Exploiting Deep Reinforcement Learning for Edge Caching in Cell-Free Massive MIMO Systems

Yu Zhang, Shuaifei Chen, and Jiayi Zhang

Abstract—Cell-free massive multiple-input-multiple-output is promising to meet the stringent quality-of-experience (QoE) requirements of railway wireless communications by coordinating many successional access points (APs) to serve the onboard users coherently. A key challenge is how to deliver the desired contents timely due to the radical changing propagation environment caused by the growing train speed. In this paper, we propose to proactively cache the likely-requesting contents at the upcoming APs which perform the coherent transmission to reduce end-to-end delay. A long-term QoE-maximization problem is formulated and two cache placement algorithms are proposed. One is based on heuristic convex optimization (HCO) and the other exploits deep reinforcement learning (DRL) with soft actor-critic (SAC). Compared to the conventional benchmark, numerical results show the advantage of our proposed algorithms on QoE and hit probability. With the advanced DRL model, SAC outperforms HCO on QoE by predicting the user requests accurately.

I. INTRODUCTION

Proactively caching popular services at the local trackside access points (APs) is promising to improve the perceptual quality-of-experience (QoE) of the onboard wireless communication services for railway systems [1]. However, with the enormously increasing traffic load demands and the dynamic radio propagation environment caused by the growing train operation speed, it becomes challenging due to the limited network capacity and caching capacity at the local APs [2]. The former causes the wireless delay in the data transmission between the user equipments (UEs) and the APs; the latter induces additional download delay for collecting the transmitted contents, which are not locally cached, from the adjacent APs or the core network. Both of them will substantially reduce the QoE. Therefore, collaboration among the local APs is needed for appropriate content transmission and caching.

As a potential enabler for the beyond fifth-generation (5G) wireless, the cell-free massive multiple-input-multiple-output (CF mMIMO) system is capable of reducing the wireless delay by providing strong macro diversity and coordinating many APs serving the UEs simultaneously, which improves the spectral efficiency (SE) [3]. In railway systems, CF mMIMO alleviates the violent propagation environment fluctuation by removing the communication service outage caused by handover between the base stations in the cellular mMIMO. Moreover, it also increases the content diversity by performing collaborative cache placement at the local APs, which essentially reduces the download delay [4]–[6].

However, most of the existing caching schemes could not meet the strict QoE requirements of the onboard railway communications due to the heavy computation complexity or lacking the ability to predict the upcoming content requests [7]. Such cache placement optimization problems are usually formulated as mixed-integer linear programming problems that have been proven to be NP-hard [8]. Many heuristic, stochastic, and various optimization techniques have been applied to solving the cache placement problems but either are too simple to provide a good result or become too complicated to implement in the dynamic network [7]. Deep reinforcement learning (DRL) is recently proposed as an efficient tool to solve such NP-hard problems. Many policies based on DRL are adopted for edge caching in cellular systems (e.g., [9]), which are not suitable for the CF mMIMO system due to the difficulty in modeling the problem of collaboration. That motivates this work. We find that adopting an appropriate coding method can solve the problem of modeling so that DRL can utilize the global live spatial information provided by the CF mMIMO system to perform better. We summarize our major contributions as follows.

- We design a cache framework in the railway communications where CF mMIMO is used. A QoE maximization problem is formulated where the cache placement is optimized to reduce the download delay.
- Based on this framework, we propose a heuristic convex optimization (HCO) algorithm to improve QoE indirectly and design a soft actor-critic (SAC)-based caching algorithm to maximize the QoE directly.
II. System Model

We consider a cache-aided railway system where a train travels through the consecutive coverage areas served by a set of track-side multi-antenna APs, each equipped with a local cache of limited capacity $c_{ap}$ Mbits. As illustrated in Fig. [I] all APs coordinated by a central processing unit (CPU) via the fronthaul connections to perform the collaborative proactive caching and joint wireless transmission for the onboard UEs [10], [11]. We consider $K$ UEs in the carriage, communicating with the nearby $L$ track-side APs via a single-antenna relay node (RN) mounted on the top of the train. (In this paper, the relay is used for signal transmission so that the number of relays will not affect so much. The problem of multi-relay cooperation is left for future work.) The inboard transmission between the RN and the UE is assumed to be error- and delay-free. The outboard transmission between the RN and the track-side APs is assumed to have a stable achievable data rate of $R$ thanks to the almost uniform wireless communication service provided by CF mMIMO (compared to the cellular mMIMO), especially in the railway systems [12], [13].

A. Service Request

We consider $M$ kinds of data services (e.g., the video streaming services), each segmented into $s_m$ small content chunks following its fixed chunk order with normalized size. In each period $t \in \{1, \ldots, T\}$ of time $\tau$, each UE makes an independent request for one content chunk of a data service with the probability according to a given popularity pattern, e.g., the Zipf distribution [14]

$$p_m = \frac{(m)^{-\epsilon}}{\sum_{m'=1}^{M} (m')^{-\epsilon}}$$

where $p_m$ is the request probability of service $m$, satisfying $0 \leq p_m \leq 1$ and $\sum_{m=1}^{M} p_m = 1$, and $\epsilon$ is the skewness parameter. Each UE requests the content chunk next to its previous request of the current desired service following the chunk order. The service requests are collected by the RN and then reported to the CPU via the serving APs. We denoted by $r^t = [r^t_1, \ldots, r^t_M] \in \mathbb{N}^M$ the service requests collected in period $t$, where $r^t_m$ is the number of UEs requesting for the service $m$. Each UE has a certain patience for requesting a data service, which is decremented by 1 every period if the UE does not receive its requested data service. If the patience reduces to 0, then this UE will refresh its patience and request a service according to the (1) again.

B. Content Preparation, Transmission, and Placement

After receiving the service requests from the RN, the CPU coordinates the serving APs to a) first prepare the desired content chunks in their local caches, b) then jointly transmit these content chunks to the RN in sequence, and c) place the likely-requesting content chunks in local caches of the serving APs in the future coming periods to avoid congestion, which is elaborated in Section [II] and Section [IV].

The serving APs prepare the data of transmitted content chunks by acquiring them from their own caches, or the caches of other nearby APs via fronthaul connections, or the core network via the backhaul connections. To perform joint coherent transmission, the required content chunks need to be duplicated and cached in every serving APs. To simplify this operation, we adopt the Raptor Codes [15] to encode services as several fragments that can be stored at the serving APs and recover services during the transmission. More precisely, we denote by $\chi^t = \left\{ \chi^t_m : l = 1, \ldots, L, m = 1, \ldots, M \right\}$ the caching policy in period $t$, where element $0 \leq \chi^t_m \leq 1$ indicates $\chi^t_m$ fraction of the cache at AP $l$ is dedicated to the coded data (fragments) of service $m$, satisfying $\sum_{m=1}^{M} \chi^t_m = 1$, $\forall l, t$. With the help of the Raptor Codes, any serving AP can jointly recover the services when it gathers enough number of fragments from nearby $L'$ APs in those $L$ APs, i.e.,

$$\sum_{l=1}^{L'} \chi^t_{lm} c_{ap} \geq s_m, \quad m = 1, \ldots, M, L' \leq L. \tag{2}$$

and then send any chunk of the required service to the UEs.

Compared to cellular mMIMO, CF mMIMO alleviates the violent data rate fluctuation caused by physical movement in the railway systems, and, thus, is capable to provide stable wireless communication services [12], [16]–[18]. Given a limited achievable data rate $R$ for outboard transmission between the RN and the serving APs, the RN can download at most $\tau R$ data in a period of time $\tau$. For the case that more than $\tau R$ data of content chunks are requested, some of them might not be downloaded at the RN timely, causing the so-called data stream stalling. The service with chunks that have been either requested by many UEs and/or installed for a long time should be answered with priority. With this in mind, we use the total number of requests for the installed services by $t$ to compute $\omega_m^t$ to indicate the weight of priorities by $t$. It’s cumulative, i.e.,

$$\omega_m^t = \begin{cases} \omega_m^{t-1} + r^t_m & \text{if } m \text{ is installed} \\ 0 & \text{otherwise} \end{cases}$$

Then, we let $u^t = [u^t_1, \ldots, u^t_M] \in \mathbb{N}^M$, where $u^t_m = r^t_m + \omega_m^{t-1} \in \mathbb{N}$, to represent the priorities of dealing with the services in period $t$. The serving APs will jointly recover the contents of the requested services and transmit content chunks of the requested services in the descending order of $u^t$’s element.

C. Delay Model and QoE

Recall that the serving APs could acquire the transmitted content chunks from different entities, which causes different preparation delays. If an AP locally caches enough fragments to recover services to gain transmitted content chunks, then no preparation delay is induced. If an AP $l$ does not cache enough fragments but could recover the service $m$ exploiting the cached fragments in other $L'_l$ APs around itself satisfying $\sum_{l'=1}^{L'_l} \chi_{lm}^l c_{ap} \geq s_m$ and $L'_l \leq L$, the AP $l$ will find an AP $j_l$ that costs the most time in those $L'_l$ APs, i.e.,

$$j_l = \arg \max_{j \in L'_l} \chi_{lj}^l \lambda_{lj} \tag{3}$$

where $\lambda_{lj}$ is the weight related to the distance between the AP
the AP $l$. The preparation delay of $m$ in the fronthaul connections is related to the AP that costs the most time, i.e.,

$$\tau_{bh} (m) = \delta_{cap} \cdot \max_{l} \chi_{lm}^t \lambda_{jl}$$  \hspace{1cm} (5)$$

where $\delta$ is scale coefficient. If $\sum_{l=1}^{L} \chi_{lm}^t c_{ap} < s_m$, the serving APs have to get the rest fragments of the service $s_m - \sum_{l=1}^{L} \chi_{lm}^t c_{ap}$ from the core network via the CPU, which causes the preparation delay in the backhaul connections, as

$$\tau_{bh} (m) = \delta' \left( s_m - \sum_{l=1}^{L} \chi_{lm}^t c_{ap} \right)$$  \hspace{1cm} (6)$$

where $\delta'$ is another scale coefficient. The total delay with respect to service $m$ containing the wireless delay and download delay composed of $\tau_{bh}$ and $\tau_{re}$ can be formulated as:

$$\tau_{re} (m) = \tau_{bh} (m) + \tau_{re} (m) + \frac{1}{T}$$  \hspace{1cm} (7)$$

where $n_m$ is the number of chunks of $m$ required by UEs in total in the current period. According to the total delay of requesting to each service, we can get the number of installed services and compute the $\omega_m^t$. We define the QoE as below.

$$\text{QoE}^t = \frac{\sum_{m=1}^{M'} (r_m^t + \omega_m^t)}{\sum_{m=1}^{M} (r_m^t + \omega_m^t)}$$  \hspace{1cm} (8)$$

where $M'$ is the maximum of the number of services that can be served in the current period, i.e., $M' = \arg \max_{M} \sum_{m=1}^{M} \tau_{re} (m) \leq \tau$. Moreover, we introduce the hit probability of request

$$\sum_{m=1}^{M} r_m^t \min \left( \frac{c_{ap} \sum_{l=1}^{L} \chi_{lm}^t}{s_m}, 1 \right) / K$$  \hspace{1cm} (9)$$

to evaluate the performance of the cache solving the requests and the hit probability of QoE

$$\sum_{m=1}^{M} \left( r_m^t + \omega_m^t \right) \min \left( \frac{c_{ap} \sum_{l=1}^{L} \chi_{lm}^t}{s_m}, 1 \right) / \sum_{m=1}^{M} \left( r_m^t + \omega_m^t \right)$$  \hspace{1cm} (10)$$

to evaluate the performance of the cache improving the QoE in period $t$.

**III. PROBLEM FORMULATION**

The collaborative caching optimization problem can be formulated as the following long-term maximization problem

$$\max_{\{x^t\}} \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \text{QoE}^t$$

s.t. \hspace{1cm} \sum_{m=1}^{M} \chi_{lm}^t \leq 1 \hspace{1cm} (11)$$

which is non-convex and NP-hard. To solve this problem, we need to improve the hit probability of request of APs’ caches to reduce $\tau_{bh}$ and place the fragments cached at the APs more efficiently to reduce the $\tau_{bh}$. We develop two algorithms for this problem. One exploits a HCO algorithm to reduce the $\tau_{bh}$, and the other adopts a SAC-based caching algorithm to reduce both $\tau_{bh}$ and $\tau_{re}$ comprehensively.

**A. Heuristic Convex Optimization (HCO) Algorithm**

Since delay $\tau_{bh}$ on the backhaul connections has a domi- native impact, we could simplify the problem in (11) by maximizing the hit probability of request to minimize $\tau_{bh}$ without considering $\tau_{re}$.

Concentrated on the hit probability of request, we can formulate the collaborative caching optimization problem as

$$\max_{\{x^t\}} \sum_{m=1}^{M} r_m^t \min \left( \frac{c_{ap} \sum_{l=1}^{L} \chi_{lm}^t}{s_m}, 1 \right)$$

s.t. \hspace{1cm} \sum_{m=1}^{M} \chi_{lm}^t \leq 1 \hspace{1cm} (12)$$

Problem in (12) is a convex problem which can be solved by convex optimization since the constraints and objective function are composed of the maximum value and weighted sum of the convex functions. We design a heuristic algorithm based on convex optimization to realize the proactive cache. We first collect the statistics of the $N$ historical requests. Then, we obtain the cache solution of maximizing the hit probability of request according to these statistics. Due to the stationarity of the Zipf, we can treat this solution as an approximate solution for requests in the current period.
B. DRL Approach

Since HCO is aimed to reduce the $\tau_{th}$ based on the statistics of historical information, HCO is not accurate enough to predict the instantaneous request and could not deal with the issue of $\tau_{th}$ well. We utilize the DRL for each period as a remedy, which is shown in detail in the next section.

IV. SAC-BASED CACHING

Reinforcement learning (RL) is a promising tool to solve such NP-hard problems by learning policies like the human brain. The entire learning process includes so many episodes that an episode is the time for the train finishing the whole journey at one time. Each episode has lots of periods to train. In each period, based on observing the current state of the environment, an agent takes an action following a certain policy and obtains the reward and the next state from the environment. Utilizing the reward, the agent learns to develop the optimal policy to maximize the expected sum of rewards. And then, based on the next state, the agent will repeat the interactive process for the next period until the end of the episode. In general, RL problems can be defined as policy searches in an MDP.

A. Define of MDP

An MDP is characterized by a tuple $(\mathcal{S}, \mathcal{A}, p, \gamma_t)$ to describe the interaction between the agent and the environment, where $\mathcal{S}$, $\mathcal{A}$, $p$, and $\gamma_t$ represent the state space, the agent’s action space, the probability of the transition from the current state to the next state, the reward of performing an action $A_t \in \mathcal{A}$ based on a state $S_t \in \mathcal{S}$, respectively. We also use $\rho_\pi(S_t)$ and $\rho_\pi(S_t, A_t)$ to denote the state and state-action marginals of the trajectory distribution induced by a policy $\pi(S_t, A_t)$.

Exploiting the centralized network architecture, we treat the CPU as an agent to control the cache state of each AP that is going to serve the UEs. Based on this, the state, the action, and the reward are defined as below:

- We consider the $\chi^t$ and $r^t$ as the state $S_t$.
- Then, we assume that the action of the CPU or the agent in episode $t$ is described as a matrix $A_t \in \mathbb{R}^{L \times M}$, where the element $a_{lm}^t \in [-1, 1]$ at $l$-th row and $m$-th column of $A_t$ means the percentage of the cache that AP $l$ decides to adjust to store the $m$-th service.
- The reward $\gamma_t(S_t, A_t)$ is defined as $Q_0E^t$. And the expected sum of rewards is denoted by $\sum_{t=1}^{T} \mathbb{E}(S_t, A_t) \sim \rho_\pi[\gamma_t(S_t, A_t)]$.

B. SAC-Based Caching

To solve such an MDP more effectively, DRL utilizes the neural network (NN) as a function estimator based on RL to save the storage space, speed up the learning process, and solve large-scale RL problems. DRL has three implementation forms: value-based approaches, policy-based approaches, and actor-critic (AC) approaches. For our problem, we have a large and continuous state space and action space that value-based approaches and policy-based approaches are impractical to converge. Combining the advantages of them, AC approaches perform perfectly. It derives from the policy iteration that computes a value to evaluate a policy called policy evaluation and uses the value to evaluate a better policy called policy improvement. In this case, the policy and the value is regarded as the actor and the critic, respectively. Hence, we design a SAC-based caching algorithm under the category of AC, which is shown in Fig. 2.

SAC has five neural networks in total including an actor network, two critic networks, and two target critic networks. The actor network characterized by $\pi_\mu (A_t|S_t)$ is used to output means and covariances of Gaussians to generate elements of $A_t$ based on the input state $S_t$. Performing this action on the environment, the agent gets the reward $\gamma_t$ and the next state $S_{t+1}$. We store the transition $(S_t, A_t, \gamma_t, S_{t+1})$ in the replay buffer $B$ as the experience of the current period $t$. Then, we use the transitions from $B$ to update the networks to achieve our objective. Note that, different from the conventional RL, SAC aims to obtain the optimal policy to maximize the maximum entropy objective that generalizes the standard objective by augmenting it with an entropy term $H$ in [19], i.e.,

$$\pi^* = \arg \max_{\pi} \sum_{t=1}^{T} \mathbb{E}(S_t, A_t) \sim \rho_\pi[\gamma_t(S_t, A_t) + \alpha H(\pi(S_t))]$$

(13)

where $\alpha$ is the temperature parameter that determines the relative importance of the entropy term versus the reward and thus controls the stochasticity of the optimal policy. Utilizing such an objective, the policy explores much more widely and the learning speed is improved a lot.

The algorithm alternates between policy evaluation and policy improvement in the maximum framework for updating. In the policy evaluation step, we compute the soft $Q$ value of the policy $\pi$ by the $Q$-function according to the maximum entropy objective. We use the critic characterized by $Q(S_t, A_t|\theta)$ as the function estimator for the $Q$-function, which can be trained to minimize the soft Bellman residual [14],

$$J_Q(\theta) = \mathbb{E}(S_t, A_t) \sim B \left[ \frac{1}{2} (Q_{\theta}(S_t, A_t) - \gamma_t(S_t, A_t) + \zeta \mathbb{E}_{S_{t+1} \sim \rho(S_t, A_t)}[V_{\theta'}(S_{t+1})])^2 \right]$$

(14)

where $\zeta$ is the discount factor and $V$ is computed by

$$V_{\theta'}(S_t) = \mathbb{E}_{A_t \sim \pi_\mu}(Q_{\theta'}(S_t, A_t) - \alpha \log \pi_\mu(A_t|S_t))$$

(15)

using the target critic network characterized by $Q'(S_t, A_t|\theta')$. The target critic is aimed to output a target soft $Q$ value regarded as the true value of input action and state for updating the $Q(S_t, A_t|\theta)$ since the update of the network would change the soft $Q$ value that makes the parameters hard to converge in practice. The target critic has the same structure as the critic but is different in using Poyak averaging to update:

$$\theta' \leftarrow \sigma \theta + (1 - \sigma) \theta'$$

where $\sigma \in (0, 1)$ is the soft coefficient that is small to make sure that the target network updates slowly enough so that its output can be regarded as the true value. In the policy improvement step, we use the soft $Q$ value to gain the optimal...
Algorithm 1: SAC-Based Caching

1. Randomly initialize parameters $\theta_i$ and $\mu$ of the critic networks $Q(S_t, A_t | \theta_i)$, $i \in \{1, 2\}$ and actor network $\pi_{\mu}(A_t | S_t)$
2. Initialize parameters of the target networks with $\theta_i' \leftarrow \theta_i$, $i \in \{1, 2\}
3. Initialize replay buffer $B$

for each environment step do
   $A_t \sim \pi_{\mu}(A_t | S_t)$
   $S_{t+1} \sim p(S_{t+1} | S_t, A_t)$
   $B \leftarrow B \cup \{(S_t, A_t, r(S_t, A_t), S_{t+1})\}$

for each update step do
   $\theta_i \leftarrow \theta_i - \gamma Q_{\theta}(\theta_i)_{\theta_i}$, $i \in \{1, 2\}$
   $\mu \leftarrow \mu - \lambda \nabla_{\mu} J_{\mu}(\mu)$
   $\alpha \leftarrow \alpha - \lambda \nabla_{\alpha} J_{\alpha}(\alpha)$
   $\theta_i' \leftarrow \sigma \theta_i' + (1 - \sigma) \theta_i'$, $i \in \{1, 2\}$

V. NUMERICAL RESULTS

A. Setups

We consider that there are $K = 300$ UEs on the train and $L = 15$ APs equipped with half-wavelength spaced uniform linear arrays and a local cache of $c_{ap} = 6$ Mbits to serve the UEs at a certain period. Those UEs request services following the Zipf with the skewness parameter $\epsilon = 1.1$. Each request has a different size about 10 to 20 Mbits. Unless otherwise specified, we employ the well-known 3GPP Suburban Microcell model to compute the large-scale propagation conditions and other system parameters in Table I.

We train $2 \times 10^5$ episodes for our proposed SAC-based caching algorithm. Then, we use independent 10000 samples to form the test dataset for those three algorithms. Particularly, to use $\eta$HC with the Raptor Codes wisely, the cache in each AP is dedicated to all the contents fragments based on their percentage of preference ($\eta = 1$). Moreover, other details of SAC parameters are shown in Table I.

B. Results and discussions

We firstly evaluate the QoE of three algorithms with $M = 20$ and $M = 30$ in Fig. 3. It can be seen that with the SAC-based caching assignment, the $90\%$-likely QoE can be improved by about $10\%$ and $6\%$ compared to the $\eta$HC and the HCO algorithm for $M = 20$, respectively. As the number of services increases to $M = 30$, the QoE under three algorithms all decrease a lot while the gain of the SAC still remains about $10\%$ and $6\%$. The reason is that the SAC is implemented to learn the best policy to maximize the QoE straightly while the HCO is applied only to minimize the $\tau_{bh}$ to improve the QoE indirectly and ignores the influence of $\tau_{bh}$.

The hit probability of QoE of three algorithms is shown in Fig. 4. Focus on maximizing the hit probability of request, the HCO also performs well on the hit probability of QoE since QoE is highly related to the request, i.e., caching popular services is more likely to gain a high hit probability of QoE. However, as mentioned above, the HCO makes the service hard to recover that significantly increases $\tau_{bh}$ and then leads to a lower QoE than the SAC. Although the hit probability of QoE of the SAC performs less well, it is more concentrated.

TABLE I

| Parameters               | Values               |
|--------------------------|----------------------|
| Optimizer                | Adam                 |
| Learning rate            | $3 \times 10^{-4}$   |
| Discount factor $\zeta$  | 0.99                 |
| Size of replay buffer $B$| $1 \times 10^6$      |
| Number of hidden units per layer | 256          |
| Number of hidden layers  | 2                    |
| Entropy target           | -dim($A$)(e.g.,−20)  |
| Activation function      | ReLU                 |
| Soft update parameter $\sigma$ | 0.005          |
SAC outperforms the HCO. Since we only consider one relay running speed and QoE performance. Generally speaking, the maximize the QoE straightly, which leads to a better stability, the maximum entropy objective to learn the best policy to QoE but makes the service hard to recover that finally leads of request, the HCO improves the most hit probability of significantly improve QoE. Focus on maximizing the hit probability of request, the SAC-based caching algorithm to improve the QoE straightly. The simulation results show that both algorithms can significantly improve QoE. To analyze the time complexity of the investigated schemes, we investigate the running time on the ordinary computing platform with different $M$ in Fig. 5. Utilizing the maximum entropy objective as the learning target, the SAC runs much faster than the HCO.

VI. Conclusion

In this paper, we investigated the caching placement in a CF railway system. To maximize the QoE, we developed two algorithms including the HCO algorithm to maximize the hit probability of request to improve the QoE indirectly and the SAC-based caching algorithm to improve the QoE straightly. The simulation results show that both algorithms can significantly improve QoE. Focus on maximizing the hit probability of request, the HCO improves the most hit probability of QoE but makes the service hard to recover that finally leads to a lower QoE than the SAC. By contrast, SAC utilizes the maximum entropy objective to learn the best policy to maximize the QoE straightly, which leads to a better stability, running speed and QoE performance. Generally speaking, the SAC outperforms the HCO. Since we only consider one relay in this paper, the scenario of more relays cooperation is left for future work.

References

[1] J. Zhang, H. Liu, Q. Wu, Y. Jin, Y. Chen, B. Ai, S. Jin, and T. J. Cui, “Ris-aided next-generation high-speed train communications: Challenges, solu-
lutions, and future directions,” IEEE Wireless Commun., vol. 28, no. 6, pp. 145–151, Dec. 2021.
[2] A. Ioannou and S. Weber, “A survey of caching policies and forwarding mechanisms in information-centric networking,” IEEE Commun. Surv. Tutor., vol. 18, no. 4, pp. 2847–2886, May 2016.
[3] J. Zhang, E. Björnsson, M. Matthaiou, D. W. K. Ng, H. Yang, and D. J. Love, “Prospective multiple antenna technologies for beyond 5G,” IEEE J. Sel. Areas Commun., vol. 38, no. 8, pp. 1637–1660, June. 2020.
[4] E. Björnsson and L. Sanguinetti, “Making cell-free massive MIMO competitive with MMSE processing and centralized implementation,” IEEE Trans. Wireless Commun., vol. 19, no. 1, pp. 77–90, Jan. 2019.
[5] J. Zhang, J. Zhang, E. Björnsson, and B. Ai, “Partial local zero-forcing combining for cell-free massive MIMO systems,” IEEE Trans. Commun., vol. 69, no. 12, pp. 8459–8473, Dec. 2021.
[6] J. Zhang, J. Zhang, D. W. K. Ng, S. Jin, and B. Ai, “Improving sum-rate of cell-free massive MIMO with expanded compute-and-forward,” IEEE Trans. Signal Process., vol. 70, pp. 202–215, 2021.
[7] S. Anokye, M. SEID et al., “A survey on machine learning based proactive caching,” ZTE commun., vol. 17, no. 4, pp. 46–55, Feb. 2020.
[8] S. Chen, J. Zhang, E. Björnsson, S. Wang, C. Xing, and B. Ai, “Wireless caching: cell-free versus small cells,” in Proc. ICC, 2021, pp. 1–6.
[9] Y. Wei, F. R. Yu, M. Song, and Z. Han, “Joint optimization of caching, computing, and radio resources for fog-enabled iot using natural actor–critic deep reinforcement learning,” IEEE Internet Things J., vol. 6, no. 2, pp. 2061–2073, Oct. 2018.
[10] Y. Jin, J. Zhang, S. Jin, and B. Ai, “Channel estimation for cell-free mmwave massive MIMO through deep learning,” IEEE Trans. Veh. Tech., vol. 68, no. 10, pp. 10325–10329, Oct. 2019.
[11] J. Zhang, S. Chen, Y. Lin, J. Zheng, B. Ai, and L. Hanzo, “Cell-free massive MIMO: A new next-generation paradigm,” IEEE Access, vol. 7, pp. 99,878–99,888, Jul. 2019.
[12] J. Zheng, J. Zhang, E. Björnsson, Z. Li, and B. Ai, “Cell-free massive MIMO-OFDM for high-speed train communications,” IEEE J. Sel. Areas Commun., to appear, 2022.
[13] H. Liu, J. Zhang, X. Zhang, A. Kurniawan, T. Juhana, and B. Ai, “Tabu-search-based pilot assignment for cell-free massive MIMO systems,” IEEE Trans. Veh. Tech., vol. 69, no. 2, pp. 2286–2290, Feb. 2019.
[14] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, “Web caching and zipf-like distributions: Evidence and implications,” in IEEE INFO-
COM’99, vol. 1. IEEE, 1999, pp. 126–134.
[15] A. Shokrollahi, “Raptor codes,” IEEE Trans. Inf. Theory, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.
[16] Z. Wang, J. Zhang, B. Ai, C. Yuen, and M. Debbah, “Uplink performance of cell-free massive MIMO with multi-antenna users over jointly-correlated Rayleigh fading channels,” IEEE Trans. Wireless Commun., to appear, 2022.
[17] S. Chen, J. Zhang, E. Björnsson, J. Zhang, and B. Ai, “Structured massive access for scalable cell-free massive MIMO systems,” IEEE J. Sel. Areas Commun., vol. 39, no. 4, pp. 1086–1100, Apr. 2021.
[18] Z. Wang, J. Zhang, E. Björnsson, and B. Ai, “Uplink performance of cell-free massive MIMO over spatially correlated rician fading channels,” IEEE Commun. Lett., vol. 25, no. 4, pp. 1348–1352, Apr. 2020.
[19] B. D. Ziebart, “Modeling purposeful adaptive behavior with the principle of maximum causal entropy,” Carnegie Mellon University, 2010.
[20] S. Fujimoto, H. Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods,” in ICML, 2018, pp. 1587–1596.
[21] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in ICML, 2018, pp. 1861–1870.
[22] E. Shi, J. Zhang, S. Chen, J. Zheng, Y. Zhang, D. W. Kwan Ng, and B. Ai, “Wireless energy transfer in RIS-aided cell-free massive MIMO systems: Opportunities and challenges,” IEEE Commun. Mag., vol. 60, no. 3, pp. 26–32, Mar. 2022.