Deep Reinforcement Learning Based Energy-efficient Task Offloading for Secondary Mobile Edge Systems

Xiaojie Zhang, Amitangshu Pal, Saptarshi Debroy
City University of New York, Temple University
Email: *xzhang6@gradcenter.cuny.edu, amitangshu.pal@temple.edu, saptarshi.debroy@hunter.cuny.edu*

***Abstract*—In order to support last-mile wireless connectivity of computation-intensive applications, edge systems can benefit from secondary (i.e., opportunistic) utilization of licensed spectrum. However, spectrum sensing for such secondary utilization can end up causing considerable energy expenditure for already energy-constrained mobile devices. In this paper, we propose an energy-aware task offloading strategy for secondary edge systems that aims to find trade-offs between channel sensing and task offloading for mobile device energy optimization. The proposed strategy employs a Deep Reinforcement Learning based approach that rewards secondary mobile devices for taking part in cooperative spectrum sensing by allowing them to offload their compute-intensive tasks to edge servers in order to conserve energy. Using simulations, we demonstrate how effectively the proposed strategy can capture dynamic channel states and enforce intelligent offloading decisions. Results show our strategy's benefits over optimization-based approaches and demonstrate its practicality for real-world use-cases where devices are controlled by different stakeholders.**

***Index Terms*—Mobile edge computing, energy efficiency, task offloading, deep reinforcement learning, cooperative spectrum sensing, secondary users.**

I. Introduction

Rapid emergence and deployment of complex, computation-intensive, and mission-critical applications, such as 3D reconstruction [1] of static or dynamic scenes, augmented/virtual/mixed reality (AR/VR/XR) [2]–[4], and visual computing are placing considerable computation resource demands on camera-enabled mobile devices that are capturing the images and videos used for such applications. However, these end devices (e.g., drones, robots, and vehicles) are limited by their physical size, computation capability, and/or energy budget are proving to be incapable of handling all in-device computations required by such applications. At the same time, offloading all computations to cloud data centers is proving to be counter-productive as it incurs considerable end-to-end latency that fails to satisfy the often real-time latency requirements of such latency-sensitive applications [5]. Alternatively, Mobile-edge computing (MEC) as a new computing paradigm can eliminate 'device-computation' constraints and 'cloud-computation' delays by offering considerable computation resources closer to the application site [6], [7].

Many of these edge-assisted applications are deployed in emerging and dynamic use-cases (e.g., disaster response [5], tactical scenarios [8], and connected automated vehicles (CAVs) [9]) that are challenged in terms of availability of licensed spectrum for last mile (mobile device to edge) wireless connectivity. Thus researchers have recently proposed

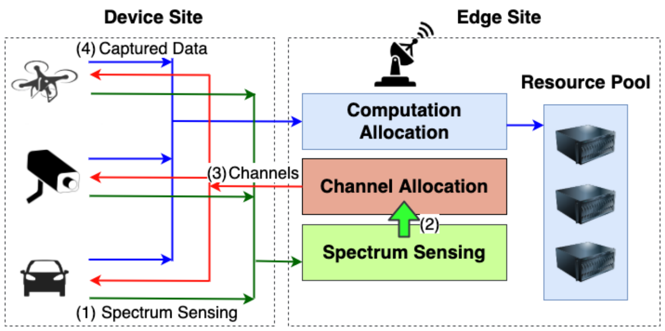

Fig. 1: Modern edge-assisted compute-intensive application use cases using unlicensed spectrum for device-to-edge wireless communication

dynamic spectrum access [10]–[13] that support such last mile connectivity between mobile devices and edge servers by opportunistically using licensed spectrum as secondary users (SU) when channel license holders i.e., primary users (PU) are not using the channels. Adoption of such secondary mode of wireless communication is facilitated by Federal Communications Commission's (FCC) new ruling on unlicensed communications on 6 GHz spectrum band where secondary mobile devices are allowed to use licensed spectrum in a power-efficient manner without interfering with PU transmission [14]. In order to achieve that, mobile devices acting as SUs can: (1) take part in cooperative spectrum sensing [15], [16] where such SUs periodically sense local spectrum for PU transmission, (2) fuse individual spectrum information at a fusion center (FC), (3) use the fusion outcome to identify available (free from PU transmission) channels within the spectrum, and (4) use such available channels for data transmission during task offloading to edge servers (as shown in Fig. 1).

Unlike traditional use cases, such Citizen Broadband radio Service (CBRS) in 3.5 Ghz [17] where dedicated sensing infrastructures are deployed to perform spectrum sensing on behalf of SUs, the aforementioned emerging and dynamic use cases in 6 GHz band have to reply on mobile SU devices themselves taking part in the sensing process. However achieving cooperative spectrum sensing in a MEC system supporting such use cases is challenging as the SU devices there are also responsible for data transmission and local-computation. Long periods of spectrum sensing can consume non-negligible amount of energy that puts pressure on already energy-constrained mobile devices as their primary roles in compute-intensive applications are to contribute to the local computation process and fully/partially offload data to edge servers - both of which result considerable energy consumption. Therefore, the default strategy of such devices would be to act selfishly by sticking to their primary roles and not contribute to the cooperative sensing process in order to conserve energy. However, lack of cooperation in sensing by SU devices significantly compromises the accuracy of the fused channel availability (from PU transmission) information with false alarms. This in turn limits the mobile devices' ability to use available channels for task offloading (for false positives) as well as increasing the chances of harmful interference with the PUs (for false negatives). Thus for MEC systems operating in an opportunistic manner on licensed spectrum, there is a need to find a trade-off between channel sensing and task offloading for device energy optimization.

In this paper we propose an energy-aware task offloading strategy for MEC systems that utilize the licensed spectrum in an opportunistic way for data transmission from mobile devices to edge servers. The proposed strategy deploys a deep reinforcement learning (DRL) based approach in order to capture the spatio-temporal characteristics of PU spectrum usage activity that is key to improve the device sensing efficiency without compromising energy expenditure for task offloading. Unlike prior works using DRL [12], [13] that tackle the problem of making sensing decision with fully cooperative SUs, we use deep learning for scenarios where mobile devices acting as SUs are selfish in nature and are trying to minimize their own energy consumption. In our strategy, SU devices are rewarded for taking part in cooperative sensing by allowing them to offload their computation-intensive tasks to the edge servers to converse energy. In particular: i) We design a DRL-based sensing strategy which helps to improve sensing efficiency and save device sensing energy; ii) We devise a novel rule-agnostic spectrum information fusion mechanism to identify available channels; iii) We develop a reward-based sensing mechanism which allows SU devices to compete for computation (from common edge server) and network/spectrum resources (available channels) as an incentive for taking part in cooperative spectrum sensing; and iv) We propose a mechanism to provide SU devices the flexibility to configure computation speed and transmission power. This allows SU devices to intelligently offload tasks to the edge servers either partially or fully based on their energy budget;

We perform extensive and realistic simulations in order to verify the learning ability of the proposed DRL based task offloading strategy. The results demonstrate high adaptability and efficiency in optimizing the weighted objective. We show how effectively the algorithm can capture the nature of channel states and can propose intelligent policy for offloading decision-making under different system environment settings (e.g., fusion rules). We also show existence of significant correlation between the channel-sensing contribution that a SU device is willing to make and its core computation requirements (e.g., computation cycles and data size). In addition, compared to non-intelligent channel-sensing policies, such as greedy sensing (i.e., strategy where SU devices always sense all the channels), we demonstrate that the reward mechanism guarantees higher benefit by using our proposed learning-based sensing policy. These highlight that our reward-based solution can be practical in many real-world use cases, such as disaster response, tactical scenarios, and CAVs where SU devices are controlled by different stakeholders and are only motivated by their own energy preservation requirements. We also compare our solution to non-learning based Lyapunov optimization approaches that are widely used for dynamical systems [18], [19]. We show that without complete knowledge about the features of underlying mathematical model (which Lyapunov optimization approaches demand), our proposed DRL based algorithm has the ability to intelligently adjust computation speed and transmission power according to the changes in energy preservation and task offloading requirements.

The rest of the paper is organized as follows. Section II presents the related work. Section III proposes the system model. Section IV presents the deep reinforcement learning based strategy. Section V proposes the Lyapunov optimization. Section VI discusses the simulation results. Section VII concludes the paper.

II. Related Work

Deep Reinforcement Learning (DRL) has a wide footprint in literature due to many usages of neural networks in different computer scientific and engineering problems. However, most of the previous works such as [20]–[22] require the action space to be either discrete or continuous. Therefore such works may fail to provide practical solutions for the real-world use cases. To deal with such issues, a new learning pattern with a discrete-continuous hybrid action space has emerged that is first introduced in [23] and is widely extended in other works, such as [24]–[26]. These DRL techniques show excellent ability in handling agent control problems in complicated environment with high-dimensional state and action spaces. Compared to [23], the authors in [24] propose a Parameterized Deep Q-Network (PDQN) for learning behaviours in hybrid action spaces without approximation or relaxation. In [25], the authors introduce a new algorithm which separates the action-parameter inputs and performs multiple passes in a single Q-network. While both [24] and [25] consider single-agent decision making, authors in [26] propose a coordinated approach for multi-agent learning settings based on centralized training and decentralized execution framework. *However, none of these efforts are specifically designed for mobile edge computing systems that employ cooperative spectrum sensing for channel identification.*

The application of deep learning in **Edge Computing** is becoming increasingly popular. Works such as [27]–[30] are aimed at optimize long-term performance of edge systems with respect to the energy preservation. Authors in [27] propose a post-decision state (PDS) based learning algorithm for the edge resource management. Their algorithm decomposes the optimization process into the offline value iteration and the online reinforcement learning. Authors in [30] establish a deep reinforcement learning-based online offloading (DROO) framework to find the binary offloading decisions and the wireless resource allocations under time-varying wireless channel conditions. In [30], the actions are generated based on the order-preserving quantization method. The authors of [28] and [29] consider ultra-dense networks where computational tasks are offloaded to the edge server via different base stations. They also apply a double deep Q-network (DQN) to learn the optimal computation offloading policy without a priori knowledge of network dynamics. *However, the above works only consider data communication using licensed spectrum. Thus their algorithms and models are not directly applicable to MEC systems where mobile devices' offloading decisions are governed by such devices' participation in spectrum sensing.*

Learning-based **Cooperative Spectrum Sensing** for unlicensed networks is becoming another popular research focus as the need for new models for unlicensed communication keep increasing with FCC releasing more licensed bands for unlicensed transmission. In fact, as PUs in such licensed bands come with varied transmission characteristics, it is getting increasingly difficult to predict PU activities for efficient and interference free communication by unlicensed SUs. DRL based techniques provide mechanisms that can successfully capture the features of channel states without any given prior knowledge of the PU activities. In related works, authors in [11] propose a distributed multi-agent sensing network where each SU collects information from the environment and decides its own sensing policy. However, reference [11] simply considers sensing reward without taking into account energy consumption for sensing which is an important factor both in MEC. In contrast, authors in [12] and [13] propose learning-based algorithms for individual SU sensing decision making under the constraint of limited energy. *Although works such as [11]–[13] have taken advantage of cooperative sensing, none of the aforementioned works consider the selfish nature of the SUs in certain environments and neither do they explore any incentive mechanism to motivate SU cooperation.*

III. Task Offloading Through Stochastic Communication

Application use cases that adopt traditional secondary networks, the licensed spectrum can only be used opportunistically after sensing, reporting, fusing, and decision making stages are completed. Such process where SU devices only get to access available channels after a long and energy consuming sensing process would be detrimental for MEC system adopted by dynamic and emerging use cases. Thus, we propose a novel paradigm for future secondary mobile edge systems where the SU devices acquire edge resources as a compensation of sensing. However optimizing energy efficiency considering SU device sensing, computation, and transmission considering edge resource availability along with the resource demands by involved applications become a non-trivial edge resource management problem to solve.

In this paper, we assume a time-slotted communication between SU devices and the edge site for the proposed energy-aware edge network as shown in Fig. 2. For our system model, we assume time slots as $\mathcal{T} \triangleq \{1, 2, ...\}$ where each time slot is of length τ. A time slot is further divided into channel *Sensing Period* and *Data Transmission/Computation Period*. We assume that the *Sensing Period* is long enough so that all channels can be sensed during this period and communication between the SUs and the edge site is carried out through dedicated low bandwidth common *Control Link*. We consider a system with N SU devices with $\mathcal{N} \triangleq \{1, 2, ..., N\}$ adopt cooperative spectrum sensing by sharing local spectrum information with the edge site using the *Control Link* in order to opportunistically access a set of unused licensed frequency channels $\mathcal{I} \triangleq \{1, 2, ..., I\}$ (i.e., becoming the *Data Links* shown in Fig. 2) to offload their computations to the edge site containing edge servers. In our model, the edge server contains a *Fusion Centre* (i.e., FC for sensing reward and spectrum decision management) and a *Resource Manager* (i.e., for CPU allocation and channel assignment) as shown in Fig. 2. Computational tasks are executed inside *Application Container*. The PUs are assumed to be independent of each other where each PU device occupies only one channel. We also assumed the SU devices are cognitive radio enabled with limited computation capacity [31], [32].

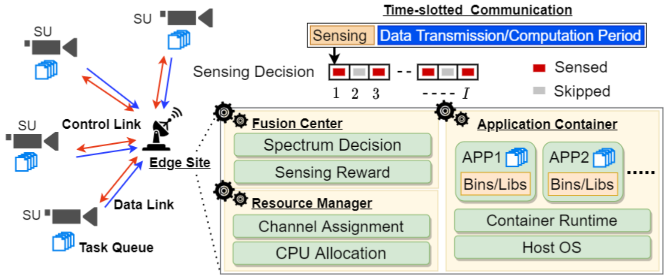

Fig. 2: System model showing sensing, transmission, and computation periods and internal components of the edge site

In our application model, we assume that different SU devices belong to different stakeholders and thus are focused on their individual mission-critical applications. Each application is modeled as a set of identical tasks and a subset of tasks is published/released at regular intervals. More specifically, at the beginning of a time slot, each SU device $n \in \mathcal{N}$ releases b_n number of identical tasks. Each task contains d_n bits of input data and requires l_n CPU cycles for computation. Such computations can be executed locally on the SU devices, or the data can be transmitted to and processed on the edge server using the *Data Link* as shown in Fig. 2 which is nothing but opportunistically acquired licensed channels through the cooperating sensing-fusion-decision making process. For simplicity but without any loss of generality, it is assumed that the transmitted data can be processed only in the next time slot and unprocessed data will be stored in the task queue. The notations used for the overall system and application model are summarized in Table II.

A. Trade-off in Local Sensing

In order to conserve energy, the SU devices are allowed to sense part of the spectrum band and report their partial local sensing results to the FC. However, the typical fusion rule adopted by FCs in cooperative or collaborative sensing is majority voting [16] that requires all SUs to submit their complete local sensing results - this might not be viable in dynamic application use cases.

***Remark** 1:* Thus, we propose a conservative yet flexible 'K-out-N' [33] fusion rule as follows: If any SU device reports that a channel is busy, the channel is marked as unavailable; contrarily the channel is marked as available if at least K SU devices reported the channel as idle and no other SU device reports that channel is busy.

Recent FCC guidelines [14] reveal potential means and methods for future secondary spectrum access in bands such as 6 GHz. According to these guidelines, in WiFi-like small networks in 6 GHz, distributed SU devices can employ collaborative spectrum sensing to obtain local PU activity information and fuse that data at a FC for more precise spectrum availability determination. However, in mission-critical and dynamic application use cases that reply on secondary access of licensed spectrum for data transfer, such continuous spectrum sensing may result in considerable energy consumption that such networks can ill afford. Besides, energy consumption for spectrum sensing can outweigh the energy benefits of offloading computation-intensive jobs to edge server. This makes energy-efficient task offloading in a secondary mobile edge system that operates in 'sense-and-use' mode a non-trivial problem to solve.

For our work, we assume that all SU devices are inherently selfish because they want to acquire wireless channels for themselves (i.e., their stakeholders), and therefore, their willingness to contribute towards sensing varies greatly. In this paper, we study the energy consumption trade-off between local spectrum sensing, computation, and transmission with the following realistic and intuitive considerations: 1) In order to declare a licensed channel available for secondary usage, the FC needs at least K SU devices to report the channel as 'idle' as part of their local sensing report (according to **Remark 1**). Any additional 'idle' reporting (by other SU devices) is considered unnecessary and wastage of the energy by the reporting SU; 2) When only a few SU devices contribute to local sensing, the number of licensed channels available for the secondary usage reduces, resulting in SU devices to be unable to complete their data offloading requirements; and 3) It is unfair to force a SU device with little data transmission/computation requirements to spend significant energy towards sensing; contrarily, given a choice SU devices would only perform computation and transmission without contributing to sensing for the sake of energy saving. In our work, we aim at developing a distributed spectrum sensing scheme which allows the SU devices to perform spectrum sensing intelligently based on their transmission and computation requirements. Our goal is to find an optimal spectrum sensing strategy that can save the energy consumption of sensing while meeting the task offloading requirements of SU devices. We assume that all the SU devices use the same configurations (e.g., sensing power and duration) for individual channel sensing. We also assume that the task of sensing all the channels can be completed within the sensing period.

TABLE I: Notation used

Symbol	Definition
$\mathcal{I}$	The set of licensed frequency channels
$\mathcal{N}$	The set of SU devices
$\mathcal{T}$	Time Slots
τ_d	The transmission/computation period
b_n	The number of released tasks at the beginning of each time slot
d_n	The size of task input data
l_n	The CPU cycles for computation
$\mathrm{x}_n^i(t)$	The sensing decision on channel i of SU n in time slot t
$\mathrm{y}_n^i(t)$	The local sensing result on channel i of SU n in time slot t
$\mathrm{z}^i(t)$	The global sensing result on channel i in time slot t
$w_n^i(t)$	The sensing reward of SU n on channel i in time slot t
$d_n(t)$	The number of transmitted tasks from SU n to edge in time slot t
$c_n^l(t)$	The number of tasks completed locally of SU n in time slot t
$c_n^r(t)$	The number of tasks completed at edge of SU n in time slot t
$f_n^l(t)$	The CPU speed of SU n in time slot t
$f_n^r(t)$	The CPU speed allocated to SU n from server in time slot t
$Q_n^l(t)$	The local queue length of SU n in time slot t
$Q_n^r(t)$	The remote queue length of SU n in time slot t

B. Fusion Rules and Sensing Reward

***Remark** 2:* We define $\mathbf{x}_n(t) \triangleq [\mathrm{x}_n^1(t), \mathrm{x}_n^2(t), ..., \mathrm{x}_n^I(t)]$ as the sensing decision in time slot t by the SU device n, where $\mathrm{x}_n^i(t) \in \{1 \text{ (sensed)}, 0 \text{ (skipped)}\}$ and $i \in \mathcal{I}$. The energy consumption for spectrum sensing can be stated by

$$\mathrm{E}_n^s(t) = \sum_{i=1}^{I} \mathrm{x}_n^i(t) \cdot (\tau_s \times p_s + \epsilon_s) \tag{1}$$

where $(\tau_s \times p_s + \epsilon_s)$ indicates the energy spent on collecting and computing the local sensing result upon received PU signal samples for a single channel. Specifically, τ_s is the sensing time, p_s is the sensing power, and ϵ_s represents the energy spent on computing local sensing results (a very lightweight computation, thereby ignored in the simulation later in Section VI).

In this paper, we consider the fusion rule that we proposed in **Remark 1**. The local sensing results from SU devices are denoted by $\mathbf{y}_n(t) \triangleq [\mathrm{y}_n^1(t), \mathrm{y}_n^2(t), ..., \mathrm{y}_n^I(t)]$ where $\mathrm{y}_n^i(t) \in \{-1 \text{ (skipped)}, 1 \text{ (sensed idle)}, 0 \text{ (sensed occupied)}\}$. As shown in Fig. 3, upon receiving the state vectors $\mathbf{y}(t) \triangleq [\mathbf{y}_1(t), \mathbf{y}_2(t), ..., \mathbf{y}_N(t)]$ from SU devices during the sensing period, the FC hosted by the edge site applies the 'K-out-N' decision rule and determines a global sensing outcome, which indicates the channel availability for the secondary usage.

We also denote $\mathbf{z}(t) \triangleq [\mathrm{z}^1(t), \mathrm{z}^2(t), ..., \mathrm{z}^I(t)]$ as the global sensing outcome determined by the FC, where $\mathrm{z}^i(t) \in \{1 \text{ (sensed idle)}, 0 \text{ (sensed occupied)}\}$. If the FC determines

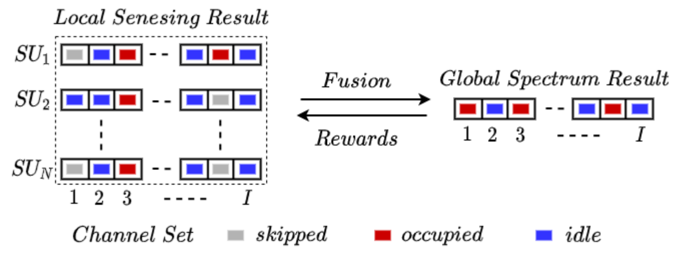

Fig. 3: The model of CSS and the reward for sensing. The SU devices who reported the same channel state as the global spectrum result are rewarded.

that a channel is idle (i.e., $\mathrm{y}_n^i(t) \neq 0,\ \forall n \in \mathcal{N}$ and $\sum_{n=1}^{N} \mathrm{y}_n^i(t) \geq K$), only the SU devices that had reported the same sensing scalar are rewarded. The reward mechanism on individual channels is as follows:

$$w_n^i(t){=}\begin{cases}1/\sum_{n=1}^{N}\mathrm{y}_n^i(t) & \mathrm{z}^i(t)=1,\ \mathrm{y}_n^i(t)=1\\ 0 & \mathrm{z}^i(t)=0\end{cases} \tag{2}$$

This incentive mechanism aims to motivate the SU devices to contribute to the sensing process. The reward function in Eq. (2) also implies that the SU devices should sense on channels that are worthy for sensing (i.e, with the value of $P\{\mathrm{z}^i(t)=1\}$ is high and keeps $\sum_{n=1}^{N}\mathrm{y}_n^i(t)$ to K). This is done by evenly distributing fixed revenue to the SU devices that receive the reward and only rewarding the sensing on channels that the FC determines to be idle. We propose the efficiency of sensing for SU devices in terms of reward and energy consumption that can be stated as:

$$\text{sensing efficiency} = \frac{1}{N}\sum_{n=1}^{N}\left(\big(\sum_{i=1}^{I}w_n^i(t)\big)/\mathrm{E}_n^s(t)\right) \tag{3}$$

The sensing efficiency indicates the amount of rewards obtained from all the channels; on the other hand it also shows the energy consumption spent on sensing. In order to improve efficiency, SU devices need to reduce energy consumption while obtaining higher rewards.

C. Conservative Resource Allocation

We employ a queue system for both the processing on the SU devices and the edge site. After obtaining the global sensing results, the edge site first assigns the free channels in set $\mathcal{I}$ to the SU devices (for task offloading) according to the fusion decision and allocates its computation resources to process the received data. It is assumed that the channel allocation follows the sensing reward ordering of the SU device obtained at the beginning of the current time slot t. More specifically, if there are less than N channels that are detected as idle, only the SU devices with higher sensing reward will get a channel and transmit their data to the edge servers. We assume the bandwidth of the channel to be B and the transmission power of the SU device to be $p_n(t)$. Thus the number of task that can be transmitted within the transmission period of time slot t is

$$d_n(t) = \tau_d \cdot \frac{B\log_2(1+\frac{p_n(t)h_n^2}{N_0})}{d_n}$$

In the case where a SU device does not have any access opportunities in a few time slots, the SU device might need to perform local computation to reduce the queue length. We Denote the computation speed of the SU devices for their un-offloaded data as $f_n^l(t)$. Thus, the number of tasks that can be locally processed is

$$c_n^l(t) = \tau_d \cdot \frac{f_n^l(t)}{l_n} \tag{4}$$

and the local queue length increases as

$$Q_n^l(t+1) = max\{Q_n^l(t) - d_n(t) - c_n^l(t), 0\} + b_n \tag{5}$$

We propose a conservative resource allocation policy to guarantee basic service to all the SU devices in the case of intensive computation resource competition. According to this policy, at each time slot, the edge site first reserves a part of edge servers' computation capacity and performs a baseline allocation proportional to the current offloaded computation needs. The rest of the computation resources are allocated based on sensing rewards that the SU devices received at the beginning of each time slot. $Q_n^r(t)$ denotes the remote queue length for the SU device n. The CPU allocation follows

$$f_n^r(t) = \mathrm{F}\cdot\left(\mathrm{c}_1\cdot\frac{Q_n^r(t)\cdot l_n}{\big(\sum_{n'=1}^{N}Q_{n'}^r(t)\cdot l_{n'}\big)} + \mathrm{c}_2\cdot\frac{w_n(t)}{\sum_{n=1}^{N}w_n(t)}\right) \tag{6}$$

where F denotes the total number of CPU cycles that the edge server can run within one time slot and $\mathrm{c}_1+\mathrm{c}_2=1$ identifies the amount of computation resources that are reserved for the baseline allocation. Given $f_n^r(t)$, we can easily calculate the number of processed tasks generated by the SU device n and stored in the edge server as

$$c_n^r(t) = \frac{f_n^r(t)}{l_n} \tag{7}$$

Therefore, the remote queue length $\mathrm{q}_n(t)$ of the SU device has the following evolution

$$Q_n^r(t+1) = max\{Q_n^r(t) - c_n^r(t), 0\} + d_n(t) \tag{8}$$

Based on above discussions, the SU device energy components thus considered are sensing energy, transmission energy, and the local computation energy. Thus total energy $\mathrm{E}_n(t)$ expenditure for device n at time slot t can be represented as

$$\mathrm{E}_n(t) = \mathrm{E}_n^s(t) + p_n\tau_d + \kappa\cdot\tau_d[f_n^l(t)]^3 \tag{9}$$

where κ is a energy consumption factor based on chip architecture [34].

D. Problem Formulation

***Remark** 3:* In order to compute b_n tasks within time slot t, the device can adjust its CPU frequency by

$$f_n^l(t) = min\{\frac{b_n l_n}{\tau}, f_n^{l,max}\} \tag{10}$$

where $f_n^{l,max}$ denotes the maximum CPU capacity of the SU device n. This gives us the minimum energy consumption for local-only computation that can be computed by

$$\mathrm{E}_n^l(t) = min\{\kappa \cdot \frac{(b_n l_n)^3}{\tau^2}, \kappa \cdot (f_n^{l,max})^3 \cdot \tau\} \tag{11}$$

Therefore, the energy threshold for a beneficial task offloading is $\mathbb{E}' = \mathbb{E}[\mathrm{E}_n^l(t)]$; otherwise the SU devices would just choose **local-only** computation and consequently will not participate in the spectrum sensing.

As it is impractical to optimize both energy consumption and queue length at the same time, in this paper we optimize the long-term average performance on the weighted objective function. The weight factor V shows the importance of energy saving compared to the average queue length. Without any loss of generality, we define the local computation profile as $\mathbf{f}^l(t) = [\mathrm{f}_1^l(t), \mathrm{f}_2^l(t), ..., \mathrm{f}_N^l(t)]$, the transmission power profile as $\mathbf{p}(t) = [\mathrm{p}_1^l(t), \mathrm{p}_2^l(t), ..., \mathrm{p}_N^l(t)]$, and the spectrum sensing decision profile as $\mathbf{x}(t) = [\mathbf{x}_1^l(t), \mathbf{x}_2^l(t), ..., \mathbf{x}_N^l(t)]$. Our optimization problem with weight V can thus be stated as follows:

$$\min_{\substack{\mathbf{f}^l(t),\mathbf{p}(t),\\ \mathbf{x}(t)}} \lim_{T\to+\infty} \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\big[\sum_{n=1}^{N} \mathrm{V}\cdot \mathrm{E}_n(t) + Q_n^l(t) + Q_n^r(t)\big]$$

$$\begin{aligned} &\textbf{s.t.}\\ &\textbf{C1}: 0 \le f_n^l(t) \le f_n^{l,max},\ \forall n \in \mathcal{N}\\ &\textbf{C2}: 0 \le p_n(t) \le p_n^{max},\ \forall n \in \mathcal{N}\\ &\textbf{C3}: \mathrm{x}_n^i(t) \in \{0,1\},\ \forall i \in \mathcal{I},\ \forall n \in \mathcal{N} \end{aligned} \tag{\textbf{P1}}$$

In (**P1**), the weight V indicates the trade-off between minimizing the energy consumption and minimizing the queue length. It is not difficult to identify that (**P1**) is a challenging stochastic optimization problem due to the following characteristics: i) over the time space $\mathcal{T}$, channel availability fluctuates, making the sensing behavior and the queue length unpredictable; and ii) the computation resource at SU devices and the edge site are limited, leading to inferior resource competition. Such time-varying variables add additional challenges towards decision making.

IV. Deep Multi-Agent Reinforcement Learning

In this paper, we consider the proposed stochastic optimization problem (**P1**) as a discrete Markov decision process (MDP) and propose a DRL based approach for continuous decision making. More specifically, we apply a DQN [24]–[26] architecture to implement our sensing and task offloading strategy, which is designed to optimize the long-term average performance stated in problem (**P1**). With respect to reinforcement learning, we assign a learning agent to each SU device. The agent has a finite set of states $\mathcal{S}_n$ and a finite set of actions $\mathcal{A}_n$.

A. Learning agent for SU device

We acknowledge that for individual SU devices, the environment is only partially observable by the learning agent, therefore the system state of an SU device only includes the local information and resources.

State: The state of SU device n denoted by $\mathrm{s}_n(t) \in \mathcal{S}_n$ records the lengths of the local and the remote queues. It also contains the selected local CPU frequency $f_n^l(t)$ and the transmission power $p_n(t)$ as well as the sensing decision vector $\mathbf{x}_n(t)$. Additionally, it should store the information of the resources allocated by the edge server in the current time slot. Furthermore, the energy consumption $\mathrm{E}_n(t)$ and the reward $w_n(t)$ by taking the current sensing action should also be considered in the state. Therefore, the state of a SU device $\mathrm{s}_n(t)$ can be stated as:

$$\mathrm{s}_n(t) = [\overbrace{Q_n^l(t), Q_n^r(t)}^{\text{Queue}}, \overbrace{f_n^l(t), p_n(t), \mathbf{x}_n(t)}^{\text{Device}}, \overbrace{f_n^r(t), \mathrm{ch}_n(t)}^{\text{Resource}}$$
$$, \overbrace{w_n(t), \mathrm{E}_n(t), \mathrm{R}_n(t)}^{\text{Payoff}}] \tag{12}$$

where $\mathrm{ch}_n(t) \in \{0$ (channel acquisition unsuccessful), 1 (channel acquisition successful)$\}$.

Action: The action space of SU device n denoted by $a_n(t) \in \mathcal{A}_n$ is a Discrete-Continuous hybrid action space. For spectrum sensing decisions $\mathbf{x}_n(t)$, the SU device may select any index from channel set $\mathcal{I}$. The selected local CPU frequency $f_n^l(t)$ and the transmission power $p_n(t)$ are continuous values within the capacities of the SU devices. On the bassis of such observations, a Parameterized Deep Q-Networks (P-DQN) with two discrete actions can be used as a possible solution. However, unlike the examples proposed in [24]–[26] where actions are mutually exclusive (e.g., an drone cannot fly to right and left at the same time), updating the channel sensing decisions and adjusting the CPU and transmission power can occur simultaneously. Therefore, the system is just performing a single action with multiple action parameters. To cope the Discrete-Continuous hybrid natures of action parameters, we build a priority based learning sensing policy. More specifically, the action is defined as:

$$a_n(t) = [v_1, v_2, ..., v_I, v_{I+1}, v_{I+2}],\ \forall n \in \mathcal{N}$$

where the action space contains $I+2$ action-parameters and $v_i \in [0,1]$. The first I parameters indicate the sensing priority of the individual channels. In each time slot, a priority $v_i(t)$ is assigned to each channel and the sensing decision is made by the following rule:

$$\mathrm{x}_n^i(t) = \begin{cases} 1 & \text{if } v_i(t) > 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

The last two action-parameters perform CPU frequency setting with range $v_{I+1} \cdot [0, f_n^{l,max}]$ and transmission power setting with range $v_{I+2} \cdot [0, p_n^{max}]$ respectively.

System Reward: The reward function $\mathrm{R}_n(t)$ implies the immediate reward received after the transitioning from state $\mathrm{s}_n(t)$ to state $\mathrm{s}_n(t+1)$ by taking action $\mathrm{a}_n(t)$, which is implemented as a weighted function:

$$\mathrm{R}_n(t) = -\mathrm{V} \cdot \mathrm{E}_n(t) - (Q_n^l(t) + Q_n^r(t)) \tag{14}$$

In Eq. (14), $-\mathrm{V} \cdot \mathrm{E}_n(t)$ signifies the penalty in terms of energy consumption while $Q_n^l(t) + Q_n^r(t)$ signifies the queue length. Therefore, minimizing the individual objective functions for each SU device in problem (**P1**) is equivalent to maximizing the SU learning agent's accumulated discounted reward

$$\max_{\mathbf{a}_n(t) \in \mathcal{A}_n} \lim_{T \to +\infty} \sum_{t=1}^{T} \gamma^t \mathrm{R}_n(t), \ \forall n \in \mathcal{N} \tag{15}$$

B. Network Architecture and Algorithm

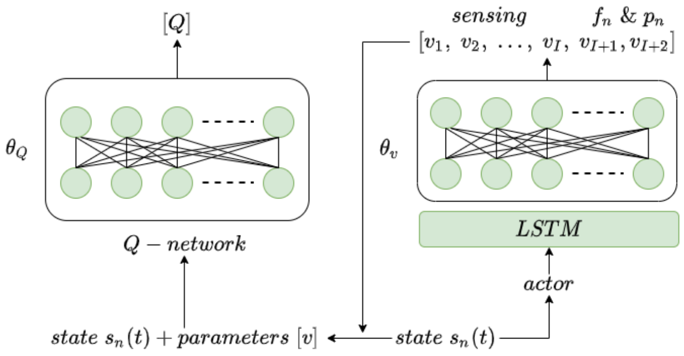

Fig. 4: The network architecture for the proposed spectrum sensing and task offloading learning [24].

In order to capture both spatial and temporary correlations among consecutive time varying environment states of the proposed stochastic task offloading system, we also add Long Short-Term Memory (LSTM) element into our network architecture. It helps to better understand the channel availability and improve the sensing efficiency. As shown in Fig. 4, our network architecture follows a standard P-DQN. Our basic DQN network architecture contains two fully connected layers with feature sizes 512 and 128, respectively. For LSTM, the hidden layer has 128 features. We apply the ϵ-greedy exploration policy for the learning duration and select 0.01 as the learning rate for both networks (Q-network θ_Q and the actor-parameter network θ_v). A replay memory is used to store the history transitions. The online learning algorithm for joint sensing and computing optimization is described in Algorithm 1. In the initialization phase, we make the SU devices to randomly sense the channels and only use half of their maximum CPU and transmission power. Then, the SU devices continuously interact with the current environment and makes sensing and computing decisions at the beginning of each time slot. In the meantime, the network weights are updated from a random sample mini-batch of past transitions.

Algorithm 1: Online Learning for Joint Sensing and Computing Optimization

```
1  Initialize: $\mathrm{x}_n^i(t) = random(0,1)$, $f_n^l = 0.5 \cdot f_n^{l,max}$,
       $p_n = 0.5 \cdot p_n^{max}$
2  for every time slot $t$ do
3      At the beginning of the time slot, each SU device
         observes its current state $s_n(t) \in \mathcal{S}_n$.
4      Input $s_n(t)$ to the actor DQN with weight $\theta_v$ and
         generate the action-parameters $v$.
5      Input both $s_n(t)$ and $v$ to the Q-network with
         weight $\theta_Q$ and compute the $Q$ value.
6      The SU devices update $\mathbf{x}(t)$, $\mathbf{f}^l(t)$, $\mathbf{p}(t)$ based on
         the chosen actions and start the process of
         channel sensing and send local sensing results
         $\mathbf{y}(t)$ to the edge sever.
7      The edge server computes the global sensing
         results $\mathbf{z}(t)$ and calculates the sensing reward
         $w_n(t)$.
8      The edge server performs conservative resource
         allocation as discussed in section III-C.
9      The SU devices observe the new/next state
         $s_n(t+1)$ and store the transition
         $(s_n(t), a_n(t), s_n(t+1), \mathrm{R}(t))$ into replay memory.
10     Get a sample mini-batch of transitions from the
         replay memory, update DQN weights $\theta_Q$ and $\theta_v$.
```

V. Baseline Approach for Transmission Power and Computation Optimization

When the spectrum sensing decisions are given, the remaining problem of finding the optimal transmission power and computation speed of SUs becomes a classical dynamic programming problem that can be solved using Lyapunov optimization [18], [19]. However, one of the primary conditions of such optimization based approach is to have complete knowledge about the features of the underlying mathematical model which our DRL based approach does not require. Nevertheless, a Lyapunov optimization based approach can optimize the following factors: 1) how many task data can be transmitted to the edge server, 2) how many tasks can be executed locally. The success of optimizing these factors would also have a significant impact on the length of local and remote queues. Thus we use Lyapunov optimization as baseline and compare the efficacy of our proposed DRL based approach in finding these two optimal factors against the baseline.

A. Lyapunov Optimization

In this subsection, we discuss the baseline Lyapunov optimization based approach that can find the optimal solutions for the aforementioned factors. Our objective is to generate the decision vector $\mathbf{f}^l(t)$ and $\mathbf{p}(t)$ for each time slot and keep a stable sum of queue length $Q_n(t) = Q_n^l(t) + Q_n^r(t)$. Using this approach, the DQN networks are applied to learn the behavior of spectrum sensing that will only generate the sensing vector

$\mathbf{x}(t)$. In the evaluation section, we will compare the Lyapunov optimization results against our proposed DRL based results.

Using Lyapunov optimization basd approach, at the very beginning of each time slot t, a Lyapunov function will be defined which is a quadratic equation based on $Q_n^l(t)$ and $Q_n^r(t)$ that can be expressed as

$$L\big(Q_n(t)\big) = \frac{1}{2}\big(Q_n^l(t)^2 + Q_n^r(t)^2\big)$$

There is also a need to define the Lyapunov drift $\Delta(Q_n(t)) = \mathbb{E}[L\big(Q_n(t+1)\big) - L\big(Q_n(t)\big)|Q_n(t)]$ that is used to measure the difference in function $L(Q_n(t))$ between two adjacent time slots. The difference between t and $t+1$ can be computed as

$$\begin{aligned} L\big(Q_n(t+1)\big) - L\big(Q_n(t)\big) = &\frac{1}{2}\big(Q_n^l(t+1)^2 - Q_n^l(t)^2\big) \\ &+ \frac{1}{2}\big(Q_n^r(t+1)^2 - Q_n^r(t)^2\big) \end{aligned}$$

According to similar analysis given in [18], [19], the Lyapunov drift can be rewritten to the following inequality with respect to the queue constraint.

$$\begin{aligned} L\big(Q_n(t+1)\big) - L\big(Q_n(t)\big) \le \ & C - \big(Q_n^r(t)\big(c_n^r(t) - d_n(t)\big) \\ &+ Q_n^l(t)\big(d_n(t) + c_n^l(t) - b_n\big)\big) \end{aligned}$$

Where

$$C = \frac{1}{2}\Big(\big(b_n\big)^2 + \big(d_n(t) + c_n^l(t)\big)^2 + \big(c_n^r(t)\big)^2 + \big(d_n(t)\big)^2\Big)$$

B. Per-time Slot Problems

Using Lyapunov optimization method for a given sensing decision $\mathbf{x}(t)$, problem (**P1**) will be transformed to individual deterministic per-time slot problems. Since our proposed work focuses on the reinforcement learning methods, the detail analysis of the baseline Lyapunov optimization is beyond the scope of this paper. Using such approach, the original problem (**P1**) can be accomplished by solving the following per-time slot problem:

$$\begin{aligned} \min_{\mathbf{f}^l(t),\mathbf{p}(t)} \sum_{n=1}^{N} & V_L \cdot \mathrm{E}_n(t) - Q_n^r(t)\big(c_n^r(t) - d_n(t)\big) \\ & - Q_n^l(t)\big(d_n(t) + c_n^l(t) - b_n\big) \\ s.t.\ & \mathbf{C2},\ \mathbf{C3},\ \forall n \in \mathcal{N} \end{aligned} \qquad (\mathbf{P2})$$

where V_L is the balancing factor that control the importance of energy preservation versus the queue stability. Based on the convex nature of (**P2**), $\mathbf{f}^l(t)$ and $\mathbf{p}(t)$ can be obtained separately by using classical convex optimization methods that can be found in in [18], [19]. In Section VI we will compare the optimality of the solutions achieved by our proposed DRL based approach against the solutions achieved through the baseline Lyapunov optimization based approaches. The objective of such comparison will be to achieve optimality close to Lyapunov optimization based approaches.

VI. Simulation Result

We evaluate the performance of the proposed DRL based task offloading scheme with different system parameters using a simple yet realistic simulation. Specifically, we analyze our work to demonstrate (a) the effect of weight factor V (from Section IV) on controlling the trade-off between the energy consumption and the queue length, (b) the comparison of our learning mechanism with the Lyapunov optimization algorithm proposed in Section V, (c) the impact of fusion rules on channel sensing, (d) the comparison of our proposed DRL-based sensing methods to greedy sensing strategies, and (e) finally, the capability on finding the optimal transmission power and computation speed of SU devices in dynamic systems. In the simulation, we consider a small-scale network with 8 SU devices and 12 licensed channel each having a bandwidth of 3 MHz. The key system parameters are summarized in Table II, which are compatible with the typical CR scenarios proposed in [33].

TABLE II: Simulation Parameters

Variable	Value
Number of SU devices N	8
Transmission / Computation period τ_d	300 ms
Sensing period τ_s	10 ms
Sensing power p_s	0.1 Watts
Number of channels I	12
Channel bandwidth B	3 MHz

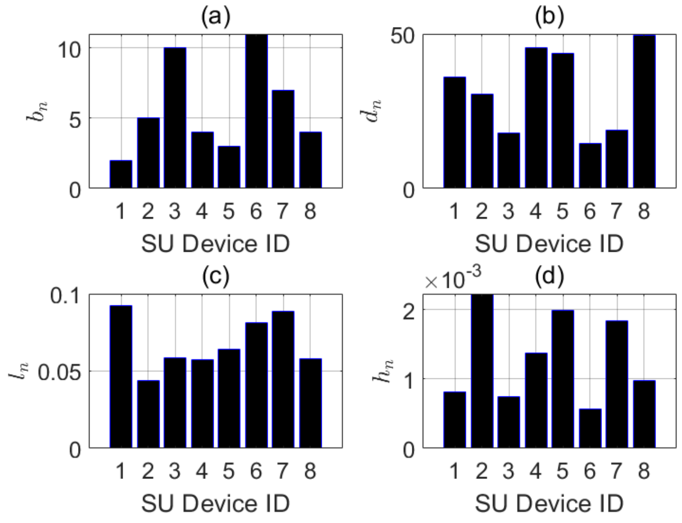

Fig. 5: For each SU device $n \in \mathcal{N}$: a) Number of tasks created at the beginning of time slot. b) The data size of each task. c) The computation requirement of each task. d) The channel gain between SU device and the edge server which is simplified by the distances.

A. Parameter Setup

In the simulation, we first construct tasks with heterogeneous features for the 8 SU devices. In Fig. 5, we represent a) the number of released tasks, b) the size of task input data, c) the CPU requirement of each task and d) the channel gain of individual SU devices. In Fig. 5(d), the channel gains are simulated with independent Rayleigh fading with average power loss set as 10^{-3} [35]. It is reasonable to speculate that the SU devices 3, 6 and 7 are potential candidates who have high intention to offload most of their tasks to edge server (i.e. high computation demands with considerable lower data transmission requirements). Therefore, these devices are likely to spend more effort on spectrum sensing in order to meet their high resource demands. We will prove this point in subsequent results.

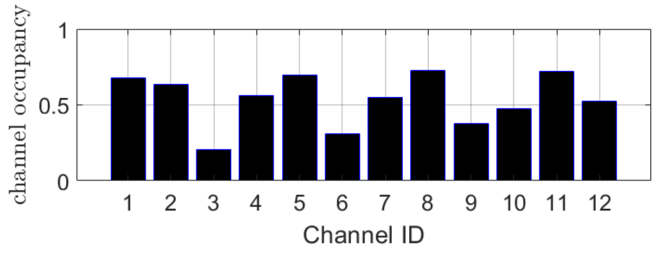

Fig. 6: The licensed channel occupancy probability for different channels in the spectrum

In this paper, the simulation environment contains a PU network using 12 licensed channels. The bandwidth of each channel is set to 3 MHz. We assume a widely accepted model where the PU transmission activities are independent and identically distributed (IID) on each channel [36]. The SU devices compete for secondary access to these 12 licensed channels when the PUs are not using them. We show the PU occupancy probabilities of individual channels used for this simulation in Fig. 6, which are generated uniformly randomly. We demonstrate the channel sensing characteristics in terms of the following two metrics: (a) the frequency of a channel being sensed within a single time slot and (b) the number of channels sensed for each SU device within a single time slot. From Fig. 6 we can observe that the channels 3, 6 and 9 show considerably low utilization from PU point of view, which provides a possibility that they might be popular channels for spectrum sensing by the SU devoces.

B. Effects of Weight Factor V

We first demonstrate how the weight factor V in the objective function (**P1**) impacts the system performance in terms of the average energy consumption and the average task queue length in a long-term period (10K time slots). For this experiment, we set $K = 1$ for the selected fusion rule as discussed in **Remark 1**. The results shown in Fig. 7 demonstrate the mutually exclusive nature of the two performance metrics. Even though we cannot minimize both the energy consumption and the queue length simultaneously, we can still choose the weight that maintains "best" trade-off between the two. In Fig. 7, the recommended weight is neither close to 1 nor close to 90 as these two boundary regions generate either high energy consumption or large queue length. In addition, we found long-tail patterns in Fig. 7 (a) and (b) with the increase of weight V. Based on this observation, we can argue that the optimal weight exists near the starting point of the tail (e.g.,

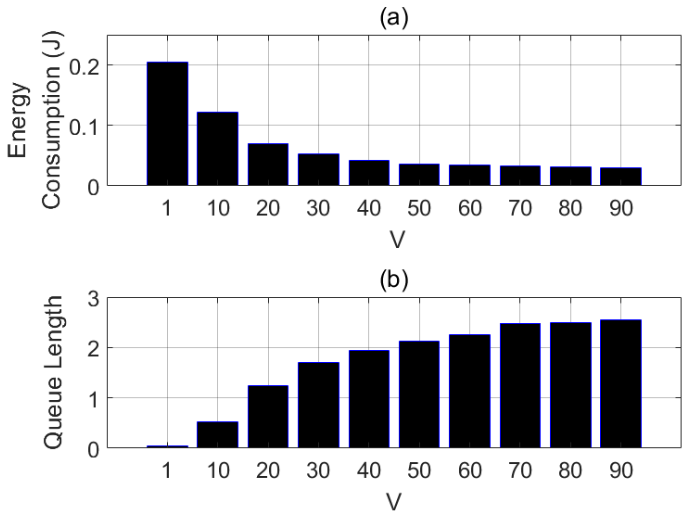

Fig. 7: The impact of the weight factor. (a) The average energy consumption and (b) the average task queue length against different weight factors.

near 20 to 40). Consequently, we select $V = 20$ as the 'near optimal point' having energy consumption and queue length equal to 0.071J and 1.237 respectively. For the rest of the simulations and comparisons, we keep $V = 20$ as default.

C. Comparison with Lyapunov Optimization

We compare the performance of our learning-based strategy against Lyapunov Optimization based approaches with parameters $V = 20$ and $K = 1$. In order to integrate the Lyapunov methods, the learning network generates 12 action parameters for controlling the channel sensing decision-making. We then compute the optimal CPU frequency and transmission power at each time slot by solving (**P2**). Although balance factors V (for DRL strategy) and V_L (for Lyapunov Optimization) in (**P1**) and (**P2**) are generated from two different systems, the way they control the balance between the energy consumption and the task queue length remains comparable (i.e. their characteristics follows similar long-tail patterns). Fig. 8 shows the variation of energy consumption and queue length of Lyapunov Optimization based strategy with different V_L. As shown in Fig. 8(a)-(b), the increase of V_L results in decreasing energy consumption whereas increase of the task queue length.

From Fig. 9 we can observe that the Lyapunov Optimization achieves its optimal solution point (with energy consumption and queue length equal to 0.07J and 1.174 respectively) at $V_L = 175$ and the system reward starts to slowly decrease when $V_L \geq 175$. The maximum system reward is close to our scheme (in subsection VI-B). As both algorithms consider the trade-off between energy consumption and task queue length while making task offload decisions, we argue that without any knowledge of a detailed analytical model, our proposed reinforcement learning-based algorithm still has the ability to scientifically adjust the CPU frequency and transmission power of the SU devices according to the changes in energy consumption and task queue length. *In addition, our proposed reinforcement learning-based algorithm does not have the*

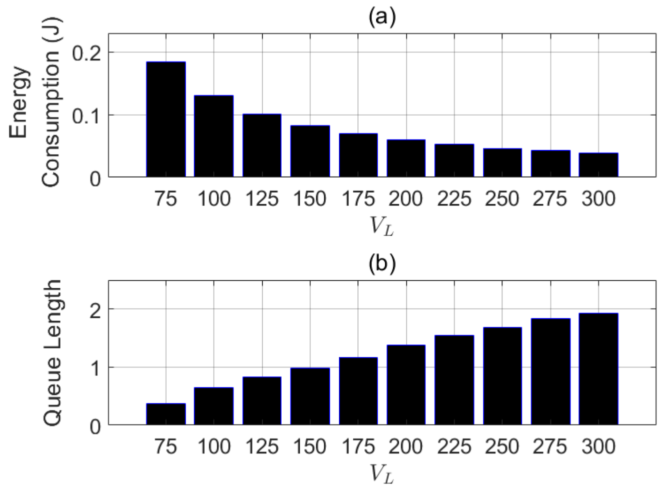

Fig. 8: The impact of the weight factor V_L for Lyapunov Optimization. (a) The average energy consumption and (b) the average task queue length against different weight factors.

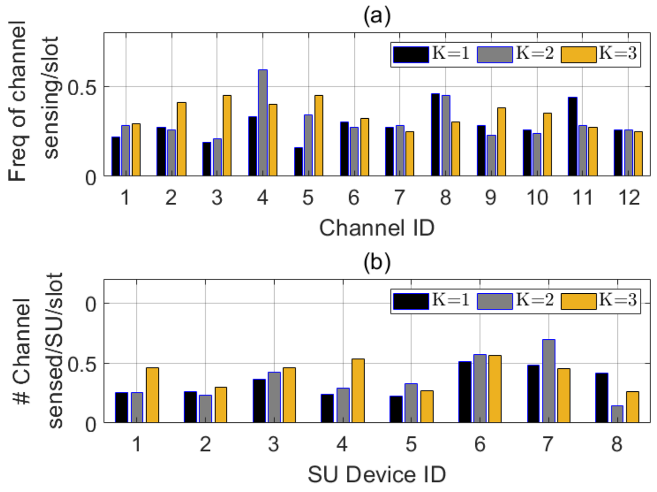

Fig. 10: The channel sensing information against different K-out-N rules: (a) the frequency of a channel been sensed within single time slot, (b) the number of channels sensed for each SU device within single time slot.

requirement to find the optimal weight factor, which is a challenging problem for online algorithms.

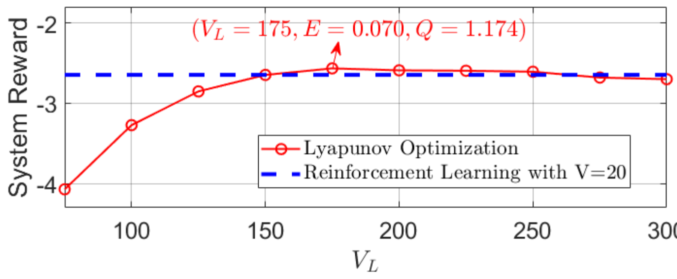

Fig. 9: The system reward of Lyapunov Optimization against reinforcement learning. Lyapunov Optimization achieves its maximum system reward at $V_L = 175$.

D. The Impact of K-out-N Rule on Channel Sensing

Fig. 10 highlights the characteristics of our strategies while jointly considering the channel availability and the sensing rewards. We keep $V = 20$ for Fig. 10. The results also show the relationship between the sensing activities and the chosen K-out-N fusion rule (from Section III) with different values of K. In Fig. 10(a), the frequencies of a channel been sensed within single time slot are 0.28 ($K = 1$), 0.30 ($K = 2$) and 0.34 ($K = 3$) respectively. Similarly in Fig. 10(b) the average number of channels sensed from SU devices are 3.44 ($K = 1$), 3.68 ($K = 2$) and 4.12 ($K = 3$) respectively. Obviously, the larger the K, the more spectrum sensing tasks must be performed.

On the other hand, as we discussed previously that channels 3, 6, 9 are used less frequently by PUs than any other channels as shown in Fig. 6 – therefore they should be more attractive to SU devices for gaining sensing reward. However, the simulation results in Fig. 10(a) do not support such speculation. The results indicate that there is no significant correlation between the channel occupancy probability and the frequency of that channel been sensed by the SUs. The maximum standard deviation of such sensing frequency is less than 10% for both K values presented in Fig. 10(a). The possible reason of such characteristic might be the fact that our sensing reward mechanism forces SU devices not to target the same (popular) channel for sensing. Although there is a high availability probability for a certain channel in the next time slot, if too many SU devices report their sensing results on this channel at the same time, the FC only gives a very limited sensing reward to a single reporting SU. This leads to SU devices evenly distributing their sensing interest on every channel.

Fig. 10(b) shows the number of channels sensed by each SU device within a single time slot. From Fig. 10(b) we can observe that the SU devices 3, 6, and 7 sense more channels than others. As we discussed earlier, this can be explained by their task characteristics presented in Fig. 5. These three particular SU devices have significantly higher computation requirements (b_n and l_n) and just carry fewer task input data d_n that needs to be transmitted to the edge site, which makes them ideal candidates for task offloading from energy saving perspective.

E. Comparison with Greedy Sensing Strategy

We next study the effect of *greedy* sensing strategy where the SU devices try to maximize their own benefits by performing spectrum sensing on all the channels. We demonstrate how such strategy can be detrimental to the long-term performance in terms of system reward, sensing efficiency (from Eq. 3) and sensing characteristics. Fig. 11 shows the comparison of two scenarios: (a) in the first scenario, we make the SU device 6 (the one with highest computation demands) as the *greedy sensing actor* and all other SU devices follow the proposed learning-based sensing strategy, whereas (b) in second scenario we assume all the devices use our learning-based strategy.

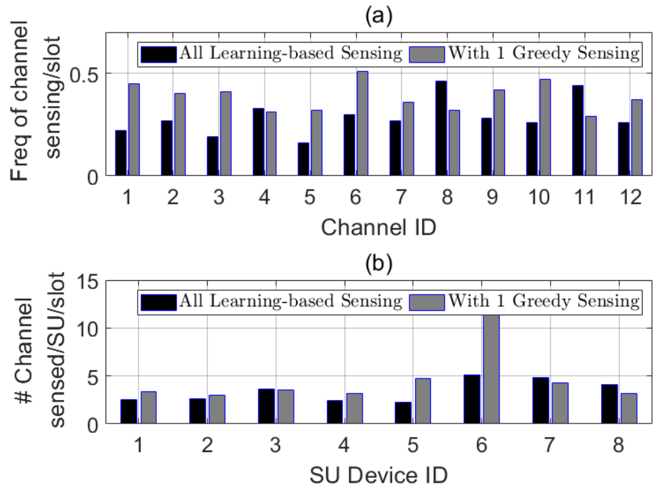

Fig. 11: The channel sensing information.

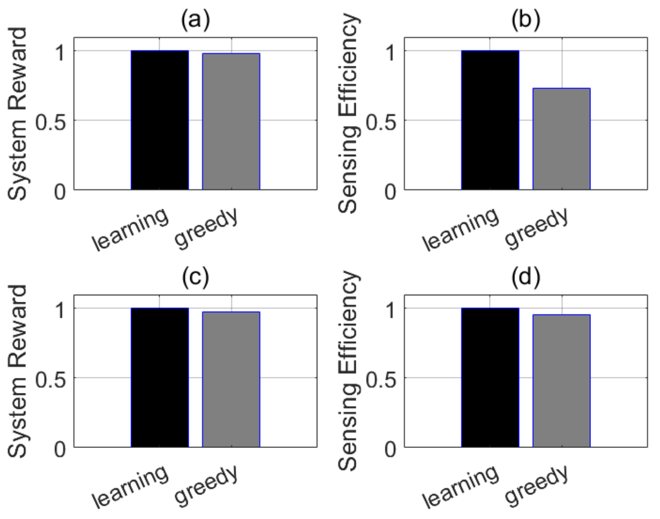

Fig. 12: The normalized system reward and sensing efficiency of the sensing scenarios. (a) and (b): for all SU devices. (c) and (d): for SU device 6 only.

From Fig. 11(a) we can observe that as compared to our learning based sensing strategy, the greedy SU device increases the overall sensing frequency from 0.28 to 0.38 while the number of channels sensed by other SU devices (in Fig. 11(b)) remain the same – which means that the sensing efficiency is decreased due to more sensing overlapping. In Fig. 12(a)-(b), we show that the existence of the greedy SU device compromises the overall system performance. While the system reward only reduces by 2% in Fig. 12(a), the average sensing efficiency is significantly decreased (around 26.4%) due the greedy SU that senses and obtains reward from all the channels. It is evident that a greedy SU brings more unnecessary competition for resources, which is harmful towards energy-saving.

Next, we explore how our proposed learning based scheme can prevent SU devices from becoming greedy and how it can encourage the SU devices to apply the 'best-practice' of learning-based sensing. The results shown in Fig. 12(c)-(d) demonstrate how our scheme can successfully prevent SU devices becoming greedy. It turns out that sensing all the channels does not guarantee an advantage towards task offloading. Due to the proposed sensing reward mechanism, the extra energy spent on channel sensing incurs negative effects on both system reward and sensing efficiency. From these results, we argue that no SU device would like to follow the greedy sensing policy as it jeopardises its task offloading interests. Therefore, we establish that our proposed reward mechanism is practical for the real-world scenarios.

VII. Conclusions

In this paper we proposed a deep reinforcement learning based cooperative sensing strategy for secondary edge systems, where the devices are controlled by multiple stakeholders having their own energy saving objectives. In such a scenario, the proposed strategy incentivizes the SU devices to take part in cooperative spectrum sensing more actively through a novel reward based mechanism. Through simulations we demonstrated that the proposed learning-based mechanism achieves high energy benefits by capturing the unpredictability of spectrum availability while enforcing intelligent energy-aware task offloading. The results validate practicality of our reward-based strategy for real-world use cases where devices are controlled by different stakeholders and are only motivated by their own energy preservation requirements. In future we want to build a prototype setup to test the effect of such energy-aware spectrum-sensing and task offloading strategy in a real edge-computing environment.

References

[1] W. Zhang *et al.*, "Hetero-edge: Orchestration of real-time vision applications on heterogeneous edge clouds," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1270–1278.

[2] J. Ren *et al.*, "An edge-computing based architecture for mobile augmented reality," *IEEE Network*, vol. 33, no. 4, pp. 162–169, 2019.

[3] Y. Sun, Z. Chen, M. Tao, and H. Liu, "Communications, caching, and computing for mobile virtual reality: Modeling and tradeoff," *IEEE Transactions on Communications*, vol. 67, no. 11, pp. 7573–7586, 2019.

[4] J. Lindqvist, "Edge computing for mixed reality," 2019.

[5] R. Gargees *et al.*, "Incident-supporting visual cloud computing utilizing software-defined networking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 1, pp. 182–197, 2016.

[6] C.-F. Liu, M. Bennis, M. Debbah, and H. V. Poor, "Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing," *IEEE Transactions on Communications*, vol. 67, no. 6, pp. 4132–4150, 2019.

[7] H. A. Alameddine *et al.*, "Dynamic task offloading and scheduling for low-latency iot services in multi-access edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 668–682, 2019.

[8] *Smart Emergency Response System (SERS) - http://smartamerica.org/teams/smart-emergency-response-system-sers/.*

[9] S. Liu *et al.*, "Edge computing for autonomous driving: Opportunities and challenges," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1697–1716, 2019.

[10] X. Zhang and S. Debroy, "Migration-driven resilient disaster response edge-cloud deployments," in *2019 IEEE 18th International Symposium on Network Computing and Applications (NCA)*. IEEE, 2019, pp. 1–8.

[11] Y. Zhang, P. Cai, C. Pan, and S. Zhang, "Multi-agent deep reinforcement learning-based cooperative spectrum sensing with upper confidence bound exploration," *IEEE Access*, vol. 7, pp. 118 898–118 906, 2019.

[12] V. Q. Do and I. Koo, "Learning frameworks for cooperative spectrum sensing and energy-efficient data protection in cognitive radio networks," *Applied Sciences*, vol. 8, no. 5, p. 722, 2018.

[13] H. He and H. Jiang, "Deep learning based energy efficiency optimization for distributed cooperative spectrum sensing," *IEEE Wireless Communications*, vol. 26, no. 3, pp. 32–39, 2019.

[14] *FCC Opens 6 GHz Band to Wi-Fi and Other Unlicensed Uses - https://www.fcc.gov/document/fcc-opens-6-ghz-band-wi-fi-and-other-unlicensed-uses.*

[15] S. Debroy, S. Bhattacharjee, and M. Chatterjee, “Spectrum map and its application in resource management in cognitive radio networks,” *IEEE Transactions on Cognitive Communications and Networking*, vol. 1, no. 4, pp. 406–419, 2015.

[16] S. Bhattacharjee, S. Debroy, and M. Chatterjee, “Quantifying trust for robust fusion while spectrum sharing in distributed dsa networks,” *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 2, pp. 138–154, 2017.

[17] *3.5 GHz Band Overview - https://www.fcc.gov/wireless/bureau-divisions/mobility-division/35-ghz-band/35-ghz-band-overview*.

[18] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, “Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems,” *IEEE Transactions on Wireless Communications*, vol. 16, no. 9, pp. 5994–6009, 2017.

[19] S. Pan and Y. Chen, “Energy-optimal scheduling of mobile cloud computing based on a modified lyapunov optimization method,” *IEEE Transactions on Green Communications and Networking*, vol. 3, no. 1, pp. 227–235, 2019.

[20] H. Van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double q-learning,” in *AAAI conference on artificial intelligence*, 2016.

[21] V. Mnih *et al.*, “Asynchronous methods for deep reinforcement learning,” in *International conference on machine learning*, 2016, pp. 1928–1937.

[22] T. P. Lillicrap *et al.*, “Continuous control with deep reinforcement learning,” *arXiv preprint arXiv:1509.02971*, 2015.

[23] M. Hausknecht and P. Stone, “Deep reinforcement learning in parameterized action space,” *arXiv preprint arXiv:1511.04143*, 2015.

[24] J. Xiong *et al.*, “Parametrized deep q-networks learning: Reinforcement learning with discrete-continuous hybrid action space,” *arXiv preprint arXiv:1810.06394*, 2018.

[25] C. J. Bester, S. D. James, and G. D. Konidaris, “Multi-pass q-networks for deep reinforcement learning with parameterised action spaces,” *arXiv preprint arXiv:1905.04388*, 2019.

[26] H. Fu *et al.*, “Deep multi-agent reinforcement learning with discrete-continuous hybrid action spaces,” *arXiv preprint arXiv:1903.04959*, 2019.

[27] J. Xu, L. Chen, and S. Ren, “Online learning for offloading and autoscaling in energy harvesting mobile edge computing,” *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 3, pp. 361–373, 2017.

[28] X. Chen *et al.*, “Performance optimization in mobile-edge computing via deep reinforcement learning,” in *IEEE VTC-Fall*.

[29] X. Chen *et al.*, “Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning,” *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4005–4018, 2019.

[30] L. Huang, S. Bi, and Y. J. Zhang, “Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks,” *IEEE Transactions on Mobile Computing*, 2019.

[31] B. Liu *et al.*, “Energy-efficient cooperation in mobile edge computing-enabled cognitive radio networks,” *IEEE Access*, vol. 7, pp. 45 382–45 394, 2019.

[32] I. Kakalou, K. E. Psannis, P. Krawiec, and R. Badea, “Cognitive radio network and network service chaining toward 5g: Challenges and requirements,” *IEEE communications Magazine*, vol. 55, no. 11, pp. 145–151, 2017.

[33] M. Zheng, W. Liang, H. Yu, and M. Song, “Smcss: A quick and reliable cooperative spectrum sensing scheme for cognitive industrial wireless networks,” *IEEE Access*, vol. 4, pp. 9308–9319, 2016.

[34] X. Zhang and S. Debroy, “Energy efficient task offloading for compute-intensive mobile edge applications,” in *Proceedings of the IEEE ICC*, 2020.

[35] C. You, K. Huang, H. Chae, and B. Kim, “Energy-efficient resource allocation for mobile-edge computation offloading,” *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2017.

[36] S. Debroy, S. De, and M. Chatterjee, “Contention based multichannel mac protocol for distributed cognitive radio networks,” *IEEE Transactions on Mobile Computing*, vol. 13, no. 12, pp. 2749–2762, 2014.