Joint Optimization for Cooperative Service-Caching, Computation-Offloading, and Resource-Allocations Over EH/MEC 6G Ultra-Dense Mobile Networks

Zhian Chen, Fei Wang, and Xi Zhang, *Fellow, IEEE*

***Abstract*— Service-caching, computation-offloading, and *mobile edge-computing* (MEC) have been widely recognized as three key 6G wireless technologies which can efficiently support implementing the *ultra-dense networks* (UDNs) with massive *small-cell base stations* (SBSs). But, these impose the new challenges for the UDNs to solely rely on grid power for energy supplying and to jointly optimize service-caching, computation-offloading, and resource-allocations. To overcome the above described difficulties, integrating *energy-harvesting* (EH) techniques with MEC-enabled 6G UDNs, we propose to develop the joint optimization schemes for cooperative service-caching, computation-offloading, and resource-allocations. In our considered UDNs, there exist a large number of EH-based *stationary users* (SUs) or *mobile users* (MUs), and a mixture of *on-grid SBSs* powered by electric grid and *off-grid SBSs* power-supplied by solar, *radio frequency* (RF) energy, etc. Specifically, first we formulate an energy minimization problem under a non-linear RF-energy EH model to minimize the sum of weighted energy consumption of users and off-grid SBSs. Second, for scenarios with SUs, we develop a two-timescale based joint cooperative service-caching, computation-offloading, and resource-allocations scheme using the *hierarchical multi-agent deep reinforcement learning*. We derive cooperative service-caching in each time frame, and then derive computation-offloading and resource-allocations in each time slot. Third, we extend our work to scenarios with MUs, where MUs can move with certain trajectories at low speeds. Finally, we validate and evaluate the performances of our proposed schemes through the extensive simulations.**

***Index Terms*— EH/MEC-based 6G UDNs, cooperative service-caching, computation-offloading, resource-allocations, HMDRL.**

I. Introduction

MOBILE edge-computing (MEC) enabled ultra-dense networks (UDNs), which merge edge-computing with UDNs [1], [2], [3], [4], can provide enormous benefits, e.g., ultra-low latency and super-high data rates. UDNs increase network capacity and provide users with flexible radio access services by densely deploying short-range small-cell base stations (SBSs), and MEC provides users with the efficiently complementary cooperations between the cloud computing and the edge computing. However, under the constrained computational power and caching resources at SBSs, the severe interference, etc., the quality of services (QoS) [5], [6], [7] improvement-levels gained by MEC-enabled UDNs heavily depend on the efficient deployments and cooperations of three key 6G techniques, including service-caching, computation-offloading, and MEC. Correspondingly, the joint co-designs and optimizations over computation-offloading, service-caching, and resource-allocations for MEC-enabled UDNs have been highly demanded, while having not been well studied yet.

A number of works have studied the problem of computation-offloading and/or resource-allocations for MEC-enabled UDNs. The authors of [4] and [8] minimized the task processing delay of users, the energy consumption of users and SBSs, etc., by developing suitable computation-offloading and/or resource-allocations schemes. In [3] and [9], taking into account the mobility of a representative user, the authors considered the joint problem of computation-offloading, computation migration, and wireless handover in MEC-enabled UDNs. However, it is not always feasible to provide grid power to all SBSs due to their possible outdoor/remote/hard-to-reach locations. Moreover, the density of SBSs in UDNs is larger than 1 SBS$/1000$ m^2. Therefore, to reduce the dependence of SBSs on grid power for energy supplying, it is necessary to integrate energy-harvesting (EH) techniques, which enable SBSs to harvest energy from solar, wind, radio-frequency (RF) signals, etc., with MEC-enabled UDNs [10]. Hence, the authors of [11] considered the computation-offloading and resource-allocations problem for MEC-enabled UDNs with EH capabilities, where SBSs harvest energy from solar or wind. The authors of [12] considered energy efficiency maximization for downlink transmission of millimeter-wave-based UDNs with EH SBSs, where SBSs harvest energy from the ubiquitous RF signals in UDNs. However, computation-offloading and/or resource-allocations for MEC-enabled UDNs with RF-energy harvesting capabilities has/have not been widely studied.

In addition, for processing the offloading tasks of users, SBSs must have cached the required services of these users. However, the above works, including [4], [8], [9], [10], [11],

AQ:1 Received 6 October 2023; revised 14 March 2024; accepted 17 February 2025. The work of Xi Zhang was supported in part by the U.S. National Science Foundation under Grant CCF-2142890, Grant CCF-2008975, Grant AQ:2 ECCS-1408601, and Grant CNS-1205726. An earlier version of this paper was presented in part at the 2023 IEEE International Conference on Communications (IEEE ICC 2023) [1]. The associate editor coordinating the review of this article and approving it for publication was X. Wang. *(Corresponding author: Fei Wang.)*

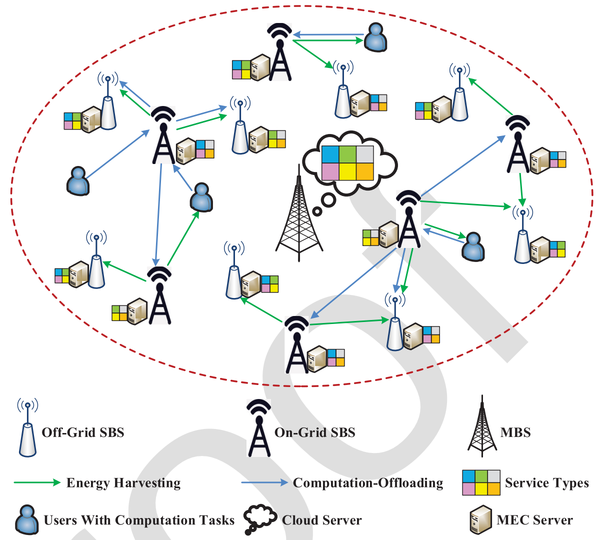
AQ:3 Zhian Chen and Fei Wang are with the College of Electronic and Information Engineering, Southwest University, Chongqing 400715, China (e-mail: cza0311@163.com; wsf0107@163.com).

Xi Zhang is with the Networking and Information Systems Laboratory, Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843 USA (e-mail: xizhang@ece.tamu.edu).

Digital Object Identifier 10.1109/TWC.2025.3549415

and [12], assume that each SBS has cached all the service programs required by users. In practice, due to limited caching storage capacity, each SBS can only selectively cache a subset of service programs [13]. Hence, it is necessary to optimize the utilization of the limited caching resources to improve entire network performances. Therefore, the authors of [14] studied the joint optimization problem of service-caching and computation-offloading, where SBSs cooperatively serve users or they offload users' tasks to the remote cloud. The work [15] considered the joint optimization problems of cooperative service-caching and computation-offloading among SBSs, where SBSs collaboratively serve users relying on their caching services. Notice that the schemes proposed in [15] update service-caching and computation-offloading in the same timescale. However, unlike computation-offloading generally updating at a time level of less than hundreds of milliseconds, the downloading and installation of a service program generally takes more than tens of seconds (even several hours or days) [13]. Hence, the authors of [14] and [16] developed two-timescale schemes, which update service-caching of SBSs in a *slow timescale*, e.g., in each time frame, but optimize computation-offloading and subcarrier allocations in a *fast timescale*, e.g., in each time slot. On the other hand, the work of [14] assumes that each user only requests one type of service in one time frame and the work of [16] assumes that all SBSs cache the same services. Besides, the works of [14], [15], and [16] did not consider the mobility of users and assume that all SBSs are powered by electric grid, which is unrealistic for wireless UDNs.

To overcome the above-mentioned shortcomings, in this paper we propose to develop the joint optimization schemes for cooperative service-caching, computation-offloading, and resource-allocations for EH/MEC-based 6G UDNs, where a large number of users,[1] including *stationary users* (SUs) or *mobile users* (MUs), with RF-energy harvesting capabilities and a mixture of on-grid SBSs, powered by electric grid, and off-grid SBSs, powered by solar and/or RF-energy, coexist. We formulate an energy minimization problem to minimize the sum of weighted energy consumption of all users and off-grid SBSs under a non-linear RF-energy EH model. Also, for the scenarios with SUs, we develop the two-timescale based joint cooperative service-caching, computation-offloading, and resource-allocations scheme using the *hierarchical multi-agent deep reinforcement learning* (HMDRL). Leveraging HMDRL, we derive SBSs' cooperative service-caching policies which are updated in each frame consisting of multiple time slots. According to the obtained cooperative service-caching policies, we first derive users' and SBSs' computation-offloading policies and then derive SBSs' computation resource-allocations policies, which are updated in each time slot. Furthermore, taking into account the mobility of users, we extend our work to the scenarios with MUs, where each MU can move with a certain trajectory at a low speed. In addition, we validate and evaluate the performances of



AQ:4

Fig. 1. System model for our proposed EH/MEC-based 6G UDNs, which consist of a large number of EH-based SUs or MUs, on-grid SBSs powered by grid power, and off-grid SBSs powered by solar and/or RF-energy.

our developed schemes through the extensive simulations. The simulation results show that the sum of weighted energy consumption of users and off-grid SBSs can be significantly reduced by using our proposed schemes, especially when the densities of users and off-grid SBSs increases.

The rest of this paper is organized as follows. Section II builds up the system models. Sections III and IV develop our proposed joint cooperative service-caching, computation-offloading, and resource-allocations schemes for scenarios with SUs and MUs, respectively. Section V validates and evaluates our proposed schemes through the extensive simulations. The paper concludes with Section VI.

II. The System Models

A. Architecture for Our Proposed EH/MEC-Based UDNs

Consider a mobile edge-computing (MEC) enabled energy harvesting (EH) 6G ultra-dense network (UDN) depicted in Fig. 1, which consists of multiple EH-based *stationary users* (SUs) or *mobile users* (MUs), multiple *small-cell base stations* (SBSs) each equipped with two antennas, and a macro base station (MBS). Each SBS is equipped with an MEC server and the MBS is equipped with a cloud server. Users connect to SBSs through wireless links, while SBSs connect to each other and the MBS through wired links [17]. The SBSs can be classified into two types, i.e., the on-grid SBSs, powered by the conventional grid power, and the off-grid SBSs,[2] powered by the solar energy and the radio frequency (RF) energy harvested from ambient on-grid SBSs. However, the users can only harvest RF-energy from the on-grid SBSs, since it may not be able to equip users with solar panels due to their size limitations [18]. Besides, the on-grid SBSs, the off-grid SBSs, and the users are spatially distributed according to three independent homogeneous Poisson Point Processes (HPPPs), denoted by $\mathcal{B}_{\rm g}, \mathcal{B}_{\rm e}$, and Ω, respectively, with spatial densities $\lambda_{\rm g}$, $\lambda_{\rm e}$, and ρ, respectively. Due to the limited computation

[1]Throughout this paper, we use *user* to represent either stationary user (SU) or mobile user (MU) or both stationary user (SU) and mobile user (MU), unless specifically stating that it represents the stationary user (SU) or the mobile user (MU) for the particular scenario, otherwise.

[2]In practice, solar-powered SBSs have been realized and applied in practice, but SBSs powered only by RF-energy have not been realized because the limited RF-energy cannot support the huge energy consumption of SBSs.

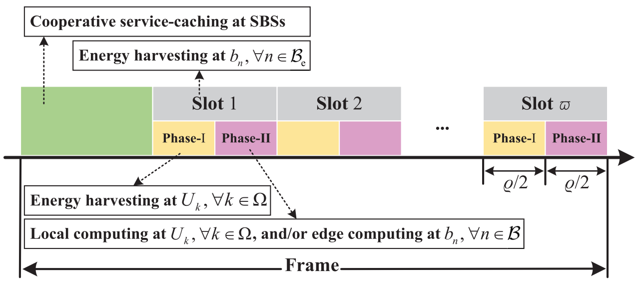


Fig. 2. Frame structure for our proposed EH/MEC-based 6G UDNs, where computation-offloading and service-caching update in two timescales.

capability, each user $U_k, \forall k \in \Omega$, may need to offload part of its computation task to a nearby SBS. Moreover, SBSs work in a cooperative service-caching manner. That is, if an SBS does not cache services required by some users, it can offload these users' tasks to other SBSs which have cached the services required by these users.

The updating of service-caching generally follows a slow timescale (e.g., tens of seconds or several hours) [16]. In contrast, computation-offloading follows a relatively fast timescale (e.g., milliseconds). Therefore, service-caching and computation-offloading work in two different timescales. As shown in Fig. 2, in each frame, SBSs first collaboratively update service-caching within several time slots. Then, in each of the remaining ϖ time slots (each with duration of ϱ), each user $U_k, \forall k \in \Omega$, first scavenges energy from RF signals radiated by ambient on-grid SBSs over system downlink spectrum in Phase-I with duration $\varrho/2$, and then uses the harvested energy to process its own task locally and/or offload the task to a nearby SBS $b_n, \forall n \in \mathcal{B} \triangleq (\mathcal{B}_g \cup \mathcal{B}_e)$, over system uplink spectrum in Phase-II with duration $\varrho/2$. In addition, each off-grid SBS harvests energy from the solar and/or the RF signals emitted by the on-grid SBSs nearby in the whole time slot, and in the meantime utilizes the harvested energy to help users process tasks in Phase-II. The main symbols used in this paper are listed in Table I.

B. The Communication and Computation Models

Different SBSs can use the same spectrums while users accessing the same SBS use the orthogonal uplink spectrums. Thus, there exists interference among users accessing different SBSs, if these SBSs share the same spectrums. Moreover, to reduce energy consumption, each user U_k first offloads task to its nearest on-grid SBS. In time slot t, we denote $\mathcal{U}_n[t]$ as the set of users whose nearest on-grid SBS is $b_n, \forall n \in \mathcal{B}_g$. Also, to reduce interference among users accessing different SBSs, we consider bandwidth allocations and let $\theta_{n,k}[t]$ denote the proportion of b_n's spectrum allocated to user $U_k, \forall k \in \mathcal{U}_n[t]$. Then, we can express the achievable rate from user $U_k, \forall k \in \mathcal{U}_n[t]$, to on-grid SBS b_n in time slot t, denoted by $R_{k,n}[t]$, as follows:

$$R_{k,n}[t] \triangleq \theta_{n,k}[t] W_n \times \log_2\left(1+\frac{P_{k,n}[t]\big|h_{k,n}[t]\big|^2}{\sum\limits_{m\in\Gamma(n)}\sum\limits_{k'\in\mathcal{U}_m[t]} P_{k',m}[t]\big|h_{k',n}[t]\big|^2+\sigma^2}\right) \tag{1}$$

TABLE I
SYSTEM VARIABLES

Symbol	Description
$\theta_{n,k}[t]$	Proportion of on-grid SBS b_n's spectrum allocated to user U_k in time slot t
$y_{k,n,m}[t]$	Binary variable indicating whether SBS b_n offloads user U_k's task to SBS b_m in time slot t
$c_{n,i}[T]$	Caching state of service i at SBS b_n in frame T
$x_{n,i}[t]$	Usage state of service i at SBS b_n in time slot t
$f_{n,k}[t]$	Computation resource of SBS b_n allocated to user U_k in time slot t
$\mathcal{B}_g$	Set of on-grid SBSs
$\mathcal{B}_e$	Set of off-grid SBSs
$\mathcal{B}$	Set of all on-grid SBSs and off-grid SBSs
Ω	Set of all users
$\mathcal{I}$	Set of service types
$\mathcal{U}_n[t]$	Set of users whose nearest on-grid SBS is b_n in time slot t
$\mathcal{B}(n)$	Set of SBSs which connect to on-grid SBS b_n (including SBS b_n)
$\mathcal{L}_n[t]$	Set of users for which SBS b_n needs to provide computing services in time slot t
$\xi^{\mathrm{u}}_k[t]$	Task processing time at user U_k in time slot t for local computing
$E^{\mathrm{u}}_k[t]$	Energy consumption at user U_k in time slot t for local computing
$\xi^{\mathrm{tr}}_{k,n}[t]$	Transmission time at user U_k to offload data to its nearest on-grid SBS b_n in time slot t
$E^{\mathrm{tr}}_{k,n}[t]$	Energy consumption at user U_k to offload data to its nearest on-grid SBS b_n in time slot t
$\xi^{\mathrm{tr}}_{n,m}[t]$	Transmission time at SBS b_n to offload data to SBS b_m in time slot t
$\xi^{\mathrm{pr}}_{k,n,m}[t]$	Task processing time at SBS b_m in $\mathcal{B}(n)$ to process user U_k's task in time slot t
$E^{\mathrm{pr}}_{k,n,m}[t]$	Energy consumption at SBS b_m in $\mathcal{B}(n)$ to process user U_k's task in time slot t

where W_n is the bandwidth of on-grid SBS b_n, $\Gamma(n)$ is the set of other on-grid SBSs that share the same spectrums with SBS b_n, $P_{k,n}[t]$ and $h_{k,n}[t]$ are the transmit power and the channel fading gain from user $U_k, \forall k \in \mathcal{U}_n[t]$, to on-grid SBS b_n in time slot t, respectively, and σ^2 is the power of the additive white Gaussian noise. Moreover, we define the channel fading gain in time slot t as $h_{k,n}[t] \triangleq \bar{h}_{k,n}[t]\kappa_{k,n}[t]$, where $\kappa_{k,n}[t]$ is the small-scale fading which follows Rayleigh fading and $\bar{h}_{k,n}[t]$ is the large-scale fading which follows the free-space path loss model [19], [20]. Similar to [20], we have

$$\bar{h}_{k,n}[t] = A_{\mathrm{d}}\left(\frac{3\times 10^8}{4\pi f_{\mathrm{c}} d_{k,n}[t]}\right)^{d_{\mathrm{e}}}, \tag{2}$$

where A_{d} is the antenna gain, f_{c} is the carrier frequency, d_{e} is the path loss exponent, and $d_{k,n}[t]$ is the distance between user U_k and on-grid SBS b_n in time slot t.

We let $D_k[t]$ denote the data size (in bits) of user U_k's task and $Z_k[t]$ be the number of CPU cycles required for computing one bit of U_k's task in time slot t. User U_k can partition its task into two parts [21], where one part with $D^{\mathrm{u}}_k[t]$ bits is executed locally, and the other part with $D_{k,n}[t]$ bits is offloaded to its nearest on-grid SBS b_n. We denote f_k as the computation resource, i.e., the clock frequency of the CPU chip, at user U_k for task processing [17]. Thus, if the $D^{\mathrm{u}}_k[t]$ bits of input data to be processed locally at U_k, then the task processing time and the energy consumption at U_k, denoted by $\xi^{\mathrm{u}}_k[t]$ and $E^{\mathrm{u}}_k[t]$, respectively, are given as follows:

$$\begin{cases} \xi^{\mathrm{u}}_k[t] \triangleq \dfrac{1}{f_k}\Big[D^{\mathrm{u}}_k[t] Z_k[t]\Big], & (3)\\ E^{\mathrm{u}}_k[t] \triangleq \nu f_k^2 D^{\mathrm{u}}_k[t] Z_k[t], & (4)\end{cases}$$

where ν is the effective switched capacitance [18]. Moreover, we can express the transmission time and the energy consumption at U_k to offload $D_{k,n}[t]$ bits of data to its nearest on-grid SBS b_n in time slot t, denoted by $\xi^{\text{tr}}_{k,n}[t]$ and $E^{\text{tr}}_{k,n}[t]$, respectively, as follows:

$$\begin{cases} \xi^{\text{tr}}_{k,n}[t] \triangleq \dfrac{D_{k,n}[t]}{R_{k,n}[t]}, & (5) \\ E^{\text{tr}}_{k,n}[t]{-}0.7cm \triangleq P_{k,n}[t]\xi^{\text{tr}}_{k,n}[t]. & (6) \end{cases}$$

There exist $\mathcal{I}$ types of services, indexed by $\mathcal{I} \triangleq \{1, 2, \ldots, i, \ldots, I\}$. Each user U_k needs one type of services, denoted by $i_k[t] \in \mathcal{I}$, in time slot t. When on-grid SBS b_n cannot complete the offloading task of user $U_k, \forall k \in \mathcal{U}_n[t]$, or it cannot provide service $i_k[t]$ to user U_k, it further transfers the offloading task of user U_k to other SBSs that support service $i_k[t]$ and have light computation workloads through wired links [22]. Therefore, similar to [21], the computation-offloading in our paper contains two tiers, i.e., computation-offloading from user $U_k, \forall k \in \mathcal{U}_n[t]$, to its nearest on-grid SBS b_n and computation-offloading from on-grid SBS b_n to other SBSs. We denote $\mathcal{B}(n)$ as the set of SBSs (including SBS b_n) which connect to on-grid SBS b_n. Moreover, let the binary variable $y_{k,n,m}[t] \in \{0, 1\}$ denote whether on-grid SBS b_n offloads the task of user $U_k, \forall k \in \mathcal{U}_n[t]$, to SBS $b_m, \forall m \in \mathcal{B}(n)$. The variable $y_{k,n,m}[t] = 1$, if on-grid SBS b_n offloads the task data of $U_k, \forall k \in \mathcal{U}_n[t]$, to SBS $b_m, \forall m \in \mathcal{B}(n)$; otherwise $y_{k,n,m}[t] = 0$. Please notice that $y_{k,n,n}[t] = 1$ means that on-grid SBS b_n will process task for $U_k, \forall k \in \mathcal{U}_n[t]$, by itself. Then, we can obtain that

$$\begin{aligned} D_k[t] &= D^{\text{u}}_k[t] + D_{k,n}[t] \\ &= D^{\text{u}}_k[t] + D_{k,n}[t] \sum_{m \in \mathcal{B}(n)} y_{k,n,m}[t]. \end{aligned} \tag{7}$$

Similar to [22], if the task is offloaded from on-grid SBS b_n to SBS $b_m, \forall m \in \mathcal{B}(n)$, it will be processed at SBS b_m. Let $r_{n,m}$ and $\xi^{\text{tr}}_{n,m}[t]$ denote the data transmission rate of the wired link and the transmission time for data offloading from on-grid SBS b_n to SBS $b_m, \forall m \in \mathcal{B}(n)$, respectively. Then, for SBS b_m, $\forall m \in (\mathcal{B}(n) \setminus \{n\})$, we can express $\xi^{\text{tr}}_{n,m}[t]$ as follows:

$$\xi^{\text{tr}}_{n,m}[t] \triangleq \frac{1}{r_{n,m}} \left\{ \sum_{k \in \mathcal{U}_n[t]} \Big[y_{k,n,m}[t] D_{k,n}[t] \Big] \right\}, \tag{8}$$

while $\xi^{\text{tr}}_{n,n}[t] = 0$. Let $c_{m,i}[T]$ denote the caching state of service i at SBS $b_m, \forall m \in \mathcal{B}$, in frame T, where $c_{m,i}[T] \in \{0, 1\}$. $c_{m,i}[T] = 1$ indicates that SBS b_m needs to cache service i; otherwise $c_{m,i}[T] = 0$. In addition, let $x_{m,i}[t] \in \{0, 1\}$ denote the usage state of service i at SBS b_m in time slot t, where $x_{m,i}[t] = 1$ means that service i has not been used at b_m, and $x_{m,i}[t] = 0$ otherwise. We assume that in time slot t, each SBS can simultaneously process multiple tasks requiring different services. Moreover, we denote $f_{m,k}[t]$ as the computation resource allocated to user U_k at SBS b_m in time slot t. In general, $f_{m,k}[t]$ is much larger than f_k. Then, we can express the task processing time and the computation energy consumption at SBS $b_m, \forall m \in \mathcal{B}(n)$, for processing the task of $U_k, \forall k \in \mathcal{U}_n[t]$, denoted by $\xi^{\text{pr}}_{k,n,m}[t]$ and $E^{\text{pr}}_{k,n,m}[t]$, respectively, as follows:

$$\begin{cases} \xi^{\text{pr}}_{k,n,m}[t] \\ \triangleq \dfrac{1}{f_{m,k}[t]} \Big[c_{m,i_k[t]}[T] x_{m,i_k[t]}[t] y_{k,n,m}[t] D_{k,n}[t] Z_k[t] \Big], & (9) \\ E^{\text{pr}}_{k,n,m}[t] \\ \triangleq c_{m,i_k[t]}[T] x_{m,i_k[t]}[t] y_{k,n,m}[t] \nu (f_{m,k}[t])^2 D_{k,n}[t] Z_k[t] & (10) \end{cases}$$

C. Services Fetching and Caching

The MBS has cached all services in $\mathcal{I}$. Whether SBS b_n needs to fetch service i from the MBS in frame T depends on the specific values of $c_{n,i}[T-1]$ and $c_{n,i}[T]$. We can express the time duration for SBS b_n to fetch service i from the MBS in frame T, denoted by $\xi_{n,i}[T]$, as follows:

$$\xi_{n,i}[T] \triangleq \frac{1}{r_n} \left[\beta_i c_{n,i}[T] \left(c_{n,i}[T-1] \oplus c_{n,i}[T] \right) \right], \tag{11}$$

where r_n (in bps) is the data rate of the wired link between the MBS and SBS b_n [17], β_i is the size of the ith service program (in bits), and $\oplus$ is the exclusive-or (XOR) operation. When $c_{n,i}[T-1]$ and $c_{n,i}[T]$ take the same values, we have $c_{n,i}[T-1] \oplus c_{n,i}[T] = 0$, and $c_{n,i}[T-1] \oplus c_{n,i}[T] = 1$ when $c_{n,i}[T-1]$ and $c_{n,i}[T]$ take different values. Therefore, using Eq. (11), we can know that only when $c_{n,i}[T-1] = 0$ and $c_{n,i}[T] = 1$, SBS b_n needs to fetch service i from the MBS with time duration $\xi_{n,i}[T] > 0$. Moreover, SBSs first cooperatively update their caching services at the beginning of frame T, and then help users process tasks when all SBSs finish updating services. Then, we can express the time duration for all SBSs updating services in frame T, denoted by $\xi^{\text{us}}[T]$, as follows:

$$\xi^{\text{us}}[T] \triangleq \max_{n \in \mathcal{B}} \left\{ \sum_{i \in \mathcal{I}} \xi_{n,i}[T] \right\}. \tag{12}$$

D. Non-Linear Energy Harvesting Model

From the practical point of view, the RF-based EH circuits typically exhibit non-linear end-to-end wireless power transfer [23]. Adopting the non-linear EH model developed in [23], we can express the amount of RF-energy harvested by $U_k, \forall k \in \Omega$, denoted by $E^{\text{h}}_k[t]$, and the amount of energy harvested by off-grid SBS $b_n, \forall n \in \mathcal{B}_{\text{e}}$, denoted by $E^{\text{h}}_n[t]$, in time slot t as follows:

$$\begin{cases} E^{\text{h}}_k[t] \triangleq \dfrac{\varrho}{2} \left[\dfrac{\Phi^{\text{NL}}_k[t] - M_k \varsigma_k}{1 - \varsigma_k} \right], & (13) \\ E^{\text{h}}_n[t] \triangleq \varrho \left[\dfrac{\Phi^{\text{NL}}_n[t] - M_n \varsigma_n}{1 - \varsigma_n} \right] + E^{\text{s}}_n[t], & (14) \end{cases}$$

respectively, where $E^{\text{s}}_n[t]$ denotes the amount of solar energy harvested by off-grid SBS b_n in time slot t, and M_k and M_n are the maximum harvested powers at U_k and b_n, respectively, when the EH circuits saturate. Besides, $\varsigma_k \triangleq 1/(1 + \exp(s_k z_k))$ and $\varsigma_n \triangleq 1/(1 + \exp(s_n z_n))$ are used to guarantee a zero input/output response, respectively, where s_k, z_k, s_n, and z_n are constants related to the non-linear EH circuit characteristics, e.g., the capacitance, resistance, etc. Furthermore,

$$\begin{cases} \Phi_k^{\mathrm{NL}}[t] \triangleq \dfrac{M_k}{1+\exp\left(-s_k\left(P_k^{\mathrm{h}}[t]-z_k\right)\right)}, & (15)\\ \Phi_n^{\mathrm{NL}}[t] \triangleq \dfrac{M_n}{1+\exp\left(-s_n\left(P_n^{\mathrm{h}}[t]-z_n\right)\right)}, & (16)\end{cases}$$

are the traditional logistic functions, where

$$\begin{cases} P_k^{\mathrm{h}}[t] \triangleq \sum\limits_{m\in\mathcal{E}_k[t]} \left(P_m\left|h_{m,k}[t]\right|^2\right), & (17)\\ P_n^{\mathrm{h}}[t] \triangleq \sum\limits_{m\in\mathcal{E}(n)} \left(P_m\left|h_{m,n}[t]\right|^2\right), & (18)\end{cases}$$

are the received powers for EH at user U_k and off-grid SBS b_n, respectively, where $\mathcal{E}_k[t]$ and $\mathcal{E}(n)$ denote the sets of on-grid SBSs that can wirelessly power user U_k and off-grid SBS b_n in time slot t, respectively, P_m is the transmit power of on-grid SBS b_m, and $h_{m,k}[t]$ and $h_{m,n}[t]$ are the channel fading gains from on-grid SBS b_m to user U_k and off-grid SBS b_n in time slot t, respectively.

Then, we can express the amount of energy that can be used by off-grid SBS b_n, denoted by $E_n[t]$, and that can be used by user U_k, denoted by $E_k[t]$, in time slot t as follows:

$$E_n[t] \triangleq \min\left\{E_n^{\mathrm{h}}[t-1]+E_n[t-1] - \sum_{k\in\mathcal{U}_m[t]} E_{k,m,n}^{\mathrm{pr}}[t-1], E_n^{\max}\right\}, \tag{19}$$

and

$$E_k[t] \min\left\{E_k^{\mathrm{h}}[t]+E_k[t-1]-E_k^{\mathrm{u}}[t-1] - \sum_{m:k\in\mathcal{U}_m[t]} E_{k,m}^{\mathrm{tr}}[t-1], E_k^{\max}\right\}, \tag{20}$$

respectively, where $E_k^{\max}$ and $E_n^{\max}$ are the maximum battery capacities of user U_k and off-grid SBS b_n, respectively.

E. The Optimization Problems Formulations

We aim to minimize the sum of weighted energy consumption of all off-grid SBSs in $\mathcal{B}_{\mathrm{e}}$ and all users in Ω, while satisfying the quality of services (QoS) of SBSs and users, e.g., users' task completion time. Therefore, we can formulate the considered optimization problem as follows:

$$\min_{\Theta,\mathcal{C},\mathcal{Y},\mathcal{F},\mathcal{D}} \left\{\sum_{t=1}^{\varpi}\left(\zeta\left[\sum_{k\in\Omega}E_k^{\mathrm{u}}[t]+\sum_{n\in\mathcal{B}_{\mathrm{g}}}\sum_{k\in\mathcal{U}_n[t]}E_{k,n}^{\mathrm{tr}}[t]\right]+(1-\zeta) \times\left[\sum_{n\in\mathcal{B}_{\mathrm{g}}}\sum_{k\in\mathcal{U}_n[t]}\sum_{m\in(\mathcal{B}(n)\cap\mathcal{B}_{\mathrm{e}})}E_{k,n,m}^{\mathrm{pr}}[t]\right]\right)\right\} \tag{21}$$

s.t.:

$$\begin{aligned}
&\mathrm{C1}: \sum_{i\in\mathcal{I}}\left(c_{n,i}[T]\beta_i\right) \le C_n[T], \quad \forall n\in\mathcal{B},\\
&\mathrm{C2}: \sum_{k\in\mathcal{U}_n[t]} \theta_{n,k}[t] \le 1, \quad \forall t, n\in\mathcal{B}_{\mathrm{g}},\\
&\mathrm{C3}: \sum_{m\in\mathcal{B}(n)} y_{k,n,m}[t] \le 1, \quad \forall t, n\in\mathcal{B}_{\mathrm{g}}, \quad k\in\mathcal{U}_n[t],\\
&\mathrm{C4}: c_{n,i}[T]\le 1, \quad \forall n\in\mathcal{B}, i\in\mathcal{I},\\
&\mathrm{C5}: \xi_k^{\mathrm{u}}[t] \le \frac{\varrho}{2}, \quad \forall t, k\in\Omega,\\
&\mathrm{C6}: \xi_{k,n}^{\mathrm{tr}}[t]+\xi_{n,m}^{\mathrm{tr}}[t]+\xi_{k,n,m}^{\mathrm{pr}}[t] \le \frac{\varrho}{2},\\
&\qquad\qquad \forall t, n\in\mathcal{B}_{\mathrm{g}}, \quad k\in\mathcal{U}_n[t], m\in\mathcal{B}(n),\\
&\mathrm{C7}: E_k^{\mathrm{u}}[t]+E_{k,n}^{\mathrm{tr}}[t] \le E_k[t], \quad \forall t, n\in\mathcal{B}_{\mathrm{g}}, \quad k\in\mathcal{U}_n[t],\\
&\mathrm{C8}: \sum_{m\in(\mathcal{B}_{\mathrm{g}}\cap\mathcal{B}(n))}\sum_{k\in\mathcal{U}_m[t]} E_{k,m,n}^{\mathrm{pr}}[t] \le E_n[t], \quad \forall t, n\in\mathcal{B}_{\mathrm{e}},\\
&\mathrm{C9}: \sum_{m\in(\mathcal{B}_{\mathrm{g}}\cap\mathcal{B}(n))}\sum_{k\in\mathcal{U}_m[t]} \left(y_{k,m,n}[t]f_{n,k}[t]\right) \le F_n^{\max},\\
&\qquad\qquad \forall t, n\in\mathcal{B},\\
&\mathrm{C10}: D_k^{\mathrm{u}}[t]+D_{k,n}[t]=D_k[t], \quad \forall t, n\in\mathcal{B}_{\mathrm{g}}, \quad k\in\mathcal{U}_n[t],
\end{aligned}$$

where

$$\begin{cases}
\Theta \triangleq \left\{0\le\theta_{n,k}[t]\le 1, \quad \forall t, n\in\mathcal{B}_{\mathrm{g}}, k\in\mathcal{U}_n[t]\right\},\\
\mathcal{C} \triangleq \left\{c_{n,i}[T]\in\{0,1\}, \quad \forall n\in\mathcal{B}, i\in\mathcal{I}\right\},\\
\mathcal{Y} \triangleq \left\{y_{k,n,m}[t]\in\{0,1\}, \quad \forall t, n\in\mathcal{B}_{\mathrm{g}}, k\in\mathcal{U}_n[t], m\in\mathcal{B}(n)\right\},\\
\mathcal{F} \triangleq \left\{0\le f_{n,k}[t]\le F_n^{\max}, \quad \forall t, n\in\mathcal{B}, k\in\Omega\right\},\\
\mathcal{D} \triangleq \left\{0\le D_k^{\mathrm{u}}[t], D_{k,n}[t]\le D_k[t], \quad \forall t, n\in\mathcal{B}_{\mathrm{g}}, k\in\mathcal{U}_n[t]\right\},
\end{cases}$$

where $C_n[T]$ is the available caching storage capacity of SBS b_n in frame T, $F_n^{\max}$ is the total computation resource of SBS b_n, and $\zeta \in [0,1]$ is a weight factor. C1 is the caching capacity constraint of each SBS. C2 is the bandwidth allocations constraint of SBS $b_n, \forall n\in\mathcal{B}_{\mathrm{g}}$. C3 indicates that SBS b_n can only offload user U_k's task to one SBS b_m in $\mathcal{B}(n)$. C5-C6 are the task completion time constraints of user $U_k, \forall k\in\Omega$, for local-computing and edge-computing, respectively. C7-C8 are the energy consumption constraints of user $U_k, \forall k\in\Omega$, and off-grid SBS $b_n, \forall n\in\mathcal{B}_{\mathrm{e}}$, respectively. C9 is the computation resource-allocations constraint of SBS $b_n, \forall n\in\mathcal{B}$. C10 is the flow conservation constraint for user $U_k, \forall k\in\Omega$.

III. Jointly Optimizing Cooperative Service-Caching, Computation-Offloading, and Resource-Allocations for Scenarios With SUs

For UDNs, it is challenging to solve the large-size optimization problem in Eq. (21) by using the traditional optimization based methods with low complexity. Hence, we will leverage the advanced deep reinforcement learning (DRL) based methods to solve this problem with the help of deep neural networks (DNNs) [2], [18]. Service-caching and computation-offloading work in two different timescales. Therefore, based on the hierarchical multi-agent deep reinforcement learning (HMDRL), we will develop a two-timescale based joint cooperative service-caching, computation-offloading, and resource-allocations scheme for scenarios with SUs [24]. Specifically, using HMDRL, we first derive SBSs' cooperative service-caching policies in each time frame T. Then, based on the cooperative service-caching policies, we derive users' and SBSs' computation-offloading policies in each time slot t. Finally, we derive SBSs' computation resource-allocations policies in each time slot t according to the obtained service-caching and computation-offloading policies.

First, to reduce energy consumptions of off-grid SBSs and all users while satisfying their delay and energy constraints, we define the total reward in time frame T, denoted by $r[T]$, as follows:

$$
\begin{aligned}
r[T] \triangleq \sum_{t=1}^{\varpi} \Bigg\{ & -\zeta \Bigg[\sum_{k\in\Omega} E_k^{\mathrm{u}}[t] + \sum_{n\in\mathcal{B}_{\mathrm{g}}} \sum_{k\in\mathcal{U}_n[t]} E_{k,n}^{\mathrm{tr}}[t] \Bigg] \\
& - (1-\zeta) \sum_{n\in\mathcal{B}_{\mathrm{g}}} \sum_{k\in\mathcal{U}_n[t]} \sum_{m\in(\mathcal{B}(n)\cap\mathcal{B}_{\mathrm{e}})} E_{k,n,m}^{\mathrm{pr}}[t] \\
& + \sum_{n\in\mathcal{B}_{\mathrm{g}}} \sum_{m\in\mathcal{B}(n)} \sum_{k\in\mathcal{U}_n[t]} \Upsilon_{k,n,m}^{\mathrm{ti}}[t] + \sum_{n\in\mathcal{B}_{\mathrm{e}}} \Upsilon_n^{\mathrm{en}}[t] \\
& + \sum_{k\in\Omega} \Big[\Upsilon_k^{\mathrm{ti}}[t] + \Upsilon_k^{\mathrm{en}}[t] \Big] \Bigg\}, \qquad (22)
\end{aligned}
$$

where $\Upsilon_k^{\mathrm{ti}}[t], \Upsilon_k^{\mathrm{en}}[t], \Upsilon_{k,n,m}^{\mathrm{ti}}[t], \ \Upsilon_n^{\mathrm{en}}[t] \ \leq \ 0$ are defined as follows:

$$
\Upsilon_k^{\mathrm{ti}}[t] \triangleq \begin{cases} \varphi_k^{\mathrm{ti}}, & \text{if } \xi_k^{\mathrm{u}}[t] > \dfrac{\varrho}{2}, \\ 0, & \text{otherwise}, \end{cases} \qquad (23)
$$

$$
\Upsilon_{k,n,m}^{\mathrm{ti}}[t] \triangleq \begin{cases} \varphi_{k,n,m}^{\mathrm{ti}}, & \text{if } \xi_{k,n}^{\mathrm{tr}}[t] + \xi_{n,m}^{\mathrm{tr}}[t] + \xi_{k,n,m}^{\mathrm{pr}}[t] > \dfrac{\varrho}{2}, \\ 0, & \text{otherwise}, \end{cases} \qquad (24)
$$

$$
\Upsilon_k^{\mathrm{en}}[t] \triangleq \begin{cases} \varphi_k^{\mathrm{en}}, & \text{if } E_k^{\mathrm{u}}[t] + E_{k,n}^{\mathrm{tr}}[t] > E_k[t], \\ 0, & \text{otherwise}, \end{cases} \qquad (25)
$$

and (26)

$$
\Upsilon_n^{\mathrm{en}}[t] \triangleq \begin{cases} \varphi_n^{\mathrm{en}}, & \text{if } \displaystyle\sum_{m\in(\mathcal{B}_{\mathrm{g}}\cap\mathcal{B}(n))} \sum_{k\in\mathcal{U}_m[t]} E_{k,m,n}^{\mathrm{pr}}[t] \\ & \qquad > E_n[t], \\ 0, & \text{otherwise}, \end{cases} \qquad (27)
$$

respectively, where $\varphi_k^{\mathrm{ti}}, \varphi_{k,n,m}^{\mathrm{ti}}, \varphi_k^{\mathrm{en}}$, and φ_n^{en} are all negative constants which are introduced to punish users or SBSs for violating time constraints C5-C6 and energy constraints C7-C8, respectively.

A. Slow Timescale: Cooperative Service-Caching

In service-caching, each SBS is treated as an agent and all SBSs cooperate with each other to decide the service-caching variables $c_{n,i}[T]$'s in each time frame T. Since there are a large number of discrete variables $c_{n,i}[T]$'s, we will utilize Deep Deterministic Policy Gradient (DDPG) [25], which can learn the deterministic policy for high-dimensional continuous action spaces, to decide $c_{n,i}[T]$'s by relaxing $c_{n,i}[T]$'s as real-valued variables taking values within [0, 1].

In time frame T, we define the state of SBS $b_n, \forall n \in \mathcal{B}$, for service-caching, denoted by $\mathcal{O}_n^{\mathrm{c}}[T]$, as follows:

$$
\mathcal{O}_n^{\mathrm{c}}[T] \triangleq \begin{cases} \big\{ c_{m,i}[T-1], \psi_{n,i}[T-1], \quad \forall i \in \mathcal{I}, \\ \qquad\qquad m \in \mathcal{B}(n) \big\}, \quad \forall n \in \mathcal{B}_{\mathrm{g}}, \\ \big\{ c_{n,i}[T-1], c_{m,i}[T], \psi_{n,i}[T-1], \\ \quad \forall i \in \mathcal{I}, m \in (\mathcal{B}(n) \cap \mathcal{B}_{\mathrm{g}}) \big\}, \quad \forall n \in \mathcal{B}_{\mathrm{e}}, \end{cases} \qquad (28)
$$

where $\psi_{n,i}[T-1]$ denotes the number of times service i, $\forall i \in \mathcal{I}$, is requested at SBS $b_n, \forall n \in \mathcal{B}$, in time frame $(T-1)$. For cooperative service-caching, on-grid SBS b_n, $\forall n \in \mathcal{B}_{\mathrm{g}}$, needs to know the caching state $c_{m,i}[T-1]$ of service i, $\forall i \in \mathcal{I}$, at SBS $b_m, \forall m \in \mathcal{B}(n)$, in time frame $(T-1)$. Similarly, off-grid SBS b_n, $\forall n \in \mathcal{B}_{\mathrm{e}}$, needs to know the caching state $c_{m,i}[T]$ of service i, $\forall i \in \mathcal{I}$, at on-grid SBS $b_m, \forall m \in (\mathcal{B}(n) \cap \mathcal{B}_{\mathrm{g}})$, in time frame T. Moreover, all on-grid SBSs perform service-caching simultaneously before the off-grid SBSs, and they will cache as many services as possible to reduce energy consumption of the off-grid SBSs for task processing.

Furthermore, in time frame T, we define the action of SBS $b_n, \forall n \in \mathcal{B}$, for service-caching, denoted by $a_n^{\mathrm{c}}[T]$, as follows:

$$
a_n^{\mathrm{c}}[T] \triangleq \{ c_{n,1}[T], \ldots, c_{n,i}[T], \ldots, c_{n,I}[T] \}. \qquad (29)
$$

In addition, we define the reward of SBS $b_n, \forall n \in \mathcal{B}$, for service-caching in time frame T, denoted by $r_n^{\mathrm{c}}[T]$, as follows:

$$
r_n^{\mathrm{c}}[T] \triangleq \frac{\sum_{i\in\mathcal{I}} c_{n,i}[T]\psi_{n,i}[T]}{\sum_{i\in\mathcal{I}} \psi_{n,i}[T]} - \sum_{i\in\mathcal{I}} \xi_{n,i}[T], \qquad (30)
$$

where the first term of the right hand side of Eq. (30) is the service-caching hit rate of SBS b_n, which is obtained by dividing the requested times, i.e., $\sum_{i\in\mathcal{I}} c_{n,i}[T]\psi_{n,i}[T]$, of SBS b_n's cached services by the total number of times all services in $\mathcal{I}$ are requested at SBS b_n in time frame T, i.e., $\sum_{i\in\mathcal{I}} \psi_{n,i}[T]$ [24]. By defining $r_n^{\mathrm{c}}[T]$, we aim to maximize the cumulative service-caching hit rate of all services while reducing the service fetching time at SBS b_n. When the reward $r_n^{\mathrm{c}}[T]$ defined in Eq. (30) takes a large value, the hit rate generally takes a large value and the service fetching time takes a small value [24]. As a result, users can have more opportunities to offload tasks to MEC servers, and then $r[T]$ defined in Eq. (22) increases.

Based on the above defined $\mathcal{O}_n^{\mathrm{c}}[T]$, $a_n^{\mathrm{c}}[T]$, and $r_n^{\mathrm{c}}[T]$, we use DDPG to derive the service-caching policies of all SBSs. The DDPG includes four DNNs: the actor network, the critic network, and two corresponding target networks [25]. Based on the observed state $\mathcal{O}_n^{\mathrm{c}}[T]$, the actor network of SBS $b_n, \forall n \in \mathcal{B}$, will train a policy function $\pi_n^{\mathrm{c}}(\mathcal{O}_n^{\mathrm{c}}; \boldsymbol{\omega}_{\mathrm{an},n}^{\mathrm{c}})$ to generate an action $a_n^{\mathrm{c}}[T]$ for SBS b_n at the beginning of time frame T, where $\boldsymbol{\omega}_{\mathrm{an},n}^{\mathrm{c}}$ is the parameter vector (including the weight parameters and the bias parameters) of the actor network [25]. Moreover, in order to explore more actions, the DDPG will add a Gaussian noise u to the policy function $\pi_n^{\mathrm{c}}(\mathcal{O}_n^{\mathrm{c}}; \boldsymbol{\omega}_{\mathrm{an},n}^{\mathrm{c}})$. Then, the DDPG decides $a_n^{\mathrm{c}}[T]$ by using the following policy [25]:

$$
a_n^{\mathrm{c}}[T] \triangleq \pi_n^{\mathrm{c}}(\mathcal{O}_n^{\mathrm{c}}[T]; \boldsymbol{\omega}_{\mathrm{an},n}^{\mathrm{c}}) + u. \qquad (31)
$$

To evaluate $a_n^{\mathrm{c}}[T]$, the critic network of SBS b_n will generate a Q-value, i.e., $Q_n^{\mathrm{c}}(\mathcal{O}_n^{\mathrm{c}}[T], a_n^{\mathrm{c}}[T]; \boldsymbol{\omega}_{\mathrm{cn},n}^{\mathrm{c}})$, based on its Q-function $Q_n^{\mathrm{c}}\left(\mathcal{O}_n^{\mathrm{c}}, a_n^{\mathrm{c}}; \boldsymbol{\omega}_{\mathrm{cn},n}^{\mathrm{c}}\right)$, where $\boldsymbol{\omega}_{\mathrm{cn},n}^{\mathrm{c}}$ is the parameter vector of the critic network [25]. Once we obtain $a_n^{\mathrm{c}}[T]$, we can determine the service-caching of SBS b_n, $\forall n \in \mathcal{B}$. Specifically, we first sort the services in $\mathcal{I}$ in the descending order of the obtained real-valued $c_{n,i}[T]$'s, and then services are cached according to the above-sorted sequence (i.e., the service i with the largest $c_{n,i}[T]$ will be cached first) until constraint C1 is violated.

Furthermore, the DDPG will utilize the experience replay buffer and the target networks to improve and stabilize the training process [26]. In each time frame T, SBS $b_n, \forall n \in \mathcal{B}$, will store the current transition, i.e., $(\mathcal{O}_n^{\mathrm{c}}[T]$, $a_n^{\mathrm{c}}[T]$, $r_n^{\mathrm{c}}[T]$, $\mathcal{O}_n^{\mathrm{c}}[T+1])$, into its experience replay buffer $\mathcal{M}_n^{\mathrm{c}}$. Also, it will randomly sample a batch of transitions in $\mathcal{M}_n^{\mathrm{c}}$ to train its actor and critic networks. Let $\widetilde{\pi}_n^{\mathrm{c}}(\mathcal{O}_n^{\mathrm{c}}; \widetilde{\boldsymbol{\omega}}_{\mathrm{an},n}^{\mathrm{c}})$ and $\widetilde{Q}_n^{\mathrm{c}}\left(\mathcal{O}_n^{\mathrm{c}}, a_n^{\mathrm{c}}; \widetilde{\boldsymbol{\omega}}_{\mathrm{cn},n}^{\mathrm{c}}\right)$ denote the policy function and Q-function of the target actor network and target critic network, respectively, where $\widetilde{\boldsymbol{\omega}}_{\mathrm{an},n}^{\mathrm{c}}$ and $\widetilde{\boldsymbol{\omega}}_{\mathrm{cn},n}^{\mathrm{c}}$ are the corresponding parameter vectors. Randomly sampling a batch of transitions $(\mathcal{O}_n^{\mathrm{c}}[T']$, $a_n^{\mathrm{c}}[T']$, $r_n^{\mathrm{c}}[T']$, $\mathcal{O}_n^{\mathrm{c}}[T'+1])$ with size Ψ from $\mathcal{M}_n^{\mathrm{c}}$, the DDPG updates $\boldsymbol{\omega}_{\mathrm{cn},n}^{\mathrm{c}}$ of its critic network by minimizing the following loss function [26]:

$$L\left(\boldsymbol{\omega}_{\mathrm{cn},n}^{\mathrm{c}}\right) \triangleq \frac{1}{\Psi}\left\{\sum_{T'}\left(J_n^{\mathrm{c}}[T'] - Q_n^{\mathrm{c}}\left(\mathcal{O}_n^{\mathrm{c}}[T'], a_n^{\mathrm{c}}[T']; \boldsymbol{\omega}_{\mathrm{cn},n}^{\mathrm{c}}\right)\right)^2\right\}, \tag{32}$$

where in time frame T',

$$\begin{aligned} J_n^{\mathrm{c}}[T'] = r_n^{\mathrm{c}}[T'] + \gamma \widetilde{Q}_n^{\mathrm{c}}\big(&\mathcal{O}_n^{\mathrm{c}}[T'+1], \\ &\widetilde{\pi}_n^{\mathrm{c}}\left(\mathcal{O}_n^{\mathrm{c}}[T'+1]; \widetilde{\boldsymbol{\omega}}_{\mathrm{an},n}^{\mathrm{c}}\right); \widetilde{\boldsymbol{\omega}}_{\mathrm{cn},n}^{\mathrm{c}}\big), \end{aligned} \tag{33}$$

where $\gamma \in (0,1)$ is a discount factor.

Utilizing the selected transitions from $\mathcal{M}_n^{\mathrm{c}}$, the DDPG updates the parameter vector $\boldsymbol{\omega}_{\mathrm{an},n}^{\mathrm{c}}$ of the actor network by using the following formula [26]:

$$\begin{aligned} \boldsymbol{\omega}_{\mathrm{an},n}^{\mathrm{c}} \leftarrow \boldsymbol{\omega}_{\mathrm{an},n}^{\mathrm{c}} \frac{\alpha_{\mathrm{an}}^{\mathrm{p}}}{\Psi}\bigg\{\sum_{T'}\Big(&\nabla_{a_n^{\mathrm{c}}} Q_n^{\mathrm{c}}\left(\mathcal{O}_n^{\mathrm{c}}[T'], a_n^{\mathrm{c}}[T']; \boldsymbol{\omega}_{\mathrm{cn},n}^{\mathrm{c}}\right) \\ &\times \nabla_{\boldsymbol{\omega}_{\mathrm{an},n}^{\mathrm{c}}} \pi_n^{\mathrm{c}}\left(\mathcal{O}_n^{\mathrm{c}}[T']; \boldsymbol{\omega}_{\mathrm{an},n}^{\mathrm{c}}\right)\Big)\bigg\}, \end{aligned} \tag{34}$$

where $\alpha_{\mathrm{an}}^{\mathrm{p}} \in (0,1)$ is the learning rate of the DDPG's actor network for updating $\boldsymbol{\omega}_{\mathrm{an},n}^{\mathrm{c}}, \nabla_{a_n^{\mathrm{c}}} Q_n^{\mathrm{c}}(\mathcal{O}_n^{\mathrm{c}}[T'], a_n^{\mathrm{c}}[T']; \boldsymbol{\omega}_{\mathrm{cn},n}^{\mathrm{c}})$ is the gradient of the critic network's Q-function $Q_n^{\mathrm{c}}\left(\mathcal{O}_n^{\mathrm{c}}, a_n^{\mathrm{c}}; \boldsymbol{\omega}_{\mathrm{cn},n}^{\mathrm{c}}\right)$ with respect to (w.r.t.) action a_n^{c} in time frame T', and $\nabla_{\boldsymbol{\omega}_{\mathrm{an},n}^{\mathrm{c}}} \pi_n^{\mathrm{c}}\left(\mathcal{O}_n^{\mathrm{c}}[T']; \boldsymbol{\omega}_{\mathrm{an},n}^{\mathrm{c}}\right)$ is the gradient of the actor network's policy function $\pi_n^{\mathrm{c}}\left(\mathcal{O}_n^{\mathrm{c}}; \boldsymbol{\omega}_{\mathrm{an},n}^{\mathrm{c}}\right)$ w.r.t. $\boldsymbol{\omega}_{\mathrm{an},n}^{\mathrm{c}}$ in time frame T'. Notice that since the state $\mathcal{O}_n^{\mathrm{c}}[T]$ (see Eq. (28)) of SBS b_n is related to the service-caching policies of SBS $b_m, \forall m \in \mathcal{B}(n)$, the updating of $\boldsymbol{\omega}_{\mathrm{an},n}^{\mathrm{c}}$ and $\boldsymbol{\omega}_{\mathrm{cn},n}^{\mathrm{c}}$ is affected by the states and actions of SBS $b_m, \forall m \in \mathcal{B}(n)$.

Every G^{c} time frames, we update the parameter vectors of the target networks by using the following operational formulas [26]:

$$\begin{cases} \widetilde{\boldsymbol{\omega}}_{\mathrm{an},n}^{\mathrm{c}} \leftarrow \tau^{\mathrm{p}} \boldsymbol{\omega}_{\mathrm{an},n}^{\mathrm{c}} + (1-\tau^{\mathrm{p}})\, \widetilde{\boldsymbol{\omega}}_{\mathrm{an},n}^{\mathrm{c}}, & (35) \\ \widetilde{\boldsymbol{\omega}}_{\mathrm{cn},n}^{\mathrm{c}} \leftarrow \tau^{\mathrm{p}} \boldsymbol{\omega}_{\mathrm{cn},n}^{\mathrm{c}} + (1-\tau^{\mathrm{p}})\, \widetilde{\boldsymbol{\omega}}_{\mathrm{cn},n}^{\mathrm{c}}, & (36) \end{cases}$$

where $\tau^{\mathrm{p}} \in (0,1)$ is the learning rate of DDPG for updating $\widetilde{\boldsymbol{\omega}}_{\mathrm{an},n}^{\mathrm{c}}$ and $\widetilde{\boldsymbol{\omega}}_{\mathrm{cn},n}^{\mathrm{c}}$.

B. Fast Timescale: Computation-Offloading

Since user U_k first offloads task to its nearest on-grid SBS, only on-grid SBSs need to decide the computation-offloading variables, i.e., $D_{k,n}[t]$'s and $y_{k,n,m}[t]$'s, and the related spectrum allocation variables $\theta_{n,k}[t]$'s. We use Dueling Deep Q Network (Dueling DQN) and DDPG to decide discrete variables $y_{k,n,m}[t]$'s and continuous variables $D_{k,n}[t]$'s and $\theta_{n,k}[t]$'s, respectively.

For SBS $b_n, \forall n \in \mathcal{B}_{\mathrm{g}}$, we define the states of DDPG and Dueling DQN for computation-offloading in time slot t, denoted by $\mathcal{O}_n^{\mathrm{oc}}[t]$ and $\mathcal{O}_n^{\mathrm{od}}[t]$, respectively, as follows:

$$\begin{cases} \mathcal{O}_n^{\mathrm{oc}}[t] \triangleq \big\{h_{k,n}[t], D_k[t], Z_k[t], E_k[t], E_m[t], \\ \qquad\qquad \forall k \in \mathcal{U}_n[t], m \in (\mathcal{B}(n) \cap \mathcal{B}_{\mathrm{e}})\big\}, & (37) \\ \mathcal{O}_n^{\mathrm{od}}[t] \triangleq \big\{i_k[t], E_m[t], c_{\iota,i}[T], x_{\iota,i}[t], \forall k \in \mathcal{U}_n[t], \\ \qquad\qquad i \in \mathcal{I}, m \in (\mathcal{B}(n) \cap \mathcal{B}_{\mathrm{e}}), \iota \in \mathcal{B}(n)\big\}. & (38) \end{cases}$$

Moreover, the corresponding actions of DDPG and Dueling DQN, denoted by $a_n^{\mathrm{oc}}[t]$ and $a_n^{\mathrm{od}}[t]$, respectively, are defined as follows:

$$\begin{cases} a_n^{\mathrm{oc}}[t] \triangleq \{\theta_{n,k}[t], D_{k,n}[t], \forall k \in \mathcal{U}_n[t]\}, & (39) \\ a_n^{\mathrm{od}}[t] \triangleq \{y_{k,n,m}[t], \forall k \in \mathcal{U}_n[t], m \in \mathcal{B}(n)\}. & (40) \end{cases}$$

In Eqs. (37)-(38), for effective computation-offloading, $\mathcal{O}_n^{\mathrm{oc}}[t]$ and $\mathcal{O}_n^{\mathrm{od}}[t]$ of SBS b_n also include the available energy $E_m[t]$ at off-grid SBS b_m, $\forall m \in (\mathcal{B}(n) \cap \mathcal{B}_{\mathrm{e}})$, and/or the service-caching and usage states, i.e., $c_{\iota,i}[T]$'s and $x_{\iota,i}[t]$'s, at SBS b_ι, $\forall \iota \in \mathcal{B}(n)$. Moreover, notice that SBSs b_n's in $\mathcal{B}_{\mathrm{g}}$ derive $a_n^{\mathrm{od}}[t]$'s in a specified sequence [27]. Then, in time slot t, the action $a_{n'}^{\mathrm{od}}[t]$ taken by a given SBS $b_{n'}$ in $\mathcal{B}_{\mathrm{g}}$ may influence the service usage state $x_{m,i}[t]$ of SBS b_m, $\forall m \in \mathcal{B}(n')$. Therefore, $x_{m,i}[t]$ may take different values in different on-grid SBSs' states $\mathcal{O}_n^{\mathrm{oc}}[t]$'s in time slot t.

Furthermore, for SBS $b_n, \forall n \in \mathcal{B}_{\mathrm{g}}$, we define the rewards of DDPG and Dueling DQN in time slot t, denoted by $r_n^{\mathrm{oc}}[t]$ and $r_n^{\mathrm{od}}[t]$, respectively, as follows:

$$\begin{cases} r_n^{\mathrm{oc}}[t] \triangleq -\displaystyle\sum_{k \in \mathcal{U}_n[t]}\left(E_k^{\mathrm{u}}[t] + E_{k,n}^{\mathrm{tr}}[t]\right) \\ \qquad + \displaystyle\sum_{k \in \mathcal{U}_n[t]}\left(\Upsilon_k^{\mathrm{ti}}[t] + \Upsilon_k^{\mathrm{en}}[t]\right) + (1-\zeta)\sum_{m \in \mathcal{B}(n)} r_m^{\mathrm{cr}}[t], & (41) \\ r_n^{\mathrm{od}}[t] \triangleq \displaystyle\sum_{k \in \mathcal{U}_n[t]}\sum_{m \in \mathcal{B}(n)} y_{k,n,m}[t]\left(\frac{c_{m,i}[T] + x_{m,i}[t]}{2}\right), & (42) \end{cases}$$

where $r_m^{\mathrm{cr}}[t]$ in Eq. (41) is the computation resource-allocations reward of SBS b_m, $\forall m \in \mathcal{B}(n)$, which will be defined in the next subsection, and $\zeta \in [0,1]$ is the weight parameter given in Eq. (21). For SBS b_n, since the action $a_n^{\mathrm{oc}}[t]$ also affects the computation resource-allocations of SBS b_m, $\forall m \in \mathcal{B}(n)$, we also consider $r_m^{\mathrm{cr}}[t]$ in Eq. (41). By defining $r_n^{\mathrm{od}}[t]$ given in Eq. (42), SBS b_n aims to offload user U_k's task to SBS b_m, $\forall m \in \mathcal{B}(n)$, which has cached service $i_k[t]$ but has not utilized service $i_k[t]$ in time slot t. The higher $r_n^{\mathrm{od}}[t]$ is, the more users can select suitable SBSs for task processing. Hence, the total reward $r[T]$ given in Eq. (22) will become more and more large.

When deciding $a_n^{\rm oc}[t]$'s, the detailed updating process of DDPG is similar to that in Section III-A. Hence, we only introduce Dueling DQN in the following. The Dueling DQN includes two DNNs: the Q network and the target Q network [28]. Based on the observed state $\mathcal{O}_n^{\rm od}[t]$, the Q network gets an action $a_n^{\rm od}[t]$ by adopting the following ϵ-greedy policy:

$$a_n^{\rm od}[t] \triangleq \begin{cases} \underset{a_n^{\rm od}\in\mathcal{A}_n}{\operatorname{argmax}}\ Q_n^{\rm od}\big(\mathcal{O}_n^{\rm od}[t], a_n^{\rm od}; \boldsymbol{\omega}_n^{\rm od}, \boldsymbol{\omega}_{{\rm sv},n}^{\rm od}, \boldsymbol{\omega}_{{\rm av},n}^{\rm od}\big), & \text{if } p_n[t] > \epsilon \\ \text{Randomly select an action,} & \text{otherwise} \end{cases} \tag{43}$$

where for SBS b_n, $\mathcal{A}_n$ is the action space of Dueling DQN, $p_n[t] \in [0,1]$ is a random value chosen in time slot t, and the Q-function $Q_n^{\rm od}\left(\mathcal{O}_n^{\rm od}, a_n^{\rm od}; \boldsymbol{\omega}_n^{\rm od}, \boldsymbol{\omega}_{{\rm sv},n}^{\rm od}, \boldsymbol{\omega}_{{\rm av},n}^{\rm od}\right)$ is defined as follows:

$$\begin{aligned} &Q_n^{\rm od}\left(\mathcal{O}_n^{\rm od}, a_n^{\rm od}; \boldsymbol{\omega}_n^{\rm od}, \boldsymbol{\omega}_{{\rm sv},n}^{\rm od}, \boldsymbol{\omega}_{{\rm av},n}^{\rm od}\right) \\ &\triangleq V_n\left(\mathcal{O}_n^{\rm od}; \boldsymbol{\omega}_n^{\rm od}, \boldsymbol{\omega}_{{\rm sv},n}^{\rm od}\right) + A_n\left(\mathcal{O}_n^{\rm od}, a_n^{\rm od}; \boldsymbol{\omega}_n^{\rm od}, \boldsymbol{\omega}_{{\rm av},n}^{\rm od}\right) \\ &\quad - \frac{1}{|\mathcal{A}_n|}\left[\sum_{\widetilde{a}_n^{\rm od}\in\mathcal{A}_n} A_n\left(\mathcal{O}_n^{\rm od}, \widetilde{a}_n^{\rm od}; \boldsymbol{\omega}_n^{\rm od}, \boldsymbol{\omega}_{{\rm av},n}^{\rm od}\right)\right]. \end{aligned} \tag{44}$$

In Eq. (44), $V_n(\mathcal{O}_n^{\rm od}; \boldsymbol{\omega}_n^{\rm od}, \boldsymbol{\omega}_{{\rm sv},n}^{\rm od})$ denotes the state-value of state $\mathcal{O}_n^{\rm od}$ with network parameters $\boldsymbol{\omega}_n^{\rm od}$ and $\boldsymbol{\omega}_{{\rm sv},n}^{\rm od}$ [28]. $A_n(\mathcal{O}_n^{\rm od}, a_n^{\rm od}; \boldsymbol{\omega}_n^{\rm od}, \boldsymbol{\omega}_{{\rm av},n}^{\rm od})$ denotes the action-advantage value of $a_n^{\rm od}$ under state $\mathcal{O}_n^{\rm od}$ with network parameters $\boldsymbol{\omega}_n^{\rm od}$ and $\boldsymbol{\omega}_{{\rm av},n}^{\rm od}$ [28]. Besides, for SBS b_n, $\forall n \in \mathcal{B}_{\rm g}$, $|\mathcal{A}_n|$ is the cardinality of the action space $\mathcal{A}_n$ [28].

Selecting a min-batch of transitions $\left(\mathcal{O}_n^{\rm od}[t'], a_n^{\rm od}[t'], r_n^{\rm od}[t'], \mathcal{O}_n^{\rm od}[t'+1]\right)$ with size Ψ from the replay buffer, for SBS $b_n, \forall n \in \mathcal{B}_{\rm g}$, Dueling DQN updates $\boldsymbol{\omega}_n^{\rm od}$, $\boldsymbol{\omega}_{{\rm sv},n}^{\rm od}$, and $\boldsymbol{\omega}_{{\rm av},n}^{\rm od}$ by minimizing the following loss function:

$$\begin{aligned} L\left(\boldsymbol{\omega}_n^{\rm od}, \boldsymbol{\omega}_{{\rm sv},n}^{\rm od}, \boldsymbol{\omega}_{{\rm av},n}^{\rm od}\right) \triangleq \frac{1}{\Psi}\Bigg\{\sum_{t'}\bigg(J_n^{\rm od}[t'] - Q_n^{\rm od}\Big(\mathcal{O}_n^{\rm od}[t'], \\ a_n^{\rm od}[t']; \boldsymbol{\omega}_n^{\rm od}, \boldsymbol{\omega}_{{\rm sv},n}^{\rm od}, \boldsymbol{\omega}_{{\rm av},n}^{\rm od}\Big)\bigg)^2\Bigg\}, \end{aligned} \tag{45}$$

where in time slot t',

$$\begin{aligned} J_n^{\rm od}[t'] = r_n^{\rm od}[t'] + \gamma \widetilde{Q}_n^{\rm od}\big(\mathcal{O}_n^{\rm od}[t'+1], \widetilde{a}_n^{\rm od}[t'+1]; \\ \widetilde{\boldsymbol{\omega}}_n^{\rm od}, \widetilde{\boldsymbol{\omega}}_{{\rm sv},n}^{\rm od}, \widetilde{\boldsymbol{\omega}}_{{\rm av},n}^{\rm od}\big), \end{aligned} \tag{46}$$

where $\widetilde{Q}_n^{\rm od}\left(\mathcal{O}_n^{\rm od}, a_n^{\rm od}; \widetilde{\boldsymbol{\omega}}_n^{\rm od}, \widetilde{\boldsymbol{\omega}}_{{\rm sv},n}^{\rm od}, \widetilde{\boldsymbol{\omega}}_{{\rm av},n}^{\rm od}\right)$ is the Q-function of the target Q network of Dueling DQN, and $\widetilde{\boldsymbol{\omega}}_n^{\rm od}, \widetilde{\boldsymbol{\omega}}_{{\rm sv},n}^{\rm od}$, and $\widetilde{\boldsymbol{\omega}}_{{\rm av},n}^{\rm od}$ are its network parameter vectors. Moreover,

$$\widetilde{a}_n^{\rm od}[t'+1] = \underset{a_n^{\rm od}\in\mathcal{A}_n}{\operatorname{argmax}}\ \widetilde{Q}_n^{\rm od}\left(\mathcal{O}_n^{\rm od}[t'+1], a_n^{\rm od}; \widetilde{\boldsymbol{\omega}}_n^{\rm od}, \widetilde{\boldsymbol{\omega}}_{{\rm sv},n}^{\rm od}, \widetilde{\boldsymbol{\omega}}_{{\rm av},n}^{\rm od}\right) \tag{47}$$

is the action of the target Q network in time slot t' obtained based on state $\mathcal{O}_n^{\rm od}[t'+1]$. Dueling DQN minimizes the loss function $L\left(\boldsymbol{\omega}_n^{\rm od}, \boldsymbol{\omega}_{{\rm sv},n}^{\rm od}, \boldsymbol{\omega}_{{\rm av},n}^{\rm od}\right)$ by using the gradient descent method. For example, we can update $\boldsymbol{\omega}_n^{\rm od}$ by using the following operational formula [29]:

$$\boldsymbol{\omega}_n^{\rm od} \leftarrow \boldsymbol{\omega}_n^{\rm od} - \alpha^{\rm q} \nabla_{\boldsymbol{\omega}_n^{\rm od}} L\left(\boldsymbol{\omega}_n^{\rm od}, \boldsymbol{\omega}_{{\rm sv},n}^{\rm od}, \boldsymbol{\omega}_{{\rm av},n}^{\rm od}\right), \tag{48}$$

where $\alpha^{\rm q} \in (0,1)$ is the learning rate of Dueling DQN, $\nabla_{\boldsymbol{\omega}_n^{\rm od}} L\left(\boldsymbol{\omega}_n^{\rm od}, \boldsymbol{\omega}_{{\rm sv},n}^{\rm od}, \boldsymbol{\omega}_{{\rm av},n}^{\rm od}\right)$ is the gradient of $L\left(\boldsymbol{\omega}_n^{\rm od}, \boldsymbol{\omega}_{{\rm sv},n}^{\rm od}, \boldsymbol{\omega}_{{\rm av},n}^{\rm od}\right)$ w.r.t. $\boldsymbol{\omega}_n^{\rm od}$. In addition, every $G^{\rm od}$ time slots, the target Q network updates $\widetilde{\boldsymbol{\omega}}_n^{\rm od}, \widetilde{\boldsymbol{\omega}}_{{\rm sv},n}^{\rm od}$, and $\widetilde{\boldsymbol{\omega}}_{{\rm av},n}^{\rm od}$ by setting $\widetilde{\boldsymbol{\omega}}_n^{\rm od} = \boldsymbol{\omega}_n^{\rm od}$, $\widetilde{\boldsymbol{\omega}}_{{\rm sv},n}^{\rm od} = \boldsymbol{\omega}_{{\rm sv},n}^{\rm od}$, and $\widetilde{\boldsymbol{\omega}}_{{\rm av},n}^{\rm od} = \boldsymbol{\omega}_{{\rm av},n}^{\rm od}$, respectively.

If SBS b_n has cached one type of service, e.g., $i_k[t]$, required by user U_k, but $D_{k,n}[t] = 0$ or $y_{k,m,n}[t] = 0$, $\forall m \in \mathcal{B}_{\rm g}, n \in \mathcal{B}(m)$, SBS b_n will reset $i_k[t]$ to be the unoccupied state in time slot t, i.e., $x_{n,i_k[t]}[t] = 1$. Hence, the value of the number of times, i.e., $\psi_{n,i_k[t]}[T]$, that service $i_k[t]$ is requested at SBS b_n in time frame T will be affected. Then, the service-caching of SBS b_n in time frame $(T+1)$ may be affected because service-caching reward $r_n^{\rm c}$ given in Eq. (30) is related to $\psi_{n,i_k[t]}[T]$. Moreover, since the output layer of DDPG uses the hyperbolic tangent functions as activation functions, each output value of DDPG is within $[-1,1]$, which is then normalized to $[0,1]$. To guarantee constraint C2, the outputs of DDPG for $\theta_{n,k}[t]$'s in $a_n^{\rm oc}[t]$, denoted by $o_{n,k}^{\rm oc,\,s}[t]$'s, are used to calculate $\theta_{n,k}[t]$'s by using $\theta_{n,k}[t] \triangleq o_{n,k}^{\rm oc,\,s}[t] \Big/ \left(\sum_{k'\in\mathcal{U}_n[t]} o_{n,k'}^{\rm oc,\,s}[t]\right)$. To guarantee constraint C10, for each U_k, we obtain $D_{k,n}[t]$ in $a_n^{\rm oc}[t]$ by using $D_{n,k}[t] \triangleq o_{k,n}^{\rm oc}[t] D_k[t]$, where $o_{k,n}^{\rm oc}[t]$ is the output of DDPG for $D_{k,n}[t]$ and it is used as the task offloading proportion for U_k.

C. Fast Timescale: Computation Resource-Allocations

Based on the obtained $D_{k,n}[t]$'s and $y_{k,n,m}[t]$'s, we still utilize DDPG to derive the computation resource-allocations variables, i.e., $f_{n,k}[t]$'s, where each SBS $b_n, \forall n \in \mathcal{B}$, is treated as an agent. We define the state, action, and reward of SBS b_n, $\forall n \in \mathcal{B}$, for computation resource-allocations in time slot t, denoted by $\mathcal{O}_n^{\rm cr}[t]$, $a_n^{\rm cr}[t]$, and $r_n^{\rm cr}[t]$, respectively, as follows:

$$\begin{aligned} \mathcal{O}_n^{\rm cr}[t] \triangleq \big\{&D_{k,n}[t], y_{k,m,n}[t], Z_k[t], i_k[t], c_{n,i}[T], \xi_{k,m,n}^{\rm tr}[t], \\ &\forall m \in \mathcal{B}_{\rm g}, n \in \mathcal{B}(m), k \in \mathcal{L}_n[t], i \in \mathcal{I}\big\}, \end{aligned} \tag{49}$$

$$a_n^{\rm cr}[t] \triangleq \{f_{n,1}[t], \ldots, f_{n,K}[t]\}, \tag{50}$$

and

$$\begin{aligned} r_n^{\rm cr}[t] \triangleq &-\sum_{m\in(\mathcal{B}_{\rm g}\cap\mathcal{B}(n))}\ \sum_{k\in\mathcal{U}_m[t]} E_{k,m,n}^{\rm pr}[t] \\ &+ \sum_{m\in(\mathcal{B}_{\rm g}\cap\mathcal{B}(n))}\ \sum_{k\in\mathcal{L}_n[t]} \Upsilon_{k,m,n}^{\rm ti}[t] + \Upsilon_n^{\rm en}[t], \end{aligned} \tag{51}$$

where $\xi_{k,m,n}^{\rm tr}[t] \triangleq \xi_{k,m}^{\rm tr}[t] + \xi_{m,n}^{\rm tr}[t]$ is the time consumption for offloading $D_{k,n}[t]$ bits of user U_k's task data to SBS b_n, and $\mathcal{L}_n[t]$ is the set of users for which SBS $b_n, \forall n \in \mathcal{B}$, needs to provide computing services in time slot t. Moreover, by defining $r_n^{\rm cr}[t]$'s, we can minimize off-grid SBSs' energy consumptions while guaranteeing the related constraints C6 and C8 of all SBSs. Accordingly, by defining $r_n^{\rm oc}[t]$ in Eq. (41), we can minimize the energy consumptions of all users and off-grid SBSs while ensuring that constraints C5-C8 can be satisfied.

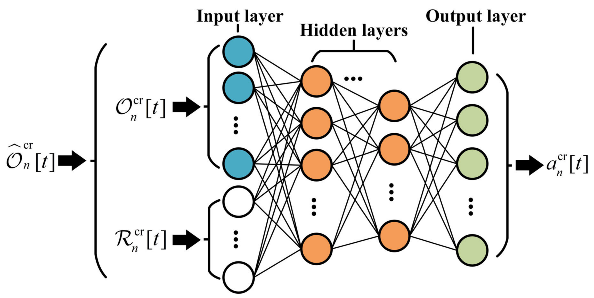


Fig. 3. Structure of the actor network in DDPG for computation resource-allocations, where $\left|\widehat{\mathcal{O}}_n^{\rm cr}[t]\right| = 5\,|\Omega| + |\mathcal{I}|$, $|\mathcal{O}_n^{\rm cr}[t]| = 5\,|\mathcal{L}_n[t]| + |\mathcal{I}|$, and $|\mathcal{R}_k^{\rm cr}[t]| = 5\,(|\Omega| - |\mathcal{L}_n[t]|)$.

However, since the set $\mathcal{L}_n[t]$ may dynamically change as time goes on, the dimension of $\mathcal{O}_n^{\rm cr}[t]$ given in Eq. (49) may take different values for different time slots. In reinforcement learning, DNNs are trained by iteratively updating network parameters. Since the change of the dimension of $\mathcal{O}_n^{\rm cr}[t]$ may lead to the change of the number of DNN input layer neurons and the dimension of DNN parameter set, the network structure of DNN may change, which in fact leads to the generation of another DNN. To keep the DNN structure unchanged, we generate DNNs with the possible maximum number of input layer neurons. Since one element in state $\mathcal{O}_n^{\rm cr}[t]$ corresponds to one neuron in the DNN's input layer, the maximum number of DNN's input layer neurons is $\max\{|\mathcal{O}_n^{\rm cr}[t]|\} = \max\{5\,|\mathcal{L}_n[t]|\} + |\mathcal{I}| = 5\,|\Omega| + |\mathcal{I}|$. Then, for computation resource-allocations, we generate DNNs which are shown in Fig. 3, where we re-define SBS b_n's state for computation resource-allocations in time slot t, denoted by $\widehat{\mathcal{O}}_n^{\rm cr}[t]$, as:

$$\widehat{\mathcal{O}}_n^{\rm cr}[t] \triangleq \{\mathcal{O}_n^{\rm cr}[t], \mathcal{R}_n^{\rm cr}[t]\}, \tag{52}$$

where $\left|\widehat{\mathcal{O}}_n^{\rm cr}[t]\right| = 5\,|\Omega| + |\mathcal{I}|$ and $|\mathcal{R}_n^{\rm cr}[t]| = \left|\widehat{\mathcal{O}}_n^{\rm cr}[t]\right| - |\mathcal{O}_n^{\rm cr}[t]| = 5\,(|\Omega| - |\mathcal{L}_n[t]|)$. Here, $\mathcal{R}_n^{\rm cr}[t]$ is used to guarantee that the number of elements in $\widehat{\mathcal{O}}_n^{\rm cr}[t]$ is $5 \times |\Omega| + |\mathcal{I}|$, and the values of elements in $\mathcal{R}_n^{\rm cr}[t]$ are set as 0's so that they do not affect the outputs of DNNs [30].

In addition, similar to $\theta_{n,k}[t]$'s and $D_{k,n}[t]$'s in Section III-B, we can obtain $f_{n,k}[t]$'s based on the outputs of DDPG, denoted by $o_{n,k}^{\rm cr}[t]$'s, by using the following equation:

$$f_{n,k}[t] \triangleq \begin{cases} F_n^{\max} o_{n,k}^{\rm cr}[t], & \text{if } \sum\limits_{k' \in \mathcal{L}_n[t]} o_{n,k'}^{\rm cr}[t] \le 1, \\ \dfrac{F_n^{\max} o_{n,k}^{\rm cr}[t]}{\sum\limits_{k' \in \mathcal{L}_n[t]} o_{n,k'}^{\rm cr}[t]}, & \text{if } \sum\limits_{k' \in \mathcal{L}_n[t]} o_{n,k'}^{\rm cr}[t] > 1, \end{cases} \tag{53}$$

where the output value $o_{n,k}^{\rm cr}[t]$ of DDPG is normalized to $[0, 1]$. Specifically, when $\sum_{k' \in \mathcal{L}_n[t]} o_{n,k}^{\rm cr}[t] \le 1$, to reduce energy consumption of SBS b_n, we let $f_{n,k}[t] \triangleq F_n^{\max} o_{n,k}^{\rm cr}[t]$. On the contrary, when $\sum_{k' \in \mathcal{L}_n[t]} o_{n,k'}^{\rm cr}[t] > 1$, to satisfy constraint C9, we let

$$f_{n,k}[t] \triangleq \frac{F_n^{\max} o_{n,k}^{\rm cr}[t]}{\sum\limits_{k' \in \mathcal{L}_n[t]} o_{n,k'}^{\rm cr}[t]}. \tag{54}$$

Algorithm 1 HMDRL-Based Algorithm for Solving the Optimization Problem in Eq. (21) for Scenarios With SUs

1: **Initialize**: The network parameter vectors and reply buffers of all DDPGs and Dueling DQNs.
2: **For** each episode $= 1, 2, \ldots,$ **do**
3: Reset the environment.
4: **For** each frame $T = 1, 2, \ldots,$ **do**
5: Each SBS b_n chooses action $a_n^{\rm c}$ based on state $\mathcal{O}_n^{\rm c}[T]$, where on-grid SBSs choose actions before off-grid SBSs.
6: **For** each time slot $t = 1, 2, \ldots,$ **do**
7: **For** each on-grid SBS $b_n, n = 1, 2, \ldots,$ **do**
8: Choose actions $a_n^{\rm oc}[t]$ and $a_n^{\rm od}[t]$ based on states $\mathcal{O}_n^{\rm oc}[t]$ and $\mathcal{O}_n^{\rm od}[t]$, respectively.
9: Get reward $r_n^{\rm od}[t]$ and next state $\mathcal{O}_n^{\rm od}[t+1]$.
10: Store transition $\left(\mathcal{O}_n^{\rm od}[t], a_n^{\rm od}[t], r_n^{\rm od}[t], \mathcal{O}_n^{\rm od}[t{+}1]\right)$ in Dueling DQN's replay buffer and sample a mini-batch of transitions from this buffer.
11: Update the Q network by minimizing the loss function $L\left(\boldsymbol{\omega}_n^{\rm od}, \boldsymbol{\omega}_{{\rm sv},n}^{\rm od}, \boldsymbol{\omega}_{{\rm av},n}^{\rm od}\right)$ given by Eq. (45).
12: Update the target Q network every $G^{\rm od}$ time slots.
13: **End for**
14: **For** SBS $b_n, \forall n \in \mathcal{B}$, **do**
15: Choose action $a_n^{\rm cr}[t]$ based on state $\widehat{\mathcal{O}}_n^{\rm cr}[t]$, and get reward $r_n^{\rm cr}[t]$ and next state $\widehat{\mathcal{O}}_n^{\rm cr}[t+1]$.
16: Store transition $\left(\widehat{\mathcal{O}}_n^{\rm cr}[t], a_n^{\rm cr}[t], r_n^{\rm cr}[t], \widehat{\mathcal{O}}_n^{\rm cr}[t{+}1]\right)$ in DDPG's replay buffer for computation resource-allocations and sample a mini-batch of transitions from this buffer.
17: Update the actor network by using Eq. (34), and update the critic network by minimizing the loss function given by Eq. (32). Also, update the target networks by using Eqs. (35) and (36).
18: **End for**
19: **For** SBS $b_n, \forall n \in \mathcal{B}_{\rm g}$, **do**
20: Get reward $r_n^{\rm oc}[t]$ and next state $\mathcal{O}_n^{\rm oc}[t+1]$.
21: Update DDPG for computation-offloading by using methods similar to lines $16-17$.
22: **End for**
23: **End for**
24: **For** SBS $b_n, \forall n \in \mathcal{B}$, **do**
25: Get reward $r_n^{\rm c}[T]$ and next state $\mathcal{O}_n^{\rm c}[T+1]$.
26: Update DDPG for service-caching by using methods similar to lines $16-17$.
27: **End for**
28: **End for**
29: **End for**

For scenarios with SUs, we summarize the HMDRL-based algorithm to solve the optimization problem specified by Eq. (21) in **Algorithm 1**. Notice that in line 7 of **Algorithm 1**, for on-grid SBSs b_n and b_m, if $n < m$, then SBS b_n performs computation-offloading before SBS b_m.

D. Computational Complexity of *Algorithm 1*

When performing service-caching, for SBS b_n, $\forall n \in \mathcal{B}$, let $H_{n,l}^{\rm an}$ and $H_{n,l}^{\rm cn}$ be the numbers of neurons in the l-th hidden layer of DDPGs' actor network and critic network, respectively, and $L_n^{\rm an}$ and $L_n^{\rm cn}$ be the numbers of hidden layers in DDPGs' actor network and critic network, respectively. Therefore, in the training process, the complexity in cooperative service-caching is

$$O\left(\sum_{n\in\mathcal{B}} NY\Psi\left\{\left|\mathcal{O}_n^{\mathrm{c}}\right| H_{n,1}^{\mathrm{an}}+\sum_{l=2}^{L_n^{\mathrm{an}}} H_{n,l-1}^{\mathrm{an}} H_{n,l}^{\mathrm{an}}+H_{n,L_n^{\mathrm{an}}}^{\mathrm{an}}\left|a_n^{\mathrm{c}}\right|\right.\right.$$
$$\left.\left.+H_{n,1}^{\mathrm{cn}}\left[\left|\mathcal{O}_n^{\mathrm{c}}\right|+\left|a_n^{\mathrm{c}}\right|\right]+\sum_{l=2}^{L_n^{\mathrm{cn}}} H_{n,l-1}^{\mathrm{cn}} H_{n,l}^{\mathrm{cn}}+H_{n,L_n^{\mathrm{cn}}}^{\mathrm{cn}}\right\}\right), \tag{55}$$

where $|\mathcal{O}_n^{\mathrm{c}}|$ and $|a_n^{\mathrm{c}}|$ are the cardinality of SBS b_n's state $\mathcal{O}_n^{\mathrm{c}}$ and action a_n^{c}, respectively, N is the number of frames in each episode, Y is the number of episodes, and Ψ is the mini-batch sampling size given in Eq. (32). Similarly, we can analyze the computational complexities for computation-offloading and computation resource-allocations.

IV. Jointly Optimizing Cooperative Service-Caching, Computation-Offloading, and Resource-Allocations for Scenarios With MUs

We extend the work in Section III to more realistic scenarios with MUs, where each MU moves with a certain trajectory at a low speed within the considered area. When taking into account user mobility, since the set $\mathcal{U}_n[t]$ may dynamically change, the dimensions of actions $a_n^{\mathrm{oc}}[t]$ and $a_n^{\mathrm{od}}[t]$ and states $\mathcal{O}_n^{\mathrm{oc}}[t]$ and $\mathcal{O}_n^{\mathrm{od}}[t]$ in computation-offloading may dynamically change as time slot t changes. Hence, for computation-offloading, the network structures of DNNs built in Section III-B may dynamically change.

Thus, similar to Section III-C, we generate DNNs with the number of input layer neurons being $\max\{|\mathcal{O}_n^{\mathrm{oc}}[t]|\} = \max\{4\,|\mathcal{U}_n[t]|\} + |\mathcal{B}(n)\cap\mathcal{B}_{\mathrm{e}}| = 4\,|\Omega| + |\mathcal{B}(n)\cap\mathcal{B}_{\mathrm{e}}|$ and the number of output layer neurons being $\max\{|a_n^{\mathrm{oc}}[t]|\} = \max\{2\,|\mathcal{U}_n[t]|\} = 2\,|\Omega|$ to obtain $D_{k,n}[t]$'s and $\theta_{n,k}[t]$'s in $a_n^{\mathrm{oc}}[t]$. Similarly, we generate DNNs with the number of input layer neurons being $\max\left\{\left|\mathcal{O}_n^{\mathrm{od}}[t]\right|\right\} = \max\{|\mathcal{U}_n[t]|\} + |\mathcal{B}(n)\cap\mathcal{B}_{\mathrm{e}}| + 2\,|\mathcal{B}(n)|\,|\mathcal{I}| = |\Omega| + |\mathcal{B}(n)\cap\mathcal{B}_{\mathrm{e}}| + 2\,|\mathcal{B}(n)|\,|\mathcal{I}|$ and the number of output layer neurons being $\max\left\{\left|a_n^{\mathrm{od}}[t]\right|\right\} = \max\{|\mathcal{U}_n[t]|\} = |\Omega|$ to obtain $y_{k,n,m}[t]$'s in $a_n^{\mathrm{od}}[t]$. Accordingly, for SBS $b_n, \forall n\in\mathcal{B}_{\mathrm{g}}$, we define the sets of input states, denoted by $\widehat{\mathcal{O}}_n^{\mathrm{oc}}[t]$ and $\widehat{\mathcal{O}}_n^{\mathrm{od}}[t]$, respectively, and output actions, denoted by $\widehat{a}_n^{\mathrm{oc}}[t]$ and $\widehat{a}_n^{\mathrm{od}}[t]$, respectively, for scenarios with MUs in time slot t as:

$$\begin{cases} \widehat{\mathcal{O}}_n^{\mathrm{oc}}[t] \triangleq \left\{\mathcal{O}_n^{\mathrm{oc}}[t], \mathcal{R}_n^{\mathrm{sc}}[t]\right\}, & (56)\\ \widehat{\mathcal{O}}_n^{\mathrm{od}}[t] \triangleq \left\{\mathcal{O}_n^{\mathrm{od}}[t], \mathcal{R}_n^{\mathrm{sd}}[t]\right\}, & (57)\end{cases}$$

and

$$\begin{cases} \widehat{a}_n^{\mathrm{oc}}[t] \triangleq \left\{a_n^{\mathrm{oc}}[t], \mathcal{R}_n^{\mathrm{ac}}[t]\right\}, & (58)\\ \widehat{a}_n^{\mathrm{od}}[t] \triangleq \left\{a_n^{\mathrm{od}}[t], \mathcal{R}_n^{\mathrm{ad}}[t]\right\}, & (59)\end{cases}$$

where

$$\begin{cases} \left|\widehat{\mathcal{O}}_n^{\mathrm{oc}}[t]\right| = 4\,|\Omega| + |\mathcal{B}(n)\cap\mathcal{B}_{\mathrm{e}}|, & (60)\\ \left|\widehat{\mathcal{O}}_n^{\mathrm{od}}[t]\right| = |\Omega| + |\mathcal{B}(n)\cap\mathcal{B}_{\mathrm{e}}| + 2\,|\mathcal{B}(n)|\,|\mathcal{I}|, & (61)\\ \left|\widehat{a}_n^{\mathrm{oc}}[t]\right| = 2\,|\Omega|, & (62)\\ \left|\widehat{a}_n^{\mathrm{od}}[t]\right|\,|\Omega|. & (63)\end{cases}$$

Similar to $\mathcal{R}_n^{\mathrm{cr}}[t]$, the sets $\mathcal{R}_n^{\mathrm{sc}}[t]$, $\mathcal{R}_n^{\mathrm{sd}}[t]$, $\mathcal{R}_n^{\mathrm{ac}}[t]$, and $\mathcal{R}_n^{\mathrm{ad}}[t]$ are used to keep $\left|\widehat{\mathcal{O}}_n^{\mathrm{oc}}[t]\right|$, $\left|\widehat{\mathcal{O}}_n^{\mathrm{od}}[t]\right|$, $|\widehat{a}_n^{\mathrm{oc}}[t]|$, and $\left|\widehat{a}_n^{\mathrm{od}}[t]\right|$ as constants, respectively. Moreover, we set the values of elements in $\mathcal{R}_n^{\mathrm{sc}}[t]$ and $\mathcal{R}_n^{\mathrm{sd}}[t]$ as 0's, and the elements in $\mathcal{R}_n^{\mathrm{ac}}[t]$ and $\mathcal{R}_n^{\mathrm{ad}}[t]$ are not utilized to calculate $D_{k,n}[t]$'s and $\theta_{n,k}[t]$'s in $a_n^{\mathrm{oc}}[t]$ and $y_{k,n,m}[t]$'s in $a_n^{\mathrm{od}}[t]$.

Moreover, unlike Section III-B, we derive discrete variables $y_{k,n,m}[t]$'s by using DDPG instead of Dueling DQN. This is because if Dueling DQN is leveraged, the dimensionality of the action space grows exponentially with the number of users. Similar to Section III-B, the output values of DDPG are normalized to $[0,1]$. At SBS b_n, to obtain $y_{k,n,m}[t]$'s for user U_k, $\forall k\in\mathcal{U}_n[t]$, the interval $[0,1)$ is evenly divided into $|\mathcal{B}(n)|+1$ intervals, with each corresponding to one choice, i.e., local computing at user U_k or data offloading to one SBS $b_m, \forall m\in\mathcal{B}(n)$. Specifically, user U_k or each of SBS $b_m, \forall m\in\mathcal{B}(n)$, is assigned an index number. If $\left\lfloor o_{k,n,m}^{\mathrm{od}}\times(|\mathcal{B}(n)|+1)\right\rfloor$ is equal to the index number of one SBS $b_m, \forall m\in\mathcal{B}(n)$, we set $y_{k,n,m}[t]=1$; otherwise we set $\sum_{m\in\mathcal{B}(n)} y_{k,n,m}[t]=0$ and user U_k processes task by itself, where $\lfloor\cdot\rfloor$ denotes the floor function.

In addition, since the dimensions of each SBS b_n's state $\mathcal{O}_n^{\mathrm{c}}[T]$ and action $a_n^{\mathrm{c}}[T]$ are not affected by the mobility of users, the cooperative service-caching scheme among SBSs for scenarios with MUs is the same as that for scenarios with SUs. While for computation resource-allocations of SBSs, since the dimension of $\mathcal{L}_n[t]$ always dynamically changes whether or not user mobility is taken into account, the computation resource-allocations scheme proposed in Section III can still be used for scenarios with MUs. Then, for scenarios with MUs, we can develop an HMDRL based algorithm as shown in **Algorithm 2** to solve the problem given in Eq. (21).

V. Performances Evaluations

SBSs and users are distributed in a 50 m $\times$ 50 m area. Unless otherwise stated, for each user U_k, we take $f_k = 10^9$ Hz and the penalty parameters $\varphi_k^{\mathrm{ti}} = \varphi_k^{\mathrm{en}} = -0.02$ in Eqs. (23) and (25), respectively. Besides, we take $F_n^{\max} = 10^{10}$ Hz for SBS b_n, $\forall n\in\mathcal{B}$, and the penalty parameter $\varphi_n^{\mathrm{en}} = -0.02$ in Eq. (27) for off-grid SBS b_n, $\forall n\in\mathcal{B}_{\mathrm{e}}$. Moreover, for SBSs b_n and b_m, $\forall n\in\mathcal{B}_{\mathrm{g}}$, $m\in\mathcal{B}(n)$, and user $U_k, \forall k\in\mathcal{U}_n[t]$, we set the transmit power from user U_k to SBS b_n as $P_{k,n}[t] = 0.05$ W and the penalty parameter $\varphi_{k,n,m}^{\mathrm{ti}} = -0.02$ in Eq. (24). Also, in large-scale fading $\overline{h}_{k,n}[t]$, we take the antenna gain $A_{\mathrm{d}} = 4.11$, the carrier frequency $f_{\mathrm{c}} = 915$ MHz, and the path loss exponent $d_{\mathrm{e}} = 2.8$. We take the noise power $\sigma^2 = 10^{-9}$ W, the effective switched capacitance $\nu = 10^{-28}$, and the weight parameter $\zeta = 0.9$ in Eq. (21), and set the spatial densities of on-grid SBSs, off-grid SBSs, and users as $\lambda_{\mathrm{g}} = 0.0016/\mathrm{m}^2$, $\lambda_{\mathrm{e}} = 0.0048/\mathrm{m}^2$, and $\rho = 0.0048/\mathrm{m}^2$, respectively. In addition, for all users, the data input size and the number of CPU circles required per bit follow uniform distribution with $D_k[t]\in[1\times10^5, 2\times10^5]$ bits and $Z_k[t]\in[7.5\times10^2, 10^3]$ cycles/bit, respectively. Furthermore, we take the average total reward as the average value of total rewards $r[T]$'s defined in Eq. (22) over 20 time

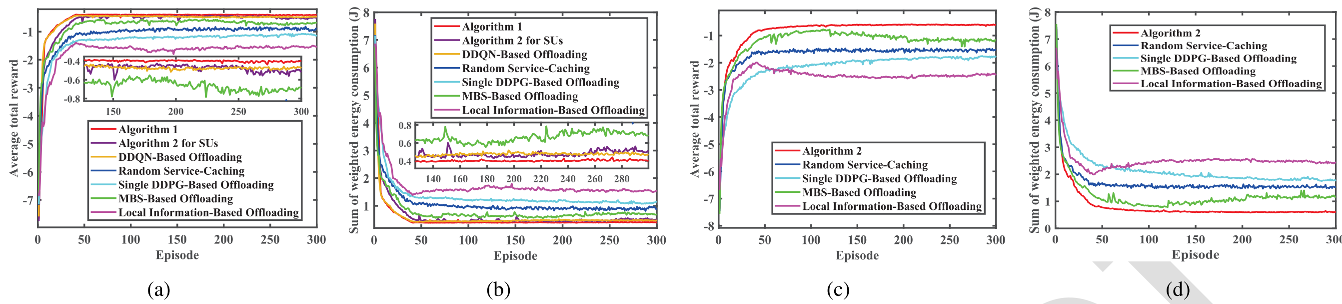


Fig. 4. Convergence performance comparisons: (a) Average total reward for scenarios with SUs; (b) Sum of weighted energy consumption for scenarios with SUs; (c) Average total reward for scenarios with MUs; (d) Sum of weighted energy consumption for scenarios with MUs.

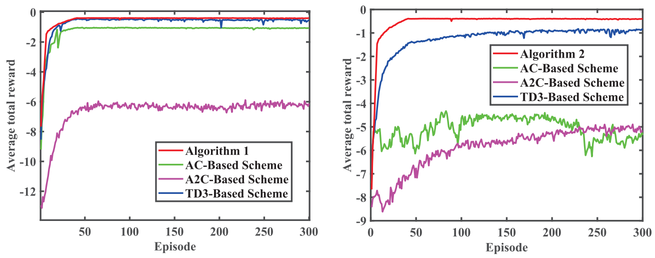
Algorithm 2 HMDRL-Based Algorithm for Solving the Optimization Problem in Eq. (21) for Scenarios With MUs

Initialize: The network parameter vectors and reply buffers of all DDPGs.
For each episode $= 1, 2, \ldots,$ **do**
  Reset the environment.
  For each frame $T = 1, 2, \ldots,$ **do**
    SBS $b_n, \forall n \in \mathcal{B}$, chooses action $a_n^{\mathrm{c}}[T]$ based on state $\mathcal{O}_n^{\mathrm{c}}[T]$ similar to **Algorithm 1**.
    For each time slot $t = 1, 2, \ldots,$ **do**
      For each on-grid SBS $b_n, n = 1, 2, \ldots,$ **do**
        Choose actions $\widehat{a}_n^{\mathrm{oc}}[t]$ and $\widehat{a}_n^{\mathrm{od}}[t]$ based on states $\widehat{\mathcal{O}}_n^{\mathrm{oc}}[t]$ and $\widehat{\mathcal{O}}_n^{\mathrm{od}}[t]$, respectively.
        Get reward $r_n^{\mathrm{od}}[t]$ and next state $\widehat{\mathcal{O}}_n^{\mathrm{od}}[t+1]$.
        Update DDPG for obtaining $\widehat{a}_n^{\mathrm{od}}[t]$ in computation-offloading similar to lines $16-17$ of **Algorithm 1**.
      End for
      For SBS $b_n, \forall n \in \mathcal{B}$, **do**
        Choose action $a_n^{\mathrm{cr}}[t]$ based on state $\widehat{\mathcal{O}}_n^{\mathrm{cr}}[t]$, and get reward $r_n^{\mathrm{cr}}[t]$ and next state $\widehat{\mathcal{O}}_n^{\mathrm{cr}}[t+1]$.
        Update DDPG for computation resource-allocations similar to lines $16-17$ of **Algorithm 1**.
      End for
      For SBS $b_n, \forall \in \mathcal{B}_{\mathrm{g}}$, **do**
        Get reward $r_n^{\mathrm{oc}}[t]$ and next state $\widehat{\mathcal{O}}_n^{\mathrm{oc}}[t+1]$.
        Update DDPG for obtaining $\widehat{a}_n^{\mathrm{oc}}[t]$ in computation-offloading similar to lines $16-17$ of **Algorithm 1**.
      End for
    End for
    For SBS $b_n, \forall n \in \mathcal{B}$, **do**
      Get reward $r_n^{\mathrm{c}}[T]$ and next state $\mathcal{O}_n^{\mathrm{c}}[T+1]$.
      Update DDPG for service-caching by using methods similar to lines $16-17$ of **Algorithm 1**.
    End for
  End for
End for

frames (one episode), where each frame consists of $\varpi = 50$ time slots. To evaluate the performances of our proposed schemes **Algorithms 1-2**, we also consider the following baseline schemes:

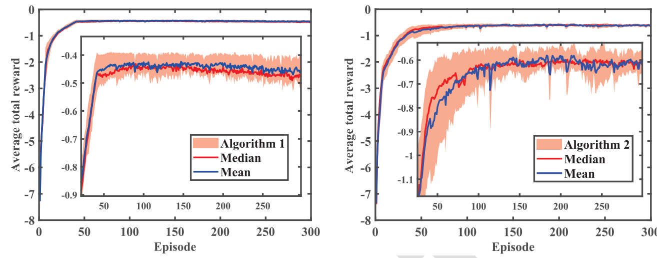
- *Random Service-Caching*: This scheme leverages random service-caching policies.
- *Single DDPG-Based Offloading*: In computation-offloading, each on-grid SBS b_n uses a single DDPG to decide the discrete variables $y_{k,n,m}[t]$'s and the continuous variables $\theta_{n,k}[t]$'s and $D_{k,n}[t]$'s simultaneously.
- *MBS-Based Offloading*: MBS acts as an agent which collects information from all users and SBSs to simultaneously decide $y_{k,n,m}[t]$'s, $\theta_{n,k}[t]$'s, and $D_{k,n}[t]$'s for all users and SBSs [31]. But, MBS does not provide computing services to users.
- *Local Information-Based Offloading*: Each on-grid SBS b_n also uses the states and actions of on-grid SBSs in the set $(\mathcal{B}(n) \setminus \{n\})$ to train its own network parameters, e.g., ω_n^{od}, similar to [32].
- *DDQN-Based Offloading*: This scheme uses Double Deep Q Network (DDQN) instead of Dueling DQN to decide discrete variables $y_{k,n,m}[t]$'s.

We first compare the convergence performances of **Algorithms 1-2** and the above-mentioned schemes in Fig. 4, where each episode consists of multiple frames. Analyzing Fig. 4, we can observe that **Algorithm 1** reaches convergence within about 40 episodes, while **Algorithm 2** can reach convergence within about 50 episodes. But, the average total reward and sum of weighted energy consumption for each of the above-mentioned baseline schemes first converge and then oscillate over relatively wide ranges. Fig. 4 also shows that the average total reward of our proposed schemes **Algorithms 1-2** are larger than those of the baseline schemes, while the sums of weighted energy consumptions of **Algorithms 1-2** are lower than those of the baseline schemes. In particular, compared with Random Service-Caching, it is necessary to decide service-caching for all SBSs based on cooperative service-caching. Besides, in MBS-Based Offloading, it is unreasonable to let MBS function as an agent to collect state information from all users and SBSs and make computation-offloading decisions for them. This is because it is challenging for MBS to extract featuring information about each user or SBS from too much state information. Instead, each on-grid SBS b_n should act as an agent to collect its own related information and decide the computation-offloading policies for itself and related users. Moreover, in Local Information-Based Offloading, each on-grid SBS b_n does not need to take into account too much states and actions information about on-grid SBSs in the set $(\mathcal{B}(n) \setminus \{n\})$. The reason for this is that when training neural network parameters to



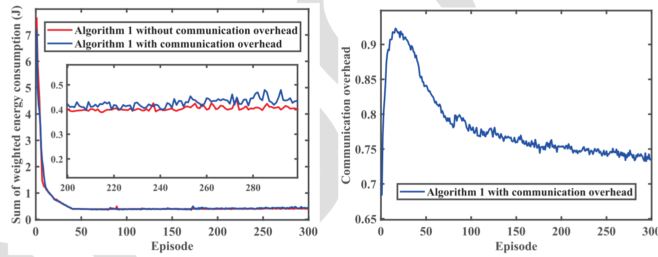
(a) (b)

Fig. 5. Convergence performance comparisons of **Algorithms 1-2** with AC-Based Scheme, A2C-Based Scheme, and TD3-Based Scheme: (a) Average total reward for scenarios with SUs; (b) Average total reward for scenarios with MUs.



(a) (b)

Fig. 6. Convergence range of average total reward: (a) Average total reward of **Algorithm 1**; (b) Average total reward of **Algorithm 2**.



(a) (b)

Fig. 7. Performances of **Algorithm 1** with and without communication overhead considerations: (a) Sum of weighted energy consumption; (b) Communication overhead.

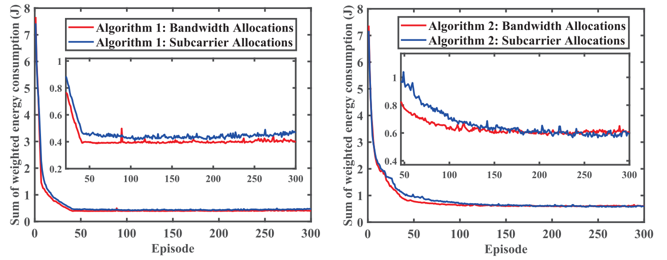
make computation-offloading decisions, it is also difficult for SBS b_n to extract its relevant featuring information from too much collected information. Besides, compared with Single DDPG-Based Offloading, it is necessary to decide the discrete variables $y_{k,n,m}[t]$'s and the continuous variables $\theta_{n,k}[t]$'s and $D_{k,n}[t]$'s separately in computation-offloading and evaluate their values based on different reward functions. In addition, compared with DDQN-Based Offloading and **Algorithm 2** for SUs, we can know that Dueling DQN can find more suitable $y_{k,n,m}[t]$'s than DDPG and DDQN.

Figure 5 compares the convergence performances of **Algorithms 1-2** with some other mainstream algorithms, i.e., *AC-Based Scheme*, *A2C-Based Scheme*, and *TD3-Based Scheme*, which use Actor-Critic (AC), Advantage Actor Critic (A2C), and Twin Delayed Deep Deterministic policy gradient (TD3), respectively, instead of DDPG to decide service-caching variables $c_{n,i}[T]$'s, offloading variables $D_{k,n}[t]$'s and $\theta_{n,k}[t]$'s, and resource-allocation variables $f_{n,k}[t]$'s. The results observed in Fig. 5 show that **Algorithms 1-2** can achieve the best convergence performances for scenarios with SUs and MUs, respectively. This is due to the fact that AC-Based Scheme and A2C-Based Scheme do not incorporate the techniques of experience replay and target networks when training neural networks [33] [34]. Moreover, TD3 is more suitable for dealing with complex optimization problems with high-dimensional state and action spaces, because of the utilization of a double Q-network and delayed updates which avoid the overfitting to the current policy [35]. In this paper, we simplify the considered complex optimization problem by first decomposing it into service-caching sub-problems, offloading sub-problems, and resource-allocation sub-problems, and then utilizing multiple agents to find solutions to these sub-problems through the cooperations. In this case, the relatively simpler method of DDPG is easier and more efficient to converge and find more effective solutions. Fig. 6 shows the convergence range of average total reward caused by **Algorithms 1-2**. Similar to [36], the results are obtained by running **Algorithms 1-2** with 6 different random seeds which determine system channel fading gains, each user's data size and required services, etc., in each time slot. The *Mean* and *Median* are the average and middle values of the results over the 6 random seeds, respectively. Besides, the shaded area shows the range between the maximum and minimum values of the results over all 6 random seeds. From Fig. 6, we can observe that the average total reward functions of **Algorithms 1-2** oscillate within the constrained range while still capturing their random characteristics.

Figure 7 shows the performances of **Algorithm 1** with and without communication overhead considerations, where the communication overhead occurs when offloading data from one SBS to another [37]. When considering communication overhead, the optimization objective of the problem formulated in Eq. (21) becomes:

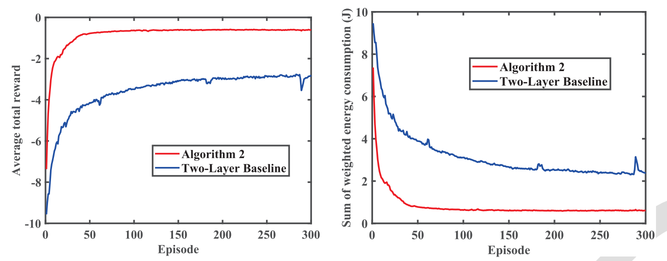
$$\begin{aligned}\min_{\Theta,\mathcal{C},\mathcal{Y},\mathcal{F},\mathcal{D}} \Bigg\{ \sum_{t=1}^{\varpi} \Bigg(\varepsilon \Bigg(\zeta \Bigg[\sum_{k\in\Omega} E_k^{\mathrm{u}}[t] + \sum_{n\in\mathcal{B}_{\mathrm{g}}} \sum_{k\in\mathcal{U}_n[t]} E_{k,n}^{\mathrm{tr}}[t] \Bigg] \\ + (1-\zeta) \Bigg[\sum_{n\in\mathcal{B}_{\mathrm{g}}} \sum_{k\in\mathcal{U}_n[t]} \sum_{m\in(\mathcal{B}(n)\cap\mathcal{B}_{\mathrm{e}})} E_{k,n,m}^{\mathrm{pr}}[t] \Bigg] \Bigg) \Bigg) \\ + (1-\varepsilon) \Bigg(\sum_{n\in\mathcal{B}_{\mathrm{g}}} \sum_{k\in\mathcal{U}_n[t]} \sum_{m\in(\mathcal{B}(n)\cap\mathcal{B}_{\mathrm{e}})} y_{k,n,m}[t]\vartheta \Bigg) \Bigg\} \quad (64)\end{aligned}$$

where ϑ is the communication overhead when SBS $b_n, \forall n \in \mathcal{B}_{\mathrm{g}}$, offloads user U_k's, $\forall k \in \mathcal{U}_n[t]$, task data to SBS $b_m, \forall m \in (\mathcal{B}(n) \cap \mathcal{B}_{\mathrm{e}})$, and ε is a weight parameter that balances the sum of weighted energy consumption and the communication overhead. From Fig. 7(a), we can observe that the sums of weighted energy consumptions are almost the same whether or not we consider the communication overhead. This is because to reduce energy consumption for off-grid SBSs and communication overhead among SBSs, each on-grid SBS will try to process the offloading tasks of users by itself as much as possible. Just as shown in Fig. 7(b), the communication overhead becomes smaller and smaller as the training episode increases. Also, to reduce interference among users accessing different SBSs, we consider bandwidth allocations specified by constraint C2 instead of subcarrier allocations developed in [38] for users in each SBS. From Fig. 8, we observe that



(a) (b)

Fig. 8. Sum of weighted energy consumption of **Algorithms 1-2** for bandwidth allocations and subcarrier allocations: (a) Sum of weighted energy consumption of **Algorithm 1**; (b) Sum of weighted energy consumption of **Algorithm 2**.

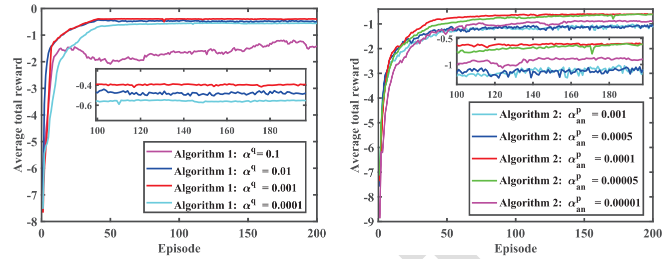


(a) (b)

Fig. 9. Convergence performance comparisons between **Algorithm 2** and Two-Layer Baseline: (a) Average total reward; (b) Sum of weighted energy consumption.

Algorithms 1-2 can always achieve satisfactory convergence performances no matter whether we employ bandwidth allocations or subcarrier allocations. Moreover, the sums of weighted energy consumptions obtained when considering bandwidth allocations are almost the same as those when considering subcarrier allocations. Therefore, by bandwidth allocations, the interference among different SBSs can also be significantly reduced.

In Fig. 9, we compare the convergence performances of **Algorithm 2** and a Two-Layer Baseline, which first decides service-caching in each frame and then decides computation-offloading and computation resource-allocations in each time slot simultaneously. Analyzing Fig. 9, we can observe that **Algorithm 2** performs much better than Two-Layer Baseline. This is because in **Algorithm 2** when performing computation resource-allocations, each SBS has already known the computation-offloading policies of nearby users, while in Two-Layer Baseline SBSs need to decide computation-offloading and resource-allocations simultaneously. Fig. 10(a) shows the average total reward of **Algorithm 1** versus the learning rate α^{q} (see Eq. (48)) of Dueling DQN, and Fig. 10(b) shows the average total reward of **Algorithm 2** versus DDPG's learning rate $\alpha_{\mathrm{an}}^{\mathrm{p}}$ in Eq. (34). Analyzing Fig. 10(a), we can observe that **Algorithm 1** may fail to converge and the average total reward is very small when α^{q} takes a large value, e.g., $\alpha^{\mathrm{q}} = 0.1$. Similarly, when $\alpha_{\mathrm{an}}^{\mathrm{p}}$ takes a relatively large value, e.g., $\alpha_{\mathrm{an}}^{\mathrm{p}} = 0.001$, the average total reward caused by **Algorithm 2** is also very small. On the contrary, when α^{q} or $\alpha_{\mathrm{an}}^{\mathrm{p}}$ takes a small value, e.g., $\alpha^{\mathrm{q}} = 0.0001$ or $\alpha_{\mathrm{an}}^{\mathrm{p}} = 0.00001$, the convergence speed of **Algorithm 1** or **Algorithm 2** becomes relatively low. Hence, we should choose suitable α^{q} and $\alpha_{\mathrm{an}}^{\mathrm{p}}$, e.g., $\alpha^{\mathrm{q}} = 0.001$ and $\alpha_{\mathrm{an}}^{\mathrm{p}} = 0.0001$.



(a) (b)

Fig. 10. Average total reward of **Algorithms 1-2** versus different learning rate parameters: (a) Average total reward of **Algorithm 1** versus Dueling DQN's learning rate α^{q}; (b) Average total reward of **Algorithm 2** versus DDPG's learning rate $\alpha_{\mathrm{an}}^{\mathrm{p}}$.

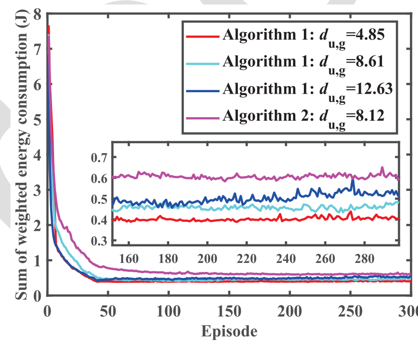


Fig. 11. Sum of weighted energy consumption versus the average distance, e.g., denoted by $d_{\mathrm{u,g}}$, between users and their nearest on-grid SBSs.

Figure 11 shows the sum of weighted energy consumption of **Algorithm 1** versus the average distance, e.g., denoted by $d_{\mathrm{u,g}}$, between users and their nearest on-grid SBSs. It can be seen that the smaller $d_{\mathrm{u,g}}$ is, the lower the sum of weighted energy consumption is. This is because the smaller $d_{\mathrm{u,g}}$ is, the higher the uplink transmission rate $R_{k,n}[t]$ from user U_k to its nearest on-grid SBS b_n is. Consequently, user U_k consumes lower energy for data transmission. Moreover, Fig. 11 also shows the sum of weighted energy consumption of **Algorithm 2** for scenarios with MUs. We can see that the sum of weighted energy consumption of **Algorithm 2** is much higher than that of **Algorithm 1**, even if $d_{\mathrm{u,g}}$ takes a much smaller value in **Algorithm 2**. The reason for this is that during movement the distance between user U_k and its nearest on-grid SBS in time slot t may become very large. Hence, user U_k needs to consume much more energy for data transmission.

Figure 12 plots the sum of weighted energy consumption versus the density of off-grid SBSs, i.e., λ_{e}, and the density of users, i.e., ρ. Analyzing Figs. 12(a) and 12(c), we can observe that the sums of the weighted energy consumptions of **Algorithms 1-2** and the baseline schemes except for Single DDPG-Based Offloading decrease as λ_{e} increases. Moreover, the sums of weighted energy consumptions imposed by **Algorithms 1-2** are always lower than those imposed by all baseline schemes whatever λ_{e} is. In addition, we can see from Figs. 12(b) and 12(d) that the sum of weighted energy consumption increases as ρ increases. When ρ takes small values, there is no significant difference between the sums of weighted energy consumptions caused by **Algorithms 1-2** and those caused by the baseline schemes. However, when ρ

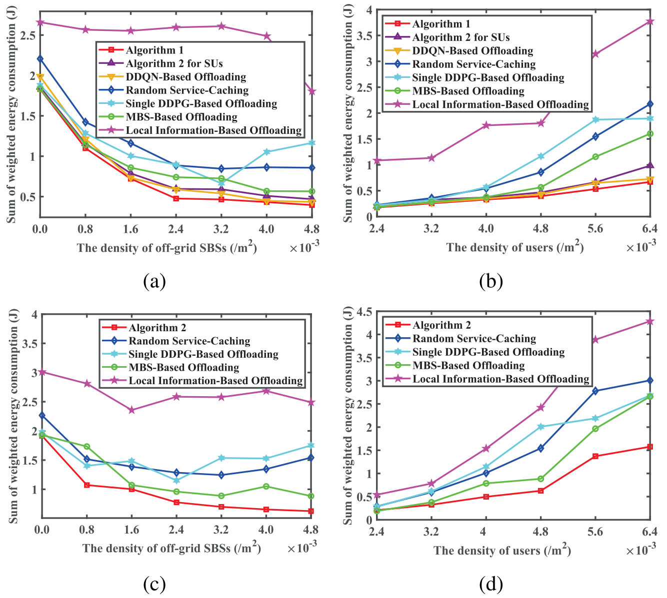


Fig. 12. Sum of weighted energy consumption versus density of off–grid SBSs, i.e., λ_e, and density of users, i.e., ρ: (a) Sum of weighted energy consumption versus λ_e for scenarios with SUs; (b) Sum of weighted energy consumption versus ρ for scenarios with SUs; (c) Sum of weighted energy consumption versus λ_e for scenarios with MUs; (d) Sum of weighted energy consumption versus ρ for scenarios with MUs.

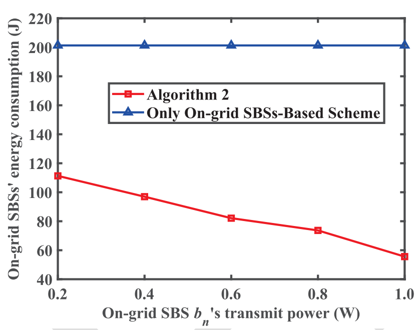


Fig. 13. Task processing energy consumption of on-grid SBSs for scenarios with MUs versus transmit power P_n of on-grid SBS b_n.

takes a relatively large value, the sums of weighted energy consumptions caused by **Algorithms 1-2** are much lower than those imposed by the baseline schemes.

For scenarios with MUs, Fig. 13 plots the task processing energy consumption of on-grid SBSs versus on-grid SBS b_n's transmit power P_n, where in Only On-grid SBSs-Based Scheme all SBSs all powered by electric grid. From Fig. 13, it is evident that the energy consumption of on-grid SBSs imposed by Only On-grid SBSs-Based Scheme is significantly higher than that caused by **Algorithm 2**, where off-grid SBSs, powered by solar and RF-energy, can also help users process tasks. Moreover, Fig. 13 also shows that with the increase of P_n, the task processing energy consumption of on-grid SBSs imposed by **Algorithm 2** decreases. This is because as P_n increases, off-grid SBSs can harvest more RF-energy to help users process tasks. Therefore, off-grid SBSs, powered by solar and RF-energy, can indeed help to significantly reduce the energy consumption of on-grid SBSs.

Also, for scenarios with MUs, Fig. 14 shows the sum of weighted energy consumption versus the weight parameter ζ which balances the energy consumptions of users and

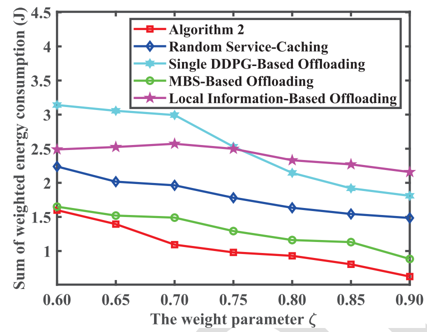


Fig. 14. Sum of weighted energy consumption for scenarios with MUs versus weight parameter ζ.

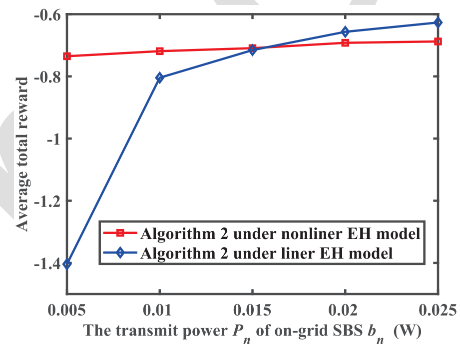


Fig. 15. Average total reward caused by **Algorithm 2** under non-linear EH and linear EH models versus transmit power P_n of on-grid SBS b_n.

off-grid SBSs. It is obvious that the sum of weighted energy consumption decreases as ζ increases. This is because the larger ζ is, the more tasks are offloaded to SBSs for processing. Hence, the energy consumption of users can be significantly reduced, which then reduces the sum of weighted energy consumption of users and off-grid SBSs. For example, when $\zeta = 0.60$, the average task offloading rate is 88.99%, the energy consumptions of users and off-grid SBSs are 0.95 J and 2.58 J, respectively, and the sum of weighted energy consumption is 1.60 J. When $\zeta = 0.90$, the above values become 96.71%, 0.37 J, 2.90 J, and 0.62 J, respectively.

Figure 15 plots the average total reward caused by **Algorithm 2** under the *non-linear* EH and *linear* EH models versus the transmit power P_n of on-grid SBS b_n. For off-grid SBS b_m, we take its battery capacity $E_m^{\max} = 0.05$ J. From Fig. 15, we can see that the average total reward increases with the increase of P_n, since users and off-grid SBSs can harvest much more energy as P_n increases. Moreover, when P_n takes a relatively small value, e.g., $P_n = 0.01$ W, the average total reward obtained under the non-linear EH model is larger than that obtained under the linear EH model. However, when P_n takes a relatively large value, e.g., $P_n = 0.025$ W, the average total reward obtained under the non-linear EH model is lower than that obtained under the linear EH model. This is due to the fact that in the linear EH model the harvested energy of user U_k or off-grid SBS b_m is linearly proportional to the received RF power. In the non-linear EH model, although the harvested energy of user U_k or off-grid SBS b_m also increases as the received RF power increases, it cannot exceed the maximum harvested power M_k of U_k or M_m of b_m. Hence, when P_n increases beyond a certain value, the harvested energy under

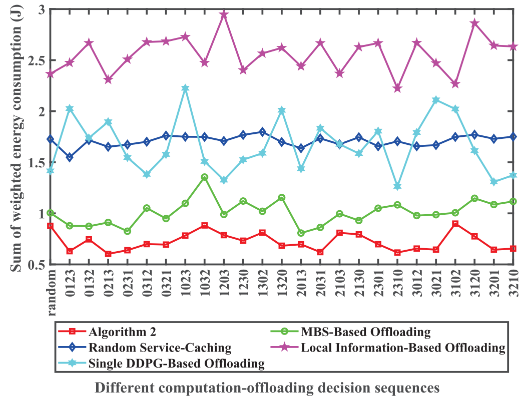


Fig. 16. Sum of weighted energy consumption for scenarios with MUs versus computation-offloading sequence of on-grid SBSs.

the non-linear EH becomes much lower than that obtained under the linear EH.

For scenarios with MUs, Fig. 16 depicts the sum of weighted energy consumption versus the computation-offloading sequence of on-grid SBSs, where the values below the X-axis, e.g., 0123, 3210, are the specified computation-offloading sequences of on-grid SBSs while *random* indicates the random computation-offloading sequence. Analyzing Fig. 16, we can see that although the sum of weighted energy consumption for each scheme fluctuates within a certain range, the fluctuation range of **Algorithm 2** is relatively small. That is, the computation-offloading sequence of on-grid SBSs does not have significant effects on the performances of **Algorithm 2**.

VI. Conclusion

We proposed the cooperative service-caching, computation-offloading, and resource-allocations schemes for EH/MEC-based 6G UDNs, where a large number of EH-based SUs or MUs and a mixture of on-grid SBSs and off-grid SBSs coexist. First, under a non-linear EH model, we developed a two-timescale based joint cooperative service-caching, computation-offloading, and resource-allocations scheme based on HMDRL. Using HMDRL, we derived SBSs' cooperative service-caching policies in each frame, and then derived users' and SBSs' computation-offloading policies and SBSs' computation resource-allocations policies in each time slot. Second, we extended our work to scenarios with MUs. Finally, we validated and evaluated the performances of our proposed schemes through the extensive simulations.

References

[1] Z. Chen, F. Wang, and X. Zhang, "Joint optimization for cooperative service-caching, computation-offloading, and resource-allocations over EH/MEC-based ultra-dense mobile networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Rome, Italy, May 2023, pp. 716–722.

[2] X. Li, Z. Xu, F. Fang, Q. Fan, X. Wang, and V. C. M. Leung, "Task offloading for deep learning empowered automatic speech analysis in mobile edge-cloud computing networks," *IEEE Trans. Cloud Comput.*, vol. 11, no. 2, pp. 1985–1998, Jun. 2023.

[3] Y. Sun, S. Zhou, and J. Xu, "EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 1, pp. 2637–2646, Nov. 2017.

[4] H. Guo, J. Liu, and J. Zhang, "Computation offloading for multi-access mobile edge computing in ultra-dense networks," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 14–19, Aug. 2018.

[5] X. Zhang, J. Tang, H.-H. Chen, S. Ci, and M. Guizani, "Cross-layer-based modeling for quality of service guarantees in mobile wireless networks," *IEEE Commun. Mag.*, vol. 44, no. 1, pp. 100–106, Jan. 2006.

[6] J. Tang and X. Zhang, "Quality-of-service driven power and rate adaptation over wireless links," *IEEE Trans. Wireless Commun.*, vol. 6, no. 8, pp. 3058–3068, Aug. 2007.

[7] H. Su and X. Zhang, "Cross-layer based opportunistic MAC protocols for QoS provisionings over cognitive radio wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 1, pp. 118–129, Jan. 2008.

[8] F. Guo, H. Zhang, H. Ji, X. Li, and V. C. M. Leung, "An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing," *IEEE Trans. Netw.*, vol. 26, no. 6, pp. 2651–2664, Dec. 2018.

[9] H. Zhang, R. Wang, W. Sun, and H. Zhao, "Mobility management for blockchain-based ultra-dense edge computing: A deep reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 20, no. 11, pp. 7346–7359, Nov. 2021.

[10] Y. Mao, Y. Luo, J. Zhang, and K. B. Letaief, "Energy harvesting small cell networks: Feasibility, deployment, and operation," *IEEE Commun. Mag.* vol. 53, no. 6, pp. 94–101, Jun. 2015.

[11] Q. Tang, R. Xie, T. Huang, W. Feng, and Y. Liu, "Dynamic computation offloading with imperfect state information in energy harvesting small cell networks: A partially observable stochastic game," *IEEE Wireless Commun. Lett.*, vol. 9, no. 8, pp. 1300–1304, Aug. 2020.

[12] B. Li, Y. Dai, Z. Dong, E. Panayirci, H. Jiang, and H. Jiang, "Energy-efficient resources allocation with millimeter-wave massive MIMO in ultra dense HetNets by SWIPT and CoMP," *IEEE Trans. Wireless Commun.*, vol. 20, no. 7, pp. 4435–4451, Jul. 2021.

[13] J. Yan, S. Bi, L. Duan, and Y.-J. A. Zhang, "Pricing-driven service caching and task offloading in mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 20, no. 7, pp. 4495–4512, Jul. 2021.

[14] D. Ren, X. Gui, and K. Zhang, "Adaptive request scheduling and service caching for MEC-assisted IoT networks: An online learning approach," *IEEE Internet Things J.*, vol. 9, no. 18, pp. 17372–17386, Sep. 2022.

[15] X. Ma, A. Zhou, S. Zhang, and S. Wang, "Cooperative service caching and workload scheduling in mobile edge computing," in *Proc. Conf. Comput. Commun. (IEEE INFOCOM)*, Toronto, ON, USA, Jul. 2020, pp. 2076–2085.

[16] S. Yu, X. Chen, Z. Zhou, X. Gong, and D. Wu, "When deep reinforcement learning meets federated learning: Intelligent multitimescale resource management for multiaccess edge computing in 5G ultradense network," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2238–2251, Feb. 2021.

[17] M. Chen, S. Guo, K. Liu, X. Liao, and B. Xiao, "Robust computation offloading and resource scheduling in cloudlet-based mobile cloud computing," *IEEE Trans. Mobile Comput.*, vol. 20, no. 5, pp. 2025–2040, May 2021.

[18] X. Jiao et al., "Deep reinforcement learning empowers wireless powered mobile edge computing: Towards energy-aware online offloading," *IEEE Trans. Commun.*, vol. 71, no. 9, pp. 5214–5227, Sep. 2023.

[19] S. Sun et al., "Propagation path loss models for 5G urban micro- and macro-cellular scenarios," in *Proc. IEEE 83rd Veh. Technol. Conf. (VTC)*, Nanjing, China, May 2016, pp. 1–6.

[20] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2581–2593, Nov. 2020.

[21] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint computation offloading and user association in multi-task mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12313–12325, Dec. 2018.

[22] L. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1619–1632, Aug. 2018.

[23] E. Boshkovska, D. W. K. Ng, N. Zlatanov, and R. Schober, "Practical non-linear energy harvesting model and resource allocation for SWIPT systems," *IEEE Commun. Lett.*, vol. 19, no. 12, pp. 2082–2085, Dec. 2015.

[24] J. Zhang, Y. Shen, Y. Wang, X. Zhang, and J. Wang, "Dual-timescale resource allocation for collaborative service caching and computation offloading in IoT systems," *IEEE Trans. Ind. Informat.*, vol. 19, no. 2, pp. 1735–1746, Feb. 2023.

[25] Z. Ding, R. Schober, and H. V. Poor, "No-pain no-gain: DRL assisted optimization in energy-constrained CR-NOMA networks," *IEEE Trans. Commun.*, vol. 69, no. 9, pp. 5917–5932, Sep. 2021.

[26] X. Kong et al., “Deep reinforcement learning-based energy-efficient edge computing for Internet of Vehicles,” *IEEE Trans. Ind. Informat.*, vol. 18, no. 9, pp. 6308–6316, Sep. 2022.

[27] L. Yue, R. Yang, J. Zuo, Y. Zhang, Q. Li, and Y. Zhang, “Unmanned aerial vehicle swarm cooperative decision-making for SEAD mission: A hierarchical multiagent reinforcement learning approach,” *IEEE Access*, vol. 10, pp. 92177–92191, 2022.

[28] T. He, N. Zhao, and H. Yin, “Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach,” *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 44–55, Jan. 2018.

[29] X. Wang, R. Li, C. Wang, X. Li, T. Taleb, and V. C. M. Leung, “Attention-weighted federated deep reinforcement learning for device-to-device assisted heterogeneous collaborative edge caching,” *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 154–169, Jan. 2021.

[30] A. Vaswani et al., “Attention is all you need,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, Long Beach, CA, USA, 2017, pp. 1–15.

[31] H. Liu and G. Cao, “Deep reinforcement learning-based server selection for mobile edge computing,” *IEEE Trans. Veh.*, vol. 70, no. 12, pp. 13351–13363, Dec. 2021.

[32] Z. Chen, L. Zhang, Y. Pei, C. Jiang, and L. Yin, “NOMA-based multi-user mobile edge computation offloading via cooperative multi-agent deep reinforcement learning,” *IEEE Trans. Cogn. Commun. Netw.*, vol. 8, no. 1, pp. 350–364, Mar. 2022.

[33] W. Jiang, D. Feng, Y. Sun, G. Feng, Z. Wang, and X. Xia, “Proactive content caching based on actor-critic reinforcement learning for mobile edge networks,” *IEEE Trans. Cogn. Commun. Netw.*, vol. 8, no. 2, pp. 1239–1252, Jun. 2022.

[34] Y. Sun and X. Zhang, “A2C learning for tasks segmentation with cooperative computing in edge computing networks,” in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, Rio de Janeiro, Brazil, Oct. 2022, pp. 2236–2241.

[35] F. Rezazadeh, H. Chergui, L. Alonso, and C. Verikoukis, “Continuous multi-objective zero-touch network slicing via twin delayed DDPG and OpenAI gym,” in *Proc. IEEE Global Commun. Conf.*, Taipei, Taiwan, Dec. 2020, pp. 1–6.

[36] H. V. Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double Q-learning,” in *Proc. 30th Conf. Artif. Intell. (AAAI)*, Phoenix, AZ, USA, Mar. 2016, pp. 2094–2100.

[37] Z. Liang, Y. Liu, T. Lok, and K. Huang, “Multi-cell mobile edge computing: Joint service migration and resource allocation,” *IEEE Trans. Wireless Commun.*, vol. 20, no. 9, pp. 5898–5912, Sep. 2021.

[38] Z. Yao, S. Xia, Y. Li, and G. Wu, “Cooperative task offloading and service caching for digital twin edge networks: A graph attention multi-agent reinforcement learning approach,” *IEEE J. Sel. Areas Commun.*, vol. 41, no. 11, pp. 3401–3413, Nov. 2023.

Zhian Chen received the B.E. degree in electronic information science and technology from China West Normal University, Nanchong, China, in 2021. He is currently pursuing the M.E. degree with Southwest University, Chongqing, China. His research interests include mobile edge computing and unmanned aerial vehicles.

Fei Wang received the B.S. degree in mathematics from Liaocheng University, Liaocheng, China, in 2005, the M.S. degree from Wuhan University, Wuhan, China, in 2007, and the Ph.D. degree in computer science from Chongqing University, China, in 2012. She is currently an Associate Professor with Southwest University, Chongqing, China. She was a Visiting Ph.D. Student under the supervision of Prof. Xi Zhang with the Networking and Information Systems Laboratory, Department of Electrical and Computer Engineering, Texas A&M University , College Station, TX, USA, from March 2017 to March 2018. Her research interests include resource allocation in wireless networks, intelligent reflecting surface (IRS), unmanned aerial vehicle (UAV) communications, and distributed algorithm design in wireless networks. She received the Best Paper Award at IEEE GLOBECOM 2024.

Xi Zhang (Fellow, IEEE) received the B.S. and M.S. degrees in electrical engineering and in computer science from Xidian University, Xi’an, China, the M.S. degree in electrical engineering and in computer science from Lehigh University, Bethlehem, PA, USA, and the Ph.D. degree in electrical engineering and in computer science (electrical engineering systems) from the University of Michigan, Ann Arbor, MI, USA.

He is currently a Full Professor and the Founding Director of the Networking and Information Systems Laboratory, Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX, USA. He was with the Networks and Distributed Systems Research Department, AT&T Bell Laboratories, Murray Hill, NJ, USA, and AT&T Laboratories Research, Florham Park, NJ, USA, in 1997. He was a Research Fellow with the School of Electrical Engineering, University of Technology Sydney, Sydney, NSW, Australia, and the Department of Electrical and Computer Engineering, James Cook University, Australia. He has published more than 430 research articles on wireless networks and communications systems, network protocol design and modeling, statistical communications, random signal processing, information theory, and control theory and systems.

Prof. Zhang is a fellow of the IEEE “For contributions to quality of service (QoS) theory in mobile wireless networks.” He received the U.S. National Science Foundation CAREER Award in 2004 for his research in the areas of mobile wireless and multicast networking and systems. He received seven Best Paper Awards at IEEE GLOBECOM 2024, IEEE GLOBECOM 2020, IEEE ICC 2018, IEEE GLOBECOM 2014, IEEE GLOBECOM 2009, IEEE GLOBECOM 2007, and IEEE WCNC 2010, respectively. One of his IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS articles has been listed as the IEEE Best Readings Paper (receiving the highest citation rate among all IEEE TRANSACTIONS/journal articles in the area) on wireless cognitive radio networks and statistical QoS provisioning over mobile wireless networking. He is an IEEE Distinguished Lecturer of IEEE Communications Society and IEEE Vehicular Technology Society. He received the TEES Select Young Faculty Award for Excellence in Research Performance from the College of Engineering, Texas A&M University, in 2006, and the Outstanding Faculty Award from Texas A&M University in 2020. He is serving or has served as an Editor for IEEE TRANSACTIONS ON COMMUNICATIONS, IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING, and IEEE TRANSACTIONS ON NETWORK AND SCIENCE AND ENGINEERING; twice as a Guest Editor for IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS for two special issues on “Broadband Wireless Communications for High Speed Vehicles” and “Wireless Video Transmissions;” an Associate Editor for IEEE COMMUNICATIONS LETTERS; twice as a Lead Guest Editor for *IEEE Communications Magazine* for two special issues on “Advances in Cooperative Wireless Networking” and “Underwater Wireless Communications and Networks: Theory and Applications;” a Guest Editor for *IEEE Wireless Communications Magazine* for Special Issue on “Next Generation CDMA Versus OFDMA for 4G Wireless Applications;” an Editor for *Wireless Communications and Mobile Computing* (Wiley), *Journal of Computer Systems, Networking, and Communications*, and *Security and Communications Networks* (Wiley); and an Area Editor for *Computer Communications* (Elsevier). He is serving or has served as the TPC Chair for IEEE GLOBECOM 2011, the TPC Chair for IEEE ICDCS 2026, the TPC Vice-Chair for IEEE INFOCOM 2010, the TPC Area Chair for IEEE INFOCOM 2012, the Panel/Demo/Poster Chair for ACM MobiCom 2011, the General Chair for IEEE ICDCS 2024 Workshop on “Digital Twin-Enabled 6G Multi-Tier Distributed Computing Systems,” the General Chair for IEEE WCNC 2013, and the TPC Chair for IEEE INFOCOM (2017–2019) Workshops on “Integrating Edge Computing, Caching, and Offloading in Next Generation Networks.”