Reinforcement Learning for Radio Resource Management in RAN Slicing: A Survey

Mohammad Zangooei, Niloy Saha, Morteza Golkarifard and Raouf Boutaba
{mzangooei, n6saha, mgolkari, rboutaba}@uwaterloo.ca
University of Waterloo, Canada

***Abstract*—Dynamic radio resource allocation to network slices in mobile networks is challenging due to the diverse requirements of RAN slices and the dynamic environment of wireless networks. Reinforcement learning (RL) has been successfully applied to solve different network resource allocation problems where an agent learns how to choose the best action from the interactions with the environment. This survey studies the state-of-the-art RL approaches that address radio resource management in radio access network (RAN) slicing. To this end, we first categorize different problem definitions based on the network environment. Then, we explain how each environment can be modeled as a Markov decision process (MDP) and what RL algorithms can be used to solve them. In addition, we discuss the challenges present in existing works and suggest strategies to address them.**

***Index Terms*—RAN Slicing, Radio Resource Management, Reinforcement Learning, Markov Decision Process, Next-generation Networks.**

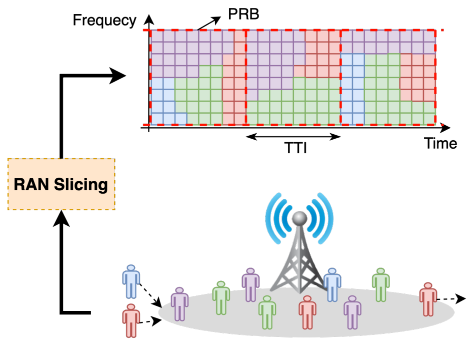

Fig. 1. Radio resource allocation to requests of different slices. Users of different slices (depicted by distinct colors) arrive at the BS coverage over times. RAN slicing module decides how physical resource blocks (PRBs) should be dynamically allocated to slices.

I. Introduction

Next-generation mobile networks are envisioned to support a broad range of services with diverse requirements such as high data rate, ultra-reliability, sub-millisecond latency, and a large number of connections [1], [2]. Meeting all these stringent and heterogeneous requirements is challenging and cannot be met using a shared one-size-fits-all network setting. Network slicing enabled by software-defined networking and network function virtualization become a promising solution that provides these services by allowing multiple logical networks, *i.e.*, network slices to operate on top of the same physical infrastructure [1], [2].

Network slicing involves both the core network and radio access network (RAN). Network slicing in the core network, *i.e.* core slicing, is well studied and can be achieved by scaling up computing resources; however, network slicing in RAN, *i.e.* RAN slicing, still remains a challenging problem due to the scarcity of radio resources, dynamic conditions of wireless channels, and interference. Efficient management of radio resources improves service delivery and utilization of network resources, resulting in higher revenues and lower costs for mobile network operators (MNO).

Slicing-aware radio resource management (RRM) can be seen as a two-level hierarchical problem: inter-slice and intra-slice resource allocations. The former problem determines the share of radio resources that should be allocated to network slices and the latter schedules the radio resources among different users of the same slice [1], [2]. An overview of the slicing-aware RRM problem is depicted in Figure 1 where RAN slicing module is responsible for satisfying the demands by dynamically and efficiently allocating radio resources to requests.

Reinforcement learning (RL) techniques have received considerable attention for solving the RAN slicing problem since traditional methods fail to deal with the problem's complexities. Firstly, traditional methods require a closed-form formulation that explains the relationship between resource allocation and service-level agreement (SLA) satisfaction. However, there exists no such accurate model due to the heterogeneity of slice SLAs, the stochasticity of wireless communications, and the complexity of the underlying queuing-based resource-sharing methods. Consequently, approximated formulations are commonly utilized despite accuracy concerns. In contrast, model-free RL methods do not need such a model; instead, they proactively interact with the environment to develop a model within their available computation capacity [3], [4].

Secondly, future slice demands are unknown and need to be estimated at RAN slicing decision time. Nonetheless, traditional methods fall short in demand estimation for all possible scenarios. Contrarily, relying on deep neural networks, RL algorithms can adapt themselves to different environments and implicitly account for future demands [5].

Nonetheless, RL methods should be carefully leveraged to address RRM problems. Not all RL-based propositions follow the same problem formulations and exploit the same algorithms to attack the problem. These different approaches come with their advantages and disadvantages, which will be discussed in Section III. Moreover, we identify particular challenges in

Authorized licensed use limited to: University of Waterloo. Downloaded on February 14,2023 at 17:31:44 UTC from IEEE Xplore. Restrictions apply.

effectively defining the corresponding Markov decision process (MDP) and leveraging RL techniques to solve the RAN slicing problem in Section IV. In the same section, we suggest methods for coping with these challenges based on the existing proposals in the literature.

II. Related work

Traditional methods including queuing theory, Lagrange methods for optimization, Thompson sampling, genetic methods, and heuristic methods, have been used to solve RAN slicing problems [4], [6]. However, these methods are not well suited to the specific characteristics of next-generation mobile networks and their heterogeneous service requirements.

First of all, the approximate mathematical models that are used in traditional optimization methods cannot thoroughly represent complex dynamics in real networks [7], [8]. Furthermore, the complexity of wireless networks is growing with increasing device numbers, evolving communication models, and heterogeneous quality of service (QoS) requirements [3]–[5], [7], [9], [10]. According to [9], the number of settings required for optimization has increased from 1500 settings for 4G to more than approximately 2000 for 5G.

Moreover, high computational complexities of searching in large, complex, and dynamic scenarios often lead to suboptimal performance of the mentioned methods [4], [9], [11]. Lastly, network conditions often demonstrate long-term and short-term trends due to the dynamics of service traffic and the physical layer of networks, respectively. But traditional methods do not adapt to such hidden network dynamics as they lack the ability to learn data patterns [7].

With all of these limitations, researchers have recently resorted to RL-based approaches. Several RL-based RAN slicing methods have been proposed in the literature to consider a delicate investigation into them. A group of related survey papers discuss the application of machine learning (ML) in network slicing, while other similar papers [1], [2] specifically investigate the application of ML in RAN slicing, which are closer to this article. Such studies provide neither insights on how RL algorithms should be exploited nor a comparative investigation of different RL components in RAN slicing. Specifically, they offer a general overview of ML methods in user and slice admission control, resource scheduling, energy efficiency, isolation, resource virtualization, and power management in slicing. Besides, they cover a limited number of RL-based proposals and only consider single-cell scenarios.

To fill this gap, we offer a comprehensive review of RL-based RAN slicing proposals. Notably, we identify the single-cell and multi-cell taxonomies, discuss the definition of MDP components and the underlying advantages and disadvantages, argue the trade-off between different RL algorithms, and provide the crucial challenges in the effective use of RL for the RAN slicing problem as well as the suggested solutions for them in the literature.

III. Problem Formulation Taxonomy

Formulating an RL problem is to define the corresponding MDP, and the triple of (state, action, reward), which characterizes how an RL agent interacts with the environment. The state is the set of variables the agent observes from the environment and takes action based on that. The ultimate goal of the RL agent is to maximize the cumulative reward it receives from the environment upon taking action per step in an episode of interactions.

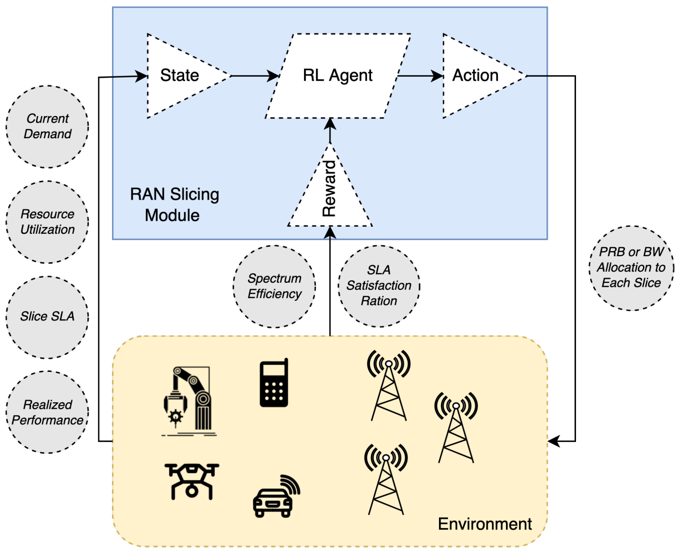

Fig. 2. Elements of MDP in radio resource management problems.

In this article, we focus on the inter-slice resource allocation problem where propositions in the literature can be classified into two general groups: single-cell and multi-cell RAN slicing.

In the first group, the algorithm is locally implemented in the base station (BS) itself, which brings the following advantages: (i) the problem is decomposed into different entities, so the complexity is reduced, and (ii) no communication is conducted between the BSs and a designated controller, which mitigates the latency and overhead of the control loop. As a result, RAN slicing quickly adapts to changes in the system's state.

Contrarily, coordinating neighboring BSs in the second group of proposals provides the following benefits: (i) resource compensation in case of congestion [12], and (ii) interference management to avoid service degradation [13]. These two different problem formulations and RL-based algorithms are elaborated in III-A and III-B.

A. Single-cell RAN Slicing RL Formulation

A single-cell RAN slicing problem models how to distribute the available radio resources in the form of physical resource blocks (PRBs) between different slices to satisfy their KPI requirements. In what follows, we extensively discuss how different elements of the MDP are defined and RL algorithms are exploited to solve RAN slicing in the single-cell scenario. An illustration of RL agent interaction with the single-cell RAN slicing environment is depicted in Figure 2, and a summary of the state, action, reward, and RL algorithm options is presented in Table I.

State. Variables that have been used as the MDP state in the literature can be categorized into four different classes capturing demand, resource utilization, slice realized performance, and slice requirements. Specifically, the number of arrived (buffered)

Authorized licensed use limited to: University of Waterloo. Downloaded on February 14,2023 at 17:31:44 UTC from IEEE Xplore. Restrictions apply.

TABLE I
SUMMARY OF STATES, ACTIONS, REWARDS, AND RL ALGORITHMS USED IN THE LITERATURE FOR SINGLE-CELL RAN SLICING

Ref.	State	Action	Reward	RL Algorithm
[3]	number of arrived packets in each slice	per slice number of PRBs	weighted sum of latency and throughput of slices	PPO, DQN, Dueling DQN, AC, and A2C
[4]	number of arrived packets in each slice	per slice number of PRBs	weighted sum of spectrum efficiency and SSR	GAN-DDQN and Dueling GAN-DDQN
[5]	number of arrived packets in each slice	per slice number of PRBs	a handcrafted function of spectrum efficiency and SSR	LSTM-A2C
[6]	particular state for eMBB and mMTC slices	per slice number of PRBs	SLA satisfaction	Model-Based RL with Kernels
[7]	number of users in each slice	per slice number of PRBs	total throughput	Adaptive IPO and TRPO
[9]	PRB allocation and usage, SLAs, SSR, number of arrived, buffered, and sent packets	per slice number of PRBs	SSR × spectrum efficiency	DQN
[10]	particular state for eMBB and uRLLC	per slice number of PRBs	particular reward for eMBB and uRLLC	correlated DQN

packets [3] [4] [5] [6] [9], transmitted packets [9], and users [7] [6] in each slice advise the agent of the current demand per slice whilst PRB usage ratio and the number of allocated PRBs [9] [6] give hint about resource utilization. Furthermore, throughput and delay requirement [9] highlight requests of each slice whereas slices' SLA satisfaction ratio (SSR) [9] [6] indicates the realized performance of each slice.

There is a consensus in the literature as to include the demand metrics in the MDP state while the other three types of metrics are not present in every RL-based proposal. On the one hand, providing a comprehensive view of the system comprising slices' demand, resource allocation, and SLA requirements and satisfaction guides the agent to find the relationship between them. On the other hand, including many variables in the MDP raises dimensionality concerns, leading to convergence issues.

Action. According to the literature [3]–[7], [9], [10], the number of PRBs per slice is recommended as the action of RL agent.

Reward. SSR and Service efficiency (SE) are the major components of the proposed reward functions in the literature. The reward of the agent in [4] is a weighted sum of SE and SSR in different slices; however, authors in [5] claim that if a linear combination of SSR and SE is used, it could lead to a blind sacrifice of SSR in exchange for an increase in SE, resulting in SLA violations. In response, they propose a handcrafted reward function to avoid such entanglement. Following the same objective, the reward is defined as SSR × SE in [9] to keep both metrics high at the same time. Additionally, constrained RL approaches provide a means to separate the SSR from SE in a such way that one can define SLA-related constraints to keep SLA violations below the intended threshold while optimizing for higher SE. Such a method is shown to be superior to the previous ones in dealing with these two conflicting objectives [7].

Reward in [3] only encourages higher resource utilization without considering SLAs, hindering multiplexing gains in RAN slicing. On the other hand, in a multi-agent framework, specialized rewards are defined per slice type to better capture the corresponding service type: eMBB's reward is proportional to the throughput sum [10] and SLA satisfaction in terms of average buffer length, data rate, and PRB usage [6] while the reward is inversely proportional to the queuing delay for the uRLLC [10] and mMTC [6] agents.

RL algorithm. The choice of RL algorithm is also of paramount importance as it affects the agent's convergence in terms of speed and stability. In this part, we discuss various aspects of the RL algorithms that have been exploited in RAN slicing literature.

Value-learning vs. Policy-gradient. In value learning methods, observed rewards are exploited to fit a value function for each pair of state and action. Classic value learning methods, including deep Q learning (DQN) suffer from the value overestimation problem. In this regard, advanced methods like double and dueling DQN (DDQN) and distributional RL are proposed to solve that limitation through decoupling action selection from evaluation and calculating the complete distribution of Q-values [4], [11], [12] which come with higher computational costs. In contrast, action selection policy is directly developed in policy-gradient methods by making high-reward actions more likely using gradient ascent (*e.g.*, A2C, DDPG, and TD3) [6], [13]. Controlling the difference between the new and old policies, trust region policy optimization (TRPO) enables a smooth performance improvement [6] despite incurring high computation costs. To reduce the required computations, proximal policy optimization (PPO) follows a simpler technique in estimating the difference between policies [6], [14].

On-policy vs. Off-policy. In on-policy RL algorithms (*e.g.*, TRPO, PPO, and A2C), the policy that is under development is also utilized to generate new samples. However, off-policy algorithms (*e.g.*, DQN, DDPG, TD3) maintain separate policies for development and sample generation. Such algorithms keep a buffer of the agent's experiences collected at any time and update the under-development policy after a few interactions with the environment [14]. Although off-policy algorithms enjoy a higher sample-efficiency (thus faster convergence), on-policy algorithms together with the actor-critic methods provide higher stability during training with monotonic policy improvement. Besides, off-policy algorithms require extensive hyperparameter search which hurts stability and generalizability [14].

Single-agent vs. Multi-agent. Single-agent RL approaches require too long training in scenarios with high action and state dimensions. One way to cope with this problem is to

Authorized licensed use limited to: University of Waterloo. Downloaded on February 14,2023 at 17:31:44 UTC from IEEE Xplore. Restrictions apply.

decompose it into multiple sub-problems and formulate it as a multi-agent RL [13]. Agents in a multi-agent RL approach can operate with or without coordination among them. Coordination among the agents to better estimate the global state and other agents' policies improves the overall performance, although it incurs communication overhead [13].

Model-free vs. Model-based. Model-free RL approaches (*e.g.*, A2C, PPO, and DQN) are most useful when the agents are trained before real-world deployments, as they require a large number of samples for learning. These long training periods, which involve inefficient policies, can lead to frequent SLA violations and/or excessive resource over-provisioning in RAN slicing [6]. On the contrary, model-based RL approaches [6] overcome such limitations, allowing for greater sample efficiency and accelerated learning. However, model-based RL approaches suffer from their limited capacity in learning compared to model-free RL methods.

TABLE II
ADVANTAGES AND DISADVANTAGES OF RL TECHNIQUES IN THE LITERATURE FOR SINGLE-CELL RAN SLICING

RL approach	Advantages and disadvantages
value-learning [4], [11], [12] vs. policy-gradient [6], [13]	Policy-gradient methods offer a stable behavior improvement as they limit policy modification in each training step, so they are more suitable for online learning.
on-policy [6], [13], [14] vs. off-policy [4], [11], [12]	Off-policy approaches provide faster convergence, but they require extensive hyperparameter tuning which aggravates generalizability of such methods.
single-agent [4], [11], [14] vs. multi-agent [10], [12], [13]	Single-agent RL algorithms fall short when the size of action-state spaces increases in case of numerous slices while decomposing the problem using multi-agent RL addresses the high-dimensional problems.
model-free [5], [7], [8] vs. model-based [6]	Model-free RL algorithms provide a higher sample efficiency although they are limited in their learning and generalization capacities.

The presented decision dimensions in RAN slicing suggest the complications of deciding about the RL algorithm. A summary of the advantages and disadvantages of different RL methods is presented in Table II.

B. Multi-cell RAN Slicing RL Formulation

As opposed to single-cell RAN slicing, different problem formulations with distinct objectives have been pursued in the multi-cell RAN slicing literature. Particularly, they consider jointly deciding users' serving BS, slice, and bandwidth on the corresponding BS [11], selecting per slice bandwidth, scheduling algorithm, and modulation and coding offset [8], [14], compensating resources across BSs [12], and accounting for the effect of inter-cell interference [13].

Due to this heterogeneity in objectives, we refuse to compare their MDP components and RL algorithms as we did for the case of single-cell. Instead, we carefully investigate their method, highlighting the advantages and disadvantages. Nonetheless, we summarize the different components of multi-cell RAN slicing in Table III for survey completeness.

To efficiently manage the access of different devices to a pair of (BS, slice) while considering security and privacy concerns, authors in [11] resort to federated RL. Accordingly, they train local models on devices with two-layer aggregations: samples of the same service types and then different service types. Nonetheless, this approach comes with the following shortcomings: (i) although the security and privacy of users are guaranteed in this scheme, it reveals the data of slices' available bandwidth on each BS which raises security concerns for mobile networks; (ii) the association of each device with a pair of (BS, slice) is decided separately rather than considering a holistic view of devices, so it is pruned to suboptimality.

Opening the door of resource compensation across BSs, authors in [12] devise a means of PRB redistribution in the lower level of their two-level resource allocation scheme between the consecutive high-level RAN slicing decisions. To deal with the high dimensionality of the problem, they resort to distributed multi-agent RL techniques in such a way that each BS is considered as an agent. They further propose to conduct the training procedure in an offline manner from a collected dataset due to the required heavy computations in online learning. Nonetheless, they do not provide any insights on how this offline learning generalizes to real-world deployments. Additionally, different agents are trained independently, which raises suboptimality concerns as they do not share their experiences and consider each other's demands.

The authors in [13] highlight the effect of inter-BS interference on overall network performance and SLA satisfaction. To take it into account, they exploit coordinated multi-agent methods in a way that each agent is associated with a specific BS. Assuming that load-coupling inter-BS interference is the main cause of the inter-agent dependencies, they let each agent communicate its per slice load information with its neighboring agents which will be fed to the RL agent as part of the state. In comparison to a centralized approach deciding RAN slicing on all BSs using a single agent, the evaluations reveal that the proposed coordinated multi-agent approach converges faster and gives better performance.

IV. CHALLENGES IN EFFECTIVE USE OF RL IN RAN SLICING

Although exploiting RL techniques can potentially improve service delivery and resource utilization, certain challenges might hinder the expected performance. Particularly, a RAN slicing RL agent can take random actions during training, leading to slice SLA violation or resource over-provisioning. Moreover, MNOs should be able to accommodate new slices upon request, but most of the proposed RL-based RAN slicing problem formulations cannot provide such a capability. In this section, we present different challenges to be considered in devising an RL-based RAN slicing and argue how proposals in the literature failed or succeeded in dealing with them as summarized in Table IV.

A. *Constraint-awareness*

One main issue in real-world deployments of RL methods is the possibility of taking random actions during online training.

Authorized licensed use limited to: University of Waterloo. Downloaded on February 14,2023 at 17:31:44 UTC from IEEE Xplore. Restrictions apply.

TABLE III
SUMMARY OF STATES, ACTIONS, REWARDS, AND RL ALGORITHMS USED IN THE LITERATURE FOR MULTI-CELL RAN SLICING

	State	Action	Reward	RL Algorithm
[8]	average slices traffic and users' channel quality, PRB usage, SSR, and SLA thresholds	per slice number of PRBs, MCS offset, scheduling algorithm	negative total resource usage, a separate cost value capturing performance degradation	multi-agent PPO (each agent representing a slice) along with an action modifier across slices
[11]	current serving BS and slice, allocated PRBs to slices on each BS	joint selection of per user BS, slice, and number of PRBs	bandwidth efficiency minus signaling overhead	DDQN
[12]	**higher-level**: previously allocated PRBs to each BS, sets of BSs and PRBs; **lower-level**: channel gain between each BS and its users, minimum data rate and maximum delay, users with a BS and its PRBs	**higher-level**: distributing PRBs to BSs; **lower-level**: assigning BS PRBs to its users and request additional PRBs from other BSs when needed	**higher-level**: sum of achieved users' data rate; **lower-level**: total sum-rate subject to ultra-low latency requirements of uRLLC services and minimum data rate requirements of eMBB services	**higher-level**: multi-armed bandit; **lower-level**: distributed multi-agent DDQN without coordination (each agent representing a BS)
[13]	average per slice user throughput, load, number of active users, and neighbouring load	per slice number of PRBs on each cell	SSR	distributed multi-agent TD3 with coordination (each agent representing a BS) following actor-critic
[14]	per slice average rate and buffer size, and assigned PRBs	per slice number of PRBs, scheduling algorithm	eMBB slice rate, mMTC slice transfer block size, and uRLLC slice negative buffer size	PPO

Especially in the case of RAN slicing, where the network is supposed to accommodate different slices' SLAs, RL agents' exploration can lead to not respecting these demands. In particular, authors in [8] demonstrated that an RL agent could have more than 30% violations of the slices' SLA during the online learning phase. Therefore, appropriate strategies should be adopted to mitigate this issue.

Two strategies with different effective time spans are pursued to take SLA requirements into account. One of them motivates respecting constraints in the long run by incorporating them into reward while the other approach guarantees that each action does not immediately violate the constraints.

To realize the first strategy, a notion of slices' SSR is included in reward utilizing a log-barrier function [7] or Lagrangian primal-dual method [8] to adaptively encourage the solver to respect the constraints. Moreover, following a model-based RL method, a predictor is quickly trained in [6] to estimate whether each action for a given state leads to SLA satisfaction.

A safer learning process is provided when SLA constraints are taken into account using specialized functions rather than negative values in reward design. The agent in the latter scenario is unaware of the trade-offs involved between resource violations and SLA satisfaction, whereas we make the agent aware in the former case. Model-based methods are limited in their learning capabilities — they only consider a limited set of scenarios and fail to adapt to the varying network conditions.

Although the first strategy is effective in the long run, constraints should not be violated frequently even in a short time span. In this regard, the action generated by the policy network is projected to a feasible space in which the accumulated constraints remain below a certain threshold in [7]. Following the same objective, authors in [8] design a proactive policy switching mechanism to switch to a baseline policy for managing resources if the RL policy is predicted to violate the slice SLA.

SLAs cannot be effectively met by the proposed methods for two reasons; it is not argued [7] how a feasible action space should be appropriately estimated in the early stages of learning when the agent does not know the impact of each action. Additionally, modifying RL agents' actions can jeopardize the learning process in the long run as it increases the correlation between the agent's interactions with the environment and limits its space exploration [7] [8].

B. *Generalizability*

The rationale behind offline training is that it is inefficient to allow an RL agent to learn online from scratch within real networks. This is because the agent usually requires a large number of training steps during which the it is prone to take random actions. To mitigate this issue, agents can learn offline to imitate a baseline policy based on the dataset collected from the interactions between that policy and real networks. In particular, behavior cloning can be leveraged to train a policy to minimize the differences between generated actions by the under-train policy and the baseline policy with supervised learning [8].

Nevertheless, we cannot rely on offline training due to the specific characteristics of a deployment scenario that is not part of the training dataset. In online training, the RL agent uses live data from the RAN and performs exploration steps on the online RAN infrastructure. Therefore, the agent will adapt itself to the deployment-specific features, leading to improved generalizability. In this regard, the results in [14] confirm that online training can help pre-trained models evolve and meet the demands of the specific environment in which they are deployed; however, at the cost of reduced RAN efficiency during the online training phase. However, the methods provided in Section IV-A can help reduce such costs during online training.

Transfer learning is another approach to convergence acceleration of RL algorithms by which the RL agent at an expert BS learns a policy from scratch until convergence, while the RL agent at a learner BS reuses the expert BS's learned policy by initializing the target model with the architecture and weights from the trained models [3]. Nonetheless, the proposed method in [3] is limited to the cases where the configurations of the RL agent do not change from the expert BS to the learner

Authorized licensed use limited to: University of Waterloo. Downloaded on February 14,2023 at 17:31:44 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
SUMMARY OF RAN SLICING CHALLENGES ADDRESSED IN THE LITERATURE.

Challenge	Description	Suggested solution approach	References
Constraint-awareness	slice SLAs impose QoS requirements that should be satisfied over both short and long time spans, but they might be violated during the training stage	guaranteeing SLAs using constrained-RL approaches that offer safe learning	[6]–[8]
Generalizability	training RL algorithms from scratch in production networks results in frequent SLA violations, necessitating training in a generalizable manner before deployment	Offline learning methods such as imitation learning and behaviour cloning based on the interaction trace of a baseline policy with the production network	[3], [8], [14]
Flexibility in number of slices	Tenants request slices over time, which requires algorithms that support a dynamic number of slices	Considering a separate agent for each slice with knowledge reuse across slices of the same slice type	[9], [10]
Scalability in number of slices	when the number of slices grows, the algorithms should not be negatively affected	Space reduction methods to avoid too large state and action sizes which deteriorate the agent's performance	[9], [15]
Robustness	RL algorithms rely on measurements from the underlying infrastructure that can be noisy. Thus, they should tolerate noises in such measurements	Extracting significant features of the state space	[4], [14]

BS, so it does not generalize in this sense. Specifically, the effectiveness of the method has only been tested in simulation, but not yet in real-world deployments.

C. Flexibility in Number of Slices

MNOs should not be limited in the number of slices they can offer, as tenants may demand new customized services anytime. In addition, the temporary need for a slice suggests supporting the capability of enabling or disabling slices on the fly. Therefore, an effective RAN slicing management scheme should be flexible in accepting and removing slices on demand.

Inflexibility in the number of slices comes from the fact that only a fixed number of slices is considered in the corresponding MDP, and then when the model is trained based on that, it will not be possible to add a new slice to the system unless the model is retrained. In particular, statistics of a fixed number of slices are gathered as the state and actions are taken also with respect to that fixed number of slices [3]–[5], [7].

To provide flexibility, an agent is responsible for allocating the minimum required radio resource blocks to a slice, and the agent is replicated or removed as the number of slices fluctuates in [9]. As different slices' actions might conflict, a network slice controller is responsible for coordination between agents. It is worth mentioning that because the actors (agents) follow the policy trained by the learner, they all have the same policy, enabling the fluctuation of the number of agents during execution. Following the same policy would not have been possible in their proposition unless the state, reward, and action of different actors share the same structure despite the difference between their requirements.

Specifically, they only consider one type of service with different threshold levels as different slices. Nonetheless, slices come with different KPI requirements and different impacting factors, so a single slice type cannot accommodate the heterogeneous set of services in 5G. Furthermore, a heuristic algorithm is in charge of enforcing resource capacity constraints which raises suboptimality concerns.

In contrast, a game theoretic approach can be followed where specialized agents are designed for each slice that maximizes their rewards, and the correlated equilibrium method balances the reward of these agents to increase the overall reward of the system [10]. Similar to the previous method, each agent competes for more PRBs to achieve higher rewards which in turn causes resource allocation conflicts in this multi-agent system. To resolve this issue, a game theory-based method such as correlated equilibrium is exploited to manage resource distribution between these agents. This method, however, does not explicitly investigate flexibility in the number of slices, but it is potentially suitable for enabling flexibility as it considers separate entities per slice.

D. Scalability in Number of Slices

As mentioned in the previous section, any limitation on the number of slices is not tolerable for MNOs. Particularly, when the number of slices increases unprecedentedly, the space size of the state and action of a single agent can become overwhelmingly large, degrading the RL-based algorithm's decision time and efficiency.

To address the enlargement problem of state-action size in the case of many slices, each agent can be associated with a separate agent [9]. Accordingly, the number of agents is increased rather than the size of action-state space in the case of adding more slices. This strategy encourages knowledge reuse across agents, which reduces the complexity of training multiple agents. However, it fails in supporting multiple policies.

Additionally, action space reduction methods are proven to be effective in accelerating convergence and accommodating scalability. For example, the action reducer technique proposed in [15] is essentially a conventional optimization framework that reduces the action size and then converts it back to the original action space within a polynomial time complexity.

E. Robustness

RL algorithms depend highly on the measurements gathered from the underlying network, notably the state and reward variables. Such data is prone to noise and redundancy because of the inherent measurement errors. Therefore, appropriate techniques should be exploited to make the RL methods robust to such noises.

Authorized licensed use limited to: University of Waterloo. Downloaded on February 14,2023 at 17:31:44 UTC from IEEE Xplore. Restrictions apply.

In this regard, a combination of distributional RL and GAN can be leveraged to compute the distribution of action value instead of its expected value as in regular DQN. The distributional method is known to be robust against noises in the environment [4].

To further robustify the RL methods, vital information should be extracted from the measurements using compression and feature selection techniques. Indeed, RAN produces a massive amount of data that does not necessarily provide meaningful insights into the actual state of the system due to redundancy. To deliver a high-quality representation of the state, auto-encoders can be utilized before feeding the selected features of the data to the RL agent [14].

Computing the complete distribution of Q-values is computationally expensive and cannot scale with the size of the state and action spaces. Moreover, the general idea of value distribution calculation is only applicable to value learning RL methods but not policy-gradient algorithms which are interesting in RAN slicing because of their stable training behavior. Therefore, the second approach [14] where the state and action are compressed to extract the vital information is more suitable.

V. Conclusion and Future Directions

Many attempts have been made to solve the RAN slicing problem using RL. In this article, we present a thorough investigation of how the corresponding MDP is formulated and RL techniques are leveraged to solve the RAN slicing problem in the literature. Furthermore, we provide the taxonomy on single-cell and multi-cell RAN slicing scenarios and discuss the underlying trade-off between them.

Additionally, we identify specific challenges in proposing an effective RL-based RAN slicing method and categorize proposed solutions based on how they fail or succeed in solving those challenges. This study suggests that different papers address disjoint subsets of challenges, and there exists a shortage of investigating the possible trade-offs in jointly addressing these challenges and a full-fledged solution that solves them altogether.

References

[1] X. Shen, J. Gao, W. Wu, K. Lyu, M. Li, W. Zhuang, X. Li, and J. Rao, "AI-assisted network-slicing based next-generation wireless networks," *IEEE Open Journal of Vehicular Technology*, vol. 1, pp. 45–66, 2020.

[2] Y. Azimi, S. Yousefi, H. Kalbkhani, and T. Kunz, "Applications of machine learning in resource management for RAN-Slicing in 5G and beyond networks: A survey," *IEEE Access*, vol. 10, pp. 106 581–106 612, 2022.

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

[4] 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.

[5] 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.

[6] J. J. Alcaraz, F. Losilla, A. Zanella, and M. Zorzi, "Model-based reinforcement learning with kernels for resource allocation in RAN slices," *IEEE Transactions on Wireless Communications*, 2022.

[7] Y. Liu, J. Ding, and X. Liu, "A constrained reinforcement learning based approach for network slicing," in *IEEE 28th International Conference on Network Protocols (ICNP)*, 2020, pp. 1–6.

[8] 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*, 2021, pp. 141–153.

[9] Y. Abiko, T. Saito, D. Ikeda, K. Ohta, T. Mizuno, and H. Mineno, "Flexible resource block allocation to multiple slices for radio access network slicing using deep reinforcement learning," *IEEE Access*, vol. 8, pp. 68 183–68 198, 2020.

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

[11] Y.-J. Liu, G. Feng, J. Wang, Y. Sun, and S. Qin, "Access control for RAN slicing based on federated deep reinforcement learning," in *IEEE International Conference on Communications (ICC)*, 2021, pp. 1–6.

[12] A. Filali, Z. Mlika, S. Cherkaoui, and A. Kobbane, "Dynamic SDN-Based radio access network slicing with deep reinforcement learning for urllc and embb services," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 4, pp. 2174–2187, 2022.

[13] T. Hu, Q. Liao, Q. Liu, D. Wellington, and G. Carle, "Inter-cell slicing resource partitioning via coordinated multi-agent deep reinforcement learning," *arXiv preprint arXiv:2202.12833*, 2022.

[14] 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, 2022.

[15] 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.

Biographies

Mohammad Zangooei is a Ph.D. student at the David R. Cheriton School of Computer Science at the University of Waterloo. He received his Bachelor's degree in Electrical Engineering from the Sharif University of Technology, Tehran, Iran. His research interests revolve around next-generation mobile networks, artificial intelligence, and programmable data planes.

Niloy Saha is a PhD student at the David R. Cheriton School of Computer Science at the University of Waterloo. He received his Masters degree in computer science from the Indian Institute of Technology, Kharagpur, India. His research interests are focused on building next-generation mobile networks and intelligent algorithms for their orchestration and management.

Morteza Golkarifard received his B.Sc., M.Sc., and Ph.D. degrees in computer engineering from the Sharif University of Technology. He is currently a post-doctoral fellow at the David R. Cheriton School of Computer Science at the University of Waterloo. His research interests include 5G networks, NFV, and SDN.

Raouf Boutaba received his M.Sc. and Ph.D. degrees in computer science from Sorbonne University in 1990 and 1994, respectively. He is currently a University Chair Professor and the director of the School of Computer Science at the University of Waterloo. He is the founding Editor-in-Chief of IEEE Transactions on Network and Service Management and a Fellow of the Engineering Institute of Canada, the Canadian Academy of Engineering, and the Royal Society of Canada.

Authorized licensed use limited to: University of Waterloo. Downloaded on February 14,2023 at 17:31:44 UTC from IEEE Xplore. Restrictions apply.