,m,n} \sim \text{Gumbel}(0,1)$. Because (13) is not differentiable concerning $p_{i,m}$, so backpropagation cannot be used to update network parameters. The continuous Softmax function

$$e_{i,m,n} = \frac{\exp((g_{i,m,n} + \log p_{i,m,n})/\tau)}{\sum_{n\in\mathcal{N}}\exp((g_{i,m,n} + \log p_{i,m,n})/\tau)} \quad (14)$$

is used to approximate (13), obtaining a differentiable Gumbel-Softmax sampling

$$G(p_{i,m}) = (e_{i,m,1},\dots,e_{i,m,N}). \quad (15)$$

In (14), τ is the temperature coefficient used to control the degree of approximation of $G(p_{i,m})$ to $F(p_{i,m})$. To describe the allocation of multiple subtasks, we use $p = (p_1,\dots,p_M)$ to denote a multidimensional probability distribution, and use $G(p) = (G(p_1),\dots,G(p_M))$ to denote the Gumbel-Softmax sampling of p.

C. Model Update Strategy

The agent uses the policy network, π_ϕ, to interact with the environment and obtain samples (s_i, a_i, r_i, s_{i+1}) to fill the experience replay pool (see step ① in Fig. 3). Let s'_k represent the next state of s_k. Let $\mathcal{B}$ represent the set of random indices for B samples in the experience replay pool. When there are enough samples in the experience replay pool, the agent will randomly sample B tuples $\bigcup_{k\in\mathcal{B}}(s_k, a_k, r_k, s'_k)$, and inputs s'_k to target policy network $\pi_{\phi'}$ to obtain the action probabilities of M subpolicies $\pi_{\phi'}(\cdot|s'_k) \triangleq (\pi_{\phi',1}(\cdot|s'_k), \ldots, \pi_{\phi',M}(\cdot|s'_k))$. Each subpolicy of $\pi_{\phi'}(\cdot|s'_k)$ is sampled to obtain a concatenated action vector $\tilde{a}_k \triangleq (\tilde{a}_{k,1}, \ldots, \tilde{a}_{k,M})$. Then $\tilde{a}_k$ and s'_k are input into $Q_{\theta'_1}$ and $Q_{\theta'_2}$ respectively to obtain two temporal-difference targets $\hat{y}_{k,1}$ and $\hat{y}_{k,2}$ as

$$\hat{y}_{k,l} = r_k + \gamma Q_{\theta'_h}(s'_k, \tilde{a}_k), \qquad h \in \{1,2\}. \tag{16}$$

Let $\hat{y}_k = \min(\hat{y}_{k,1}, \hat{y}_{k,2})$ and it can be regarded as the target value of the critic networks. Mean square error (MSE) was used to establish the loss functions of critic networks Q_{θ_1} and Q_{θ_2}, expressed as

$$loss_h = \frac{1}{B}\sum_{k\in\mathcal{B}}(\hat{y}_k - Q_{\theta_h}(s_k, a_k))^2, \qquad h \in \{1,2\}. \tag{17}$$

Then the Nadam optimizer updates Q_{θ_1} and Q_{θ_2} by using the gradient of loss_1 and loss_1 with respect to θ_1 and θ_2, respectively (see step ② in Fig. 3). After the critic networks, Q_{θ_1} and Q_{θ_2}, are updated c times, the agent updates policy network π_ϕ once to ensure model training stability (see step ③ in Fig. 3).

According to the deterministic policy gradient (DPG) theorem [32], the policy gradient of the expected return in (10) is expressed as

$$\begin{aligned}\nabla_\phi \mathcal{J}(\phi) &= \mathbb{E}_{s\sim\rho^{\pi_\phi}, a\sim\pi_\phi(\cdot|s)}[\nabla_\phi Q^{\pi_\phi}(s,a)] \\ &\approx \frac{1}{B}\sum_{k\in\mathcal{B}} \nabla_\phi Q^{\pi_\phi}(s_k, a_k) \\ &= \frac{1}{B}\sum_{k\in\mathcal{B}} \nabla_{a_k} Q^{\pi_\phi}(s_k, a_k)\nabla_\phi G(\pi_\phi(\cdot|s_k)) \\ &= \frac{1}{B}\sum_{k\in\mathcal{B}} \nabla_{a_k} Q^{\pi_\phi}(s_k, a_k)\nabla_p G(p)\nabla_\phi \pi_\phi(\cdot|s_k)\end{aligned} \tag{18}$$

where ρ^{π_ϕ} denotes the discounted state distribution [32], Q^{π_ϕ} represents the state-action value function based on policy π_ϕ. In (18), $G(\pi_\phi(\cdot|s_k))$ approximates a one-hot vector, which conforms to the discrete action form. To solve $\nabla_\phi \mathcal{J}(\phi)$, TD3 uses Q_{θ_1} instead of Q^{π_ϕ} to ensure the differentiability with respect to a_k. Value function Q^{π_ϕ} can be estimated by value network Q_{θ_1} or Q_{θ_2}. Since the two networks are equivalent, the mean of the two is used to approximate the policy gradient. Accordingly, $Q^{\pi_\phi}(s_k, a_k)$ is approximated as

$$Q^{\pi_\phi}(s_k, a_k) = \frac{1}{2}(Q_{\theta_1}(s_k, a_k) + Q_{\theta_2}(s_k, a_k)) \tag{19}$$

and $\nabla_\phi \mathcal{J}(\phi)$ is used by Nadam optimizer to update policy network π_ϕ. After updating π_ϕ, the parameters of target critic networks $Q_{\theta'_2}$, $Q_{\theta'_2}$ and target policy network $\pi_{\phi'}$ are updated by (see step ④ in Fig. 3)

$$\begin{aligned}\theta'_h &\leftarrow (1-\beta)\theta'_h + \beta\theta_h, \qquad h \in \{1,2\} \\ \phi' &\leftarrow (1-\beta)\phi' + \beta\phi\end{aligned} \tag{20}$$

where $\beta \ll 1$ is the soft update factor.

Algorithm 1: TDDS-Based Parallel Subtask Allocation

```
   Input: Sampling batch $B$, soft update factor $\beta$,
          discount factor $\gamma$, policy network update period
          $c$, maximum training rounds $\mathrm{max\_epochs}$
   Output: Policy network $\pi_\phi$ for task allocation
1  Initialize critic networks $Q_{\theta_1}, Q_{\theta_2}$ as $\theta_1, \theta_2$, initialize the
     parameters of policy network $\pi_\phi$ as $\phi$, assign target
     network as $\theta'_1 \leftarrow \theta_1, \theta'_2 \leftarrow \theta_2, \phi' \leftarrow \phi$;
2  for epoch $= 1$ to $\mathrm{max\_epochs}$ do
3      Get the initial system state $s_1$;
4      $i \leftarrow 1$;
5      while $s_i$ is not the terminated state do
6          Select action $a_i \sim \pi_\phi(\cdot|s_i)$ according to the
             current policy $\pi_\phi$;
7          Execute action $a_i$, calculate reward $r_i$, system
             transitions to the next state $s_{i+1}$;
8          Put $(s_i, a_i, r_i, s_{i+1})$ into experience replay pool;
9          $i \leftarrow i + 1$;
10     Sample $B$ tuples from the experience replay pool;
11     Calculate the temporal-difference targets $\hat{y}_{k,1}$ and
         $\hat{y}_{k,2}$ according to (16);
12     $\hat{y}_k \leftarrow min(\hat{y}_{k,1}, \hat{y}_{k,2})$;
13     Calculate $loss_1$ and $loss_2$ according to (17);
14     Update the critic networks;
15     if epoch mod $c = 0$ then
16         Calculate $\nabla_\phi \mathcal{J}(\phi)$ according to (18);
17         Update the policy network;
18         Update the target networks according to (20);
19 return $\pi_\phi$
```

The execution of TDDS-based parallel subtask allocation is summarized as Algorithm 1. Initially, the parameters of the critic networks and policy network are randomly initialized and assigned to the corresponding target network (line 1). The agent then periodically interacts with the environment, collecting a variety of samples to populate the experience replay pool (lines 3–9) and updates the critic networks depending on minimizing the loss function at each time window (lines 10–14). Whenever the critic networks update c times, the policy network is updated by using the policy gradient (lines 15–17), and the target networks are updated simultaneously (line 18). This process is repeated until the TDDS training converges.

IV. Performance Evaluation

PyTorch was used as the deep learning framework to implement the proposed solution. The crowdsensing area was set to 2 × 2 km. The inter-arrival time of tasks (in minutes) follows an

TABLE II
EXPERIMENTAL PARAMETERS

Parameter	Value
Discount factor (γ)	0.8
Soft update factor (β)	0.005
Temperature coefficient (τ)	0.5
Objective function weight parameter (ζ)	0.75
Reward function parameters (σ_1,σ_2)	56, 29
Policy network update cycle (c)	8

exponential distribution with the rate parameter λ. The platform divided a task into one to eight subtasks, each with a size ranging from 0.5 to 1 GB. The number of workers varied between 8 and 16. The sensing rate (GB/min), transmission rate (GB/min), and movement rate (km/min) varied in the ranges of [0.1,0.2], [0.09,0.21], and [0.3,0.6], respectively. The experience replay pool capacity was set to 100 000. The number of policy network linear layer dimensions, D, was set to 1000. The critic networks used a $264\times1000\times1$ MLP with Prelu as the activation function. Other simulation parameters are given in Table II.

For comprehensive comparison and verification, three baseline approaches were selected and designed as follows.

1) Random assignment (RA) [33]: The agent randomly assigns one worker for each subtask.
2) PPO [34]: The agent uses PPO to perform task partition and allocation, where the policy network depends on an MLP to generate parallel subpolicies.
3) Independent deep Q-network (IDQN) [35]: Multiple agents perform parallel subtask allocation, where each agent uses one Q-network to allocate each subtask.

A. Convergence Analysis

The first experiments evaluated the convergence of TDDS under different learning rates by calculating the average cumulative reward of the policy network over multiple time windows. Sampling larger batches of tuples from the experience replay pool can ensure learning stability [36]. We set the sampling batch $B = 4096$ first.

The Nadam optimizers for critic and policy networks have the same learning rate, denoted as η. As shown in Fig. 5(a), when $\eta = 0.025$, the convergence curve of TDDS showed large oscillations, and the policy network converged to stability after 340 updates; when $\eta = 0.001$, the average cumulative reward converged slowly to 73. This showed that large or small η affected the convergence of TDDS. When η was set to 0.01 and 0.005, respectively, TDDS could balance the convergence speed and stability, and the average cumulative reward could stabilize at around 77 after 300 updates. Thus, the subsequent experiments all took $\eta = 0.01$.

Fig. 5(b) tested the effect of B on the convergence of TDDS when $\eta = 0.01$. In the case of $B = 512$, small batch sampling made the gradient estimation inaccurate and led to slow convergence speed and the convergence value was only around 64. Increasing B to 1024 and 2048, respectively, TDDS accelerated the convergence speed, but the convergence curve had slight fluctuations after reaching stability. The convergence curve of $B = 4096$ was close to that of $B = 8192$ after 170 updates. This reflected that increasing the sampling batch to a large number might not necessarily improve the convergence of TDDS and might even increase model training cost. Hence, the subsequent experiments all took $B = 4096$.

To evaluate the convergence of TDDS in detail, we extracted multiple variables in training. Because the loss of the two critic networks was almost the same, only loss_1 is given. Fig. 6(a) demonstrates that the loss_1 curve converges steadily, proving that the critic networks can accurately predict the expected return after training, providing a solid foundation for updating the policy network. In Fig. 6(b), the expected return curve rose steadily, indicating that the policy network was gradually optimized. In Fig. 6(c), the policy entropy [37] gradually decreased from 2.5, reflecting that the agent's exploratory gradually decreased and gradually tended to be stable.

B. Adaptability Analysis

Multiple indicators are extracted to evaluate the allocation driven by TDDS. We took $T = 30\,000$ for all subsequent evaluations to evaluate the applicability to different environments. Fig. 7(a) shows that the expected task completion delay $\mathbb{E}(d_{t,i})$ of TDDS was 44%, 65%, and 70% of RA, PPO, and IDQN, respectively. Let $z^{\text{tra}}_{i,n,t}$ be the value of $z^{\text{tra}}_{i,n}$ in time window t. Fig. 7(b) and 7(c) show the expected length of sensing and transmission queue $\mathbb{E}(z^{\text{sen}}_{t,i,n})$, $\mathbb{E}(z^{\text{tra}}_{t,i,n})$ were both smaller than the other three algorithms. This indicated that TDDS effectively reduced the queuing cost in executing subtasks. At the same time, in Fig. 7(d), the expected movement distance to complete a task of TDDS was 0.16 km, while RA, PPO, and IDQN were all larger than 0.37 km. This meant that TDDS could achieve the optimal matching according to the spatial information of subtasks and workers and had stronger environmental adaptability. In Fig. 7(e), the expected balance, $\mathbb{E}(\varphi_{t,i})$, of TDDS was much smaller than RA and 72% and 81% of PPO and IDQN respectively, so TDDS can provide a low-load scheduling strategy. The above experimental results show that the five indicators yielded consistent outcomes. Accordingly, we used the objective value of $\mathcal{P}1$ as a simple and effective criterion to evaluate the following simulations.

C. Impact of Task Attributes

When evaluating the average impact of a certain quantity on the objective value under different environments, we fix this quantity in all test environments and keep the rest of the variables taking values under the original distribution.

This group of experiments first considered the impact of task reaching intensity in each time window on objective value. The task inter-arrival time followed an exponential distribution with λ, so the higher the λ, the higher the task arrival rate, the more subtasks accumulated by workers, and the objective value showed a rapid upward trend, as shown in Fig. 8(a). When $\lambda = 1$, IDQN, PPO, and RA found it difficult to cope with the densely arriving tasks, and their objective values were 138, 127, and 151, respectively, while TDDS was only 94. In addition, when $\lambda = (1/5)$, the task arrival time interval was longer, so

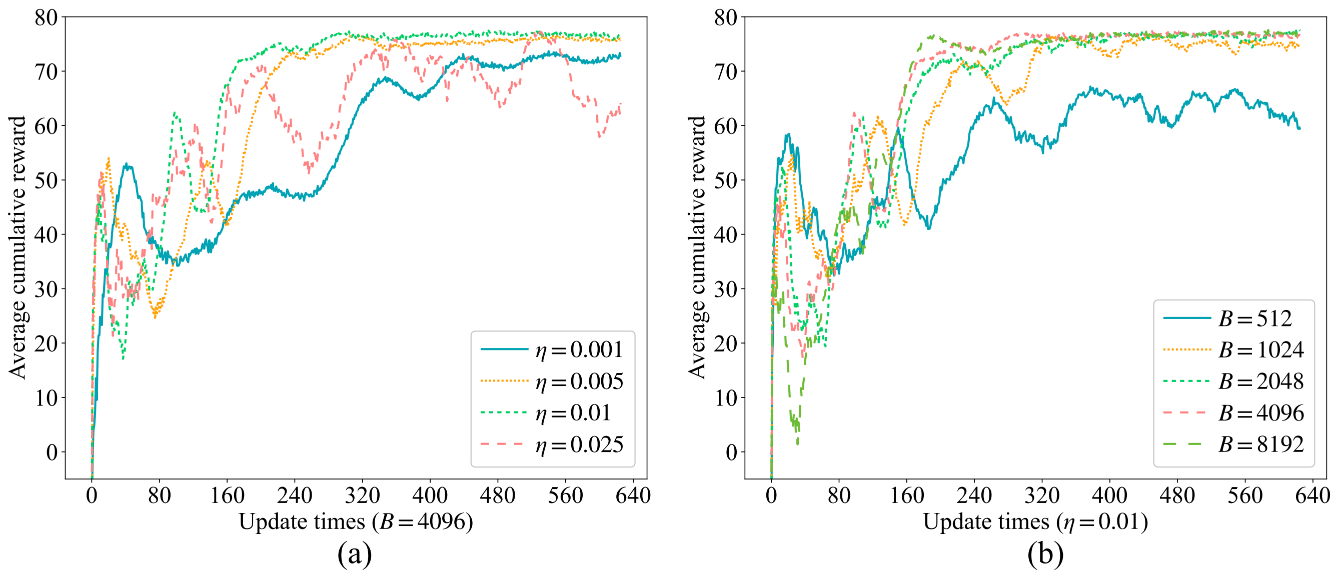

Fig. 5. Convergence curves of TDDS. (a) Convergence curves with varying learning rate. (b) Convergence curves with varying batch size.

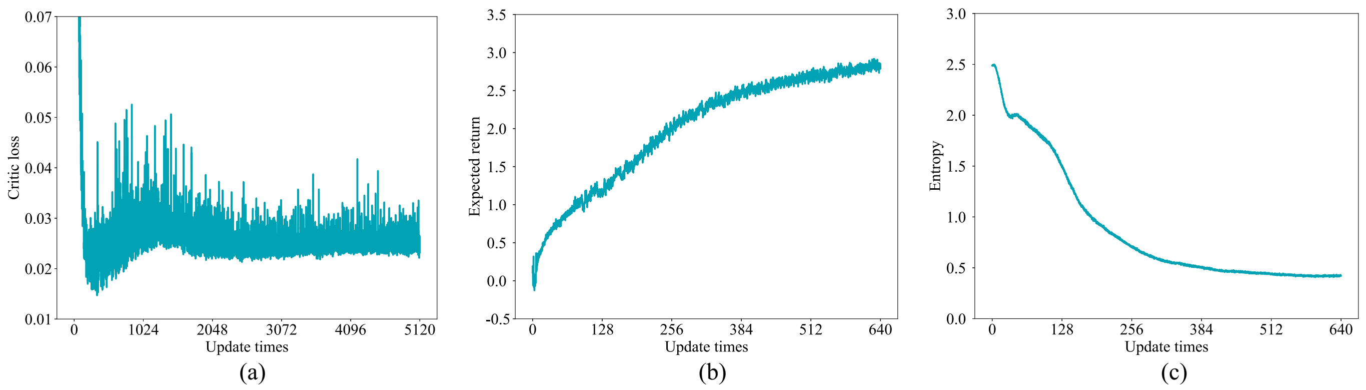

Fig. 6. Variation of loss_1, $\mathcal{J}(\phi)$ and policy entropy. (a) Variation of loss_1. (b) Variation of $\mathcal{J}(\phi)$. (c) Variation of policy entropy.

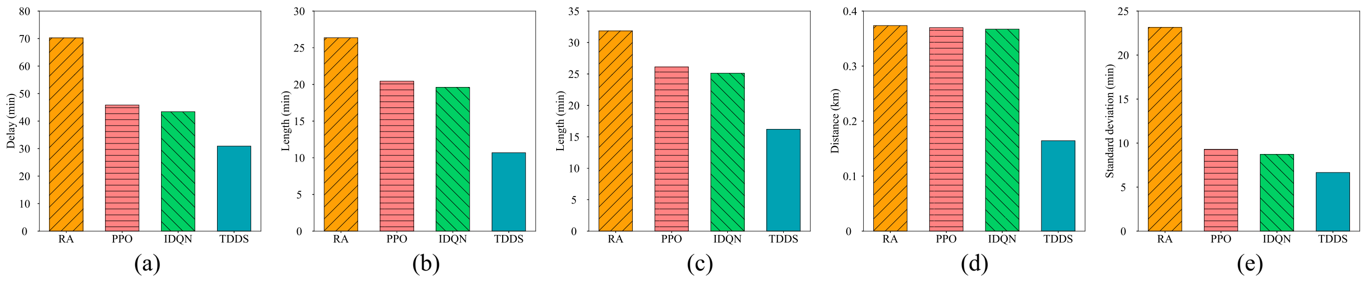

Fig. 7. Comprehensive analysis of task allocation. (a) Comparison of $\mathbb{E}(d_{t,i})$. (b) Comparison of $\mathbb{E}(z^{\text{sen}}_{t,i,n})$. (c) Comparison of $\mathbb{E}(z^{\text{tra}}_{t,i,n})$. (d) Comparison of moving. (e) Comparison of $\mathbb{E}(\varphi_{t,i})$.

the workers had relatively sufficient time to complete each subtask, and the objective values of the four algorithms were all small.

The impact of the number of tasks in each time window I_t variation on objective value was considered. When the number of tasks in the time window increased, the cumulative effect caused the unprocessed subtasks to accumulate continuously, affecting the completion delay of the subsequent arrival tasks. When $\lambda = (2/7)$, Fig. 8(b) shows the growth trend of the objective value of the four algorithms under varying task numbers, where TDDS still had the optimal allocation, and its delay growth rate was 36%. When $I_t = 500$, the objective values

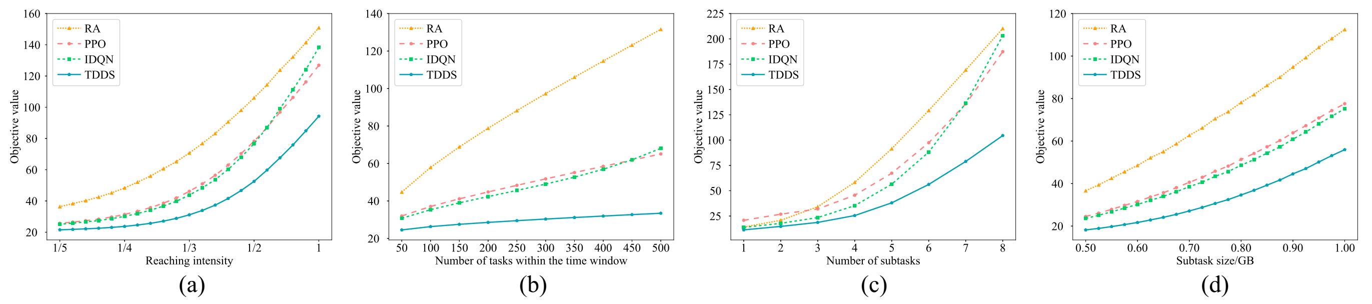

Fig. 8. Impact of task properties on objective value. (a) Impact of λ. (b) Impact of I_t. (c) Impact of $M_{t,i}$. (d) Impact of $o_{i,m}$.

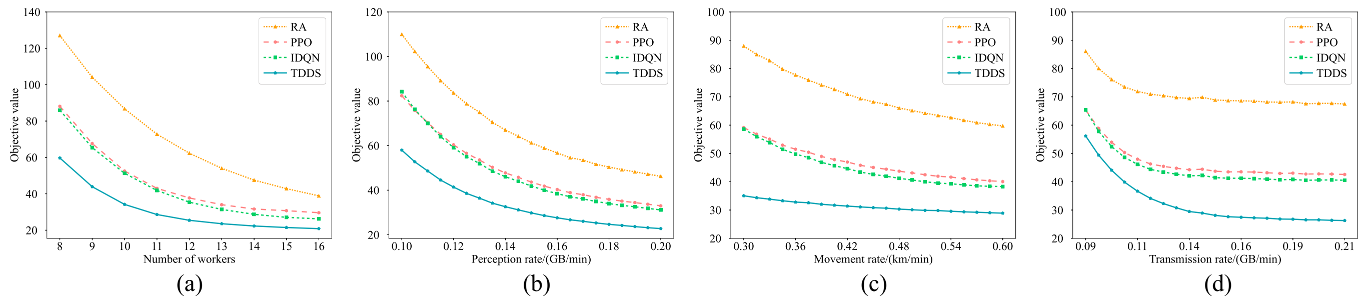

Fig. 9. Impact of worker properties on objective value. (a) Impact of N_t. (b) Impact of v_n^{sen}. (c) Impact of v_n^{mov}. (d) Impact of v_n^{tra}.

of RA, IDQN, and PPO were 132, 68, and 65, respectively, while TDDS was 33. Facing the long-term high demand of task requesters, TDDS provided the highest service quality.

Let $M_{t,i}$ be the number of subtasks from task i in time window t. As shown in Fig. 8(c), as the number of subtasks $M_{t,i}$ divided by each task gradually increased, the objective values of all algorithms showed an upward trend, but TDDS rose the slowest, and the maximum objective value was 104. When $M_{t,i} \in [1, 2]$, the low-difficulty allocation work made the task completion situation of each algorithm similar. However, the objective value of TDDS increased by 86 when $M_{t,i} \in [3, 8]$, while the other three algorithms were all over 155. Among them, IDQN was most affected by subtask number variation, and the objective value increased by 180. This was because, in IDQN, each agent tended to assign subtasks to workers with strong abilities, which might have caused most of the subtasks to be assigned to the same worker, thus delaying the completion of the entire task. Especially when $M_{t,i}$ was large, the objective value of IDQN was close to RA.

From Fig. 8(d), the objective value was positively correlated with subtask size. Randomly assigning subtasks increased the difficulty of low-ability workers in handling complex tasks, so RA's objective value was much higher than PPO, IDQN, and TDDS. PPO and IDQN output subtask allocation strategies that could usually match subtasks and workers well, so the objective value was significantly lower than RA. TDDS's attention aggregation layer further enhanced the matching degree of subtasks and workers, and its objective value was 43%–50%, 67%–74%, and 70%–77% of RA, PPO, and IDQN, respectively, when $o_{i,m} \in [0.5, 1]$.

D. Impact of Worker Attributes

As shown in Fig. 9(a), the gradually increasing workers could share more subtasks, so the objective values of the four algorithms all dropped rapidly. However, TDDS achieved the lowest objective value by using a more optimal allocation strategy, which was about 26-67 and 5-18 less than the three algorithms when there were 8 and 16 workers, respectively. This showed that TDDS had obvious advantages under different numbers of workers.

The sensing rate v_n^{sen} was limited by the ability of sensing devices carried by workers. For example, sensing devices with high-definition cameras and GPU chips could sense high-quality data faster. Fig. 9(b) shows that the faster sensing rate promoted the task completion speed of the four algorithms. At the same sensing rate, the objective value of TDDS was much smaller than the other three algorithms. For example, when $v_n^{\text{sen}} = 0.1$, the objective values of IDQN, PPO, and RA were 84, 82, and 110, respectively, while TDDS was only 58. On the other hand, the speed of worker movement also affected the allocation.

In Fig. 9(c), the faster-moving workers reach the subtask location earlier, which helps reduce the objective value. TDDS's curve was relatively flat, unlike the other three algorithms' fluctuating curves. When v_n^{mov} increased from 0.3 to 0.6, the change in the objective value of TDDS was 6, while IDQN, PPO, and RA were 20, 19, and 28, respectively. This is because TDDS's average movement distance of workers was short, which reduced the movement delay.

Workers could transmit data while sensing and moving, so data transmission did not affect the sensing of the next subtask.

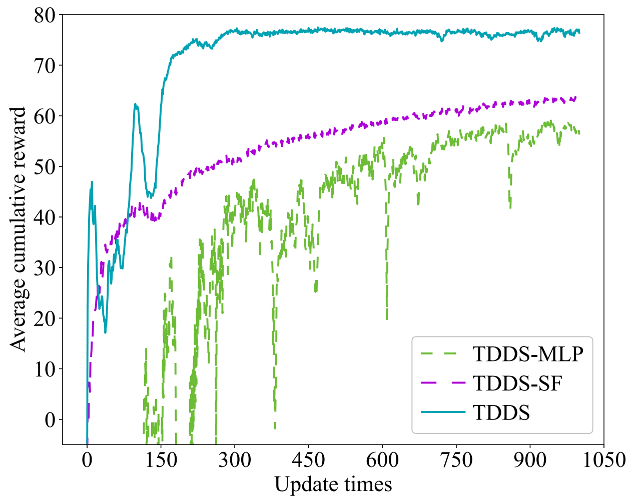

Fig. 10. Convergence curves of different models.

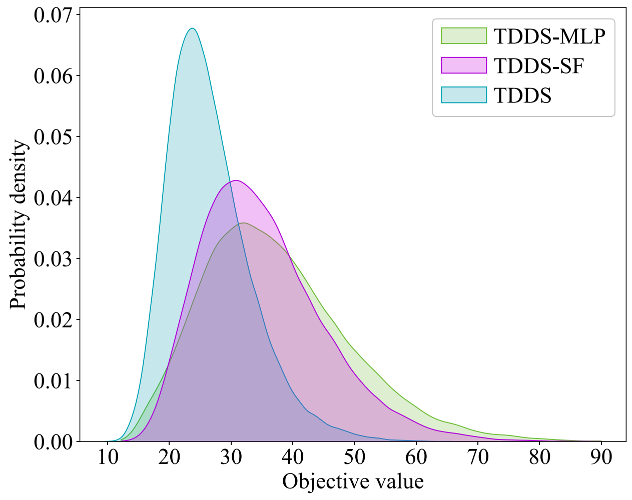

Fig. 11. Probability density of objective values.

In Fig. 9(d), the objective values of the four algorithms did not change much due to the increase of transmission rate when $v_n^{\text{tra}} \geq 0.14$, and TDDS still had the lowest objective value, which was about 62%–69% of IDQN and PPO. Fig. 9 showed that TDDS can adapt to the changes in worker attributes and continuously output better online allocation strategies than the other three algorithms.

E. Ablation Experiments

To verify the effectiveness of TDDS, two task allocation models were set up for ablation experiments.

1) TDDS-SF: The policy network uses the score function estimator (SF) to calculate gradient instead of Gumbel-Softmax sampling. In this case, the policy gradient changes according to (21), shown at the bottom of the page.
2) TDDS-MLP: The policy network does not use the attention mechanism but relies on an MLP to output eight subpolicies, with the structure as 136 $\times$ 500 $\times$ 500 $\times$ 128.

In Fig. 10, the average cumulative rewards of TDDS-SF and TDDS-MLP after convergence were 62 and 58, respectively, lower than the 77 of TDDS. At the same time, the MLP of TDDS-MLP did not easily capture the correlation between subtasks and workers, which made the curve fluctuate greatly, so the model stability needed to be improved.

From Fig. 11, the probability density curves of objective value for the three models within 30 000-time windows indicated that the objective values of TDDS-SF, TDDS-MLP, and TDDS were around 34, 36, and 26, with TDDS having the narrowest curve width. The mean and variance of each curve demonstrated that TDDS performs an optimal allocation.

To further verify the advantages of TDDS, we compared the task allocation performance of the three models under different environmental states. As shown in Fig. 12(a), the objective value range of the three models was similar (between 11 and 24) when $M_{t,i} \in [1, 3]$. However, when the number of subtasks increased to 8, the objective value of TDDS-MLP rose to 155, which was 37 and 62 higher than that of TDDS-SF and TDDS, respectively. The effect of N_t on the objective values of the three models is illustrated in Fig. 12(b). As N_t increased, the objective values of the three models decreased rapidly. However, TDDS outperformed TDDS-SF and TDDS-MLP in all cases. When $N_t \in [8, 11]$, TDDS-SF had a lower objective value than TDDS-MLP, but still 10–19 higher than TDDS. When $N_t \in [12, 16]$, the objective values of TDDS-MLP and TDDS-SF were similar (about 26–35) but 5–9 higher than TDDS. Fig. 12(c) shows the impact of I_t on the objective values. With the increase of I_t, the objective values of the three models also increased rapidly. However, TDDS had a lower objective value than TDDS-SF and TDDS-MLP in all scenarios. When $I_t = 500$, TDDS had a 36% and 48% lower objective value than TDDS-SF and TDDS-MLP.

The results in Fig. 12 demonstrate that using the score function estimator SF instead of Gumbel-Softmax sampling leads to inaccurate calculation of the policy network gradient, and using MLP instead of the attention mechanism fails to capture the correlation between subtasks and workers. As a result, TDDS-SF and TDDS-MLP's policy networks cannot optimally match subtasks and workers, which results in significantly higher objective values than TDDS under different environmental states. The proposed task allocation model used Gumbel-Softmax sampling and attention mechanism to help the policy network generate allocation strategies, with which the MCS system can

$$\nabla_\phi J(\phi) \approx \frac{1}{B} \sum_{k \in \mathcal{B}} \left(Q^{\pi_\phi}(s_k, (\tilde{a}_{k,1}, \ldots, \tilde{a}_{k,M})) \nabla_\phi \log \prod_{m=1}^{M} \pi_{\phi,m}(\cdot|s_k) \right). \quad (21)$$

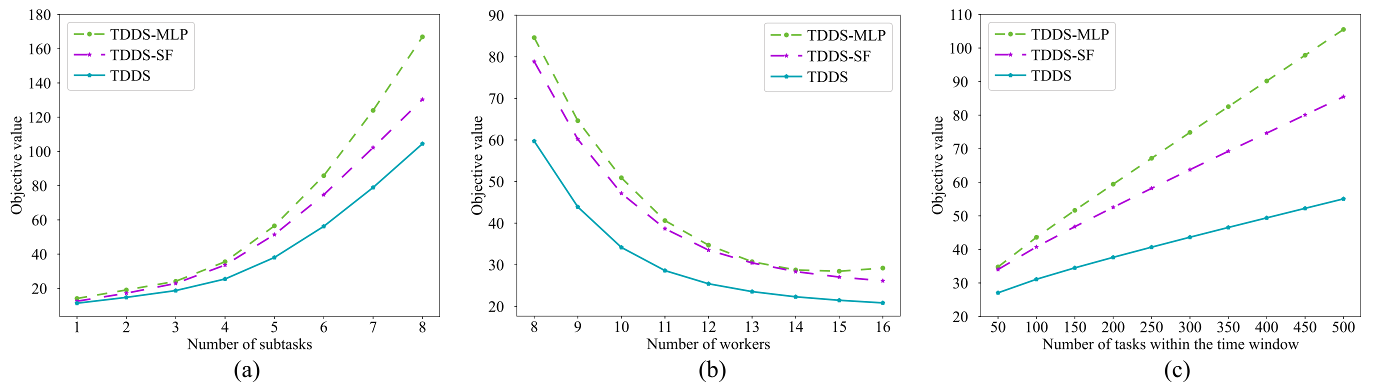

Fig. 12. Impact of worker and task's properties on objective value. (a) Impact of $M_{t,i}$. (b) Impact of N_t. (c) Impact of I_t.

meet task needs in a more timely manner while balancing the load for workers.

V. Conclusion

We have presented a TDDS-based approach for continuous parallel subtask assignment in MCS. The policy network in TDDS uses shared linear layers to reduce network parameters and introduces a masked attention mechanism to match the dynamically changing number of subtasks and workers. Considering that off-policy DRL has high sample utilization and good generalization, we introduce Gumbel-Softmax sampling so that the off-policy TD3 algorithm can be applied to discrete action spaces, and the feasibility of the proposed algorithm is proved through convergence analysis. Compared with mainstream DRL baseline algorithms, TDDS shortens the task completion delay by 30%–56% while balancing the load and reducing workers' movement distance. Regarding adapting to the dynamics of tasks and workers, TDDS performs more stably and is less affected by environmental fluctuations than other baseline algorithms. Ablation studies verify the effectiveness of masked attention and Gumbel-Softmax in TDDS.

When tasks arrive intensively, previous tasks' allocation significantly impacts subsequent tasks' allocation, and this approach may not achieve global optimality. If the offline method is integrated into online assignments, we can assign tasks after receiving multiple tasks and improve allocation efficiency by controlling when to perform task assignments, which is our follow-up research direction.

References

[1] X. Cheng, B. He, G. Li, and B. Cheng, "A survey of crowdsensing and privacy protection in digital city," *IEEE Trans. Comput. Social Syst.*, vol. 10, no. 6, pp. 3471–3487, Dec. 2023.

[2] S. Du and S. Wang, "An overview of correlation-filter-based object tracking," *IEEE Trans. Comput. Social Syst.*, vol. 9, no. 1, pp. 18–31, Feb. 2022.

[3] I. Koukoutsidis, "Estimating spatial averages of environmental parameters based on mobile crowdsensing," *ACM Trans. Sensor Netw.*, vol. 14, no. 1, pp. 1–26, 2017.

[4] Y. Gu, H. Shen, G. Bai, T. Wang, and X. Liu, "QOL-aware incentive for multimedia crowdsensing enabled learning system," *Multimedia Syst.*, vol. 26, pp. 3–16, Feb. 2020.

[5] L. Zhang, Y. Ding, X. Wang, and L. Guo, "Conflict-aware participant recruitment for mobile crowdsensing," *IEEE Trans. Comput. Social Syst.*, vol. 7, no. 1, pp. 192–204, Feb. 2020.

[6] H. Shen, G. Bai, Y. Hu, and T. Wang, "P2TA: Privacy-preserving task allocation for edge computing enhanced mobile crowdsensing," *J. Syst. Archit.*, vol. 97, pp. 130–141, 2019.

[7] Y. Tong, Z. Zhou, Y. Zeng, L. Chen, and C. Shahabi, "Spatial crowdsourcing: A survey," *VLDB J.*, vol. 29, pp. 217–250, Jan. 2020.

[8] T. Song, K. Xu, J. Li, Y. Li, and Y. Tong, "Multi-skill aware task assignment in real-time spatial crowdsourcing," *GeoInformatica*, vol. 24, pp. 153–173, Jan. 2020.

[9] H. Schmitz and I. Lykourentzou, "Online sequencing of non-decomposable macrotasks in expert crowdsourcing," *ACM Trans. Social Comput.*, vol. 1, no. 1, pp. 1–33, 2018.

[10] Y. Xu, Y. Wang, J. Ma, and Q. Jin, "PSARE: A RL-based online participant selection scheme incorporating area coverage ratio and degree in mobile crowdsensing," *IEEE Trans. Veh. Technol.*, vol. 71, no. 10, pp. 10923–10933, Oct. 2022.

[11] C. Xu and W. Song, "Decentralized task assignment for mobile crowdsensing with multi-agent deep reinforcement learning," *IEEE Internet Things J.*, vol. 10, no. 18, pp. 16564–16578, Sep. 2023.

[12] W. Ding, Z. Ming, G. Wang, and Y. Yan, "System-of-systems approach to spatio-temporal crowdsourcing design using improved PPO algorithm based on an invalid action masking," *Knowl. Based Syst.*, vol. 285, 2024, Art. no. 111381.

[13] S. Xie, X. Wang, B. Yang, M. Long, J. Zhang, and L. Wang, "A multi-stage framework for complex task decomposition in knowledge-intensive crowdsourcing," in *Proc. IEEE Int. Conf. Ind. Eng. Eng. Manage. (IEEM)*, 2021, pp. 1432–1436.

[14] Z. Liu and Z. Zhao, "Multiattribute E-CARGO task assignment model based on adaptive heterogeneous residual networks," *IEEE Trans. Comput. Social Syst.*, early access, doi: 10.1109/TCSS.2023.3344173.

[15] Y. Sun, J. Wang, and W. Tan, "Dynamic worker-and-task assignment on uncertain spatial crowdsourcing," in *Proc. IEEE Int. Conf. Comput. Supported Cooperative Work Des. (CSCWD)*, 2018, pp. 755–760.

[16] C. H. Liu, Z. Dai, H. Yang, and J. Tang, "Multi-task-oriented vehicular crowdsensing: A deep learning approach," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2020, pp. 1123–1132.

[17] H. Shen, Y. Tian, T. Wang, and G. Bai, "Slicing-based task offloading in space-air-ground integrated vehicular networks," *IEEE Trans. Mobile Comput.*, early access, doi: 10.1109/TMC.2023.3283852.

[18] J. Han, Z. Zhang, and X. Wu, "A real-world-oriented multi-task allocation approach based on multi-agent reinforcement learning in mobile crowd sensing," *Information*, vol. 11, no. 2, 2020, Art. no. 101.

[19] Q. Qi et al., "Scalable parallel task scheduling for autonomous driving using multi-task deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 13861–13874, Nov. 2020.

[20] B. Zhao, H. Dong, and D. Yang, "A spatio-temporal task allocation model in mobile crowdsensing based on knowledge graph," *Smart Cities*, vol. 6, no. 4, pp. 1937–1957, 2023.

[21] X. Tao and A. S. Hafid, "DeepSensing: A novel mobile crowdsensing framework with double deep Q-network and prioritized experience replay," *IEEE Internet Things J.*, vol. 7, no. 12, pp. 11547–11558, Dec. 2020.

[22] L. Li, H. Xu, J. Ma, A. Zhou, and J. Liu, "Joint EH time and transmit power optimization based on DDPG for EH communications," *IEEE Commun. Lett.*, vol. 24, no. 9, pp. 2043–2046, Sep. 2020.

[23] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1587–1596.

[24] B. Zhao, H. Dong, Y. Wang, and T. Pan, "PPO-TA: Adaptive task allocation via proximal policy optimization for spatio-temporal crowdsourcing," *Knowl. Based Syst.*, vol. 264, 2023, Art. no. 110330.

[25] P. Zhao, X. Li, S. Gao, and X. Wei, "Cooperative task assignment in spatial crowdsourcing via multi-agent deep reinforcement learning," *J. Syst. Archit.*, vol. 128, 2022, Art. no. 102551.

[26] Y. Ma, Z. Bi, Z. Yin, and A. Chai, "Research and implementation of a real-time task dynamic scheduling model based on reinforcement learning," in *Proc. Int. Conf. Intell. Comput. Technol. Automat. (ICICTA)*, 2020, pp. 717–722.

[27] A. Bjorklund, "Determinant sums for undirected Hamiltonicity," *SIAM J. Comput.*, vol. 43, no. 1, pp. 280–299, 2014.

[28] M. Min et al., "Geo-perturbation for task allocation in 3-D mobile crowdsourcing: An A3C-based approach," *IEEE Internet Things J.*, vol. 11, no. 2, pp. 1854–1865, Jan. 2024.

[29] J. Jin and Y. Xu, "Optimal policy characterization enhanced proximal policy optimization for multitask scheduling in cloud computing," *IEEE Internet Things J.*, vol. 9, no. 9, pp. 6418–6433, May 2022.

[30] S. Huang and S. Ontañón, "A closer look at invalid action masking in policy gradient algorithms," in *Proc. Int. FLAIRS Conf. Proc.*, vol. 35, May 2022.

[31] C. J. Maddison, D. Tarlow, and T. Minka, "A* sampling," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, pp. 3086–3094, 2014.

[32] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 387–395.

[33] D. Li, J. Zhu, and Y. Cui, "Prediction-based task allocation in mobile crowdsensing," in *Proc. Int. Conf. Mobile Ad-Hoc Sensor Netw. (MSN)*, 2019, pp. 89–94.

[34] J. Jin and Y. Xu, "Optimal policy characterization enhanced proximal policy optimization for multitask scheduling in cloud computing," *IEEE Internet Things J.*, vol. 9, no. 9, pp. 6418–6433, May 2022.

[35] A. Tampuu et al., "Multiagent cooperation and competition with deep reinforcement learning," *PLoS One*, vol. 12, no. 4, 2017, Art. no. e0172395.

[36] T. Ben-Nun and T. Hoefler, "Demystifying parallel and distributed deep learning: An in-depth concurrency analysis," *ACM Comput. Surveys*, vol. 52, no. 4, pp. 1–43, 2019.

[37] C. E. Shannon, "A mathematical theory of communication," *Bell System Tech. J.*, vol. 27, no. 3, pp. 379–423, 1948.

Tianjing Wang (Member, IEEE) received the B.Sc. degree in mathematics from Nanjing Normal University, Nanjing, China, in 2000, the M.Sc. degree in mathematics from Nanjing University, Nanjing, China, in 2002, and the Ph.D. degree in signal and information system from Nanjing University of Posts and Telecommunications (NUPT), Nanjing, China, in 2009.

From 2011 to 2013, she was a Full-Time Postdoctoral Fellow with the School of Electronic Science and Engineering, NUPT. From 2013 to 2014, she was a Visiting Scholar with the Department of Electrical and Computer Engineering at the State University of New York, Stony Brook, NY, USA. She is an Associate Professor with the Department of Communication Engineering, Nanjing Tech University, Nanjing, China. Her research interests include mobile crowdsensing, cellular V2X communication networks, and distributed machine learning for multimedia networking. She has published research papers in prestigious international journals and conferences, including IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON BROADCASTING, *Journal of Systems Architecture*, *Multimedia Systems*, *Peer-to-Peer Networking and Applications*, *IEEE ICC*, and *IEEE ISCC*.

Yu Zhang received the B.M. degree in engineering management from Beijing University of Civil Engineering and Architecture, Beijing, China. He is currently working toward the M.S. degree in computer science with Nanjing Tech University, Nanjing, China.

His research interests include combinatorial optimization, deep reinforcement learning, and its applications in crowdsensing.

Hang Shen (Member, IEEE) received the Ph.D. degree (with honors) in computer science from Nanjing University of Science and Technology, in 2015.

He worked as a Full-Time Postdoctoral Fellow with the Broadband Communications Research (BBCR) Lab, ECE Department, University of Waterloo, Waterloo, ON, Canada, from 2018 to 2019. He is an Associate Professor with the Department of Computer Science and Technology, Nanjing Tech University, Nanjing, China. His research interests involve mobile crowdsensing, vehicular networks, cybersecurity, and privacy computing.

Dr. Shen serves as an Associate Editor for *Journal of Information Processing Systems* and IEEE ACCESS. He was a Guest Editor for the *Peer-to-Peer Networking and Applications* and a TPC member of the 2021 Annual International Conference on Privacy, Security and Trust (PST). He is a Senior Member of CCF and an Executive Committee Member of the ACM Nanjing Chapter.

Guangwei Bai received the B.Eng. and M.Eng. degrees in computer engineering from Xi'an Jiaotong University, Xi'an, China, in 1983 and 1986, respectively, and the Ph.D. degree in computer science from the University of Hamburg, Hamburg, Germany, in 1999.

From 1999 to 2001, he worked as a Research Scientist with the German National Research Center for Information Technology, Germany. In 2001, he joined the University of Calgary, Calgary, AB, Canada, as a Research Associate. Since 2005, he has been working as a Professor in computer science with Nanjing Tech University, Nanjing, China. From October to December 2010, he was a Visiting Professor with the ECE Department at the University of Waterloo, Waterloo, ON, Canada. His research interests include architecture and protocol design for future networks, QoS provisioning, cybersecurity, and privacy computing. He has authored and coauthored more than 70 peer review papers in international journals and conferences including IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON BROADCASTING, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, *Performance Evaluation*, *Ad Hoc Networks*, *Journal of Systems Architecture*, *Multimedia Systems*, *Computer Communications*, IEEE ICC, and IEEE LCN.

Dr. Bai is an ACM member and a CCF Distinguished Member.