When User-Centric Network Meets Mobile Edge Computing: Challenges and Optimization Research Proposal

Langtian Qin, Hancheng Lu, and Feng Wu

August 17, 2022

1 System Model

1.1 Network Model

In this paper, we assume there are M single antenna users and N single antenna APs in UCMEC system, the user set and AP set are represented by $\mathcal{M} = \{1, 2, ..., m, ..., M\}$, $\mathcal{N} = \{1, 2, ..., n, ..., N\}$, respectively. For user m, $m \in \mathcal{M}$, it has a computation intensive task L_m. All APS are connected to a CPU integrated with MEC server through a reliable and unlimited bandwidth wired backhaul. Among them, the AP is only responsible for receiving and transmitting signals, while the CPU performs active user detection (AUD), channel estimation (CE) and calculation processing of user offloading tasks. Each user can choose to use the local limited computing resource for task calculation, or offload the tasks to the CPU with more powerful computing resource for calculation. We assume that the local computing resource of user$m$ is C_m^{l} (in CPU-cycle frequency), while the computing resource of MEC server is C_{e}.

1.2 Channel and Communication Model

We assume that all communications are carried out in the same frequency band, and the UCMEC system operates in a time division multiplexed (TDD) mode. All users can send orthogonal pilot sequences at the same time. Assume $g_{m,n}$ is the channel between user m and AP n, which remains unchanged in the coherence time and satisfies the following formula

$$g_{m,n} = \sqrt{\beta_{m,n}} h_{m,n}, \tag{1}$$

where $\beta_{m,n}$ is the scalar coefficients of channel shadow fading (large scale fading) between user m and AP n, $h_{m,n}$is the coefficient of small scale channel fading between user m and AP n, which satisfies $h_{m,n} \sim \mathcal{CN}(0, 1)$. Similar to [1], $\beta_{m,n}$ can be expressed by

$$\beta_{m,n} = 10^{\frac{\mu_{m,n}}{10}} 10^{\frac{\sigma_s \theta_{m,n}}{10}}, \tag{2}$$

where $\mu_{m,n}$ is the path loss (in dB) between user m and AP n , $10^{\frac{\sigma_s \theta_{m,n}}{10}}$ is the shadow fading with standard deviation σ_s between user m and AP n, $\theta_{m,n}$ is the shadow fading coefficient between user m and AP n. We use the following three slope model[2] to describe the path loss

$$\begin{cases} -L-35\log_{10}(d_{m,n}), & d_{m,n}>d_1 \\ -L-10\log_{10}(d_1^{1.5}d_{m,n}^2), & d_0<d_{m,n}\le d_1 \\ -L-10\log_{10}(d_1^{1.5}d_0^2), & d_{m,n}<d_0, \end{cases} \tag{3}$$

where $d_{m,n}$ is the distance (in km) between user m and AP n, d_0 and d_1 are the breakpoints of three slope model. L is given by

$$L = 46.3+33.9\log_{10}(f)-13.82\log_{10}(h_a)-[1.11\log_{10}(f)-0.7]h_u+1.56\log_{10}(f)-0.8, \tag{4}$$

where f is the carrier frequency (in MHz), h_{a} and h_{u} indicates the antenna heights of the AP and the user, respectively. Since there are obstacles between the AP and the user in the real environment, through the correction of shadow fading[3], $\theta_{m,n}$ consists of two parts

$$\theta_{m,n} = \sqrt{\delta}a_n+\sqrt{1-\delta}b_m, m=1,2,...,M, n=1,2,...,N, \tag{5}$$

where a_n 和 b_m obey Gaussian distribution$\mathcal{N}(0,1)$, $0\le\delta\le1$ is a parameter.

Similar to [4], We assume that at the beginning of each coherence time t_{c}, user m need to randomly choose a pilot symbol from pilot sequence$\left\{\lambda_1,\lambda_2,...,\lambda_\tau,...,\lambda_{\tau_p}\right\}\in \mathrm{C}^{\tau\times1}$ and transmit to the AP cluster for channel estimation, where $||\lambda_\tau||^2=1$. the pilot signals are orthogonal and have a period of t_{p}. We can separate users into τ_p sets $S_1,S_2,...,S_{\tau_p}$, Users in set $S_\tau$can transmit λ_τ pilot symbol. Let $I_m,\forall m\in\mathcal{M}$ represents the index of pilot symbol transmited by user m, and we have $I_m\in\{1,2,...,\tau_p\}$. We assume that the receiver at the AP uses large scale fading decoding (LSFD) [5] for signal detection. Each AP first estimates the received signal locally, and then transmits the result to the CPU through fronthaul. The CPU linearly integrates all received local estimates through LSFD vectors to recover the data. Therefore, the signal received by AP n can be given by

$$y_n=\sum_{n=1}^{N}\sqrt{p_m^{\mathrm{d}}}g_{m,n}x_m+z_m, \tag{6}$$

where x_m is the signal transmit by user m, p_m^{d} is the payload symbol transmit power of user m, and we have $p_{\min}\le p_m^{\mathrm{d}}\le p_{\max}$. $z_m\sim\mathcal{CN}(0,1)$ is the AWGN. According to [1], AP n can estimate the channel by $\hat{g}_{m,n}\sim\mathcal{CN}(0,\gamma_{m,n})$, where $\gamma_{m,n}=\frac{t_{\mathrm{p}}p_m^{\mathrm{p}}\beta_{m,n}^2}{1+t_{\mathrm{p}}p_m^{\mathrm{p}}\beta_{m,n}}$, and p_m^{p} is the power of user m when transmitting pilot symbol. m (We assume that the user sends the pilot signal with the maximum transmit power, i.e., $p_m^{\mathrm{p}}=p_{\max}$)

In UCMEC, each user will be served by a specific AP cluster. In order to save the backhaul bandwidth, we assume that the cluster size is s, and the AP cluster is divided according to the channel quality $g_{m,n}$ in descending order (i.e., take the first sAPs with good channel quality). For user m, its AP cluster is represented by Ω_m. For AP n, the set of all users is represented by Φ_n.

The user's access to the AP clusters are coordinated by adopting the orthogonal frequency division multiple access (OFDMA) technique[6]. After each AP performs local estimation, the CPU uses the LSFD vector $v_m=[v_{m,1},v_{m,2},...,v_{m,N}]^T$ for decoding, thus the SINR of user m is

$$SINR_m=\frac{p_m^{\mathrm{d}}\left(\sum_{n\in\Omega_m}v_{m,n}\gamma_{m,n}\right)^2}{\sum_{i=1}^{M}p_m^{\mathrm{d}}\sum_{n\in\Omega_m}v_{m,n}^2\gamma_{m,n}\beta_{i,n}+\sum_{n\in\Omega_m}v_{m,n}^2\gamma_{m,n}}. \tag{7}$$

To ensure that each user has the maximum SINR, the optimal LSFD vector[7] of user m is given by

$$\boldsymbol{v}_m = \left(p_m^p \sum_{i \in S_{b_m} \setminus \{m\}} \boldsymbol{\gamma}_m \boldsymbol{\gamma}_m^H + \Lambda_m \right)^{-1} \boldsymbol{\gamma}_m, \tag{8}$$

where

$$\boldsymbol{\Lambda}_m = \mathrm{diag}\left(\mathrm{p_m^p} \sum_{\mathrm{m=1}}^{\mathrm{M}} \boldsymbol{\gamma}_{\mathrm{m}} \boldsymbol{\beta}_{\mathrm{m}} + \boldsymbol{\gamma}_{\mathrm{m}} \right) \tag{9}$$

where $\boldsymbol{\gamma}_m = \{\gamma_{m,1}, \gamma_{m,2}, ..., \gamma_{m,n}\}, \forall m \in \mathcal{M}$, $\boldsymbol{\beta}_m = \{\beta_{m,1}, \beta_{m,2}, ..., \beta_{m,n}\}, \forall m \in \mathcal{M}$, $\boldsymbol{v}_m = \{v_{m,1}, v_{m,2}, ..., v_{m,n}\}, \forall m \in \mathcal{M}$.

Therefore, the reachable rate of user m is

$$r_m = B_0 \log_2(1 + SINR_m) \tag{10}$$

where B_0 is the transmission bandwidth.

1.3 Offloading Model

We assume the computing task $Task_m$ of user m, $m \in \mathcal{M}$ can be represented by a triple, i.e., $Task_m = \{D_m, \rho_m, \tau_m\}$, which represent the data size (in bits), the computing density (in CPU cycles per bit) of the task and the maximum tolerable delay (in second) of the task, respectively. Users can choose to calculate the tasks locally or offloading the tasks to the MEC server on the CPU through AP clusters. We consider the partition offloading ion this paper, and use a continuous variable $a_m \in [0, 1]$ to represent the offloading decision of user m. The a_m part of the task will be offloaded to MEC, and the remaining $1 - a_m$ part of the task is processed locally. Since the downlink data is relatively small, we ignore the overhead of downlink transmission process.

1.3.1 Local Computing Model

We can obtain the delay of user m for local processing by

$$T_m^l = (1 - a_m) \frac{D_m \rho_m}{C_m^{\mathrm{l}}} \tag{11}$$

1.3.2 Edge Computing Model

After the user sends an offloading request, the user will first send data signals to the AP cluster. Therefore, the transmission delay of user m is

$$T_m^t = a_m \frac{D_m}{r_m} \tag{12}$$

We do not consider the overhead of fronthaul transmission in this paper. We assume that the computing resource provided by the MEC server for user m is c_m, thus the computing delay of user m when offloading to the MEC is

$$T_m^e = a_m \frac{D_m \rho_m}{c_m} \tag{13}$$

Thus the offloading delay of user m can be given by

$$T_m^o = T_m^t + T_m^e. \tag{14}$$

Thus the total delay of user m is

$$T_m = max\left\{T_m^l, T_m^o\right\}. \tag{15}$$

1.4 Performance Analysis

1.4.1 Energy Consumption Analysis

The energy consumption of user m when computing the task locally is

$$E_m^l = \eta(1 - a_m)D_m\rho_m\left(C_m^l\right)^2 \tag{16}$$

where η is the effective switched capacitance

Since the user needs to send signals to its AP cluster, the energy consumption of the user m for data transmission is

$$E_m^t = \sum_{n\in\Omega_m} p_m^d T_m^t \tag{17}$$

The energy consumed by the user m when waiting for the MEC to process its task is

$$E_m^e = p_m^i T_m^e \tag{18}$$

where p_m^i is the operating power of the user in the idle state. Therefore, the total energy consumed by user m is

$$E_m = E_m^l + E_m^t + E_m^e \tag{19}$$

1.4.2 Successful Transmission Probability (STP) Analysis

According to [8], the successful transmission of user m means that the uplink SINR of the user is greater than a threshold ζ. Thus the STP can be defined as the proportion of users with successful transmission.

1.4.3 Successful Computing Probability (SCP) Analysis

According to [8], the successful computation of user m means that the total delay is less than the delay threshold of the calculation task τ_m. Thus the SCP can be defined as the proportion of users with successful computation.

In addition, we can also define the user wit successful offloading as the user who satisfies both successful transmission and computation.

1.5 Problem Formulation

Our optimization goal is to minimize the total energy consumption of all users in the network. Assume that the offloading decision matrix of all users is $\boldsymbol{A} = \{a_m, m \in \mathcal{M}\}$; The transmit power

decision matrix of all users is $\boldsymbol{P} = \left\{p_m^{\mathrm{d}}, m \in \mathcal{M}\right\}$. The computing resource allocation matrix is $\boldsymbol{C} = \{c_m, m \in \mathcal{M}\}$. The optimization problem can be formulated as

$$\mathrm{P1}: \quad \min_{\boldsymbol{A},\boldsymbol{P},\boldsymbol{C}} \sum_{m=1}^{M} E_m \tag{20a}$$

$$s.t. \quad \sum_{m}^{\mathcal{M}} c_m \leq C_e, \tag{20b}$$

$$p_{\min} \leq p_m^{\mathrm{d}} \leq p_{\max}, \forall m \in \mathcal{M}, \tag{20c}$$

$$T_m \leq \tau_m, \forall m \in \mathcal{M}, \tag{20d}$$

$$0 \leq c_m \leq C_e, \tag{20e}$$

$$a_m \in [0,1], \forall n \in \mathcal{N}, \forall m \in \mathcal{M}, \tag{20f}$$

where (20b) represents that the amount of computing resource allocated by the MEC server cannot exceed the computing resource capacity. (20c) represents the transmit power of the user should be within a range; (20d) represents the total delay of the user cannot exceed the maximum tolerable delay of the task. (20e) indicating the the feasible region of allocated computing resource. (20f) represents the feasible region of the offloading decision variable. Problem (20) is a nonlinear problem with coupling between variables. In addition, the optimization objective is non convex and the problem can be proved to be NP-hard.

2 Algorithm Design

In this paper, we use the proximal policy optimization (PPO)-based DRL algorithm to solve the above optimization problem. First, we give the Markov decision process (MDP) form of the optimization problem, and then give the derivation of PPO-based DRL algorithm and the corresponding pseudo code.

We can divide a coherent time into several discrete time slots $\mathcal{T} = \{1, 2, ..., t, ..., T\}$ and the length of each slot is much less than the coherence timet_c. Each time slot is the minimum time unit for agent to make decisions. At the beginning of each time slot, the agent observes the environment of the ucmec system. We can consider the user's offloading process in ucmec as an MDP, which includes the following elements$\{\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma\}$。 In time slot t, $\mathcal{S} = \{\boldsymbol{s}(t), \forall t \in \mathcal{T}\}$ is the state set, $\mathcal{S} = \{\boldsymbol{a}(t), \forall t \in \mathcal{T}\}$ is the action set; $\mathcal{P} = \{P(s(t+1)|s(t), a(t)), \forall t \in \mathcal{T}\}$ is the state transition probability set, which can describe the dynamic changes of the system. $\mathcal{R} = \{r(t+1)|s(t), a(t)), \forall t \in \mathcal{T}\}$ is the reward set, which can be used to describe the reward received by the agent for doing a certain action in a certain state. $\gamma \in [0,1]$ is the future discount factor, which is used for describing the weight of future reward to the total reward. A larger γ means that the agent is more concerned about the reward in the future, otherwise it can be considered short-sighted. In time slot t, the agent obtains the state set by observing the environment $\boldsymbol{s}(t)$, and make an action $a(t)$, then the environment will move to the next state according to the state transition probability $s(t+1)$, and give corresponding rewards to the agent $R(t)$. The above process will be repeated until the reward coverge to a relatively maximum value. At this time,

we can regard the optimal strategy is obtained. Next, we will give the definitions of observation set, action set and reward set, and give the policy optimization objective of the agent.

- **State Space.** The state space in time slot t can be defined as $\boldsymbol{s}(t) \triangleq \{r_m(t), \forall t \in \mathcal{T}\}$
- **Action Space.** The action space in time slot t can be defined as $\boldsymbol{a}(t) \triangleq \left[a_m(t), p_m^d(t), c_m(t) \forall m \in \mathcal{M}, \forall t \in \mathcal{T}\right]$
- **Reward Space.** After the agent makes actions, the system will move to the next state $\boldsymbol{s}(t+1)$. In each time slot, the agent will obtain a reward according to the current state and the action it taken. We assume the reward in time slot t as $r(t) = -\kappa_1 * \sum_{m=1}^{M} E_m + \kappa_2 * \sum_{m=1}^{M}(\tau_m - T_m) + \kappa_3 * C_e - \sum_{m=1}^{M}(c_m)$, where $\kappa_1, \kappa_2, \kappa_3$ are adjustable parameters.

We assume that the policy of the agent is a variable about the parameter $\boldsymbol{\theta}$, which is represented by $\pi(\theta)$. We can get the optimization objectives of agent as follows

$$\begin{aligned} \boldsymbol{\theta}^* &= \text{argmax}_{\boldsymbol{\theta}} L(\pi(\boldsymbol{\theta})) \\ &= \text{argmax}_{\boldsymbol{\theta}} \mathbb{E}\left[V_{\pi(\boldsymbol{\theta})}(\boldsymbol{s})\right] \\ &= \text{argmax}_{\boldsymbol{\theta}} \mathbb{E}\left[Q_{\pi(\boldsymbol{\theta})}(\boldsymbol{s}, \boldsymbol{a})\right], \end{aligned} \tag{21}$$

where

$$V_{\pi(\theta)}(\boldsymbol{s}) = \mathbb{E}\left[R(t) | \boldsymbol{s}(t) = \boldsymbol{s}\right], \tag{22}$$

$$Q_{\pi(\theta)}(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}\left[R(t) | \boldsymbol{s}(t) = \boldsymbol{s}, \boldsymbol{a}(t) = \boldsymbol{a}\right], \tag{23}$$

$$R(t) = \sum_{j=t}^{T} \gamma^{j-t} r(t). \tag{24}$$

In the above formula, $V_{\pi(\theta)}$ is the value function for observation, $Q_{\pi(\theta)}$ is the value function for observation and action. $R(t)$ is the discounted expected future award of user m in time slot t.

We assign an actor and a critical neural network to the agent and use PPO algorithm to train the agent. According to [9], the policy gradient of the agent can be calculated by

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} L(\pi(\boldsymbol{\theta})) &= \mathbb{E}_{\pi(\boldsymbol{\theta})}\left[\nabla_{\boldsymbol{\theta}} \text{log}_{\pi(\boldsymbol{\theta})} \text{Q}_{\pi(\theta)}(\boldsymbol{s}, \boldsymbol{a})\right] \\ &= \mathbb{E}_{\pi(\boldsymbol{\theta})}\left[\nabla_{\boldsymbol{\theta}} \text{log}_{\pi(\boldsymbol{\theta})} \text{A}_{\pi(\boldsymbol{\theta})}(\boldsymbol{s}, \boldsymbol{a})\right] \\ &\approx \mathbb{E}_{\pi(\hat{\boldsymbol{\theta}})}\left[f \nabla_{\boldsymbol{\theta}} \text{log}_{\pi(\boldsymbol{\theta})} \text{A}_{\pi(\boldsymbol{\theta})}(\boldsymbol{s}, \boldsymbol{a})\right], \end{aligned} \tag{25}$$

where $f = \frac{\pi(\boldsymbol{\theta})}{\pi(\hat{\boldsymbol{\theta}})}$,$A_{\pi(\boldsymbol{\theta})}(\boldsymbol{s}, \boldsymbol{a}) = Q_{\pi(\theta)}(\boldsymbol{s}, \boldsymbol{a}) - V_{\pi(\theta)}(\boldsymbol{s}, \boldsymbol{a})$ is the advantage function for observation and action, $\hat{\boldsymbol{\theta}}$ is the policy parameter obtained by sampling。According to PPO[10] theory, we can clip the policy gradient as:

$$\nabla_{\boldsymbol{\theta}} L(\pi(\boldsymbol{\theta})) \approx \mathbb{E}_{\pi(\hat{\boldsymbol{\theta}})}\left[\nabla_{\boldsymbol{\theta}} \text{log}_{\pi(\boldsymbol{\theta})} \text{C}(\boldsymbol{s}, \boldsymbol{a})\right], \tag{26}$$

where

$$C(\boldsymbol{s}, \boldsymbol{a}) = \min\left[f A_{\pi(\boldsymbol{\theta})}(\boldsymbol{s}, \boldsymbol{a}), \upsilon(f) A_{\pi(\boldsymbol{\theta})}(\boldsymbol{s}, \boldsymbol{a})\right], \tag{27}$$

$$\upsilon(x) = \begin{cases} 1+\epsilon, & x > 1+\epsilon, \\ x, & 1-\epsilon \le x \le 1+\epsilon, \\ 1-\epsilon, & x < 1-\epsilon. \end{cases} \tag{28}$$

where ϵ is an adjustable parameter. We define the loss function of the critic network as

$$J(\boldsymbol{\phi}) = \mathbb{E}_{\pi(\boldsymbol{\phi})}\left[-V_{\boldsymbol{\phi}}(\boldsymbol{s}) + \mathbb{E}_{\pi(\boldsymbol{\phi}}\left[r + V_{\boldsymbol{\phi}}(\boldsymbol{s}^{'})\right]\right] \tag{29}$$

where $\boldsymbol{\phi}$ is the parameter of the critic network, $\boldsymbol{s}^{'}$ is the observation result of the agent at the next time slot. The estimated gradient of the critic network is calculated by

$$\nabla_{\boldsymbol{\phi}} J^{'} = \frac{1}{B}\sum_{t=0}^{B-1}\left[V_{\boldsymbol{\phi}(\boldsymbol{s})} - Y(t)\frac{\mathrm{d}V_{\boldsymbol{\phi}}(\boldsymbol{s})}{\mathrm{d}\boldsymbol{\phi}}\right], \tag{30}$$

where

$$Y(t) = R(t) - \gamma^{B-t}R(B) + \gamma^{B-t}V_{\boldsymbol{\phi}}(B), \tag{31}$$

B is the mini-bath size when updating the critic network. Therefore, the parameter of the critic network can be updated through mini-batch random gradient descent method

$$\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - l^{c}\nabla_{\boldsymbol{\phi}} J^{'} \tag{32}$$

where l^{c} is the learning rate of updating the critic network. Similarly, the estimated gradient of the actor network is calculated by

$$\nabla_{\boldsymbol{\theta}} L^{'} = \frac{1}{B}\sum_{t=0}^{B-1}\nabla_{\boldsymbol{\theta}}\log_{\pi(\boldsymbol{\theta})}\mathrm{C}(\boldsymbol{s}(\mathrm{t}), \boldsymbol{a}(\mathrm{t})), \tag{33}$$

Thus $\boldsymbol{\theta}$ can be updated through mini-batch random gradient descent method

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + l^{a}\nabla_{\boldsymbol{\theta}} L^{'}, \tag{34}$$

where l^{a} is the learning rate of updating the actor network.

In summary, the PPO-based offloading and resource allocation algorithm is summarized in **Algorithm 1**.

Algorithm 1 PPO-based Offloading and Resource Allocation Algorithm

Initialization:

Agent initialization γ, $l^a, l^c, \boldsymbol{\theta}, \boldsymbol{\phi}$, $\forall m \in \mathcal{M}$

for time slot t in 1,2,...T **do**

Observe from the environment, update $\boldsymbol{s}(t)$.

Input $\boldsymbol{s}(t)$to the actor network, output the actions in current time slot $\boldsymbol{a}(t)$.

Calculate current reward $r(t)$.

Store $\{\boldsymbol{s}(t), \boldsymbol{a}(t), r(t)\}$ to $\mathcal{B}$.

if $t\%B == 0$ **then**

Calculate $\nabla_{\boldsymbol{\theta}} L^{'}$ and $\nabla_{\boldsymbol{\theta}} J^{'}$.

Update $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$.

Clear the replay Buffer $\mathcal{B}$

end if

end for

Output: Optimal offloading strategy, power control strategy and computing resource allocation strategy $\boldsymbol{A}^*, \boldsymbol{P}^*, \boldsymbol{C}^*$

3 Simulation Settings

Table 1: SUPER PARAMETERS of PPO

Parameter	Value
Learning rate of actor network l^a	3×10^{-4}
Learning rate of critic network l^c	3×10^{-4}
Discount factor γ	0.99
GAE parameter	0.95
PPO clip parameter	0.2
Batch size	2048
Mini batch size	64
Number of hidden fully connected layers (Orthogonal Initialization)	2
Number of neurons in each hidden layer	64

Table 2: SYSTEM SIMULATION PARAMETERS

Parameter	Value
Maximum Transmit power of users p_{max}	100mW
Minimum Transmit power of users p_{min}	0 W
Idle power of users p_{idle}	10mW
Computation resource of MEC server C_e	40 GHz
Computation resource of user m	[0.6,1] GHz
Task data size $D_{L_{n,k}}$	[0.05,0.1] MB
Computation density ρ_n	[10000,18000] CPU cycles/bit
Max tolerance delay τ_m	[1,2]s
effective switched capacitance of users η	10^{-27}
Gaussian white noise power z	-174 dBm/Hz
carrier frequency f	1.9GHz
system bandwidth W	20MHz
standard deviation of the shadow fading σ_s	8dB
d_0	10m
d_1	50m
antenna height at the AP h_a	15m
antenna height at the user h_u	1.65m
δ	0.2
coherence time τ_c	1ms

References

[1] H. Ngo, A. Ashikhmin, H. Yang, E. Larsson, and T. Marzetta, "Cell-free massive mimo versus small cells," *IEEE Transactions on Wireless Communications*, vol. PP, Feb. 2016.

[2] S. Buzzi, C. D'Andrea, and A. Zappone, "User-centric 5g cellular networks: Resource allocation and comparison with the cell-free massive mimo approach," *IEEE Transactions on Wireless Communications*, vol. PP, Mar. 2018.

[3] Z. Wang, E. Tameh, and A. Nix, "Joint shadowing process in urban peer-to-peer radio channels," *IEEE Transactions on Vehicular Technology*, vol. 57, pp. 52 – 64, Feb. 2008.

[4] Z. Chen, E. Björnson, and E. Larsson, "Dynamic resource allocation in co-located and cell-free massive mimo," *IEEE Transactions on Green Communications and Networking*, vol. PP, pp. 1–1, Nov. 2019.

[5] E. Björnson and L. Sanguinetti, "Making cell-free massive mimo competitive with mmse processing and centralized implementation," *IEEE Transactions on Wireless Communications*, vol. PP, pp. 1–1, Sep. 2019.

[6] S. Seng, C. Luo, X. Li, H. Zhang, and H. Ji, "User matching on blockchain for computation offloading in ultra-dense wireless networks," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1167–1177, Jun. 2021.

[7] E. Nayebi, A. Ashikhmin, T. Marzetta, and B. Rao, "Performance of cell-free massive mimo systems with mmse and lsfd receivers," Nov. 2016, pp. 203–207.

[8] S. Mukherjee and J. Lee, "Edge computing-enabled cell-free massive mimo systems," *IEEE Transactions on Wireless Communications*, vol. PP, pp. 1–1, Jan. 2020.

[9] S. Li, X. Hu, and Y. Du, "Deep reinforcement learning and game theory for computation offloading in dynamic edge computing markets," *IEEE Access*, vol. PP, pp. 1–1, Aug. 2021.

[10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," Jul. 2017.