Joint Multi-objective Optimization for Radio Access Network Slicing Using Multi-agent Deep Reinforcement Learning

Guorong Zhou, Liqiang Zhao, *Member, IEEE*, Gan Zheng, *Fellow, IEEE*, Zhijie Xie, S.H. Song, *Member, IEEE*, and Kwang-Cheng Chen, *Fellow, IEEE*

***Abstract*—Radio access network (RAN) slices provide multiple customized services by virtual base stations (vBSs) and virtual radio resources allocation. As a result, multiple performance metrics need to be jointly considered. There have been some works on multi-objective optimization for RAN slicing, but only in the scalar form. In this paper, we consider non-scalar multi-objective optimization for RAN slicing with three types of slices, *i.e.*, the high-bandwidth slice, the low-delay slice, and the wide-coverage slice over the same underlying physical network. We jointly optimize the throughput, the transmission delay, and the coverage area by dynamic vBSs deployment, and sub-channel and power allocation. An improved multi-agent deep deterministic policy gradient (IMADDPG) algorithm, having the characteristics of centralized training and distributed execution, is proposed to solve the above non-deterministic polynomial-time hard (NP-hard) problem. The rank voting method is introduced in the testing process to obtain near-Pareto optimal solutions. Simulation results verify that the proposed method can ensure better performance than the equal resource allocation algorithm and the multi-agent deep deterministic policy gradient (MADDPG) algorithm. The proposed algorithm has the advantage of approaching any point of the Pareto boundary, while the traditional scalar method only subjectively approaches one of the Pareto optimal solutions. Furthermore, our proposal strikes a compelling tradeoff among three types of RAN slices due to the non-dominance between Pareto optimal solutions.**

***Index Terms*—Radio access network slicing, multi-objective optimization, non-scalarization, multi-agent deep reinforcement learning, rank voting method.**

I. Introduction

As an important part of end-to-end network slicing [1], [2], radio access network (RAN) slicing [3], [4] can dynamically allocate virtual radio resources and deploy virtual base stations (vBSs) in RAN to provide customized services to users. Different from the mature core network (CN) slicing and transport network (TN) slicing, many research challenges remain on RAN slicing, particularly diverse service requirements in the time-varying wireless channel environments.

Guorong Zhou, Liqiang Zhao (e-mail: guor_zhou@163.com, lqzhao@mail.xidian.edu.cn) are with the State Key Laboratory of Integrated Service Networks at Xidian University, Xi'an 710071, China. Gan Zheng (e-mail: g.zheng@lboro.ac.uk) is with the Wolfson School of Mechanical, Electrical and Manufacturing Engineering, Loughborough University, UK. Zhijie Xie, S.H. Song (e-mail: shiehshiehzhijie@gmail.com, eeshsong@ust.hk) are with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong. Kwang-Cheng Chen (e-mail: kwangcheng@usf.edu) is with the Department of Electrical Engineering, University of South Florida, Tampa, Florida, USA.

The current research on RAN slicing mainly focuses on performance optimization. For example, for the multi-tenant heterogeneous cloud RAN, Lee *et al.* [5] proposed a dynamic network slicing scheme to achieve higher network throughput. Xiang *et al.* [6] developed two reinforcement learning based algorithms to solve the high-complexity system power optimization problem under traditional and fog users' specific performance requirements. Tang *et al.* [7] investigated the average delay optimization problem of network slicing in the fog RAN. Overall, all these works focused on single-objective optimization problem (SOOP). However, RAN slicing shall provide various customized services, as it is insufficient to optimize only one criterion in a multi-slicing network.

Specifically, through RAN slicing, the physical network is divided into multiple virtual networks, which provide different types of customized services according to specific requirements. Each of these customized slices has its concerned performance metrics. For example, bandwidth-extensive slicing hopes to provide users with high-throughput services, while delay-sensitive slicing prefers to ensure ultra-low delay. It is difficult to utilize a centralized controller to simultaneously manage multiple customized slices [8]–[11], or to optimize a multi-slicing network with only one performance metric [12], [13]. To meet complex demands, it is indispensable to bring multi-objective optimization problem (MOOP) formulation into RAN slicing.

MOOP is a mathematical framework to jointly optimize multiple objectives, which is fundamentally different from SOOP. The global optimal solution of SOOP is uniquely determined, while the solutions of MOOP are composed of a number of Pareto optimal solutions [14]. These solutions are not dominated by each other, specifically defined are as follows [15]:

Definition 1 (Pareto optimal solutions). Assume the attainable objective set of $F = \{\mathbf{f}(\mathbf{x}), \mathbf{x} \in \chi\}$ contains all the combinations of M objectives $f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_M(\mathbf{x})$ that are simultaneously attainable under the available resources region χ. Then $\mathbf{f}^*(\mathbf{x}) \in F$ is one of the Pareto optimal solutions if there does not exist any $\mathbf{f}'(\mathbf{x}) \in F \setminus \{\mathbf{f}^*(\mathbf{x})\}$ with $f'_m(\mathbf{x}) \geq f^*_m(\mathbf{x})$ for $\forall m = 1, 2, \cdots, M$.

Therefore, the non-dominance of Pareto optimal solutions means that the improvement of a certain objective may cause the degradation of other objectives, and it is impossible to optimize multiple objectives at the same time.

There have been some studies on MOOP for RAN slicing,

but only in the scalar form. The scalar method transforms the MOOP into a SOOP for a tractable solution, mainly including the weighted sum method, the constraint method, the mini-max method, etc. Its optimal solution only subjectively corresponds to one of the Pareto optimal solutions. For example, Shi *et al.* [16] analyzed the energy efficiency (EE) and delay in wireless networks virtualization, but only EE was regarded as the objective while the delay was treated as a constraint. Although both the spectrum efficiency and the service level agreement (SLA) satisfaction ratio were considered, Hua *et al.* [17] used the weighted sum to project two metrics into a unified utility function. Afolabi *et al.* [18] used the constraint method to minimize the amount of computational resources under the constraint of mean response time in the end-to-end mobile network slice. Xu *et al.* [19] optimized the amount of allocated resources with the delay constraints. Guan *et al.* [20] jointly considered three typical slices by optimizing the infrastructure resource efficiency. Although the traditional scalar method has low computational complexity, it suffers some drawbacks. Firstly, the objective functions are empirically designed and not feasible to dynamically obtain the satisfactory solution in practice. Secondly, the optimization result of each slice is highly dependent on the self-defined weight, which results in the poor generalization. Finally, the scalar method requiring extensive prior knowledge cannot warrant convergence to the non-convex region on Pareto boundary.

A better way to solve MOOP of RAN slicing is the non-scalar method [21], [22]. However, the non-scalar MOOP for multi-slicing network is always a nondeterministic polynomial-time hard (NP-hard) problem due to the expansion of optimization objectives. Traditional algorithms usually generate high computational complexity, while the deep reinforcement learning (DRL) algorithms demonstrate superior computing and learning ability in solving high-complexity problems [23]–[25]. When using the DRL algorithm to solve non-scalar MOOPs, each agent can only define one reward function to learn one of the optimization objectives, so we have to utilize the multi-agent DRL algorithm to manage multiple RAN slices. Wherein, the multi-agent deep deterministic policy gradient (MADDPG) algorithm can define multiple agents for different kinds of RAN slices, which is suitable for solving complex non-scalar MOOPs [24]–[26]. Nevertheless, the MADDPG algorithm is eventually confined to converge to one solution and we cannot guarantee that it is (near-)Pareto optimal. Therefore, we propose the improved multi-agent deep deterministic policy gradient (IMADDPG) algorithm by adding rank voting method in the testing process. The algorithm not only retains the advantages of DRL algorithm which can avoid local optimization and deal with non-differentiable, discontinuous and non-convex problems, but also can learn multiple solutions in parallel to approximate various Pareto optimal solutions.

In this paper, we establish three kinds of RAN slices simultaneously, *i.e.* the high-bandwidth slice, the low-delay slice and the wide-coverage slice, upon the same underlying physical network. Since the performance indicators of slices are different from each other, a non-scalar MOOP is formulated to jointly optimize the throughput, the transmission delay and the coverage area by dynamically deploying vBSs and allocating suitable subchannels and power resources to users on each slice. In order to solve the above NP-hard problem, the IMADDPG algorithm is proposed. The simulation results show that the IMADDPG algorithm can optimize three kinds of RAN slices simultaneously and efficiently, ensure better performance than benchmark algorithms and obtain near-Pareto optimal solutions. The main contributions of this paper are as follows:

- A dynamically adaptive model is built to jointly support multiple categories of RAN slices upon the same underlying physical network, and we consider to optimize three typical types of slices in this paper. Then, we jointly deploy proper vBSs and allocate suitable subchannels and power resources to users on each RAN slice in order to approach the Pareto optimum of three slices.
- We formulate a non-scalar MOOP to jointly optimize the throughput, the transmission delay and the coverage area on RAN slicing, whilst the exiting works have considered the scalar MOOP. The non-scalar method successfully avoids the difficulty of using the scalarization, where the performance optimization result of each slice depends heavily on its weight. And the non-scalar method has a better generalization, as it does not need to design the utility function in advance.
- To our best knowledge, this is new to solve the non-scalar MOOP by a multi-agent DRL algorithm. We present the IMADDPG model structure and propose the IMADDPG algorithm, where the structure has the characteristics of centralized training and distributed execution, and the algorithm can obtain better optimization results. Compared with the MADDPG algorithm, we introduce rank voting method in the testing process of the IMADDPG algorithm for obtaining near-Pareto optimal solutions.
- The proposed method strikes a compelling tradeoff among three very different types of RAN slices. Due to the non-dominance between Pareto optimal solutions, the improvement of the metric on one slice class will be accompanied by the degradation of the metrics on the other one or two slice classes. Therefore, the throughput, the transmission delay and the coverage area cannot be improved at the same time. It requires the mobile network operator (MNO) to actually execute one of the near-Pareto optimal solutions according to the priority of slices. Our main contributions are also contrasted at a glance boldly and explicitly to the relevant references in Table 1.

The rest of the paper is organized as follows. In Section II, the system model is proposed, and three different types of RAN slices on the foundation of the same physical network are described in detail. In Section III, we formulate the non-scalar MOOP on RAN slicing. In Section IV, we solve the proposed MOOP by presenting an IMADDPG model structure and using the IMADDPG algorithm. In Section V, we discuss simulation results. Finally, the paper is concluded in Section VI.

Table 1: A comparative summary of contributions of the salient existing works.

Novelty	[5]	[6]	[7]	[11]	[16]	[17]	[20]	[27]	[28]	[29]	Proposed
Jointly establishing at least three typical types of slices						✓	✓		✓		✓
Considering a dynamically adaptive multi-slicing network	✓	✓			✓	✓	✓	✓	✓	✓	✓
Considering vBSs' deployment	✓	✓	✓	✓			✓				✓
Allocation of subchannel resource	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓
Allocation of power resource	✓	✓			✓			✓	✓	✓	✓
Optimization of network throughput	✓			✓							✓
Optimization of transmission delay		✓	✓	✓	✓	✓					✓
Optimization of coverage area										✓	✓
Simultaneously optimizing at least three performance metrics				✓			✓				✓
Formulating a non-scalar MOOP							✓				✓
Solving the non-scalar MOOP by a multi-agent DRL algorithm											✓
Introducing ensemble strategy (*e.g.*, rank voting method)											✓
Obtaining near-Pareto optimal solutions											✓
Tradeoff among different types of RAN slices											✓

II. System Model

In this paper, we focus on analyzing three typical classes of RAN slices [30] upon one physical network, as shown in Fig. 1. Firstly, without loss of generality, we consider that there are M physical base stations (BSs) with K users distributed in a physical RAN, where each BS can support multiple slices. Let $\mathbf{M}=\{1,...,M\}$ and $\mathbf{K}=\{1,...,K\}$ denote the sets of BSs and users, respectively. Suppose the available power of each BS is P_B, and the orthogonal frequency division multiple access (OFDMA) method is adopted in our system, where the intra-cell interference is not present. The system bandwidth of B Hz is divided into N subchannels with the set of $\mathbf{N}=\{1,...,N\}$ and each subchannel has a bandwidth of B/N Hz, so the BS can provide uplink or downlink communication for users associated with these subchannels.

Then, for virtualizing three typical communication scenarios into the high-bandwidth slice, the low-delay slice and the wide-coverage slice, which are referred to as class-s slices, $s\in\{1,2,3\}$, respectively, MNO abstracts the communication resources in the system into multiple shared virtual resources. Besides, as seen in Fig. 1, a physical BS will be virtualized into three vBSs, each of which is associated with a specific service (*i.e.*, class-s slice) [31]. Among different scenarios, MNO determines the flexible deployment of vBSs and the real-time amount of virtual spectrum and power resources allocated to each vBS based on users' diverse location information and QoS requirements, in order to form customized RAN slices with isolation guarantee. Note that since the locations of vBSs served in class-s slices correspond to those of physical BSs one by one, we can represent them with the same parameter $m,\forall m\in\mathbf{M}$ as physical ones. We consider a dynamically adaptive multi-slicing network, and define the time slot as a time interval $[t,t+1)$, $t\in\{1,2,...,T\}$, where T is assumed as the final time slot of our considered scenario and the slot duration is δ s/slot.

We assume that K_s users would like to request class-s slices, and the set of users is denoted as $\mathbf{K_s}=\{1,...,K_s\}$, $\mathbf{K_s}\subset\mathbf{K}$. Therefore, each vBS needs to access multiple users of class-s slices to provide multiple slices. The channel coefficient $h^n_{k,s,m}(t)$ of user $k,\forall k\in\mathbf{K_s}$ on subchannel $n,\forall n\in\mathbf{N}$ associated with vBS $m,\forall m\in\mathbf{M}$ on class-s slices at slot t is independent and identically distributed (i.i.d.) over time

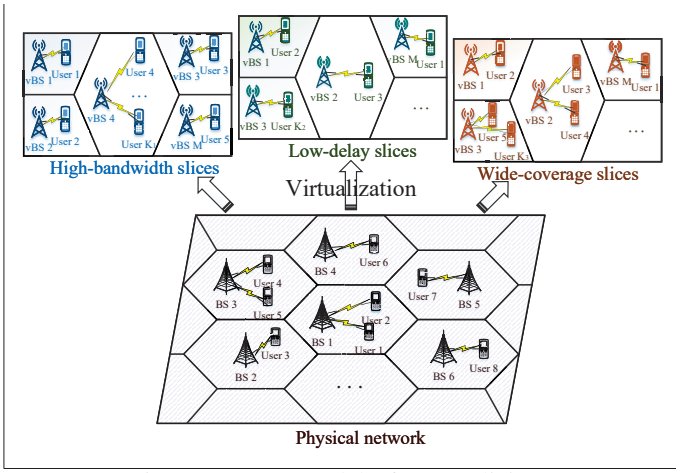


Fig. 1: System model of RAN slicing.

and $p^n_{k,s,m}(t)$ is the transmit power of user k associated with vBS m and subchannel n on class-s slices at slot t, while $\mathbf{P_s}(t)=\{p^n_{k,s,m}(t)\}$ is the set of transmit powers. In addition, we define $\mathbf{X_s}(t)=\{x^n_{k,s,m}(t)\}$ as the set of subchannel allocation indicators and $x^n_{k,s,m}(t)$ is a binary factor of user k associated with vBS m and subchannel n on class-s slices at slot t. $\mathbf{Y_s}(t)=\{y_{k,s,m}(t)\}$ is defined as the set of vBS association indicators, where $y_{k,s,m}(t)$ is a binary factor of user k associated with vBS m on class-s slices at slot t. Their specific expressions are as follows:

$$x^n_{k,s,m}(t)=\begin{cases}1, \text{if } n \text{ is allocated to user } k \text{ of slice } s,\\ 0, \text{otherwise},\end{cases} \tag{1}$$

and

$$y_{k,s,m}(t)=\begin{cases}1, \text{if user } k \text{ of slice } s \text{ is associated with } m,\\ 0, \text{otherwise}.\end{cases} \tag{2}$$

We can get the inter-cell interference of user k associated with vBS m and subchannel n on class-s slices at slot t as:

$$I^n_{k,s,m}(t)=\sum_{j\in\mathbf{M}\backslash\{m\}}\sum_{i\in\mathbf{K_s}\backslash\{k\}} y_{i,s,j}(t)\,x^n_{i,s,j}(t)\,p^n_{i,s,j}(t)\left|h^n_{i,s,j}(t)\right|^2, \tag{3}$$

where $h^n_{i,s,j}(t)$ is the channel propagation coefficient of the interference user $i,\forall i\in\mathbf{K_s}\backslash\{k\}$ on subchannel n associated

with vBS $j, \forall j \in \mathbf{M} \backslash \{m\}$ on class-s slices at slot t. $p^n_{i,s,j}(t)$ is the interference power of user i associated with vBS j and subchannel n on class-s slices at slot t. Also, $x^n_{i,s,j}(t)$ and $y_{i,s,j}(t)$ are defined as the subchannel allocation indicators as well as vBS association indicators of the interference users on class-s slices, respectively.

Specifically, for class-1 slices, such as the video streaming service, high-bandwidth data are transmitted through the wireless downlink. The data rate $r^n_{k,1,m}(t)$ of user k on subchannel n associated with vBS m on class-1 slices at slot t is given by

$$r^n_{k,1,m}(t) = \frac{B}{N}\log_2\left[1 + \frac{p^n_{k,1,m}(t)\left|h^n_{k,1,m}(t)\right|^2}{I^n_{k,1,m}(t) + (\frac{B}{N})N_0}\right], \quad (4)$$

where N_0 is the power spectral density (PSD) of the additive white Gaussian noise (AWGN). Then the total downlink data rate of user k on class-1 slices at slot t is formulated as

$$R_{k,1}(t) = \sum_{n\in\mathbf{N}}\sum_{m\in\mathbf{M}} y_{k,1,m}(t)x^n_{k,1,m}(t)r^n_{k,1,m}(t). \quad (5)$$

Hence, throughput for the whole class-1 slices at slot t is expressed as

$$R^{sum}_1(t) = \sum_{k\in\mathbf{K_1}} R_{k,1}(t). \quad (6)$$

Obviously, $R^{sum}_1(t)$ is the main performance metric that MNO needs to optimize in order to satisfy users' requirements across the whole class-1 slices.

As for class-2 slices, such as virtual reality (VR)/augmented reality (AR), its primary target is to reduce the transmission delay to guarantee users' quality of service (QoS). The vBSs transmit low-delay tasks for users through downlink. The data rate $r^n_{k,2,m}(t)$ of user k on subchannel n associated with vBS m on class-2 slices at slot t is given by

$$r^n_{k,2,m}(t) = \frac{B}{N}\log_2\left[1 + \frac{p^n_{k,2,m}(t)\left|h^n_{k,2,m}(t)\right|^2}{I^n_{k,2,m}(t) + (\frac{B}{N})N_0}\right]. \quad (7)$$

Then the total downlink data rate of user k on class-2 slices at slot t is formulated as

$$R_{k,2}(t) = \sum_{n\in\mathbf{N}}\sum_{m\in\mathbf{M}} y_{k,2,m}(t)x^n_{k,2,m}(t)r^n_{k,2,m}(t). \quad (8)$$

Subsequently, we define $A_2(t)=\{A_{1,2}(t),A_{2,2}(t),...,A_{k,2}(t), ...\}$ as the process of random data arrivals on class-2 slices at slot t, where $A_2(t)$ is assumed to be independent among the users and i.i.d. across the slots, which follows a Poisson arrival process with average rate λ (in bits/slot). Let $Q_2(t) = \{Q_{1,2}(t), Q_{2,2}(t), ..., Q_{k,2}(t), ...\}$ be the current data queue length on class-2 slices at slot t. Based on queuing theory, the queue evolution of user k on class-2 slices at slot t can be written as [32]:

$$Q_{k,2}(t+1) = \max\{Q_{k,2}(t) - R_{k,2}(t)\,\delta + A_{k,2}(t), 0\}, \forall k. \quad (9)$$

Due to the fact that the user's average queuing delay is proportional to the average queue length according to Little's Theorem [33], we can represent the delay of user k by the queue length $Q_{k,2}(t)$. We do not consider the access delay in this paper due to its small proportion [34].

Therefore, the average delay of all users on class-2 slices can be calculated as

$$Q^{ave}_2(t) = \frac{1}{K_2}\sum_{k\in\mathbf{K_2}} Q_{k,2}(t). \quad (10)$$

In order to meet the low-delay requirements of all users on class-2 slices, we aim to minimize $Q^{ave}_2(t)$.

Class-3 slices provide the basic access services with best effort for a large area with a low user density, such as a desert area with few people. Due to the time-varying channel state and diverse transmit power, the basic access services are also variable-rate. Consequently, the main performance metric of class-3 slices is to provide the user coverage as wide as possible. We focus on improving the signal to interference plus noise power ratio (SINR) of all users to ensure the wide coverage area. The SINR of user k on class-3 slices at slot t is as follows:

$$SINR_{k,3}(t) = \sum_{m\in\mathbf{M}}\sum_{n\in\mathbf{N}} \frac{y_{k,3,m}(t)x^n_{k,3,m}(t)p^n_{k,3,m}(t)\left|h^n_{k,2,m}(t)\right|^2}{I^n_{k,3,m}(t) + (\frac{B}{N})N_0}. \quad (11)$$

Therefore, we get the average SINR of class-3 slices below:

$$SINR^{ave}_3(t) = \frac{1}{K_3}\sum_{k\in\mathbf{K_3}} SINR_{k,3}(t). \quad (12)$$

III. Multi-Objective Problem Formulation

Although we have established an independent optimization objective in the previous section, these slices are created based on the same physical network. Explicitly, the vBSs and the underlying bandwidth and power resources are shared by three types of slices, so we should give a MOOP in the multi-slicing network.

Firstly, we formulate the power consumption of each vBS as:

$$P_m(t) = \sum_{s\in\{1,2,3\}}\sum_{n\in\mathbf{N}}\sum_{k\in\mathbf{K_s}} y_{k,s,m}(t)x^n_{k,s,m}(t)p^n_{k,s,m}(t). \quad (13)$$

As for the dynamically adaptive multi-slicing network, instead of optimizing instantaneous objectives, it is more effective to adopt the long-term optimization objectives to estimate network performances [35]. Therefore, we can get the MOOP to jointly optimize three different long-term objectives of the throughput, the average delay and the average SINR for three types of RAN slices, which is formulated in a non-scalar

form, as follows:

$$
\begin{aligned}
&\mathbf{P}^*: \\
&\max_{\mathbf{X_s},\mathbf{Y_s},\mathbf{P_s}} \left\{ \sum_{t=1}^{T} R_1^{sum}(t), \sum_{t=1}^{T} [\beta - Q_2^{ave}(t)], \sum_{t=1}^{T} SINR_3^{ave}(t) \right\}, \\
&\text{s.t. } C1: \sum_{k\in\mathbf{K_s}} x_{k,s,m}^{n}(t)\, y_{k,s,m}(t) \le 1, \forall n \in \mathbf{N}, m \in \mathbf{M}, \\
&\quad C2: \sum_{n\in\mathbf{N}} \sum_{m\in\mathbf{M}} x_{k,s,m}^{n}(t)\, y_{k,s,m}(t) \le 1, \forall k \in \mathbf{K_s}, \\
&\quad C3: \sum_{m\in\mathbf{M}} y_{k,s,m}(t) \le 1, \forall k \in \mathbf{K_s}, \\
&\quad C4: p_{k,s,m}^{n}(t) \ge 0, \forall k \in \mathbf{K_s}, n \in \mathbf{N}, m \in \mathbf{M}, \\
&\quad C5: x_{k,s,m}^{n}(t), y_{k,s,m}(t) \in \{0,1\}, \forall k \in \mathbf{K_s}, n \in \mathbf{N}, m \in \mathbf{M}, \\
&\quad C6: P_m(t) \le P_B, \forall m \in \mathbf{M}, \\
&\quad C7: \left(\sum_{k\in\mathbf{K_s}} \sum_{m\in\mathbf{M}} x_{k,s,m}^{n}(t)\, y_{k,s,m}(t) \right) \\
&\qquad \cdot \left(\sum_{k\in\mathbf{K_{s'}}} \sum_{m\in\mathbf{M}} x_{k,s',m}^{n}(t)\, y_{k,s',m}(t) \right) = 0, \\
&\qquad if\ s \ne s', \forall n \in \mathbf{N}, s, s' \in \{1,2,3\}, \\
&\quad C8: \sum_{s\in\{1,2,3\}} \sum_{n\in\mathbf{N}} \sum_{k\in\mathbf{K_s}} x_{k,s,m}^{n}(t) \le N, \forall m \in \mathbf{M},
\end{aligned}
\tag{14}
$$

where β is the initial threshold to guarantee the non-negativity of optimization objective. $C1$ means that a subchannel could only be allocated to at most one user in the same cell in the same slot, $C2$ represents that we could assign at most one subchannel to a user in the same slot, and $C3$ indicates that a user could only be associated with one vBS in the same time slot. $C4$ is the non-negative constraint of the transmit power. $C5$ represents the value range of binary variables $x_{k,s,m}^{n}(t)$ and $y_{k,s,m}(t)$. $C6$ represents that the actual power consumption of each vBS should not exceed the total available power P_B. $C7$ is to ensure the isolation of spectrum resources between slices, that is, the subchannels used by class-s slices at slot t do not overlap with each other. $C8$ guarantees that the sum of used subchannels in each vBS at slot t is no more than the total number of subchannels N.

However, since our proposed model considers inter-cell interference and introduces binary variables $\{\mathbf{X_s}, \mathbf{Y_s}\}$, (14) is a non-convex problem. Moreover, (14) is established as a non-scalar MOOP, which is an NP-hard problem. It is difficult to use the traditional methods to approach Pareto optimal solutions.

IV. Problem Transformation and Solution

In this section, based on the MADDPG algorithm [26], we present the IMADDPG model structure and propose the online off-policy IMADDPG algorithm to solve (14). Then, we add the rank voting method in the testing process of the IMADDPG algorithm to approach Pareto optimal solutions.

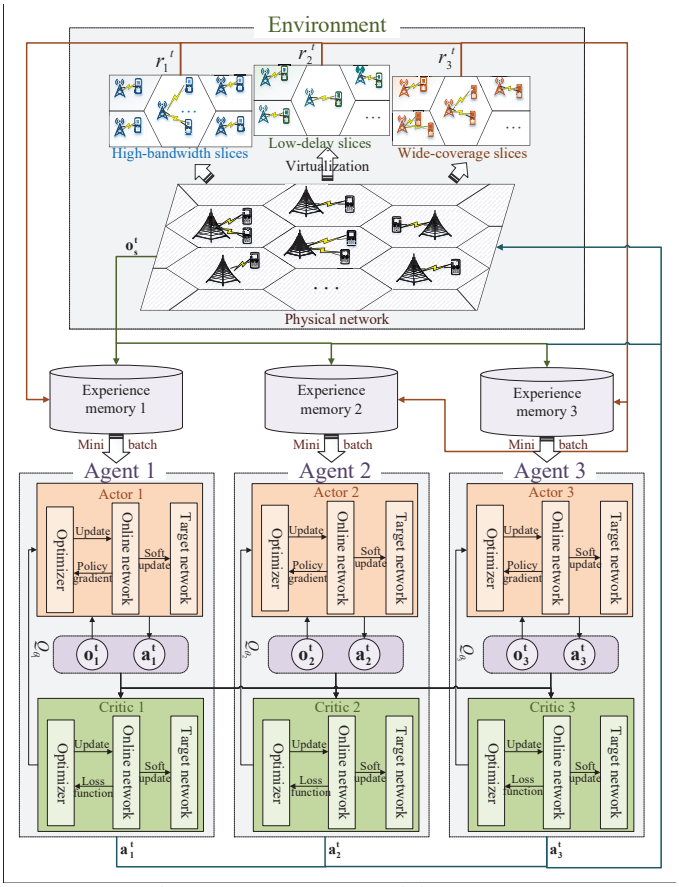


Fig. 2: IMADDPG model structure.

A. The IMADDPG Model Structure

The IMADDPG model structure is shown in Fig. 2. We assume that an agent's tasks satisfy Markov attributes in the IMADDPG model, where the agent is the main body of DRL, equivalent to the player in a game, and each agent can be given different reward functions for multi-objective optimization. A Markov decision process (MDP) consists of a four-tuple of $\langle \mathbf{O}, \mathbf{A}, \mathbf{R}, \mathbf{O}' \rangle$, where $\mathbf{O}, \mathbf{A}, \mathbf{R}$ and $\mathbf{O}'$ are the set of state observation, action space, reward functions and the next state observation, respectively. In the MDP, the agent dynamically takes actions based on users' requests and channel conditions in units of time steps. That is, at one time step, the agents observe the current state by online interacting with environment and then select an action from the action space based on some policies. By executing this action, a reward could be received from the environment and the agents transfer to the state of the next time step. However, we cannot collect the global action information to directly know the optimal MDP due to the huge continuous action space in the IMADDPG model. Hence, the agents have to train deep neural networks (DNNs) approximating policy functions and value functions, in order to learn the optimal decision making.

Considering that the dynamic resource sharing among slices upon the same physical network is similar to the multi-agent cooperation scenario of the IMADDPG algorithm, we can transform three classes of RAN slices in the system model into three agents in the IMADDPG model, as shown in Fig. 2.

Then, the MDP's four-tuples of $\left\langle \mathbf{O}, \mathbf{A}, \mathbf{R}, \mathbf{O}^{'} \right\rangle$ could be corresponded to the elements in the system model.

State and the next state: The state observation is a vector of channel states and users' requests in the system model. Hence, at the time step t (corresponded to the slot t in the system model), the state observation of agent s is defined as [17], [36]:

$$\mathbf{o_s^t} = \left\{ \mathbf{h_s}(\mathbf{t}), \mathbf{h_s^{in}}(\mathbf{t}), \mathbf{A_s}\left(\mathbf{t}\right) \right\}, \tag{15}$$

where $\mathbf{h_s}(\mathbf{t})$ is the set of the channel coefficient $h_{k,s,m}^{n}(t)$ and $\mathbf{h_s^{in}}(\mathbf{t})$ is the set of the channel coefficient $h_{i,s,j}^{n}(t)$ from interference users, and $\mathbf{A_s}\left(\mathbf{t}\right)$ is the set of the data arrival process in agent s.

Obviously, the next state observation of agent s can be denoted as:

$$\mathbf{o_s^{t+1}} = \left\{ \mathbf{h_s}(\mathbf{t+1}), \mathbf{h_s^{in}}(\mathbf{t+1}), \mathbf{A_s}\left(\mathbf{t+1}\right) \right\}, \tag{16}$$

where $\mathbf{h_s}(\mathbf{t+1})$, $\mathbf{h_s^{in}}(\mathbf{t+1})$ and $\mathbf{A_s}\left(\mathbf{t+1}\right)$ are the set of the channel coefficient, the channel coefficient from interference users and the data arrival process in agent s at the next time step.

Action: At the time step t, the MNO needs to deploy suitable vBSs and allocates optimal subchannels and power resources to users of agent s according to the state observation $\mathbf{o_s^t}$. Hence, the action set is denoted as [17], [36]:

$$\mathbf{a_s^t} = \left\{ \mathbf{X_s}, \mathbf{Y_s}, \mathbf{P_s} \right\}. \tag{17}$$

Here, in order to facilitate mathematical analysis, we relax the binary variables in $\mathbf{X_s}$ and $\mathbf{Y_s}$ to continuous variables with value range of [0,1], which also conforms to the condition of continuous action space in IMADDPG algorithm. Note that the selected actions have to satisfy the constraints of $\mathbf{P}^*$, in which those unsatisfied actions will be modified. Specifically, at the time step t, agent s makes the resource decision to obtain $\mathbf{a_s^t}$, and then checks whether the action set satisfies the constraints. If some of actions $\mathbf{X_s}$ and $\mathbf{Y_s}$ violate the resource constraints $C1$-$C3$, $C7$ and $C8$, only one of them retains its original value, while the other conflicting actions will be modified to 0 to meet the constraints, which means that the corresponding users will not be served. If some of actions $\mathbf{P_s}$ have the resource collision of $C4$ and $C6$, they are replaced with a low value to satisfy the constraints.

Reward: We define different reward functions for each agent in the IMADDPG model because the main performance metrics of three classes of RAN slices are different. According to (14), the reward space of agent s can be expressed as follows [35]:

$$r_s^t = \begin{cases} \overline{R_1^{sum}}\left(t\right), s = 1, \\ \left[1 - \overline{Q_2^{ave}}\left(t\right)\right], s = 2, \\ \overline{SINR_3^{ave}}\left(t\right), s = 3, \end{cases} \tag{18}$$

where we normalize $R_1^{sum}\left(t\right)$, $Q_2^{ave}\left(t\right)$ and $SINR_3^{ave}\left(t\right)$ by 0-1 normalization, respectively, in order to improve the convergence rate of the model.

Basic structure of model: The IMADDPG model employs the actor-critic (AC) algorithm [26] as the basic structure of each agent. Different from the traditional AC structure applied to deep deterministic policy gradient (DDPG) algorithm, our IMADDPG model utilizes three parallel distributed AC structures, thus has the characteristics of centralized training and distributed execution, which is suitable for complex multi-agent scenarios to solve (14). Specifically, the agent s contains an actor which only needs local information for policy decision and a critic which collects all agents' global policies for action value learning, as seen from Fig. 2. We use different DNNs as the approximation of the policy function $\mu_{\theta_s}\left(\mathbf{o_s^t}|\theta^{\mu_s}\right)$ (namely actor s) and the action-value function $Q_{\theta_s}\left(\mathbf{o_s^t}, \mathbf{a_s^t}, \mathbf{a_{S\backslash s}^t}|\theta^{Q_s}\right)$ (namely critic s), respectively, and use the stochastic gradient method to train θ^{μ_s} and θ^{Q_s}. Wherein, actor s's input is the current state $\mathbf{o_s^t}$ and the output is the deterministic action $\mathbf{a_s^t}$, while critic s's inputs include the state $\mathbf{o_s^t}$, the action $\mathbf{a_s^t}$ of the current agent s and $\mathbf{a_{S\backslash s}^t}$ of the other two agents generated by the actor, and the output is the estimated Q-value. That is, the actor is responsible for selecting an action based on the current local state observation, then the critic evaluates this action according to all agents' global information and gives a score, and finally the actor modifies the probability of subsequent action selection based on the critic's score. In this way, the learning process of IMADDPG algorithm is more stable and can converge faster. Therefore, we can get the optimal reward r_s^t by continuously updating the actor and critic.

Update of actor: By considering (14) and (18), the objective function of agent s in the IMADDPG algorithm is defined as the expectation of the long-term discounted cumulative reward, namely:

$$J_s\left(\mu\right) = E_\mu\left[r_s^1 + \gamma r_s^2 + \gamma^2 r_s^3 + \cdots + \gamma^{T-1} r_s^T\right], \tag{19}$$

where γ is the discount factor. The objective of the actor s is to find the optimal deterministic policy $\mu_{\theta_s}^*$, which is equivalent to the maximization of $J_s\left(\mu\right)$.

$$\mu_{\theta_s}^* = \arg\max_{\mu} J_s\left(\mu\right). \tag{20}$$

In this way, the deterministic action $\mathbf{a_s^t}$ of agent s at time step t can be obtained through the optimal policy function $\mu_{\theta_s}^*$:

$$\mathbf{a_s^t} = \mu_{\theta_s}^*\left(\mathbf{o_s^t}|\theta^{\mu_s}\right). \tag{21}$$

It has been proved that the gradient of $J_s\left(\mu\right)$ with respect to θ^{μ_s} is equivalent to the expected gradient of $Q_{\theta_s}\left(\mathbf{o_s^t}, \mathbf{a_s^t}, \mathbf{a_{S\backslash s}^t}|\theta^{Q_s}\right)$ with respect to θ^{μ_s} in [37].

As for the action value function $Q_{\theta_s}\left(\mathbf{o_s^t}, \mathbf{a_s^t}, \mathbf{a_{S\backslash s}^t}|\theta^{Q_s}\right)$ of agent s, based on MDP, it can be formally defined using Bellman's equation, as follows:

$$\begin{aligned} &Q_{\theta_s}\left(\mathbf{o_s^t}, \mathbf{a_s^t}, \mathbf{a_{S\backslash s}^t}|\theta^{Q_s}\right) \\ &= r_s^t + \gamma \max_{\mathbf{a}} Q_{\theta_s}\left(\mathbf{o_s^{t+1}}, \mathbf{a_s^{t+1}}, \mathbf{a_{S\backslash s}^{t+1}}|\theta^{Q_s}\right), \end{aligned} \tag{22}$$

where r_s^t is the current reward function. $\mathbf{o_s^{t+1}}$, $\mathbf{a_s^{t+1}}$ and $\mathbf{a_{S\backslash s}^{t+1}}$ are the next state of agent s, the next action of agent s and the next action of the other two agents, respectively.

Subsequently, the update of $Q_{\theta_s}\left(\mathbf{o_s^t}, \mathbf{a_s^t}, \mathbf{a_{S\backslash s}^t}|\theta^{Q_s}\right)$ is as follows:

$$Q_{\theta_s}\left(\mathbf{o_s^t}, \mathbf{a_s^t}, \mathbf{a_{S\backslash s}^t}|\theta^{Q_s}\right) \leftarrow Q_{\theta_s}\left(\mathbf{o_s^t}, \mathbf{a_s^t}, \mathbf{a_{S\backslash s}^t}|\theta^{Q_s}\right) + \alpha\delta_t, \tag{23}$$

where

$$\begin{aligned}\delta_t = & r_s^t + \gamma \max_{\mathbf{a}} Q_{\theta_s}\left(\mathbf{o_s^{t+1}}, \mathbf{a_s^{t+1}}, \mathbf{a_{S\backslash s}^{t+1}} | \theta^{Q_s}\right) \\ & - Q_{\theta_s}\left(\mathbf{o_s^t}, \mathbf{a_s^t}, \mathbf{a_{S\backslash s}^t} | \theta^{Q_s}\right),\end{aligned} \tag{24}$$

where $r_s^t + \gamma \max_{\mathbf{a}} Q_{\theta_s}\left(\mathbf{o_s^{t+1}}, \mathbf{a_s^{t+1}}, \mathbf{a_{S\backslash s}^{t+1}} | \theta^{Q_s}\right)$ is the target, which represents the actual reward of prediction. δ_t is the error, which is used to estimate the action value function. α is the learning rate.

Therefore, we follow the chain rule to derive $J_s(\mu)$, and get the update method of the actor network, as shown below:

$$\begin{aligned}\nabla_{\theta^{\mu_s}} J_s = & E_{\mathbf{o_s^t}, \mathbf{a_s^t} \sim \mathbf{D_s}} \left[\nabla_{\theta^{\mu_s}} \mu_{\theta_s}\left(\mathbf{o_s^t} | \theta^{\mu_s}\right)\right. \\ & \left. \cdot \nabla_{\mathbf{a_s^t}} Q_{\theta_s}\left(\mathbf{o_s^t}, \mathbf{a_s^t}, \mathbf{a_{S\backslash s}^t} | \theta^{Q_s}\right) |_{\mathbf{a_s^t} = \mu_{\theta_s}(\mathbf{o_s^t} | \theta^{\mu_s})}\right],\end{aligned} \tag{25}$$

where $\mathbf{D_s}$ is the experience memory of the agent s, given $\left\langle \mathbf{o_s^t}, \mathbf{o_{S\backslash s}^t}, \mathbf{a_s^t}, \mathbf{a_{S\backslash s}^t}, r_s^t, r_{S\backslash s}^t, \mathbf{o_s^{t+1}}, \mathbf{o_{S\backslash s}^{t+1}} \right\rangle$. Similarly, $\mathbf{o_{S\backslash s}^t}$, $r_{S\backslash s}^t$ and $\mathbf{o_{S\backslash s}^{t+1}}$ are the current state observation, reward space and the next state observation of the other two agents, respectively.

Using the gradient ascending algorithm, the parameter θ^{μ_s} in (25) is updated and the action $\mathbf{a_s^t}$ is optimized along the direction of improving the action value function $Q_{\theta_s}\left(\mathbf{o_s^t}, \mathbf{a_s^t}, \mathbf{a_{S\backslash s}^t} | \theta^{Q_s}\right)$.

Update of critic: According to (23), we update the critic by minimizing the loss function, which is defined as:

$$\begin{aligned}L_s = & E_{\mathbf{o_s^t}, \mathbf{a_s^t}, r_s^t, \mathbf{o_s^{t+1}} \sim \mathbf{D_s}} \\ & \left[\left(\text{Target}\, Q_s - Q_{\theta_s}\left(\mathbf{o_s^t}, \mathbf{a_s^t}, \mathbf{a_{S\backslash s}^t} | \theta^{Q_s}\right)\right)^2\right],\end{aligned} \tag{26}$$

and

$$\text{Target}\, Q_s = r_s^t + \gamma Q'_{\theta_s}\left(\mathbf{o_s^{t+1}}, \mu'_{\theta_s}\left(\mathbf{o_s^{t+1}} | \theta^{\mu'_s}\right) | \theta^{Q'_s}\right), \tag{27}$$

where $\theta^{\mu'_s}$ and $\theta^{Q'_s}$ in $\text{Target}\, Q_s$ represent the parameters of the target actor and the target critic, respectively.

Therefore, the gradient of the critic network is expressed as follows:

$$\begin{aligned}\nabla_{\theta^{Q_s}} L_s = & E_{\mathbf{o_s^t}, \mathbf{a_s^t}, r_s^t, \mathbf{o_s^{t+1}} \sim \mathbf{D_s}} [(\text{Target}\, Q_s - \\ & Q_{\theta_s}\left(\mathbf{o_s^t}, \mathbf{a_s^t}, \mathbf{a_{S\backslash s}^t} | \theta^{Q_s}\right)) \nabla_{\theta^{Q_s}} Q_{\theta_s}\left(\mathbf{o_s^t}, \mathbf{a_s^t}, \mathbf{a_{S\backslash s}^t} | \theta^{Q_s}\right)].\end{aligned} \tag{28}$$

We use the gradient descent method to update the parameter θ^{Q_s} and then find the optimal value of $Q^*_{\theta_s}$.

Online network and target network: Furthermore, in order to improve the stability of the learning process, the IMADDPG algorithm creates two DNNs for each actor and critic, respectively: online network and target network, as shown below.

$$\text{actor} \begin{cases} \text{online} : \mu_{\theta_s}\left(\mathbf{o_s^t} | \theta^{\mu_s}\right), \text{update}\ \theta^{\mu_s}, \\ \text{target} : \mu'_{\theta_s}\left(\mathbf{o_s^t} | \theta^{\mu'_s}\right), \text{update}\ \theta^{\mu'_s}, \end{cases} \tag{29}$$

and

$$\text{critic} \begin{cases} \text{online} : Q_{\theta_s}\left(\mathbf{o_s^t}, \mathbf{a_s^t}, \mathbf{a_{S\backslash s}^t} | \theta^{Q_s}\right), \text{update}\ \theta^{Q_s}, \\ \text{target} : Q'_{\theta_s}\left(\mathbf{o_s^t}, \mathbf{a_s^t}, \mathbf{a_{S\backslash s}^t} | \theta^{Q'_s}\right), \text{update}\ \theta^{Q'_s}. \end{cases} \tag{30}$$

That is, the IMADDPG algorithm uses a total of 12 DNNs ((1[online network] + 1[target network]) × 2[actor&critic] × 3[agents]) in our paper, as shown in Fig. 2. The update rule is that firstly agent s updates θ^{μ_s} and θ^{Q_s} of the online network through the gradient ascent or gradient descent algorithm after finishing a mini-batch data training, and then updates $\theta^{\mu'_s}$ and $\theta^{Q'_s}$ of the target network through the soft update method, as follows:

$$\begin{cases} \theta^{Q'_s} \leftarrow \tau \theta^{Q_s} + (1-\tau)\, \theta^{Q'_s}, \\ \theta^{\mu'_s} \leftarrow \tau \theta^{\mu_s} + (1-\tau)\, \theta^{\mu'_s}, \end{cases} \tag{31}$$

where the default value of τ is 0.001.

Optimizer (DNN configuration): As for the actor, we use a DNN with 3 inputs and $(3\times K)$ outputs, because we define 3 states and $(3\times K)$ actions for each agent. And as for the critic, we set up a DNN with $(3+9\times K)$ inputs and 1 output, because we need to provide state and global action information for the critic, and then the network generates Q-value of the current agent. However, there is no known rule for determining the number of hidden layers and neurons. It is proper to choose these sizes in the trial-and-error manner [38]. Therefore, we chose 2 hidden layers, each with 64 neurons for both the actor and critic. We set the relu activation function for the hidden layer and the sigmoid activation function for the output layer because the outputs are all positive.

B. The IMADDPG Algorithm

Using the conventional MADDPG algorithm, we may only eventually obtain one solution and we cannot guarantee that this solution is (near-)Pareto optimal due to the randomness of the exploration probability and the resources game between multiple agents. Hence, the IMADDPG algorithm is proposed by adding rank voting method in the testing process for obtaining near-Pareto optimal solutions. The IMADDPG algorithm could be divided into two main processes, *i.e.*, the training and testing processes.

The training process of IMADDPG algorithm: As shown in Fig. 2, the IMADDPG algorithm abides by the rule of MDP, and each agent includes an actor as well as a critic, in which the actor is responsible for the policy and the critic is for the action value. Furthermore, each actor or critic contains online and target networks, respectively. Three agents make the interaction with the environment in a parallel and asynchronous manner. Their four-tuples of $\left\langle \mathbf{o_s^t}, \mathbf{o_{S\backslash s}^t}, \mathbf{a_s^t}, \mathbf{a_{S\backslash s}^t}, r_s^t, r_{S\backslash s}^t, \mathbf{o_s^{t+1}}, \mathbf{o_{S\backslash s}^{t+1}} \right\rangle$ are generated and then stored in the experience memory of $\mathbf{D_s}$,$s \in \{1,2,3\}$, respectively. Next, we randomly sample D_{mini} mini-batches from $\mathbf{D_s}$ to the critics and actors for online off-policy training. The specific training process is summarized in Algorithm 1.

The testing process of IMADDPG algorithm: After completing the training process of IMADDPG algorithm, the DNN parameters used to approximate actors and critics (*i.e.*, θ^{μ_s}, θ^{Q_s}, $\theta^{\mu'_s}$ and $\theta^{Q'_s}$) will be stored, that is, the final trained data are saved. In this way, in the subsequent testing process, as long as the scale of the DRL model (*e.g.*, the input and output sizes of actors and critics) is consistent, the trained data can be directly invoked to run Algorithm 2 to approach Pareto optimal solutions, without running the training algorithm again.

Moreover, we add an ensemble strategy of rank voting method [39] in the testing process. Specifically, we use the

Algorithm 1 The Training Process of IMADDPG Algorithm.

Initialization:
Randomly initialize the parameters θ^{Q_s} and θ^{μ_s} of the critic $Q_{\theta_s}\left(\mathbf{o_s^t}, \mathbf{a_s^t}, \mathbf{a_{S\backslash s}^t}|\theta^{Q_s}\right)$ and actor $\mu_{\theta_s}\left(\mathbf{o_s^t}|\theta^{\mu_s}\right)$ for agent s.
Initialize the target network Q'_{θ_s} and μ'_{θ_s}, and set their parameters as $\theta^{Q'_s} \leftarrow \theta^{Q_s}$ and $\theta^{\mu'_s} \leftarrow \theta^{\mu_s}$.
Initialize the experience memory $\mathbf{D_s}$.
Set the number of users K_s in the current training process.
For episode = 1, ..., E **do**:
Initialize a random process $\mathbf{N_t}$ for action exploration.
Obtain the initial set of states $\mathbf{o_s^0}$.
For time step = 1, ..., T **do**:
As for agent $s \in \{1, 2, 3\}$, select action $\mathbf{a_s^t} = \mu_{\theta_s}\left(\mathbf{o_s^t}|\theta^{\mu_s}\right) + \mathbf{N_t}$ based on current policy and exploration in an asynchronous manner.
Perform actions $\mathbf{a_s^t}$ and record rewards r_s^t and new states $\mathbf{o_s^{t+1}}$.
In the experience memory $\mathbf{D_s}$, store four-tuples of $\left\langle \mathbf{o_s^t}, \mathbf{o_{S\backslash s}^t}, \mathbf{a_s^t}, \mathbf{a_{S\backslash s}^t}, r_s^t, r_{S\backslash s}^t, \mathbf{o_s^{t+1}}, \mathbf{o_{S\backslash s}^{t+1}} \right\rangle$.
Randomly sample D_{mini} mini-batches of $\left\langle \mathbf{o_s^j}, \mathbf{o_{S\backslash s}^j}, \mathbf{a_s^j}, \mathbf{a_{S\backslash s}^j}, r_s^j, r_{S\backslash s}^j, \mathbf{o_s^{j+1}}, \mathbf{o_{S\backslash s}^{j+1}} \right\rangle$ from $\mathbf{D_s}$.
Calculate Target Q_s by (27).
Update the critic by minimizing loss function of (28).
Update the actor using the gradient policy algorithm of (25).
Update parameters of the target network for each agent by (31).
$\mathbf{o_s^t} \leftarrow \mathbf{o_s^{t+1}}$.
End for
End for
Store the final trained data θ^{μ_s}, θ^{Q_s}, $\theta^{\mu'_s}$ and $\theta^{Q'_s}$ of DNNs.

Algorithm 2 The Testing Process of IMADDPG Algorithm.

Initialization:
Complete the training process of Algorithm 1 or directly invoke the final trained data of DNNs in Algorithm 1.
Initialize the experience memory $\mathbf{D'_s}$.
Given a testing state $\mathbf{o_s^0}$.
REPEAT:
Initialize a random process $\mathbf{N'_i}$ for action exploration.
As for agent s, select action $\mathbf{a_s^i} = \mu_{\theta_s}\left(\mathbf{o_s^i}|\theta^{\mu_s}\right) + \mathbf{N'_i}$ based on current policy and exploration in an asynchronous manner.
Perform actions $\mathbf{a_s^i}$ and record rewards r_s^i and new states $\mathbf{o_s^{i+1}}$.
In the experience memory $\mathbf{D'_s}$, store four-tuples of $\left\langle \mathbf{o_s^i}, \mathbf{o_{S\backslash s}^i}, \mathbf{a_s^i}, \mathbf{a_{S\backslash s}^i}, r_s^i, r_{S\backslash s}^i, \mathbf{o_s^{i+1}}, \mathbf{o_{S\backslash s}^{i+1}} \right\rangle$.
$\mathbf{o_s^i} \leftarrow \mathbf{o_s^{i+1}}$.
STOP when $\mathbf{D'_s}$ is full.
Sort four-tuples in the experience memory $\mathbf{D'_s}$ in the descending order according to the reward value, and assign a sequence number to each four-tuple and each agent by (32).
For each four-tuple, add the sequence numbers of three agents together by (33) to get the sum of sequence numbers.
By (34), the four-tuple with the smallest sum of sequence numbers is a near-Pareto optimal solution.

trained network model from Algorithm 1 and introduce the new exploration probability $\mathbf{N'_i}$ for action selection in the testing process. After that, we store the generated four-tuples $\left\langle \mathbf{o_s^i}, \mathbf{o_{S\backslash s}^i}, \mathbf{a_s^i}, \mathbf{a_{S\backslash s}^i}, r_s^i, r_{S\backslash s}^i, \mathbf{o_s^{i+1}}, \mathbf{o_{S\backslash s}^{i+1}} \right\rangle$ in the new experience memory $\mathbf{D'_s}$, which is used for collecting potential Pareto optimal solutions. Then, the four-tuples in $\mathbf{D'_s}$ are ranked in the descending order according to the reward value of agent s. A sequence number $\xi_\mathbf{s}^\mathbf{i}$ is assigned to four-tuple i and agent s, as follows:

$$\xi_\mathbf{s}^\mathbf{i} = \mathbf{desc}\left\langle \mathbf{o_s^i}, \mathbf{o_{S\backslash s}^i}, \mathbf{a_s^i}, \mathbf{a_{S\backslash s}^i}, r_s^i, r_{S\backslash s}^i, \mathbf{o_s^{i+1}}, \mathbf{o_{S\backslash s}^{i+1}} \right\rangle, \quad (32)$$

where **desc** $\langle\cdot\rangle$ is the function enabling to rank its elements in the descending order. Finally, we add the sequence numbers of three agents for each four-tuple, and select the optimal four-tuple with the smallest sum of sequence numbers, that is:

$$\xi^\mathbf{i} = \xi_1^\mathbf{i} + \xi_2^\mathbf{i} + \xi_3^\mathbf{i}, \quad (33)$$

and

$$\langle \mathbf{o_1^*}, \mathbf{o_2^*}, \mathbf{o_3^*}, \mathbf{a_1^*}, \mathbf{a_2^*}, \mathbf{a_3^*}, r_1^*, r_2^*, r_3^* \rangle = \arg\min_i \xi^\mathbf{i}. \quad (34)$$

When the space of $\mathbf{D'_s}$ is large enough, we prove that the optimal four-tuple of $\langle \mathbf{o_1^*}, \mathbf{o_2^*}, \mathbf{o_3^*}, \mathbf{a_1^*}, \mathbf{a_2^*}, \mathbf{a_3^*}, r_1^*, r_2^*, r_3^* \rangle$ is a near-Pareto optimal solution in APPENDIX A. Furthermore, the testing process is summarized in Algorithm 2. We can harvest multiple near-Pareto optimal solutions so as to approximate the Pareto boundary by iterating Algorithm 2 for many times.

V. Simulation Results and Discussions

In this section, we provide simulation results to verify the theoretical analysis and compare the performance with other benchmark algorithms. Suppose the fixed size of the experience memory $\mathbf{D_s}$ in Algorithm 1 is 2000 four-tuples, and we sample D_{mini}=32 mini-batches for training at each time step. In algorithm 2, the capacity of the experience memory $\mathbf{D'_s}$ is 5000 four-tuples. The contents of $\mathbf{D_s}$ and $\mathbf{D'_s}$ are stored according to the first in first out (FIFO) principle. The total available power of a physical BS is $P_B = 40$W and the available bandwidth in the system is $B = 10$MHz. The other parameters are given as follows: $N = 8$, $M = 2$, $\delta = 10$ms/slot, $\beta = 0.2$s, $\lambda = (500 \times K_2)$kbps.

In order to reflect the advantages of our proposed schemes, we give three benchmark algorithms and schemes for comparison. Firstly, we compare the IMADDPG algorithm with the MADDPG algorithm. Secondly, referring to [31], we take the traditional scalar method of solving MOOP as the comparison scheme. From the perspective of MNO, we convert the key performance metrics of three slices into the utility through a weighted sum method. Therefore, the network-wide utility can

be defined as:

$$U(t) = \omega_1 R_1^{sum}(t) - \omega_2 Q_2^{ave}(t) + \omega_3 SINR_3^{ave}(t), \quad (35)$$

where ω_1, ω_2 and ω_3 are the unit price of throughput gain, delay gain and SINR gain charged by MNO, respectively. Significantly, the throughput, average delay and average SINR of three slices are highly dependent on their weights of ω_1, ω_2 and ω_3, respectively, which are usually defined through experience. In other words, optimizing utility may not satisfy users' QoS on each slice class. Therefore, we give three different-weight utility schemes as three comparison schemes. On the basis of the normalization of the throughput, average delay and average SINR among three slices, we adjust their weights ratio to 1:1:1, 2:1:2 and 1:2:1, namely utility(1:1:1), utility(2:1:2) and utility(1:2:1) schemes, respectively, and we use the DDPG algorithm to solve them. Finally, we give the equal resource allocation algorithm among three slices [40] to compare with our proposed dynamic resources allocation algorithm. In this algorithm, no matter how the channel states and the traffic load change, all subchannels and BSs' power are always evenly distributed to users of each slice class.

Fig. 3 shows the convergence performance of the IMADDPG algorithm during the training process. Among them, rewards 1, 2 and 3 represent the three reward functions defined in (18). We use a total of 20,000 iterations of 20 episodes multiplying 1000 time steps. It can be seen that each of rewards 1, 2 and 3 converges to a relatively stable state after about 8000 training iterations, respectively. There are still some small fluctuations mainly due to the addition of exploration probability $\mathbf{N_t}$ when selecting actions in Algorithm 1 and due to the resource game among the three slices, which are eliminated in the testing process. In addition, the number of users on each slice class also affects the value of rewards. As the number of users increases, reward 1 increases while reward 2 and 3 decrease, which corresponds to the increase of throughput on class-1 slices and average delay on class-2 slices as well as the decrease of average SINR on class-3 slices, respectively.

Fig. 4 characterizes the throughput versus the number of users K_1 of class-1 slices for the proposed scheme and other comparison schemes. For guaranteeing the accuracy of all the results, we perform 10 experiment trials for the same hyperparameter configuration, and show the bootstrap mean and 95% confidence bounds [41]. Note that in order to facilitate the result comparison, the values of performance metrics shown in this section are time-averaged, *i.e.*, $\frac{1}{T}\sum_{t=1}^{T} R_1^{sum}(t)$, $\frac{1}{T}\sum_{t=1}^{T} Q_2^{ave}(t)$ and $\frac{1}{T}\sum_{t=1}^{T} SINR_3^{ave}(t)$. It can be seen that with the increase of K_1, the throughput of all schemes continue to increase. When K_1 exceeds 5, the throughput of our scheme, utility(1:2:1) scheme, the MADDPG and the equal resource allocation algorithm tend to be saturated, because it exists the resource contention among three classes of slices at this moment. Compared with the equal resource allocation and the MADDPG algorithm, our IMADDPG algorithm has better performance, as the IMADDPG algorithm dynamically shares and releases resources, and circumvents the fluctuation

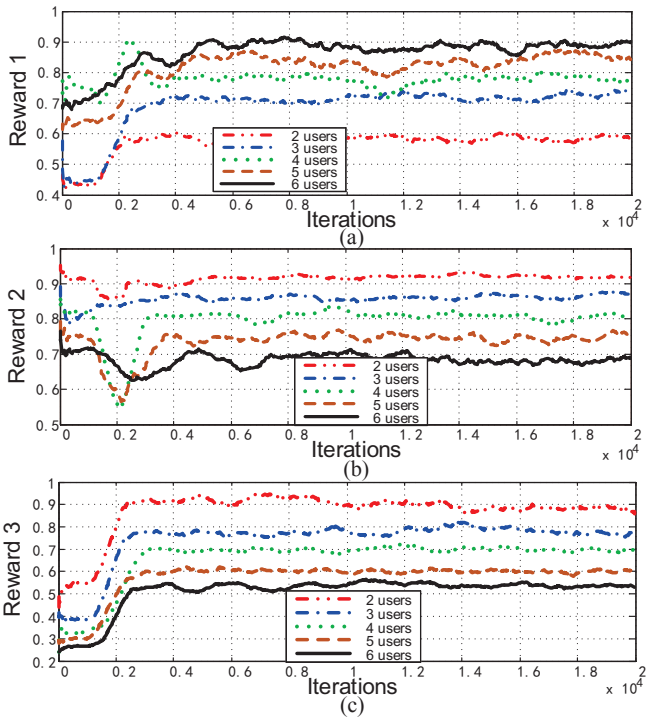


Fig. 3: Convergence performance of IMADDPG algorithm.

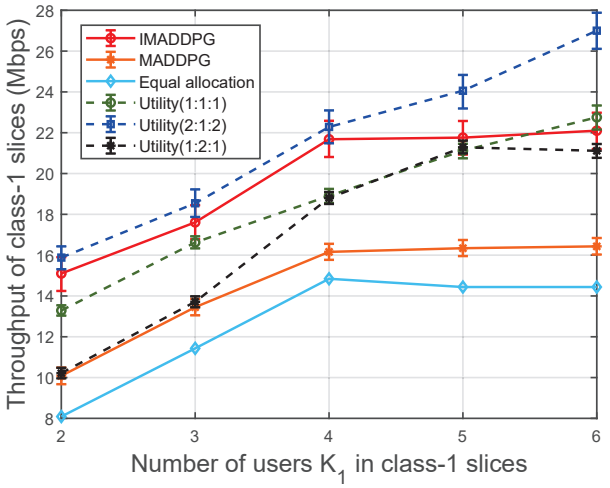


Fig. 4: Throughput versus the number of users K_1 of class-1 slices.

problem of the MADDPG algorithm, respectively. Among the three utility schemes, the larger the weight assigned to throughput, the higher the obtained throughput. Therefore, the utility(2:1:2) scheme gets a better throughput performance. Moreover, we can see that the variation of 95% confidence intervals is relatively small, which has little impact on the performance comparison between the proposed and the benchmark schemes.

Fig. 5 and Fig. 6 depict the average delay and average SINR versus the number of users K_2 of class-2 slices and users K_3 of class-3 slices for all schemes, respectively. Similarly, compared with the MADDPG and the equal resource allocation algorithm, the proposed scheme has better performance. Among the three utility schemes of Fig. 4, Fig. 5 and Fig. 6, the utility(1:2:1) scheme has the lowest average delay, but the

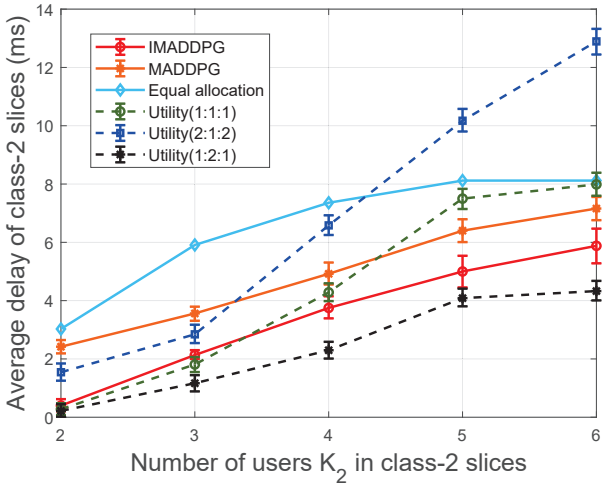


Fig. 5: Average delay versus the number of users K_2 of class-2 slices.

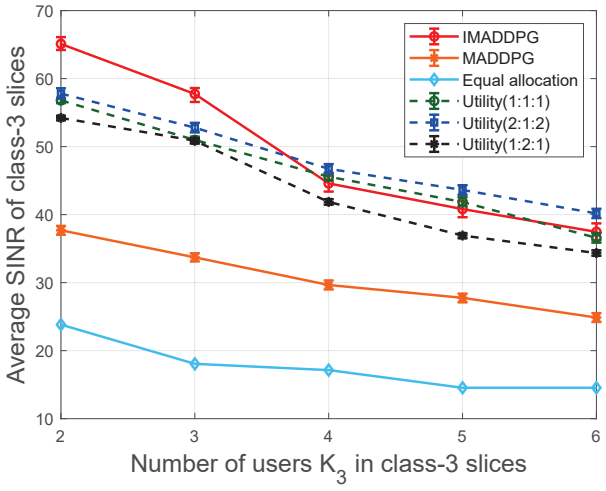


Fig. 6: Average SINR versus the number of users K_3 of class-3 slices.

throughput and average SINR of this scheme are poor. On the contrary, the SINR advantage of the utility(1:1:1) scheme and the utility(2:1:2) scheme is more obvious, but in exchange, the delay of both is high. Therefore, the metric of each slice class in the utility scheme is highly dependent on its weight, which is subjectively designed in advance. However, our scheme is universal among different types of RAN slices without designing the utility function and the weights in advance. It can obtain the near-Pareto optimal solutions, and these solutions are not dominated by each other. Meanwhile, it could be seen by jointly considering Fig. 4, Fig. 5 and Fig. 6 that although the IMADDPG scheme does not achieve the best result for all slices, its overall performance achieves the best tradeoff between the three classes of slices considered.

Fig. 7 describes the average delay versus the data arrival rate λ of class-2 slices for all schemes when $K_2 = 5$. With the increase of λ, the average delay of each scheme continues to increase. We can always obtain lower average delay by the IMADDPG algorithm than that by the MADDPG and equal resource allocation algorithm. Obviously, the delay of

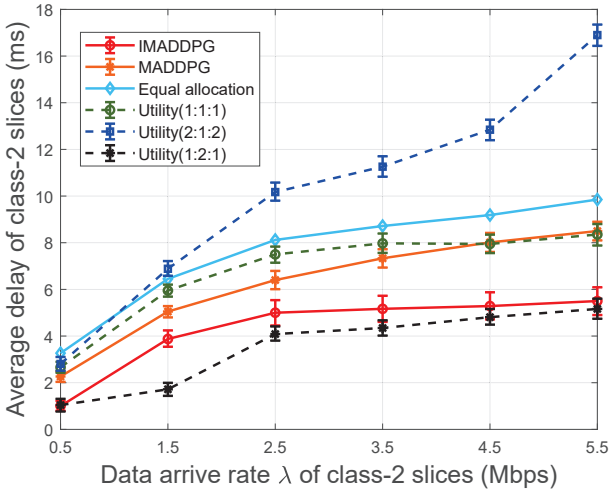
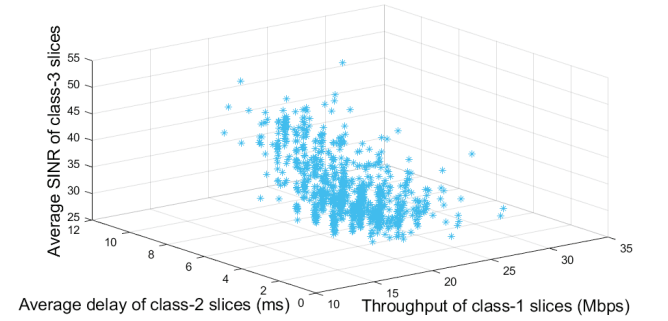


Fig. 7: Average delay versus the data arrival rate λ of class-2 slices.

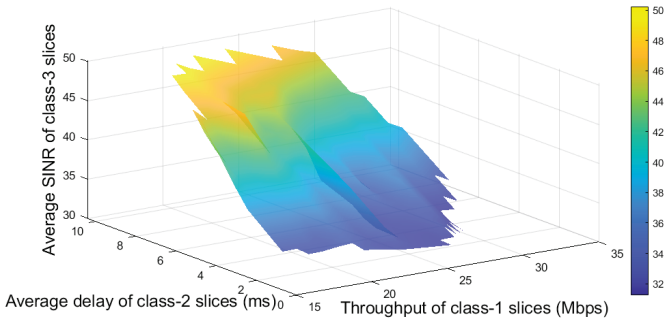
our proposed scheme and the utility(1:2:1) scheme are stable within 6 ms, even if the data arrival rate of class-2 slices increases to more than 5 Mbps. At the same time, when the traffic load is too high, the performance of the utility(2:1:2) scheme is poor, even exceeding 10 ms.

Fig. 8(a) is the three-dimensional scatter diagram of near-Pareto optimal solutions for the three-objective optimization problem of (14), which is obtained by iterating Algorithm 2 for 1000 times when the number of users per slice class is 5, and Fig. 8(b) is the schematic diagram of near-Pareto boundary by using the interpolation method. Since all points on the near-Pareto boundary are not dominated by each other, the improvement of the metric on one slice will be accompanied by the degradation of the metrics on the other one or two slices. That is, throughput, average delay and average SINR cannot be improved at the same time, indicating their tradeoff. For example, when the average delay of class-2 slices is less than 5ms, the average SINR of class-3 slices will also be reduced to below 35, and the throughput of class-1 slices will not exceed 30Mbps. On the near-Pareto boundary, the MNO can actually execute one of the near-Pareto optimal solutions according to the priority of slices, where each solution can correspond to an explicit resource allocation scheme. For instance, if the low-delay slice has a higher priority, the MNO can select a near-Pareto optimal solution that has the lowest delay with guaranteeing basic requirements of the high-bandwidth slice and wide-coverage slice.

Without loss of generality, we also show schematic diagrams of the near-Pareto boundary when the number of users per slice is 3 and 6, respectively in Fig. 9. Similarly, we can observe their mutual non-dominance. For example, in Fig. 9(a), when the average SINR of class-3 slices exceeds 58.5, the throughput of class-1 slices will not exceed 20Mbps, and the average delay of class-2 slices will be greater than 2ms. In Fig. 9(b), when the throughput of class-1 slices is greater than 28Mbps, the average delay of class-2 slices will exceed 4ms, and the average SINR of class-3 slices will be limited to 38. Overall, our proposal characterizes a compelling tradeoff among three types of RAN slices.

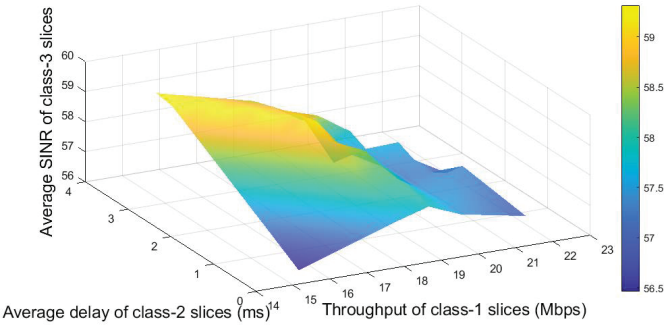


(a)

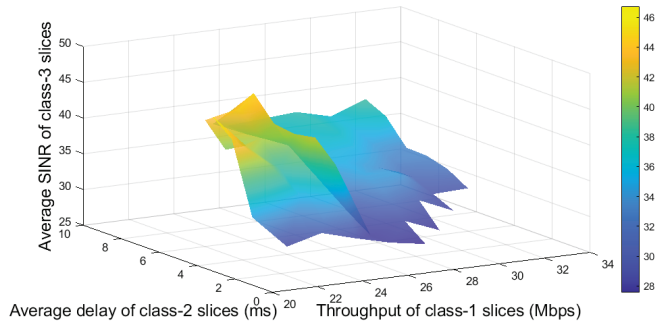


(b)

Fig. 8: (a) Near-Pareto optimal scatter diagram; (b) Near-Pareto boundary ($K_1 = K_2 = K_3 = 5$).



(a)



(b)

Fig. 9: (a) Near-Pareto boundary ($K_1 = K_2 = K_3 = 3$); (b) Near-Pareto boundary ($K_1 = K_2 = K_3 = 6$).

VI. Conclusion

In this paper, we proposed an effective multi-agent DRL algorithm to jointly optimize three kinds of RAN slices with different objectives. Specifically, the high-bandwidth slice, the low-delay slice and the wide-coverage slice were supported simultaneously under the same physical network. A non-scalar MOOP was formulated by dynamically deploying proper vBSs and allocating subchannels and power to users of each slice, in order to jointly optimize the throughput, average delay and average SINR. We presented an IMADDPG model structure and proposed an online off-policy IMADDPG algorithm to solve the above MOOP, where three parallel distributed AC structures were employed for centralized training and distributed execution. By adding the rank voting in the testing process, we aimed to obtain the near-Pareto optimal solutions. Simulation results verified that we achieved the near-Pareto optimum of three classes of slices, and the proposed method outperformed available algorithms including the traditional scalar method, the equal resource allocation algorithm and MADDPG algorithm. Furthermore, the proposed method approximated multiple Pareto optimal solutions, while the traditional scalar method only subjectively approached one of the Pareto optimal solutions. We also gave some guidance on how to choose Pareto optimal solutions in practical application, *e.g.*, MNO could execute one of them according to the priority of slices. The non-dominance between Pareto optimal solutions indicates the improvement of one slice always leads to the degradation of other slices, characterizing the tradeoff between different RAN slices.

Appendix A
Proof of Near-Pareto Optimal Solution

In the case of the infinite space of $\mathbf{D}'_\mathbf{s}$ completely containing all potential Pareto optimal solutions, we first use the proof by contradiction to verify that $\langle \mathbf{o}_1^*, \mathbf{o}_2^*, \mathbf{o}_3^*, \mathbf{a}_1^*, \mathbf{a}_2^*, \mathbf{a}_3^*, r_1^*, r_2^*, r_3^* \rangle$ in $\mathbf{D}'_\mathbf{s}$ is one of the Pareto optimal solutions.

Specifically, we assume that the optimal four-tuple of $\langle \mathbf{o}_1^*, \mathbf{o}_2^*, \mathbf{o}_3^*, \mathbf{a}_1^*, \mathbf{a}_2^*, \mathbf{a}_3^*, r_1^*, r_2^*, r_3^* \rangle$ is not a Pareto optimal solution. According to *Definition 1*, there must be another solution of $\langle \mathbf{o}'_\mathbf{1}, \mathbf{o}'_\mathbf{2}, \mathbf{o}'_\mathbf{3}, \mathbf{a}'_\mathbf{1}, \mathbf{a}'_\mathbf{2}, \mathbf{a}'_\mathbf{3}, r'_1, r'_2, r'_3 \rangle$ with $r'_s \geq r^*_s, s = 1, 2, 3$. In this way, the sequence number $\xi'_\mathbf{s}$ of r'_s must be smaller than the sequence number $\xi^*_\mathbf{s}$ of r^*_s, and then the sum of three agents' sequence numbers ξ' must be smaller than the sum of three agents' sequence numbers ξ^*. However, we have known ξ^* is the smallest sum of sequence numbers. That is, there is the contradiction between ξ^* and ξ', which indicates that the assumption is not true. Therefore, $\langle \mathbf{o}_1^*, \mathbf{o}_2^*, \mathbf{o}_3^*, \mathbf{a}_1^*, \mathbf{a}_2^*, \mathbf{a}_3^*, r_1^*, r_2^*, r_3^* \rangle$ is a Pareto optimal solution when the space of $\mathbf{D}'_\mathbf{s}$ is infinite.

Nevertheless, it is obvious that we cannot give an infinite experience memory of $\mathbf{D}'_\mathbf{s}$ in actual situations. Therefore, when the storage space is limited but large enough, the optimal four-tuple of $\langle \mathbf{o}_1^*, \mathbf{o}_2^*, \mathbf{o}_3^*, \mathbf{a}_1^*, \mathbf{a}_2^*, \mathbf{a}_3^*, r_1^*, r_2^*, r_3^* \rangle$ in $\mathbf{D}'_\mathbf{s}$ is the near-Pareto optimal [21], [22].

REFERENCES

[1] 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, thirdquarter 2018.

[2] B. Han, V. Sciancalepore, X. Costa-Prez, D. Feng and H. D. Schotten, "Multiservice-Based Network Slicing Orchestration With Impatient Tenants," *IEEE Transactions on Wireless Communications*, vol. 19, no. 7, pp. 5010-5024, July 2020.

[3] R. Ferrus, O. Sallent, J. Perez-Romero and R. Agusti, "On 5G Radio Access Network Slicing: Radio Interface Protocol Features and Configuration," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 184-192, May 2018.

[4] H. Xiang, S. Yan and M. Peng, "A Realization of Fog-RAN Slicing via Deep Reinforcement Learning," *IEEE Transactions on Wireless Communications*, vol. 19, no. 4, pp. 2515-2527, April 2020.

[5] Y. L. Lee, J. Loo, T. C. Chuah and L. C. Wang, "Dynamic Network Slicing for Multitenant Heterogeneous Cloud Radio Access Networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 4, pp. 2146-2161, April 2018.

[6] H. Xiang, M. Peng, Y. Sun and S. Yan,"Mode Selection and Resource Allocation in Sliced Fog Radio Access Networks: A Reinforcement Learning Approach," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4271-4284, April 2020.

[7] L. Tang, X. Zhang, H. Xiang, Y. Sun and M. Peng, "Joint resource allocation and caching placement for network slicing in fog radio access networks," 2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), Sapporo, 2017, pp. 1-6.

[8] C. Song *et al.*, "Hierarchical edge cloud enabling network slicing for 5G optical fronthaul," *Journal of Optical Communications and Networking*, vol. 11, no. 4, pp. B60-B70, April 2019.

[9] X. Shen *et al.*, "AI-Assisted Network-Slicing Based Next-Generation Wireless Networks," *IEEE Open Journal of Vehicular Technology*, vol. 1, pp. 45-66, 2020.

[10] I. Kovacevic, A. S. Shafigh, S. Glisic, B. Lorenzo and E. Hossain, "Multi-Domain Network Slicing With Latency Equalization," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2182-2196, Dec. 2020.

[11] S. Dawaliby, A. Bradai and Y. Pousset, "Distributed Network Slicing in Large Scale IoT Based on Coalitional Multi-Game Theory," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1567-1580, Dec. 2019.

[12] G. Wang, G. Feng, T. Q. S. Quek, S. Qin, R. Wen and W. Tan, "Reconfiguration in Network Slicing-Optimizing the Profit and Performance," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 591-605, June 2019.

[13] B. Han, V. Sciancalepore, X. Costa-Prez, D. Feng and H. D. Schotten, "Multiservice-Based Network Slicing Orchestration With Impatient Tenants," *IEEE Transactions on Wireless Communications*, vol. 19, no. 7, pp. 5010-5024, July 2020.

[14] Z. L. Lyu, X. Q. Wang and Z. X. Yang, "Optimal dynamic dispatch of pareto frontier for microgrid based on MOIBBO algorithm," 2017 IEEE Power & Energy Society General Meeting, Chicago, IL, 2017, pp. 1-5.

[15] E. Bjornson, E. Jorswieck, M. Debbah, *et al.*, "Multi-Objective Signal Processing Optimization: The Way to Balance Conflicting Metrics in 5G Systems," *IEEE Signal Processing Magazine,* vol. 31, no. 6, pp. 14-23, 2014.

[16] Q. Shi, L. Zhao, Y. Zhang, G. Zheng, F. R. Yu and H. Chen, "Energy-Efficiency Versus Delay Tradeoff in Wireless Networks Virtualization," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 837-841, Jan. 2018.

[17] 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, Feb. 2020.

[18] I. Afolabi, J. Prados-Garzon, M. Bagaa, T. Taleb and P. Ameigeiras, "Dynamic Resource Provisioning of a Scalable E2E Network Slicing Orchestration System," *IEEE Transactions on Mobile Computing*, vol. 19, no. 11, pp. 2594-2608, 1 Nov. 2020.

[19] Q. Xu, J. Wang and K. Wu, "Learning-Based Dynamic Resource Provisioning for Network Slicing with Ensured End-to-End Performance Bound," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 1, pp. 28-41, 1 Jan.-March 2020.

[20] W. Guan, X. Wen, L. Wang, Z. Lu and Y. Shen, "A Service-Oriented Deployment Policy of End-to-End Network Slicing Based on Complex Network Theory," *IEEE Access*, vol. 6, pp. 19691-19701, 2018.

[21] J. Cui, S. X. Ng, D. Liu, J. Zhang, A. Nallanathan and L. Hanzo, "Multiobjective Optimization for Integrated Ground-Air-Space Networks: Current Research and Future Challenges," *IEEE Vehicular Technology Magazine*, vol. 16, no. 3, pp. 88-98, Sept. 2021.

[22] J. Cui, H. Yetgin, D. Liu, J. Zhang, S. X. Ng and L. Hanzo, "Twin-Component Near-Pareto Routing Optimization for AANETs in the North-Atlantic Region Relying on Real Flight Statistics," *IEEE Open Journal of Vehicular Technology*, vol. 2, pp. 346-364, 2021.

[23] Z. Du, Y. Deng, W. Guo, A. Nallanathan and Q. Wu, "Green Deep Reinforcement Learning for Radio Resource Management: Architecture, Algorithm Compression, and Challenges," *IEEE Vehicular Technology Magazine*, vol. 16, no. 1, pp. 29-39, March 2021.

[24] J. Wang, C. Jiang, H. Zhang, Y. Ren, K. -C. Chen and L. Hanzo, "Thirty Years of Machine Learning: The Road to Pareto-Optimal Wireless Networks," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1472-1514, thirdquarter 2020.

[25] Z. Fei, B. Li, S. Yang, C. Xing, H. Chen and L. Hanzo, "A Survey of Multi-Objective Optimization in Wireless Sensor Networks: Metrics, Algorithms, and Open Problems," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 550-586, Firstquarter 2017.

[26] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, pp. 6379-6390, 2017.

[27] X. Li, K. Jiao, F. Jiang, J. Wang and M. Pan, "A Service-Oriented Spectrum-Aware RAN-Slicing Trading Scheme Under Spectrum Sharing," *IEEE Internet of Things Journal*, vol. 7, no. 11, pp. 11303-11317, Nov. 2020.

[28] D. Wu, Z. Zhang, S. Wu, J. Yang and R. Wang, "Biologically Inspired Resource Allocation for Network Slices in 5G-Enabled Internet of Things," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 9266-9279, Dec. 2019.

[29] P. Yang, X. Xi, T. Q. S. Quek, J. Chen, X. Cao and D. Wu, "RAN Slicing for Massive IoT and Bursty URLLC Service Multiplexing: Analysis and Optimization," *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 14258-14275, 15 Sept.15, 2021.

[30] 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, May 2017.

[31] G. Zhou, L. Zhao, K. Liang, G. Zheng and L. Hanzo, "Utility Analysis of Radio Access Network Slicing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 1, pp. 1163-1167, Jan. 2020.

[32] X. Chen, *et al.*, "Multi-Tenant Cross-Slice Resource Orchestration: A Deep Reinforcement Learning Approach," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2377-2392, Oct. 2019.

[33] D. P. Bertsekas and R. G. Gallager, *Data Networks (2nd edition)*. Prentice Hall, 1992.

[34] W. Wang, V. K. N. Lau and M. Peng, "Delay-Aware Uplink Fronthaul Allocation in Cloud Radio Access Networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 7, pp. 4275-4287, July 2017.

[35] Y. Cao, S. -Y. Lien, Y. -C. Liang, K. -C. Chen and X. Shen, "User Access Control in Open Radio Access Networks: A Federated Deep Reinforcement Learning Approach," *IEEE Transactions on Wireless Communications*, doi: 10.1109/TWC.2021.3123500.

[36] R. Li *et al.*,"Deep Reinforcement Learning for Resource Management in Network Slicing," *IEEE Access*, vol. 6, pp. 74429-74441, 2018.

[37] D. Silver, J. Schrittwieser, K. Simonyan, *et al.*, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, pp. 354-359, Ooc. 2017.

[38] G. Sun, Z. T. Gebrekidan, G. O. Boateng, D. Ayepah-Mensah and W. Jiang, "Dynamic Reservation and Deep Reinforcement Learning Based Autonomous Resource Slicing for Virtualized Radio Access Networks," *IEEE Access*, vol. 7, pp. 45758-45772, 2019.

[39] T. Brys, *et al.*, "Multi-objectivization and Ensembles of Shapings in Reinforcement Learning," *Neurocomputing*, vol. 263, pp. 48-59, nov. 2017.

[40] Y. L. Lee, J. Loo, T. C. Chuah and L. Wang, "Dynamic Network Slicing for Multitenant Heterogeneous Cloud Radio Access Networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 4, pp. 2146-2161, April 2018.

[41] Peter Henderson, *et al.*, "Deep reinforcement learning that matters." *Proceedings of the AAAI conference on artificial intelligence*, vol. 32. no. 1, 2018.