Industrial Engineering Journal
ISSN: 0970-2555
Volume : 51, Issue 04, April : 2022

# Problems and Solutions in Deep Reinforcement Learning for End-to-End Network Slicing

*Mr.Sakti Charan Panda[1]*, Mr.Bikash Ranjan Das[2]*
*[1]*Assistant Professor,Dept. Of Computer Science and Engineering, NIT , BBSR*
*[2]Assistant Professor,Dept. Of Computer Science and Engineering, NIT , BBSR*
*sakticharan@thenalanda.com*, bikashranjan@thenalanda.com*

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 51, Issue 04, April : 2022

*Abstract*—Vertical industry use cases with a range of performance requirements are anticipated to be enabled by 5G and beyond. Network slicing is crucial in dynamically generating virtual end-to-end networks in response to particular resource demands in order to serve these use cases cost-effectively. Traditional model-based solutions cannot be used to orchestrate resources for network slices since a network slice may contain hundreds of customizable characteristics spanning various technical domains that define the performance of the network slice. This article describes a model-free method for solving the network slicing problem called deep reinforcement learning (DRL). The network slicing issue is first examined, and a standard-compliant system architecture is then shown, allowing DRL-based solutions in 5G and future networks. Second, we offer a thorough examination of the difficulties in developing and implementing DRL in network slicing systems. Third, we investigate several interesting methods for automating end-to-end network slicing, including safety and distributed DRL and imitation learning.

## I. Introduction

5G and beyond will be a catalyst to digitalize the econ- omy and contribute toward global digital transformation. The explosion of networking connections and mobile data will dramatically increase the complexity of the network. An increasing number of new use cases will be enabled for various industries such as automotive, manufacturing, logistics, and energy [1]. These new use cases have highly diverse, and even conflicting, communication and networking requirements such as latency, data rates, availability, and reliability. The growing network complexity and service diversity challenge network operators to dynamically orchestrate and coordinate network resources to offer a different mix of capacities for supporting services with diverse requirements simultaneously.

Since it is not economically viable to build a dedicated network for each type of service, network slicing emerges as a key technology in 5G wireless networks for efficiently supporting highly diverse use cases [1]. Network slicing allows network operators to run multiple logical networks (aka. network slices) on top of common physical network infras- tructures. For each network slice, customized functionalities and network operations can be implemented according to the specific need of an individual use case, e.g., enhanced mobile broadband (eMBB), ultra reliable low latency communications (URLLC) and massive machine type communication (mMTC). For example, a network slice can be customized to support IoT services for a large number of devices operated at low data

Qiang Liu is with the School of Computing, University of Nebraska-Lincoln. E-mail: qiang.liu@unl.edu
Nakjung Choi is with Nokia Bell Labs. E-mail: nakjung.choi@nokia-bell-labs.com
Tao Han is with the Department of Electrical and Computer Engineering,

Fig. 1: An illustration of end-to-end network slicing.

rates. At the same time, another network slice can be tailored to provision latency-critical services such as vehicle-to-vehicle communications and smart grid controls.

Network slices usually span across multiple technical do- mains of the network, as shown in Fig. 1, e.g., radio access networks, transportation networks, core networks, and edge and cloud computing. Hence, end-to-end resource orchestra- tion is essential in network slicing to dynamically manage resource allocations to different slices in multiple domains for achieving optimized performances. A network slice may have hundreds of configuration parameters defining its functions and performance according to the service level agreement (SLA). These configurations are connected to the underlying resource demands in different domains. Thus, it is impractical to develop a closed-form model of the network performance versus the joint resource allocations in multiple domains.

Recent advances in ML, especially deep learning (DL) and deep reinforcement learning (DRL), have demonstrated great potential to learn and understand complex and high- dimensional correlations by leveraging artificial neural net- work (ANN) architectures. ML as a model-free approach requires no prior knowledge in advance, which can automati- cally learn the complex and unknown correlations in network slicing. Thus, there is increasing popularity of exploring ML to automate network slicing [2], [3], [4], e.g., admission control and resource management. Recent works reveal that the resource orchestration in network slicing has Markov property [5], i.e., the orchestration decision at the current time influences not only the current performance of slices but also the future network states, e.g., queuing. As a result, the resource orchestration

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 51, Issue 04, April : 2022

turns out to be a Markov decision process (MDP), which cannot be resolved with DL techniques.Hence, DRL becomes the most suitable approach to deal with the MDP and intelligently manage the resource orchestration in network slicing.

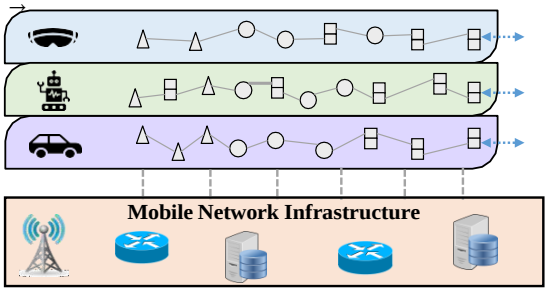
Following these initiatives, this article dives deep into DRL-based network slicing solutions. First, we introduce the end-to-end resource orchestration problem in network slicing and show that the problem can be naturally formulated as a Markov decision process (MDP). To deploy DRL-based network slicing, a new end-to-end network system architecture is engineered with the network layer, orchestration layer and intelligence layer. Second, we discuss the challenges of design-ing DRL-based network slicing solutions from the perspectives of both algorithms and system. Third, we explore and envision multiple promising techniques, i.e., safety and distributed DRL, and transfer learning, to address these challenges in automating the end-to-end network slicing.

## II. DRL FOR NETWORK SLICING

A fundamental research problem in network slicing is the design of end-to-end resource orchestration that jointly orchestrates resources in multiple domains, e.g., radio access networks and edge computing, based on service requirements of network slices. In this section, we analyze the end-to- end resource orchestration problem and present the system architecture for DRL-based network slicing.

### *A. Resource Orchestration Problem*

The rationale behind using DRL is that the end-to-end re-source orchestration problem can be reasonably formulated as a Markov decision process (MDP) when considering the tem- poral dependency of the resource orchestration and network performance [5]. A MDP can be denoted by $S, A, r, P, \mu$ ,where $\mathbf{s}^t$ $S$ is defined as the ***state*** that describes the networkstatus and traffic load distributions in the current time slot $t$. $\mathbf{a}^t$ $A$ is defined as ***action*** that describes how resources are orchestrated on every edge node, e.g., radio base stations and edge servers, for every network slices in time slot $t$. $r : S \quad A \quad R$ is defined as the ***reward function*** that models the performance of the network system (e.g., resource usage). $c : S \quad A \quad C$ is defined as the ***cost function*** that models the constraints of the network system (e.g., SLA requirement). $P : S \quad A \quad S \quad R$ is defined as the ***state transition func- tion*** describes the underlying temporal dynamics of the state depending on the current action and ***initial state distribution*** $\mu : S \quad [0, 1]$. The resource orchestration action is denoted by $\mathbf{a}^t = a^t$ where $a^t$ is the amount of type-$m$ resource allocated to slice $k$ in time slot $t$. A resource orchestration policy $\pi : S \rightarrow Pr(A)$ prescribes a probability distribution



Industrial Engineering Journal
ISSN: 0970-2555
Volume : 51, Issue 04, April : 2022

over actions given the current state. Given policy $\pi$, the end- to-end networking system receives a sequence of rewards and costs (i.e., network resource usage and system performance)$r^1, r^2, \ldots$ and $c^1, c^2,$ The end-to-end orchestration problem is to find the policy that maximizes the long-term system reward, i.e., $\arg\max_\pi \mathrm{E}^{\pi_\theta}[\sum_{t=0}^{\infty} \gamma^t r^t(\mathbf{s}^t, \mathbf{a}^t),\ s.t.\ c^t(\mathbf{s}^t, \mathbf{a}^t) \leq 0]$ where $\gamma$ is the discount factor that balances immediate and future rewards.

### B. DRL-based Network Slicing System Architecture

To facilitate DRL-based end-to-end network slicing, we design a new and standard-compliant system architecture with three functional layers as shown in Fig. 2. The network layer consists of network infrastructure devices, e.g., eNB and gNB, switches/routers, network functions, and edge and cloud servers. The orchestration layer provides end-to-end resource management functions that allocate resources to network slices to ensure their performances according to SLAs. Domain managers are deployed in each technical domain, e.g., access, edge, transport, and core networks, with the action and state APIs to execute the orchestration. The domain managers realize the network slice subnet management functions (NSSMF) suggested by 3GPP [7]. The end-to-end orchestration and service assurance modules, which are similar to the 3GPP network slice management function (NSMF), aim to coordinate domain managers to ensure the end-to-end system performance of network slices [7]. These modules are designed with interfaces that allow DRL agents to receive orchestration actions and feedback the slice performance statistics. The intelligence layer implements DRL agents that learn the optimal resource orchestration policy for end-to-end network slicing. Since the machine learning components, i.e., DRL agents, are decoupled from orchestration functions, various machine learning techniques, e.g., safety learning, distributed learning and imitation learning, can be deployed without changing the underlying network slice functions in the orchestration layer.

## III. DRL Challenges

Although many existing works focus on applying DRL to solve various networking problems, deploying DRL-based network slicing solutions in practical end-to-end networks still faces multiple research challenges in terms of the performance assurance, solution scalability, and convergence speed of DRL.

### A. Performance Assurance

Network operators aim to satisfy service level agreements (SLAs) of slices with the minimum usage of cross-domain resources and thus reduce the operating expenses (OPEX). Realizing this goal is non-trivial because of the following research challenges.

*1) DRL explainability:* To provide performance assurance, it is important to understand why a DRL-based slicing solution can lead to certain network performance. The difficulties come from two aspects. First, it is non-trivial to interpret the impact of the state space of DRL on the optimal policy. The state space should provide a comprehensive representation of

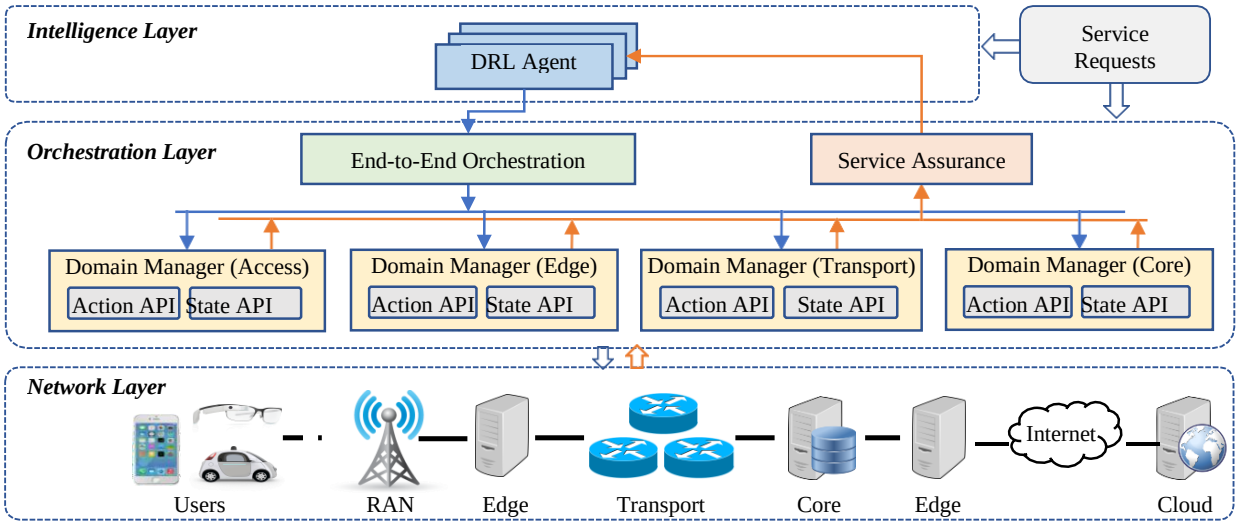



Fig. 2: DRL-powered end-to-end network slicing system architecture.

network status to DRL agents. End-to-end networking systems generate a huge amount of measurement data from different domains. There still has no clear path to follow and build the state space based on these network measurements in order to achieve the optimal network slicing policy.

Second, the reason why the *actions* generated by the DRL agent can achieve the optimal performance in long term is difficult to interpret. For a DRL policy, *actions* are calculated through mathematical operations, e.g., addition, multiplex and activation in hidden layers and the output layer of a neural network. Although the mathematical calculations are known, analyzing them can only provide very limited understandings about why an action is generated instead of the others, especially considering deep neural networks with heterogeneous layers such as convolutional and dropout. Without such interpretable knowledge, it's hard for network operators to directly control or modify *actions* generated by DRL agents.

*2) Exploration in policy optimization:* DRL usually relies on random exploration mechanisms to find a better policy for improving long-term rewards. A practical useful exploration mechanism explores a nearby space of *action* $\mathbf{a}_t$, e.g., $\mathbf{a}_t + \epsilon$ where $\epsilon$ is sampled from a normal distribution with zero mean and a predefined deviation. As the new action deviates from the original action $\mathbf{a}_t$, the performances of network slices, e.g., throughput and delay, may be degraded to the extent that SLAs are violated. It is straightforward to limit the magnitude of the action exploration, e.g., $\mathbf{a}_t + clip(\epsilon, -H, H)$, where $H$ is a given maximum allowed deviation. However, this method reduces the exploration efficiency as the exploration is clipped. As a result, it may lead to a suboptimal policy or require more interactions learn the optimal policy in practice.

*3) Quantization error and parameterization:* DRL agents update their neural parameters by using gradient descent methods, e.g., SGD, Adam and Momentum [8], based on the collected historical trajectories (state-action-reward pairs). Although the step size of gradient descent methods are usually small, e.g., 1E-3 or 1E-4, the performance of the DRL agent after one single policy update can be significantly different. The reason is that DRL policies are often parameterized by millions of neurons, a slight change in each neuron can result in a dramatic discrepancy in output actions. Hence, when learning the optimal policy via interacting with a real end-to-end network, DRL agents may show changing performances during the learning phase. As a result, it is challenging to assure a predictable performance of network slices using DRL-based slicing solutions.

### *B. Solution Scalability*

Learning-based networking mechanisms are designed to solve complex networking problems in a large scale. Hence, the scalability of DRL-based network slicing solutions determines whether they can be deployed in practical networking systems with heterogeneous infrastructures, e.g., base stations and servers, and dynamic service demands. The challenges of scaling DRL-based solutions up-and-down are from two aspects: communication and computing overhead, and neural network design.

*1) Communications and computing overhead:* DRL agents need to collect network measurements from distributed infrastructures to train the neural network [9]. As the scale of the network increases, more measurement data need to be transmitted over the network, which may incur excessive communication overhead. According to OpenCelliD database, the number of base stations (e.g., GSM, UMTS, CDMA, LTE) of a medium city downtown (several square miles) could reach 1000. Consider each base station transmits hundreds of parameters every millisecond, the transmission demands of such area will reach up to gigabits per second. Meanwhile, managing a large-scale network also requires DRL agents to add more dimensions in their *state* and *action* spaces. As a result, the DRL agent consumes more computing resources and takes a longer time to learn the optimal policy.

*2) Neural network design:* DRL policies are parameterized by deep neural networks whose input and output dimensions are selected based on the *states* and *actions* [8], where the dimensions stay unchanged throughout the policy training. When the networking condition changes, e.g., new admitted network slices, both the *states* and *actions* space need to be

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 51, Issue 04, April : 2022

revised to reflect this change. As a result, the input and output dimensions of the neural network need to be updated, and the DRL policy has to be trained under the new networking condition. Although recently emerged recurrent neural networks (RNN) and graph neural networks (GNN) can handle the flexible sizes of inputs and outputs, their implementations in the context of DRL still need further investigations in terms of data efficiency, generalization and convergence.

### C. Convergence Speed of DRL

DRL agents learn the optimal policy by directly interacting with real networking systems. To obtain the optimal policy, a DRL agent usually needs to be trained by a large number of online interactions. Depending on how frequently the network measurements can be collected, it may take a very long time to achieve the convergence of the DRL policy. The large-scale 5G network normally has a long orchestration interval [10] as it involves the reconfiguration for various network devices and hardware. Considering 15 minutes as a practical reconfiguration interval, the feedback of an orchestration action generated by the DRL policy can only be obtained every 15 minutes. Since a single policy update usually requires more than one thousand transitions [8], it needs, at least, a few days to achieve one policy update.

## IV. System Challenges

Deploying DRL-based network slicing solutions faces several system level challenges such as preparing networking data and providing appropriate network programmability for DRL.

### A. Networking Data for DRL

DRL agents are trained with measurement data collected from heterogeneous and distributed network infrastructures. Preparing the data for DRL training can be a huge burden on networking systems.

*1) Data heterogeneity:* The openness evolution of 5G leads to the dis-aggregation of network components, which allows flexible network deployment strategies and increases the heterogeneity of these data in terms of formats, volume and time scales. For example, 3GPP suggests splitting the gNB function into a central unit (CU) and a distributed unit (DU). The DU focuses on signal and data processing at PHY and MAC layers, and CU manages controlling functions at higher layers such as PDCP and RRC. O-RAN [6] further split the gNB functions into CU, DU and radio unit (RU), where RU only provides functions related to radio frequency (RF) and Low-PHY processing such as fast Fourier transform (FFT), inverse FFT and physical random access channel (PRACH) extraction. To efficiently manage measurement data from various network functions, O-RAN introduces the key performance measurement (KPM) function that defines a collection of information elements such as 5G QoS Identifier (5QI), QoS Class Identifier (QCI), time stamp, and Cell Object ID. These information elements contain highly heterogeneous data. In addition, the service-based architecture of 5G core (5GC) implements new network functions such as access and mobility management function (AMF), session management function (SMF), anduser plane function (UPF), which further increases the amount and heterogeneity of network data.

*2) Data processing capability:* Since a large amount of heterogeneous network data have to be collected and analyzed for DRL-based network slicing solutions, additional computing devices and functions are necessary to meet such data processing demands. For example, the logging size of an operating gNB could easily exceed 1GB every minute [11], which is difficult to be transmitted from infrastructures to DRL agents. To reduce the data size of runtime data, the function of feature extraction, e.g., autoencoder, may be implemented to generate a concise representation. The feature extractor usually achieves a better compression rate and accuracy of representation if more computing resources are enforced, e.g., denser neural network architectures. As a result, there is a tradeoff between the deployed computation resources and the extraction performances, which requires further investigations.

### B. Network Programmability

To reconfigure network slices dynamically, network systems need to provide virtualization functions and programming interfaces to allow DRL agents to configure end-to-end network slices.

*1) End-to-end infrastructure virtualization:* The infrastructure virtualization is a key technology to provide necessary isolations among network slices, which assure that the performance and functions of a network slice are not affected by the operations of any other slices. The existing virtualization solutions are designed for individual technical domains, e.g., RAN, transportation networks, and edge and cloud computing. For example, FlexRAN virtualizes physical resource blocks (PRBs) in the MAC layer as virtual radio bandwidth in RAN [11]; OpenFlow allows the creation of virtual networks using software defined networking (SDN) in transport networks; virtual machine and docker container techniques enable the virtualization of computing resources in edge and cloud computing. It is, however, challenging to integrate virtualization solutions from different technical domains due to the heterogeneity of time scale, programming interfaces, and data models. For instance, the PRBs in RAN can be modified in milliseconds, while the scaling of docker containers in edge computing requires seconds to take effect.

*2) Network resource model:* The network resource model (NRM) is introduced to provide predefined interfaces based on either XML, JSON, or YANG to query and manage network resources [12]. The NRM defines various network resources that allow efficient network management. For example, *RRMPolicyRatio* defines *rRMPolicyMaxRatio*, *rRMPolicyMinRatio*

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 51, Issue 04, April : 2022

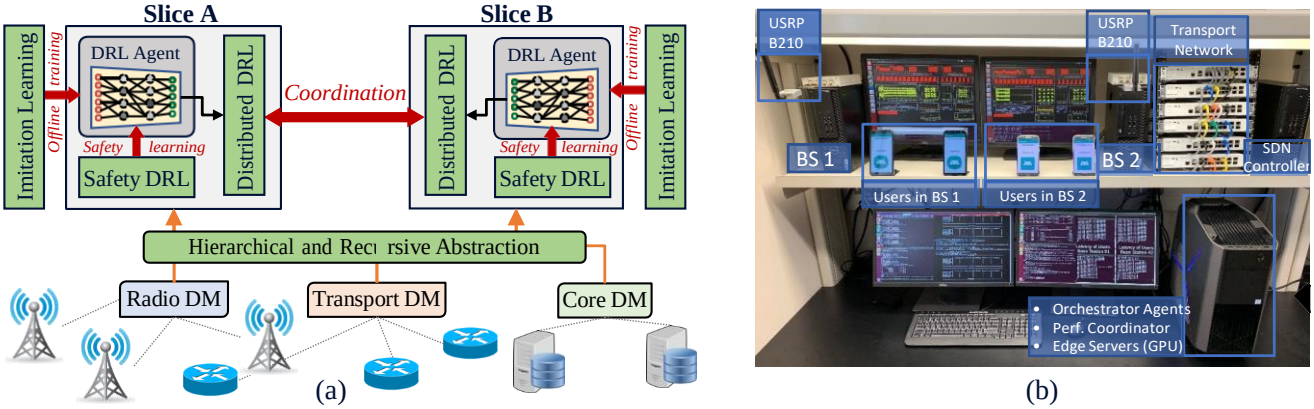



Fig. 3: a) the explored techniques for addressing challenges; b) the network slicing testbed [5].

performance of users and enable more customization features for slices are still open problems.

## V. DRL-BASED NETWORK SLICING SOLUTIONS

In this section, we discuss several promising techniques, as summarized in Table I, to address the aforementioned algorithm and system challenges. Fig. 3 (a) illustrates how these technologies can be deployed for end-to-end network slicing.

- The safety DRL addresses the challenge of the performance assurance by tackling the unpredictable exploitation and random exploration, and assuring performance requirements of slices throughout the learning phase.
- The distributed DRL addresses the scalability challenge by leveraging multi-agent DRL and allowing effective coordination among distributed agents.
- The imitation learning addresses the challenge of DRL convergence by offline imitating a baseline policy and obtaining a good start point for online learning.

We implement and evaluate these solutions using the testbed shown in Fig. 3 (b). The radio domain manager is designed based on OpenAirInterface (OAI) project with FlexRAN support [11] and use Ettus USRP B210 as the RF front-end of a base station. The transport domain manager is developed based on OpenDayLight (ODL) with OpenFlow 1.3 to control the transportation network composed of Ruckus ICX series SDN switches. We use OpenAir-CN project to deploy core network and docker container technique to enable edge computing virtualization. The DRL agents are developed by using PyTorch 1.5 with 3-layer fully-connected neural networks.

### *A. Safety DRL*

The safety DRL aims to train the DRL agent to maximize the cumulative rewards while maintaining different constraints. It is accomplished by integrating the Lagrangian primal dual method for statistical performance assurance and the baseline switching mechanism for avoiding instantaneous violations.

The reward shaping method, which re-weights the reward if constraints are violated, has been widely used to penalize the DRL agent and guide its training. However, it is difficult to choose the optimal weights especially under multiple constraints. For example, too small weights leads to insufficient penalization for the violations of constraints, while too large weights would suppress the exploration of DRL. To address this problem, recent advances adaptively incorporate these constraints into the reward function by using the Lagrangian primal dual method [13]. In particular, Lagrangian multipliers are introduced to re-weight the reward function, which are updated with the sub-gradient descent method in a larger time scale than that of resource orchestration. On learning the resource orchestration policy, the reward function of the DRL agent is rewritten by the Lagrangian function, which integrates both the original objective and the weighted constraints by using the updated Lagrangian multipliers. The constraints can be eventually satisfied by alternatively updating the multipliers and training the DRL agent.

The constraints, however, can still be violated before the Lagrangian primal dual method converges. As shown in Fig. 4 (a), a baseline switching mechanism is developed to enable the dynamic switching between the policy of the DRL agent and a baseline policy. We consider there is a baseline policy, e.g., rule-based or heuristic, which can guarantee the slice SLA but with higher resource usages [14]. The policy evaluation module is created to predict whether enforcing the current action generated by the DRL agent will break the slice SLA. In particular, we design the module to learn the value of constraints under different states and actions. This module can be implemented with DNN, where the input is the combination of state and action space, and the output is set to be the value of constraints, i.e., $c^t(\mathbf{s}^t, \mathbf{a}^t)$. If the prediction result is above zero with high confidence, the baseline policy will be invoked, which can maintain slice SLA at a cost of high resource usage. Fig. 4 (b) shows the cumulative probability of the SLA violation by two schemes, i.e., with and without the baseline switching mechanism. The SLA violation is defined if the constraints are violated, i.e., $c^t(\mathbf{s}^t, \mathbf{a}^t) \geq 0$. It can be seen that, without the baseline switching mechanism, the SLA violation

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 51, Issue 04, April : 2022

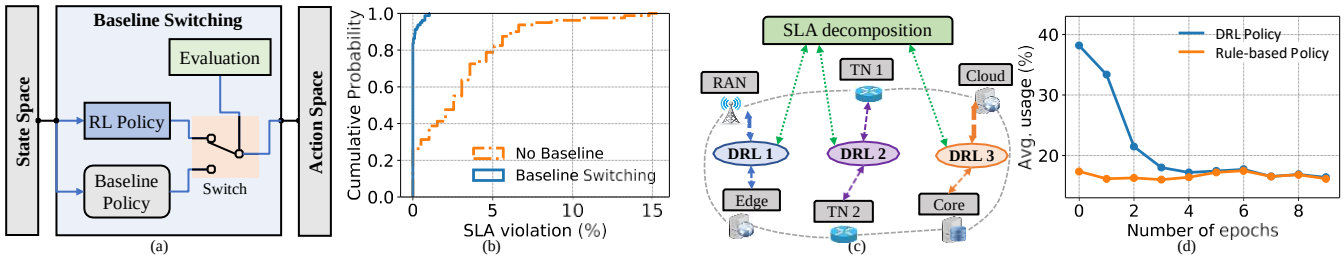



Fig. 4: a) overview of baseline switching mechanism; b) SLA violation with baseline switching mechanism; c) illustration of distributed coordination; d) resource usage with imitation learning;

| Challenges | Existing Work | Our Solution |
|---|---|---|
| Performance Assurance | unawareness of performance requirements, free action space exploration | **Safety DRL**: constraint-awareness update, baseline switching scheme |
| Solution Scalability | communication & computing overhead, fixed input/output DNN design | **Distributed DRL**: distributed multi-agents, coordination mechanism among agents |
| DRL Convergence | massive online interactions required, poor performance at early stage | **Imitation Learning**: offline imitate from baseline for online acceleration |

TABLE I: Summary of challenges and solutions

distributed networks, which creates multiple cooperative or competitive agents to achieve the global optima. For example, an individualized DRL agent can be created to orchestrate the resource for each network slice, where these agents are competitive due to the limitation of resource capacity. This individualized slice agent scheme helps to resolve the dynamics of slice admission and departure, where the action space varies. The training of distributed agents can be accomplished by centrally aggregating their experiences and updating the policy of all agents simultaneously.

To orchestrate the cross-domain resources for end-to-end slices, multiple individualized DRL agents can be created in distributed infrastructures. As illustrated in Fig. 4 (c), the DRL agent 1 orchestrates RAN and edge networks, the DRL agent 2 controls transport networks, and the DRL agent 3 manages core networks and cloud computing resources. These agents learn to allocate resources of infrastructures to slices independently, e.g., the DRL agent 2 allocates the bandwidth in transport networks. As these agents only focus on their technical domains, the end-to-end performance requirement of slices may not be guaranteed if no collaborations. Thus, we propose an SLA decomposition module to decompose the end-to-end performance requirements of slices into different technical domains, which allow the DRL agents aware of local requirements. For example, the end-to-end latency and reliability performance of a slice may be decomposed to local requirement in RAN, TN, CN, and Edge. Then, the DRL agents in each domain is focused to allocate their resources to meet the local performance requirement of this slice. In this scenario, the SLA decomposition module needs to balance the resource utilization in different domains, and is responsible for satisfying the end-to-end performance requirement.

### *B. Imitation Learning*

Imitation learning aims to train the DRL agent to mimic the behavior of a target agent, where the target agent canbe either human, model-based rules or another DRL agent.The main methods of imitation learning are behavior cloning,

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 51, Issue 04, April : 2022

direct policy learning via interactive demonstrator, and
*Offline DRL*

Offline DRL can be explored to offline train the DRL agents only based on collected online data sets. It helps to avoid expensive and unsafe online interactions with real networks. The main difficulty is the distributional shift issue, i.e., although its function approximators (e.g., policy and value networks) is trained under one distribution (i.e., limited data set), it will actually perform on a different distribution (i.e., the real network).

### A. Transfer Learning

Transfer learning has shown a great potential to address challenges regarding scalability, model reproducibility, and sample efficiency [15]. The basic idea is to exploit prior learned knowledge to benefit the learning process in target tasks. For example, several works have been done to transfer resource allocation policies from one network to another with similar settings, which accelerates the convergence speed of DRL agents [15]. Hence, the exploration of leveraging transfer learning in dealing with DRL challenges needs further research.

### B. Explainable AI

The explainable artificial intelligence (XAI) aims to understand, interpret and explain the black-box DNN-based policies via different approaches, e.g., visual explanation, local explanations, illustrative examples, and simplifications. Various techniques, e.g., linear regression, decision trees, K-nearest neighbors, and Bayesian models, can be exploited to improve the explainability of DNN-based policies. For example, given a state observation, the Q-value of each possible action can be obtained from the Q network in the Deep Q-Network (DQN) agent. To maximize the long-term cumulative reward, the action with the highest Q-value is usually selected by the DQN agent.

inverse reinforcement learning. Consider the network operator has a baseline policy, e.g., model-based rules, the imitation learning can train the DRL agent to mimic its behaviors, e.g., mapping the observed states to actions. The advantage of offline imita- tion learning is that the transitions of the baseline policy are not expensive to obtain. In Fig. 4 (d), we offline train the DRL agent to imitate the behavior of the baseline policy by using the behavior cloning method. Specifically, the offline training is accomplished by minimizing the action differences between the DRL policy and the baseline policy under different states. We see that the DRL agent uses a similar resource usage as the baseline policy does after several offline training epochs. The DRL agent, after the imitation learning, serves as the start point for the online learning phase, which then continues learning and improving.

## VI. Future Directions

We envision several promising techniques that may help to alleviate and resolve the aforementioned system and DRL challenges for end-to-end network slicing.

### C. Hierarchical and Recursive Abstraction

The hierarchical and recursive abstraction (HRA) aims to adaptively abstract and manage resources in different domains both vertically and horizontally. In the vertical direction, domain managers (DMs) in the same technical domain are hierarchically abstracted to provide high-level programmability. For example, the coverage in RAN can be accomplished by abstracting radio resources in multiple base stations in the geographic area. In the horizontal direction, DMs in different technical domains are recursively abstracted to provide end-to-end programmability. For example, the end-to-end data rate can be achieved by abstracting virtual resources in both radio, transport and core DMs. These DMs are responsible for enabling the network programmability of infrastructures, collecting the network state with standardized interfaces, and enforcing the management actions made by the DRL agents.

## VI Summary

This article discusses DRL-based end-to-end network slicing. We have briefly studied the end-to-end resource orchestration problem and presented a new system architecture to enable DRL-based network slicing. We have also analyzed the challenges of deploying DRL-based solutions from the perspectives of both algorithm and system. Moreover, we have explored and envision multiple promising techniques, e.g., safety and distributed DRL and transfer learning, for automating end-to-end network slicing.

## REFERENCES

[1] Global Mobile Suppliers Association, "5G network slicing for vertical industries." [Online]. Available: http://www.huawei.com/minisite/5g/img/5g-network-slicing-for-vertical-industries-en.pdf

[2] L. U. Khan, I. Yaqoob, N. H. Tran, Z. Han, and C. S. Hong, "Network slicing: Recent advances, taxonomy, requirements, and open research challenges," *IEEE Access*, vol. 8, pp. 36 009–36 028, 2020.

[3] J. A. Hurtado Sánchez, K. Casilimas, and O. M. Caicedo Rendon, "Deep reinforcement learning for resource management on network slicing: A survey," *Sensors*, vol. 22, no. 8, p. 3031, 2022.

[4] V. P. Kafle, P. Martinez-Julia, and T. Miyazawa, "Automation of 5g network slice control functions with machine learning," *IEEE Communications Standards Magazine*, vol. 3, no. 3, pp. 54–62, 2019.

[5] Q. Liu, T. Han, and E. Moges, "Edgeslice: Slicing wireless edge computing network with decentralized deep reinforcement learning," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2020, pp. 234–244.

[6] O-RAN, "O-RAN Architecture Description," Open Radio Access Network Alliance (O-RAN), Tech. Rep., 2019, version 3.0.

[7] 3GPP, "Management and orchestration, Architecture framework," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 28.533, 2020, version 16.4.0.

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

[9] H. Mao, M. Schwarzkopf *et al.*, "Learning scheduling algorithms for data processing clusters," in *Proceedings of the ACM Special Interest Group on Data Communication*. ACM, 2019, pp. 270–288.

[10] C. Marquez, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "How should i slice my network? a multi-service empirical evaluation of resource sharing efficiency," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, 2018, pp. 191–206.

[11] X. Foukas, N. Nikaein, M. M. Kassem, M. K. Marina, and K. Kontovasilis, "Flexran: A flexible and programmable platform for software-defined radio access networks," in *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, 2016, pp. 427–441.

[12] 3GPP, "Management and orchestration; 5G Network Resource Model (NRM)," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 28.541, 2020, version 16.6.0.

[13] C. Tessler, D. J. Mankowitz, and S. Mannor, "Reward constrained policy optimization," *arXiv preprint arXiv:1805.11074*, 2018.

[14] 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.

[15] Z. Zhu, K. Lin, and J. Zhou, "Transfer learning in deep reinforcement learning: A survey," *arXiv preprint arXiv:2009.07888*, 2020.