Meta-Scheduling Framework With Cooperative Learning Toward Beyond 5G

Kyungsik Min, Yunjoo Kim, and Hyun-Suk Lee

Abstract— **In this paper, we propose a novel meta-scheduling framework with cooperative learning that fully exploits a functional split structure of the base station (BS) consisting of a central unit (CU) and digital units (DUs). To this end, we first design a meta-scheduling policy structure to find a scheduling strategy that can be applied to any BS regardless of BS-specific characteristics such as the number of users, various quality-of-service requirements, and the number of resource blocks. In the proposed framework, the CU only needs to manage objective-specific meta-scheduling policies in a centralized manner while each DU performs scheduling in a decentralized manner by simply using the policy at the CU without any responsibility to manage or train policies. Besides, the policies at the CU can be cooperatively trained using the experiences obtained from all DUs. Therefore, the proposed framework not only provides computational efficiency but also supports network scalability. We show via experiments that the proposed meta-scheduling framework achieves competitive performances compared with the near-optimal conventional schedulers tailored to each BS even though it manages and exploits only one meta-scheduling policy for each objective. Furthermore, our meta-scheduling framework effectively adapts to the change of BS-specific characteristics.**

Index Terms— **Cellular networks, cooperative learning, deep reinforcement learning, meta-learning, meta-scheduling.**

I. Introduction

FIFTH-GENERATION (5G) networks have been launched starting from Korea and the United States in December 2018. Inspired by the successful commercialization and mass deployment of 5G networks in Korea and the United States, other countries are rushing to provide 5G network services. 5G networks aim to provide high data rate, ultra-reliable low latency, and massive connectivity [1]. To support such key performances of 5G, new radio (NR) standard introduced a network architecture that splits a base station (BS) into a central unit (CU), digital unit (DU), and radio unit (RU) [2]. CU-DU split structure allows service providers to construct both service-oriented network architecture such as network slicing and cost-effective network deployment considering the capacity and availability of the optical interface. Moreover, network equipment manufacturers can enable flexible hardware and software implementations tailored to the targeted markets and customers.

In a typical centralized radio access network (RAN), hundreds of DUs are orchestrated by one CU. Considering mass connection and wide deployment of the DUs, the whole network covered by the CU has various environments (for example, line-of-sight (LOS) and non-LOS (NLOS) channels) and service objectives (for example, high data rate for dense urban areas and large coverage requirement for rural areas). Such diverse environments and objectives disturb inheriting traditional mathematical model-based schedulers designed with specific requirements and corresponding customized models [3], [4].

As a viable solution to overcome such a limitation of the traditional schedulers, data-driven scheduling based on artificial intelligence (AI) and machine learning (ML) has emerged [3]. The data-driven scheduler learns a scheduling strategy using observed data from cellular networks instead of designing a scheduler based on mathematical models. In particular, deep reinforcement learning (DRL)-based schedulers have been widely studied [4], [5], [6], [7], [8], [9], [10], [11], [12], [13] since DRL is one of the representative AI/ML methods for complex decision-making problems. In [4], [5], and [6], DRL-based schedulers are studied for a single BS. On the other hand, in [7], [8], [9], and [10], DRL-based schedulers are studied to address a scheduling problem for multiple BSs in a centralized manner. However, centralized scheduling for multiple BSs is too complex for the CU to be implemented in practical networks [11].

To mitigate the complexity issue, decentralized approaches in which each BS or wireless link determines its own scheduling decisions are proposed in [11], [12], and [13]. However, they lack a generalization capability over different BS-specific characteristics such as the number of resource blocks (RBs), QoS requirements, and the number of users [6]. Specifically, a scheduler trained in given BS-specific characteristics is not usable for other BS-specific characteristics. This also implies that a scheduler trained in one BS cannot be used in another BS unless the BSs have identical characteristics. Furthermore,

Manuscript received 25 October 2022; revised 10 February 2023; accepted 7 April 2023. Date of publication 8 May 2023; date of current version 19 June 2023. This work was supported by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by the Korean Government (MSIT), Development of Intelligent Wireless Access Technologies, under Grant 2021-0-00972. *(Corresponding author: Hyun-Suk Lee.)*

Kyungsik Min is with the Department of Information and Telecommunications Engineering, The University of Suwon, Hwaseong 18323, South Korea (e-mail: kyungsik@suwon.ac.kr).

Yunjoo Kim is with the Division of Mobile Communication Research, Electronics and Telecommunications Research Institute (ETRI), Daejeon 34129, South Korea (e-mail: yunjoo@etri.re.kr).

Hyun-Suk Lee is with the Department of Intelligent Mechatronics Engineering, Sejong University, Seoul 05006, South Korea (e-mail: hyunsuk@sejong.ac.kr).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2023.3273698.

Digital Object Identifier 10.1109/JSAC.2023.3273698

the schedulers designed for a single BS in [4], [5], and [6] have the same drawback. Such an unsharable scheduler causes inevitable responsibilities for DUs to learn and manage all the schedulers for different characteristics, which makes it difficult to fully exploit the advantages of the CU-DU split structure.

The concept of meta-learning, also known as "learning to learn", was established to empower AI/ML methods to have such a generalization capability. In specific, meta-learning enables a deep learning model to learn new tasks much faster by using previous experiences as a basis [14]. In the same context, meta-RL has been widely studied as well so as to enable an agent to efficiently address new tasks or unseen environments. To this end, meta-RL methods find common neural network parameters that can quickly adapt to new tasks [15], infer tasks over a latent common context [16], [17], translate policy structures in different environments into a latent policy structure [18], [19], redesign policy neural network architecture [20], [21], [22], [23], control exploration strategies [24], and augment state or reward [25], [26].

Most of the meta-RL methods in [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], and [26] commonly focus on the family of problems, which share the same dynamics and environment but differ in the task specified by the reward function. However, contrary to those problems, the target scheduling problems in cellular networks share an identical task (i.e., scheduling) and have different dynamics and environments due to the different network BS characteristics. Hence, only a limited number of researches have been studied to apply meta-RL to develop *meta-RL schedulers* for cellular networks. In [27], a dynamic resource allocation algorithm for vehicular networks is proposed based on the meta-hierarchical RL framework in [22]. Specifically, the algorithm can quickly adapt to a new environment by learning the low-level subnetwork on the basis of the master network. In [28], [29], and [30], resource allocation algorithms are developed by using the meta-learning method in [15]. The resource allocation algorithms proposed in [28], [29], and [30] identify common neural network parameters that are effective for fast adaptation to a new environment. Then, by using the parameters as initialization parameters, the proposed algorithms can quickly learn a scheduling policy for the new environment only using a small number of experiences.

However, in a view of a scheduling framework, finding new policies quickly implies that an additional learning phase of the schedulers is required for every change of BS-specific characteristics. Then, DUs have additional responsibilities to learn and manage all the policies for different BS-specific characteristics same as in the DRL-based schedulers, which is contrary to the purpose of the CU-DU split structure pursuing cost-effective network deployment. Besides, the related works on scheduling frameworks do not consider practical issues in cellular networks as shown in Table I. In particular, the scheduling frameworks with meta-RL schedulers in [28], [29], and [30] consider resource allocation only for a single BS. On the other hand, the meta-RL scheduler considering multiple RBs is proposed with centralized manner in [27], which is still too complex for the CU.

Furthermore, in practical networks, BSs may have different scheduling objectives; some BSs focus on maximizing sum-rate, while other BSs focus on satisfying QoS requirements. Moreover, the BSs may have various QoS requirements such as not only minimum data rate but also latency. However, most related works on DRL-based scheduling frameworks typically consider an identical objective for all BSs [7], [8], [9], [10], [11], [12], [13], [27]. In addition, the QoS requirements are only considered in the limited number of related works. Therefore, to realize an authentic meta-scheduling framework for practical cellular networks, it is required not only to design a scheduler that does not depend on any BS-specific characteristics but also to address diverse objectives in a single framework. Moreover, to follow the purpose of the CU-DU split structure, it should avoid the computational burden on the DUs as much as possible. However, to the best of our knowledge, there is no work on a meta-RL-based scheduling framework that comprehensively considers practical cellular networks.

In this paper, we study a novel meta-scheduling framework for 5G and beyond-5G networks. We incorporate a concept of cooperative learning into the framework to fully exploit the CU-DU split structure and the generalization capability of meta-scheduling together. Specifically, in the proposed framework, the CU manages meta-scheduling policies for diverse objectives and QoS requirements in a centralized manner, whereas each DU performs scheduling in a decentralized manner simply using the policy at the CU. Furthermore, the policies at the CU are cooperatively trained by using the experiences obtained from different DUs. The proposed framework not only provides computational efficiency but also ensures network scalability. We summarize the comparison of our work with the related works in Table I.

The contributions of the paper are summarized as follows:

- We propose a meta-scheduling framework for diverse objectives in which the CU only needs to manage as many meta-scheduling policies as the number of objectives in the network while each DU has no responsibility to manage or train scheduling policies.
- As an enabler of the framework, we design a meta-scheduling policy structure inspired by [19] that can be applied to any BS-specific characteristics rather than tailored to certain BS-specific characteristics. We also develop a meta-scheduling procedure to learn and utilize the meta-scheduling policy in practical cellular networks.
- In the proposed meta-scheduling framework with cooperative learning, the larger numbers of BSs and RBs do not increase scheduling complexity significantly. Rather, more effective learning is allowed.
- Through experiments, we show that the proposed meta-scheduling framework achieves competitive performances compared with the near-optimal conventional schedulers tailored to each BS, even though the propose framework only manages one policy for each objective in the network. Besides, the propose framework effectively adapts to the change of BS characteristics, which cannot be shown in the conventional schemes.

TABLE I
COMPARISON WITH RELATED WORKS ON DRL-BASED SCHEDULING FRAMEWORK

	Meta-learning	Cooperative learning	Feature of Scheduling Framework		
			Scheduling operation	Objectives of BSs	QoS requirements
[4], [5]	Not considered	Not considered	Not applicable (considering single BS)		Not considered
[6]	Not considered	Not considered	Not applicable (considering single BS)		Average data rate
[7]–[9]	Not considered	Not considered	Centralized	Identical	Not considered
[10]	Not considered	Not considered	Centralized	Identical	Not considered
[11]	Not considered	Not considered	Decentralized	Identical	Instantaneous SINR
[12]	Not considered	Not considered	Decentralized	Identical	Not considered
[13]	Not considered	Not considered	Decentralized	Identical	Average data rate
[27]	Shared network [22]	Not considered	Centralized	Identical	Not considered
[28], [29]	Fast adaptation [15]	Not considered	Not applicable (considering single BS)		Not considered
[30]	Fast adaptation [15]	Not considered	Not applicable (considering single BS)		Instantaneous data rate
Ours	Latent policy structure [19]	Considered	Decentralized	Diverse	Average data rate, latency

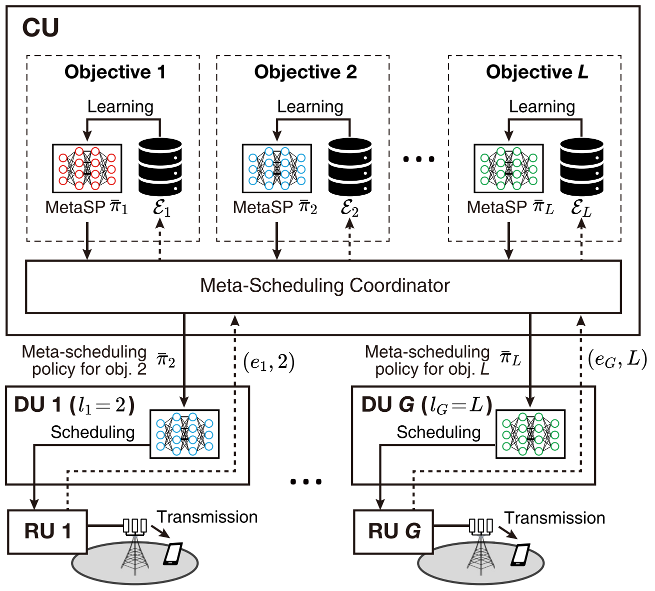


Fig. 1. The illustration of the overall meta-scheduling architecture with cooperative learning.

The rest of this paper is organized as follows. Section II provides the system model and the problem formulation. In Section III, a meta-scheduling framework with cooperative learning is proposed, and in Section IV, a meta-RL scheduling policy to enable the meta-scheduling framework is introduced. Section V provides simulation results. We finally conclude in Section VI.

II. Scheduling Problem in Cellular Networks for Diverse Objectives

A. System Model

We consider a cellular network whose frequency bandwidth and time domain are divided into subcarriers and timeslots, respectively. The cellular network consists of one CU and G BSs each of which is composed by DU and RU, as illustrated in Fig. 1. The set of the BSs is denoted by $\mathcal{G} = \{1, 2, \ldots, G\}$. We assume that U_g users are associated with BS g. The set of the users associated with BS g is denoted by $\mathcal{U}_g = \{1, 2, \ldots, U_g\}$.

For radio resource management, an RB is defined based on the subcarriers and timeslots. For example, in 5G NR, an RB is defined as 12 subcarriers whose spacing depends on numerology. In time domain, an RB consists of set of symbols whose number also depends on numerology. Without loss of generality, we assume that all BSs in the network operate over a common discrete time horizon $t \in \{1, 2, \ldots\}$, where a timeslot is a minimum time unit. Each BS can have different numbers of RBs within a timeslot according to bandwidths and numerologies. The number of RBs at BS g within a timeslot is denoted by N_g which set is denoted by $\mathcal{N}_g = \{1, 2, \ldots N_g\}$.

We consider L different scheduling objectives such as sum-rate maximization and satisfaction of QoS requirements. The set of the scheduling objectives is denoted by $\mathcal{L} = \{1, 2, \ldots, L\}$ and the scheduling objective of BS g is denoted by l_g.[1] We define the set of BSs that have scheduling objective l as $\mathcal{G}(l) = \{g \in \mathcal{G} | l_g = l\}$. In timeslot t, N_g RBs of BS g are allocated to the associated users to achieve the scheduling objective l_g. To this end, BS g collects a system state information of its user $u \in \mathcal{U}_g$ as $h^t_{g,u} = (h^t_{g,u,1}, \ldots, h^t_{g,u,K})$, where $h^t_{g,u,k}$ denotes k-th state information of user $u \in \mathcal{U}_g$ in timeslot t, such as channel quality indicator (CQI), buffer status, delay budgets, and retransmission indicators. Without loss of generality, we assume that each feature information is bounded by $[0, 1]$.

B. Problem Formulation and Decentralized Scheduling Based on DRL

We now formulate a scheduling problem for diverse objectives. We first define a state of BS g in timeslot t as

$$s^t_g = (h^t_{g,1}, \ldots, h^t_{g,U_g}) \in \mathcal{S}_g, \tag{1}$$

where $\mathcal{S}_g = [0, 1]^{U_g \times K}$ is a state space of BS g. By collecting the state space, a state of the network in timeslot t is defined as $s^t = (s^t_1, \ldots, s^t_G) \in \mathcal{S}$, where $\mathcal{S} = \prod_{g \in \mathcal{G}} \mathcal{S}_g$ is a state space of the network. We also define an action of BS g in timeslot t as

$$a^t_g = (u^t_{g,1}, \ldots, u^t_{g,N_g}) \in \mathcal{A}_g, \tag{2}$$

where $u^t_{g,n}$ is the scheduled user on RB n of BS g in timeslot t and $\mathcal{A}_g = \mathcal{U}_g^{N_g}$ is an action space of BS g. Then, we define an

[1]In 5G networks, multiple objectives in each BS are typically considered by network slicing with splitting radio resources [31]. Hence, we can easily extend the system model in our paper for multiple objectives in each BS by introducing a virtual BS that uses splitted radio resources for each objective.

action of the network in timeslot t as $a^t = (a_1^t, \dots, a_G^t) \in \mathcal{A}$, where $\mathcal{A} = \prod_{g\in\mathcal{G}} \mathcal{A}_g$ is an action space of the network. In scheduling problems, scheduling objectives are represented by utility functions [32], [33]. Thus, we denote a utility function for objective l as $r_l(\cdot)$ and define a total utility of the network in timeslot t as

$$r(s^t, a^t) = \sum_{g\in\mathcal{G}} r_{l_g}(s_g^t, a_g^t).$$

We define a policy for scheduling in the network $\pi : \mathcal{S} \to \mathcal{A}$ to map states to actions. Then, we formulate the problem as

$$\underset{\pi:\mathcal{S}\to\mathcal{A}}{\text{maximize}} \; U^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} (\gamma)^t r(s^t, \pi(s^t)) \middle| s^0 = s\right], \quad (3)$$

where γ is a discount factor and s_0 is an initial state. The optimal value function of the problem is defined by

$$J^*(s) = \max_\pi U^\pi(s), \quad \forall s \in \mathcal{S} \quad (4)$$

and the corresponding optimal policy is defined by $\pi^* = \text{argmax}_\pi U^\pi(s), \; \forall s \in \mathcal{S}$.

We introduce typical ways to solve the scheduling problem of the network in (3). First, if the system uncertainties such as channel statistics are perfectly known, it can be solved by using dynamic programming (DP) methods such as the well-known value iteration. However, in the real-world, it is impractical to acquire the perfect information on the system uncertainties in advance. Hence, recently, instead of the DP methods, DRL has been typically used, which directly learns the optimal value function or optimal policy in an online manner. However, due to the nature of policy learning, if any BS changes its objective, DRL should learn a new policy for the changed objective again from the scratch. Besides, DRL typically converges slower and requires much more experience as the size of the problem becomes larger [13]; the size of the problem in (3) exponentially grows according to the number of BSs. Consequently, it is still ineffective to directly apply DRL to solve the problem.

To address those issues, we consider a decentralized approach in which DRL methods are applied for the individual scheduling problem of each BS instead of the entire scheduling problems of the network. We can formulate a decentralized scheduling problem of BS g as

$$\underset{\pi_g:\mathcal{S}_g\to\mathcal{A}_g}{\text{maximize}} \; U_g^{\pi_g}(s_g) = \mathbb{E}\left[\sum_{t=0}^{\infty} (\gamma)^t r_{l_g}(s_g^t, \pi_g(s_g^t)) \middle| s_g^0 = s_g\right]. \quad (5)$$

Similar to the problem in (3), we can define the optimal value function of the decentralized scheduling problem of BS g as

$$J_g^*(s_g) = \max_{\pi_g} U_g^{\pi_g}(s_g), \quad \forall s_g \in \mathcal{S}_g. \quad (6)$$

In the decentralized approach, each BS learns its scheduling policy, π_g, independently from other BSs. The assumption of independent learning is reasonable in cellular networks since BSs typically determine their own scheduling decisions independently. Specifically, a BS does not intervene in the scheduling decisions of other BSs. If any interventions in scheduling decisions are required for interference coordination or cooperative transmission, a higher-level entity is involved.

C. Way to Evolve Decentralized Scheduling With Cooperative Learning in 5G Networks and Beyond

Typically, the objective of a BS and its BS-specific characteristics, such as the number of users, QoS requirements, and the number of RBs, may change over time. However, a policy by the conventional DRL-based scheduling algorithms [4], [5], [6], [7], [8], [9], [10], [11], [12], [13] has no capability of generalization over such diverse objective and BS-specific characteristics. (We will discuss this later in Section IV-A.) This implies that each BS should learn and manage a lot of policies to ensure supporting its diverse objectives and especially, its different BS-specific characteristics whose number could be countless. Hence, even if the decentralized approach can significantly reduce the complexity of the problem, applying DRL may be still inefficient in terms of network management and the expenses of DUs, which is against the concept of the CU-DU split structure pursuing cost-effective and flexible network deployment.

We can resolve the issues of scalability and inefficiency by exploiting the fact that BSs share the identical task in the problem–scheduling–for a limited number of objectives in $\mathcal{L}$. Specifically, if there exists a *generalized* scheduling policy that can be applied to BSs with *any* BS-specific characteristics to achieve a specific objective, the scheduling problem can be addressed by using only L policies each of which corresponds to each objective. Then, all the network has to do is to learn and manage L policies. To realize this approach more cost-efficiently, we can apply a concept of cooperative learning for the generalized scheduling policies. Specifically, with cooperative learning, instead of being learned by each individual BS, a generalized policy can be cooperatively learned by using the experiences from each BS. Furthermore, the functional split structure in the NR standard is a natural fit for cooperative learning, where the DU can perform decentralized scheduling by simply using the policies learned at the CU. In summary, such a scheduling framework with cooperative learning not only improves the computational efficiency thanks to the decentralized scheduling at each DU but also ensures network scalability by effectively exploiting the CU-DU split structure in 5G networks.

III. Meta-Scheduling With Cooperative Learning

In this section, we propose a meta-scheduling framework with cooperative learning to enjoy the advantages described in Section II-C. Here, we suppose that the generalized scheduling policy for each objective exists at the CU. We call such a generalized scheduling policy a *meta*-scheduling policy and will develop *meta*-scheduling policy in Section IV.

We first describe an overall architecture of the meta-scheduling framework in Fig. 1. The proposed architecture consists of one CU and G BSs each of which is composed of DU and RU. The CU has the meta-scheduling policy

Algorithm 1 Meta-Scheduling Framework With Cooperative Learning

```
1: Initialize meta-scheduling policies $\bar{\pi}_1, \ldots, \bar{\pi}_L$ at CU.
2: Initialize the policy at each BS $g$ by duplicating $\bar{\pi}_{l_g}$.
3: Set $t = 1$ and initial states $s^1 = (s^1_1, \ldots, s^1_G)$.
4: while TRUE do
     ▷ Scheduling phase
5:     for each BS $g \in \mathcal{G}$ do
6:         DU $g$ determines $a^t_g$ based on $s^t_g$ by using its $\bar{\pi}_{l_g}$.
7:         RU $g$ transmits data according to $a^t_g$.
8:         DU $g$ observes $e^t_g$.
9:     end for
     ▷ Cooperative learning phase
10:    Each DU $g$ reports $e^t_g$ along with $l_g$.
11:    CU conveys $e^t_g$'s from all DUs to $\mathcal{E}_l$'s according to $l_g$'s.
12:    CU learns $\bar{\pi}_l$ by using $e \in \mathcal{E}_l$ for all $l \in \mathcal{L}$.
13:    CU distributes the updated $\bar{\pi}_l$'s to corresponding DUs.
14:    DUs substitute their policies to the updated policies.
15: end while
```

for scheduling objective $l \in \mathcal{L}$, $\bar{\pi}_l$,[2] and its corresponding experience collection, $\mathcal{E}_l$, to learn the policy. In addition, the CU has a meta-scheduling coordinator which distributes the policies to the BSs according to their objectives and conveys the experiences from the BSs to the right collections. Through the coordinator, each DU acquires the policy from the CU according to its scheduling objective and makes the scheduling decisions by using the acquired policy in a decentralized manner. Then, the DUs send the experiences generated from their transmission results to the CU for cooperatively training the policies. Finally, the CU collects the experiences for each policy, trains the policies by using them, and periodically redistributes the updated policies to the corresponding BSs.

Now, we propose the meta-scheduling framework to perform scheduling in a decentralized manner at the DU of each BS while the policies are cooperatively learned in a centralized manner at the CU. The meta-scheduling framework consists of a scheduling phase and cooperative learning phase. In the scheduling phase, BSs determine scheduling decisions and report the experiences generated from the transmission results to the CU. Then, in the cooperative learning phase, the CU learns and redistributes the policies by using the experiences collected from all BSs. The meta-scheduling framework is summarized in Algorithm 1.

In the meta-scheduling framework, the meta-scheduling policies, $\bar{\pi}_l$'s, should be initialized in the CU at first (line 1). Then, each BS g initializes the policy by duplicating the policy corresponding to the objective l_g,[3] $\bar{\pi}_{l_g}$, from the CU (line 2). The initial state $s^1 = (s^1_1, \ldots, s^1_G)$ is initialized as well (line 3).

In the scheduling phase, each DU g allocates the RBs to users (i.e., determines the action a^t_g) by using the state of BS g, s^t_g, and the meta-scheduling policy, $\bar{\pi}_{l_g}$, obtained from the CU (line 6). According to the decision a^t_g, each RU g transmits data to the scheduled users (line 7). Then, each DU g can observe the transmission results and generate the experience e^t_g from the observed results (line 8). In RL, the experience consists of the state, action, utility (i.e., reward signal) from the transmission, and next state, can be used to learn the policy $\bar{\pi}_{l_g}$. The detailed definitions of the experience will be described in Section IV.

In the cooperative learning phase, the generated experience of DU g is reported to the CU along with the objective as (e^t_g, l_g) to specify the corresponding objective of the experience (line 10). The CU receives the experiences and corresponding objectives from all DUs, (e^t_g, l_g)'s. Then, each experience e^t_g is conveyed to the corresponding experience collection $\mathcal{E}_{l_g}$ (line 11). The CU learns the meta-scheduling policies $\bar{\pi}_l$'s by using the experiences in the collections $\mathcal{E}_l$'s (line 12). The detailed algorithm for learning the meta-scheduling policies will be proposed in Section IV. After learning the policies, the CU distributes each updated policy for objective l, $\bar{\pi}_l$, to the DUs with objective l (i.e., DU $l \in \mathcal{G}_l$), (line 13) and each DU updates the policy (line 14). By this way, the policies are cooperatively learned by using the experiences from all BSs. Therefore, the BSs can reduce the computational burden to learn and manage the policies.

Remark 1: In the meta-scheduling framework, the cooperative learning phase can be substituted by federated learning (FL) [34]. However, in the framework, the major advantages of FL such as data privacy protection and reduction of communication costs are not attractive; the experience e^t_g of DU g can be transmitted to the CU without any privacy issues and massive communication costs. Rather, FL imposes DUs additional computational costs due to learning their local policies.

IV. Meta-Scheduling Policy Based on Meta-RL

In this section, we propose a meta-scheduling policy for the cellular network based on meta-RL. Then, we integrate the proposed meta-scheduling policy into the meta-scheduling framework. For the sake of simplicity, we omit the BS index g in this section if there is no confusion.

A. Meta-RL: Promising Way to Realize Meta-Scheduling Policy

A meta-scheduling policy can be defined as a policy that can be used in any BS-specific characteristics without any changes of the scheduling policy. Such a policy is hard to be implemented if typical state and action representations given in (1) and (2), respectively, are used to define the policy. This is because they describe the system information and scheduling decisions in a way depending on BS-specific characteristics as in most conventional DRL-based scheduling algorithms [4], [5], [7], [8], [9], [10], [11], [12], [13]. Specifically, the representations directly depend on the number of users at BS g, U_g, the user index corresponding to each user, and the number of RBs at BS g, N_g. For example, suppose that a policy is learned from BS 1 with N_1 RBs and U_1 users whose average data rate requirement is given by 1 Mbps. Then, the policy is hard to be used for BS 2 with N_2 RBs and U_2 users whose average data rate requirement is given by 2 Mbps since

[2]It is worth noting that the bar notation $\bar{\cdot}$ is used to distinguish the definitions related to meta-scheduling from the conventional ones in this paper.

[3]Here, we do not consider a time-varying scheduling objective since the objectives are generally changing over much larger time-scale compared with scheduling decision. However, even if the objective of a BS changes, we can simply address it by adding a procedure that the BS substitutes its policy to the policy at the CU which corresponds to its changed objective.

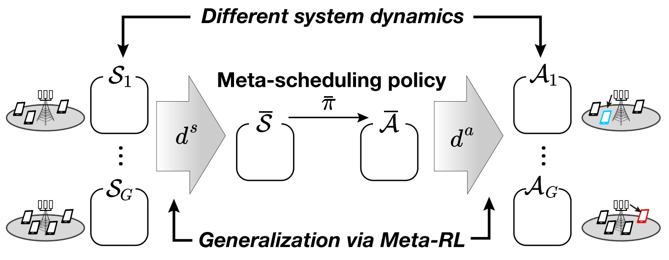


Fig. 2. Illustration of a meta-RL structure for meta-scheduling.

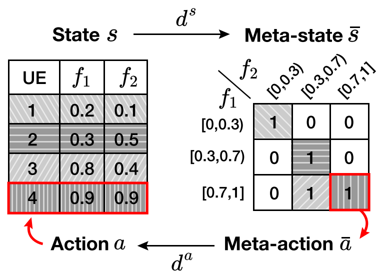


Fig. 3. Illustration of state and action transformations for meta-scheduling.

the state and action spaces have the different dimensions and the corresponding user for each user index has the different QoS requirement. Hence, to realize meta-scheduling, we need a policy structure that does not depend on any BS-specific characteristics.

To this end, we adopt the concept of meta-RL that has been widely studied to enable a policy to efficiently address a family of problems having different tasks and/or different system dynamics [14]. Specifically, we consider the meta-RL methods addressing a family of problems with an *identical* task having *different* system dynamics [18], [19], since the meta-scheduling problems consider an identical scheduling task for a specific objective in different BS-specific characteristics. To address the family of problems with different system dynamics, the meta-RL methods construct a latent policy structure and translate the different environments into the latent policy structure. By using meta-RL, we propose the meta-scheduling policy that can play a role of the generalized scheduling policy, which is required to realize the meta-scheduling framework. As illustrated in Fig. 2, in the meta-scheduling policy, the state of any BS can be transformed into the generalized state and the generalized action can be transformed into the action of any BS.

B. Structure of Meta-Scheduling Policy

We now propose a meta-RL structure for meta-scheduling policies to solve the problem in (5). To this end, we borrow the concept of the system-agnostic policy in [19], which is originally proposed to address the change of system dynamics, but can be used to define the meta-RL structure as well. Since the structure of the system-agnostic policy can consider only a single RB, we need to generalize the structure for multiple RBs. However, we cannot simply extend the structure for N RBs in a combinatorial manner since the structure may require an impractical computational complexity due to the combinatorial decision. Besides, we cannot achieve meta-scheduling via such an extension since the number of RBs, N, is one of the BS-specific characteristics as well.[4] Hence, in this subsection, we first focus on a meta-RL structure for a *single* RB. Then, we design a meta-scheduling procedure for multiple RBs.

1) Meta-RL Structure: The meta-RL structure consists of a generalized state space, a state transform function, a generalized action space, and an action transform function. Then, the meta-scheduling policy can be defined by using the generalized spaces as illustrated in Fig. 2. For the sake of simplicity, we call the generalized states and actions meta-state and meta-action, respectively. We first define a meta-state, $\bar{s}$, so as to describe the system state information which does not rely on BS-specific characteristics. To this end, we partition the state space of k-th state information, h_k, into H_k multiple disjoint intervals. Then, the entire state space $\mathcal{S}$ can be partitioned into $M = \prod_{k\in\{1,\ldots,K\}} H_k$ subsets, and any state that each user could have belongs to one of the subsets. Hence, we can represent the system states (i.e., the state information of the users) as follows: *whether any user that has a state belonging to each subset exists or not*. Specifically, the meta-state in timeslot t, $\bar{s}^t$, is defined by a K-dimensional matrix whose each element indicates the existence of any user in the corresponding subset by the value of 1 or 0 as illustrated in Fig. 3. Contrary to the conventional state in (1), the meta-state describes the system states in an implicit way which can be interpreted as a user condition map. With the definition of the meta-state, any state s^t can be transformed as $\bar{s}^t = d^s(s^t)$, where $d^s(\cdot)$ is a transformation function for states. For simple notation, we index the meta-state as $m \in \{1,\ldots,M\}$, where M is the number of elements in meta-states. Then, the meta-state space is defined by $\bar{\mathcal{S}} = \{0,1\}^M$.

A meta-action, $\bar{a}$, can be defined as the index of the element in the meta-state, $\bar{a}^t = m^t$. Then, the policy chooses user u^t among the set of the users in the condition corresponding to m^t to serve. A transformation function for actions is defined as $a^t = d^a(\bar{a}^t)$. To avoid choosing the condition with no user, the set of the available meta-actions is defined as $\bar{\mathcal{A}}(\bar{s}^t) = \{\bar{a}^t \in \bar{\mathcal{A}} | \bar{s}^t(m^t) = 1\}$, where $\mathcal{A} = \{1,\ldots,M\}$ is a meta-action space and $\bar{s}^t(m)$ denotes the value of m-th element of $\bar{s}^t$. Finally, a meta-scheduling policy can be defined to map meta-states to meta-actions as $\bar{\pi} : \bar{\mathcal{S}} \to \bar{\mathcal{A}}$.

2) Theoretical Analysis: We first define identical statistics of the system uncertainties as follows.

Definition 1: The statistics of the system uncertainties of users are identical if the following conditions hold: for any two users u_1, u_2, $\mathbb{P}_{u_1}(h'|h,u') = \mathbb{P}_{u_2}(h'|h,u')$ and $\mathbb{P}_{u_1}(h'|h,u_1) = \mathbb{P}_{u_2}(h'|h,u_2)$, $\forall h', h \in [0,1]^K, u' \in \mathcal{N} \setminus \{u_1,u_2\}$, where $\mathbb{P}_u(h'|h,u')$ represents the transition probability over the system uncertainties of user u with given state information of a user h and scheduled user u'.
In wireless communications, the most significant uncertainties are caused by channel statistics comprised of various factors such as fading and shadowing. We show that typical wireless channel models hold the identical statistics condition in the following lemma.

Lemma 1: The wireless channel models such as TR38.901 [35] hold the identical statistics condition if all the

[4]It is worth noting that even if all BSs have an identical bandwidth, their N's may be different if they use different numerologies.

parameters of the channel models are identified in the state information.

Proof: Typically, channel models consist of deterministic components, such as LOS and NLOS pathloss exponents, and statistical components, such as the probability of LOS, fading, and shadowing. In general, given parameters of a channel model, the deterministic component and the probability distribution of the statistics component are determined. We define the state information of a user u as $h_u = (p_u^1, \dots, p_u^{K_P}, f_u^1, \dots, f_u^{K_f})$, where p_u^k is k-th channel model parameter and f_u^k is k-th channel output. If the parameters are static, $\mathbb{P}_u(h'|h, u')$ indicates the transition probability of the channel outputs from given state information h and scheduled user u' to state information h'. From the definition of the state information, we can see that the probability boils down the probability that the channel outputs change to $f_u^{k'}$'s from f_u^k's with given parameters p_u^k's since the user scheduling does not affect the channel. If two users have identical parameters, the channel outputs are generated by a completely identical process with the identical statistics component, which implies that their probability distributions are identical as well. Hence, the wireless channels hold the identical statistics condition. ■

We assume that the meta-state is defined by uniformly partitioning each state information into 2^ξ intervals whose length is given by $2^{-\xi}$, where ξ is the partitioning parameter. We denote the optimal value function based on the meta-RL structure with the partitioning parameter b by $\bar{J}_\xi^*(\bar{s})$. Then, in the following theorem, the optimality of the meta-scheduling policy is described.

Theorem 1: For meta-scheduling, if the statistics of the system uncertainties are identical, the optimal value of the scheduling problem for a given meta-state $d_s(s)$ with parameter b, $\bar{J}_\xi^*(d_s(s))$, converges to the optimal value of the scheduling problem a given conventional state s, $J^*(s)$, as $\xi \to \infty$ (i.e., $J^*(s) = \lim_{\xi\to\infty} \bar{J}_\xi^*(d_s(s))$).

Proof: Here, we provide an outline of the proof with similar steps in [19]. Suppose $\xi \to \infty$. Then, any two users whose state information belongs to an identical subset m of the meta-RL structure become indistinguishable in terms of system state information. Along with the identical statistics condition, such indistinguishability implies that even if whoever will be scheduled among the two users, its consequent utilities are indistinguishable. Consequently, when $\xi = \infty$, choosing any user among the users whose state information belongs to the same m has identical consequences to the value function. We then suppose a meta-scheduling policy that for any given state s, chooses the meta-action to which the user chosen by the optimal policy $\pi^*(s)$ belongs. Then, as $\xi \to \infty$, the meta-scheduling policy becomes the optimal one, $\bar{\pi}^*$, and its value function converges to the optimal value function, $J^*(s)$, since the meta-scheduling policy guarantees to choose the users that are indistinguishable from the user chosen by $\pi^*(s)$ in terms of the value function. More details can be found in [19]. ■

Theorem 1 shows that the meta-scheduling policy can achieve the optimal value when the partition is fine enough and the statistics of all the users are identical.

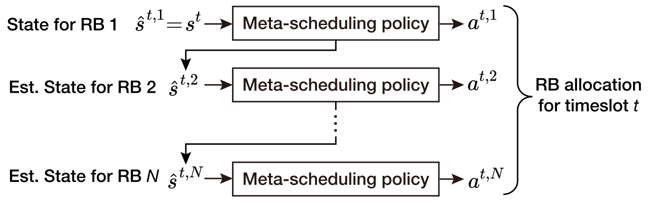


Fig. 4. Illustration of incremental scheduling.

C. Meta-Scheduling Procedure for Cellular Networks

In this subsection, we design a meta-scheduling procedure for *multiple* RBs in practical cellular networks. To this end, we extend the meta-scheduling policy for a single RB in the previous subsection to allocate multiple RBs within each timeslot.

1) Incremental Scheduling: We first propose an incremental scheduling method that allows a scheduler to allocate the RBs by using the meta-scheduling policy. When allocating RBs one by one, the effect of one RB allocation should be assessed and reflected to the next RB allocation. Hence, in the proposed method, the change of system state s^t in timeslot t according to each RB allocation is consecutively estimated and used to determine the next RB allocation. To this end, we first classify the state information into two groups according to the dependence of RB allocations. For example, the information such as achieved data rate and buffer depends on each RB allocation, while the information such as CQI and remaining delay budget does not. We denote the set of the indices of the information which depends on RB allocations by $\mathcal{K}_{RB}$ and that of the other information by $\mathcal{K}_{RB}^{\complement} = \{1, \dots, K\} \setminus \mathcal{K}_{RB}$. At the beginning of scheduling, RB allocation 1 is determined by using the state, $s^t = \hat{s}^{t,1}$,[5] and the meta-scheduling policy. For RB allocation 1, the effects on the k-th state information of user u_1^t for all $k \in \mathcal{K}_{RB}$ are evaluated and the evaluation is incrementally reflected to derive the estimated state $\hat{s}^{t,2}$. With $\hat{s}^{t,2}$, the scheduler determines the RB allocation 2 based on the meta-scheduling policy. By repeating scheduling and evaluation, the scheduler can allocate N RBs one by one considering the effect of each RB allocation. The incremental scheduling method is illustrated in Fig. 4.

2) Experience Decomposition: The incremental scheduling allows the scheduler to allocate the RBs within a single timeslot one by one considering the effects of each RB allocation. However, at the same time, a problem appears when learning the scheduling policy using RL. Since the policy allocates each RB, to learn the policy, we need an individual experience for each RB allocation, which is constructed based on the utility and consecutive state of the RB allocation. However, in typical cellular networks, a unit of transmission is given by a timeslot, which implies that after the transmission in timeslot t, we can observe only the transmission results of timeslot t and the system state in timeslot $t+1$. Both the transmission results and system state are determined by combining the transmission results of the entire RB allocations in timeslot t. Hence, we need to appropriately decompose such combined information to generate an individual experience for each RB allocation.

[5]Note that the hat notation $\hat{\cdot}$ denotes an estimated value in this paper.

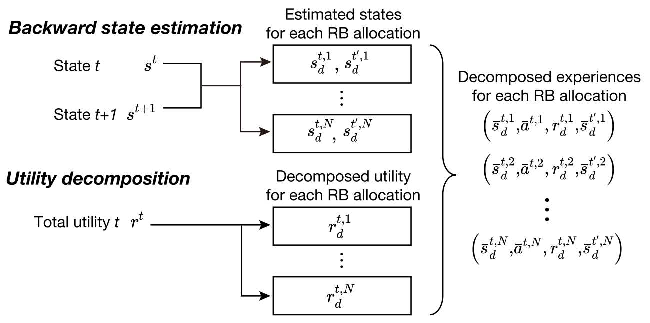

Backward state estimation
Estimated states for each RB allocation
State t s^t
State $t+1$ s^{t+1}
$s_d^{t,1}$, $s_d^{t',1}$
$s_d^{t,N}$, $s_d^{t',N}$
Utility decomposition
Decomposed utility for each RB allocation
Total utility t r^t
$r_d^{t,1}$
$r_d^{t,N}$
Decomposed experiences for each RB allocation
$\left(\bar{s}_d^{t,1}, \bar{a}^{t,1}, r_d^{t,1}, \bar{s}_d^{t',1}\right)$
$\left(\bar{s}_d^{t,2}, \bar{a}^{t,2}, r_d^{t,2}, \bar{s}_d^{t',2}\right)$
$\left(\bar{s}_d^{t,N}, \bar{a}^{t,N}, r_d^{t,N}, \bar{s}_d^{t',N}\right)$

Fig. 5. Illustration of experience decomposition.

To this end, we propose an experience decomposition method. Before describing the method, we provide several notations and definitions. We define the difference of k-th state information of user u between s^{t+1} and s^t as

$$d_{u,k}^t = h_{u,k}^{t+1} - h_{u,k}^t. \tag{7}$$

We also define the set of RBs allocated to user u in timeslot t as $\mathcal{N}^t(u) = \{n \in \mathcal{N} : u_n^t = u\}$. Since the transmission results of user u in timeslot t are observed in a combined manner, we need to represent the contribution of each RB to the results. Thus, we define the contribution of RB n to user u as $c_u^{t,n}$, which satisfies

$$c_u^{t,n} = 0 \text{ if } n \notin \mathcal{N}^t(u) \text{ and } \sum_{n \in \mathcal{N}} c_u^{t,n} = 1. \tag{8}$$

For example, it can be determined as equal to each allocated RB or as considering the CQI of each allocated RB.

In the proposed experience decomposition method, we virtually generate the experience of each RB allocation as if each RB is transmitted consecutively. Since the action is the known information in the experience, for each RB allocation, we need to estimate the changes of the states from the state in timeslot t to the state in timeslot $t+1$ and to decompose the total utility of timeslot t. We first describe a backward state estimation for each RB allocation. We denote the estimated state at RB allocation n in timeslot t by $s_d^{t,n}$ and the consecutive estimated state by $s_d^{t',n}$. Then, we have $s_d^{t',N} = s^{t+1}$ and $s_d^{t,1} = s^t$. We denote the k-th state information of user u in the estimated state $s_d^{t,n}$ by $h_{d,u,k}^{t,n}$. All we need to estimate are $h_{d,u,k}^{t,n}$'s and $h_{d,u,k}^{t',n}$'s, $\forall k \in \mathcal{K}_{RB}$, since the k-th state information for all $k \in \mathcal{K}_{RB}^{\complement}$ are independent to RB allocations (i.e., $h_{d,u,k}^{t,n} = h_{u,k}^t$ and $h_{d,u,k}^{t',n} = h_{u,k}^{t+1}$, $\forall k \in \mathcal{K}_{RB}^{\complement}$). To this end, we estimate the states in a backward direction starting with the consecutive state of RB allocation N, $s_d^{t',N} = s^{t+1}$, which is the state on which all RB allocations are fully reflected. We estimate the k-th state information of user u in the state at RB allocation N, $s_d^{t,N}$, by using the difference in (7) as

$$h_{d,u,k}^{t,N} = h_{d,u,k}^{t',N} - c_u^{t,N} d_{u,k}^t, \quad \forall u \in \mathcal{U}, \ \forall k \in \mathcal{K}_{RB}. \tag{9}$$

For the estimation of $s_d^{t,N-1}$, we set $h_{d,u,k}^{t',N-1}$ to be $h_{d,u,k}^{t,N}$, $\forall k \in \mathcal{K}_{RB}$. For $(N-1)$-th RB allocation, we estimate $h_{d,u,k}^{t,N-1}$ by using $h_{d,u,k}^{t',N-1}$, $c_u^{t,N-1}$, and $d_{u,k}^t$, similarly to (9). Through iterative estimation, we can estimate all the decomposed states as illustrated in Fig. 5.

Now, we describe a utility decomposition for each RB allocation. We first assume that the utility r^t is defined by

$$r^t = r_C^t + \sum_{u \in \mathcal{U}} r_u^t, \tag{10}$$

where r_C^t is the common utility over all users which is hard to be decomposed for each RB allocation, and r_u^t is the decomposable utility of user u. For example, the cost due to QoS outages is a common utility since QoS outages typically depend on the combinations of all RB allocations rather than a single RB allocation. The common utility is broadcasted when decomposing the utility, while the decomposable utilities are reflected only to the corresponding users. Furthermore, the decomposable utility of user u, r_u^t, is comprised of the utility of RB allocation n to user u, $r_u^{t,n}$, as $r_u^t = \sum_{n \in \mathcal{N}^t(u)} r_u^{t,n}$. Note that if only the utility of user u, r_u^t, is observable and the individual utility of each RB allocation to user u is not, we can decompose $r_u^{t,n}$ by using the contribution of each RB as $r_u^{t,n} = c_u^{t,n} r_u^t$. Then, we define the decomposed utility of RB allocation n in timeslot t as

$$r_d^{t,n} = r_C^t + r_{u_n}^{t,n}, \tag{11}$$

By using the estimated states and decomposed utilities, we can generate the experience of each RB allocation n for the meta-scheduling policy as

$$e^{t,n} = (\bar{s}_d^{t,n}, \bar{a}^{t,n}, r_d^{t,n}, \bar{s}_d^{t',n}), \quad \forall n \in \mathcal{N}, \tag{12}$$

where $\bar{s}_d^{t,n}$ and $\bar{a}^{t,n}$ are the meta-state and meta-action corresponding to $s_d^{t,n}$ and $a^{t,n}$.

D. Procedure of Meta-Scheduling Framework

Here, we describe a meta-scheduling procedure where meta-scheduling policies are applied to the framework in Section III. For the algorithm, we use a typical deep Q-network (DQN) algorithm [36] to learn and utilize the meta-scheduling policy, but any other DRL approaches can be used as well. In the algorithm, a deep neural network (DNN), $\boldsymbol{\theta}$, is considered to approximate the optimal action-value function based on the meta-RL structure, $Q^*(\bar{s}, \bar{a})$. We denote the approximation of the optimal action-value function by $Q(\bar{s}, \bar{a}; \boldsymbol{\theta})$.

We provide the detailed procedure in Algorithm 2. For the initialization of the meta-scheduling framework (lines 1–2 in Algorithm 1), the meta-scheduling procedure sets the parameters (line 1). For each scheduling objective $l \in \mathcal{L}$, the procedure initializes the corresponding experience collection, $\mathcal{E}_l$ (line 2). In addition, two DNNs with weights $\boldsymbol{\theta}_l$ and $\boldsymbol{\theta}_l^-$ are generated and the weights $\boldsymbol{\theta}_l^-$ are initialized to $\boldsymbol{\theta}_l$ (line 3). The DNN with weights $\boldsymbol{\theta}_l$ is the Q-netwok for objective l and that with weights $\boldsymbol{\theta}_l^-$ is a target Q-network for objective l to be used when training $\boldsymbol{\theta}_l$. Then, each DU $g \in \mathcal{G}$ duplicates the Q-network for objective l_g, $\boldsymbol{\theta}_{l_g}$, from the CU (line 4–6).

For the scheduling at DU g (line 6 in Algorithm 1), DU g schedules the RBs one by one via the incremental scheduling method (lines 7–11). The method transforms the state into the meta-state by using the transformation function d^s (line 8).

Algorithm 2 Procedure of Meta-Scheduling Framework

```
    ▷ For lines 1–2 in Algorithm 1
1:  (CU) Set parameters $\gamma$, $T^{replay}$, and $T^{target}$.
2:  (CU) Initialize experience collection for objective $l$ $\mathcal{E}_l$, $\forall l \in \mathcal{L}$.
3:  (CU) Initialize Q-network for objective $l$ with weights $\theta_l$, and target Q-network for objective $l$ with weights $\theta_l^- = \theta_l$, $\forall l \in \mathcal{L}$.
4:  for each BS $g \in \mathcal{G}$ do
5:      (DU $g$) Initialize Q-network $Q_{l_g}$ by duplicating $\theta_{l_g}$.
6:  end for
    ▷ For line 6 in Algorithm 1 (Incremental Scheduling)
7:  for each RB $n \in \{1, \ldots, N_g\}$ do
8:      (DU $g$) Obtain $\hat{s}_g^{t,n}$ and transform it into $d^s(\hat{s}_g^{t,n})$.
9:      (DU $g$) Determine $\bar{a}_g^{t,n}$ using $d^s(\hat{s}_g^{t,n})$ according to $\epsilon$-greedy.
10:     (DU $g$) Estimate $\hat{s}^{t,n+1}$ for the next RB allocation.
11: end for
    ▷ For line 8 in Algorithm 1 (Experience Decomposition)
12: (DU $g$) Perform experience decomposition using $s_g^t$, $s_g^{t+1}$, $r_g^t$, and $\bar{a}_g^{t,n}$'s.
13: (DU $g$) Obtain $e_g^{t,n}$ for all $n \in \mathcal{N}_g$ as in (12).
14: (DU $g$) Report experiences to CU as $(e_g^{t,n}, l_g)$ for all $n \in \mathcal{N}_g$.
    ▷ For lines 10–11 in Algorithm 1
15: (CU) Store experience sample $e_g^t$'s in $\mathcal{E}_{l_g}$.
16: if ($t$ mod $T^{replay}$) == 0 then
17:     for each objective $l \in \mathcal{L}$ do
18:         (CU) Sample random batches of samples $(\bar{s}, \bar{a}, r, \bar{s}')$'s from $\mathcal{E}_l$.
19:         (CU) Set $y(\theta_l^-) = r + \gamma \max_{\bar{a} \in \bar{\mathcal{A}}(\bar{s}')} Q(\bar{s}', \bar{a}; \theta_l^-)$.
20:         (CU) Perform a gradient descent method to minimize the target error (i.e., loss function)
                $$L(\theta_l, \theta_l^-) = (y(\theta_l^-) - Q(\bar{s}, \bar{a}; \theta_l))^2 \quad (13)$$
            with respect to the network weights $\theta_l$.
21:     end for
22: end if
23: Every $T^{target}$ timeslots $\theta_l^- = \theta_l$, $\forall l \in \mathcal{L}$.
    ▷ For lines 12–13 in Algorithm 1
24: if ($t$ mod $T^{replay}$) == 0 then
25:     for each objective $l \in \mathcal{L}$ do
26:         (CU) Distribute $\theta_l$ to DU $g \in \mathcal{G}(l)$.
27:     end for
28:     for each BS $g \in \mathcal{G}$ do
29:         (DU $g$) Substitute its $\theta_{l_g}$ into the updated one.
30:     end for
31: end if
```

Then, the meta-scheduling policy is utilized with the following ϵ-greedy policy (line 9):

$$\bar{a}_g^{t,n} = \begin{cases} \text{a random action}, & \text{with probability } \epsilon, \\ \underset{\bar{a} \in \bar{\mathcal{A}}(d^s(\hat{s}_g^{t,n}))}{\text{argmax}} \; Q(d^s(\hat{s}_g^{t,n}), \bar{a}; \theta_{l_g}), & \text{otherwise.} \end{cases}$$

Then, the consecutive state is estimated for the next RB allocation. To observe the experiences at DU g in the meta-scheduling framework (line 8 in Algorithm 1), DU g conducts the experience decomposition method by using s_g^t, s_g^{t+1}, r_g^t, and $\bar{a}_g^{t,n}$'s (line 12). Then, DU g can get N_g experiences (i.e., $e_g^{t,n}$'s for all $n \in \mathcal{N}_g$) (line 13), and reports the experiences to the CU (line 14).

For the cooperative learning in the meta-scheduling framework (lines 10–11 in Algorithm 1), the CU stores the experiences received from all DUs in the corresponding experience collections $\mathcal{E}_l$'s (line 15), and trains the DNNs for meta-scheduling policies by using the experience collections (lines 16–23). In every T^{replay} timeslots, for each scheduling objective l, batches of experiences are randomly sampled from the experience collection $\mathcal{E}_l$ to train the Q-network (line 18). Using the sampled experiences, the CU sets the target Q-values by using the experience and the target Q-network (line 19). The randomly sampled batches reduce the correlation among the experiences. Then, it trains the Q-network (i.e., the weights θ_l) to minimize the target error in (13) (line 20). The target Q-network is updated as $\theta_l^- = \theta_l$ in every T^{target} timeslots so as to mitigate the non-stationarity of the target Q-value. Both random batch and target Q-network are proposed in [36] to help stable learning. We refer the readers to [36] for more details. Finally, for distributing the updated meta-scheduling policy in the meta-scheduling framework (lines 12–13 in Algorithm 1), the CU distributes the weights θ_l to all DUs that have scheduling objective l (i.e., $g \in \mathcal{G}(l)$) for all $l \in \mathcal{L}$ if there exists updates (i.e., in every T^{replay} timeslots) (lines 25–27). Then, each DU g substitutes θ_{l_g} into the updated one (lines 28–30).

Remark 2: Since the procedure is proposed based on the DQN algorithm, its convergence follows the convergence analysis of the DQN algorithm. Typically, it is difficult to investigate the theoretical convergence of the DQN algorithm due to the function approximation nature of DNNs. Nevertheless, recently, several works have investigated the theoretical convergence of the DQN algorithm [37], [38]. As shown in the analyses, we can expect the DQN algorithm to approximate the optimal action-value function closely.

Remark 3: The computational complexity of the meta-learning framework follows that of the DQN algorithm and the number of scheduling objectives L. Suppose that a neural network used in the DQN algorithm consists of fully-connected layers, whose numbers of layers and units per layer are given by m and n, respectively. Then, the computational complexity of the framework is given by $O(Lmn \log n)$ since the computational complexity for the DQN algorithm is given by $O(mn \log n)$ [13], [39].

E. Implementations of Meta-Scheduling Policy for Different Objectives

In this subsection, we provide several examples of implementating the meta-scheduling policy for different objectives. The examples are summarized in Table II.

1) Sum-Rate Maximization With Average Data Rate Requirements: The sum-rate maximization, which is one of basic and traditional objectives, can be easily achieved by using a linear user utility function (i.e., $r_u^t = g_u^t$). Here, we additionally consider the minimum average data rate requirements of each user. To this end, we track the unsatisfaction degree of the data rate requirement (UDR) of user u in timeslot t, μ_u^t. The Lagrangian multipliers in the Lagrangian approach [6], [40] and the virtual queue length for the constraints in the Lyapunov optimization framework [41], [42] can be described as such an unsatisfaction degree. Inspired by the scheduling method based on the Lagrangian approach [40], we define the unsatisfaction degree as $\mu_u^{t+1} = \mu_u^t - \nu^t(g_u^t - G_u)$, where ν^t is the step size in timeslot t, and G_u is the average data rate requirement of user u. Then, the state information of each user can be defined as the CQI and UDR. The CQI is the information that does not depend on RB allocations (i.e., it belongs to $\mathcal{K}_{RB}^{\complement}$), while the UDR does (i.e., it belongs to $\mathcal{K}_{RB}$). The utility of RB allocation n to user

TABLE II
EXAMPLES OF IMPLEMENTATION OF META-SCHEDULING POLICY

	Sum-Rate Max.	**Energy Consumption Min.**	**Satisfaction of QoS Requirements**
State	CQI, UDR	CQI, UDR	CQI, UDR, buffer, left delay budget
Common utility	-	-	$\sum_{u\in\mathcal{U}} r_{fail} q^t_{u,fail}$
User utility of each RB	$c^{t,n}_{u_n}(1+\mu^t_{u_n})g^t_{u_n}$	$\begin{cases} c^{t,n}_{u_n}\mu^t_{u_n}g^t_{u_n} - r_{alloc}, & \text{if RB } n \text{ is allocated} \\ 0, & \text{otherwise} \end{cases}$	$c^{t,n}_{u_n}(\mu^t_{u_n}g^t_{u_n} + r_{succ}q^t_{u_n,succ})$
Contribution of each RB	CQI based contribution		

u_n in timeslot t is defined as $r^t_{u_n} = c^{t,n}_{u_n}(1+\mu^t_{u_n})g^t_{u_n}$. The contribution of each RB to user n in timeslot t is determined by the corresponding expected data rate of the CQI of the RB while satisfying the condition in (8).

2) Energy Consumption Minimization With Average Data Rate Requirements: We consider a scheduler to minimize the energy consumption, which is one of important objectives in green communications, while satisfying the minimum average data rate requirement. Inspired by the related works [43], [44], we assume that each RB does not need to be allocated always and the transmission power is proportional to the number of allocated RBs. Then, we can achieve the objective by minimizing the number of the allocated RBs. To this end, we define the state information of each user by the CQI and UDR same as in the case of the sum-rate maximization objective. If RB allocation n is performed, its utility of user u_n in timeslot t is defined as $r^t_{u_n} = c^{t,n}_{u_n}\mu^t_{u_n}g^t_{u_n} - r_{alloc}$, and if not, the utility in timeslot t is zero, where r_{alloc} denotes the energy cost. The contribution of each RB to user n in timeslot t is determined identically to the sum-rate maximize objective example.

3) Satisfaction of Various QoS Requirements: Here, we consider latency requirements in addition to the minimum average data rate requirements. For a user with the latency requirements, the BS should transmit the packet in the buffer of the user within the delay budget. We first define the state information of each user as CQI, buffer, left delay budget, and UDR, where buffer belongs to $\mathcal{K}_{RB}$ and left delay budget belongs to $\mathcal{K}^{\complement}_{RB}$. In the utility function, the utility of RB allocation n to user u in timeslot t is defined as $r^t_u = c^{t,n}_u(\mu^t_u g^t_u + r_{succ}q^t_{u,succ})$, where r_{succ} is the utility parameter for successful packet transmission and $q^t_{u,succ}$ is the successful packet transmission indicator of user u in timeslot t. The second term denotes the utility for the latency requirements. In addition, we need to consider the common utility due to the packet transmission failure. We define the common utility in timeslot t as $r^t_C = r_{fail}\sum_{u\in\mathcal{U}} q^t_{u,fail}$, where r_{fail} is the negative utility parameter for packet transmission failures, and $q^t_{u,fail}$ is the packet transmission failure indicator of user u in timeslot t. The contribution of each RB to user n in timeslot t is determined as in the settings for the other objectives.

F. Implementation Aspects of Meta-Scheduling in NR and ORAN

Here, we briefly discuss the feasibility of the proposed meta-scheduling framework for existing network architectures and the way forward. Functional split options have been defined in the NR standard [45], as illustrated in Fig. 6. The meta-scheduling framework can be used with the functional split options from 1 to 4 in which a scheduling functionality (generally in MAC) is located in DUs. Especially, the functional split option 2 may be the most attractive candidate option in terms of efficient network management. It allows a CU not only to support dual-connectivity in NR [46] but also to manage logical network slices [47]. Besides, for the option 2, the interface between the CU and DU has been standardized as F1 interface [2]. With the aid of the F1 interface, MAC schedulers in each DU can be coordinated by the CU. Such a coordination-available structure allows us to migrate the meta-scheduling framework to the current 5G network architecture.

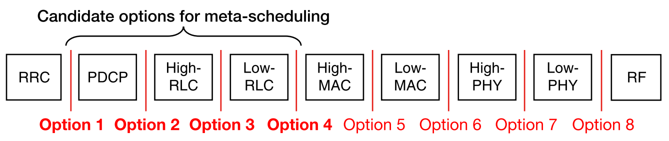


Fig. 6. The functional split options [45]. The abbreviations stand for radio resource control layer (RRC), packet data convergence protocol layer (PDCP), radio link control layer (RLC), media access control layer (MAC), physical layer (PHY), and radio frequency layer (RF).

Furthermore, open RAN architecture (ORAN), which targets to provide virtualized network elements and an open interface, has introduced a radio intelligence controller (RIC) designed for intelligent radio resource management and RAN optimization [48]. In ORAN architecture, the E2 interface between the RIC and DU is proposed for RIC to directly control the DU. The E2 interface and corresponding structure can also integrate the meta-scheduling framework into the ORAN architecture. As a future work, the interface should be elaborated to support functionalities, such as policy synchronization (to DU) and scheduling experience aggregation (from DU), for practical implementation of the meta-scheduling framework in beyond 5G networks and ORAN architectures.

Lastly, we briefly discuss the impact of the meta-scheduling framework on the interfaces in terms of overhead and delay. Compared with the amount of information exchanged between the CU and DUs via the existing interfaces [2], the information exchanged in the framework (i.e., experiences and Q-networks) is relatively insignificant. Besides, each DU can determine its scheduling decisions without any delay due to the training of DNNs since in the framework, the DNNs are trained at the CU separately from the DU. These show that the meta-scheduling framework can be practically implemented in 5G and beyond-5G networks.

TABLE III
TYPES OF TRAFFICS

Types	A	B	C	D
Data rate requirement (Mbps)	2	8	-	-
Packet data amount (Kbytes)	-	-	255	1354
Delay budget (ms)	-	-	5	10
Arrival prob. in each timeslot	-	-	0.1	0.05

V. SIMULATION RESULTS

In the simulation, we consider a multicell network with 21 BSs whose inter-site distance is given by 500 m. Each BS serves 20 users, uses 20 MHz bandwidth consisting of 50 RBs, and a total transmission power is 43 dBm. We generate the channel gain for each RB by using a urban macro pathloss model in [35] as $13.54 + 39.08\log_{10}(d) + 20\log_{10}(f_c) + 3$ dB, where d is the distance between the BS and user and $f_c = 2.35$ GHz. For shadowing and small-scale fading, we consider a log-normal shadowing with 6 dB standard deviation and a Rayleigh model, respectively. We set the noise spectral density to be -174 dBm/Hz. In specific, we use the modulation and coding schemes and block error rates corresponding to their signal-to-interference-plus-noise ratio (SINR) based on the mutual information effective SINR mapping model [49]. We set the discount factor, γ, to be 0.9. The simulation is run over 10000 timeslots. The simulation codes are available on our repository: https://bitbucket.org/main_lab_public/metasp-b5g

To consider different QoS requirements, we introduce four types of different traffics described in Table III. The traffic types represent different classes of services defined by referring to 5G QoS Identifier [50]. Traffic types A and B require minimum average data rate requirements with full traffic assumptions and the others require low-latency requirements. The QoS satisfaction of the traffic types with the data rate requirements is determined according to the average data rate of each user at the end of transmission and that with the low-latency requirements is determined according to the successful transmission of each packet within the delay budget. For traffic types C and D, a packet randomly arrives if there is no packet being transmitted.

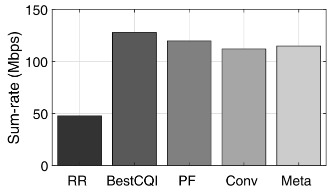
We consider following three objectives provided in IV-E: sum-rate maximization with minimum average data rate requirements, energy consumption minimization with minimum average data rate requirements, and satisfaction of various QoS requirements. From total 21 BSs in the network, 7 BSs are associated to each objective. The distance between the user and associated BS is randomly chosen among $\{100, 125, 150, 175, 200\}$ m. In addition, the traffic type of each user is randomly selected between traffic types A and B in the BSs with the sum-rate maximization and energy minimization objectives and among traffic types A, B, C, and D in the BSs with the QoS satisfaction objective. For the energy consumption minimization objective, we assume that the duration of a timeslot is 1 millisecond and set the utility parameters as $r_{alloc} = 1$. The energy consumption in each timeslot is calculated proportional to the number of allocated RBs in the timeslot. In the QoS satisfaction objective, we set the utility parameters as $r_{succ} = 10$ and $r_{fail} = -50$ for traffic types C and D.

In the proposed meta-scheduling framework, we apply the meta-scheduling policy for each objective as provided in Section IV-E. To represent each policy, we use a fully-connected neural network with 4 hidden layers each of which is composed of 500 units. It is worth noting that in the simulation, the framework manages only three neural networks since we consider three objectives. For the DQN algorithm, we set the learning rate to be 0.0001. In addition, we set the parameter ϵ-greedy to be 0.1 and the batch size to be 64. We set the training interval T^{replay} to be 5 and the target Q-network update interval T^{target} to be 10. The size of the experience collection is set to be 10000 for each BS (i.e., the size of $\mathcal{E}_l$ is set to be $10000|\mathcal{G}(l)|$, $\forall l \in \mathcal{L}$). The number of training epochs in every T^{replay} is proportional to the number of acquired experiences. To partition each state information for the meta-RL structure, we use the following partitions regardless of the objectives. The space of the CQI is uniformly partitioned into 15 subsets. For the space of the buffer size, we partition it in an exponential manner as $\{[0, 0.1), [0.1, 1), [1, 10), [10, \infty)\}$ Mbytes. Similarly, we partition the spaces of the left delay budget and the UDR as $\{[0, 2), [2, 5), [5, 10), [10, \infty)\}$ and $\{[0, 0.01), [0.01, 0.05), [0.05, 0.1), [0.1, 0.5), [0.5, 1), [1, 5), [5, \infty)\}$, respectively.

For comparison with the proposed meta-scheduling framework (Meta), we consider the following different schedulers on each BS:

1) *Round robin scheduler (RR)*: RR allocates each RB in a circular order of the user indices.
2) *Best CQI scheduler (BestCQI)*: BestCQI allocates each RB to the user who has the best CQI for the RB.
3) *Proportional fairness scheduler with URLLC (PF)*: Inspired by [51], PF first allocates RBs to the users with the latency requirements. Then, it allocates the rest of the RBs considering the fairness over the average data rates of the users. Specifically, each RB n is allocated to the user who has the largest PF metric η_u^t which is defined as follows: $\eta_u^t = \frac{g_{u,n}^t}{\tilde{G}_u^{t,wnd}}$, where $g_{u,n}^t$ is the data rate of user u with RB n in timeslot t and $\tilde{G}_u^{t,wnd}$ is the moving average of the data rate of user n in timeslot t with an average window 2000 ms.
4) *Conventional DRL-based scheduler (Conv)*: Conv allocates each RB based on the typical DRL structure in (1) and (2), which also correspond to the DRL-based schedulers in [7] and [13], where the near-optimal policy can be learned if the environment is static. However, Conv cannot be directly applied to multiple RBs due to impractical complexity from the combinatorial decisions by multiple RBs. Hence, we adopt our incremental scheduling and experience decomposition methods in Section IV-C to Conv. For fair comparison, we use the neural network structure and state information identical to Meta.

First, we consider a static scenario in which the simulation settings do not change during the simulation. In the static scenario, Conv on each BS learns the policy tailored to the


(a) Sum-rates.

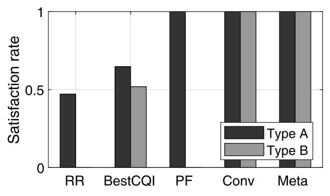
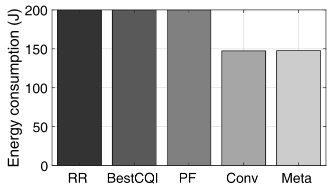

(b) QoS satisfaction rates.

Fig. 7. The performance of the schedulers on the BSs with the sum-rate maximization objective in the static scenario.


(a) Energy consumption.

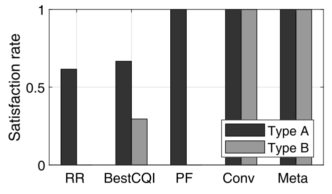

(b) QoS satisfaction rates.

Fig. 9. The performance of the schedulers on the BSs with the energy consumption minimization objective in the static scenario.

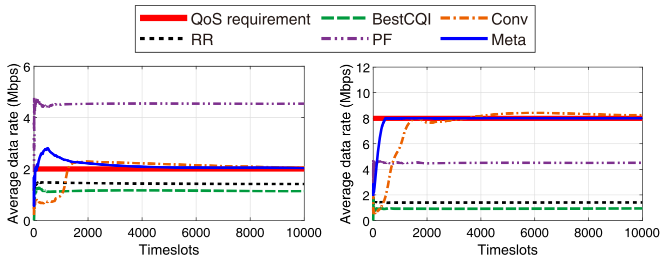

(a) Traffic type A. (b) Traffic type B.

Fig. 8. The average data rates of users that have the farthest distance from the BS with different traffic types.

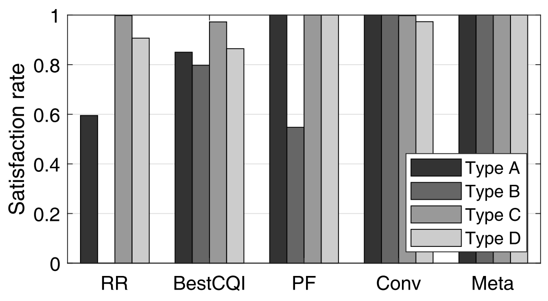


Fig. 10. The QoS satisfaction rates of the schedulers on the BSs with the QoS satisfaction objective in the static scenario.

BS without any change of environments, thereby achieving the near-optimal performance regardless of the objectives [6]. Hence, we can consider Conv as a benchmark scheduler. All the results for each objective are averaged over the BSs with the objective. In Fig. 7, we evaluate the average sum-rate and QoS satisfaction rate with the sum-rate maximization objective. As shown in Fig. 7a, BestCQI achieves the largest sum-rate, and PF, Conv, and Meta achieve the sum-rates close to that of BestCQI. On the other hand, RR achieves significantly lower sum-rate compared with other schedulers since RR does not exploit the channel diversity gain. In Fig. 7b, the QoS satisfaction rates of the users associated in the BSs are provided. Both Conv and Meta which are based on DRL address the data rate requirements well, whereas RR, BestCQI, and PF fail to satisfy the data rate requirements. In particular, RR fails to satisfy the requirements of both traffic types A and B. PF satisfies the requirement of traffic type A since fairness over users is considered. However, PF fails to support traffic type B, which has a higher data rate requirement. Moreover, BestCQI fails to satisfy the requirements of almost half of the users even though the largest sum-rate is achieved.

For further analysis, in Fig. 8, we provide the average data rates of users who have the farthest distance from the BS (200 m) with different traffic types. The users farthest from BS are the most difficult to satisfy the requirements. We can see from Fig. 8a, that PF, Conv, and Meta satisfy the requirements of traffic type A, whereas BestCQI and RR fail to satisfy. The data rate of the user with PF is high enough to satisfy the requirement because of considering fairness. On the other hand, with RR and BestCQI, the requirement of the user is not satisfied because the users farthest from BS have inferior channel condition, which incurs less chance to be allocated. In addition, Conv and Meta try to achieve the exact requirement, which is reasonable since choosing the users near the BS is more favorable to maximize the sum-rate as long as the requirement of the user is not violated. Therefore, from Figs. 7 and 8, we can see that both Conv and Meta satisfy the data rate requirements while effectively maximizing the sum-rates at the same time.

In Fig. 9, we provide the average energy consumption and QoS satisfaction rates with the energy consumption minimization objective. Fig. 9a provides the average total energy consumption of the BSs. Since the traditional schedulers simply allocate all RBs, the energy for full-power transmission is used in all timeslots. On the other hand, Conv and Meta achieve 26 % energy consumption reduction than the traditional schedulers, since Conv and Meta minimize the number of the allocated RBs. In Fig. 9b, we provide the QoS satisfaction rates of the users associated in the BSs. Similar to the result with the sum-rate maximization objective, Conv and Meta satisfy the data rate requirements well while the others fail to satisfy, even though Conv and Meta do not fully utilize the RBs for energy efficiency.

In Fig. 10, we provide the QoS satisfaction rates of the schedulers on the BSs with the QoS requirement satisfaction objective. Both Conv and Meta satisfy the QoS requirements regardless of the traffic types. PF satisfies the latency requirements of traffic types C and D well thanks to the prioritization mechanism. PF also satisfies the requirement of traffic type A thanks to considering fairness, but fails to support traffic type B for half of the users. BestCQI shows satisfaction rates of larger than or similar to 0.8 in all traffic types. However, with BestCQI, the requirements of the users far from the BS are mostly violated since such users are rarely scheduled. Hence, BestCQI incurs a critical unfairness over users. RR can satisfy the latency requirements in some degree thanks to the fair opportunity to be scheduled. However, RR still fails to satisfy the data rate requirements.

From the results in the static scenario, we can see that even though Meta learns and manages only three policies for the three objectives, Meta shows performance almost close to Conv. Besides, it is shown that the single meta-scheduling

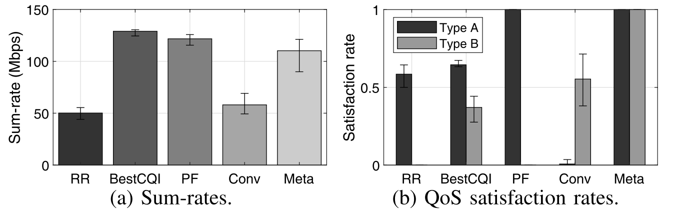


Fig. 11. The performance of the schedulers on the BSs with the sum-rate maximization objective in the dynamic scenario.

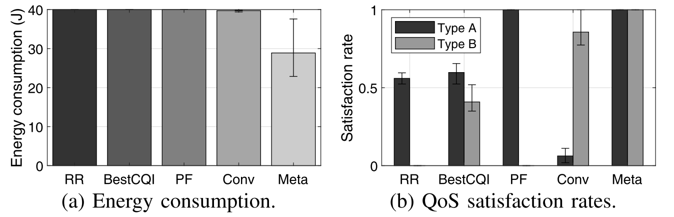


Fig. 12. The performance of the schedulers on the BSs with the energy consumption minimization objective in the dynamic scenario.

policy for each objective can be used in any BSs with the objective even though the objectives have different BS-specific characteristics. Therefore, we can conclude that the simulation results clearly show the effectiveness of the meta-scheduling framework.

Next, we consider a dynamic scenario in which the environments of all BSs vary during the simulation consisting of five periods each of which has 2000 timeslots. In specific, the environments including the objective of each BS, the distances of the users at each BS, and the traffic types of the users at each BS are randomly changed in every period. For fair comparison, we consider three policies for Conv on each BS. During each period, Conv learns and uses the policy corresponding to the objective of the period. All the results for each objective are averaged over the BSs with the objective in each period. Also, their minimum and maximum values over the periods are provided by error bars in bar plots to show the variances.

In Fig. 11, we provide the sum-rates and QoS satisfaction rates of the schedulers with the sum-rate maximization. It can be shown in Figs. 11a and 11b that the sum-rate and QoS satisfaction rate of Meta are similar to those of conventional schedulers, while Conv shows significantly lower sum-rate and QoS satisfaction rate compared with those in the static scenario. It is because traditional schedulers are designed not to be affected by the change of environments. Contrarily, the DRL-based schedulers (i.e., both Conv and Meta) typically depend on the environment because the DRL-based schedulers solve the stochastic scheduling problem specified by the environment. Therefore, it can be concluded from Fig. 11 that our proposed meta-scheduling policy can adapt to different environments well, while the conventional DRL-based scheduling policy cannot.

In Fig. 12, we provide the average total energy consumption and QoS satisfaction rates of the schedulers with the energy consumption minimization objective. As in the static scenario, Meta reduces energy consumption by about 26 % compared with the traditional schedulers. Furthermore, Meta satisfies all requirements while the other schedulers fail. Conv fails not only to reduce the energy consumption but also to satisfy the requirements. Therefore, we can conclude from Fig. 12 that Meta has capability to adapt the environmental change, whereas Conv does not have.

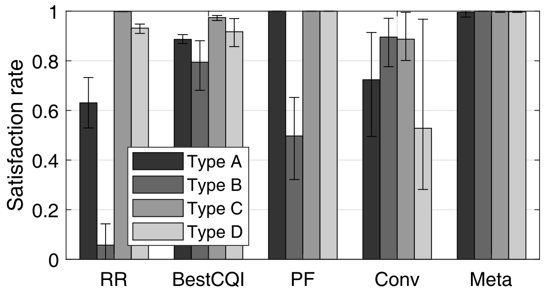


Fig. 13. The QoS satisfaction rates of the schedulers on the BSs with the QoS satisfaction objective in the dynamic scenario.

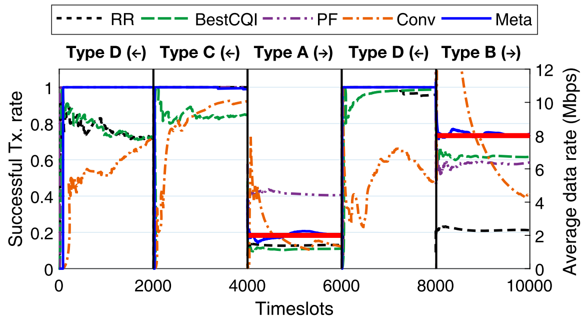
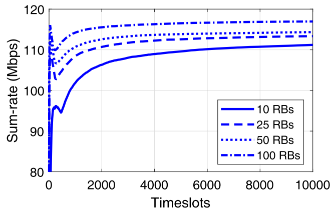


Fig. 14. The average data rate and successful transmission rate of a user at the BS with the QoS satisfaction objective in the dynamic scenario. The arrows besides the traffic types indicate the corresponding axis.

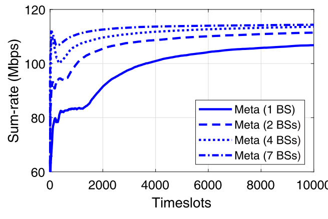
In Fig. 13, we provide the QoS satisfaction rates of the schedulers on the BSs with the QoS requirement satisfaction objective. From Fig. 13, we can see that only Meta satisfies the QoS requirements regardless of the traffic types. Similar to Figs. 11 and 12 which are evaluated in the dynamic scenario, the traditional schedulers have similar behavior of QoS satiafaction rate in both static and dynamic scenario. Conv fails to satisfy the requirements due to the change of environments.

For further analysis, in Fig. 14, we provide the average data rate and successful packet transmission rate of a user at the BS with the QoS satisfaction objective in the dynamic scenario. In each period, the distance between the user and BS is changed in order of 200, 200, 200, 175, and 150 m. Further, the traffic type is changed in order of traffic types D, C, A, D, and B. It can be shown from Fig. 14 that Meta satisfies the QoS requirements regardless of traffic types. Furthermore, Meta does not go through any learning phase after the change of the environments thanks to the cooperative learning of the policies at the CU. On the other hand, Conv fails to adapt to the change of environments. The change of the environments breaks the policy of Conv because the policy is dependent on environments.

From the simulation results in the dynamic scenario of Figs. 11–14, we can see that Meta is not affected by the change of environments. In Meta, the CU manages the meta-scheduling policies each of which can address any environment with the corresponding objective. On the other hand, Conv fails to adapt to the change of environments. As the environment changes, the target policy which Conv learns



(a) Number of RBs.



(b) Number of BSs.

Fig. 15. The sum-rates of Meta varying the simulation parameters.

changes as well. Then, an infinite number of policies are needed to enable Conv to adapt to any environment, which is impractical. In conclusion, Meta can effectively address practical cellular networks whose environment is time-varying by exploiting the CU-DU split structure.

We now show the effectiveness of Meta on learning according to the numbers of the RBs and BSs. In Fig. 15, we provide the average sum-rates with the sum-rate maximization objective, to observe the practical performance-wise learning speed. In Fig. 15a, we provide the sum-rates varying the number of RBs. For the comparison, we assume that the total bandwidth of the network is given by 20 MHz regardless of the number of RBs. From Fig. 15a, we can see that the learning speed of Meta gets faster as the number of RBs increases, since the experience decomposition method generates more experiences as the number of RBs increases. Furthermore, the sum-rates increase as well because of the diversity gain from increasing number of RBs. In Fig. 15b, we provide the sum-rates varying the number of the BSs. The learning speed of Meta gets faster as the number of the BSs increases because of cooperative learning. Therefore, Fig. 15 clearly shows that Meta learns the policy more effectively as the numbers of the RBs and the BSs increase thanks to the feature of the meta-scheduling.

VI. Conclusion

In this paper, we have proposed the meta-scheduling framework for beyond 5G networks to support diverse scheduling objectives while effectively exploiting the CU-DU split architecture in a cooperative manner. The framework learns the scheduling policies at the CU by cooperatively using the experiences from the DUs and allows each DU to utilize the scheduling policies to determine its own scheduling decisions. As the enabler of the meta-scheduling framework, we have proposed the meta-scheduling policy structure based on meta-RL and developed the meta-scheduling procedure so as to incorporate the meta-scheduling policy into practical cellular networks. We have also provided examples to implement meta-scheduling policies for several practical scheduling objectives. Through the simulations, we have shown that our meta-scheduling framework achieves performance close to the near-optimal conventional DRL-based schedulers. In addition, it can address the time-varying environments common in cellular networks while the conventional DRL-based scheduler cannot.

References

[1] *Detailed Specifications of the Terrestrial Radio Interfaces of IMT-2020*, document ITU-R M.2150-1, Feb. 2022.

[2] *F1 General Aspects and Principles (Release 17)*, document TS 38.470, 3GPP, Jun. 2022.

[3] R. Shafin, L. Liu, V. Chandrasekhar, H. Chen, J. Reed, and J. Zhang, "Artificial intelligence-enabled cellular networks: A critical path to beyond-5G and 6G," *IEEE Wireless Commun.*, vol. 27, no. 2, pp. 212–217, Apr. 2020.

[4] F. Al-Tam, N. Correia, and J. Rodriguez, "Learn to Schedule (LEASCH): A deep reinforcement learning approach for radio resource scheduling in the 5G MAC layer," *IEEE Access*, vol. 8, pp. 108088–108101, 2020.

[5] C. He, Y. Hu, Y. Chen, and B. Zeng, "Joint power allocation and channel assignment for NOMA with deep reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2200–2210, Oct. 2019.

[6] H.-S. Lee, J.-Y. Kim, and J.-W. Lee, "Resource allocation in wireless networks with deep reinforcement learning: A circumstance-independent approach," *IEEE Syst. J.*, vol. 14, no. 2, pp. 2589–2592, Jun. 2020.

[7] Y. Wei, F. R. Yu, M. Song, and Z. Han, "User scheduling and resource allocation in HetNets with hybrid energy supply: An actor-critic reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 17, no. 1, pp. 680–692, Jan. 2018.

[8] F. A. Asuhaimi, S. Bu, P. V. Klaine, and M. A. Imran, "Channel access and power control for energy-efficient delay-aware heterogeneous cellular networks for smart grid communications using deep reinforcement learning," *IEEE Access*, vol. 7, pp. 133474–133484, 2019.

[9] H. Ke, J. Wang, H. Wang, and Y. Ge, "Joint optimization of data offloading and resource allocation with renewable energy aware for IoT devices: A deep reinforcement learning approach," *IEEE Access*, vol. 7, pp. 179349–179363, 2019.

[10] X. Liao, J. Shi, Z. Li, L. Zhang, and B. Xia, "A model-driven deep reinforcement learning heuristic algorithm for resource allocation in ultra-dense cellular networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 1, pp. 983–997, Jan. 2020.

[11] N. Zhao, Y.-C. Liang, D. Niyato, Y. Pei, M. Wu, and Y. Jiang, "Deep reinforcement learning for user association and resource allocation in heterogeneous cellular networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 11, pp. 5141–5152, Nov. 2019.

[12] F. Meng, P. Chen, L. Wu, and J. Cheng, "Power allocation in multi-user cellular networks: Deep reinforcement learning approaches," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6255–6267, Oct. 2020.

[13] H.-S. Lee, D.-Y. Kim, and J.-W. Lee, "Radio and energy resource management in renewable energy-powered wireless networks with deep reinforcement learning," *IEEE Trans. Wireless Commun.*, vol. 21, no. 7, pp. 5435–5449, Jul. 2022.

[14] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," 2020, *arXiv:2004.05439*.

[15] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. ICML*, 2017, pp. 1126–1135.

[16] Z. Lin, G. Thomas, G. Yang, and T. Ma, "Model-based adversarial meta-reinforcement learning," in *Proc. NIPS*, vol. 33, 2020, pp. 10161–10173.

[17] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen, "Efficient off-policy meta-reinforcement learning via probabilistic context variables," in *Proc. ICML*, 2019, pp. 5331–5340.

[18] Y. Hua et al., "HMRL: Hyper-meta learning for sparse reward reinforcement learning problem," in *Proc. ACM SIGKDD KDD*, 2021, pp. 637–645.

[19] H.-S. Lee, "System-agnostic meta-learning for MDP-based dynamic scheduling via descriptive policy," in *Proc. AISTATS*, 2022, pp. 169–187.

[20] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "RL^2: Fast reinforcement learning via slow reinforcement learning," 2016, *arXiv:1611.02779*.

[21] J. X. Wang et al., "Learning to reinforcement learn," 2016, *arXiv:1611.05763*.

[22] K. Frans, J. Ho, X. Chen, P. Abbeel, and J. Schulman, "Meta learning shared hierarchies," in *Proc. ICLR*, 2018, pp. 1–11.

[23] L. Lan, Z. Li, X. Guan, and P. Wang, "Meta reinforcement learning with task embedding and shared policy," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 2794–2800.

[24] A. Gupta, R. Mendonca, Y. Liu, P. Abbeel, and S. Levine, "Meta-reinforcement learning of structured exploration strategies," in *Proc. NIPS*, 2018, pp. 5307–5316.

[25] R. Raileanu, M. Goldstein, D. Yarats, I. Kostrikov, and R. Fergus, "Automatic data augmentation for generalization in deep reinforcement learning," 2020, *arXiv:2006.12862*.

[26] C. Florensa, D. Held, X. Geng, and P. Abbeel, "Automatic goal generation for reinforcement learning agents," in *Proc. ICML*, 2018, pp. 1515–1528.

[27] Y. He, Y. W. Wang, Q. Lin, and J. Li, “Meta-Hierarchical Reinforcement Learning (MHRL)-based dynamic resource allocation for dynamic vehicular networks,” *IEEE Trans. Veh. Technol.*, vol. 71, no. 4, pp. 3495–3506, Apr. 2022.

[28] Y. Yuan, G. Zheng, K.-K. Wong, and K. B. Letaief, “Meta-reinforcement learning based resource allocation for dynamic V2X communications,” *IEEE Trans. Veh. Technol.*, vol. 70, no. 9, pp. 8964–8977, Sep. 2021.

[29] I. Nikoloska and O. Simeone, “Fast power control adaptation via meta-learning for random edge graph neural networks,” in *Proc. IEEE 22nd Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Sep. 2021, pp. 146–150.

[30] Y. Zou, Y. Liu, K. Han, X. Liu, and K. K. Chai, “Meta-learning for RIS-assisted NOMA networks,” in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2021, pp. 1–6.

[31] S. Zhang, “An overview of network slicing for 5G,” *IEEE Wireless Commun.*, vol. 26, no. 3, pp. 111–117, Jun. 2019.

[32] D. P. Palomar and M. Chiang, “A tutorial on decomposition methods for network utility maximization,” *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1439–1451, Aug. 2006.

[33] M. J. Neely, “Delay-based network utility maximization,” *IEEE/ACM Trans. Netw.*, vol. 21, no. 1, pp. 41–54, Feb. 2013.

[34] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed, “Federated learning: A survey on enabling technologies, protocols, and applications,” *IEEE Access*, vol. 8, pp. 140699–140725, 2020.

[35] *Study on Channel Model for Frequencies From 0.5 to 100 GHz (Release 17)*, document TR 38.901, 3GPP, Mar. 2022.

[36] V. Mnih et al., “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[37] J. Fan, Z. Wang, Y. Xie, and Z. Yang, “A theoretical analysis of deep Q-learning,” in *Proc. PMLR*, 2020, pp. 486–489.

[38] A. Ramaswamy and E. Hullermeier, “Deep Q-learning: Theoretical insights from an asymptotic analysis,” *IEEE Trans. Artif. Intell.*, vol. 3, no. 2, pp. 139–151, Apr. 2022.

[39] Y. Wang et al., “Towards ultra-high performance and energy efficiency of deep learning systems: An algorithm-hardware co-optimization framework,” in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, Feb. 2018, pp. 4235–4243.

[40] X. Liu, E. K. P. Chong, and N. B. Shroff, “A framework for opportunistic scheduling in wireless networks,” *Comput. Netw.*, vol. 41, no. 4, pp. 451–474. 2003.

[41] W. Wu, Q. Yang, B. Li, and K. S. Kwak, “Adaptive resource allocation algorithm of Lyapunov optimization for time-varying wireless networks,” *IEEE Commun. Lett.*, vol. 20, no. 5, pp. 934–937, May 2016.

[42] H.-S. Lee and J.-W. Lee, “Resource and task scheduling for SWIPT IoT systems with renewable energy sources,” *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2729–2748, Apr. 2019.

[43] J. Gong, J. S. Thompson, S. Zhou, and Z. Niu, “Base station sleeping and resource allocation in renewable energy powered cellular networks,” *IEEE Trans. Commun.*, vol. 62, no. 11, pp. 3801–3813, Nov. 2014.

[44] A. El Amine, H. A. H. Hassan, and L. Nuaymi, “Battery-aware green cellular networks fed by smart grid and renewable energy,” *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 2, pp. 2181–2192, Jun. 2021.

[45] *Study on New Radio Access Technology; Radio Access Architecture and Interfaces*, document TR 38.801, 3GPP, Mar. 2017.

[46] I. A. Alimi, A. L. Teixeira, and P. P. Monteiro, “Toward an efficient C-RAN optical fronthaul for the future networks: A tutorial on technologies, requirements, challenges, and solutions,” *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 708–769, 1st Quart., 2018.

[47] *Network Slicing*, Samsung Electron., Suwon-s, South Korea, Apr. 2020. [Online]. Available: https://images.samsung.com/is/content/samsung/assets/global/business/networks/insights/white-paper/network-slicing/200420_Samsung_Network_Slicing_Final.pdf

[48] S. Niknam et al., “Intelligent O-RAN for beyond 5G and 6G wireless networks,” 2020, *arXiv:2005.08374*.

[49] S. Lagen, K. Wanuga, H. Elkotby, S. Goyal, N. Patriciello, and L. Giupponi, “New radio physical layer abstraction for system-level simulations of 5G networks,” in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–7.

[50] *System Architecture for the 5G System (5GS); Stage 2*, document TS 23.501, 3GPP, Jun. 2022.

[51] H. Yin, L. Zhang, and S. Roy, “Multiplexing URLLC traffic within eMBB services in 5G NR: Fair scheduling,” *IEEE Trans. Commun.*, vol. 69, no. 2, pp. 1080–1093, Feb. 2021.

Kyungsik Min received the B.S. and Ph.D. degrees in electrical and electronic engineering from Yonsei University, Seoul, South Korea, in 2011 and 2017, respectively. From 2017 to 2023, he was a Staff Engineer with Samsung Electronics, working on the physical and MAC layer design for 5G systems. Since 2023, he has been an Assistant Professor with the Department of Information and Telecommunications Engineering, The University of Suwon, South Korea. His research interests include 5G/6G communication systems, massive MIMO systems, full-duplex radio, and mmWave communications. He was a recipient of the Global Ph.D. Fellowship from the National Research Foundation of Korea in 2011.

Yunjoo Kim received the B.S. and M.S. degrees in computer science and engineering from Ewha Womans University, Seoul, South Korea, in 2000 and 2002, respectively. Since 2002, she has been with the Electronics and Telecommunications Research Institute (ETRI), Daejeon, Republic of Korea, where she is currently a Principal Researcher. Her research interests include 5G/6G wireless communication systems and intelligent communication systems.

Hyun-Suk Lee received the B.S. and Ph.D. degrees in electrical and electronic engineering from Yonsei University, Seoul, South Korea, in 2012 and 2018, respectively. He was a Post-Doctoral Researcher with the School of Electrical and Electronic Engineering, Yonsei University, from 2018 to 2020, and the Department of Applied Mathematics and Theoretical Physics, University of Cambridge, U.K., from September 2019 to August 2020. Since September 2020, he has been with the School of Intelligent Mechatronics Engineering, Sejong University, Seoul, where he is currently an Assistant Professor. His research interests include machine learning and communication networks.