Offline Reinforcement Learning for Wireless Network Optimization with Mixture Datasets

Kun Yang*, Chengshuai Shi*, Cong Shen*, Jing Yang†, Shu-ping Yeh‡, Jaroslaw J. Sydir‡
* Department of Electrical and Computer Engineering, University of Virginia, USA
† Department of Electrical Engineering, The Pennsylvania State University, USA
‡ Intel Corporation, USA

***Abstract*—The recent development of reinforcement learning (RL) has boosted the adoption of online RL for wireless radio resource management (RRM). However, online RL algorithms require direct interactions with the environment, which may be undesirable given the potential performance loss due to the unavoidable exploration in RL. In this work, we first explore the use of *offline* RL algorithms in solving the RRM problem. We evaluate several state-of-the-art offline RL algorithms for a practical RRM problem that aims at maximizing a linear combination of total rates and 5-percentile rates via user scheduling. Our findings indicate that the performance of offline RL for the RRM problem is heavily contingent upon the behavior policy deployed for data collection. We propose an innovative offline RL approach utilizing heterogeneous datasets from various behavior policies. This method demonstrates that a strategic mixture of datasets enables near-optimal RL policy generation, even with suboptimal behavior policies. Additionally, we introduce two enhancements: an ensemble-based policy to augment dataset mixture training efficiency, and a novel offline-to-online strategy for seamless adaptation to new environments. Our data mixture approach achieves over 95% efficiency of an online RL agent in the absence of expert data. The ensemble algorithm notably reduces training duration by half compared to the data mixture method. Furthermore, our model, when applied with offline-to-online fine-tuning, surpasses existing benchmarks by approximately 5% in our user scheduling problem.**

***Index Terms*—Radio Resource Management, Offline Reinforcement Learning, Deep Reinforcement Learning.**

I. Introduction

There is a growing interest in applying reinforcement learning (RL) to solving radio resource management (RRM) problems in wireless networks. Several unique properties in wireless RRM are the driving force behind this new trend. First, many of the RRM operations are sequential in nature, where a resource allocation decision is made, the network performance is observed, and then fed back to the decision maker to update the policy. Second, real-world wireless network optimization problems are often too complex to be modeled as simple optimization problems, which calls for *model-free* solutions that can be adaptive to the unknown deployment. Third, there are well-established control and feedback mechanisms in modern wireless networks, making it easy to observe system states and collect performance indicators.

These features have sparked significant efforts in developing RL solutions for wireless RRM. An overview of related works is given in Section II. The majority, if not all, of the existing works utilize *online RL*, where the RL policy gradually improves by interacting with the environment with no data prior to deployment. The exploration of the originally unknown environment, especially during the early stages where information about the environment is scarce and RL exploration is almost random, is an indispensable component for online RL but is also one of the major obstacles that prevent state-of-the-art RL algorithms from being deployed in real-world wireless networks. The lack of performance guarantee during RL exploration means that the network users may have to temporarily suffer from poor Quality of Service (QoS) so that the learning agent can gather information about the deployment for a potentially better RL policy. This tradeoff, however, is undesirable for the wireless network operator compared with model-based or rule-based solutions, which may not achieve as good a performance as online RL after it converges, but does not suffer from potentially significant initial performance degradation. This gap between the previous online RL solution and the real-world wireless device deployment motivates us to find better algorithms that can train RL policies without costly online interactions, which is the strength of offline RL.

In this paper, we advocate adopting *offline reinforcement learning* [2] for wireless network optimization. Offline RL aims at training RL agents using accessible datasets collected *a priori* and thus completely gets around online interactions. This paradigm is particularly suitable for wireless RRM, because in practice wireless operators already have deployed some policy that controls resource allocation, and there are mature mechanisms to collect the operational data. Our main contributions are summarized as follows.

- To the best of our knowledge, this work marks the first introduction of offline RL to the domain of wireless network optimization. This approach, while not requiring real-time interactions, is well suited for wireless systems and represents an important step towards practical RL implementations in this field.
- We have identified that combining sub-optimal datasets when solving offline RL in an RRM system can yield near-optimal performances. We further develop a data

A preliminary version of this work was presented at the 57th Asilomar Conference on Signals, Systems, and Computers [1]. The work of K. Yang, C. Shi, and C. Shen was partially supported by the U.S. National Science Foundation (NSF) under awards CNS-2002902, CNS-2003131, ECCS-2029978, ECCS-2030026, ECCS-2143559, and SII-2132700. The work of J. Yang was supported in part by the U.S. NSF under awards CNS-1956276, CNS-2003131 and CNS-2030026.

mixture strategy that removes the strict requirement of a high-quality offline dataset.
- We theoretically prove that as long as the behavior policy used for data collection satisfies certain coverage requirements of the system, data mixture is helpful. This theoretical finding is general and not limited to the RRM setting.
- We further enhance the data mixture strategy by incorporating ensemble methods and an offline-to-online fine-tuning process.

The rest of the paper is organized as follows. Related works are surveyed in Section II. The wireless network model and figure of merit are presented in Section III. The Markov Decision Process formulation of the user scheduling problem and the online RL solution are discussed in Section IV. Section V presents the basic framework of offline RL for wireless user scheduling, and reports the initial experiment results. The new solution of offline RL with mixture datasets is presented in Section VI, together with the experimental results. Theoretical analysis is given in Section VII. The two enhancement methods are presented in Section VIII and IX, respectively. Finally, Section XI concludes the paper.

II. Related Works

Online RL for RRM. To the best of our knowledge, prior literature on solving the wireless RRM problem via RL all rely on *online* RL. Examples include solving coverage and mobility problems using bandits [3]–[6], solving the power allocation problem using deep Q-networks in a centralized setting [7], [8], or solving a joint power and channel allocation problem using single agent deep RL [9]. Furthermore, the actor-critic structure is introduced for power allocation in wireless networks in [10]. A comparative study of several popular online RL algorithms for wireless network optimization is reported in [11]. In addition to a single resource control, [12] introduced a joint control of the power and spectrum resources using online RL. Beside centralized algorithms, multi-agent reinforcement learning (MARL) is another popular online RL framework that has been adopted in wireless RRM, such as power allocation [13], user scheduling [14] and general resource management and interference mitigation [15]. In a recent work [16], the authors discuss a novel scenario for packet routing, proposing solutions using MARL.

Offline RL. Unlike the online RL algorithms, offline RL focuses on learning RL policies exclusively from offline datasets, and has attracted significant interest in RL research [2]. Because offline RL cannot update policy by interacting with the environment, most methods choose to be conservative to mitigate potential distributional shift. Among the algorithms, batch-constrained Q-learning (BCQ) [17], conservative Q-learning (CQL) [18] and implicit Q-learning (IQL) [19] are the most state-of-the-art model-free deep offline RL algorithms. We will adopt these algorithms in our paper. Theoretical understanding towards optimal offline RL is also an active research direction, where data coverage [20], [21] and critical states [22] have been investigated.

Ensemble methods in offline RL. The concept of ensemble methods is a well-explored area within the reinforcement learning community, serving various purposes. In the context of online reinforcement learning, ensemble methods have been utilized to enhance exploration efficiency [23]–[25] and to mitigate modeling errors [26]–[28]. Furthermore, in the realm of offline reinforcement learning, such methods have been employed to manage out-of-distribution (OOD) estimation errors [29], [30]. However, our application in wireless RRM presents a unique scenario. Here, we have rule-based behavior policies that are optimized for specific regimes but perform suboptimally on a broader scale. We would like to harness the strengths of these baseline policies as we construct our offline reinforcement learning policies. Consequently, we propose an ensemble algorithm that builds upon these fundamental behavior policies.

Offline-to-online RL. To more efficiently utilize the offline dataset, offline-to-online RL (also known as hybrid RL) has emerged in the past few years. Among all the proposed methods, one of the directions is the research on how to effectively utilize offline datasets together with the online environment [31]–[34]. Another one is to directly fine-tune the offline policy in an online environment [35]–[37]. In this paper, we assume that the online environment is not always available; thus we focus on the fine-tuning methods. Existing fine-tuning methods often utilize regularization or punishment terms to force the policy to adapt pessimistically, in order to avoid severe performance collapse. Our method presented in Section IX improves the previous conservative method in [35].

III. System Model

In this section, we present the wireless environment and then discuss the figure of merit for the RRM problem.

A. Wireless Environment

We consider a wireless network consisting of N access points (APs) and M user equipments (UEs), as depicted in Fig. 1. This system is designed to evolve discretely where changes happen only at discrete time slots $t \in \{1, \cdots, T\}$, where T is the maximum length of the performance evaluation of RRM. We consider an episodic setup where each episode consists of the aforementioned T slots, and the system resets after an episode completes.

The APs are randomly placed in the intended coverage area, and their locations are fixed throughout the whole duration of RRM (i.e., they do not change across episodes). The rationale for randomly dropping APs is to ensure the collected datasets cover a diverse range of scenarios. By enforcing a minimum distance between different APs, we ensure that this process remains practical while effectively representing diverse situations. At the beginning of each episode, we randomly generate UEs and place them in the $l \times l$ square as shown in Fig. 1. There are (different) minimum distance requirements for both AP–UE (d_0) and AP–AP (d_1) distances, and the random placement is repeated until these requirements are satisfied.

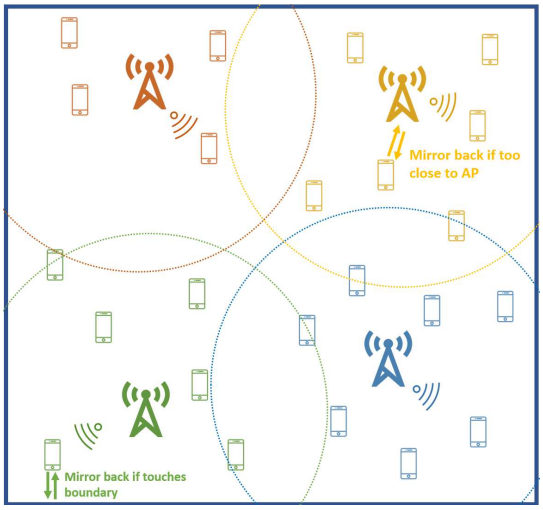


Fig. 1. Illustration of the wireless environment.

Our primary consideration is a wireless system in which pedestrian UEs can move around at a slow pace. More specifically, in each episode, each of the UEs can move independently randomly with a speed v_t that is sampled from $[0, 1]$ m/s, with their locations updated in each time slot. We adopt a mirror-back mechanism to handle the situations when the UEs move to the coverage boundary or violate the distance constraint as illustrated in Fig. 1.

UEs are associated with one of the APs at the beginning of each episode, and we ensure that every UE will be associated with one and only one AP. At each time slot t, the channel between AP i and UE j, denoted as $h_{i,j}(t)$, contains path-loss, shadowing, and short-term fading effects as detailed below.

1) **Path-loss.** We adopt the standard 3GPP indoor path-loss model [38]:

$$PL_{i,j} = 15.3 + 37.6 \log(d_{ij}) + L_{ow}, \quad d_{ij} > d_0, \tag{1}$$

where d_{ij} represents the distance between the AP i and UE j, and $L_{ow} = 10$ dB is a constant path-loss. We require the users to maintain at least a minimum distance $d_0 = 1$ meter from the APs.
2) **Shadowing.** We adopt a log-normal shadowing effect on all the links with a standard deviation of 7 dB.
3) **Fading.** A standard frequency-flat Rayleigh fading is simulated to capture the short-term randomness of the channel.

With this definition of channel parameters, we now denote the received signal of UE_j (associated with AP_i) at time period t as

$$y_j(t) = h_{i,j}(t)x_i(t) + \sum_{k \neq i} h_{k,j}(t)x_k(t) + n_j(t), \tag{2}$$

where $n_j(t)$ is the additive white Gaussian noise (AWGN) following $n_j(t) \sim \mathcal{CN}\{0, \sigma^2\}$. The task of our RRM problem is *user scheduling*, i.e., to determine which BS to serve which UE (or to turn off without serving any UE) for each time slot t. More specifically, at each time slot t, AP_i needs to select one of its associated users for active data communication. In reality, user association happens at a much slower time scale than user scheduling. Thus, we first perform user association at the beginning of each episode, and keep this association unchanged throughout the current episode. User scheduling then happens on a per-time-slot basis.

Under this setting, we model the instantaneous data rate for user j using Shannon capacity:

$$C_j(t) = \log_2(1 + \mathsf{SINR}_j(t)), \tag{3}$$

where $\mathsf{SINR}_j(t)$ denotes the signal to interference plus noise ratio (SINR) of user j at time slot t. We further define the *average user throughput* for user j as:

$$\bar{C}_j = \frac{1}{T} \sum_{t=1}^{T} C_j(t). \tag{4}$$

- **User association rule.** At the beginning of each episode, a user pool $\mathbb{P}_i$ is created for each AP i based on the maximum reference signal received power (RSRP) of each user [5]. More specifically, user j will be added to the user pool of AP i if $i = \arg\max_n \mathsf{RSRP}_{n,j}, \forall n \in \{1, \cdots, N\}$. An AP is allowed to only observe and measure users in its own user pool, and scheduling decisions are limited to these users.

B. Figure of Merit

If the figure of merit for wireless RRM is to maximize the system-level averaged data rate (across all users), then the solution boils down to always selecting the "best" UEs (in terms of the SINR) at each time step. This can be formulated as an optimization problem for each time slot, and there are extensive works studying different variants of this problem. However, almost all practical wireless networks must consider *fairness* across all UEs when solving the RRM problem. From the data rate perspective, the overall system figure of merit must consider both the *sum* and *tail* behaviors. In practice, this is often captured by the sum rate and 5-percentile rate as described below:

1) **Sum rate.**

$$C_{\text{sum}} = \sum_{j=1}^{M} \bar{C}_j. \tag{5}$$

2) **5-percentile rate.** We first give its definition as follows.

Definition 1 (5-percentile rate)**.** Suppose a system comprises M UEs, with each UE maintaining an average data rate $\bar{C}_j$. The 5-percentile rate, denoted as $C_{5\%}$, is defined as the highest data rate that at least 95% of UEs exceed. Formally, $C_{5\%}$ is the solution to the following optimization problem:

$$\begin{aligned} &\text{maximize} \quad C \\ &\text{subject to} \quad \mathbb{P}(\bar{C}_j > C) \geq 0.95, \quad \forall j \in \{1, \ldots, M\}. \end{aligned}$$

The main figure of merit of this work is a linear combination of the sum rate and the 5-percentile rate at time step t, parameterized by (μ, η):

$$R_{\text{score}}(t) = \mu C_{\text{sum}}(t) + \eta C_{5\%}(t). \tag{6}$$

We formally present the optimization problem as follows:

$$\begin{aligned} &\text{maximize } \sum_{t=1}^{T} R_{\text{score}}(A_t), \\ &\text{subject to } A(t) = (a_1(t), \ldots, a_N(t)), \\ &\qquad a_i(t) \in 0 \cup \mathbb{P}_i, \forall i \in \{1, \ldots, N\}. \end{aligned} \tag{7}$$

Here, $A(t)$ represents the actions taken by the APs at time t, where $a_i(t)$ is the action for the i-th AP. The decision variable $a_i(t)$ can take a value from the set $0 \cup \mathbb{P}_i$, where 0 represents the option of not serving any user at that time slot, and $\mathbb{P}_i$ is the pool of users associated with the i-th AP. This formulation encapsulates our need to control which user (if any) is served by each AP at every time slot.

We note that this weighted sum allows us to adjust the balance between sum and tail rates, by varying the parameters μ and η. However, directly maximizing Eqn. (6) is a non-trivial task with several challenges. Both the sum and 5-percentile rates are *long-term* performance measures that depend on the history of actions in an episode. The time-dependency of actions implies that we cannot take the optimization-per-slot approach to find a (near-)optimal solution. Additionally, the 5-percentile rate itself is a complicated measure that does not have a closed-form expression, and the dynamic nature of the system (channel randomness, user movement, etc.) further adds to the difficulty of optimizing Eqn. (6).

In our original definition of reward R_{score}, the second term, namely the 5-percentile rate, is hard to compute and difficult to optimize. We thus introduce a related metric *proportional fairness (PF)* ratio.

Definition 2 (Proportional Fairness (PF) ratio)**.** The PF ratio for a user j at time t, denoted as $\mathsf{PF}_j(t)$, is defined by the product of a weighting factor $w_j(t)$ and the user's data rate $C_j(t)$:

$$w_j(t) = 1/\tilde{C}_j(t), \quad \tilde{C}_j(t) = \alpha C_j(t) + (1-\alpha)\tilde{C}_j(t-1), \tag{8}$$

where

$$\tilde{C}_j(0) = C_j(0). \tag{9}$$

Although the PF ratio does not have a direct correlation with tail rates, it is inversely related to the user's long-term average data rate, denoted as $\tilde{C}_j(t)$. This implies that if a user experiences a persistently low data rate, indicating a prolonged period without adequate service, her PF ratio will subsequently be higher.

IV. RL FORMULATION

In this section, we show how to solve Problem (7) using RL. This is accomplished by first formulating the original system as a Markov Decision Process (MDP), and then presenting how to train a centralized *online* RL to control all the APs in the environment.

A. MDP Formulation

An episodic MDP is described by a tuple $M = (S, A, r, \gamma, P, T)$, where S and A stand for the state and action spaces respectively, r is the reward function mapping state-action pairs to a reward signal that reflects our design objective, $\gamma \in (0, 1)$ is the discount factor that is widely used to bound the cumulative reward in the MDP. P is the transition kernel advancing the current state-action pair to the next state in a random fashion, and T is the maximum time interval (length of the episode).

We define the key components of the episodic MDP for the wireless scheduling problem as follows.

1) **Observation.** For each AP i, we apply a top-k selection of the UEs in its user pool to collect observations. The criterion of selecting top-k UEs is by sorting all UEs in the pool based on the PF-ratio $w_{i,j}(t)$ defined in Eqn. (8) and only keeping the largest k UEs. We note that this is a common technique in the existing literature to deal with large amount of UEs [13], [15]. Then, with the top-k UEs, the AP measures the current SINR for each UE, and the local observation at AP i is defined as $o_i(t) = (\mathsf{SINR}_{i,1}(t), w_{i,1}(t), \cdots, \mathsf{SINR}_{i,k}(t), w_{i,k}(t))$. Finally, with all local observations, the learning agent creates the global observation by stacking the local ones as $O(t) = (o_1(t), \cdots, o_N(t))$.
2) **Action.** For each AP, the possible actions are to either select one from its top-k users to serve, or to turn itself off and serve no UE. This decision-making process is rooted in the definition of PF, where serving a user with a high PF ratio (indicating a higher urgency for service) results in a decrease in their PF ratio due to improved service. Conversely, users who are not served will experience an increase in their PF ratio, altering their service priority over time. The action space for each AP is thus $k + 1$, and the global action space is of size $(k + 1)^N$.
3) **Reward.** The objective (6) represents the final performance and cannot be directly decomposed into reward signals for each step. We thus adopt an existing design from [15] that has been shown to achieve a balanced tradeoff between sum and tail rates:

$$r(t) = \sum_{j=1}^{M} (w_j(t))^{\lambda} C_j(t). \tag{10}$$

By tuning the parameter λ, we can achieve the desired tradeoff between sum rate and 5% rate as detailed in [15].

We note that although the reward is defined in Eqn. (10), we still evaluate all designs using the original objective defined in (6), with $\mu = 1/M$ and $\eta = 3$, in the experiment.

B. Online RL

As a baseline, we explore the power of *online* RL for solving the aforementioned RRM problem. We choose two widely used actor-critic-type algorithms, called **Soft Actor-Critic (SAC)** [39] and **Proximal Policy Optimization (PPO)** [40], as our deep RL solutions and use them to train an online RL agent. Unlike most other RL problems which often face the issue of *training instability*, i.e., the learning agent may stuck at a saddle point of the loss function, SAC first introduces policy optimization by minimizing the KL-divergence (similar to [40]) to control the policy update, while encouraging exploration at the same time by maximizing the

entropy of each state. This is achieved by deriving the soft policy iteration steps for the actor-critic structure. SAC is off-policy and has faster convergence rate than other algorithms, and is adopted in our baseline experiment. PPO is chosen as an on-policy baseline for its ability to stabilize exploration by constraining the divergence between new and old policies. This approach ensures more consistent learning, with the potential cost that it might slightly slow the convergence.

We implement a system-level simulator that follows the wireless environment in Section III-A. In the online RL training phase, we create a pool of 20 distinct training environments, each with a unique AP and UE topology generated using different random seeds. The model then goes through 350 epochs, with each epoch comprising 5000 episodes of 200 steps. At the beginning of each episode, we randomly select an environment from this pool. For evaluation, we generate a separate set of 10 unique environments, distinct from the training set, using different random seeds. The model's performance is assessed across these 10 environments, with each conducting one episode. All results reported are from 10 independent runs, each with different random seeds. This approach ensures consistent initial states for the training and evaluation environments, while introducing variation in UE movements and shadowing effects in each sampled environment during the runs.

Other than the online RL policy trained via SAC and PPO, we also evaluate several rule-based baseline methods as follows. We note that these baseline methods will be used as behavior policies in the subsequent offline RL study.

- **Random.** At each time step, each AP randomly chooses one of its top-k users in the user pool to serve.
- **Greedy.** AP always chooses one of its top-k users with the largest SINR to serve.
- **Time division multiplexing (TDM).** All top-k UEs are served in a round-robin fashion. In each time slot, only the scheduled UE and its serving AP are active. All other APs are turned off.
- **ITLinQ.** This is a state-of-the-art, generalized independent set-based scheduling algorithm where we select UEs based on the tolerance of interference levels. The method is proved to be nearly optimal, especially under a dense network setting like our experiments. Details can be found in [41]. With a small tweak to the original algorithm where each AP can have multiple active links, we prioritize UEs associated with each AP based on their PF level. UEs are then actively served based on an interference tolerance criterion in the order of priority: $\max \mathsf{INR}_{i,j} \leq M\mathsf{SNR}_{i,j}^{\eta}$. After selecting an active UE, no further checks are performed. If no UEs are activated, the AP is turned off. In our study, we set $M = 4$ and $\eta = 0.5$.

Important simulation parameters are summarized in Table I, and the results are presented in Fig. 2. We see that with sufficient training (large training epochs), online RL has much better performance than all rule-based baselines. On the other hand, when the training is insufficient (e.g., fewer than 100 epochs), the performance of online RL agents may be worse, sometimes significantly, than the rule-based baselines. This demonstrates the price that one has to pay for *online* learning, which is due to the inevitable exploration of RL.

TABLE I
EXPERIMENT PARAMETERS

Parameter	Value
Number of APs	4
Number of UEs	10 - 24
Area	100×100 m^2
Min AP-AP distance	10 m
Min AP-UE distance	1 m
Max UE speed	1 m/s
Bandwidth	10 MHz
Transmit power	10 dBm
Episode length (T)	200
Number of UEs in the pool (k)	3
Running average parameter (α)	0.01
Reward discount factor (γ)	0.95
Weight exponent (λ)	0.8

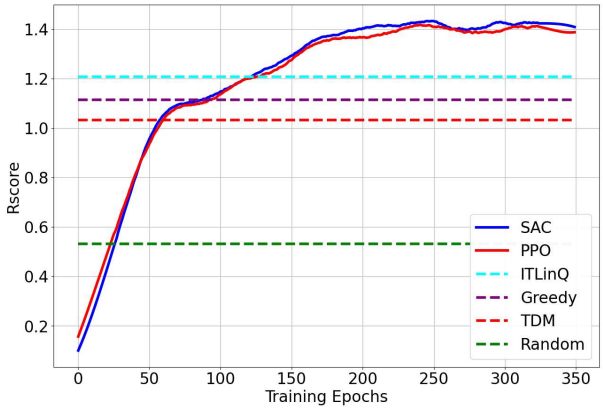


Fig. 2. Training performance of online RL algorithms SAC and PPO. Results are averaged over 10 independent runs as described in Section IV-B.

V. OFFLINE RL

To address the limitations of online RL, we resort to *offline* RL for RRM. We first describe how to collect offline dataset, and then evaluate the state-of-the-art offline RL algorithms for the wireless user scheduling problem.

A. Dataset

Offline RL allows the system to enjoy the advantages of RL without direct interaction with the environment. This is made possible by using offline datasets. The most common approach to have such datasets is through collecting operational data associated with the existing policies. For example, for the RRM problem, wireless operators often have existing solutions that have been deployed in the target environment. We can rely on the data collected by these existing solutions, which are called the *behavior policies (BPs)*, to train an offline RL policy.

In the user scheduling problem, we have implemented four rule-based policies described in Section IV-B as BPs. In addition, we include two other BPs that are based on online RL. These two policies differ in how well they are trained – one is early stopped at epoch 125 while the other is stopped

This article has been accepted for publication in IEEE Transactions on Wireless Communications. 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/TWC.2024.3395624

at epoch 350. Their performances can thus be identified from the blue curve in Fig. 2.

With datasets collected from these BPs, we summarize the offline RL experiment procedure as follows.

1) Choose a BP π_β from all available BPs.
2) Run π_β on the environment to collect a dataset D_{π_β}.
3) Train policy π_θ using an offline RL algorithm (see next subsection) on the dataset D_{π_β}.

We remark that the dataset collected in Step 2) may have poor quality because the corresponding BP may not achieve good performance. For example, as we see from Fig. 2, the four rule-based policies all have significant performance gaps compared with the well-trained online RL. We are interested in evaluating whether a "good" RL policy can be trained from datasets that may come from "bad" BPs.

In the online RL training phase, we utilize a shared pool of 20 distinct training environments, each characterized by unique AP and UE topologies, generated via different random seeds. This pool is also employed during the offline data collection phase. Here, we execute 5000 episodes for each BP, with each episode comprising 200 steps, cumulatively yielding 1 million trajectories per policy. Consistently, at the onset of each episode, we randomly select an environment from this pool, as detailed in Section IV-B, ensuring uniformity in our environment sampling strategy across both online training and offline data collection phases.

B. Offline RL for User Scheduling

Four state-of-the-art model-free offline RL algorithms are considered for solving the user scheduling RL problem.

- **Behavior constrained Q-learning (BCQ) [17].** The key idea of BCQ is to limit the actions for the policy to those already in the dataset or in the neighborhood of the observed actions. More specifically, BCQ enforces the following restrictions to optimize the reward: (1) distance between the selected actions and those in the dataset should be small, and (2) the new action should visit the existing states in the dataset. We also follow the same approach of training a *variational auto-encoder (VAE)* to avoid the complicated estimation of action distribution.
- **Conservative Q-learning (CQL) [18].** Unlike BCQ which explicitly disallows the actions to be too far away from those in the dataset, CQL introduces a regularization term to the reward such that unseen actions incur a larger loss. In [18], the authors adopt the KL-divergence as a measure for the regularization to punish actions that are too far away from those in the dataset. Compared with BCQ, CQL incorporates conservative exploration in the loss function and is easier to implement and update.
- **Implicit Q-learning (IQL) [19].** The previous two methods either explicitly constrain the actions to be in-distribution or regularize the loss values. IQL is a different approach that treats the state value function as a random variable and then utilizes expectile regression [42]. Practically, IQL is also easy to implement by changing the objective to a modified SARSA-type one.
- **Re-visiting behavior regularized Actor-Critic (ReBRAC) [43].** While maintaining a regularization term in the loss of actor and critic networks in offline RL is commonly used. ReBRAC introduces an active weight to these regularization terms, which significantly improves the performance of offline RL on some datasets. The network structure of ReBRAC largely follows the previous offline RL works.

In order to address the issue of distributional shift, all four algorithms share the core principle of *conservative exploration* [44]. These algorithms differ in how they adjust the standard RL training procedure to fit the offline dataset. Hence they are not sensitive to the specific (deep) RL structure as long as it is off-policy. In our experiments, we use the same Actor-Critic structure as used in the online RL experiment.

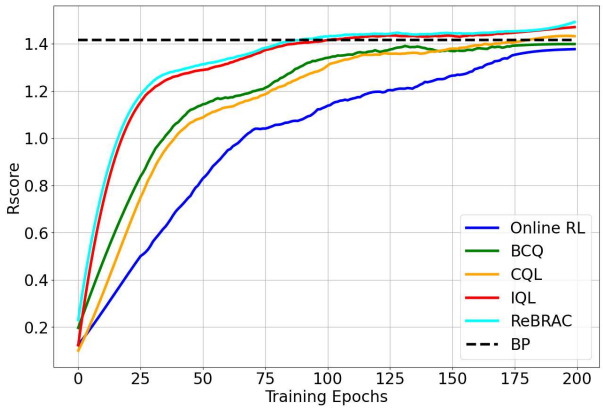


Fig. 3. Experiment result with different SOTA offline RL algorithms on an expert dataset, where the expert dataset is collected using the best online RL policy obtained from Sec. IV-B. All experiment results are conducted with 10 independent runs and evaluated on the same set of 10 validation environments as the online RL training.

Evaluation of behavior policies. To compare the training performance of offline RL methods with various behavior policies, we deploy the behavior policies in the same evaluation pool in Section IV-B. We then test each of them for 10 episodes across all evaluation environments and compute the average reward. This average value is represented as a straight line in the plots of this section. Notice that we define one *'epoch'* in our offline RL experiment as a training of 5000 batch updates, with the batch size of 200. This definition will align the data samples encountered by both online RL and offline RL in one 'epoch' to be the same. This definition of epoch stays consistent throughout the offline RL experiments.

High-quality dataset. In this test, we deploy the best online RL policy (see Sec. IV-B) as the behavior policy and collect one million trajectories as the offline dataset. In the offline RL experiments, a single epoch stands for $5,000$ mini-batch updates, and each batch contains 200 data trajectories. The results are presented in Fig. 3, where we also include the original online RL performance (blue curve) from Fig. 2 for comparison. We can see that all three offline RL algorithms are data-efficient – they have faster convergence than online RL. This is intuitively reasonable as online RL needs extra exploration while offline RL only needs to sample within the dataset. However, we also notice that all the algorithms con-

verge to approximately the same final performance, indicating that offline RL does not improve (much) over a high-quality behavior policy.

Low-quality dataset. It is natural to ask whether offline RL can produce a good policy when the dataset is of lower quality. To answer this question, we experiment using rule-based and the early-stopped online RL as behavior policies. While we also consider ReBRAC, a more advanced algorithm for offline RL, we observe only marginal improvements over IQL in our task. Moreover, ReBRAC introduces an additional hyper-parameter that requires tuning, adding complexity to the process. Therefore, for simplicity, we restrict to IQL as the offline RL algorithm for the rest of the paper since it has the second-best performance among all algorithms and also avoids the need for extensive hyper-parameter selection. The results are given in Fig. 4. We see that IQL is able to improve the offline RL performance over the baseline of each behavior policy, but this improvement is still limited. It is clear that the performance of the offline RL policy is restricted by the original behavior policy, which has motivated us to propose a new idea of mixing several low-quality datasets to boost the offline RL performance in Section VI.

Impact of dataset size. Understanding the impact of dataset size on offline RL performance is crucial in practice. We test this by varying the dataset size from $100,000$ to 5 million trajectories using the expert BP to collect datasets. The comparison is shown in Fig. 5. We can see that insufficient data can significantly impact the performance of offline RL. This gap reduces as we gradually increase the sample size from $100,000$ to $500,000$, where all four methods share a compatible performance compared to the online RL baseline. Notably, when the dataset size reaches 1 million, offline RL performance closely mirrors that of online RL. Beyond this point, from 1 million to 5 million trajectories, the benefits of increasing dataset size diminish, although they do not vanish entirely. This trend is consistent across all four state-of-the-art (SOTA) offline RL algorithms tested in the experiments, leading to the conclusion that while larger datasets generally enhance offline RL training, there exists a threshold beyond which additional data yields diminishing returns.

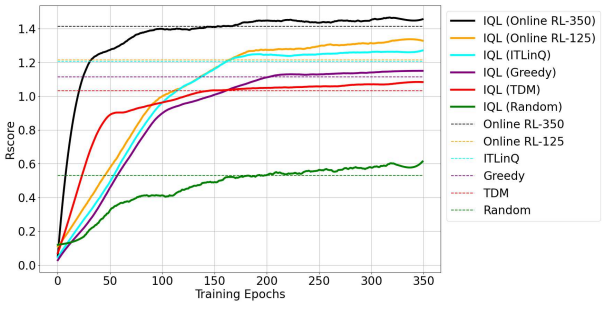


Fig. 4. Testing offline RL with datasets generated from different behavior policies. All the horizontal dash lines stand for the average performance of the behavior policy over a single episode's evaluation over all validation environments. The training results using datasets collected from different behavior policies are compared.

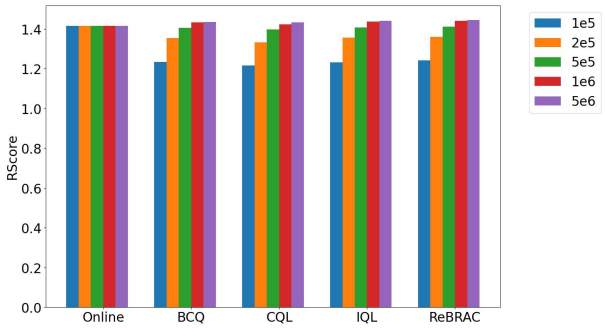


Fig. 5. Testing result with different amount of data in the dataset. Ranging from $100,000$ trajectories to $5,000,000$ trajectories. All other experimental results, unless explained separately, are tested with a dataset size of $1,000,000$ trajectories.

VI. Offline RL with Mixture Datasets

A. Mixture Datasets from Multiple BPs

The previous results demonstrate the potential of offline RL for the user scheduling problem in wireless RRM, but also suggest that the gain can be limited by the adopted behavior policy that is used to collect the dataset. The key open problem we would like to answer is the following.

Can we train a high-performance offline RL policy using datasets from low-performance behavior policies?

We answer this question positively by proposing a novel offline RL solution. Our key new idea is that although the dataset generated by a *single* low-performance BP may not contain enough information to learn a near-optimal RL policy, the cumulative dataset from *multiple* low-performance BPs may have sufficient diversity to cover the (near-)optimal state-action pairs, although each BP only covers a portion of them. Specifically, the mixture dataset is created with the following procedure.

1) For a given set of L BPs $\Pi = \{\pi_\beta^1, \pi_\beta^2, \cdots, \pi_\beta^L\}$, generate an offline dataset for each BP following the procedure in Section V-A. Evaluate $R_{\text{score},\pi_\beta^l}$ for each BP l according to Eqn. (6).
2) Select data samples from each BP's dataset uniformly at random to create the final dataset $\hat{D}$. The portion of data samples from BP l, denoted as $P_{\pi_\beta^l}$, should intuitively be proportional to the quality of this BP. One such allocation mechanism, which is adopted in our experiment, is

$$P_{\pi_\beta^l} = \frac{\exp\left(R_{\text{score},\pi_\beta^l}\right)}{\sum_{w=1}^{L} \exp\left(R_{\text{score},\pi_\beta^w}\right)}, \; l = 1, \cdots, L. \tag{11}$$

3) Use the final dataset $\hat{D}$ to train the offline RL policy.

We evaluate this solution using the same experimental setting as in Section V, and report the results in Fig. 6. We consider two different combinations of BPs, both with $L = 4$.

1) **Mixed-RL:** "bad" online RL (trained with 125 epochs), Greedy, TDM, and Random.
2) **Mixed-ITLinQ:** ITLinQ, Greedy, TDM, and Random.

To have a consistent comparison, we adopt the same data allocation[1] for both cases: online RL / ITLinQ (50%), Greedy (20%), TDM (20%), and Random (10%). The results are plotted in Fig. 6. We can see that although the trajectories come from low-performance BPs, using the mixture dataset leads to a significant reward improvement for offline RL. In fact, Mixed-RL can almost converge to the optimal online RL performance as shown in Fig. 6, and Mixed-ITLinQ is only slightly worse. Both outperform all their individual BPs by noticeable margins. These results demonstrate that even with low-performance BPs, we can still leverage the offline datasets to achieve near-optimal RL performance.

Why selecting ITLinQ and RL-125 as basic BP methods? We choose ITLinQ and RL-125 as the foundational BPs for our mixed dataset due to their distinct characteristics and performances. ITLinQ stands out as the most effective rule-based method, consistently outperforming other rule-based baselines. RL-125 is a representative early stopped online RL policy, which is commonly used in offline RL data collection for creating sub-optimal quality datasets [45]. Our action behavior analysis, illustrated in Fig. 7, shows that despite similar rewards, ITLinQ and RL-125 exhibit different behaviors. ITLinQ selects users effectively but lacks the strategy to power off when unhelpful. On the other hand, RL-125, with a more distributed action approach, learns when to disengage. The combination of these distinct strategies in the dataset mixture allows ITLinQ to benefit from TDM insights and RL-125 from Greedy strategies, suggesting a potential performance gain that will be further discussed in the subsequent section.

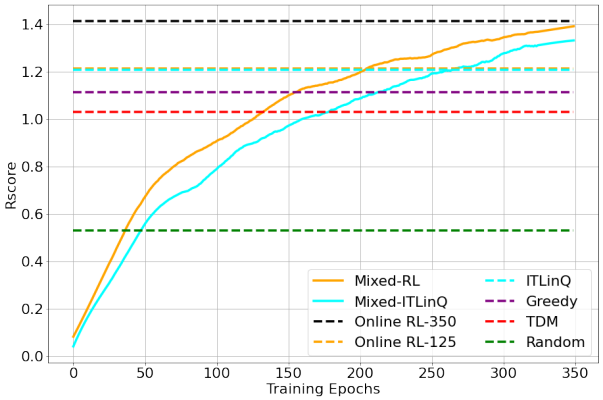
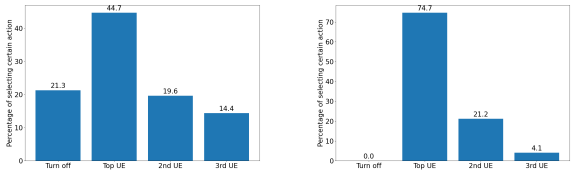


Fig. 6. Testing offline RL with a mixture dataset. where the mixed dataset consists of RL-125 / ITLinQ (50%), Greedy (∼20%), TDM (∼20%), Random (∼10%) of the data Results are validated across the same 10 validation environments as before.

B. Mixture Dataset with Varying Combinations

In our initial dataset mixture experiment, we assume stringent storage constraints, **restricting the mixed dataset to one million trajectories**. Recognizing that practical scenarios might not always prioritize storage limitations, we expand our experiment to encompass different mixtures of datasets to assess performance implications.

[1]Note that online RL-125 and ITLinQ have very similar performances; hence their allocations are the same.



(a) RL-125 (b) ITLinQ

Fig. 7. The frequency of selecting certain users or turning itself off for an AP, showing the different behavior of these two selected BPs.

TABLE II
TEST OF DIFFERENT BP DATASET MIXTURES

Mixture Type	Average reward
Original mixture (RL-125)	1.383 +- 0.035
Original mixture (ITLinQ)	1.332 +- 0.051
Full mixture	**1.435 +- 0.032**
Mixture w/o RL-125	1.386 +- 0.041
Mixture w/o ITLinQ	1.401 +- 0.025
Mixture w/o Greedy	1.412 +- 0.031
Mixture w/o TDM	1.405 +- 0.032
Mixture w/o Random	1.431 +- 0.028

As outlined in Table II, we explore various combinations of data mixtures. We designate our original mixture, constrained by a 1 million data samples limit, as the 'Original mixture'. This is compared against a comprehensive mixture comprising all suboptimal datasets (RL-125, ITLinQ, Greedy, TDM, and Random) amounting to a total of 5 million trajectories. To discern the individual contribution of each dataset, we sequentially remove one dataset at a time from the complete mixture. These variations are referred to as 'Mixture w/o [xxx]' in the table. This methodology allows us to not only confirm the advantages of integrating additional data but also to evaluate the relative importance of each behavior policy's dataset within the context of the fully mixed dataset.

The results from these varying mixtures are presented in Table II, which indicates that enhancing the dataset size can elevate the performance gained from the mixed suboptimal dataset to levels nearing those attained by the offline RL using an expert dataset. This reveals that the performance of the sampled dataset mixture can be amplified by increasing its size. Our theoretical analysis in Section VII delves into explaining this phenomenon. While expanding a single offline dataset (Fig. 5) might sometimes yield limited improvements, a combination of datasets offers distinct advantages. It is evident from our findings that among all the suboptimal behavior policies, RL-125 contributes most significantly to offline training performance, whereas the Random dataset contributes the least impact.

VII. THEORETICAL ANALYSIS

In this section, we provide some theoretical insights to explain the empirical observations in Sections V and VI, especially performing offline RL with one and mixture datasets. In particular, the concept of *data coverage* is leveraged as a key tool to obtain these theoretical insights, which is introduced in the recent offline RL theory studies [20], [21].

To gain some fundamental intuitions, we focus on the basic tabular MDP case, with $|S|, |A| < \infty$ and $\gamma = 1$. Also, to facilitate discussions, for the MDP M, we denote the deterministic optimal policy for one MDP as π^* and the value function of a policy π as $V(\pi) = \mathbb{E}[\sum_{t\in[T]} \gamma^t r_t(s_t, a_t)|M, \pi]$, where the expectation is over the randomness in the induced trajectory of π on M. Furthermore, a policy π is called ε-optimal if $V(\pi^*) - V(\pi) \leq \varepsilon$.

A. Offline RL with One Dataset

The key observation from experimental results in Section V is that when performing offline RL with one dataset collected from one behavior policy, the learned performance is often limited, i.e., it cannot approach the optimal performance. To formalize this, we denote the considered behavior policy as ρ and define its single-policy coverage coefficient as follows:

$$C^* := \max_{(s,a,t)\in S\times A\times[T]} \frac{d_t^*(s,a)}{d_t^\rho(s,a)}, \tag{12}$$

where $d_t^*(s,a)$ and $d_t^\rho(s,a)$ are probability of visiting the state-action pair (s,a) at step t induced via the optimal policy π^* and behavior policy ρ, respectively. It can be observed that a smaller C^* indicates that the behavior policy is more similar to the optimal policy. In the worst case, if the optimal policy would visit one state-action pair (s,a) that is not covered by the behavior policy, i.e., $d_t^*(s,a) > 0$ but $d_t^\rho(s,a) = 0$, the coverage coefficient would be infinity i.e., $C^* = \infty$.

Intuitively, when the behavior policy provides poor coverage of the optimal policy, i.e., a large C^*, the offline RL would be difficult since the behavior policy can only provide limited information about the optimal policy, which matches our observed experimental results. To capture this phenomenon, recent theoretical RL analyses [20], [21], [46] have established the corresponding hardness results: in the worst case, learning an ε-optimal policy $\hat{\pi}$ when the coverage coefficient is C^* would require the number of available trajectories contained in the dataset K to be

$$K \geq \Omega\left(\frac{C^* T^3 S}{\varepsilon^2}\right). \tag{13}$$

It can be observed that with a large C^*, i.e., poor coverage, a large offline dataset is required to obtain a good learned policy. Moreover, if $C^* = \infty$ as aforementioned, even an unlimited amount of available offline data is insufficient for offline RL algorithms to approach the optimal policy, which explains the observations in Section V.

B. Offline RL with Mixture Datasets

With the limitations of performing offline RL with one dataset collected via one behavior policy explained, we then move to discuss the theoretical insights of using a mixture of datasets. Especially, we consider that there are L available datasets $\{D_1, D_2, \cdots, D_L\}$, whose aggregation is denoted as D. These datasets are of size $\{K_1, K_2, \cdots, K_L\}$ and collected from behavior policies $\{\rho_1, \rho_2, \cdots, \rho_L\}$, respectively. Then, the following theorem indicates that the theoretical framework of pessimistic-value iteration (PEVI), recently proposed in [20], [21], [47], can effectively learn from the mixture of datasets. The details of PEVI are provided in Algorithm 1 for completeness.

Theorem 3. *With probability at least $1-\delta$, the output policy $\hat{\pi}$ of PEVI satisfies that*

$$V(\pi^*) - V(\hat{\pi}) \leq \tilde{O}\left(\sqrt{\tilde{C}^* T^4 S / \sum\nolimits_{l\in[L]} K_l}\right), \tag{14}$$

where

$$\tilde{C}^* := \max_{(s,a,t)\in S\times A\times[T]} \frac{\sum_{l\in[L]} K_l d_t^*(s,a)}{\sum_{l\in[L]} K_l d_t^{\rho_l}(s,a)}. \tag{15}$$

It can be observed that as long as the newly defined **collective coverage coefficient** is finite, i.e., $\tilde{C}^* < \infty$, with enough amount of data, the optimal performance can be approached. Furthermore, to have $\tilde{C}^* < \infty$, it is sufficient to guarantee that for each $(s,a,t) \in S\times A\times[T]$ that $d_t^*(s,a) > 0$, there exists a behavior policy ρ_l such that $d_t^{\rho_l}(s,a) > 0$. In other words, the behavior policies can collectively cover the optimal policy. We note that this is a much less stringent requirement than having one single policy covering the entire optimal policy itself as in Section VII-A. This result also explains our observations in Section VI that a good policy can be learned from multiple poor behavior policies since they may provide good collective coverage.

Proof of Theorem 3. From Lemma B.3 of [21], with probability at least $1-\delta$, it holds that

$$V(\pi^*) - V(\hat{\pi}) = O\left(\sum_{t\in[T]} \sum_{(s,a)\in S\times A} d_t^*(s,a)\Gamma_t(s,a)\right), \tag{16}$$

with $\Gamma_t(s,a)$ as a confidence interval and set to be $c\sqrt{T^2\log(SAT/\delta)/N_t(s,a)\vee 1}$ as in Algorithm 1, where c is a universal constant and $N_t(s,a)$ is the count of visitations on the state-action pair in the overall dataset D. Note that the effectiveness of this confidence bound can be proved following the standard Hoeffding inequality as in Lemma B.1 of [21].

Furthermore, denoting $\tilde{K} = \sum_{l\in[L]} K_l$ and $\tilde{d}_t(s,a) = \sum_{l\in[L]} K_l d_t^{\rho_l}(s,a)/\tilde{K}$ with the multiplicative Chernoff bound, it holds that

$$\mathbb{P}\left(N_t(s,a) \geq \frac{1}{2}\tilde{K}\tilde{d}_t(s,a)\right) \geq 1 - \exp\left(\frac{1}{8}\tilde{K}\tilde{d}_t(s,a)\right), \tag{17}$$

which means when $\tilde{d}_t(s,a) \geq 8\log(SAT/\delta)/\tilde{K}$, with probability at least $1 - \frac{\delta}{SAT}$, it holds that

$$N_t(s,a)\vee 1 \geq N_t(s,a) \geq \frac{1}{2}\tilde{K}\tilde{d}_t(s,a) \geq \frac{\tilde{K}\tilde{d}_t(s,a)}{8\log(SAT/\delta)}, \tag{18}$$

which is also naturally guaranteed when $\tilde{d}_t(s,a) < 8\log(SAT/\delta)/\tilde{K}$. Thus, using a union bound, with probability at least $1-\delta$, this inequality holds for all $(s,a,t) \in S\times A\times[T]$.

Finally, we can obtain that

$$V(\pi^*) - V(\hat{\pi}) \overset{(a)}{\leq} \tilde{O}\left(\sum_{t\in[T]}\sum_{(s,a)\in S\times A} d_t^*(s,a)\sqrt{\frac{T^2}{\tilde{K}\tilde{d}_t(s,a)}}\right) \overset{(b)}{\leq} \tilde{O}\left(\sqrt{\frac{\tilde{C}^* T^2}{\tilde{K}}}\sum_{t\in[T]}\sum_{s\in S}\sqrt{d_t^*(s,\pi^*(s))}\right) \overset{(c)}{\leq} \tilde{O}\left(\sqrt{\tilde{C}^* T^4 S/\tilde{K}}\right), \tag{19}$$

where inequality (a) is from the definition of $\Gamma_t(s,a)$, inequality (b) is from the definition of $\tilde{C}^*$, and inequality (c) is from the Cauchy-Schwarz inequality and the deterministic optimal policy. The theorem is then proved. □

We note that the sufficiency of collective coverage broadly applies to different offline RL settings and designs. For example, besides the model-based PEVI, the following result can be established for the model-free LCB-Q algorithm in [48] by adapting its Theorem 1.

Proposition 4. With probability at least $1-\delta$, the output policy $\hat{\pi}$ of LCB-Q (Algorithm 1 in [48]) satisfies that

$$V(\pi^*) - V(\hat{\pi}) \leq \tilde{O}\left(\sqrt{\tilde{C}^* T^5 S/\sum_{l\in[L]} K_l}\right), \tag{20}$$

where $\tilde{C}^*$ follows the definition in Eqn. (15)

Moreover, similar results hold beyond tabular MDPs. The following proposition, modified from Corollary 4.5 in [47], can be established for linear MDPs considering the features of each state-action pair (see Assumption A in [49] and Definition 4.3 in [47] for a complete definition). It can be observed that the parameter $\tilde{C}^*_{\text{lin}}$ serves as the same role as $\tilde{C}^*$ in measuring the collective coverage of behavior policies $\{\rho_1, \cdots, \rho_L\}$ for the optimal policy π^*.

Proposition 5. Considering a linear MDP with feature mapping $\phi(\cdot,\cdot): S\times A \to \mathbb{R}^d$, with probability at least $1-\delta$, the output policy $\hat{\pi}$ of the linear MDP version of PEVI (Algorithm 2 in [47]) satisfies that

$$V(\pi^*) - V(\hat{\pi}) \leq \tilde{O}\left(\sqrt{\tilde{C}^*_{\text{lin}} T^4 d^3/\sum_{l\in[L]} K_l}\right), \tag{21}$$

if there exists a constant $\tilde{C}^*_{\text{lin}} < \infty$ such that for all $t \in [T]$, $\tilde{C}^*_{\text{lin}} \cdot \sum_{l\in[L]} K_l \cdot \mathbb{E}_{\rho_l}[\phi(s_t,a_t)^\top \phi(s_t,a_t)] \succeq \sum_{l\in[L]} K_l \cdot \mathbb{E}_{\pi^*}[\phi(s_t,a_t)^\top \phi(s_t,a_t)]$.

VIII. Ensemble-based mixture algorithm

Our previous experiments, as depicted in Figs. 6 and 4, reveal that despite a limited final performance, offline RL applied to a single-source dataset (using one BP) tends to converge faster than when using mixed datasets (using multiple BPs). We hypothesize this outcome is at least partially attributed to the fact that the mixing dataset does not exploit any information related to the behavioral policies. This observation and assumption led us to develop an ensemble-based method to accelerate the convergence speed by leveraging the information embedded in different behavior policies. This new method is detailed in Algorithms 2 and 3.

Algorithm 1 PEVI [20], [21], [47]

Input: Dataset $D = \{D_l : l \in [L]\}$
$\forall (s,a,s',t) \in S\times A\times S\times[T]$, obtain $N_t(s,a,s')$, $N_t(s,a)$ as the number of visitation on (s,a,s',t) and (s,a,t) in D; then calculate $\hat{P}_t(s'|s,a) = N_t(s,a,s')/(N_t(s,a)\vee 1)$
Initialize $\hat{V}_{T+1}(s) \leftarrow 0, \forall s \in S$
for $t = T, T-1, \cdots, 1$ **do**
  for $(s,a) \in S \times A$ **do**
    Set $\Gamma_t(s,a) \leftarrow c\sqrt{\frac{T^2\log(SAT/\delta)}{N_t(s,a)\vee 1}}$
    Set
$$\hat{Q}_t(s,a) \leftarrow r_t(s,a) + \sum_{s'\in S}\hat{P}_t(s'|s,a)V_{t+1}(s') - \Gamma_t(s,a).$$
  end for
  for $s \in S$ **do**
    $\hat{\pi}_t(s) \leftarrow \arg\max_{a\in A}\hat{Q}_t(s,a)$
    $\hat{V}_t(s) \leftarrow \hat{Q}_t(s,\hat{\pi}_t(s))$
  end for
end for
Output: Policy $\hat{\pi}$

Algorithm 2 Training for Ensemble Policies

Require: N datasets collected by different behavior policies $\{D_1, D_2, ..., D_N\}$, Offline RL training algorithm $algo$, Batch size b
Initialize offline RL policies for each dataset $\{\pi_1, \pi_2, ..., \pi_N\}$
for $i = 1$ to N **do**
  while π_i is not converged **do**
    Sample a minibatch b from D_i
    Update π_i using $algo$
  end while
end for
return set of policies $\{\pi_1, \pi_2, ..., \pi_N\}$

In the training phase, our algorithm independently trains N policies, leveraging the advantage that convergence on individual suboptimal datasets is typically faster than on a mixed dataset These separately trained policies are subsequently utilized to construct an ensemble policy in the inference phase.

As delineated in Algorithm 3, in the inference phase, we iterate through the datasets to compute a similarity measure for each behavior policy. Each policy is then assigned a weight based on this similarity, which is defined in terms of the L2 norm. The ensemble process integrates these weighted policies to form a comprehensive policy. The action is subsequently determined by this ensembled policy.

To thoroughly evaluate the effectiveness of the ensemble method, we conduct a comparison between the proposed ensemble method and the data mixture method. Specifically, the ensemble method utilizes four datasets, namely RL-125, ITLinQ, Greedy, and TDM, as mentioned earlier. To ensure a fair comparison, we combine all the datasets during dataset mixture so that both methods have an equal size in total data

Algorithm 3 Ensembled Inference

Require: N datasets collected by different behavior policies $\{D_1, D_2, ..., D_N\}$, Policy set $\{\pi_1, \pi_2, ..., \pi_N\}$

Receive state for inference s_t
for $i = 1$ to N **do**
  Set minimum distance in dataset i as $d_i = \inf$
  for States s_n in D_i **do**
    Compute $d_i = \min(d_i, ||s_n - s_t||)$ ▷ *Find the most similar state in the dataset*
  end for
end for
Compute $w_i = e^{-d_i}$ ▷ *Larger distance → smaller weight*
Ensemble all policies $\pi_{ens} = \sum_i w_i * \pi_i(s_t)$
Choose action from this new policy $a_t = \arg\max \pi_{ens}$

samples. Furthermore, in each epoch, we have $5,000$ mini-batch updates in the offline RL training. For a fair comparison of the ensemble method, we equally distribute these $5,000$ mini-batches to 4 policies, meaning that each policy has $1,250$ mini-batch updates in one epoch in the ensemble training phase. The testing result, as depicted in Fig. 8, aligns with our expectations: the ensemble-based algorithm, supplemented by additional behavior policy information, converges approximately 30% faster than training on the mixed dataset, while maintaining comparable performance.

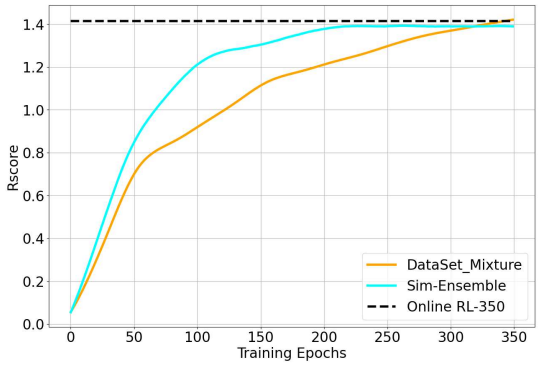


Fig. 8. The training efficiency of the ensembled policy.

While the ensemble method outlined above offers several benefits, it is important to also acknowledge areas for improvement. One aspect to consider is the complexity and efficiency. The necessity of iterating through datasets to determine similarity weights can lead to a slowdown during inference, as evidenced in Table III. However, it is worth noting that this increase in inference time is counterbalanced by the faster convergence during training on individual datasets compared to a mixed dataset. Additionally, our observations from Fig. 8 indicate that the final convergence of the ensembled policy is marginally less optimal compared to a dataset mixture policy. This could potentially be attributed to a reduced level of exploration in the absence of a mixed dataset.

To address these limitations, we outline a few promising research ideas in the following for possible future research. First, directly learning an ensembled policy, as opposed to learning individual policies and then ensembling them, may improve execution efficiency. Furthermore, the training efficiency can be enhanced by transitioning to a distributed training framework, where we have different BPs (or their datasets) at different clients.

TABLE III
COMPARISON OF INFERENCE TIMES FOR SINGLE AND ENSEMBLED POLICIES

Policy Type	Inference Time/sample (ms)
Single Policy	29
Ensembled Policy	1035

IX. OFFLINE-TO-ONLINE FINE-TUNING WITH ADAPTIVE REGULARIZATION

So far, we have focused exclusively on offline RL, where online interaction is strictly prohibited. In reality, however, once the offline policy is obtained, there can be an offline-to-online adaptation phase where the offline policy is further optimized to address the issue of potential policy collapse in an unfamiliar environment. Some SOTA offline RL works [19], [43] have discussed the effect of offline-to-online fine-tuning, and compared their post fine-tuning efficiency. However, no details of how the fine-tuning is conducted were reported. In other works that concentrate on the fine-tuning process, a significant challenge in fine-tuning methods is the trade-off between the possible performance degradation due to optimistic exploration [37] and the conservative behavior resulting from the conservative policy updates [35]. In the method proposed in [35], the authors control the stepsize of the policy optimization during fine-tuning with Langrangian-style optimization and this method uses Lagrangian dual variable γ to control the level of conservativeness, as shown in Eqn. (22), where θ and θ' stand for the parameters of the actor network and the target actor network in a Soft Actor-Critic (SAC) style algorithm.

$$\max_{\theta} \min_{\gamma \leq 0} \mathbf{E}_{s\ B}[Q_\phi(s,a) - \gamma[\epsilon - (a - \pi_{\theta'}(s))^2]], a = \pi_\theta(s). \tag{22}$$

While the approach in [35] partially addresses the conservative limitation, it still retains some level of pessimism throughout the fine-tuning process. Sustaining this pessimistic perspective may not be optimal, which motivates us to employ regularizers akin to Proximal Policy Optimization [40] with a progressively decreasing regularization weight during fine-tuning. This strategy ensures the initial conservatism to prevent performance degradation, while fostering exploration as the policy solidifies. The updated objective function encapsulating our methodology is delineated as:

$$\max_{\theta} \mathbf{E}_{s\ B}[Q_\phi(s,a) - \gamma[(a - \pi_{\theta'}(s))^2]], a = \pi_\theta(s). \tag{23}$$

In our method, γ decays by half every 25 epochs. This modification ensures early-phase conservatism to counteract instability and subsequently minimizes conservativeness to amplify the online performance.

The key difference between our algorithm and prior baselines is the dynamic regularization weight. Initially, we limit exploration to ensure the online policy mirrors the offline one.

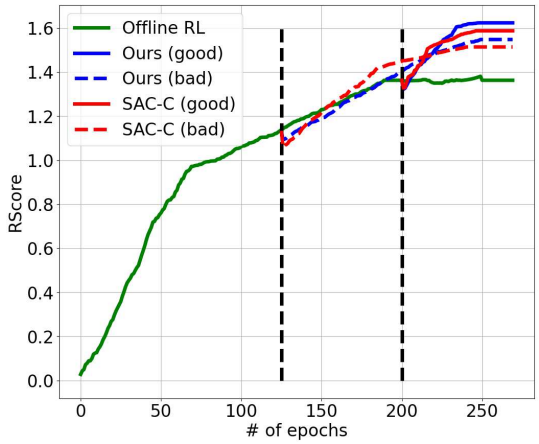


Fig. 9. The comparison of fine-tuning algorithms where we have two offline RL agents in the experiment: one at 125 epoch while the other from 200 epoch

As fine-tuning advances, these limits are gradually eased. We hypothesize that our method offers better safety levels comparable with static regularization but with enhanced performance after fine-tuning.

This hypothesis is supported by the results in Fig. 9. We evaluate the performance of our fine-tuning techniques on two offline agents, from the 125th and 200th training epochs respectively. Our observations indicate that:

1) Fine-tuning of agents with better initial performances is easier than those with poor early performances.
2) With a larger regularization term γ at the beginning phase, our fine-tuning method had a slower convergence speed compared to SAC-C. This slow-down behavior tends to be more significant with a worse starting point – where the system is regularized to a worse starting point.
3) Both strategies experience an initial performance drop, which could be mitigated with stronger safety constraints, at the cost of compromised fine-tuning results.

We also noted that refining a high-performing agent is more straightforward than a suboptimal one, with the latter demanding more online training for satisfactory results.

X. Discussions

Difference between simulator and real-world system. While our experiments show promising results, they are conducted in a simplified simulator with certain limitations. To name a few:

- We use a single pattern pathloss module, whereas real-world pathloss can be more complex.
- Our indoor environment simulation does not include obstacles, which are common in real-world scenarios.
- The system focuses solely on user scheduling, omitting other real-world resource management aspects like power and bandwidth allocation.

Generalization ability of dataset mixture. While this work focuses on the dataset mixture of user scheduling problems, the recent work [50] has deployed a similar method under the network slicing use case, showing a similar near-optimal policy performance using only sub-optimal datasets. Furthermore, the offline RL work in robotics [51] uses data augmentation methods to increase the data coverage, which shares the same philosophy with our dataset mixture approach.

Computation efficiency. While our proposed RL methods demonstrate superior performance over rule-based approaches, it is important to note that they require additional training. The training for the neural network, whose structure is detailed in Appendix A, takes around 40 hours on a single Nvidia RTX 3090. However, a two-layer neural network is expected to exhibit comparable inference times to rule-based methods when deployed on modern devices after training. Besides the training cost, the ensemble model, though faster in training, introduces greater computational complexity. This is primarily due to its search mechanism; see Section VIII and Table III.

Performance summary. Summarizing our method's performance, we find that the direct application of offline RL with an expert dataset is effective for the considered RRM problem and is a more stable solution. However, when equipped with only sub-optimal datasets, the dataset mixture strategy nearly matches the performance of online RL but requires 40% more training time. The enhanced ensemble method, while improving training efficiency, sacrifices 5% performance and incurs about 35 times slower inference speed.

XI. Conclusion

In our work, we have presented offline RL as a potential solution to the wireless RRM problem, using a specific wireless user scheduling problem as a case study. We assessed various representative offline RL algorithms for their long-term performance and convergence rate in the wireless problem. Notably, offline RL performance is significantly influenced by the quality of the behavior policy used for data collection. Motivated by this key observation, we introduced a unique offline RL strategy that mixes datasets from diverse behavior policies. Our findings indicated that this method can produce a near-optimal RL policy even when all contributing behavior policies are suboptimal. Additionally, we enhanced our data mixture technique into an ensemble-based method, utilizing state similarity for action determination. This ensemble method converges faster than the original dataset mixture, with a marginal performance decline as a trade-off. We also experimented with an adaptive fine-tuning approach, enabling offline RL agents to transition to superior online policies using minimal online steps. For possible future research directions, we note that our method relies on a centralized offline RL framework for RRM policy generation, which may hinder scalability. Future research could explore multi-agent offline RL solutions and potentially federated learning for training across diverse data sources.

Appendix

In the appendix, we discuss how we set up the wireless environment in the simulation and perform RL training in detail.

A. Wireless Environment

We implement a Python-based wireless network simulator for the experiments, which can be entirely recreated using the numpy Python package. The setup involves the following components:

TABLE IV
THE NETWORK STRUCTURE FOR ALL OFFLINE RL ALGORITHMS

Actor	Input dim: 24 Hidden dim $\times 2$: 256 Output dim: 256
Critic	Input dim: 280 Hidden dim $\times 2$: 256 Output dim: 1
BCQ (Q-learning style)	Input dim: 24 Hidden dim $\times 2$: 256 Output dim: 256

- **AP and UE generation:** APs are randomly generated with a minimum distance of 10 meters between them. If two APs are too close, one is randomly picked and regenerated. UEs are then generated ensuring that the minimum initial distance from any AP is greater than 1 meter. These positions are controlled by random seeds, allowing the reproducibility of the AP-UE topology.
- **UE association:** With the AP-UE distribution set, we assume all APs are at full power and compute the RSRP to associate UEs with APs.
- **Pathloss and shadowing:** For detailed large-scale fading modeling, please refer to Section III-A.
- **Small-scale fading:** Rayleigh fading is simulated using the 'Sum of Sinusoids' method with a total of 100 sinusoids.
- **Random walk:** A UE would randomly select a walking direction and a step length between 0 and 1. Steps are projected onto the x and y axes, with UEs bouncing back upon encountering APs or boundaries.
- **Reset:** During resets, all APs and UEs return to their initial positions, and cumulative data rates are reset to zero.

B. Deep Neural Network Training

Since all the SOTA offline RL algorithms, except for BCQ, share a similar Actor-Critic structure, all the hidden layer settings for these algorithms stay the same in the experiment, as shown in Table IV.

The batch definition is given in Section V. All the other hyper-parameters used for training are listed below. Notice that CQL, IQL, and ReBRAC have 2 critic networks that function as an 'ensemble' mechanism to improve the performance[2].

TABLE V
HYPER-PARAMETERS FOR DIFFERENT OFFLINE RL ALGORITHMS

General	Discount factor γ: 0.99 Optimizer: Adam
BCQ	τ : 0.005 lr : $3e-4$
CQL	Actor lr: $3e-5$ Critic lr: $3e-4$ Behavior cloning steps: 0 α: 3
IQL	Actor lr: $1e-4$ Critic lr: $3e-4$ β : 3 τ_{iql}: 0.7
ReBRAC	Actor lr: $1e-4$ Critic lr: $1e-3$ β_1 : 0.01 β_2: 0.1 τ: 0.005

[2]For the reference of an offline RL algorithm implementation, please refer to https://github.com/tinkoff-ai/CORL/tree/main.

REFERENCES

[1] K. Yang, C. Shen, J. Yang, S.-p. Yeh, and J. Sydir, "Offline reinforcement learning for wireless network optimization with mixture datasets," in *2023 57th Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2023.

[2] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.

[3] S. Nagaraja, F. Meshkati, M. Yavuz, S. Mitra, V. Khaitan, V. P. S. Makh, C. S. Patel, Y. Tokgoz, and C. Shen, "Power control for a network of access points," Nov. 2016, US Patent 9,497,714.

[4] N. Valliappan, C. Chevallier, A. D. Radulescu, and C. Shen, "Base station employing shared resources among antenna units," Sept. 2016, US Patent 9,451,466.

[5] C. Shen, R. Zhou, C. Tekin, and M. van der Schaar, "Generalized global bandit and its application in cellular coverage optimization," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 218–232, Feb. 2018.

[6] Y. Zhou, C. Shen, and M. van der Schaar, "A non-stationary online learning approach to mobility management," *IEEE Trans. Wireless Commun.*, vol. 18, no. 2, pp. 1434–1446, Feb. 2019.

[7] K. I. Ahmed and E. Hossain, "A deep Q-learning method for downlink power allocation in multi-cell networks," *arXiv preprint arXiv:1904.13032*, 2019.

[8] F. Meng, P. Chen, and L. Wu, "Power allocation in multi-user cellular networks with deep Q learning approach," in *IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–6.

[9] G. Zhao, Y. Li, C. Xu, Z. Han, Y. Xing, and S. Yu, "Joint power control and channel allocation for interference mitigation based on reinforcement learning," *IEEE Access*, vol. 7, pp. 177 254–177 265, 2019.

[10] Y. S. Nasir and D. Guo, "Deep actor-critic learning for distributed power control in wireless mobile networks," in *2020 54th Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2020, pp. 398–402.

[11] K. Yang, C. Shen, and T. Liu, "Deep reinforcement learning based wireless network optimization: A comparative study," in *IEEE INFOCOM Workshop on Data Driven Intelligence for Networks*, Toronto, Canada, Jul. 2020, pp. 1248–1253.

[12] Y. S. Nasir and D. Guo, "Deep reinforcement learning for joint spectrum and power allocation in cellular networks," in *2021 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2021, pp. 1–6.

[13] ——, "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.

[14] K. Yang, D. Li, C. Shen, J. Yang, S.-p. Yeh, and J. Sydir, "Multi-agent reinforcement learning for wireless user scheduling: Performance, scalablility, and generalization," in *2022 56th Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2022, pp. 1169–1174.

[15] N. Naderializadeh, J. J. Sydir, M. Simsek, and H. Nikopour, "Resource management in wireless networks via multi-agent deep reinforcement learning," *IEEE Transactions on Wireless Communications*, vol. 20, no. 6, pp. 3507–3523, 2021.

[16] Y. Zhang and D. Guo, "Distributed MARL for scheduling in conflict graphs," in *2023 59th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2023, pp. 1–8.

[17] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2052–2062.

[18] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative Q-learning for offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1179–1191, 2020.

[19] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit Q-learning," *arXiv preprint arXiv:2110.06169*, 2021.

[20] P. Rashidinejad, B. Zhu, C. Ma, J. Jiao, and S. Russell, "Bridging offline reinforcement learning and imitation learning: A tale of pessimism," *Advances in Neural Information Processing Systems*, vol. 34, pp. 11 702–11 716, 2021.

[21] T. Xie, N. Jiang, H. Wang, C. Xiong, and Y. Bai, "Policy finetuning: Bridging sample-efficient offline and online reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 27 395–27 407, 2021.

[22] A. Kumar, J. Hong, A. Singh, and S. Levine, "When should we prefer offline reinforcement learning over behavioral cloning?" *arXiv preprint arXiv:2204.05618*, 2022.

[23] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped dqn," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[24] R. Y. Chen, S. Sidor, P. Abbeel, and J. Schulman, "UCB exploration via Q-ensembles," *arXiv preprint arXiv:1706.01502*, 2017.

[25] K. Lee, M. Laskin, A. Srinivas, and P. Abbeel, "Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2021, pp. 6131–6141.

[26] H. Hasselt, "Double Q-learning," *Advances in Neural Information Processing Systems*, vol. 23, 2010.

[27] 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.

[28] 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.

[29] G. An, S. Moon, J.-H. Kim, and H. O. Song, "Uncertainty-based offline reinforcement learning with diversified Q-ensemble," *Advances in Neural Information Processing Systems*, vol. 34, pp. 7436–7447, 2021.

[30] D. Ghosh, A. Ajay, P. Agrawal, and S. Levine, "Offline RL policies should be trained to be adaptive," in *International Conference on Machine Learning*. PMLR, 2022, pp. 7513–7530.

[31] A. Nair, A. Gupta, M. Dalal, and S. Levine, "Awac: Accelerating online reinforcement learning with offline datasets," *arXiv preprint arXiv:2006.09359*, 2020.

[32] A. Wagenmaker and A. Pacchiano, "Leveraging offline data in online reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2023, pp. 35 300–35 338.

[33] S. Lee, Y. Seo, K. Lee, P. Abbeel, and J. Shin, "Offline-to-online reinforcement learning via balanced replay and pessimistic Q-ensemble," in *Conference on Robot Learning*. PMLR, 2022, pp. 1702–1712.

[34] Y. Song, Y. Zhou, A. Sekhari, J. A. Bagnell, A. Krishnamurthy, and W. Sun, "Hybrid RL: Using both offline and online data can make RL efficient," *arXiv preprint arXiv:2210.06718*, 2022.

[35] Y. Luo, J. Kay, E. Grefenstette, and M. P. Deisenroth, "Finetuning from offline reinforcement learning: Challenges, trade-offs and practical solutions," *arXiv preprint arXiv:2303.17396*, 2023.

[36] M. Nakamoto, Y. Zhai, A. Singh, M. S. Mark, Y. Ma, C. Finn, A. Kumar, and S. Levine, "Cal-QL: Calibrated offline RL pre-training for efficient online fine-tuning," *arXiv preprint arXiv:2303.05479*, 2023.

[37] M. S. Mark, A. Ghadirzadeh, X. Chen, and C. Finn, "Fine-tuning offline policies with optimistic action selection," in *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022.

[38] 3GPP, "Simulation assumptions and parameters for FDD HeNB RF requirements," Tech. Rep. R4-092042.

[39] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[40] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[41] N. Naderializadeh and A. S. Avestimehr, "ITLinQ: A new approach for spectrum sharing in device-to-device communication systems," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, pp. 1139–1151, 2014.

[42] L. S. Waltrup, F. Sobotka, T. Kneib, and G. Kauermann, "Expectile and quantile regression—david and goliath?" *Statistical Modelling*, vol. 15, no. 5, pp. 433–456, 2015.

[43] D. Tarasov, V. Kurenkov, A. Nikulin, and S. Kolesnikov, "Revisiting the minimalist approach to offline reinforcement learning," *arXiv preprint arXiv:2305.09836*, 2023.

[44] D. Li, R. Huang, C. Shen, and J. Yang, "Near-optimal conservative exploration in reinforcement learning under episode-wise constraints," in *International Conference on Machine Learning*, Jul. 2023.

[45] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl: Datasets for deep data-driven reinforcement learning," 2021.

[46] G. Li, L. Shi, Y. Chen, Y. Chi, and Y. Wei, "Settling the sample complexity of model-based offline reinforcement learning," *arXiv preprint arXiv:2204.05275*, 2022.

[47] Y. Jin, Z. Yang, and Z. Wang, "Is pessimism provably efficient for offline RL?" in *International Conference on Machine Learning*. PMLR, 2021, pp. 5084–5096.

[48] L. Shi, G. Li, Y. Wei, Y. Chen, and Y. Chi, "Pessimistic q-learning for offline reinforcement learning: Towards optimal sample complexity," in *International conference on machine learning*. PMLR, 2022, pp. 19 967–20 025.

[49] C. Jin, Z. Yang, Z. Wang, and M. I. Jordan, "Provably efficient reinforcement learning with linear function approximation," in *Conference on learning theory*. PMLR, 2020, pp. 2137–2143.

[50] K. Yang, S.-p. Yeh, M. Zhang, J. Sydir, J. Yang, and C. Shen, "Advancing RAN slicing with offline reinforcement learning," *arXiv preprint arXiv:2312.10547*, 2023.

[51] Y. Chebotar, Q. Vuong, K. Hausman, F. Xia, Y. Lu, A. Irpan, A. Kumar, T. Yu, A. Herzog, K. Pertsch *et al.*, "Q-transformer: Scalable offline reinforcement learning via autoregressive Q-functions," in *Conference on Robot Learning*. PMLR, 2023, pp. 3909–3928.

Kun Yang is currently a Ph.D. student at the Charles L. Brown Department of Electrical and Computer Engineering, University of Virginia. He received his B.E. degree in the Electronic Information Science and Technology Department from Tsinghua University in 2017. He then received his M.S. degree in the Electrical Engineering department from Texas A&M University in 2019. His current research focuses on reinforcement learning for wireless communication, federated learning and prompt engineering.

Chengshuai Shi is currently a Ph.D. student at the Charles L. Brown Department of Electrical and Computer Engineering, University of Virginia. He received his B.E. degree in Electrical Engineering from the School of the Gifted Young, University of Science and Technology of China, in 2019. His current research focuses on multi-armed bandits, federated learning, and reinforcement learning.

Cong Shen (Senior Member, IEEE) is currently an Assistant Professor at the Charles L. Brown Department of Electrical and Computer Engineering, University of Virginia. He received his B.E. and M.E. degrees from the Department of Electronic Engineering, Tsinghua University, China, and his Ph.D. degree in Electrical Engineering from University of California, Los Angeles. His general research interests are in the area of communications, wireless networks, and machine learning. He received the National Science Foundation CAREER award in 2022, and the Best Paper Award in 2021 IEEE International Conference on Communications (ICC). He currently serves as an associate editor for the IEEE Transactions on Communications, IEEE Transactions on Green Communications and Networking, and IEEE Transactions on Machine Learning in Communications and Networking.

Jing Yang (Senior Member, IEEE) received the B.S. degree from the University of Science and Technology of China (USTC), and the M.S. and Ph.D. degrees from the University of Maryland, College Park, all in electrical engineering. She is an Associate Professor of electrical engineering with the Pennsylvania State University. She received the National Science Foundation CAREER award in 2015 and the IEEE WICE Early Achievement Award in 2020, and was selected as one of the 2020 N2Women: Stars in Computer Networking and Communications. She served as a Symposium/Track/Workshop Co-Chair for Asilomar 2023, ICC 2021, INFOCOM 2021-AoI Workshop, WCSP 2019, CTW 2015, PIMRC 2014, a TPC Member of several conferences, and an Editor for IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING, from 2017 to 2020. She is now an Editor for IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS and IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING. Her research interests lie in multi-armed bandits and reinforcement learning, federated learning, and wireless communications and networking.

Shu-ping Yeh (shu-ping.yeh@intel.com) is an AI-Applied Principle Engineer in the Wireless System Research Lab at Intel, where she conducts research on wireless broadband technologies. Her current research focus includes open RAN architecture, AI/ML for RAN control and management, network slicing and interworking of multiple radio access technologies within a network. Dr. Yeh has over 10 years of research and development experience in wireless industry and holds more than 40 US patents. She received her M.S. and Ph.D. in Electrical Engineering from Stanford University in 2005 and 2010, respectively.

Jaroslaw J. Sydir is an AI Applied Research Scientist at Intel Labs, where he is involved in research on distributed CNN video analytics pipelines and the application of reinforcement learning to resource allocation and management problems. Jaroslaw has worked in a variety of research, and hardware/software development roles including PC communications health prediction, predictive analytics for wafer handling robots, location awareness and prediction, control plane software, disk drive control software, and protocol stacks. Prior to Intel, Jaroslaw held positions at IBM, SRI International, and Cplane Inc. Jaroslaw is the co-inventor of over 50 patents.