2024 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN) | 979-8-3503-1764-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/DySPAN60163.2024.10632750

Advancing RAN Slicing with Offline Reinforcement Learning

Kun Yang*, Shu-ping Yeh‡, Menglei Zhang‡, Jerry Sydir‡, Jing Yang†, and Cong Shen*
* Department of Electrical and Computer Engineering, University of Virginia, USA
† School of Electrical Engineering and Computer Science, The Pennsylvania State University, USA
‡ Intel Corporation, USA

***Abstract*—Dynamic radio resource management (RRM) in wireless networks presents significant challenges, particularly in the context of Radio Access Network (RAN) slicing. This technology, crucial for catering to varying user requirements, often grapples with complex optimization scenarios. Existing Reinforcement Learning (RL) approaches, while achieving good performance in RAN slicing, typically rely on online algorithms or behavior cloning. These methods necessitate either continuous environmental interactions or access to high-quality datasets, hindering their practical deployment. Towards addressing these limitations, this paper introduces offline RL to solving the RAN slicing problem, marking a significant shift toward more feasible and adaptive RRM methods. We demonstrate how offline RL can effectively learn near-optimal policies from sub-optimal datasets, a notable advancement over existing practices. Our research highlights the inherent flexibility of offline RL, showcasing its ability to adjust policy criteria without the need for additional environmental interactions. Furthermore, we present empirical evidence of the efficacy of offline RL in adapting to various service-level requirements, illustrating its potential in diverse RAN slicing scenarios.**

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

I. Introduction

In the rapidly evolving landscape of wireless communication, Radio Access Network (RAN) slicing plays an important role in providing heterogeneous services to diverse wireless network users, offering a paradigm shift towards more flexible and efficient use of network resources. At its core, RAN slicing involves partitioning a single physical network into multiple virtual networks, each tailored to meet a set of specific service requirements. This flexibility is pivotal in addressing the diverse demands of modern wireless communications, ranging from high-speed data services to massive machine-type communications. By enabling dynamic allocation and optimization of network resources [1], [2], RAN slicing significantly enhances network efficiency, scalability, and service customization. It is a key technology in the evolution of wireless networks, facilitating the transition to more adaptive, service-oriented architectures [3], [4].

However, the implementation of RAN slicing introduces complex challenges, particularly in Radio Resource Management (RRM) [5], [6]. The need for customized and sophisticated RRM strategies becomes paramount to ensure the near-optimal performance of each network slice without compromising the overall network integrity. Recently, Reinforcement Learning (RL) has emerged as a promising tool [7]–[10], offering adaptive and intelligent solutions to navigate the intricate RRM landscape. The ability of RL to learn and make decisions based on dynamic network environments makes it ideally suited for managing the unique demand of each network slice.

Prior to our work, RRM for RAN slicing has predominantly utilized **online** RL algorithms [11]–[14], which requires intensive and continuous environmental interaction. Other works follow the same online setting, but mainly changes on neural network architectures [15]–[17] or extend to multi-agent settings for scalability [18]–[20]. In contrast, offline-based methods, which rely on behavior cloning [21]–[23], or training online RL algorithms offline [24], face the challenge of acquiring high-quality datasets or degraded training performance. Our research aims to bridge these gaps, showcasing the potential of offline RL in efficiently managing RAN slicing without the constraints of constant environmental interactions or dependency on high-quality expert data.

Offline RL [25]–[28], as aimed to develop RL policies with only offline datasets, offers a solution that is less interaction-intensive and more adaptive. Unlike online RL, offline RL significantly reduces the need for environmental interaction and mitigates a key drawback of online RL in real-world wireless systems. Besides this advantage, a growing investigation on data coverage of offline RL [29]–[31], proving a potential of recovering near-optimal policies with sub-optimal dataset. Combining these attractive features, there are recent efforts in developing tailored offline RL solutions to wireless RRM problems [10]. We note, however, that while [10] has explored the use of offline RL in RRM, their work does not address the flexibility in reward function adaptation, overlooked a key advantage of the offline RL when applying to real-world wireless systems.

In this paper, we introduce offline RL to solving RRM problems in RAN slicing, exploring its potential in addressing complex network management challenges. Our research focuses on the adaptability of offline RL when trained on diversely collected datasets. The contributions of this study are three-fold, each highlighting a distinct aspect of offline RL in the context of RRM in RAN slicing.

The work is partially supported by the US National Science Foundation under awards CNS-2002902, ECCS-2029978, SII-2132700, CNS-2003131, ECCS-2030026, ECCS-2143559, and Intel Corp.

Learning from Sub-optimal Data: We present findings indicating that offline RL has the potential to obtain near-optimal policies even when trained on sub-optimal datasets. This suggests that offline RL might be less dependent on the quality of data compared to traditional methods, a promising direction for scenarios where optimal data is not available.

Adapting to Different SLA Requirements: Our study explores how offline RL can potentially adapt to different SLA requirements. The result indicates that, by training across datasets with varying SLA conditions, offline RL could adjust its strategies to meet the specific needs of different network slices, a valuable feature for managing diverse network demands.

Behavioral Flexibility with Tailored Reward Functions: We also investigate the ability of offline RL to alter its behavior by changing the reward functions during offline training, even when using the same dataset. This result suggests that offline RL could offer a flexible approach to RRM, adapting to various operational objectives without the need for additional data.

The remainder of this paper is organized as follows. We discussed The formulation of the RRM problem for network slicing and how it can be posed in an RL framework in Section II. Section III details our experimental settings, highlighting how offline RL is applied and elucidating the mechanisms enabling SLA and objective adaptation within this context. Key insights and observations from the experiments are discussed in Section IV. Finally, Section V concludes the paper.

II. Problem Setting

We begin with an in-depth exploration of the wireless RRM problem that we aim to address. We emphasize the importance of flexibility in adapting to different SLA requirements and optimization objectives, illustrating why this adaptability is crucial for efficient RAN slicing systems. To facilitate a comprehensive understanding, we start with the context of RAN slicing in Section II-A, setting the stage for our analysis. Building upon this foundation, we then methodically illustrate how this RRM challenge can be aptly formulated in an RL problem in Section II-B. This formulation is pivotal as it lays the groundwork for applying advanced RL techniques, including offline RL methods, to effectively manage and optimize resource allocation in RAN slicing.

A. RRM for RAN Slicing

1) RRM as An Optimization Problem: We focus on a scenario within our system that involves one cell with N slices. Here, the first $N-1$ slices are designated as high-priority, while the final slice is allocated for background traffic. Each slice comprises a set of users, denoted as $k_1, k_2, \cdots, k_N$. Our analysis unfolds in discrete time slots, labeled as t, during which radio resources need to be allocated to the first $N-1$ slices. These resources are organized into block groups, with a total of M resource block groups (RBGs) available. The considered RRM system is illustrated in Figure 1.

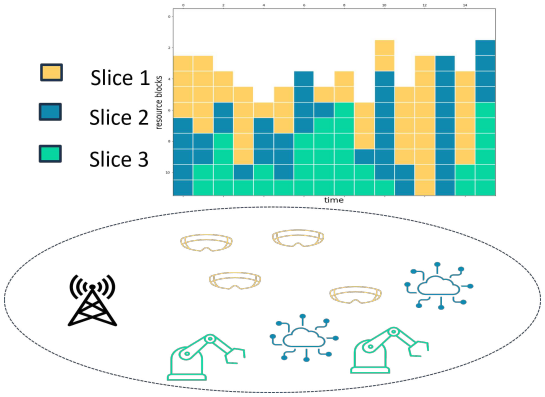


Fig. 1: An illustration of the RRM system in RAN slicing, where different resource blocks are allocated to different slices with distinctive purposes.

The primary objective of this RRM problem is the strategic allocation of these resources, represented by $\mathcal{M} = \{m_1, m_2, \cdots, m_{N-1}\}$, across the $N-1$ high-priority slices. In our particular implementation, following this allocation, a scheduler within the NS-3 framework takes over to further distribute the resources among the users in each slice.

The overarching goal of resource management is to strike an optimal balance in allocation, thereby fulfilling diverse Quality of Service (QoS) requirements captured by a utility function f_t. To formally represent this optimization challenge, we can articulate it as follows:

$$\begin{aligned} \underset{m_1, m_2, \cdots, m_{N-1}}{\text{maximize}} \quad & f_t(m_1, m_2, ..., m_{N-1}) \\ \text{subject to} \quad & \sum_{i=1}^{N-1} m_i \leq M. \end{aligned} \tag{1}$$

The optimization formulation (1) encapsulates the essence of the RRM problem in our system. It aims to maximize the QoS function f_t, considering the constraints on the total available resources. The formulation underscores the need to judiciously allocate resources across slices, ensuring that high-priority slices receive the necessary resources while also catering to the background traffic needs.

2) Choices of Resource Allocation: We now discuss the resource allocation strategies for the RAN slicing system. We categorize resources into two primary types: dedicated and prioritized [32], [33]. Dedicated resources are exclusively reserved for a specific slice and cannot be utilized by others. In contrast, prioritized resources, while initially allocated to a particular slice, may be used by other slices if any residual capacity remains. Based on these two distinct resource types, we draw our first two resource allocation strategies from the IETF report [34]. Besides these two strategies, we introduce a third strategy that is derived from the capabilities of the netgymenv simulator [35]. All three strategies are presented in the following:

1) **Hard Slicing [34]:** This strategy involves allocating only dedicated RBGs to each slice. While it simplifies the

system implementation, it can also lead to potentially inefficient resource utilization due to its rigid allocation.

2) **Limited Soft Slicing:** This approach utilizes only prioritized RBGs. It aims for more efficient resource usage by allowing the possibility of shared resources among slices, depending on the availability.
3) **Soft Slicing [34]:** A hybrid strategy that combines both dedicated and prioritized resources, offering a balance between resource efficiency and allocation specificity.

In this work, we recognize that while hard slicing offers simplicity and ease of implementation, it may not optimally utilize the available resources. On the other hand, soft slicing, though potentially more efficient, poses challenges in practical deployment due to its higher complexity. Given these considerations, we opt to focus on limited soft slicing as the primary approach. This choice is motivated by the aim to achieve a more resource-efficient allocation while maintaining a feasible level of system complexity.

B. Reinforcement Learning Formulation

As we have previously highlighted in Section I, RL has become a pivotal tool for addressing RRM challenges in RAN slicing. The core rationale for employing RL in RRM lies in its adeptness at navigating the dynamic decision-making processes, which is typical in resource management. Particularly in the context of RAN slicing, as detailed in Section II-A, the focus is on the sequential allocation of packed RBGs to optimize the QoS performance. The iterative learning and policy refinement capabilities of RL enable an agent to progressively navigate this complex decision space, ultimately leading to strategies that can significantly enhance resource utilization and overall network efficacy. This successful application of RL in RRM hinges on effectively formulating the problem as a Markov Decision Process (MDP).

Diverging from the conventional methodologies that often adhere to a predetermined structure in formulating the RRM problem as an MDP, our approach introduces an innovative and more flexible paradigm as shown in Figure 2. We break away from the standard practice of fixed observations, actions, and reward structures, instead adopting an adaptive process that is more reflective of real-world scenarios. Our system closely replicates a practical wireless network environment by capturing a comprehensive range of traffic monitoring metrics during data collection. This extensive dataset then undergoes a meticulous process of observation distillation and reward adjustment, tailored to extracting the most relevant information for the specified design objective. This nuanced approach not only brings a higher degree of realism to our simulator but also provides the adaptability necessary for a more customized RL formulation. This flexibility is critical, as it allows our system to adjust to various RRM scenarios, offering solutions that are more aligned with specific challenges and objectives encountered in real-world RAN slicing scenarios. In the following, we will elaborate on our distinct RL formulation, which we assert is well-suited for addressing the intricacies of the RRM problem.

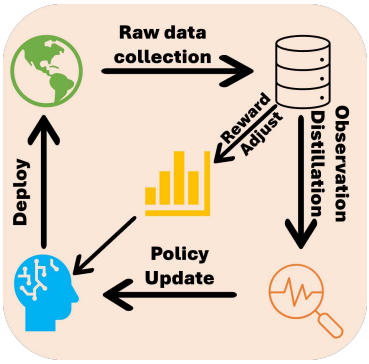


Fig. 2: Offline RL with reward adjustment and observation distillation.

- **States:** The wireless system will collect all possibly useful information from the environment, including user-level traffic load, user-level throughput, average one-way delay, maximum one-way delay, delay violation rate (with a designed threshold), resource block usage rate, and the relative location of the user. Among all the possibly useful states, we choose to collect slice-level information of throughput T_{rx}, traffic load T_{tx}, resource utilization rate U, delay violation rate D_{vio}, and average one-way delay D_{avg} from every slice in the system. The states can thus be specified as:

$$\{T_{\text{rx},i}, T_{\text{tx},i}, U_i, D_{\text{vio},i}, D_{\text{avg},i}\}_{i=1,\cdots,N}$$

- **Actions:** As stated in Section II-A, our goal is to allocate the RBGs to prioritized slices, and we choose to use the limited soft slicing technique so that we are allocating prioritized resources to prioritized slices, i.e. $A(t) = [a_1(t), \cdots, a_{N-1}(t)]$ where $a_i(t) \in [0, 1]$.
- **Reward:** The reward design of a RAN-slicing system should align with its QoS or the SLA. In our setting, we care about three components: the overall throughput of the system, the delay violation rates, and the resource utilization rate. We thus design a prioritized SLA-aware reward as the following. We first define a *priority vector* as $\mathbf{p} = [p_1, \cdots, p_i, \cdots, p_N]$. Then the reward is given as

$$R(t) = \sum_{i=1}^{N} p_i r_i(t),$$

where

$$r_i(t) = T_{\text{rx},i}(t) - \alpha D_{\text{vio},i}(t) - \delta U_i(t).$$

In our experiment, we initially set $\mathbf{p} = [\frac{1}{N-1}, ..., \frac{1}{N-1}, ..., 0]$, $\alpha = 4$, and $\delta = 1$.

We note that the reward design incorporates a flexible mechanism for adjusting the priority of different slices via the priority vector $\mathbf{p}$. Additionally, it allows fine-tuning of the significance of each key component – throughput, delay violation, and resource utilization – using hyperparameters α and δ. This flexibility is crucial for customizing the behavior

TABLE I: Experiment parameters

Parameter	Value
Number of Slices	3
Number of UEs	$6 - 20$
Delay violation threshold	[100, 50, 10] ms
Area	120×10 m^2
Downlink traffic	2 Mbp/s
Traffic pattern	Poisson arrival
UE mobility	$1 - 2$ m/s

of an RL agent to match the specific network conditions and SLA requirements. By varying these parameters, the reward function can be tailored to emphasizing different aspects of the network performance, thus ensuring that the learning process of an RL agent is aligned with the overarching goals of the RAN slicing system. This carefully crafted design enables the RL model to adaptively balance between maximizing throughput, minimizing delay violations, and optimizing resource usage, in accordance with the defined priorities and operational constraints.

Based on the MDP design, the objective of the RL system is given as:

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(t)\right].$$

III. Experimental Setup and Results

A. Simulation Setup and Baseline Methods

We now detail the experimental setup and the initial strategies employed for data collection and performance evaluation. Our experiments are conducted using the **`netgymenv` simulator** developed by Intel [35], as mentioned in Section II. This simulation environment focuses on a RAN slicing system comprising one cell and N slices. The traffic module follows the LTE module in NS-3 [36]. Specifically, our experimental setup involves a system with three slices: the first two slices are prioritized for essential services, while the third slice handles background traffic. The key parameters of our simulation are summarized in Table I.

For delay violation rate data collection, the simulator does not record individual packet delays due to the high computational overhead. Instead, it maintains a histogram of packet arrivals over a specific period of time, from which the delay violation rate is calculated.

Regarding data collection for offline RL training, we deploy the following behavior policies (BPs), which also serve as baselines for comparative evaluation.

1) **Traffic load-based resource allocation:** Resources are allocated proportionally based on the traffic load observed in the previous time period:

$$a_i(t) = \frac{T_{\text{tx},i}(t-1)}{\sum_j T_{\text{tx},j}(t-1) + \Delta}, a_i(0) = \frac{1}{N},$$

where Δ is a small positive number to avoid numerical instability.

2) **Delay violation rate-based resource allocation:** Here, resource allocation is proportionally based on the delay violation rates, using a softmax function for more stable allocation:

$$a_i(t) = \frac{\exp D_{\text{vio},i}(t-1)}{\sum_j \exp D_{\text{vio},j}(t-1)}, a_i(0) = \frac{1}{N}$$

3) **Online RL:** We also include an online RL policy as a BP for both 'expert' dataset collection and performance benchmarking. The specific algorithm we choose to use is Soft Actor-Critic (SAC) [37], selected for its actor-critic structure that aligns with the offline RL algorithms we deploy.

The formulation of the first two baseline methods, Traffic Load-Based and Delay Violation Rate-Based Resource Allocation, is specifically designed to incorporate limited aspects of system information. As a result, while they are adept at optimizing certain traffic patterns, their performances may be sub-optimal when the broader system dynamics are taken into consideration. These methods, therefore, serve as useful starting points but may not fully capture the complexity and variability of real-world traffic scenarios.

In contrast, the online RL policy, specifically the SAC algorithm, is employed with the intention of reaching a more comprehensive system optimization. To ensure its effectiveness, we commit to training this policy over an extended period, allowing it to adapt and learn from a wide range of network conditions and scenarios. This extended training is crucial for the online RL policy to develop a detailed understanding of the system and achieve near-optimal performance, potentially exceeding the more narrowly focused baseline methods. The comparison between these baseline strategies and the more dynamically trained online RL policy will provide valuable insights into the effectiveness and adaptability of different resource allocation approaches in RAN slicing.

Data Collection for Offline Datasets: To construct comprehensive offline datasets, we systematically execute each BP for 40 episodes. An individual episode comprises 200 continuous steps. At the beginning of each episode, we introduce variability by randomly selecting both the number of users and the random seed for the prioritized slices, while maintaining a constant user count of 5 for the background slice. This approach ensures a diverse dataset that encapsulates a range of possible network states and user behaviors. By adopting this method for each BP under varying SLA requirements, we accumulate a substantial dataset of approximately 80,000 data samples per BP under the same SLA requirement. This extensive collection forms the foundational dataset for training our offline RL algorithms, providing them with a broad spectrum of scenarios to learn from and adapt to. The diversity and volume of this dataset are critical for enabling the offline RL models to develop robust and effective resource allocation strategies that can cater to the dynamic and complex needs of RAN slicing.

TABLE II: Training parameters

Parameter	Value
Actor structure	2 Layer MLP with hidden dimension 64
Critic structure	2 Layer MLP with hidden dimension 64
Critic learning rate	$1e^{-3}$ for online, $3e^{-4}$ for offline
Actor learning rate	$3e^{-4}$ for online, $5e^{-5}$ for offline

B. Offline RL with Sub-optimal Datasets

The initial phase of the offline RL experimentation employs Conservative Q-learning (CQL) [38], a method widely known for its straightforward implementation and the pessimistic approach to offline RL training, characterized by the inclusion of a regularizer in its loss function. The specific design of CQL makes it an ideal algorithm to use as a starting point for offline RL experiments, particularly for evaluating the efficacy of offline RL in deriving practical algorithms. The training details for the online and offline RL clients are given in Table II.

Our experimental setup in this section is tailored to a scenario where both prioritized slices have identical SLA levels, though the number of users in each slice can vary. To maintain a level playing field in our comparative analysis, we standardize the definition of a training step across both online and offline RL. For online learning, a step involves collecting a data sample from the environment followed by a mini-batch gradient descent update. In the offline RL context, a step is defined as sampling a mini-batch from the dataset and performing a corresponding mini-batch gradient descent update.

As depicted in Figure 3a, we present the performance results of the offline RL model, which utilizes an expert dataset collected via a SAC agent. Intriguingly, the results indicate that offline RL, when trained on a dataset acquired from a high-performing online RL agent, can surpass the performance of its online counterpart within the same number of training steps. While this outcome is noteworthy, it is important to remember that access to an expert-level dataset is not always feasible in practical systems. Therefore, a more critical assessment of the capabilities of offline RL lies in its ability to learn effective policies from *sub-optimal* datasets.

In our exploration of offline RL with sub-optimal datasets, we conduct tests using the two baseline strategies outlined in Section III-A. Initially, we train an offline RL agent separately on datasets generated from each of these baselines. Subsequently, we combine these two datasets to assess any potential performance benefits from this mixed dataset approach.

The outcomes of these training exercises on the sub-optimal datasets are illustrated in Figure 3b. From these results, we observe that the performance achieved with the sub-optimal datasets notably surpasses that of the corresponding BPs, yet falls marginally short of the expert-level performance. This finding underscores the capability of offline RL to extract valuable learning even from less-than-ideal data.

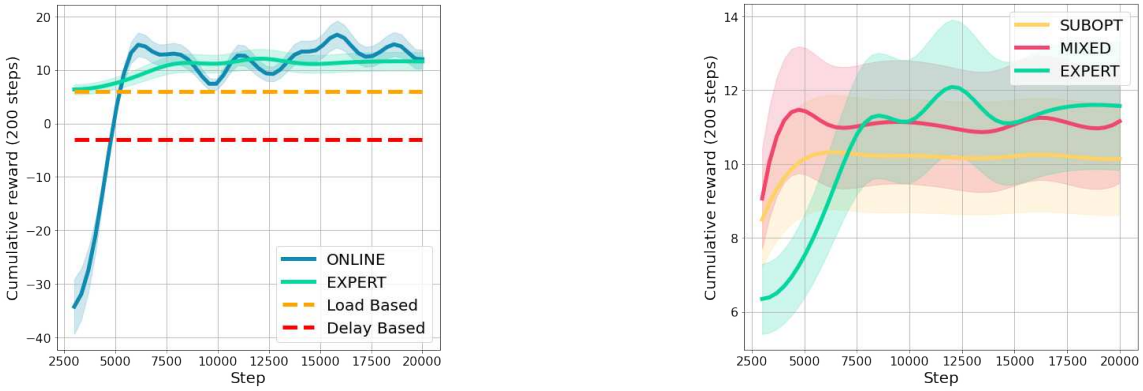
Furthermore, an interesting development emerged when we amalgamate the load-oriented and delay-oriented baseline datasets for training. This blend of datasets, encompassing a broader spectrum of network scenarios and challenges, enabled the offline RL agent to approach, and in some cases match, the performance level of the expert-level dataset. This enhancement in performance indicates that *diversity* and *comprehensiveness* in the training dataset play a crucial role in the efficacy of offline RL. It suggests that by judiciously combining datasets with varied characteristics, we can equip the offline RL model with a richer learning experience, thus enabling it to develop more robust and effective strategies that are closer to those derived from optimal conditions. This approach might be particularly beneficial in practical scenarios, where access to expert-level data is limited, and reliance on diverse sub-optimal data sources is more realistic.

Behavior Understanding and Performance Analysis: Beyond the encouraging cumulative reward outcomes observed during the training phase, it is crucial to delve deeper into the underlying factors contributing to the superior performance of the RL algorithms over the baselines. To this end, we conduct a thorough analysis across 20 distinct environments, encompassing five different user distributions, each evaluated with four unique random seeds. This comprehensive test allows us to assess the robustness and adaptability of the policies in various scenarios. The results, focusing on key metrics like throughput and delay violation rate, are presented in Figure 4. As illustrated in the figure, the RL algorithms demonstrate the ability to sustain the throughput performance (0.3 Mbp/s drop total throughput wise) while simultaneously achieving a significant reduction in delay violation rates, with a relative improvement of approximately 50%. This impressive feat is attributed to the adoption of a more conservative resource allocation strategy compared with the load or delay-aware baselines. Unlike these baseline methods, which may resort to drastic adjustments leading to increased delay violations or throughput drops, the RL algorithm implements a more balanced approach. This strategy effectively mitigates the risk of extreme resource allocation decisions that could be detrimental to overall system performance. The results clearly showcase the proficiency of RL in not only maintaining service quality but also significantly improving network reliability and user experience by reducing delay violations, a critical aspect in RAN slicing environments.

The experiment revealed an intriguing outcome: despite being designed to allocate resources based on delay violation rate, the delay-based baseline exhibited unstable and drastic allocation changes. This led to increased delay violations, contrary to our initial intent. Notably, this supports our observation that the effectiveness of RL policies stems from a conservative allocation approach. Thus, the results suggest that prioritizing a conservative strategy could more effectively balance throughput and delay violations.

C. Adaptation to Different SLA Requirements

In the preceding section, we have established that offline RL can surpass online RL in scenarios with consistent SLA requirements, demonstrating the advantage of offline RL in



(a) Expert dataset.

(b) Sub-optimal datasets.

Fig. 3: Offline training results with different datasets

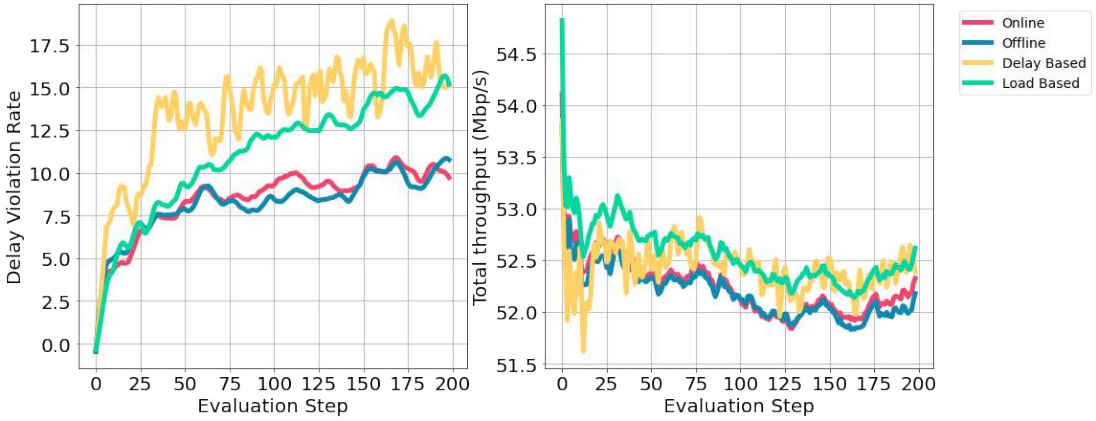


Fig. 4: Delay violation rate (average of two slices) and total throughput (sum of two slices). The results are averaged over 20 different environments as discussed in Section III-B.

leveraging mixed sub-optimal datasets. In practical RAN slicing systems, however, heterogeneity is a result of not only different BPs but also distinct SLA requirements. In this context, we explore how offline RL adapts to varying SLAs within a RAN slicing system.

Specifically, we adjust the SLA requirements for Slice 1 by reducing the delay violation threshold from 100 ms to 50 ms and further down to 30 ms. Our goal is to investigate if an offline RL policy can effectively utilize datasets collected under different SLAs to adapt to new SLA conditions that have not been seen before.

For this experiment, we train an offline RL policy using data collected at delay violation thresholds of 100 ms and 50 ms. We then test this policy in an environment where Slice 1 has a delay violation threshold of 30 ms. The performance of this offline RL policy is compared against an online RL policy: one trained in an environment with an exact 30 ms threshold. The comparative performance is illustrated in Figure 5. The result reveals that both RL methods successfully adapt to the new SLA requirement. Notably, both methods slightly compromise the performance on Slice 2 to mitigate substantial delay violations resulting from the altered SLA on Slice 1. In Figure 5, a trade-off of less than 1% in the mean delay violation rate for Slice 2 leads to a reduction of over 10% in delay violations for Slice 1, translating to a relative improvement of over 100% compared with the best-performing baseline. It is more exciting that this is accomplished while the offline policy has never seen any data collected from 30 ms SLA environment. Despite having no prior exposure to this specific SLA requirement, it manages to achieve a performance level comparable to that of the online RL agent, which necessitates tens of thousands of steps of environmental interaction.

D. Variation on Reward Functions

One unique advantage of offline RL in the context of RAN slicing lies in its ability to adjust the reward function based on existing datasets, thereby enabling tailored policy behavior. For instance, in our initial setup, we set the parameters $\alpha = 4$ and $\delta = 1$ to emphasize the impact of delay violation. To shift the focus towards throughput, we adjust the reward parameters to $\alpha = 0.5$ and $\delta = 0.5$. Subsequently, to underscore resource usage, we modify them to $\alpha = 1$ and $\delta = 4$. These modifications in the reward function are expected to

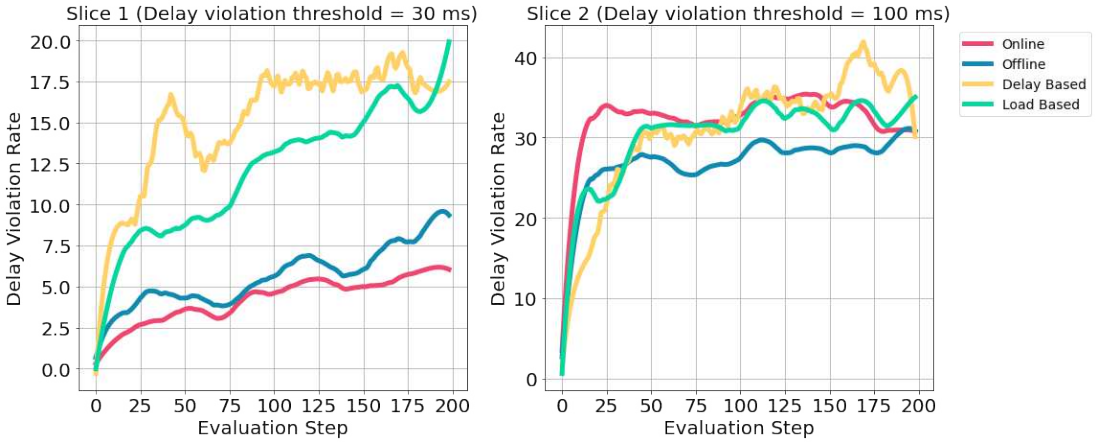


Fig. 5: Delay violation rate comparison across slices with different SLA requirements.

significantly influence the behavior of the offline RL policy, as demonstrated in Table III.

The result in Table III reveals that adapting the reward parameters indeed changes the policy behavior. The $CQL_{\text{throughput}}$ policy, with a reduced emphasis on delay violation, yields the highest total throughput but at the cost of increased delay violations. Meanwhile, CQL_{delay}, with its focus on minimizing delays, has the lowest mean delay violation rate. Interestingly, the CQL_{resource} policy, aimed at optimizing resource usage, does not demonstrate a marked improvement in resource efficiency. This is particularly notable in our tested limited soft-slicing system, where shared resources inherently limit the scope for significant optimization. This finding suggests that the system architectural setup – soft versus hard slicing – plays a critical role in determining the efficacy of different reward-oriented policies. In hard-slicing environments, where resources are exclusively allocated, the potential for a resource-oriented policy to improve utilization may become more pronounced.

IV. Discussion

We discuss several key insights gleaned from the experiments with offline RL in the context of RAN slicing.

Efficiency Gains through Reduced Interactions: Our use of the `netgymenv` simulator, a near-real-world tool developed by Intel Labs that is based on NS-3 [36], offers practical and credible experimental outcomes. However, it also highlights the increased cost associated with environmental interactions in more sophisticated simulators. In contrast to lightweight simulators where interactions are nearly cost-free and instantaneous, each interaction with `netgymenv` incurs a substantial delay of approximately 200 ms, encompassing both simulation and communication overheads (this could be optimized with improved networking solutions). In comparison, a single neural network update step takes about 100 ms. Consequently, offline RL, in addition to reducing resource interactions and circumventing sub-optimal exploratory steps inherent in online RL, offers significant time savings. In our experiments, offline RL achieves a time reduction of at least 50%, potentially extending up to 67%.

The Crucial Role of State Normalization: As outlined in Section II, the initial state definitions utilize raw data metrics including throughput, delay violation rate, and delay. However, we have observed that directly using these raw values could lead to instability in the RL model convergence, sometimes resulting in the policies getting trapped at sub-optimal performance levels. To address this issue, we have found that normalizing the state values is extremely beneficial. By scaling all state values to a normalized range of $[0, 1]$, we significantly enhance the stability of the training process. Therefore, we advocate for the implementation of state normalization in future experiments involving RL, as it markedly improves training stability and the overall effectiveness of the RL model.

V. Conclusion

This work established that offline RL can effectively extract (near)-optimal policies from sub-optimal datasets, highlighting a key advantage over online RL, especially in the context of real-world wireless systems. This is attributed to the capability of offline RL to operate without the need for costly environmental interactions. Additionally, we observed that offline RL is adept at adapting to varying Service Level Agreement (SLA) requirements, demonstrating promising transferability even to previously unseen SLA scenarios. Another important finding was the flexibility of offline RL in policy adjustment. By altering reward functions offline, it is feasible to train multiple policies with distinct objectives. We believe this work is helpful in enhancing the current workflow of applying RL to RRM in RAN slicing systems.

References

[1] 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.

[2] F. Liang, C. Shen, W. Yu, and F. Wu, "Towards optimal power control via ensembling deep neural networks," *IEEE Trans. Commun.*, vol. 68, no. 3, pp. 1760–1776, March 2020.

TABLE III: Performance Comparison for Different Reward-oriented Offline RL Policies

	Mean delay violation rate	Total throughput (Mbp/s)	Mean resource usage
CQL_{delay}	$\mathbf{6.5 \pm 3.5}$	52.48 ± 13.65	49.15
$CQL_{\text{throughput}}$	9.1 ± 4.4	$\mathbf{58.68 \pm 11.23}$	49.35
CQL_{resource}	7.3 ± 4.1	51.44 ± 12.68	**48.89**

[3] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5g: Survey and challenges," *IEEE communications magazine*, vol. 55, no. 5, pp. 94–100, 2017.

[4] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2429–2453, 2018.

[5] T. Guo and A. Suárez, "Enabling 5g ran slicing with edf slice scheduling," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2865–2877, 2019.

[6] T. Akhtar, C. Tselios, and I. Politis, "Radio resource management: approaches and implementations from 4g to 5g and beyond," *Wireless Networks*, vol. 27, pp. 693–734, 2021.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[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] C. Shen and M. van der Schaar, "A learning approach to frequent handover mitigations in 3GPP mobility protocols," in *IEEE Wireless Communications and Networking Conference (WCNC)*, March 2017, pp. 1–6.

[13] Z. Wang and C. Shen, "Small cell transmit power assignment based on correlated bandit learning," *IEEE J. Select. Areas Commun.*, vol. 35, no. 5, pp. 1030–1045, May 2017.

[14] 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.

[15] Y. Hua, R. Li, Z. Zhao, X. Chen, and H. Zhang, "GAN-powered deep distributional reinforcement learning for resource management in network slicing," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 334–349, 2019.

[16] A. T. Z. Kasgari, W. Saad, M. Mozaffari, and H. V. Poor, "Experienced deep reinforcement learning with generative adversarial networks (GANs) for model-free ultra reliable low latency communication," *IEEE Transactions on Communications*, vol. 69, no. 2, pp. 884–899, 2020.

[17] R. Li, C. Wang, Z. Zhao, R. Guo, and H. Zhang, "The LSTM-based advantage actor-critic learning for resource management in network slicing with user mobility," *IEEE Communications Letters*, vol. 24, no. 9, pp. 2005–2009, 2020.

[18] T. Hu, Q. Liao, Q. Liu, D. Wellington, and G. Carle, "Inter-cell slicing resource partitioning via coordinated multi-agent deep reinforcement learning," in *ICC 2022-IEEE International Conference on Communications*. IEEE, 2022, pp. 3202–3207.

[19] H. Zhou, M. Elsayed, and M. Erol-Kantarci, "RAN resource slicing in 5G using multi-agent correlated Q-learning," in *2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 2021, pp. 1179–1184.

[20] 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.

[21] A. M. Nagib, H. Abou-Zeid, and H. S. Hassanein, "Transfer learning-based accelerated deep reinforcement learning for 5G RAN slicing," in *2021 IEEE 46th Conference on Local Computer Networks (LCN)*. IEEE, 2021, pp. 249–256.

[22] Q. Liu, N. Choi, and T. Han, "OnSlicing: online end-to-end network slicing with reinforcement learning," in *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT)*, 2021, pp. 141–153.

[23] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "ColO-RAN: Developing machine learning-based xapps for open RAN closed-loop control on programmable experimental platforms," *IEEE Transactions on Mobile Computing*, vol. 22, no. 10, pp. 5787–5800, 2022.

[24] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "ColO-RAN: Developing Machine Learning-based xApps for Open RAN Closed-loop Control on Programmable Experimental Platforms," *IEEE Transactions on Mobile Computing*, pp. 1–14, July 2022.

[25] 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.

[26] W. Xiong, H. Zhong, C. Shi, C. Shen, L. Wang, and T. Zhang, "Nearly minimax optimal offline reinforcement learning with linear function approximation: Single-agent MDP and markov game," in *International Conference on Learning Representations (ICLR)*, May 2023.

[27] C. Shi, W. Xiong, C. Shen, and J. Yang, "Provably efficient offline reinforcement learning with perturbed data sources," in *International Conference on Machine Learning (ICML)*, July 2023.

[28] 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 (ICML)*, July 2023.

[29] 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.

[30] 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.

[31] 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.

[32] M. Dighriri, A. S. D. Alfoudi, G. M. Lee, T. Baker, and R. Pereira, "Resource allocation scheme in 5g network slices," in *2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA)*. IEEE, 2018, pp. 275–280.

[33] H. Ko, J. Lee, and S. Pack, "Priority-based dynamic resource allocation scheme in network slicing," in *2021 International Conference on Information Networking (ICOIN)*. IEEE, 2021, pp. 62–64.

[34] L. Geng, J. Dong, S. Bryant, K. Makhijani, A. Galis, X. de Foy, and S. Kuklinsk, "Network slicing architecture," Internet Engineering Task Force, Internet-Draft draft-geng-netslices-architecture-02, 2017, available online: https://datatracker.ietf.org/doc/html/draft-geng-netslices-architecture-02.

[35] M. Zhang and J. Zhu, "NetworkGym: Democratizing Network AI via Sim-aaS," https://intellabs.github.io/networkgym/, 2023.

[36] G. F. Riley and T. R. Henderson, "The NS-3 network simulator," in *Modeling and tools for network simulation*. Springer, 2010, pp. 15–34.

[37] 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.

[38] 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.