Digital Twin Assisted Deep Reinforcement Learning for Computation Offloading in UAV Systems

Kai Liang, Yawei Wang, Zan Li, *Senior Member, IEEE*, Gan Zheng *Fellow, IEEE*, Kai-Kit Wong *Fellow, IEEE* and Chan-Byoung Chae *Fellow, IEEE*

***Abstract*—The emerging digital twin technology has the ability to set up a digital clone of the physical world through techniques such as computer simulation and machine learning, facilitating management and decision-making of the physical world. This paper proposes a DT assisted deep reinforcement learning (DRL) approach to optimize the decision of computation offloading in a unmanned aerial vehicle (UAV) based communication system. We focus on minimizing the task processing delay by jointly optimizing the task offloading and the UAV's flight path. This problem can be solved by the DRL method, but it requires frequent interactions with the environment, which may increase resource consumption and encounter risks from some "bad" actions. In view of this, the proposed DT is made up of a predictive model to predict the reward and next stage of the physical environment, and a generative model to improve the sample efficiency, where these two models are realized by a fully connected neural network and a variational auto-encoder, respectively. Then, a hybrid experience reply buffer can be generated to facilitate DRL training, thus yielding faster convergence, better performance, and fewer environmental interactions. Numerical results show that the proposed method can achieve better delay performance and training efficiency than state of the art methods.**

***Index Terms*—Digital twin, mobile edge computing, deep reinforcement learning, generative and predictive models**

I. Introduction

MOBILE edge computing (MEC) can provide computational capabilities at the network edge, enabling rapid and efficient processing of data and content in proximity to end-users [1]. Meanwhile, fast-moving unmanned aerial vehicles (UAVs) are easier to deploy on-demand to places where wireless network computing resources are insufficient. This motives the UAV to extends the reach and capabilities of MEC systems [2]. Traditionally, iterative optimization algorithms are used to optimize the resource allocation and computation offloading of UAV-assisted MEC, but they suffer from high computational complexity and slow convergence that cannot satisfy delay-sensitive or computation-intensive applications. Fortunately, the blossoming of deep reinforcement learning(DRL) provides an efficient approach for UAV-assisted MEC which has been widely studied [3], [4]. Although the DRL approach exhibits performance improvement, it often requires frequent interactions with the physical environment to obtain the reward value of the current action and the next state, leading to more resource consumption and potential risk to the system arisen from "bad" actions [5].

The emerging Digital twin (DT) can set up a virtual counterpart to real-world systems and thus own the ability of low-cost trial and error, which encourages the combination of DT and MEC system. The works [6], [7] studied the computation offloading problem in DT-assisted MEC, and only focused on the system model of digital twin as a single copy of the physical edge network. Nevertheless, how to utilize DT to assist the training of DRL (e.g., reducing the interactions and improving the performance) remains a challenge. In [8], Wu *et al.* proposed a DT model in autonomous driving scenario, which improves the data efficiency of DRL by establishing a digital model to predict the transition dynamics of the driving scene represented as images. This motives us to use the DT to mimic the system transitions that offers improved performance. However, the method in [8] obtains states and corresponding actions from the physical environment, and a large amount of data requires frequent interaction between the agent and the physical environment.

In this paper, we formulate the task offloading optimization problem in a UAV system with the aim of minimizing task processing delay through the joint optimizing task offloading and the UAV's flight path. After recasting this non-convex problem into a Markov decision process (MDP), we propose a DT-assisted deep deterministic policy gradient (DDPG) approach based on generative and predictive models to solve this problem. The main contributions of this paper are summarized as follows: (i) We propose a DT-assisted DDPG approach, where the DT is comprised of a generative model and a predictive model. To be specific, the predictive model is realized by a fully connected neural network (FCNN) and can continuously fit the environment to predict the state transition dynamics and reward feedback of the environment. To address small size of sample sets, we further use a variational auto-encoder (VAE) network to form the generative model which is responsible for generating simulated samples and thus improve the performance of the predictive model. The DDPG aided by the proposed DT can combine the interaction between the physical environment and the DT to improve the training efficiency and reduce the number of interactions with the physical world. (ii) We apply this approach to the UAV based MEC system for solving optimization problem. Numerical results show that our proposed approach achieves improved processing latency and training efficiency of DRL.

Kai Liang, Yawei Wang, and Zan Li (corresponding author) are with the School of Telecommunications Engineering, Xidian University, Xi'an, 710071, China (email: kliang@xidian.edu.cn, yw.wang@stu.xidian.edu.cn, zanli@xidian.edu.cn). G. Zheng is with the School of Engineering, University of Warwick, Coventry, CV4 7AL, UK (Email: gan.zheng@warwick.ac.uk). K.-K. Wong is with the Department of Electronic and Electrical Engineering, University College London, London, WC1E 6BT, UK (Email: kai-kit.wong@ucl.ac.uk). Chan-Byoung Chae is with School of Integrated Technology, Yonsei University, Seoul, 03722, Korea (email: cbchae@yonsei.ac.kr).

II. System Model and Problem Formulation

The system comprises M mobile terminals (MTs) and a UAV equipped with a MEC server. Denote the MT set as $\mathcal{M} = \{1, 2, \cdots, M\}$. The UAV has the capability to provide communication and computing services and serves M MTs simultaneously. Each MT has limited computation capacity to handle computationally intensive tasks, so it offloads a portion of the tasks to the MEC server through the wireless channel to reduce the task processing delay. The entire time period T of UAV operation is evenly divided into I time slots and we denote the set $\mathcal{I} = \{1, 2, \cdots, I\}$. In three-dimensional space, we assume that the UAV keeps flying at a fixed altitude H, and it has the position coordinate $\mathbf{u}(i) = [x(i), y(i)]^T$ at time slot $i \in \mathcal{I}$. The position coordinate of MT $m \in \mathcal{M}$ is $\mathbf{p}_m(i) = [x_m(i), y_m(i)]^T$. We assume that MTs move slowly and randomly within a rectangular area of length L and width W, and the UAV flies for a fixed period of time (i.e., t_{fly} which is less than the duration of time slot) at the beginning of each time slot, hovering at a fixed position for the remaining time to serve the MTs. We denote that the position of the UAV at the $(i-1)$-th time slot is $\mathbf{u}(i-1)$, and at i-th time slot, the new position of the UAV can be expressed as:

$$\begin{aligned}\mathbf{u}(i) = [&x(i-1) + v(i)t_{fly}cos\theta(i), \\ &y(i-1) + v(i)t_{fly}sin\theta(i)]^T,\end{aligned} \tag{1}$$

where $v(i) \in [0, v_{max}]$ denotes UAV flight speed, $\theta(i) \in [0, 2\pi]$ denotes UAV flight angle.

Similar to [9] and [10], the communication links between the UAV and the MTs are presumed to be dominated by the line of sight (LoS) paths. The channel gain between the UAV and the MT m is denoted as :

$$g_m(i) = \alpha_0 d_m^{-2}(i) = \frac{\alpha_0}{||\mathbf{u}(i) - \mathbf{p}_m(i)||^2 + H^2}, \tag{2}$$

where α_0 denotes the channel gain when the reference distance $d = 1$ meter, and $d_m(i)$ denotes the Euclidean distance between the UAV and the MT m at time slot i. $\|\cdot\|$ denotes the Euclidean norm operator. For the sake of simplicity, we assume the overall bandwidth is evenly allocated to each MT, and denote the bandwidth of each MT as B. The wireless transmission rate can be given as:

$$R_m(i) = B\log_2\left(1 + \frac{P_{up}g_m(i)}{\sigma^2}\right), \tag{3}$$

where P_{up} and σ^2 respectively denote transmit power of the MT and noise power. In our system, a partial offloading policy is implemented for the tasks of MTs during each time slot [2]. To be specific, let $\lambda_m(i) \in [0, 1]$ denotes the ratio of tasks offloaded to the MEC server, and $(1 - \lambda_m(i))$ indicates the ratio of tasks running locally in MT m. Then the local task execution delay of the MT m can be given as:

$$t_{lo,m}(i) = \frac{(1 - \lambda_m(i))D_m(i)k}{f_{MT}}, \tag{4}$$

where $D_m(i)$ denotes the computing task data sizes of MT m, k denotes the CPU cycles required to process each unit bit, and f_{MT} denotes the MT's computing capability.

The processing delay of the MEC server can be divided into two parts, one is the transmission delay and the other is the computation delay. In this system, the size of the computation results provided by the MEC server is usually small enough to be negligible [2], so the transmission delay of the downlink is not considered here. The uplink transmission delay for MT m with the UAV can be given as:

$$t_{up,m}(i) = \frac{\lambda_m(i)D_m(i)}{R_m(i)}, \tag{5}$$

The delay resulting from the computation at the MEC server can be given as:

$$t_{UAV,m}(i) = \frac{\lambda_m(i)D_m(i)k}{f_{UAV}/M}, \tag{6}$$

where f_{UAV} denotes the computation frequency of MEC server. The computational resource is equally shared among all MTs. k denotes the CPU cycles required to process each unit bit.

This paper focuses on minimizing the sum of processing delay across all MTs as in [10], [11] by jointly optimizing the UAV flight path and offloading allocation strategies. Specifically, the optimization problem is formulated as follows:

$$\min_{\mathbf{u},\lambda} \quad \sum_{i=1}^{I}\sum_{m=1}^{M} \max\{t_{lo,m}(i), t_{up,m}(i) + t_{UAV,m}(i)\} \tag{7}$$

$$\text{s.t.} \quad 0 \leq \lambda_m(i) \leq 1, \forall i, m, \tag{7a}$$

$$\mathbf{u}(i) \in \{(x(i), y(i)) \mid x(i) \in [0, L], y(i) \in [0, W]\}, \forall i, \tag{7b}$$

$$\begin{aligned}\mathbf{p}_m(i) \in \{&(x_m(i), y_m(i)) \mid x_m(i) \in [0, L], \\ &y_m(i) \in [0, W]\}, \forall i, m,\end{aligned} \tag{7c}$$

$$\sum_{i=1}^{I}\sum_{m=1}^{M} D_m(i) = D, \tag{7d}$$

where the constraint (7a) denotes the offloading ratio of the computing task. Constraints (7b) and (7c) limit that UAV and MT can only move in the given area. The constraint (7d) specifies all of computing tasks to be completed in the whole time period T.

III. DT Assisted DDPG For Task Offloading

In addressing the optimization problem (7), traditional optimization algorithms face challenges such as exceedingly high computational complexity or difficulty in solving due to the dynamic nature of the system and the heterogeneity of tasks generated by MTs. Conversely, DRL-based approaches have been proved effective in tackling these challenges. Therefore, we adopt the DDPG algorithm to obtain effective strategies to solve this problem. However, note that during the learning process of the agent, a substantial amount of training data is needed, which requires extensive interaction with the environment and a large number of training iterations to achieve convergence. To accelerate agent training, we propose a DT assisted DDPG method facilitated by the generative and predictive models, as shown in Fig. 1, which will be detailed in the following.

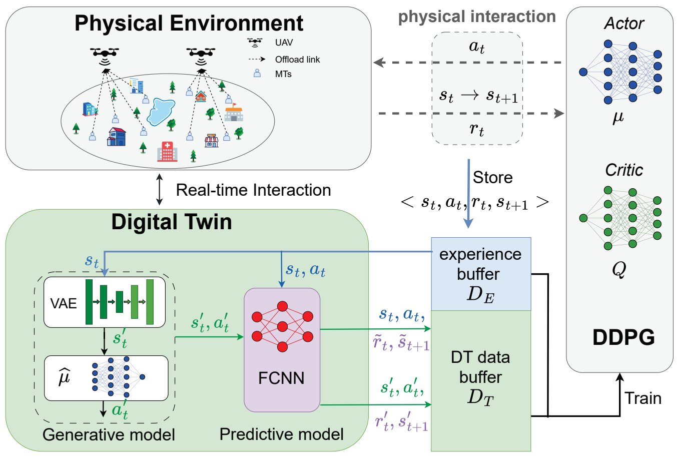

Fig. 1. DT-assisted DDPG method.

A. The DDPG Module

First, we present the formulated problem as a Markov decision process (MDP), which consists of the following key elements:

State: The system state at time slot i can be defined as: $s_i = \{\mathbf{u}(i), \mathbf{p}_1(i), ..., \mathbf{p}_M(i), D_r(i), D_1(i), ..., D_M(i)\}$, where $D_r(i)$ denotes the size of remaining tasks that the system needs to complete in the whole time period T, and $D_m(i)$ denotes the task size randomly generated by MT m at time slot i.

Action: The state changes depending on which actions are taken. The action space consists of the UAV's flight speed, the UAV's flight angle, and the task offloading ratio. Specifically, at time slot i, the action can be expressed as: $a_i = \{v(i), \theta(i), \lambda_1(i), ..., \lambda_M(i)\}$, where $v(i) \in [0, v_{max}]$, $\theta(i) \in [0, 2\pi]$, and $\lambda_m(i) \in [0, 1]$.

Reward: The agent is able to perceive the current system state, take each possible action to get a new state and receive a reward. Our goal is to minimize the processing delay defined in problem (7), so the reward is negatively correlated with the processing delay. Thus, the reward can be defined as: $r_i = -\sum_{m=1}^{M} \max\{t_{lo,m}(i), t_{up,m}(i) + t_{UAV,m}(i)\}$.

Considering the continuity of the action space, we can use the DDPG algorithm to obtain the effective strategy. DDPG combines two neural networks, the actor network $\mu(s|\theta^\mu)$, and the critic network $Q\left(s, a|\theta^Q\right)$. In addition, both the actor network and the critic network contain a target network with the same structure as them. The actor network learns a deterministic policy, meaning for a given state s_t, and directly outputs a specific action:

$$a_t = \mu\left(s_t|\theta^\mu\right) + \mathcal{N}_t, \tag{8}$$

where θ^μ is the actor network parameter and $\mathcal{N}_t$ is the exploration noise that follows a normal distribution.

Besides, DDPG is an off-policy algorithm that leverages experience replay, which means that it learns from a replay buffer $\mathcal{D}_E$ of past experiences. This buffer stores the history of state, action, reward, and next state $\langle s_t, a_t, r_t, s_{t+1}\rangle$. During training, it samples mini-batches of size B from $\mathcal{D}_E$. The critic network uses Q-function to evaluate the selected action a_t and updates its parameter θ^Q by minimizing the loss function:

$$L = \frac{1}{B}\sum_i \left(y_i - Q\left(s_i, a_i|\theta^Q\right)\right)^2, \tag{9}$$

where $y_i = r_i + \gamma Q'\left(s_{i+1}, \mu'\left(s_{i+1}|\theta^{\mu'}\right)|\theta^{Q'}\right)$, γ is the discount factor, $\theta^{\mu'}$ and $\theta^{Q'}$ are the parameters of the target actor and critic network, respectively.

B. The DT Module

In the traditional DDPG algorithm, the experience buffer records the interaction data from the physical environment. The frequent interactions between agent and environment during the training process result in significant resource consumption and potential risks. Furthermore, we introduce generative and predictive models into the DT, and establish an additional buffer for storing virtual data, enabling the agent to learn from mixed data and improving the interaction efficiency.

In DT, our predictive model can predict state transition dynamics and reward feedback by learning data from the physical world. Specifically, at time step t, the model $\mathcal{M}_P$ inputs state s_t and action a_t, while outputs next state s'_{t+1} and reward feedback r'_t, denotes as:

$$s'_{t+1}, r'_t = \mathcal{M}_P(s_t, a_t). \tag{10}$$

The predictive model is established by a deep neural network, which we implement using fully connected neural network (FCNN). This model uses a supervised learning approach and the accuracy of the prediction is continuously improved based on the agent's interaction data with the physical environment. When training the predictive model, we use interaction data $\langle s_t, a_t, r_t, s_{t+1}\rangle$ in the experience buffer as training data. The mean squared error (MSE) loss is adopted for the model, expressed as:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\left\|\left(s'_{t+1}, r'_t\right) - \left(s_{t+1}, r_t\right)\right\|^2 \tag{11}$$

where N denotes the number of data in each training batch, s'_{t+1} and r'_t represent the predicted next state and reward feedback by the model, while s_{t+1} and r_t are the actual next state and reward feedback.

In predictive network, we obtain virtual interaction data based on states and corresponding actions. However, state data is only sampled from the data obtained through interactions between the agent and the physical environment. These state data only reflect the historical environmental information of the system, making them relatively limited in scope. Therefore, in DT, we introduce the generative model to learn the probability distribution of the data, enabling the generation of diverse and forward-looking state data. We use a VAE network to form the generative model. The state s_t in the experience buffer $\mathcal{D}_E$ is used as input data to the VAE. The VAE network outputs the generated state s'_t. In addition, we create a digital copy of the Actor network of the DDPG, i.e. $\hat{\mu}\left(s|\theta^{\hat{\mu}}\right)$. It uses the generated s'_t to give the predict action a'_t.

The VAE [12] structure consists of two parts: the encoder and the decoder. The encoder is responsible for extracting essential features from the input data and generating the encoding: mean vector and deviation vector. Then the encoding is used as an input to the decoder. The decoder outputs the data similar to the original data. The loss function of the VAE model is expressed as:

$$\mathcal{L}(\theta,\phi) = -\mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] + \mathrm{KL}(q_\phi(z|x)||p(z)), \tag{12}$$

where x is the input data point. z is the latent variable, which represents the distribution parameters learned by the encoder from the data point x. $q_\phi(z|x)$ is the output of the encoder, representing the distribution of the latent variable z conditioned on the input x. $p_\theta(x|z)$ is the output of the decoder, representing the distribution of the generated data point x conditioned on the latent variable z. $\mathrm{KL}(q_\phi(z|x)||p(z))$ is the Kullback-Leibler (KL) divergence, which measures the difference between the encoder's output $q_\phi(z|x)$ and the prior distribution $p(z)$.

C. *Training Strategy*

The tuples $\langle s_t, a_t, r_t, s_{t+1}\rangle$, derived from interactions with the physical environment, are stored in the experience buffer $\mathcal{D}_E$. These data serve as the training set, fueling the training process of the model in DT. The VAE network in the generative model learns the distribution and latent representation of state data s_t, and then outputs diverse and forward-looking state s'_t which is fed into the digital copy of the Actor network $\hat{\mu}\left(s|\theta^{\hat{\mu}}\right)$ to obtain action a'_t. The input data for the FCNN in the predictive model includes not only the states and corresponding actions $\langle s_t, a_t\rangle$ from the physical environment, but also the virtual states and actions $\langle s'_t, a'_t\rangle$ based on the generative model. This makes the input data diverse and capable of modeling possible future scenarios. The FCNN will predict state transition dynamics and reward feedback.

Besides, we build an additional DT data buffer $\mathcal{D}_T$ to store data from digital interactions. The digital interaction data generated by predictive model includes $\langle s_t, a_t, \tilde{r}_t, \tilde{s}_{t+1}\rangle$ and $\langle s'_t, a'_t, r'_t, s'_{t+1}\rangle$. During DRL learning, we sample a mini-batch experiences from both the experience buffer $\mathcal{D}_E$ and the DT data buffer $\mathcal{D}_T$ with a certain ratio, e.g., data from $\mathcal{D}_T$ accounts for 0.35 of the mixed data. By combining interactions between the physical environment and the digital simulated environment, the DDPG training process can be significantly optimized in terms of time efficiency, simultaneously reducing the number of interactions between the agent and the environment.

IV. Numerical Results

In this section, we illustrate the proposed DT-assisted DRL method in the UAV task offloading system through numerical simulations. Unless otherwise specified, the system parameters is set as follows. In the UAV task offloading system, $M = 4$ MTs are randomly distributed in an area of 100×100 m^2, while the UAV has a fixed flight height $H = 100$ m [4]. The entire time period $T = 320$ s is divided into $I = 40$ time slots. At each time slot the UAV's flight time $t_{fly} = 1$ s [2] and the UAV's maximum flight speed $v_{\max} = 10$ m/s. We set the bandwidth for each MT $B = 1$ MHz, and the channel gain $\alpha_0 = -30$ dB at a reference distance of 1m. The noise power $\sigma^2 = -100$ dBm [2], the transmission power $P_{up} = 20$ dBm [4], and the required CPU cycles per bit $k = 1000$ cycles/bit [2]. We set the computing capability of the MT $f_{MT} = 0.6$ GHz and that of the MEC $f_{UAV} = 4.8$ GHz.

In DDPG, the size of the experience buffer $\mathcal{D}_E$ is 10000. Both actor network and critic network realized by FCNNs, which include three hidden layers with 400, 300, and 10 neurons per layer in that order. All hidden layers use the ReLU activation function. In the actor network, the output layer has 6 neurons and uses the Tanh activation function to output deterministic actions. In the critic network, the output layer has a neuron to output the Q value. In DT, the buffer $\mathcal{D}_T$ size is 40000. The predictive model uses a FCNN with two hidden layers, both with 64 neurons. The hidden layer uses the ReLU activation function, and the output layer uses the Sigmoid activation function. In the generative model, We use a VAE with encoder and decoder. In the encoder network, two hidden layers are used and the LeakyReLU activation function is used. The output layer uses two fully connected layers to output the mean and logarithmic variance, respectively [12]. The decoder network adopts two hidden layers with the ReLU activation function, and the output layer uses the Sigmoid activation function.

For comparison, four methods are described as follows:

- The proposed method: DT-assisted DDPG with predictive model and VAE model (denoted as DT with PM+VAE).
- The DDPG method.
- DT-assisted DDPG with predictive model only (denoted as DT with PM). In this method, only the predictive model is used to assist.
- DT-assisted DDPG with predictive model and generative adversarial network (GAN) model (denoted as DT with PM+GAN). GAN [13] has demonstrated excellent performance in computer vision and image generation tasks. Therefore, we replace the VAE in the generative model with GAN and compare the performance.

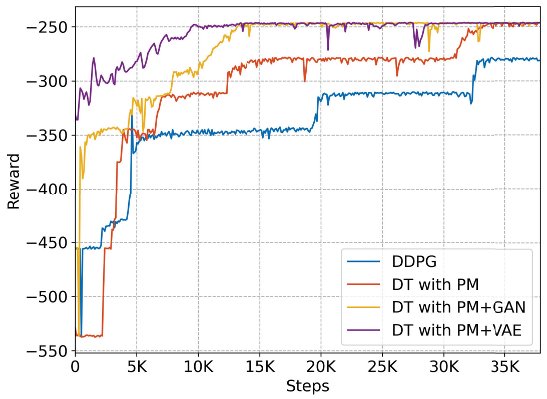

Fig. 2. Training performance of different approaches.

Fig. 2 shows the reward with the increasing training steps. We trained a total of 3.8×10^4 steps. Our proposed method achieves the highest reward level and the fastest convergence speed. We observe that the number of training steps to reach convergence for the DT with PM method, the DT with PM+GAN method, and proposed method are 33.1K, 13.3K, and 9.6K, respectively. The performance of the DT with PM+GAN method and our proposed method performs better than the DT with PM method in the training stage. This indicates that the generated diversity and forward-looking state data can better assist the predictive model in predicting the transformation characteristics of the physical environment and improve sample efficiency. The reward of the DDPG method obtains the lowest reward and convergence speed, indicating that the traditional DDPG algorithm is inefficient for training.

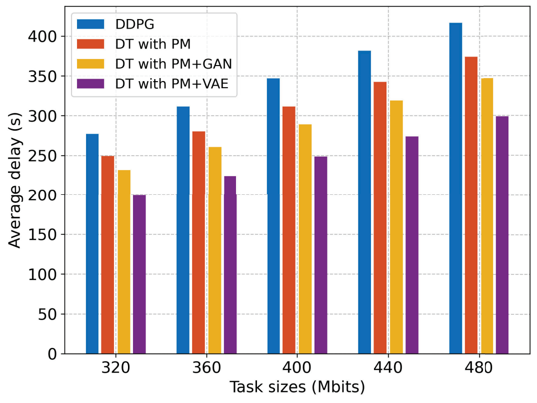

Fig. 3. Delay at different task sizes with 10K training steps.

Using the results obtained from training 10K steps, we compare these methods under different task sizes and the result is shown in Fig. 3. For the same task size, the processing delay of the proposed method is the lowest among the four approaches. This indicates that our method improves data sample efficiency during training, and good exploration results can be achieved by fewer training steps. The DDPG method always has the largest latency among the compared methods under the condition of fixed number of training steps, indicating that the DDPG algorithm requires a large space to be explored and is inefficient in training. For instance, with the task size of 480 Mbits, the latency of the proposed method is approximately 28.26%, 20.08%, and 13.80% lower than the DDPG, the DT with PM, and the DT with PM+GAN methods, respectively.

Fig. 4 shows the convergence performance of DDPG and the proposed method with different computing capabilities of MEC. We can find that when $f_{UAV} = 4.8$ GHZ, the DDPG method still does not reach convergence after 38K training steps and the delay is always higher than the proposed method. When $f_{UAV} = 5.2$ GHZ, the DDPG method converges at about 18K steps and the proposed method converges at about 8K steps. It shows that our proposed method can perform better than the traditional DDPG algorithm in different scenarios.

V. Conclusion

In this paper, we propose a DT assisted DRL approach and apply it to computation offloading in UAV assisted MEC system. We focus on minimizing the task processing delay

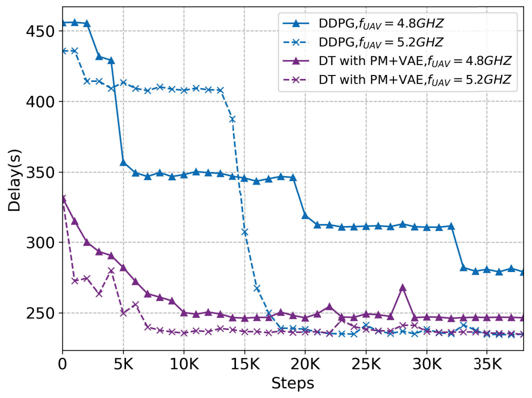

Fig. 4. Performance of DDPG and DT(PM+VAE)-DDPG under different computing capabilities of MEC.

by jointly optimizing the task offloading and the UAV's flight path. Then we transform the problem into a serialized MDP. The proposed DT is made up of a predictive model to predict the reward and next stage of physical environment, and a generative model to improve the sample efficiency. This method enables the DDPG algorithm to learn from physical and virtual interactions, improving the sample efficiency of the algorithm training process, which is effectively validated in task offloading problem. Numerical results show that our method can significantly accelerate the learning process of reinforcement learning and achieve better delay performance.

References

[1] N. Abbas, *et al.*, "Mobile edge computing: A survey", *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450-465, Feb. 2018.
[2] Q. Hu, Y. Cai, G. Yu, Z. Qin, M. Zhao and G. Y. Li, "Joint offloading and trajectory design for UAV-enabled mobile edge computing systems", *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1879-1892, Apr. 2019.
[3] N. Cheng *et al.*, "Space/Aerial-Assisted Computing Offloading for IoT Applications: A Learning-Based Approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1117-1129, May 2019.
[4] Y. Wang, W. Fang, Y. Ding and N. Xiong, "Computation offloading optimization for UAV-assisted mobile edge computing: A deep deterministic policy gradient approach", *Wireless Netw.*, vol. 27, pp. 2991-3006, 2021.
[5] J. J. Alcaraz, F. Losilla, A. Zanella and M. Zorzi, "Model-based reinforcement learning with kernels for resource allocation in RAN slices", *IEEE Trans. Wireless Commun.*, vol. 22, no. 1, pp. 486-501, Jan. 2023.
[6] W. Sun, H. Zhang, R. Wang and Y. Zhang, "Reducing offloading latency for digital twin edge networks in 6G", *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 12240-12251, Oct. 2020.
[7] Y. Lu, S. Maharjan and Y. Zhang, "Adaptive edge association for wireless digital twin networks in 6G", *IEEE Internet Things J.*, vol. 8, no. 22, pp. 16219-16230, Nov. 2021.
[8] J. Wu, Z. Huang, P. Hang, C. Huang, N. De Boer and C. Lv, "Digital twin-enabled reinforcement learning for end-to-end autonomous driving", *Proc. IEEE 1st Int. Conf. Digit. Twins Parallel Intell.*, pp. 62-65, 2021.
[9] S. Jeong, O. Simeone and J. Kang, "Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning", *IEEE Trans. Veh. Technol.*, vol. 67, no. 3, pp. 2049-2063, Mar. 2018.
[10] N. Zhao, Z. Ye, Y. Pei, Y.-C. Liang and D. Niyato, "Multi-agent deep reinforcement learning for task offloading in UAV-assisted mobile edge computing", *IEEE Trans. Wireless Commun.*, vol. 21, no. 9, pp. 6949-6960, Mar. 2022.
[11] J. Almutairi, M. Aldossary, H. A. Alharbi, B. A. Yosuf and J. M. H. Elmirghani, "Delay-optimal task offloading for UAV-enabled edge-cloud computing systems", *IEEE Access*, vol. 10, pp. 51575-51586, May 2022.
[12] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes", *Proc. Int. Conf. Learn. Representations*, 2014.
[13] I. Goodfellow *et al.*, "Generative adversarial nets", *Proc. Int. Conf. Neural Inf. Process. Syst.*, pp. 2672-2680, 2014.