This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3304969

Scheduling Real-time Wireless Traffic: A Network-aided Offline Reinforcement Learning Approach

Jialin Wan*[1], Sen Lin[1], Zhaofeng Zhang[1], Junshan Zhang[2], and Tao Zhang[3]

[1]Department of ECEE, Arizona State University
[2]Department of ECE, University of California, Davis
[3]National Institute of Standards and Technology

***Abstract*—Real-time traffic has stringent requirements in terms of latency, and deadline guarantees on packet delivery play a vital role in real-time IoT applications. Deadline-aware wireless scheduling of real-time traffic has been a long-standing open problem, despite significant efforts using analytical methods. Departing from the conventional approaches, this work studies deadline-aware traffic scheduling by taking an offline reinforcement learning (RL) approach to train scheduling algorithms, ready to be used for online scheduling. To address the challenges therein, we propose a Network-Aided Offline RL (NA-ORL) framework for deadline-aware scheduling, by making use of the fact that the network dynamics follows a well-defined physics model. Specifically, in NA-ORL the initialization of the scheduling policy is obtained through behavior cloning with a good model-based scheduling algorithm, and the network-aided actor-critic (A-C) method is utilized to train a better scheduling policy with carefully designed states and reward function, thanks to its nature of policy improvement. Building on NA-ORL, we further devise a Network-Aided Offline Meta-RL (NA-MRL) algorithm to deal with the non-stationary network dynamics. Extensive experimental results demonstrate that the proposed NA-ORL and NA-MRL algorithms can achieve better performance over Adaptive Mixing over Non-Dominated links (AMIX-ND) and Largest-Deficit-First (LDF), in various scenarios for the deadline-aware wireless scheduling.**

***Index Terms*—Real-time traffic scheduling, wireless networks, offline reinforcement learning, meta reinforcement learning**

I. Introduction

Recent years have witnessed a tremendous growth in Internet-of-Things (IoT) applications. In real-time IoT applications, intelligent decisions must take place right here right now, in order to meet the requirements for safety, accuracy, latency and user experience. For instance, for connected cars, coordinated sensing and mobility control rely heavily on real-time information exchange among vehicles so as to minimize the uncertainties and corner-cases in perception and control, which has been a notorious safety issue of self-driving in an open environment. Further, both smart health and AR applications require real-time high-definition video streaming. More than 70% of the world's network data is video, including video conferences accelerated by COVID-19, streaming media such as Netflix, and transportation cameras to realize a smart city. Clearly, deadline-aware wireless scheduling is critical and will play a vital role in real-time IoT applications, which has been a long-standing open problem. In general, deadline-aware scheduling can be cast as a Markov Decision Process (MDP) problem, for which the state space tends to grow intractably large quickly, thus making exact approaches to solving it impractical.

Existing analytical studies for deadline-aware wireless scheduling include the frame-based method [1]–[6], the greedy algorithm such as Largest-Deficit-First (LDF) [7], [8], and a very recent work using randomized algorithms, namely Adaptive Mixing over Non-Dominated links (AMIX-ND) [9], which can be regarded as the state-of-the-art scheduling algorithm for real-time traffic. Notably, there has recently been significant efforts using deep reinforcement learning (RL) [10] to solve MDP problems. RL seeks to learn the optimal policy that maximizes a long-term reward by interacting with the environment for the MDP problem, which has achieved astonishing successes in many applications such as robotics [11], [12] and games [13]–[15]. We believe that with the capability of solving sophisticated network optimizations and self-improving through exploration, RL has great potential to provide an alternative approach and yield possibly better solutions to deadline-aware wireless scheduling, compared to existing analytical methods. In light of this, in this work we aim to answer the following key question: *"How to design an efficient RL approach for deadline-aware wireless scheduling to provide reliable low-latency communications services?"*

Designing an efficient RL approach for deadline-aware wireless scheduling is highly nontrivial, due to the following reasons: (1) *Extensive online interactions*: Standard online RL requires extensive interactions with the environment for exploration, which is clearly not applicable in real-time applications; (2) *Unstable performance*: Since the performance of RL intimately depends on the initial policy and reward function, careful designs are required so as to guarantee the performance improvement over the existing methods; and (3) *Possibly non-stationary networks*: The wireless network can be non-stationary due to, e.g., users' arrival/departure, and the underlying MDP problem will change accordingly. It is

*Corresponding author: Jialin Wan, email address: jwan20@asu.edu

therefore important for RL based scheduling algorithms to be able to deal with such a challenging scenario by quickly adapting with the network dynamics.

For ease of exposition, we will focus on the basic single-hop network model (e.g., downlink transmissions from the base station to users) where only one link can be active to transmit packets at each time. To tackle the challenges noted above, we propose NA-ORL, a Network-Aided Offline RL framework for wireless traffic scheduling, based on the fact that the network dynamics follows a well-defined physics model. Specifically, NA-ORL initializes the scheduling policy with a base policy obtained by AMIX-ND, and further improves the policy via the network-aided actor-critic (A-C) method with carefully designed states and reward function. Compared to existing approaches, NA-ORL can learn a better scheduling policy in an offline manner, which is ready to be used for online scheduling of real-time traffic. Building on NA-ORL, we further propose a Network-Aided Offline Meta-RL (NA-MRL) framework to deal with the non-stationary network dynamics. Our main contributions can be summarized as follows:

- By casting the deadline-aware wireless scheduling problem as an MDP problem, we propose NA-ORL, an efficient offline RL approach, to learn a scheduling policy offline for stationary networks, based on a network-aided A-C method. The A-C method [16], [17] consists of a policy evaluation structure (Critic) and a policy improvement structure (Actor), where the Critic computes Q-values to evaluate the current policy and the Actor aims to improve the policy based on the evaluation of the Critic. In particular, NA-ORL initializes the policy for the Actor via behavior cloning with the base policy obtained by AMIX-ND. Through a careful design of the A-C method, including states, the reward function and the sampling procedure, NA-ORL can obtain a better scheduling policy over AMIX-ND, thanks to the nature of policy improvement of the A-C method.
- For the challenging scenario with non-stationary network dynamics, we cast the scheduling under different network dynamics as a unified-MDP problem with multiple different MDP sub-problems (each as an offline RL task), and devise NA-MRL to learn a meta scheduling policy offline by jointly training with multiple offline RL tasks building upon NA-ORL. More importantly, a *task-specific mask* is designed for the meta-policy to capture the network dynamics. The scheduling policy for a new task can be then quickly adapted from the meta scheduling policy given the network structure.
- We conduct extensive experiments to evaluate the performance of both NA-ORL and NA-MRL. Compared with AMIX-ND [9] and LDF [7], [8], our experimental results demonstrate that the proposed network-aided offline RL algorithms can achieve better performance under various scenarios for the deadline-aware wireless scheduling.

In the remainder of the paper, we provide a brief review of related work in Section II. We introduce in Section III the system model and problem formulation. In Section IV, we present the design details of the proposed NA-ORL scheduling algorithm for stationary network dynamics. In Section V, the proposed approach is extended to addressing non-stationary network dynamics and NA-MRL algorithm is devised accordingly. Experimental results for both stationary and non-stationary network dynamics are presented in Section VI. Finally, the conclusions and future work are discussed in Section VII.

II. Related work

In this section, we briefly review existing works related to deadline-aware wireless scheduling. A frame-based approach was first proposed in [1] and generalized in [2]–[6]. The approach assumed that all packets arrive at the beginning of a frame and must be scheduled before the end of the frame, otherwise they would be discarded. LDF is another popular algorithm proposed by [7], [8], which greedily selects the active links with largest deficit. The above algorithms indeed have very low complexity and can guarantee a lower bound of efficiency ratio. Nevertheless, they might not be suitable for high throughput real-time applications. Building on the LDF algorithm, [9] further proposed a randomized algorithm, namely AMIX-ND, to achieve a better efficiency ratio. Specifically, [9] defined a dominance order according to the deficit and earliest deadline of each link, calculated the probabilities of each link to be active, and then selected randomly a set of links to transmit the packet with earliest deadline in their buffer according to the corresponding probability. This method can achieve better performance, but still leaves much room for improvement, as shown in our experiments. To our best knowledge, this work is the first attempt to develop offline RL-based algorithm for deadline-aware wireless scheduling, which can deal with sophisticated network dynamics and obtain better scheduling policies through interactions with the well-defined physical model.

III. System Model and Problem Formulation

In this section, we present the system model and the problem formulation of deadline-aware wireless scheduling.

Wireless network model. As illustrated in Fig. (1), we consider a collocated network with a single base station (transmitter) and a set of L users. There exists a link between each user and the base station, and the set of links is denoted by $\mathcal{L} = \{1, \cdots, L\}$. In a collocated network, only one link can be active to transmit packet at any time slot $t \in \mathbb{N}_0$.

Traffic model. Consider a single-hop traffic with deadlines $d \in \{1, \cdots, d_{max}\}$ for each link $l \in \mathcal{L}$, as shown in Fig. 2. Let $\tau_{l,d}(t)$ denote the number of packets with deadline d arriving at link l during time slot t. Packets would expire and be discarded if not delivered before the deadlines. Then the arrival packets at link l during time t can be denoted by a vector $\tau_l(t) = (\tau_{l,d}(t); d = 1, \cdots, d_{\max})$, and for the entire network traffic it is given by by $\tau(t) = (\tau_l(t); l \in \mathcal{L})$. The traffic arrival pattern can be some random process and do not need to be independent and identically distributed across links.

Buffer dynamics. For each link l, there exists a buffer that contains the packets at that link which have not expired. At time slot t, the number of remaining packets in the buffer

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3304969

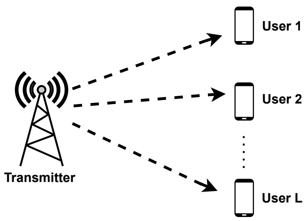


Fig. 1: Illustration of collocated network with single-hop traffic.

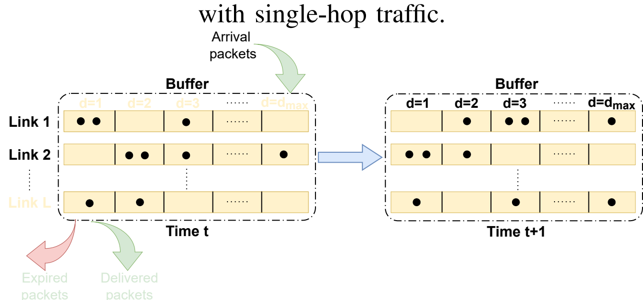


Fig. 2: Illustration of dynamic packet transmissions with stringent deadlines.

within deadline d at link l is denoted by $\Psi_{l,d}(t)$, and the buffer dynamics can be shown as

$$\Psi_{l,d}(t+1) = \Psi_{l,d+1}(t) + \tau_{l,d}(t+1) - I_{l,d+1}(t), \quad (1)$$

where action $I_{l,d}(t) = 1$ represents that link l is transmitting a packet with remaining deadline d at time slot t, and $I_{l,d}(t) = 0$ otherwise. Then the number of remaining packets at link l can be denoted by $\Psi_l(t) = (\Psi_{l,d}(t); d = 1, \cdots, d_{\max})$. The network buffer state can be defined as $\Psi(t) = (\Psi_l(t); l \in \mathcal{L})$. The action vector is denoted by $I(t) = (I_{l,d}(t); l \in \mathcal{L}, d = 1, \cdots, d_{\max})$. For convenience, we further define

$$e_l(t) = \min\{d \colon \Psi_{l,d}(t) > 0\} \quad (2)$$

as the earliest deadline of packets at link l at time slot t, which can be derived from the buffer information $\Psi(t)$. Let $e(t) = (e_l(t); l \in \mathcal{L})$ be the earliest deadlines for all links.

Packet delivery requirement and deficit. Similar to [7], [8], we can define deficit $w_l(t)$ to measure the amount of service owned to link l until time t to fulfill its delivery ratio requirement p_l^0, and

$$w_l(t+1) = \left[w_l(t) + \widetilde{v}_l(t) - I_l(t)\right]^+, \quad (3)$$

where $\widetilde{v}_l(t) = v_l(t)p_l^0$, $v_l(t) = \sum_{d=1}^{d_{max}} \tau_{l,d}(t)$ and p_l^0 is the QoS requirement of link l in terms of packet delivery ratio. At time slot t, the system can be described by the tuple of buffer information and deficit, i.e., $(\Psi(t), w(t))$ where $w(t) = (w_l(t); l \in \mathcal{L})$.

Problem formulation. Let $TA_l(t)$ and $TD_l(t)$ be the total number of arrival packets and total number of delivered packets on links l until time t, respectively. Then it holds that

$$TA_l(t) = TA_l(t-1) + \sum_{d=1}^{d_{max}} \tau_{l,d}(t), \quad (4)$$

$$TD_l(t) = TD_l(t-1) + I_l(t). \quad (5)$$

Define $p_l(t) = \frac{TD_l(t)}{TA_l(t)}$ as the achieved delivery ratio on link l until time t. Given a collocated network model and traffic pattern $\tau(t) = (\tau_l(t); l \in \mathcal{L})$, the primary objective is to find an optimal policy π to schedule links $l \in \mathcal{L}$ to be active or inactive at each time slot t, such that the overall performance is optimized, given the QoS requirements. In particular, we seek to maximize the minimal normalized delivery ratio:

$$\max_{\pi} J(\pi) = \max_{\pi} \mathbb{E}_{\pi}\left[\min_{l \in \mathcal{L}} \frac{p_l(T)}{p_l^0}\right], \quad (6)$$

which is closely related to the *efficiency ratio* [9] that measures the fraction of the real-time throughput region guaranteed by the algorithm.

IV. NA-ORL: A Network-Aided Offline RL Approach for Stationary Network Dynamics

Unlike the standard MDP problems, the random packet arrivals add exogenous dynamics complicate the underlying MDP, calling for innovative RL algorithms. Fortunately, the network dynamics follows the well-defined physical model Eq. (1) - Eq. (5), which enables an accurate simulation of the real system corresponding to a given network structure. With this insight, we propose a network-aided offline RL approach (NA-ORL) to learn the scheduling policy in an offline manner, inspired by AlphaGo [13]. In particular, NA-ORL mainly consists of two phases as illustrated in Fig. 3: (1) initialization of the policy (actor) via behavioral cloning, where the base policy is obtained based on AMIX-ND, and (2) policy improvement via the network-aided A-C method.

A. Deadline-aware Wireless Scheduling as an MDP Problem

In what follows, we first treat the deadline-aware wireless scheduling problem as an MDP defined by $(\mathcal{S}, \mathcal{A}, P, r)$, with state space $\mathcal{S}$, action space $\mathcal{A}$, state transition probability $P \colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, and stage reward $r \colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$.

State. Following the same line as in AMIX-ND, where the randomized scheduling policy is obtained based on the deficits $w(t)$ and the earliest deadlines $e(t)$ for all links, we define the system state $s_t \in \mathcal{S}$ at time t as follows:

$$s_t = [w(t), e(t)]. \quad (7)$$

Note that an important issue here is that different elements in s_t can be on different scales, particularly the deficit $w_l(t) \in \mathbb{Z}$ vs. the earliest deadline $e_l(t) \in \{1, .., d_{max}\}$, which may lead to unstable learning process for deep neural networks [18] if not handled in the correct manner. To address this, we normalize the elements in $s(t)$ using a sigmoid function, i.e., for each element x in s_t, the corresponding normalized value is

$$f(x) = \frac{1}{1 + exp(-x)}$$

where $f(x) \in (0, 1)$.

Action. As in a collocated network, only one link can be activated at each time slot. The action at time t can be then denoted by a discrete value

$$a_t \in \{1, 2, \cdots, L\}, \quad (8)$$

where $a_t = l$ indicates that only the l^{th} link is activated at time t and all other links are inactivated.

State transition probability. Unlike the typical MDP problems, the random packet arrivals add exogenous dynamics and hence complicate the underlying MDP, making it nontrivial to

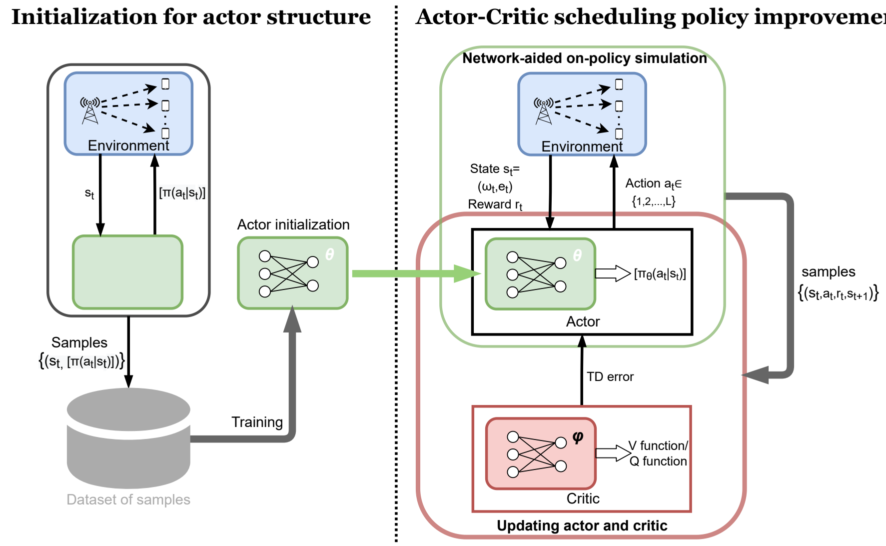


Fig. 3: Network-aided offline RL (NA-ORL) for scheduling real-time wireless traffic.

write down the state transition probability explicitly. Instead, we here utilize the physical model Eq. (1) - Eq. (5) that can be used to construct an accurate model simulator for offline interactions when learning the scheduling policy.

Reward. It is clear that the performance of RL closely hinges upon the design of the stage reward function, which serves as an important signal for evaluating and reinforcing the action selections. In this work, we design the reward function r_t as the change of the minimal normalized delivery ratio across all links from time $t-1$ to t:

$$r_t(s_t, a_t) = \min_{l \in \mathcal{L}} \frac{p_l(t)}{p_l^0} - \min_{l \in \mathcal{L}} \frac{p_l(t-1)}{p_l^0}. \tag{9}$$

Note that the instant reward r_t depends on the state s_t and the action a_t implicitly through the value of the achieved delivery ratio up to time t. Intuitively, given a system state s_t, the more the action improves the minimal normalized delivery ratio over the network, the higher reward it will achieve. Such a reward design has also captured the impact of the random packets arrival based on Eq. (4).

Scheduling policy. Suppose π_θ is the scheduling policy parameterized by θ, which maps the current system state to the probability vector of link activation. The MDP problem is to find the optimal probability vector $[\pi_\theta(a_t = 1|s_t), \pi_\theta(a_t = 2|s_t), \cdots, \pi_\theta(a_t = L|s_t)]$ maximizing the expected cumulative rewards:

$$\max_{\pi_\theta} J(\pi_\theta) = \max_{\pi_\theta} \mathbb{E}_{\pi_\theta}\Big[\sum_{t=1}^{T} r_t\Big]. \tag{10}$$

It is clear that the objective (10) is indeed equivalent to (6).

To efficiently solve the MDP problem (10), we resort to the popular A-C method [19], where the Critic uses a policy evaluation structure to compute the Q-values under the current policy being followed by the Actor; and the Actor aims to improve the policy based on the evaluation of the Critic. Neural networks are utilized to parameterize both the Actor and the Critic structures, and the two structures work in concert by updating the parameters of these two neural networks iteratively. With carefully designed states and reward function, the A-C method can be utilized to train a better scheduling policy, thanks to its nature of policy improvement.

B. Policy Initialization via Behavioral Cloning

In light of the nature of policy improvement of the A-C method, we first initialize the actor with the base policy obtained by AMIX-ND (which has been shown in [9] to achieve the largest *efficiency ratio*), which not only stabilizes the learning process but also leads to a better scheduling policy eventually. Towards this end, we seek to learn a base policy that imitates the behavior of AMIX-ND.

Behavioral cloning is one of the most popular methods to tackle an imitation learning problem [20], [21] through supervised learning. Therefore, to 'imitate' the behavior of AMIX-ND, we first use the AMIX-ND algorithm to generate abundant samples offline, and leverage behavior cloning to learn the base policy. More specifically, for the wireless network model with a given traffic pattern, we can calculate the probability of each link to be active at each time slot, according to the AMIX-ND algorithm, and choose one link to transmit packets based on the probability vector. As shown in Fig. 3, a training sample which consists of state information s_t and probability vector $[\pi_\theta(a_t = 1|s_t), \pi_\theta(a_t = 2|s_t), \cdots, \pi_\theta(a_t = L|s_t)]$ is then collected through the offline interaction with the physical model simulator, and stored in the training dataset. Multiple trajectories might be generated to obtain enough samples for the training process. The training dataset can then be used to learn the base policy from scratch via standard supervised learning. Once the supervised training process is completed, a base policy that behaves similar to the original AMIX-ND algorithm can be obtained as the initialization of Actor. Initialization through behavior cloning with a good scheduling algorithm can reduce the computation cost significantly.

C. Policy Improvement via Network-aided Actor-Critic

Based on the policy initialization mentioned earlier, we next propose a network-aided offline A-C method for policy

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3304969

Algorithm 1 NA-ORL: network-aided offline RL algorithm for real-time traffic scheduling

```
Initialize actor network π_θ with the base policy and N critic networks Q_{φ_i}, i = 1, ···, N.
Initialize target networks φ'_i ← φ_i, i = 1, ···, N.
C: number of synchronized parallel A-C agents.
K: number of samples by each A-C agent at a time slot.
𝒟 ← ∅.
for each environment step t do
    for each A-C agent c = 1, ···, C do
        for k = 1, ···, K do
            Take action a_t^{ck} ~ π_θ(· | s_t^c), observe reward r_t^{ck}, new state s_{t+1}^{ck}.
            𝒟 ← 𝒟 ∪ {(s_t^{ck}, a_t^{ck}, r_t^{ck}, s_{t+1}^{ck})}.
        end for
    end for
    for G updates do
        Mini-batch ℬ = {(s, a, r, s')} ⊂ 𝒟.
        Sample a set ℳ of M distinct indices from {1, ···, N}.
        Compute the Q target:
            y = r + γ ( min_{i∈ℳ} Q_{φ'_i}(s', ã') − α log π_θ(ã' | s') ),
            ã' ~ π_θ(· | s').
        for i = 1, ···, N do
            Update critic network Q_{φ_i} with gradient descent using
                ∇_{φ_i} (1/|B|) Σ_{(s,a,r,s')∈B} (Q_{φ_i}(s, a) − y)^2.
            Update target critic network Q_{φ'_i}: φ'_i ← (1−ρ)φ'_i + ρφ_i.
        end for
    end for
    Update actor network π_θ with gradient descent:
        −∇_θ (1/|B|) Σ_{s∈B} ( (1/N) Σ_{i=1}^{N} Q_{φ_i}(s, ã_θ(s)) − α log π_θ(ã_θ(s) | s) ),
        ã_θ(s) ~ π_θ(· | s).
end for
```

improvement. In particular, to fully unleash the potential of the physical model and improve the learning performance, we introduce a new method of data collection through offline interaction with the physical model, and leverage an ensemble of Q-functions to deal with the well-known overestimation problem in the A-C method [22]–[26]. In what follows, we present the details of the proposed method. After the offline training is completed, the policy can be directly deployed for online scheduling without additional updates, making it suitable for real-time scheduling.

Data collection: experience replay and on-policy samples via parallel A-C. Only using on-policy samples generated by rolling out the current policy from the current state s_t may suffer from strong sample correlation [27] in A-C-based algorithms, resulting in inaccurate Q-value estimations. To address this issue, we propose to collect on-policy samples with multiple parallel A-C agents for the current policy, by taking advantage of the physical model defined in Eq. (1) - Eq. (5). As shown in Fig. 4, our method builds up multiple physical model simulators, and for each simulator there is one A-C agent to collect on-policy samples by rolling out the current policy with the simulator. Note that all A-C agents share the same policy parameters but can start from different system states. More specifically, suppose that the current state of an agent c is s_t^c. Each A-C agent repeatedly generates a few on-policy samples $\{(s_t^c, a_t^c, r_t^c, s_{t+1}^c)\}$ (represented as (s, a, r, s') when no confusion occurs) at current time slot t using the current policy. All of the samples generated at current time slot are stored in the experience replay dataset $\mathcal{D}$. After that, each agent would transit to its own next state and repeat the process.

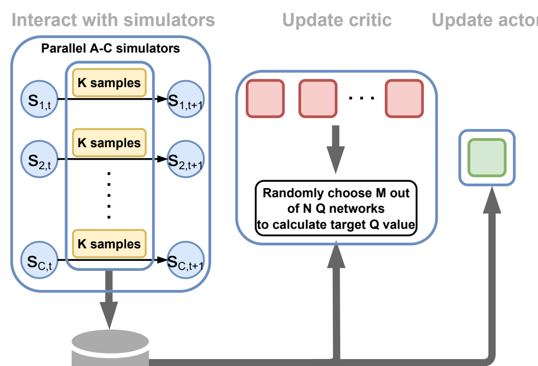


Fig. 4: Illustration of sample collection with parallel A-C simulators.

Addressing overestimation of Q function. Clearly, the performance of the deadline-aware scheduling policy depends on the policy improvement of the A-C algorithm, which hinges heavily upon the accuracy of the Q-values estimated by the critic. It is known that the Q-value estimation often suffers from overestimation bias when evaluating the target Q-value, and recent works [22], [23], [26], [28] have proposed to use an ensemble of independent Q-value estimators to reduce the overestimation bias. Therefore, in this work we consider a set of N critic networks denoted by $Q_{\phi_1}, \cdots, Q_{\phi_N}$, respectively. As shown in Fig. 4, once the sample collection is completed for the current policy, our method randomly chooses M out of N critic networks, and estimates the target Q-value for a sample (s, a, r, s') using the minimum among M Q-value estimations:

$$y = r + \gamma \left(\min_{i \in \mathcal{M}} Q_{\phi'_i}\left(s', \tilde{a}'\right) - \alpha \log \pi_\theta\left(\tilde{a}' \mid s'\right) \right), \tilde{a}' \sim \pi_\theta\left(\cdot \mid s'\right),$$

where γ is the discount factor, $\mathcal{M}$ is the set of indices of critic networks sampled from $\{Q_{\phi_1}, \cdots, Q_{\phi_N}\}$. Here $Q_{\phi'_i}$ is a target critic network for solving the moving target problem [19], and α is the temperature parameter that determines the relative importance of the entropy term against the reward, which controls the stochasticity of the optimal policy [19], [29]. Note that y serves as the common target Q-value for all N critic networks.

Critic update. To obtain an accurate estimation of the Q-values for the current policy, we update each critic network $Q_{\phi_i}, i \in \{1, 2, \cdots, N\}$ towards the common target Q-value y. This can be achieved by using gradient descent to minimize the mean squared error (MSE) loss over a batch $\mathcal{B}$ of samples from the replay buffer:

$$\min_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s') \in \mathcal{B}} \left(Q_{\phi_i}(s, a) - y\right)^2,$$

where $Q_{\phi_i}(s, a)$ is the estimated Q-value for taking action a at current state s. The target critic network $Q_{\phi'_i}$ can be then updated softly by a Polyak factor ρ as shown in Algorithm 1.

Actor update. After every G updates of critic networks, the actor network π_θ can be updated with gradient descent as shown in step 22 of Algorithm 1, to further improve the policy as in [19] by solving the following problem:

$$\max_{\theta} \frac{1}{|B|} \sum_{s \in \mathcal{B}} \left(\frac{1}{N} \sum_{i=1}^{N} Q_{\phi_i}\left(s, \tilde{a}_\theta(s)\right) - \alpha \log \pi_\theta\left(\tilde{a}_\theta(s) \mid s\right) \right), \quad \tilde{a}_\theta(s) \sim \pi_\theta(\cdot \mid s),$$

where the entropy term serves as a normalization in the total loss function based on the average Q-value estimate by each critic network Q_{ϕ_i}. Note that the fact that an discrete action a_t is sampled from the categorical distribution $[\pi_\theta(a_t = 1|s_t), \pi_\theta(a_t = 2|s_t), \cdots, \pi_\theta(a_t = L|s_t)]$ would introduce the non-differentiability issue when computing the policy gradient through backpropagation in neural networks. To solve the problem, we adopt an efficient gradient estimator that replaces the non-differentiable sampling from a categorical distribution with a differentiable sampling from a novel Gumbel-Softmax distribution [30].

The critic and actor updates alternatively until the policy converges. More details are outlined in Algorithm 1.

V. NA-MRL: A Network-Aided Meta-RL approach for Non-stationary Network Dynamics

Next, we consider a more challenging scenario where the network dynamics could be non-stationary on a larger timescale, e.g., the number of links in the wireless network changes due to users' arrival/departure or the traffic pattern changes. In general, we can treat the scheduling problem in a stationary environment as an MDP and its formulation changes when the underlying network dynamics changes. Clearly, the policy learnt offline for one specific MDP would not work well for a different MDP. Needless to say, the nature of deadline-aware scheduling dictates that it is infeasible to retrain new policy for each new MDP from scratch.

Meta-RL [31] has recently emerged as a promising approach to quickly solve a new RL task using samples from that task, by exploiting shared structures among related RL tasks during offline meta-training. The superior performance of meta-RL, in terms of sample efficiency and higher rewards, has been demonstrated in the literature [32]–[35], when compared with standard RL methods that learn from scratch [36], [37]. Thus motivated, we will resort to meta-RL to tackle the distribution shift, by learning a scheduling policy that can quickly adapt to non-stationary network dynamics.

Different from standard meta-RL problems where the task identity has to be learnt through the interaction with the environments, the system operator can construct different physical model simulators offline by modifying the network topology accordingly, thanks to the mapping between the network topology and the MDP model. Inspired by the above observation, we propose NA-MRL, a two-stage meta-RL algorithm for deadline-aware scheduling in the presence of distribution shift, including (1) offline meta training based on multiple physics model simulators corresponding to different network models; and (2) on-policy adaptation of the scheduling policy for a new network model. In the following, we first introduce the formulation of an unified MDP which takes the non-stationary network dynamics into consideration, and then present NA-MRL in details as outlined in Algorithm 2.

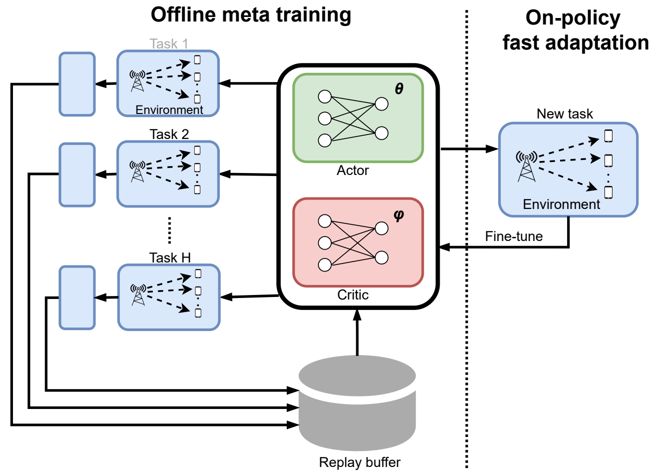


Fig. 5: Illustration of NA-MRL for deadline-aware wireless scheduling for non-stationary network.

Algorithm 2 NA-MRL: network-aided meta-RL algorithm for non-stationary network dynamics

Maximum number of links: L_{max}.
Initialize actor network π_θ and critic networks Q_ϕ.
for task $h \in [1, \cdots, H]$ **do**
    Current set of links: $\mathcal{L}_h \subset \mathcal{L}_{max}$.
    Initialize state $s_t \in \mathcal{S}$, where $s_t(l) = 0, \forall l \notin \mathcal{L}_h$.
    Initialize action $a_t \in \mathcal{A}$, where $a_t(l) = 0, \forall l \notin \mathcal{L}_h$.
    Agent interacts with simulated environment and updates policy π_θ using Algorithm 1.
end for
Save meta-policy π_θ^0 for future quick adaptation.
while number of links changes due to users' arrival/departure **do**
    Initialize state $s_t \in \mathcal{S}$ and action $a_t \in \mathcal{A}$, using the binary mask based on the network topology.
    Initialize policy $\pi_\theta = \pi_\theta^0$.
    Update policy π_θ in an on-policy manner.
    Evaluate policy.
end while

Unified MDP formulation. Without loss of generality, consider non-stationary network dynamics where the set $\mathcal{L}$ of links could change after a period of time. Within a period k, the network is stationary, and the scheduling problem can be cast as an offline RL task represented by the MDP $M_k = (\mathcal{S}_k, \mathcal{A}_k, P_k, r_k)$. Let $\mathcal{L}_k$ denote the set of links in period k and $|\mathcal{L}_k| = L_k$. Clearly, the state space $\mathcal{S}_k$ and the action space $\mathcal{A}_k$ depends on the number of links in $\mathcal{L}_k$. For simplicity, we assume that there exists a set $\mathcal{L}_{max}$ such that $\mathcal{L}_k \subset \mathcal{L}_{max}$ for any period k, and $|\mathcal{L}_{max}| = L_{max}$. We define a unified state space $\mathcal{S} = (S(1), ..., S(L_{max}))$ where $S(l)$ is the state for link $l \in \mathcal{L}_{max}$, and a unified action space $\mathcal{A} = \{1, ..., L_{max}\}$. We can reformulate each M_k as a new MDP $\tilde{M}_k = (\mathcal{S}, \mathcal{A}, \tilde{P}_k, r_k)$ by making the following changes:

1) When learning a scheduling policy for $\tilde{M}_k$, we restrict the support of actions to only a subset of $\mathcal{A}$ such that the action $a_t = l$ for $l \notin \mathcal{L}_k$ must not be selected;

2) Given a selected action a_t at time t, the state transition distribution P_k is modified to $\tilde{P}_k$ such that the dimension $S(l)$ in the state for $l \notin \mathcal{L}_k$ is always 0. This can be achieved by setting the transition probability $p(s_{t+1}|s_t, a_t) = 0$ for any state s_{t+1} with non-zero entries in $S_{t+1}(l)$ if $l \notin \mathcal{L}_k$.

As a result, the modified MDP encompasses $\{\tilde{M}_k\}$ for all periods, with the same unified state and action space, but may have different state transition distributions and reward functions.

Offline meta-training. As illustrated in Fig. 5, the system operator can build H offline RL training tasks $\{\tilde{M}_h\}_{h=1}^H$ where each task h corresponds to one network topology. Given the initialized actor network π_θ and critic networks $Q_{\phi_i}, i \in \{1, 2, \cdots, N\}$, we define *a mask as a binary vector of length L_{max} for each task $h \in \{1, 2, \cdots, H\}$ with $|\mathcal{L}_h| = L_h$ links, which maps the unified state s_t and action a_t to the task-specific state and action based on the network topology*. The i^{th} entry in the mask is 1 if $i \in \mathcal{L}_h$, otherwise the i^{th} entry is 0. For example, the mask is $[0, 1, 1, 0, 1, 0]$ for task h with set of links $\mathcal{L}_h = \{2, 3, 5\}$ and $L_{max} = 6$. The objective of offline meta-training is to learn a meta-policy π_θ^0 that performs well across all training tasks by solving the following problem:

$$\max_{\pi_\theta^0} \quad \mathbb{E}_{\pi_\theta^0}\left[\sum\nolimits_{h=1}^{H}\sum\nolimits_{t=1}^{T} r_{h,t}\right]. \tag{11}$$

This can be solved by continuously updating the meta-policy based on NA-ORL (Algorithm 1) through offline interactions with the physical model simulators for each training task.

On-policy fast adaptation. Given the meta-policy π_θ^0 obtained after offline meta-training, we next quickly adapt it to learn a task-specific policy for a new task. Specifically, since the network topology change is known, which determines the MDP model for the new task, the corresponding mask can be then determined for the new task with set of links $\mathcal{L}_k$. A scheduling policy can be quickly obtained by fine-tuning the meta-policy π_θ^0, through the interactions with the physical model of the current task based on NA-ORL. After the fast adaption is completed, the policy can be directly deployed for online scheduling, making it suitable for real-time scheduling for the new task.

TABLE I: Hyperparameters.

optimizer	Adam
learning rate	$3 \cdot 10^{-4}$
discount factor (γ)	0.99
number of links (L)	2, 5
max deadline (d_{max})	10
arrival traffic pattern	Poisson distribution
number of parallel A-C agents (C)	1
samples generated by each agent (K)	2000
batch size	128
replay buffer size	10^4
non-linearity	ReLU
number of hidden layers	1
number of hidden units per layer	8
target smoothing coefficient (ρ)	0.005
number of approximators (N)	2
ensemble size (M)	2
SAC entropy hyperparameter (α)	0
Gumbel Softmax parameter (τ)	0.01

VI. Experimental Studies

In this section, we first evaluate the performance of the proposed NA-ORL algorithm for the case with stationary network dynamics, and then investigate the performance of NA-MRL for the case with non-stationary network dynamics. In both cases, LDF [7], [8] and the most recent algorithm AMIX-ND [9] serve as the baselines, in which the performance has been evaluated in terms of *efficiency ratio*, i.e., the fraction of the real-time throughput region where the delivery ratio requirements are satisfied. In the same spirit, we compare our methods with AMIX-ND and LDF in terms of the following performance metric:

$$\max \min_{l \in \mathcal{L}} \frac{p_l(T)}{p_l^0}, \tag{12}$$

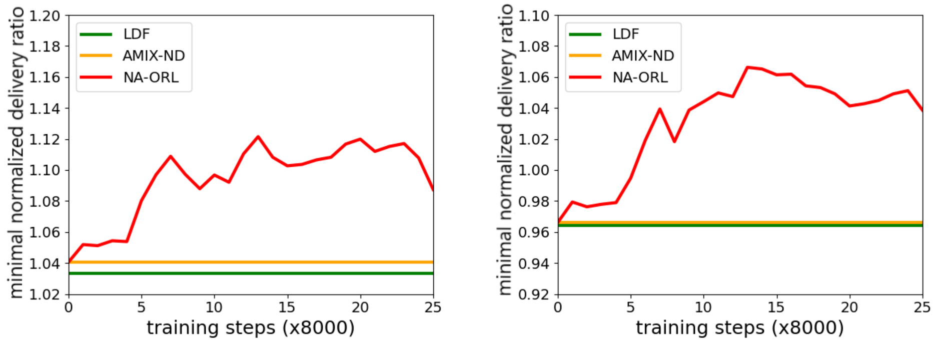
which evaluates the maximum of the minimal normalized delivery ratios among all links. We consider the collocated network setting as shown in Fig. 1 with different number of links. For each link $l \in \mathcal{L}$, the arrival pattern is determined by a Poisson process with arrival rate $\vec{\lambda}$, and the QoS requirement is p_l^0, which is the required minimum delivery ratio for that link. Hyperparameters used in the algorithms are listed in Table I.

A. Case Study with Stationary Network Dynamics

For a network with stationary dynamics, we consider the cases with different number of links, e.g., the link number $L \in \{2, 5\}$. The complexity would increase with more links for deadline-aware traffic scheduling in real world, which would be taken into account in future work. More specifically, in the first case with 2 links, the arrival rate is $\vec{\lambda} = [0.75, 0.75]$, while $\vec{\lambda} = [0.3, 0.3, 0.3, 0.3, 0.3]$ in the second case with 5 links. Intuitively, the number of arrival packets in the system is 1.5 per time slot on average, equally shared by all the links. The QoS requirements are different across links. For each case, we first run the AMIX-ND algorithm to collect samples, which would be used to train an initial policy via behavior cloning. In particular, we run the AMIX-ND algorithm for 500 episodes of length 2000, and collect 1 million samples in total as the training dataset. Here we consider a neural network with one hidden layer of size 8 for both actor network and critic network. The learning rate of behavior cloning is chosen to be 3×10^{-4}. The number of training steps is 10000. During each step, a batch of 128 samples are sampled randomly from the training dataset to update the actor network. Once the policy initialization is completed, the next step is to update the initial policy (i.e., the actor network) and Q-value estimators (i.e., the critic networks) iteratively through offline interactions with the physical model. At the beginning of each training step, 2000 on-policy samples are generated from 100 episodes of length 20 using the current policy and stored in the experience replay buffer. Then the critic networks and the actor network will be updated using a batch of samples from the replay buffer. We evaluate the learnt policy every 8000 training steps, by directly applying the policy for online scheduling, with the performance metric defined in Eq. (12). All evaluation results are averaged over 10 runs with episodes of length 10000.

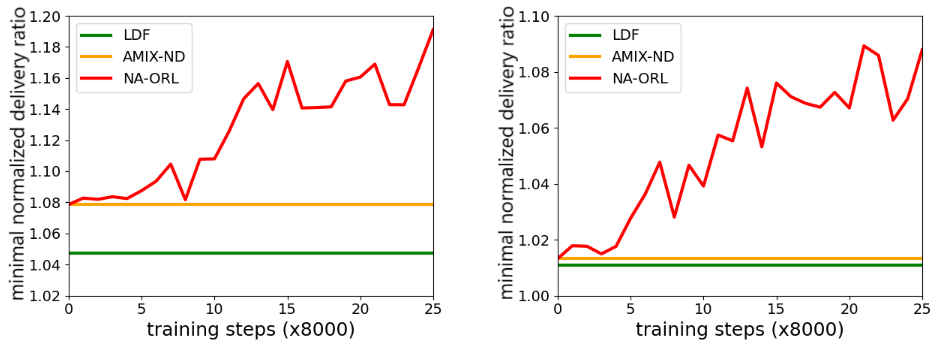
Fig. 6 and Fig. 7 illustrate the performance comparison among NA-ORL, AMIX-ND and LDF for the networks with 2

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3304969



(a) QoS requirements: $[0.7, 0.4]$. (b) QoS requirements: $[0.8, 0.3]$.

Fig. 6: Performance comparison NA-ORL against AMIX-ND and LDF: the network with 2 links of identical arrival rates $[0.75, 0.75]$ and different QoS requirements.



(a) QoS requirements: $[0.7, 0.7, 0.4, 0.4, 0.4]$. (b) QoS requirements: $[0.8, 0.8, 0.3, 0.3, 0.3]$.

Fig. 7: Performance comparison NA-ORL against AMIX-ND and LDF: the network with 5 links of identical arrival rates $[0.3, 0.3, 0.3, 0.3, 0.3]$ and different QoS requirements.

links and 5 links, respectively. Note that for all cases, AMIX-ND outperforms LDF, in line with the results in [9]. For the case of 2 links with same arrival rates and different QoS requirements, it is clear that NA-ORL performs about 10% better than AMIX-ND after 80000 training steps. In particular, in Fig. 6b, the performance of AMIX-ND is less than 1 while the performance of NA-ORL is over 1, which indicates that the system is more stable with NA-ORL. Note that the system is stable only when the performance is over 1, otherwise the QoS requirements can not be reached and the deficit may blow up. For the case of 5 links with same arrival rates and different QoS requirements, NA-ORL performs about 8% better than AMIX-ND. The superior performance of NA-ORL clearly corroborates the benefits of leveraging network-aided RL to solve the real-time scheduling problem with complicated network dynamics.

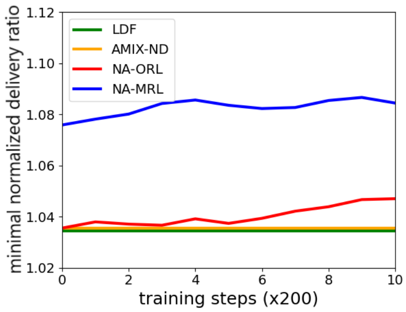
B. *Case Study with Non-stationary Network Dynamics*

We next consider the network setting with non-stationary dynamics, where NA-MRL is designed to quickly adapt to new task from an offline trained meta-policy π_θ^0. Here we set the maximum number of links $L_{max} = 6$, and consider three meta-training tasks with the link number $L_h \in \{2, 4, 6\}$. For a certain task with $|\mathcal{L}_h| = L_h$ links, the i^{th} entry in the mask is 1 if $i \in \mathcal{L}_h$, otherwise the i^{th} entry is 0. The experiments are carried out in 2 cases: (1) same arrival rates and different QoS requirements; (2) different arrival rates and same QoS requirements. In the first case, arrival rates and QoS requirements are set to be $\vec{\lambda} = [0.3, 0.3, 0.3, 0.3, 0.3, 0.3]$ and $[0.3, 0.3, 0.6, 0.6, 0.9, 0.9]$, respectively, while in the second case, $\vec{\lambda} = [0.7, 0.7, 0.3, 0.3, 0.1, 0.1]$ and QoS requirements are $[0.6, 0.6, 0.6, 0.6, 0.6, 0.6]$. Here we run the offline meta-training process up to 60000 steps to obtain the meta-policy π_θ^0, and then fine-tune the meta-policy for 2000 steps so as to learn a better policy for a new task with mask $[0, 1, 1, 0, 1, 0]$.

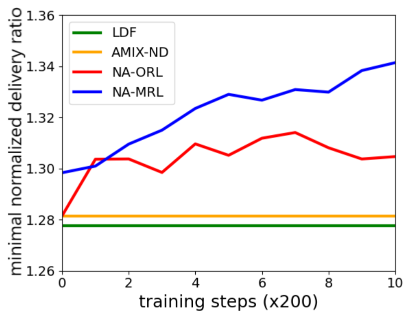
Fig. 8 shows the performance of NA-MRL and NA-ORL under above settings. As expected, in both cases, with a good meta-policy π_θ^0 (which is better than AMIX-ND), NA-MRL not only outperforms AMIX-ND and LDF, but also achieves better performance than NA-ORL after quick adaptation. The results demonstrate the superiority of our proposed NA-MRL approach in solving the scheduling problem with non-stationary dynamics by leveraging the similarity across multiple offline training tasks, and indicate that the learnt meta-policy indeed serves as a better policy initialization for quick adaptation on new tasks.

Ablation study of NA-ORL vs. DQN. Using the same setting as in Fig. 6a and Fig. 7a, we next compare the performance of NA-ORL against the well-known Deep Q-Network (DQN) approach [38], [39]. As shown in Fig. 9, NA-ORL can achieve better performance than DQN algorithm in both cases with 2 and 5 links, thanks to the nature of policy improvement of the A-C method.

Ablation study of NA-MRL. We conduct an ablation study to

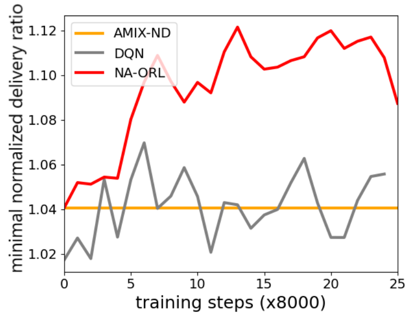


(a) Same arrival rates and different QoS requirements.

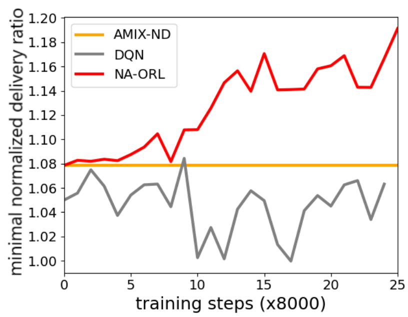


(b) Different arrival rates and same QoS requirements.

Fig. 8: Performance evaluation of NA-MRL under non-stationary network dynamics.

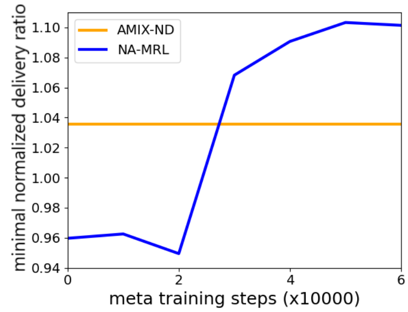


(a) QoS requirements: $[0.7, 0.4]$.

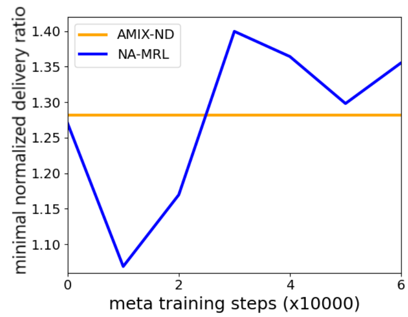


(b) QoS requirements: $[0.7, 0.7, 0.4, 0.4, 0.4]$.

Fig. 9: Ablation study of NA-ORL algorithm.



(a) Same arrival rates and different QoS requirements.



(b) Different arrival rates and same QoS requirements.

Fig. 10: Ablation study of NA-MRL algorithm.

analyze the meta-training process of the NA-MRL approach. In particular, we evaluate the performance of the learnt meta-policy after every 10000 meta-training steps, by studying the scheduling performance of the task-specific policy that is adapted from the meta-policy after 2000 steps of gradient updates. Fig. 10 shows the meta-training performance for the 2 cases regarding the arrival rates and the QoS requirements. In both cases, after enough meta training steps, NA-MRL can achieve a better policy than AMIX-ND after quick adaptation.

VII. Conclusions and Future Work

In this work, we study the deadline-aware wireless traffic scheduling by taking an offline RL approach to train scheduling policies, which is ready to be used for online scheduling. To tackle the challenges therein, we propose NA-ORL, a network-aided offline RL framework for wireless traffic scheduling, based on the fact that the network dynamics follows a well-defined physics model. In particular, NA-ORL initializes the scheduling policy through behavior cloning with a good model-based scheduling algorithm AMIX-ND, and trains a better scheduling policy by utilizing the actor-critic method with carefully crafted states and reward function. Building on NA-ORL, we further devise NA-MRL to deal with the non-stationary network dynamics, by learning an offline meta-policy to capture the network similarity among multiple offline RL tasks through offline meta-training. Extensive experiments are conducted to evaluate the performance of both NA-ORL and NA-MRL. The experimental results clearly demonstrate that the proposed NA-ORL and NA-MRL algorithms can achieve better performance over LDF and AMIX-ND, a very recent scheduling algorithm (regarded as the state-of-the-art), in various scenarios for the deadline-aware wireless scheduling.

For future work, we will investigate deadline-aware wireless scheduling problem in a more general network setting, where multiple links without interference could be activated simultaneously to transmit packets. It is worth noting that the objective formulation is the same as in the collocated network setting, but the action space is more complex. It is expected that offline RL trained scheduling algorithms have potential to significantly improve the performance.

References

[1] I.-H. Hou, V. Borkar, and P. Kumar, "A theory of qos for wireless," in *IEEE INFOCOM 2009*, 2009.

[2] I.-H. Hou and P. Kumar, "Admission control and scheduling for qos guarantees for variable-bit-rate applications on wireless channels," in *Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*, 2009, pp. 175–184.

[3] I.-H. Hou, "Scheduling heterogeneous real-time traffic over fading wireless channels," *IEEE/ACM Transactions on Networking*, vol. 22, no. 5, pp. 1631–1644, 2013.

[4] I.-H. Hou and P. Kumar, "Utility-optimal scheduling in time-varying wireless networks with delay constraints," in *Proceedings of the eleventh ACM international symposium on Mobile ad hoc networking and computing*, 2010, pp. 31–40.

[5] J. J. Jaramillo and R. Srikant, "Optimal scheduling for fair resource allocation in ad hoc networks with elastic and inelastic traffic," in *2010 Proceedings IEEE INFOCOM*. IEEE, 2010, pp. 1–9.

[6] J. J. Jaramillo, R. Srikant, and L. Ying, "Scheduling for optimal rate allocation in ad hoc networks with heterogeneous delay constraints," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 5, pp. 979–987, 2011.

[7] X. Kang, W. Wang, J. J. Jaramillo, and L. Ying, "On the performance of largest-deficit-first for scheduling real-time traffic in wireless networks," *IEEE/ACM Transactions on Networking*, vol. 24, no. 1, pp. 72–84, 2014.

[8] X. Kang, I.-H. Hou, and L. Ying, "On the capacity requirement of largest-deficit-first for scheduling real-time traffic in wireless networks," in *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2015, pp. 217–226.

[9] C. Tsanikidis and J. Ghaderi, "On the power of randomization for scheduling real-time traffic in wireless networks," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 59–68.

[10] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[11] X. Cao, C. Sun, and M. Yan, "Target search control of auv in underwater environment with deep reinforcement learning," *IEEE Access*, vol. 7, pp. 96 549–96 559, 2019.

[12] Y. Ansari, M. Manti, E. Falotico, M. Cianchetti, and C. Laschi, "Multi-objective optimization for stiffness and position control in a soft robot arm module," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 108–115, 2017.

[13] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[14] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[15] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[16] 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 *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[17] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, "Sample efficient actor-critic with experience replay," *arXiv preprint arXiv:1611.01224*, 2016.

[18] S. Bhanja and A. Das, "Impact of data normalization on deep neural network for time series forecasting," *arXiv preprint arXiv:1812.05519*, 2018.

[19] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[20] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," *arXiv preprint arXiv:1805.01954*, 2018.

[21] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 661–668.

[22] H. Hasselt, "Double q-learning," *Advances in neural information processing systems*, vol. 23, pp. 2613–2621, 2010.

[23] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[24] O. Anschel, N. Baram, and N. Shimkin, "Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2017, pp. 176–185.

[25] Q. Lan, Y. Pan, A. Fyshe, and M. White, "Maxmin q-learning: Controlling the estimation bias of q-learning," *arXiv preprint arXiv:2002.06487*, 2020.

[26] X. Chen, C. Wang, Z. Zhou, and K. Ross, "Randomized ensembled double q-learning: Learning fast without a model," *arXiv preprint arXiv:2101.05982*, 2021.

[27] S. Y. Lee, C. Sungik, and S.-Y. Chung, "Sample-efficient deep reinforcement learning via episodic backward update," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[28] H. Wang, S. Lin, and J. Zhang, "Adaptive ensemble q-learning: Minimizing estimation bias via error feedback," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[29] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1352–1361.

[30] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[31] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1126–1135.

[32] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "rl^2: Fast reinforcement learning via slow reinforcement learning," *arXiv preprint arXiv:1611.02779*, 2016.

[33] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick, "Learning to reinforcement learn," *arXiv preprint arXiv:1611.05763*, 2016.

[34] J. Humplik, A. Galashov, L. Hasenclever, P. A. Ortega, Y. W. Teh, and N. Heess, "Meta reinforcement learning as task inference," *arXiv preprint arXiv:1905.06424*, 2019.

[35] A. Gupta, R. Mendonca, Y. Liu, P. Abbeel, and S. Levine, "Meta-reinforcement learning of structured exploration strategies," *arXiv preprint arXiv:1802.07245*, 2018.

[36] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, "Learning to adapt in dynamic, real-world environments through meta-reinforcement learning," *arXiv preprint arXiv:1803.11347*, 2018.

[37] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen, "Efficient off-policy meta-reinforcement learning via probabilistic context variables," in *International conference on machine learning*. PMLR, 2019, pp. 5331–5340.

[38] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[39] Y. S. Nasir and D. Guo, "Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2239–2250, 2019.