

Deep Reinforcement Learning for Stochastic Computation Offloading in Digital Twin Networks

Yueyue Dai, *Member, IEEE*, Ke Zhang, Sabita Maharjan, *Senior Member, IEEE*,
and Yan Zhang, *Fellow, IEEE*

Abstract—The rapid development of Industrial Internet of Things (IIoT) requires industrial production towards digitalization to improve network efficiency. Digital Twin is a promising technology to empower the digital transformation of IIoT by creating virtual models of physical objects. However, the provision of network efficiency in IIoT is very challenging due to resource-constrained devices, stochastic tasks, and resources heterogeneity. Distributed resources in IIoT networks can be efficiently exploited through computation offloading to reduce energy consumption while enhancing data processing efficiency. In this paper, we first propose a new paradigm Digital Twin Networks (DTN) to build network topology and the stochastic task arrival model in IIoT systems. Then, we formulate the stochastic computation offloading and resource allocation problem to minimize the long-term energy efficiency. As the formulated problem is a stochastic programming problem, we leverage Lyapunov optimization technique to transform the original problem into a deterministic per-time slot problem. Finally, we present Asynchronous Actor-Critic (AAC) algorithm to find the optimal stochastic computation offloading policy. Illustrative results demonstrate that our proposed scheme is able to significantly outperforms the benchmarks.

Index Terms—Digital twin, Industrial Internet of Things, Deep reinforcement learning, Computation offloading.

I. INTRODUCTION

The Industrial Internet of Things (IIoT) is an enabling technology of Cyber-Physical Systems (CPSs) that can equip the industrial units, such as sensors, instruments, and devices with the ability to communicate and interact with each other. The IIoT has undergone rapid technological development in recent years. According to the report from International Data Corporation (IDC) [1], the number of connected devices will reach 41.6 billion and these devices are predicted to generate nearly 80 zettabytes of data by 2025. The high spread of IIoT requires industrial production towards network and digitalization.

Digital twin is a powerful technology to enable the digital transformation by creating virtual models of physical objects in the digital way, as shown in Fig. 1. The virtual models can understand the state of the physical entities through sensing data, so as to predict, estimate, and analyse the dynamic

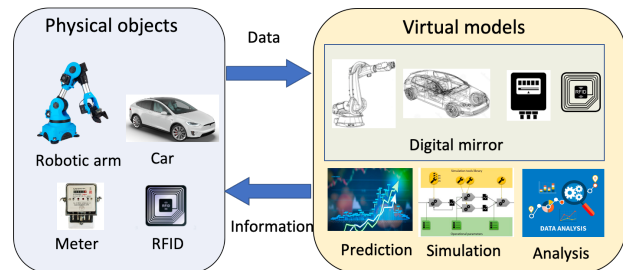


Fig. 1: Concept of digital twin

changes. The concept of digital twin is first proposed in [2] and applied by NASA to comprehensive diagnosis and maintenance of flight systems. Recently, digital twin has been expanded to smart cities, manufacturing and IIoT. Exploiting digital twin, the network topology and physical elements in IIoT can be well mirrored and we can make system management based on the mirrored models. However, there are many technical challenges in applying digital twins to IIoT. First, massive data collected from various IIoT devices needs to be processed timely. But the limited computing resources available at the local servers cannot support fast data processing and digital twin modelling in IIoT networks [3], [4]. Second, the interaction between the virtual models and the physical objects in a digital twin-enabled network requires frequent communication between them. Moreover, since the communication is wireless, the stochastic associated with the wireless channel may result in a poor transmission link, correspondingly longer service delay.

The IIoT applications, such as data analytics and smart manufacturing, involve plenty of computation tasks. To improve data/task processing efficiency and prolong battery lifetime of IIoT devices, computation offloading is a promising approach which offloads the collected data and computation tasks to distributed servers to process, such as base stations, access points, and road-side units in an IIoT network [5], [6]. There has been considerable amount of work focusing on computation offloading in wireless networks and vehicular networks. The authors in [7] proposed to offload computation tasks to lightweight and distributed road-side units to minimize task processing latency in vehicular networks. The authors in [8] proposed a joint computation offloading, power allocation, and channel assignment problem to maximize the achievable sum rate for 5G-enabled traffic management systems. The authors in [9] proposed to offload computation tasks to multiple distributed Small-cell Base Stations (SBS) and Macro-cell

This research was supported in part by the National Natural Science Foundation of China under Grant No. 61941102 and in part by the Xi'an Key Laboratory of Mobile Edge Computing and Security, under Grant No. 201805052ZD3CG36. (Corresponding author: Yan Zhang)

Y. Dai and K. Zhang are with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (email: yueyuedai@ieee.org; zhangke@uestc.edu.cn).

S. Maharjan and Y. Zhang are with Department of Informatics, University of Oslo, Norway, and also with Simula Metropolitan Center for Digital Engineering, Norway. (email: sabita@ifi.uio.no, yanzhang@ieee.org).

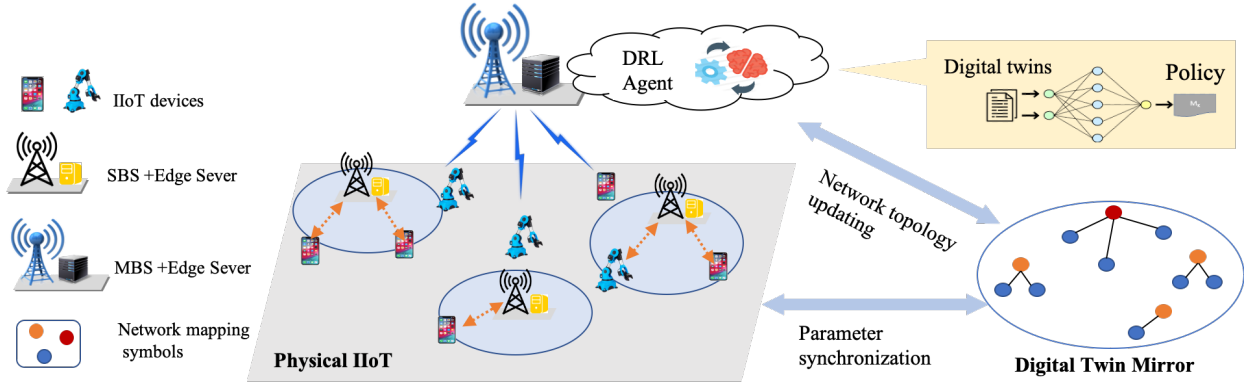


Fig. 2: The digital twin network

Base Stations (MBS) to minimize energy consumption in 5G networks. The above works typically assumed that each device executes a single computation task without considering the randomness of task arrivals [10], [11]. Such assumptions make the computation offloading design not practical for IIoT networks. Since devices in IIoT networks continuously generate data, stochastic task arrival model is more reasonable and long-term computation performance needs to be considered. Moreover, in an IIoT network with heterogeneous resources, it is challenging to jointly optimize offloading decision, transmission power, bandwidth and computation resource while also incorporating time-varying channel condition.

Deep Reinforcement Learning (DRL) is an emerging technique to address problems with characterized with time-varying feature [12], [13]. State-of-the-art works have utilized DRL for optimizing computation offloading in wireless networks and vehicular networks. For instance, the authors in [14] proposed a Deep-Q Network (DQN) based task offloading scheme to select the optimal edge server and the optimal transmission mode to maximize task offloading utility in vehicular networks. The authors in [15] proposed the double DQN based backscatter-aided hybrid data offloading scheme to reduce power consumption for data transmission. The authors in [16] proposed a Deep Deterministic Policy Gradient (DDPG) based computation offloading and resource allocation scheme to minimize system energy consumption in wireless networks. These works however mainly focus on to choosing whether to execute tasks locally or offload to edge servers with a deterministic task arrival model. These solutions are not directly applicable to IIoT networks since the task arrival model is stochastic.

In this paper, we first propose a Digital Twin Network (DTN) which utilizes digital twin to establish an efficient mapping between IIoT and digital systems. In the proposed DTN network, virtual models of IIoT entities can be created by monitoring real-time states of devices and base stations. Then, we formulate stochastic computation offloading problem as an optimization problem. Based on the virtual models and monitored information of digital twin, we design a DRL-based algorithm to solve the formulated problem. Our main contributions in this paper are summarized as follows:

- We propose an architecture to integrate digital twin with

the IIoT network to model network topology, physical devices and base stations.

- We formulate stochastic computation offloading problem as an optimization problem, and utilize Lyapunov optimization technique to equivalently transform the original problem to a deterministic per-time slot problem.
- We adopt Asynchronous Actor-Critic (AAC), a DRL-based algorithm, to solve the computation offloading and resource allocation problem. Numerical results demonstrate that our proposed algorithm significantly outperforms the benchmark policies.

II. SYSTEM MODEL DESCRIBED BY DIGITAL TWIN

A. The Digital Twin Network

Our digital twin network architecture is shown in Fig. 2, which consists of physical IIoT network and digital twin.

The physical IIoT network has three major components, i.e., distributed IIoT devices, SBSs, and centralized MBS. Each device collects the data from sensors and on-device applications, and they need to analyse the collected data. As data analysis is computation-intensive, devices with limited computation capability and battery, may not be able to complete them timely. So they have to offload these tasks to edge servers for a high level of quality of computation experience. SBSs are equipped with edge servers and they can provide devices computation resources. However, since an SBS often serves several devices, to ensure the requirements of all devices are satisfied, SBSs need to optimize computation resources, bandwidth and transmission power. The MBS is equipped with an edge server and an DRL agent, thus the MBS has sufficient communication, computation and AI-enabled processing capabilities.

The digital twin is a virtual model of physical elements and a digital representation of the physical system. Different from a virtual prototype, digital twin not only mirrors the characteristic of physical elements/system but also makes prediction, simulates the system, and can play a crucial role in optimizing the resources. In our digital twin network, we can utilize digital twin to (1) construct the network topology of physical IIoT, (2) monitor network parameters and models, i.e., dynamic changes of resources and stochastic task arrival processes, (3) optimize offloading and resource

allocation policy. Specifically, we deploy different functions on devices, SBSs and the MBS to build digital twin network. Devices run two DT functions: data collection and parameter synchronization. SBSs also run two DT functions: building local virtual models of devices and SBSs, and synchronizing parameters. The main functions of the MBS are to construct the network topology of the physical network and to design offloading and scheduling policy.

B. Network Model

Based on digital twin, the digital representation (i.e., virtual models) of the physical network is created (i.e., virtual world). The digital models here contain wireless network topology, communication model between devices and base stations, and the stochastic task queueing model.

1) Network Topology and Communication Model for DTN

Digital twin firstly models the wireless edge network as a discrete time-slotted system. A directed graph $G = (\mathcal{U}, \mathcal{B}, \varepsilon)$ is used to represent the network, where $\mathcal{U} = \{u_1, \dots, u_N\}$ and $\mathcal{B} = \{b_0, b_1, \dots, b_M\}$ denote the set of devices and base stations (b_0 is the index for MBS) respectively. ε denotes the association between devices and base stations. That is, if device u_i is connected to SBS b_j , the link will be recorded in edge set ε . Then, the digital twin uses a 3-tuple $DT_i(t)$ to characterize devices, i.e.,

$$DT_i(t) = \{p_{i,max}(t), l_i(t), f_i^l\} \quad (1)$$

where $p_{i,max}(t)$ denotes the maximal transmission power at time slot t , $l_i(t)$ denotes the current location of u_i , and f_i^l denotes the computation resources of local server. Similarly, the digital twin uses a 3-tuple $DT_j(t)$ to characterize base stations, i.e.,

$$DT_j(t) = \{l_j(t), w_j, f_j^e\} \quad (2)$$

where $l_j(t)$ denotes the current location of b_j , w_j denotes the channel bandwidth of b_j , f_j^e denotes the computation resource.

The task offloading between devices and base stations is facilitated through wireless communication. According to [9], wireless data rate is related to spectrum, interference and bandwidth. In wireless networks, to utilize spectrum efficiently, SBSs reuse the MBS's frequency resource and Orthogonal Frequency Division Multiple Access (OFDMA) is often adopted to suppress the interference. Thus, the interference between the MBS and SBSs can be ignored. Here, we consider devices communicate with the nearest base station to perform computation offloading. γ_j^s is defined as the coverage radius of SBS b_j . If the distance between device u_i and SBS b_j is less than γ_j^s (i.e., $r_{ij}^s(t) < \gamma_j^s$), device u_i can communicate with b_j . The wireless communication data rate between device u_i and SBS b_j can be expressed as

$$R_{ij}^s(t) = w_{ij}(t) \log\left(1 + \frac{p_i(t)h_{ij}^s(t)r_{ij}^s(t)^{-\alpha}}{\sigma^2 + I}\right), \quad (3)$$

where $w_{ij}(t)$ ($w_{ij}(t) \leq w_j$) is the bandwidth that SBS b_j allocates to device u_i at time slot t , $h_{ij}^s(t)$ is the current channel gain, α is path loss exponent, σ^2 is noise power, and $r_{ij}^s(t)$ is calculated based on the location of $l_i(t)$ and $l_j(t)$ (i.e., $r_{ij}^s(t) = \|l_i(t) - l_j(t)\|$). $I =$

$\sum_{i' \in \mathcal{U}/\{i\}, j' \in \mathcal{B}/\{j\}} p_{i'}(t)h_{i'j'}^s(t)r_{i'j'}^s(t)^{-\alpha}$ is the interference from other SBSs.

If device u_i does not lie within the coverage of any SBS, it will communicate with the MBS. The wireless communication data rate between device u_i and the MBS is

$$R_{i0}^m(t) = w_{i0}(t) \log\left(1 + \frac{p_i(t)h_{i0}^m(t)r_{i0}^m(t)^{-\alpha}}{\sigma^2}\right), \quad (4)$$

where $w_{i0}(t)$ ($w_{i0}(t) \leq w_0$) is the channel bandwidth between device u_i and the MBS at time slot t , $h_{i0}^m(t)$ is the channel gain between device u_i and the MBS at time slot t , $r_{i0}^m(t) = \|l_i(t) - l_0(t)\|$ is the distance between device u_i and the MBS.

2) Stochastic Task Queueing Model for DTN

At the beginning of time slot t , device u_i generates and stores $\lambda_i(t)$ (bits/slot) of computation tasks into the local dataset. Without loss of generality, we assume $\lambda_i(t)$ in different time slots are independent, and $\mathbb{E}[\lambda_i(t)] = \lambda$. Due to the limitation of computation resources, each device executes part of the computation tasks at its local server and offloads part of them to the associated base station. The rest will be queueing in the local task buffer and we consider the buffer has sufficient capacity. We denote the size of the computation tasks that is executed locally as $D_i^l(t)$ and the size of the computation tasks offloaded to base station b_j ($j \in \mathcal{B}$) as $D_{ij}^e(t)$. The queue length of local task buffer at the beginning of time slot t on device u_j is denoted as $Q_j^l(t)$ and the queue length is dynamically updated with the following equation:

$$Q_i^l(t+1) = \max\{Q_i^l(t) - \Psi_i(t), 0\} + \lambda_i(t) \quad (5)$$

where $\Psi_i(t) = D_i^l(t) + D_{ij}^e(t)$ is the size of the computation tasks that leaves the task buffer of device u_i during time slot t .

Each edge server also has a task buffer to store the offloaded but not yet executed task. We denote the queue length of edge task buffer at the beginning of time slot t on base station b_j ($j \in \mathcal{B}$) as $Q_j^e(t)$. The queue length is dynamically updated by:

$$Q_j^e(t+1) = \max\{Q_j^e(t) - \Psi_j(t), 0\} + \sum_{i \in \mathcal{U}} D_{ij}^e(t) \quad (6)$$

where $\sum_{i \in \mathcal{U}} D_{ij}^e(t)$ is the amount of tasks offloaded from devices during time slot t . $\Psi_j(t)$ is the size of the computation tasks that departs edge task buffer (i.e., executed by edge server). According to the definition of stability in [17], task queue is stable if all computation tasks satisfy the following constraints:

$$\lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i \in \mathcal{U}} \mathbb{E}\{Q_i^l(t)\} < \infty \quad (7a)$$

$$\lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{j \in \mathcal{B}} \mathbb{E}\{Q_j^e(t)\} < \infty \quad (7b)$$

C. Task Offloading Model

During time slot t , device u_i executes $D_i^l(t)$ locally and offloads $D_{ij}^e(t)$ to base station b_j . Next, we will calculate

the energy consumption of local execution and computation offloading.

1) *Local Execution:*

Let $f_i^l(t)$ denote the computation resource (i.e., CPU cycles per second) of device u_i during time slot t . c denotes the required number of CPU cycles for executing one bit of computation task. Thus, the size of computation tasks executed locally will be

$$D_i^l(t) = \frac{\tau f_i^l(t)}{c}, \quad (8)$$

where τ is duration of the time slot.

The energy consumption of unit computation resource is $\varsigma(f_i^l)^2$, where ς is the effective switched capacitance depending on the chip architecture [9]. We denote local energy consumption for computing task $D_i^l(t)$ as $E_i^l(t)$, which can be defined as

$$E_i^l(t) = \varsigma \tau f_i^l(t)^3. \quad (9)$$

2) *Edge Server Execution:*

Devices offload their tasks to base stations via wireless communication. Since devices are associated with different base stations, the offloaded tasks of device u_i during time slot t can be expressed as

$$D_{ij}^e(t) = \begin{cases} R_{ij}^s(t)\tau & j \in \mathcal{B}/\{b_0\} \\ R_{i0}^m(t)\tau & j = b_0 \end{cases} \quad (10)$$

The energy consumption in this case consists of three parts. The first one is the energy consumption of uplink wireless transmission for offloading. The second one is the computation energy consumption which is related to the allocated computation resources. The third one is the energy consumption of downlink wireless transmission for offloading computation result to the devices. Since the size of the result is very small, we ignore the energy consumption for downlink transmission. Thus, the energy consumption for executing task $D_{ij}^e(t)$ on base station b_j is given by

$$E_{ij}^e(t) = p_i(t)\tau + \frac{D_{ij}^e(t) * c}{f_{ij}^e(t)} * \varepsilon, \quad (11)$$

where $f_{ij}^e(t)$ is the computation resource that b_j allocates to device u_i at time slot t , ε is the energy consumption for unit computation on edge servers.

The total energy consumption is consisted of the energy consumption of task execution in the local and edge servers, as well as the transmission energy consumption for computation offloading. Therefore, the total energy consumption can be expressed as

$$E^{tol}(t) = \sum_{i \in \mathcal{U}} E_i^l(t) + \sum_{i \in \mathcal{U}} \sum_{j \in \mathcal{B}} E_{ij}^e(t), \quad (12)$$

III. PROBLEM FORMULATION

In this section, we first formulate the stochastic computation offloading problem of DTN as an optimization problem, and then transform the formulated problem based on Lyapunov optimization.

A. Stochastic Computation Offloading Problem

The objective of stochastic computation offloading problem is to minimize network efficiency η_{EE} . η_{EE} is defined as the ratio of long-term total energy consumption to the corresponding long-term aggregate accomplished computation tasks, i.e.,

$$\eta_{EE} = \frac{\lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{E^{tol}(t)\}}{\lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i \in \mathcal{U}} \sum_{j \in \mathcal{B}} \mathbb{E}\{D_i^l(t) + D_{ij}^e(t)\}} \quad (13)$$

The system operation at time slot t can be denoted as $\mathbf{a}(t) = [\mathbf{w}(t), \mathbf{p}(t), \Psi(t), \mathbf{f}^l(t), \mathbf{f}^e(t)]$, where $\mathbf{w}(t) = [w_{10}(t), \dots, w_{NM}(t)]$ is bandwidth allocation vector, $\mathbf{p}(t) = [p_1(t), \dots, p_N(t)]$ is transmission power vector, $\Psi(t) = [\Psi_0(t), \dots, \Psi_M(t)]$ is the vector associated with the computation task that leaves edge servers, $\mathbf{f}^l(t) = [f_1^l(t), \dots, f_N^l(t)]$ and $\mathbf{f}^e(t) = [f_{10}^e(t), \dots, f_{NM}^e(t)]$ are the vector of computation resource that edge servers allocate to devices. Taking network stability constraint into account, the optimization problem for minimizing η_{EE} can be formulated as:

$$\begin{aligned} P1: \quad & \min_{\mathbf{a}(t)} \eta_{EE} \\ \text{s.t.} \quad & \sum_{i \in \mathcal{U}} \frac{w_{ij}(t)}{w_j} \leq 1, \quad w_{ij}(t) \geq 0 \quad (14a) \\ & 0 \leq p_i(t) \leq p_{i,max}(t), \quad (14b) \\ & 0 \leq f_i^l(t) \leq f_i^l, \quad (14c) \\ & \sum_{i \in \mathcal{U}} f_{ij}^e(t) \leq f_j^e, \quad f_{ij}^e(t) \geq 0 \quad (14d) \\ & \Psi_j(t) * c \leq f_j^e \tau, \quad \Psi_j(t) \geq 0 \quad (14e) \\ & (7a) - (7b) \end{aligned}$$

Constraint (14a) is the bandwidth allocation constraint. Constraints (14b) and (14c) denote the transmission power and computation resource constraints, respectively. Constraint (14d) ensures that the sum of the computation resource of each base station allocated to all devices does not exceed the total amount of computation resource the base station has. Constraint (14e) implies that the amount of computation resource for processing task Ψ_j cannot exceed the available computation resources. Constraints (7a) and (7b) guarantee that the stability of all task queues.

Since computation resource, transmission power and bandwidth need to be determined at each time slot, P1 is a stochastic optimization problem, which is challenging to solve by applying classic convex optimization algorithms such as interior-point method and Lagrangian duality theory. From Eq. (10) and Eq. (11), wireless communication rate and allocated computation resource jointly determine the energy consumption of edge server execution. Thus, the radio resource management problem is coupled with the computation resource allocation problem. Moreover, in radio resource management problem, bandwidth and transmission power are also highly coupled. The complex coupling among optimization variables and mixed combinatorial feature makes it difficult to solve P1. Further, the stochastic task arrival, dynamic channel

state information and dynamic task buffer make designing an efficient resource management policy for devices and edge servers quite challenging.

Lyapunov optimization is a powerful methodology for solving optimization problems with long-term objective and constraints, which requires less prior information on the task arrival, channel state information, and task buffer. The principal idea behind Lyapunov optimization is to transform the optimization problem with long-term objective into a series of subproblems with short-term objective, and to transform the long-term constraints into constraints with queue stability. Besides, Lyapunov optimization is of low computational complexity by optimizing subproblem through an online algorithm. In this paper, we exploit Lyapunov optimization to transform the original stochastic optimization problem as a deterministic per-time block problem and propose a stochastic computation offloading algorithm to solve P1.

B. Lyapunov-based Problem Transformation and Digital twin-predicted Perturbation

To construct a Lyapunov optimization framework, we add a perturbation vector $\beta = [\beta_1, \dots, \beta_N]$ in Lyapunov function to keep the value of this function always small. The perturbation parameters are simulated in the digital twin and it will be given in the following. We define the quadratic Lyapunov function as the sum of squared queue backlogs,

$$L(\Theta(t)) = \frac{1}{2} \left\{ \sum_{i \in \mathcal{U}} [Q_i^l(t) - \beta_i]^2 + \sum_{j \in \mathcal{B}} Q_j^e(t)^2 \right\} \quad (15)$$

where $\Theta(t) = [Q^l(t), Q^e(t)]$ represents current task queue lengths of devices and edge servers. Further, we define the conditional drift as

$$\Delta L(\Theta(t)) = \mathbb{E}[L(\Theta(t+1)) - L(\Theta(t)) | \Theta(t)] \quad (16)$$

By minimizing $\Delta L(\Theta(t))$, we can short the length of task queues towards a smaller value.

Accordingly, the Lyapunov drift-plus-penalty function can be expressed as

$$\Delta_V L(\Theta(t)) = \Delta L(\Theta(t)) + V \mathbb{E}[\eta_{EE}(t) | \Theta(t)] \quad (17)$$

where $\eta_{EE}(t) = E^{tol}(t) / \sum_{i \in \mathcal{U}} \sum_{j \in \mathcal{B}} (D_i^l(t) + D_{ij}^e(t))$, V is a non-negative weight parameter. By minimizing $\Delta_V L(\Theta(t))$, we can ensure network stability, and meanwhile minimize network EE. We derive the upper bound of $\Delta_V L(\Theta(t))$ as,

$$\begin{aligned} \Delta_V L(\Theta(t)) &\leq C - \sum_{i \in \mathcal{U}} [Q_i^l(t) - \beta_i] \mathbb{E}[\Psi_i(t) - \lambda_i(t) | \Theta(t)] \\ &- \sum_{j \in \mathcal{B}} Q_j^e(t) \mathbb{E}[\Psi_j(t) - \sum_{i \in \mathcal{U}} D_{ij}^e(t) | \Theta(t)] + V \mathbb{E}[\eta_{EE}(t) | \Theta(t)]. \end{aligned} \quad (18)$$

where $C = \frac{1}{2} \{ \sum_{i \in \mathcal{U}} [\Psi_{i,max}^2 + \lambda_{i,max}^2] + \sum_{j \in \mathcal{B}} [\Psi_{j,max}^2 + (\sum_{i \in \mathcal{U}} D_{ij,max}^e)^2] \}$ and $\Psi_{i,max}$, $\lambda_{i,max}$, $\Psi_{j,max}$ and $D_{ij,max}^e$ are the upper bounds of $\Psi_i(t)$, $\lambda_i(t)$, $\Psi_j(t)$ and $D_{ij}^e(t)$, respectively.

Based on the Lyapunov optimization theory, we can minimize the right side of the inequality in (18) to obtain the

Algorithm 1 The Stochastic Computation Offloading Algorithm with Digital Twin-predicted Perturbation for DTN

- 1: At the beginning of each time slot, digital twin first predicts perturbation vector β based on local task queue and Eq. (19),
- 2: Digital twin observes $\Theta(t)$ and $\lambda_i(t)$ and determines $a(t)$ by solving the following problem in each time slot,

$$\begin{aligned} P2 : \min_{\mathbf{a}(t)} & V [E^{tol}(t) - \eta_{EE}(t) \sum_{i \in \mathcal{U}} \sum_{j \in \mathcal{B}} (D_i^l(t) + D_{ij}^e(t))] + \sum_{j \in \mathcal{B}} \\ & \{ Q_j^e(t) [\sum_{i \in \mathcal{U}} D_{ij}^e(t) - \Psi_j(t)] \} - \sum_{i \in \mathcal{U}} [Q_i^l(t) - \beta_i] [\Psi_i(t) - \lambda_i(t)] \\ & s.t. \quad (7a) - (7b), (14a) - (14e) \end{aligned} \quad (20)$$

- 3: Updates $Q_i^l(t)$ and $Q_j^e(t)$ based on Eq. (5) and (6),
 - 4: $t = t + 1$.
-

optimal solution of P1. According to [18], perturbation vector β is very important as it influences the performance of optimization of the designed algorithm. Based on the definition of perturbation vector in [18], β is the maximal lower bound of local task queue. Since digital twin is a mirror of physical network, it can easy get any information of the network and predicts each one of β based on

$$\beta_i = V \eta'_{EE}(t) + \Psi_{max} \quad (19)$$

where $\Psi_{max} = \max(\Psi_i(t))$.

Thus, we first utilize digital twin to simulate the perturbation parameter of devices and then optimize the right side of the inequality in (18) in each time slot. The proposed stochastic computation offloading algorithm for DTN is shown in Algorithm 1, where P2 needs to be solved per-time slot. The traditional method to solve P2 is to decompose it into several sub-problems and alternatively solving sub-problems until it converges to the global optimal solution. However, when wireless channels change or task queues update, each sub-problem needs to recalculate the optimal solution. Frequent operations to solve sub-problems will influence the convergence. DRL is an emerging technique which can find a near optimal solution in a real-time manner. Thus, we design a DRL-based algorithm to find the optimal solution of P2.

IV. DRL-EMPOWERED STOCHASTIC COMPUTATION OFFLOADING ALGORITHM FOR DTN

The framework of the digital twin enabled DRL algorithm is illustrated in Fig. 3. Digital twin mirrors the network topology and parameters of physical wireless network and transmits network state to DRL. DRL derives the best strategy to minimize network energy efficiency.

A. Digital Twin-simulated Network Environment

To solve P2, the system first constructs Markov decision process, i.e., $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, and then use DRL algorithm to explore actions. From Fig. 3, the network state $s(t)$ is constructed by digital twin and outputted to DRL agent.

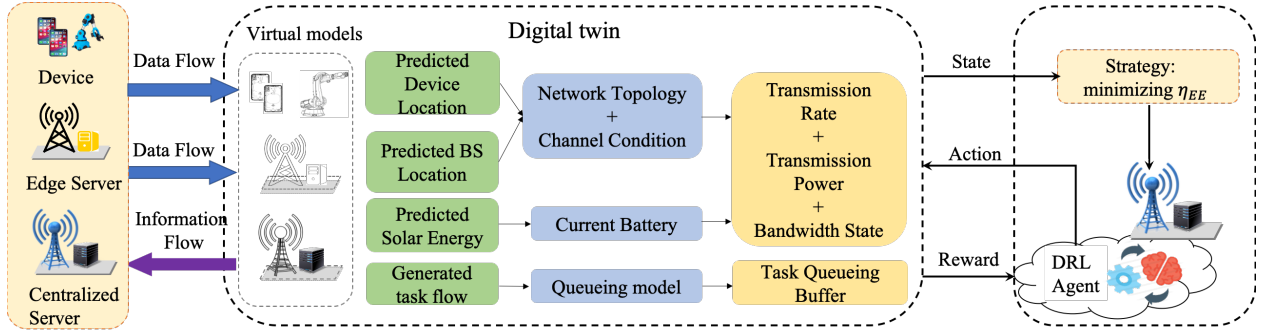


Fig. 3: Digital twin enabled DRL algorithm

To gather network environment information, digital twin needs to predict location, energy and the generated task flow of devices and base stations. Digital twin can adopt the existing K-Nearest Neighbors (KNN) classification method and position prediction algorithm in [19] to predict users' location. The maximal current transmission power $p_{i,max}(t)$ at time slot t is the summarize of the predicted energy and the $p_{i,max}(t-1)$. The generated task flow is based on the application running on each device. After gathering the network information, digital twin updates network topology, channel condition and queueing model. The main operation in the update of network topology is to make user association decision, which decides the connection between devices and base stations. Here we adopt online user association method in [20]. Then, digital twin generates current state and transmits it to the DRL agent.

Thus, at the beginning of time slot t , the DRL agent construct system state which includes transmission data rates between devices and base stations, available bandwidth, computation resources, transmission power, and queue length. We can define system state $s(t) \in \mathcal{S}$ at time slot t as

$$s(t) = \{\mathbf{R}(t), \mathbf{F}, \mathbf{p}_{max}(t), \mathbf{w}, \Theta(t)\}. \quad (21)$$

The state space \mathcal{S} is as follows:

- $\mathbf{R}(t) = \{[R_{11}^s(t), \dots, R_{1M}^s(t)], \dots, [R_{N1}^s(t), \dots, R_{NM}^s(t)], [R_{10}^m(t), \dots, R_{N0}^m(t)]\}$: $N \times (M+1)$ wireless data rate matrix where $R_{ij}^s(t) \geq 0$ and $R_{i0}^m(t) \geq 0$;
- $\mathbf{F} = [f_1^l, \dots, f_N^l, f_0^e, \dots, f_M^e]$: $1 \times (N+M+1)$ computation resource vector where $f_i^l \geq 0$ and $f_j^e \geq 0$;
- $\mathbf{p}_{max}(t) = [p_{1,max}(t), \dots, p_{N,max}(t)]$: $N \times 1$ transmission power vector at time slot t ;
- $\mathbf{w} = [w_0, \dots, w_j]$: $1 \times (M+1)$ bandwidth vector;
- $\Theta(t) = [Q^l(t), Q^e(t)]$, where $Q^l(t) = [Q_1^l(t), \dots, Q_N^l(t)]$ indicates the queue length of the local task buffer and $Q^e(t) = [Q_0^e(t), \dots, Q_M^e(t)]$ is the queue length of the task buffer on edge servers.

Since $\mathbf{a}(t) = [\mathbf{w}(t), \mathbf{p}(t), \Psi(t), \mathbf{f}^l(t), \mathbf{f}^e(t)]$ in P2 denotes system operation at time slot t , we define it as the output from the DRL agent (i.e., action). The action space \mathcal{A} includes following fields:

- $\mathbf{w}(t) = [w_{10}(t), \dots, w_{NM}(t)]$: $N \times (M+1)$ bandwidth allocation matrix where $w_{ij}(t) \in [0, w_j]$;
- $\mathbf{p}(t) = [p_1(t), \dots, p_N(t)]$: represents $N \times 1$ transmission power vector where $p_i(t) \in [0, p_{i,max}(t)]$;

- $\Psi(t) = [\Psi_0(t), \dots, \Psi_M(t)]$: $1 \times (M+1)$ vector where $0 \leq \Psi_j(t) \leq f_j^e \tau / c$ is the computation task that leaves edge server j ;
- $\mathbf{f}^l(t) = [f_1^l(t), \dots, f_N^l(t)]$: $N \times 1$ computation resource allocation vector where $f_i^l(t) \in [0, f_i^l]$;
- $\mathbf{f}^e(t) = [f_{10}^e(t), \dots, f_{NM}^e(t)]$: $N \times (M+1)$ computation resource allocation matrix where $f_{ij}^e(t) \in [0, f_e^j]$;

It is worth noting that all variables in action $\mathbf{a}(t)$ are continuous. Thus, we will utilize a policy gradient-based DRL algorithm to explore policy.

After executing action $\mathbf{a}(t)$, digital twin updates system state and estimates immediate reward $\mathcal{R}^{imm}(s(t), \mathbf{a}(t))$. In a traditional Markov decision process, the system updates its state based on the given transition probability $Pr(s(t+1)|s(t), \mathbf{a}(t))$. However, in DRL, the distribution of transition probability is often unknown. The DRL agent utilizes deep neural network to approximate it.

The immediate reward function is defined as the objective of P2 problem, i.e.,

$$\begin{aligned} \mathcal{R}^{imm}(s(t), \mathbf{a}(t)) = & -V[E^{tol}(t) - \eta_{EE}(t) \sum_{i \in \mathcal{U}} \sum_{j \in \mathcal{B}} (D_i^t(t) + D_{ij}^e(t))] \\ & + \sum_{i \in \mathcal{U}} \{Q_j^e(t)[\Psi_j(t) - \sum_{i \in \mathcal{U}} D_{ij}^e(t)]\} + \sum_{i \in \mathcal{U}} [Q_i^l(t) - \beta_i][\Psi_i(t) - \lambda_i(t)] \end{aligned} \quad (22)$$

After computing immediate reward, the system updates its state from $s(t)$ to $s(t+1)$ based on action $\mathbf{a}(t)$.

The objective of the DRL agent is to maximize the cumulative reward,

$$\mathcal{R} = \max \mathbb{E} \left[\sum_{t=0}^{T-1} \delta^t \mathcal{R}^{imm}(s(t), \mathbf{a}(t)) \right], \quad (23)$$

where $\delta \in [0, 1]$ is the discount factor. If all tasks are satisfying the constraints of P2, DRL agent gets a total reward. Otherwise, the agent receives a penalty, which is a negative constant.

B. Asynchronous Actor-Critic Algorithm

The DRL algorithm is classified into value-based and policy gradient-based. Value-based DRL algorithms, such as DQN and double DQN, estimate Q-values and ϵ -greedy strategy to explore policy with discrete action space. But value-based DRL algorithms are of limited value for problems with continuous action space. Policy gradient-based DRL can learn

stochastic policies effectively for tackling problems with continuous action space problems. The main idea of this method is to optimize a parameterized stochastic policy by estimating the gradient of the expected reward of the policy and then updating the parameters of the policy in the gradient direction. We deploy AAC algorithm in digital twin to optimize cumulative reward R .

AAC is an asynchronous learning algorithm which utilizes multiple agents to interact with its own environment and each agent contains a replica of the environment [21]. A specific AAC agent is *Actor-Critic* based, where *Actor* is used to generate actions and *Critic* is used to evaluate and criticize the current policy by processing the reward obtained from the environment.

1) *Actor-Critic based policy gradient training:*

$a(t) = \pi(s(t)|\theta_\pi)$ denotes the policy learned from current state, where $\pi(s(t)|\theta_\pi)$ is the explored offloading and resource allocation policy produced by deep neural network of actor network. The network parameter of actor network is denoted as θ_π and it is trained through the policy gradient method [21]. The gradient of the expected cumulative discounted reward is calculated by,

$$\nabla_{\theta_\pi} \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \delta^t R^{imm}(t) \right] = \mathbb{E}_\pi [\nabla_{\theta_\pi} \log \pi(s(t)|\theta_\pi) A_\pi(s, a)] \quad (24)$$

where $A_\pi(s, a)$ is the difference between the expected cumulative discounted reward starting from state s when agent chooses action a and follows policy π . Here, $A_\pi(s, a)$ is called the advantage function which indicates whether things get better or worse than expected. $A_\pi(s, a)$ is calculated using,

$$A_\pi(s, a) = \mathcal{R}^{imm}(s(t), a(t)) + \delta v_{\theta_v}(s(t+1)) - v_{\theta_v}(s(t)), \quad (25)$$

The parameter θ_π is updated based on:

$$\theta_\pi = \theta_\pi + \alpha_\pi \sum_t \nabla_{\theta_\pi} \log \pi(s(t)|\theta_\pi) A_\pi(s, a), \quad (26)$$

where α_π is the learning rate of the actor network. We use critic network to estimate the cumulative discounted reward of each state following the current actor network's policy, which is also expressed as the value of each state, $v_{\theta_v}(s(t))$. θ_v is the network parameter of critic network. The parameter θ_v is updated as follows:

$$\theta_v = \theta_v + \alpha_v \sum_t \nabla_{\theta_v} (\mathcal{R}^{imm}(s(t), a(t)) + \delta v_{\theta_v}(s(t+1)) - v_{\theta_v}(s(t)))^2 \quad (27)$$

where α_v is the learning rate of the critic network.

After the value function approximation $v_{\theta_v}(s(t))$ and parameter θ_v are updated by critic process, actor network then uses the advantage function $A_\pi(s, a)$ outputted from the critic process to update its policy parameters.

2) *Asynchronous Learning with Experience Replay:*

In DQN algorithm, there is an important component, i.e., replay memory, which disrupts the correlation between the experiences such that the sequence in DRL meets the independent and identical distribution. However, replay memory needs an off-policy learning algorithm to generate experiences

Algorithm 2 Asynchronous Actor-Critic algorithm for Each Learning Agent

- 1: Assume agent parameter θ'_π and θ'_v
 - 2: Initialize learning step counter $t = 1$;
 - 3: **repeat**
 - 4: Synchronize agent parameter $\theta'_\pi = \theta_\pi$ and θ'_v
 - 5: Update global shared counter T
 - 6: $t_{start} = t$
 - 7: Use digital twin to construct current state $s(t)$
 - 8: **repeat**
 - 9: Perform action $a(t)$ for problem P2 based on policy $\pi(s(t)|\theta'_\pi)$
 - 10: Transfer to new state $s(t+1)$ and calculate immediate reward $R^{imm}(s(t), a(t))$
 - 11: $t \leftarrow t + 1$
 - 12: $T \leftarrow T + 1$
 - 13: **until** $t - t_{start} == t_{max}$
 - 14: $R = v_{\theta'_v}(s(t))$
 - 15: **for** $i \in \{t-1, \dots, t_{start}\}$ **do**
 - 16: $reward.append(R^{imm}(s(t), a(t)) + \delta R)$
 - 17: **end for**
 - 18: send *reward* to global agent
 - 19: **until** $T > T_{max}$
-

Algorithm 3 Asynchronous Actor-Critic algorithm for Global Agent

- 1: Assume global shared parameter θ_π and θ_v , and global shared counter $T' = 0$
 - 2: **while** receive *reward* from agent **do**
 - 3: **for** $i \in \{t-1, \dots, t_{start}\}$ **do**
 - 4: Accumulate gradients with respect to θ'_π and θ'_v :
 - 5: $d\theta_\pi \leftarrow d\theta_\pi + \alpha_\pi \sum_i \nabla_{\theta'_\pi} \log \pi(s(t)|\theta'_\pi) A_\pi(s, a)$
 - 6: $d\theta_v \leftarrow d\theta_v + \alpha_v \sum_i \nabla_{\theta'_v} A_\pi^2(s, a)$
 - 7: **end for**
 - 8: perform asynchronous update of θ_π using $d\theta_\pi$ and θ_v using $d\theta_v$
 - 9: **if** $T > T_{max}$ **then**
 - 10: break
 - 11: **end if**
 - 12: **end while**
-

and needs large amount of memory to store the generated experience. AAC is an online DRL algorithm that can reduce correlation between adjacent samples by asynchronous learning with considerably less amount of computation. To implement AAC, DTN sets up a global agent and multiple learning agents. The algorithms to be carried by learning agent and global agent are given in Algorithm 2 and Algorithm 3, respectively. Learning agent is deployed at SBS and it can interact with its own environment and the environment of all agents has same settings and structures. The learning agents parallelly explore and accumulate offloading and resource allocation policy. After every t_{max} learning steps, agents will send the accumulated updates to the global agent. The global agent is deployed in MBS and it asynchronously updates θ_π and θ_v . T_{max} represents the max training episodes.

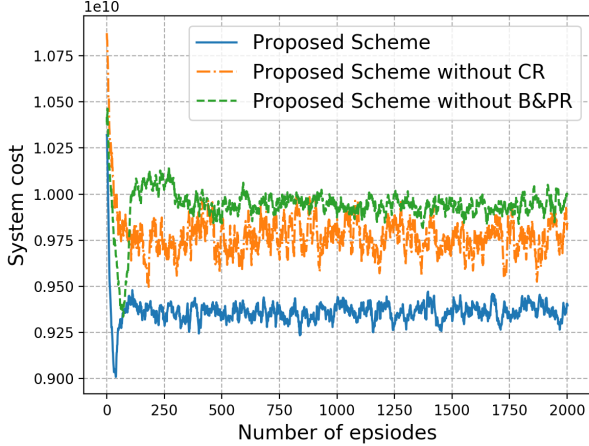


Fig. 4: System cost under different schemes.

V. NUMERICAL RESULTS

We consider a network topology with one MBS, $M = 3$ SBSs, and $N = 20$ devices. We consider Rayleigh fading channels. The maximum transmission power of devices is set to 100 mW. The noise power is $\sigma^2 = 10^{-11}$ mW. The bandwidth of the MBS and each SBSs are 10 MHz and 5 MHz, respectively. In addition, $\tau = 100$ ms, $c = 100$ cycles/bit. The CPU computation capabilities of the devices, the SBSs and the MBS, are 0.5, 10, and 50 GHz, respectively. The actor network of AAC has three fully-connected hidden layers each with 128 neurons whose activation function is ReLU and an output layer with 8 neurons using softmax function as the activation function. The critic network has three fully-connected hidden layers each with 128 neurons whose activation function is ReLU and one linear neuron as output. We use Python and TensorFlow to evaluate the performance of our proposed stochastic computation offloading algorithm. Based on the definition of immediate reward (22), the minimization of P2 is equivalent to the maximization of DRL reward. For ease of observation, we define the objective of P2 as system cost and the system cost equals to the negative value of the DRL cumulative reward.

Fig. 4 illustrates the relationship between system cost and training episodes under different schemes. The blue curve represents joint optimization of computation offloading, bandwidth and transmission power, but without computation resource allocation. The green curve shows the performance of the proposed scheme with computation offloading and computation resource allocation optimization but without bandwidth and transmission power allocation. From Fig. 4 we can see the performance of the red curve outperforms the two benchmarks, since it can concurrently optimize computation offloading, bandwidth and transmission power, and computation resource allocation. Besides, the system cost of the blue curve is lower than the system cost of the green curve. This means, compared with the optimization of computation resource, the optimization of bandwidth and transmission power has a greater influence on the performance.

Fig. 5 plots the comparison of the system cost with respect

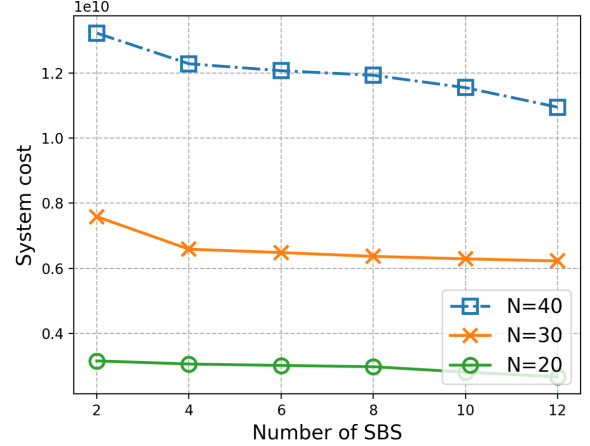


Fig. 5: System cost with respect to the number of SBSs

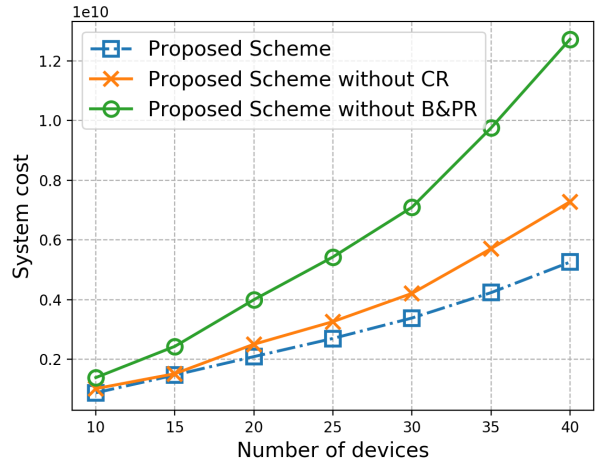


Fig. 6: System cost with respect to the number of devices under different schemes.

to the number of SBSs. We observe that when $N = 40$, the value of system cost decreases with the increase of the number of SBS. When $N = 20$, the value of system cost does not change much with the increase of the number of SBS. This indicates, when the number of devices is large, increasing the number of SBSs can reduce system cost. When the number of devices is small, increasing the number of base stations has little effect on reducing system cost.

Fig. 6 shows the comparison of the system cost with respect to the number of devices under different schemes. The number of devices ranges from 10 to 40. From Fig. 6, we can draw several observations. First, the system cost of three different schemes respectively increases as the number of devices becomes large. The reason is that the growth of devices leads to more offloading requests, which results in the consumption of more communication and computation resource. Second, the performance of the proposed algorithm outperforms two benchmarks by jointly optimizing computation offloading, bandwidth and transmission power, and computation resource allocation.

Fig. 7 illustrates the comparison of system cost with respect

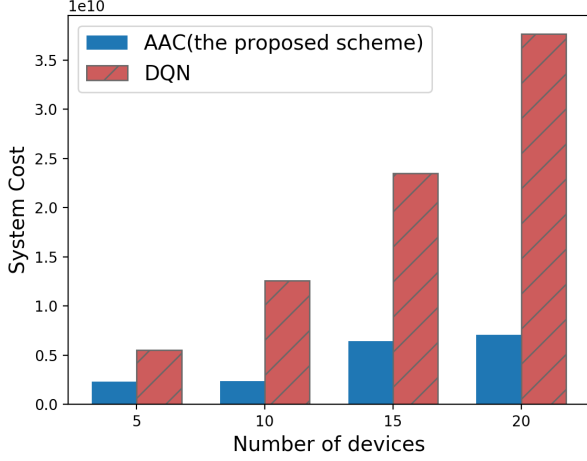


Fig. 7: System cost with respect to the number of devices under different DRL algorithms.

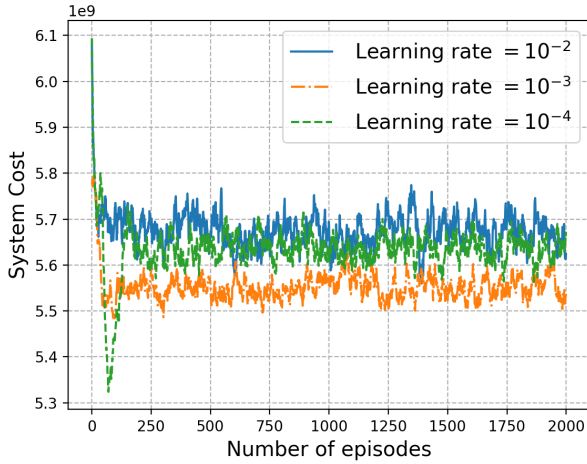


Fig. 8: Impact of the learning rate on performance.

to number of devices under different DRL algorithms. The number of devices ranges from 5 to 20. We can observe that the system cost increases with the increase in the number of devices. For given amount of computation resources, a large number of devices results in a low task execution rate and high energy consumption. This inevitably increases system cost. Moreover, our AAC based algorithm performs considerably better compared to the DQN. The main reason is that action discretization in DQN may lead to skipping better actions. Fig. 8 shows the impact of learning rate on the performance of the proposed algorithm. We can see, when learning rate is 0.001, the system cost of the proposed algorithm converges to the lowest value. Thus, 0.001 is the best learning rate for the proposed algorithm.

VI. CONCLUSIONS

In this paper, we proposed a DTN architecture for IIoT, which utilizes digital twin to construct the network topology and stochastic task arrival model in IIoT networks. Then, we formulated the stochastic computation offloading and resource

allocation problem to jointly optimize offloading decision, transmission power, bandwidth and computation resource. As the formulated problem is a non-convex stochastic programming problem, we leverage the Lyapunov optimization technique to equivalently transform the original problem to a deterministic per-time slot problem. Finally, we utilized AAC algorithm to solve the computation offloading and resource allocation problem. Numerical results demonstrate that our proposed algorithm significantly outperforms the benchmarks.

REFERENCES

- [1] IDC, "The growth in connected iot devices is expected to generate 79.4 zb of data in 2025, according to a new idc forecast," <https://www.idc.com/getdoc.jsp?containerId=prUS45213219>, 2019.
- [2] E. Glaessgen and D. Stargel, "The digital twin paradigm for future nasa and us air force vehicles," in *53rd AIAA/ASME/ASCE/AHS/ASC structures*, 2012, p. 1818.
- [3] K. Zhang, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Mobile edge computing and networking for green and low-latency internet of things," *IEEE Commun. Mag.*, vol. 56, no. 5, pp. 39–45, 2018.
- [4] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Communication-efficient federated learning for digital twin edge networks in industrial iot," *IEEE Trans. Ind. Inform.*, no. , doi:10.1109/TII.2020.3010798, 2020.
- [5] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," *IEEE Commun. Surv. Tutor.*, vol. 19, no. 3, pp. 1657–1681, 2017.
- [6] X. Li, J. Wan, H. Dai, M. Imran *et al.*, "A hybrid computing solution and resource scheduling strategy for edge computing in smart manufacturing," *IEEE Trans. Ind. Inform.*, vol. 15, no. 7, pp. 4225–4234, 2019.
- [7] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4377–4387, 2019.
- [8] Z. Ning, X. Wang, F. Xia *et al.*, "Joint computation offloading, power allocation, and channel assignment for 5g-enabled traffic management systems," *IEEE Trans. Ind. Inform.*, vol. 15, no. 5, pp. 3058–3067, 2019.
- [9] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint computation offloading and user association in multi-task mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12 313–12 325, 2018.
- [10] Y. Mao, J. Zhang, S. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Trans. Wirel. Commun.*, vol. 16, no. 9, pp. 5994–6009, 2017.
- [11] S. Mao, S. Leng, S. Maharjan, and Y. Zhang, "Energy efficiency and delay tradeoff for wireless powered mobile-edge computing systems with multi-access schemes," *IEEE Trans. Wirel. Commun.*, vol. 19, no. 3, pp. 1855–1867, 2020.
- [12] K. Ahmed, H. Tabassum, and E. Hossain, "Deep learning for radio resource allocation in multi-cell networks," *IEEE Netw.*, vol. 33, no. 6, pp. 188–195, 2019.
- [13] Y. Dai, D. Xu, K. Zhang, S. Maharjan, and Y. Zhang, "Deep reinforcement learning and permissioned blockchain for content caching in vehicular edge computing and networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4312–4324, 2020.
- [14] K. Zhang, Y. Zhu, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Deep learning empowered task offloading for mobile edge computing in urban informatics," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7635–7647, 2019.
- [15] Y. Xie, Z. Xu, J. Xu, S. Gong, and Y. Wang, "Backscatter-aided hybrid data offloading for mobile edge computing via deep reinforcement learning," in *International Conference on Machine Learning and Intelligent Communications*. Springer, 2019, pp. 525–537.
- [16] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang, "Edge intelligence for energy-efficient computation offloading and resource allocation in 5g beyond," *IEEE Trans. Veh. Technol.*, no. , doi: 10.1109/TVT.2020.3013990, 2020.
- [17] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.
- [18] L. Huang and M. J. Neely, "Utility optimal scheduling in energy-harvesting networks," *IEEE/ACM Trans. Netw.*, vol. 21, no. 4, pp. 1117–1130, 2012.

- [19] Y. Li, L. Lei, and M. Yan, "Mobile user location prediction based on user classification and markov model," in *IJCIME*. IEEE, 2019, pp. 440–444.
- [20] W. C. Ao and K. Psounis, "Approximation algorithms for online user association in multi-tier multi-cell mobile networks," *IEEE/ACM Trans. Netw.*, vol. 25, no. 4, pp. 2361–2374, 2017.
- [21] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *ICML*, 2016, pp. 1928–1937.