Deep Deterministic Policy Gradient Algorithm for Space/Aerial-Assisted Computation Offloading

Jielin Fu, Lei Liang, Yanlong Li(✉), and Junyi Wang

Guilin University of Electronic Technology, Guilin 541004, China
lylong@guet.edu.cn

Abstract. Space-air-ground integrated network (SAGIN) has been envisioned as a promising architecture and computation offloading is a challenging issue, with the growing demand for computation-intensive applications in remote area. In this paper, we investigate a SAGIN edge computing architecture considering the energy consumption and delay of computation offloading, in which ground users can determine whether take advantage of edge server mounted on the unmanned aerial vehicle and satellite for partial offloading or not. Specifically, the optimization problem of minimizing the total cost is formulated as a Markov decision process, and we proposed a deep reinforcement learning-based method to derive the near-optimal policy, adopting the deep deterministic policy gradient (DDPG) algorithm to handle the large state space and continuous action space. Finally, simulation results demonstrate that the partial offloading scheme learned from proposed algorithm can substantially reduce the user devices' total cost as compared to other greedy policies, and its performance is better than the binary offloading scheme learned from Deep Q-learning algorithm.

Keywords: Space-air-ground integrated network · Edge computing · Partial offloading · Reinforcement learning

1 Introduction

Nowadays, with the in-depth development of the 5G mobile communication system and the research on 6G, an interconnected world is gradually opening up to people. Meanwhile, the rapid development of various communication services and the continuous improvement of application demands put forward higher requirements for network coverage, data transmission rate and end-to-end delay. However, the existing ground network coverage is limited and cannot provide services to meet the services for remote areas such as mountainous areas, polar regions and oceans. Space-Air-Ground Integrated Network (SAGIN) can achieve the global seamless coverage, breaking through the limitations of Ground networks, so it has become an emerging hot research topic [1,2].

H. Gao et al. (Eds.): ChinaCom 2021, LNICST 433, pp. 523–537, 2022.
https://doi.org/10.1007/978-3-030-99200-2_39

SAGIN is based on the ground cellular network and combines the advantages of the wide coverage of the satellite network and the flexible deployment of the aerial platform to achieve seamless coverage through the convergence of heterogeneous networks [3]. However, with the rapid development of the Internet of Things (IoT), more and more computation-intensive applications pose challenges with the limited computing capability and battery life of the devices. In general, users utilize mobile edge computing (MEC) to offload computing tasks to data centers with rich computing resources for processing, which can make up for the defects in computing capability and storage resources of users' devices to some extent [4]. Therefore, MEC technology of terrestrial network is introduced in SAGIN to provide users with efficient and flexible computing services by utilizing multi-level and heterogeneous computing resources at the edge of network. Through offloading the computation intensive tasks to MEC sercer, the energy consumption and latency can be reduced. On the other hand, employing the SAGIN in computation offloading introduces several challenging issues. Firstly, different SAGIN segments possess distinct network conditions. Secondly, due to transmission delay, it may not be able to meet the requirements of time-sensitive applications such as virtual reality [5]. Therefore, it is very necessary to design an efficient computation offloading scheme.

Early studies on MEC mainly focused on looking for solutions for the allocation of computing and communications resources, as well as offloading strategies for various computing tasks,in which computation services are provided by a fixed base station in terrestrial networks. Computing offloading has been studied extensively, and most of these researches have proposed traditional optimization approaches, such as convex optimization methods [6] and Lyapunov optimization [7], and deep learning algorithms, such as Q-learning [8], Deep-Q-network (DQN) [9], and distributed deep learning [10], to solve this problem. However, the MEC services cannot effectively operate in the scenarios where communication infrastructures are sparasely distributed if the services are provided only through the terrestrial fixed facilities.

Recently, the research on computation offloading in SAGIN has been at its initial stage. Considering the coverage range and channel conditions of the UAVs in SAGIN, C. Zhou et al. [11] proposed a computation offloading scheme based on linear programming to solve the dynamic scheduling problem. To solve the allocation of communication and computing resources, an effective scheme based on reinforcement learning was proposed in literature [12]. But the above study did not consider the curse of dimensionality. Under given UAV energy consumption constraints, a risk-aware reinforcement learning algorithm was proposed to weigh delay and risk in SAGIN scenario in [13]. In [14], Thai et al. proposed a learning-based offloading scheme to optimize network performance and maximize server provider revenue. In [15], Tang et al. investigated the computation offloading decisions to minimize the sum energy consumption of ground users, proposed a distributed algorithm by leveraging the convex optimization method to approximate the solution. Although problems in high-dimensional state spaces have been successfully solved by DQN, only discrete action spaces

can be handled. In [16], N. Cheng et al. proposed a joint resource allocation and task scheduling methods based on actor-critic algorithm to effectively allocate computing resources to virtual machines and achieve lower total cost of task scheduling. However, the task partial offloading and offloading scheduling are intercoupled with each other. It is important to note that the aforementioned works considered complete offloading, while partial offloading can significantly improve the latency as the network becomes dense and the edge resources are limited [17].

As a summary, the study of partial offloading considering the cooperation of space, aerial and ground multi-layer network under multi-user environment is still missing in above literatures. In addition, traditional reinforcement learning based methods and linear programming methods cannot solve the high-dimensional or continuous action space scenes.

Therefore, this paper considers the problem of computation offloading under the SAGIN architecture with the joint communication and computing (C2) service. In order to deal with these challenges, the optimization problem is formulated as a Markov decision process (MDP), and a partial offloading strategy based on deep deterministic policy gradient (DDPG) using the actor-critic algorithm to deal with large state and continuous action spaces is proposed to minimize the weighted sum of energy consumption and delay.

The remainder of this paper is organized as follows. In Sect. 2, the SAGIN architecture and computation offloading models are introduced. In Sect. 3, we describe the formulation and transformation, followed by a DDPG based solution. The simulation results and experimental evaluation are provided in Sect. 4. Finally, we introduce the conclusion and the future work briefly in Sect. 5.

2 System Model

In this section, we first introduce the network architecture and then describe the models associated with communication and computation for task offloading.

2.1 The SAGIN Architecture

In this work, we consider an SAGIN architecture with N ground users (GUs), I UAVs and a low earth orbit (LEO) satellite constellation. There are many typical applications, such as automated drilling control and virgin forest monitoring [2]. As shown in Fig. 1, a remote region without cellular coverage is considered, therefore we provide network access, edge computing, and caching through the aerial segment. In the aerial segment, UAVs can serve as edge servers to provide computing capabilities to GUs [18], which can be regarded as the replacement of BSs. Let $\mathbf{N} = \{1, 2, \ldots, N\}$ be the set of indices of N GUs. Then, the set of SAGIN components that computing tasks can be offloaded to is denoted by $\mathbf{I} = \{0, 1, 2, \ldots, I\}$, let indexes $1, 2, \ldots, I$ and 0 denote the UAVs and the LEO satellite constellation respectively. Due to the limited computing power and battery capacity of the GU devices, some tasks need to be offloaded to the

flying UAVs configured with fixed locations that act as attitude platforms, or the LEO satellite constellation. Furthermore, we assume that GU n device has M_t independent computing tasks at the beginning of time slot t, denoted by the set $\mathbf{M} = \{1, 2, \ldots, M_t\}$. Considering a discrete time-slotted system with equal slot duration , denoted by $\mathbf{T} = \{1, 2, \ldots, T\}$.

Each GU n can determine whether or not to offload its computing task m to the edge server i, and $x_{nmi}(t) \in [0, 1]$ denote the offloading decision during the time slot t. Specifically, $N \times M_t \times (I + 1)$ matrix $\mathbf{X}(t)$ denote decisions of the tasks, $x_{nmi}(t) = 1$ denotes that GU n decides to offload its computing task m to the edge server i completely, and $x_{nmi}(t) = 0$ means that GU n disposes its task m locally. The following sections provide a comprehensive explanation of the computation and communication models.

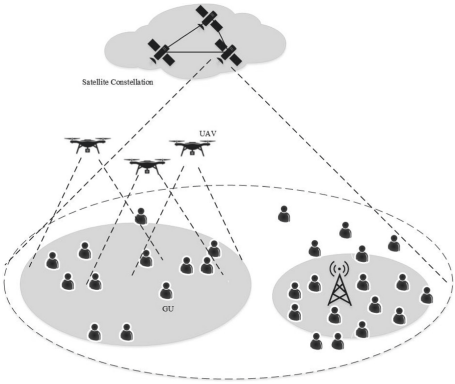

Fig. 1. The network model.

2.2 Computation Model

Without loss of generality, a tuple (ϕ, γ) is adopted to model the computing tasks from GU devices, where ϕ (in bits) represents the input data size of a computing task, and γ (in CPU cycles per bit) indicates that how many CPU cycles are required to process one bit input data [11]. The delay and energy consumption of downloading can be ignored when the computing results are transmitted back to the GUs by the edge server, because the key point of policy is task uploading in the considered scenario [19,20]. As the computation tasks can be completed locally, or offloaded to the UAVs or the LEO satellite constellation, the computing delay can be analyzed from the following aspects.

The computing capability (in CPU cycles per seconds) of servers mounted on UAVs and the satellite is denoted by f^i $(i \in \{1, 2, \ldots, I\})$ and f^0, respectively. The computing delay of GU n at all offloading destination SAGIN components is calculated as the following equation:

$$d_n^I(t) = \sum_{m=1}^{M_t} \sum_{i=0}^{I} \frac{x_{nmi}(t)\,\phi\gamma}{f^i}. \tag{1}$$

On the other hand, the delay in the local processing of a computation task consists of two parts, the computation delay and the queuing delay.Since the limited computing capability of the GU, the computation tasks may not processed or offloaded completely within a time slot. We assume that the remaining tasks wait to be processed in the computing queue. To model the queue latency, $\rho_n(t) \in [0, M_{\max}]$ denote the GU n's unaccomplished task backlog at the beginning of time slot t, $M_{\max}$ is the maximum length of the computing queue. $q_n(t)\tau$ is the queuing delay of all $q_n(t)$ tasks in the waiting queue of GU n. The number of queuing tasks denoted by:

$$q_n(t) = \max\left\{\rho_n(t) - \left\lfloor \frac{f_n\tau}{\phi\gamma} \right\rfloor - \sum_{m=1}^{M}\sum_{i=0}^{I} x_{nmi}(t), 0\right\}, \tag{2}$$

where $\lfloor\cdot\rfloor$ denotes the floor function, f_n is the computing capability of GU n, $\left\lfloor \frac{f_n\tau}{\phi\gamma} \right\rfloor$ denotes the greatest integer less than the number of computation tasks executed by GU n in time slot t. The delay of local task execution at the GU n can be given by:

$$d_n^l(t) = \frac{\sum\limits_{m,i}(1 - x_{nmi}(t))\phi\gamma}{f_n} + q_n(t)\tau. \tag{3}$$

Generally, the energy consumption of GU equipment is mainly composed of three parts, including mechanical energy consumption, communication-related energy consumption and computation-related energy consumption. The computation-related energy consumption can be calculated by:

$$e_n(t) = \xi_n \cdot \sum_{m,i}(1 - x_{nmi}(t))\phi\gamma(f_n)^2, \tag{4}$$

where ξ_n denotes the energy factor, which depends on the chip architecture [15,16].

2.3 Communication Model

Since UAVs and satellites use different frequency bands to communicate, we suppose that there is no interference between UAVs and satellite in this work [21]. Meanwhile, we neglect the propagation delay from GU devices to the UAV because we assume the UAV is sufficiently close to GU devices [22]. According to [23], the average path loss of GU n from UAV to devices can be defined as:

$$PL(r,h) = 20\log\left(\frac{4\pi f_c(h^2 + r^2)^{1/2}}{c}\right) + P_{LoS}\eta_{LOS} + (1 - P_{LoS})\eta_{NLOS}, \tag{5}$$

where h, r, η_{LOS}, η_{NLOS} denote the UAV flying altitude, horizontal distance between the UAV and the GU, the additive loss incurred on top of the free space pathloss for line-of-sight and not-line-of-sight links [24], respectively. We set the

altitude of the UAV to 10 m. f_c denotes the carrier frequency, c denotes the velocity of light, P_{LoS} represents the probability of line-of-sight link, which is an equation with respect to h, r [16]. According to [25], the values of $(\eta_{LoS}, \eta_{NLoS})$ are (0.1, 21) in remote area. Adopting the Weibull-based channel model [26], we generate the channel gain when $x_{nm0}(t) \neq 0$, which can be calculated by:

$$h = \frac{G_{tx} G_{rx} \lambda^2}{(4\pi l_{sat})} 10^{-\frac{F_{rain}}{10}}, \tag{6}$$

where G_{tx} and G_{rx} are antenna gains of the GU and the satellite, respectively. F_{rain} represents the rain attenuation, and l_{sat} denotes the distance between the GU and the satellite. Currently, the data rate denoted by $r_i(t)$ can be calculated as the following equations:

$$r_n^i(t) = \begin{cases} B_i \log_2 \left(1 + \frac{P_{n,i}(t) \cdot |h|^2}{\sigma_S^2}\right), i = 0 \\ B_i \log_2 \left(1 + \frac{P_{n,i}(t) \cdot 10^{-\frac{PL}{10}}}{\sigma_U^2}\right), i \neq 0, \end{cases} \tag{7}$$

where B_i indicate the channel bandwidth of the ground-satellite link and the ground-UAV link, indexes $1, 2, \ldots, I$ and 0 denote the UAVs and the LEO satellite constellation respectively. Analogously, $P_{n,i}$ represent the transmission power, σ_S and σ_U denote the power of noise. Thus, the transmission delay can be given by:

$$d_n^i(t) = \begin{cases} \sum_{m=1}^{M} \left(\lceil x_{nm0}(t) \rceil \, d_{sat} + \left(\frac{x_{nmi}(t)\phi}{r_n^i(t)} \right) \right), i = 0 \\ \sum_{m=1}^{M} \frac{x_{nmi}(t)\phi}{r_n^i(t)}, i \neq 0, \end{cases} \tag{8}$$

where we denote d_{sat} as the propagation delay between the LEO satellite and the GU, which cannot be ignored. $\lceil \cdot \rceil$ denotes the ceil function. Denote the communication-related energy consumption by $e_n^i(t)$, which is defined as follows:

$$e_n^i(t) = P_{n,i} d_n^i(t). \tag{9}$$

3 Problem Formulation and Algorithm

In this section, we first introduce the formulation for our optimization problem, and then the reinforcement learning-based approach is proposed to derive the near-optimum decision.

3.1 Problem Formulation

As described in the preceding section, taking communication and computing models into account, the delay and energy consumption for completing all tasks of GU n can be defined separately as follows:

$$D_n(t) = d_n^l(t) + d_n^e(t) + \sum_{i=0}^{I} d_n^i(t), \tag{10}$$

$$E_n(t) = e_n^l(t) + \sum_{i=0}^{I} e_n^i(t). \tag{11}$$

Finally, the cost function of the offloading decision can be defined as:

$$C_n(t) = \omega_1 D_n(t) + \omega_2 E_n(t), \tag{12}$$

where ω_1 and ω_2 denote the tradeoff between delay and energy consumption for the dynamic computing offloading policy, $\omega_1 = 1-\omega_2$. Let $\mathbf{X} = \{X_t, \forall t\}$ denote the tasks offloading decisions set. Since link availability and task arrival are highly dynamic, our main focus is to minimize the time-averaged delay and energy consumption for all tasks. As the number of users increases, the cost of the MEC server to collect the channel vector of all users and then distribute the task queue to each user increases. In order to make the system much more scalable, we assume that the state of each GU is only determined by its local observation and make decisions independently. The optimization problem is defined as:

$$\begin{aligned} \min_{X} \ & \lim_{T\to\infty} \tfrac{1}{T} \sum_{t=1}^{T} \omega_1 D_n(t) + \omega_2 E_n(t) \\ s.t. \quad & x_{nmi}(\mathrm{t}) \in [0,1], \forall n \in \mathbf{N}, \forall m \in \mathbf{M}, \forall i \in \mathbf{I}, \\ & \sum_{i=0}^{I} x_{nmi}(t) \in \{0,1\}, \forall n \in \mathbf{N}, \forall m \in \mathbf{M}, \\ & \sum_{m=0}^{M_t} \sum_{i=0}^{I} x_{nmi}(t) \leq M_{\max}, \forall n \in \mathbf{N}. \end{aligned} \tag{13}$$

3.2 Deep Deterministic Policy Gradient Algorithm

Above optimization problem P1 is a high-dimensional decision issue. We adopt an intelligent learning approach based on reinforcement learning (RL) to address this problem. RL algorithms optimize the action choosing behavior by massive interaction between agent and environment [12]. Compared with traditional optimization approach, deep Q-learning network (DQN) estimate the state-action value by deep neural network. Although problems in high-dimensional state spaces have been successfully solved by DQN, DDPG has been proposed to extend DRL algorithms to continuous action space [27]. Then, the state space, action space, reward function and the environment are introduced briefly in this section.

State Space: At the start of time slot t, $s_t^n = \left\{h_t^n, \phi_n^{CPR}, \rho_t^n, E_t^n\right\}$ denotes the network state, where $h_t^n, \phi_n^{CPR}, \rho_t^n, E_t^n$ represent the channel vectors, the number of offloaded tasks, the unaccomplished task in the queue and the energy consumption, respectively.

Action Space: Based on the current state s_t^n, the learning system needs to decide which access point should be selected and take action of scheduling the tasks of GU n. Let the vector $a_t^n = \{x_{nmi}(t), \forall n \in \mathbf{N}, \forall m \in \mathbf{M}, \forall i \in \mathbf{I}\}$ denotes the action space, where $x_{nmi}(t) \in [0, 1]$ indicates that user n whether partially offload the task m to the MEC server i or not.

Reward Function and Policy: With the objective of long-term weighted sum of delay and energy consumption of all tasks, the reward function can be defined as $R_n(s_t^n) = \mathbb{E}\left[\lim_{T\to\infty} \frac{1}{T} \sum_{y=t}^{T} C_n(y)|s_t^n\right]$, Denote by π the stationary policy, and a value function is defined to determine the value of reward when the system state is s_n, which is defined as:

$$V_\pi(s^n) = \mathbb{E}\left[\sum_{t=0}^{\infty} \psi R_n(s_t^n, a_t^n)\,|\pi, s_0^n = s^n\right], \tag{14}$$

where $\psi \in [0, 1]$ denotes the discounting factor. After confirming the state space, action space and reward function, a DDPG based algorithm is proposed for this Markov decision process (MDP) problem, as shown in Algorithm 1. Generally, an experienced replay buffer $\mathbb{B}$ is denoted, which stores experiences and mini-batches of experience. In Algorithm 1, mini-batches of samples $(s, a, R, s') \sim U(\mathbb{B})$ will be drawn uniformly at random from $\mathbb{B}$. Based on temporal-difference learning, a combination of Monte Carlo method and dynamic programming, the Q-value can be updated as follows:

$$Q'_\pi(s_t^n, a_t^n) = Q_\pi(s_t^n, a_t^n) + \alpha\left[R_n(s_t^n, a_t^n) + \psi Q_\pi(s_t^n, a^*)\right], \tag{15}$$

where α denotes the learning rate and $a^* = \arg\max_{a_{n(t)}} Q_\pi(s_n^t, a_n(t))$ denotes the greedy action. The following loss function can be calculated by:

$$L(\theta) = \mathbb{E}_{(s,a,R,s')\sim U(\mathbb{B})}\left[\left(R_t^n + \psi Q_\pi(s_{t+1}^n, a^*|\theta) - Q_\pi(s_t^n, a_t^n|\theta)\right)^2\right], \tag{16}$$

An actor-critic approach is adopted in DDPG algorithm, we leverage two separate DNNs to approximate Q-value network $Q\left(s, a; \theta^Q\right)$, the actor $\mu(s|\theta^\mu)$. The policy gradient of the θ^μ can be calculated as follows:

$$\nabla_{\theta^\mu} J \approx \mathbb{E}_{(s,a,R,s')\sim U(\mathbb{B})}\left[\left(\nabla_a Q(s, a|\theta^Q)\nabla_{\theta^\mu}\mu(s|\theta^\mu)\right)\right] \tag{17}$$

4 Performance Evaluation

In this section, simulation is carried out to verify the proposed model and algorithm. Specifically, we begin by elaborating on the simulation settings. Afterwards, we present an evaluation on the experiment results.

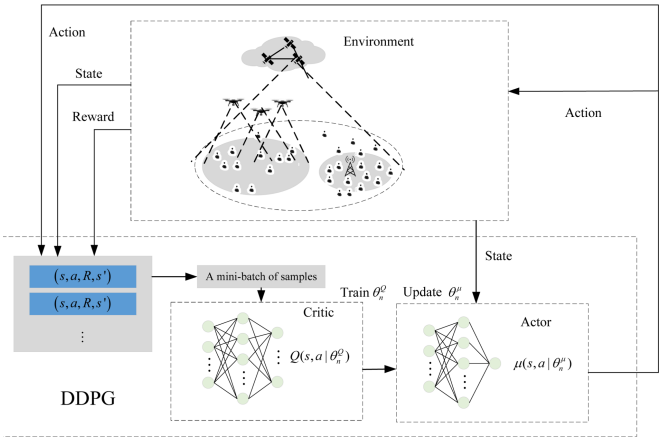

Fig. 2. The DDPG-based computation offloading scheme.

4.1 Simulation Settings

As shown in Fig. 2, the proposed approach is implemented by one actor network, one critic network and a replay buffer. Simulation environment is implemented via Python 3.6 and Tensorflow library. The DNNs' training and testing are conducted with a personal computer with AMD R7-4800H CPU. ReLU function is used as the activation function after the fully connected layer and L2 regularization is used to reduce DNN over-fitting. The number of neurons in the two hidden layers are 300 and 400, and we set 2000 and 0.001 to the number of episode and learning rate. Other important constant parameters are listed in the Table 1.

Table 1. Simulation parameters.

Parameter	Value	Parameter	Value
N	5	I	5
$f^i, i \neq 0$	5 GC/s	ϕ	5 MB
f^0	10 GC/s	γ	25 cycles/bit
f_n	200 MC/s	N_0	-100 dBm/Hz
B_U	3 MHz	P_U	1.6 W
B_S	2 MHz	P_S	5 W
d_{sat}	6.44 ms	$M_{\max}$	20

Algorithm 1: DDPG Based Computation Offloading

Initialization:
for *each GU agent* $n \in \mathbf{N}$ **do**
  Randomly initialize critic network $Q(s, a|\theta_n^Q)$ and actor $\mu(s, a|\theta_n^\mu)$ with weights θ_n^Q and θ_n^μ;
  Initialize target network Q' and μ' with weights $\theta_n^{Q'} \leftarrow \theta_n^Q$ and $\theta_n^{\mu'} \leftarrow \theta_n^\mu$;
  Initialize replay buffer $\mathbb{B}$;
end
for *episode* $k = 1, 2, \ldots, K$ **do**
  Reset simulation parameters for the environment;
  Receive initial observation state $s_{n,1}$ for GU $n \in \mathbf{N}$;
  for *time slot* $t = 1, 2, \ldots, T$ **do**
    for *GU* $n \in \mathbf{N}$ **do**
      Select action $a_t^n = \mu(s_t^n|\theta_n^\mu) + \Delta\mu$ according to the current policy and exploration noise $\Delta\mu$;
      Execute action a_t^n and observe the reward R_t^n and the next state s_{t+1}^n;
      Store transition $\left(s_t^n, a_t^n, R_t^n, s_{t+1}^n\right)$ in $\mathbb{B}$;
      Sample random mini-batch of transitions $\{(s_z^n, a_z^n, R_z^n, s_z^n)\}_{z=1}^{Z}$ from $\mathbb{B}$;
      Set $y_z = R_z^n + \psi Q'\left(s_{z+1}^n, \mu'(s|\theta_n^{\mu'})|\theta_n^{Q'}\right)$;
      Update the critic network $Q(s, a|\theta_n^Q)$ by minimize the loss $L = \frac{1}{Z}\sum_{z=1}^{Z}\left(\left(y_z - Q_\pi(s_z^n, a_z^n|\theta_n^Q)\right)^2\right)$;
      Update the actor policy by using the sampled policy gradient $\nabla_{\theta_n^\mu} J \approx \frac{1}{Z}\sum_{z=1}^{Z}\left(\left(\nabla_a Q(s_z, a|\theta_n^Q)|_{a=a_z}\nabla_{\theta_n^\mu}\mu(s_z|\theta_n^\mu)\right)\right)$;
      Update the target networks: $\theta_n^{\mu'} \leftarrow \delta\theta_n^\mu + (1-\delta)\theta_n^{\mu'}$ and $\theta_n^{Q'} \leftarrow \delta\theta_n^Q + (1-\delta)\theta_n^{Q'}$;
    end
  end
end

4.2 Simulation Results

Firstly, we show the simulation results of the proposed algorithm, and evaluate the convergence performance. Figure 3 shows the convergence performance with respect to the average reward of total tasks presented by setting $\omega_1 = 0.8$, $\omega_1 = 0.5$ and $\omega_1 = 0.2$, and the results are averaged from ten numerical simulations, proving the effectiveness of neural networks.

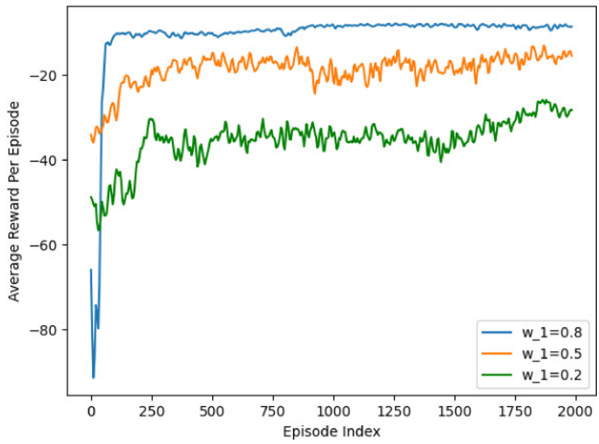

Fig. 3. The convergence of the proposed DDPG algorithm.

Meanwhile, it can be observed that the performance of the partial offloading policy learned from proposed algorithm is always better than the binary offloading policy learned from DQN for different scenarios by Fig. 4.

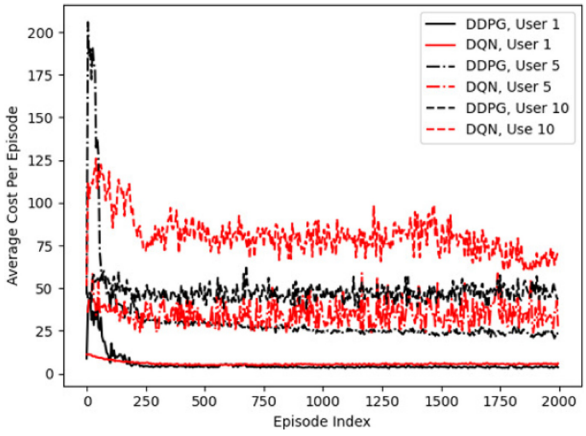

Fig. 4. Illustration of the average cost per episode.

To evaluate the computation offloading validity of the proposed DDPG scheme on SAGIN system, we adopt the other four benchmark schemes which are introduced as follows:

Greedy Local Execution (GLE): For each slot, the computation tasks will be processed locally ($x_{nmi}(t) = 0$) as many as possible.

Greedy Computation Offloading (GCO): Each GU firstly makes its best effort to offload computation tasks, and then the remaining tasks will be processed locally.

DQN based Dynamic Offloading (DQN): As shown in Fig. 5, the DQN is also implemented for the dynamic computation offloading problem, $\varepsilon-$greedy selection and Adam method are adopted for training. In the binary offloading scheme, there are only two cases with this solution: complete offloading or local processing ($x_{nmi}(t) \in \{0, 1\}$).

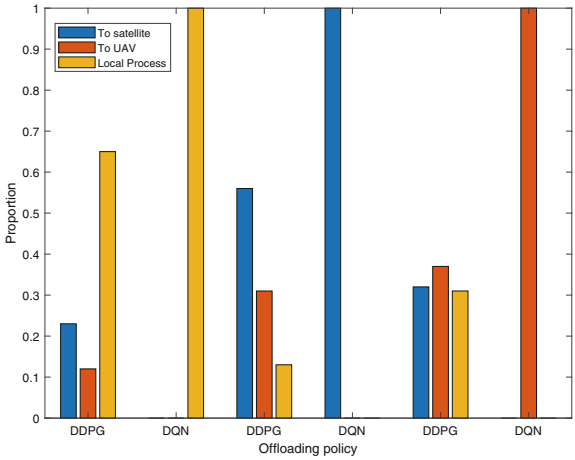

Fig. 5. Offloading proportion under different policies.

Figure 6 and Fig. 7 represent the sum cost of processing the tasks with respect to the numbers of users and average data size, respectively. The total cost of DQN, GCO and proposed DDPG schemes are all increasing as the average data size increasing. The time delay and energy consumption brought by the communication process will be more because of larger data size, so that the total cost of GCO scheme is close to GLE. However, the proposed DDPG scheme can still keep the total cost lower than DQN scheme, proving the validity of partial offloading strategy.

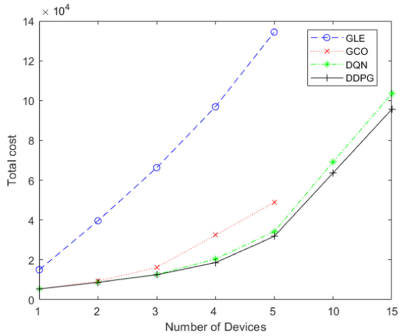

Fig. 6. Total cost vs. numbers of GU devices.

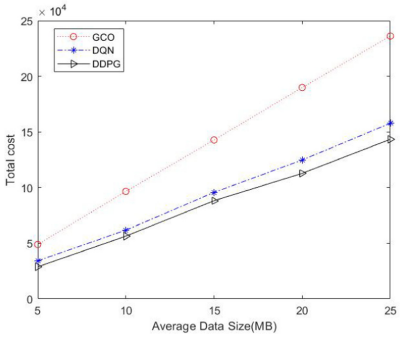

Fig. 7. Total cost versus average data size.

5 Conclusion

In this paper, an efficient computing offloading scheme is proposed for the MEC system in SAGIN. Firstly, we elaborated the SAGIN architecture. Then, we express computation offloading as a nonlinear optimization problem with the goal of minimizing the weighted sum of energy consumption and delay. On this basis, we propose an algorithm based on DDPG to solve this problem. Finally, the simulation results show that the energy consumption and time can be significantly saved by offloading the task to the edge server on the UAV or satellite, and the convergence performance and effectiveness of the proposed scheme in the simplified scenario are also proved.

In the future, the mobility management of satellites and UAVs will be further considered. In addition, the offloading scheme of SAGIN in areas with rich computing resources is also worth further study.

Ackonwledgment. This work is supported by the National Natural Science Foundation of China under grant (61761014), Guangxi Natural Science Foundation (2018GXNSFBA281131), Ministry of Education Key Laboratory of Cognitive Radio and Information Processing (CRKL190109).

References

1. Zhang, Z., et al.: 6G wireless networks: vision, requirements, architecture, and key technologies. IEEE Veh. Technol. Mag. **14**(3), 28–41 (2019)
2. Liu, J., Shi, Y., Fadlullah, Z.M., Kato, N.: Space-air-ground integrated network: a survey. IEEE Commun. Surv. Tutor. **20**(4), 2714–2741 (2018)
3. Jiang, W., Han, B., Habibi, M.A., Schotten, H.D.: The road towards 6G: a comprehensive survey. IEEE Open J. Commun. Soc. **2**, 334–366 (2021)
4. Abbas, N., Zhang, Y., Taherkordi, A., Skeie, T.: Mobile edge computing: a survey. IEEE Internet Things J. **5**(1), 450–465 (2018)
5. Khayyat, M., et al.: Multilevel service-provisioning-based autonomous vehicle applications. Sustainability **12**, 2497 (2020)
6. Mach, P., Becvar, Z.: Mobile edge computing: a survey on architecture and computation offloading. IEEE Commun. Surv. Tutor.. **19**(3), 1628–1656 (2017)
7. Lyu, X., et al.: Optimal schedule of mobile edge computing for internet of things using partial information. IEEE J. Sel. Areas Commun. **35**(11), 2606–2615 (2017)
8. Min, M., Xiao, L., Chen, Y., Cheng, P., Wu, D., Zhuang, W.: Learning-based computation offloading for IoT devices with energy harvesting. IEEE Trans. Veh. Technol. **68**(2), 1930–1941 (2019)
9. Elgendy, I.A., Zhang, W.-Z., He, H., Gupta, B.B., Abd El-Latif, A.A.: Joint computation offloading and task caching for multi-user and multi-task MEC systems: reinforcement learning-based algorithms. Wirel. Netw. **27**(3), 2023–2038 (2021). https://doi.org/10.1007/s11276-021-02554-w
10. Chen, Z., Wang, X.: Decentralized computation offloading for multi-user mobile edge computing: a deep reinforcement learning approach. EURASIP J. Wirel. Commun. Netw. **2020**(1), 1–21 (2020). https://doi.org/10.1186/s13638-020-01801-6
11. Zhou, C., et al.: Delay-aware IoT task scheduling in space-air-ground integrated network. In: 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 2019, pp. 1–6. IEEE (2019)
12. Xu, F., Yang, F., Zhao, C., Wu, S.: Deep reinforcement learning based joint edge resource management in maritime network. China Commun. **17**(5), 211–222 (2020)
13. Zhou, C., et al.: Deep reinforcement learning for delay-oriented IoT task scheduling in space-air-ground integrated network. IEEE Trans. Wirel. Commun. **20**(2), 911–925 (2021)
14. Dinh, T.H., Niyato, D., Hung, N.T.: Optimal energy allocation policy for wireless networks in the sky. In: 2015 IEEE International Conference on Communications (ICC), London, UK, 2015, pp. 3204–3209. IEEE (2015)
15. Tang, Q., Fei, Z., Li, B., Han, Z.: Computation offloading in LEO satellite networks with hybrid cloud and edge computing. IEEE Internet Things J. **11**, 9164–9176 (2021)
16. Cheng, N., et al.: Space/Aerial-assisted computing offloading for IoT applications: a learning-based approach. IEEE J. Sel. Areas Commun. **37**(5), 1117–1129 (2019)
17. Saleem, U., Liu, Y., Jangsher, S., Li, Y.: Performance guaranteed partial offloading for mobile edge computing. In: GLOBECOM (2018)

18. Wu, Q., Zeng, Y., Zhang, R.: Joint trajectory and communication design for multi-UAV enabled wireless networks. IEEE Trans. Wirel. Commun. **17**(3), 2109–2121 (2018)
19. Bi, S., Zhang, Y.: Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading. IEEE Trans. Wirel. Commun. **17**(6), 4177–4190 (2018)
20. Guo, S., Xiao, B., Yang, Y., Yang, Y.: Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing. In: IEEE International Conference on Computer Communications, pp. 1–9 (2016)
21. Zhang, N., Liang, H., Cheng, N., Tang, Y., Mark, J.W., Shen, X.S.: Dynamic spectrum access in multi-channel cognitive radio networks. IEEE J. Sel. Areas Commun. **32**(11), 2053–2064 (2014)
22. Hosseini, N., Jamal, H., Haque, J., Magesacher, T., Matolak, D.W.: UAV command and control, navigation and surveillance: a review of potential 5G and satellite systems. In: IEEE Aerospace Conference, pp. 1–10 (2019)
23. Shi, W., et al.: Multiple drone-cell deployment analyses and optimization in drone assisted radio access networks. IEEE Access **6**, 12518–1252 (2018)
24. Al-Hourani, A., Kandeepan, S., Lardner, S.: Optimal LAP altitude for maximum coverage. IEEE Wirel. Commun. Lett. **3**(6), 569–572 (2014)
25. Bor-Yaliniz, R.I., El-Keyi, A., Yanikomeroglu, H.: Efficient 3-D placement of an aerial base station in next generation cellular networks. In: IEEE ICC, pp. 1–5 (2016)
26. Kanellopoulos, S.A., Kourogiorgas, C.I., Panagopoulos, A.D., Livieratos, S.N., Chatzarakis, G.E.: Channel model for satellite communication links above 10GHz based on Weibull distribution. IEEE Commun. Lett. **18**(4), 568–571 (2014)
27. Lillicrap, T.P., et al.: Continuous control with deep reinforcement learning. Computer Science (2015)