

Learning from Peers: Transfer Reinforcement Learning for Joint Radio and Cache Resource Allocation in 5G Network Slicing

Hao Zhou, Melike Erol-Kantarci, *Senior Member, IEEE*, Vincent Poor, *Fellow, IEEE*

Abstract—Radio access network (RAN) slicing is an important part of network slicing in 5G. The evolving network architecture requires the orchestration of multiple network resources such as radio and cache resources. In recent years, machine learning (ML) techniques have been widely applied for network slicing. However, most existing works do not take advantage of the knowledge transfer capability in ML. In this paper, we propose a transfer reinforcement learning (TRL) scheme for joint radio and cache resources allocation to serve 5G RAN slicing. We first define a hierarchical architecture for the joint resources allocation. Then we propose two TRL algorithms: Q-value transfer reinforcement learning (QTRL) and action selection transfer reinforcement learning (ASTRL). In the proposed schemes, learner agents utilize the expert agents' knowledge to improve their performance on target tasks. The proposed algorithms are compared with both the model-free Q-learning and the model-based priority proportional fairness and time-to-live (PPF-TTL) algorithms. Compared with Q-learning, QTRL and ASTRL present 23.9% lower delay for Ultra Reliable Low Latency Communications slice and 41.6% higher throughput for enhanced Mobile Broad Band slice, while achieving significantly faster convergence than Q-learning. Moreover, 40.3% lower URLLC delay and almost twice eMBB throughput are observed with respect to PPF-TTL.

Index Terms—5G, network slicing, edge caching, transfer reinforcement learning

I. INTRODUCTION

Driven by the increasing traffic demand of diverse mobile applications, the 5G is expected to satisfy various quality of service (QoS) requirements of a wide variety of services as well as service level agreements of different user types [1]. Considering the diverse QoS demands of user types such as enhanced Mobile Broad Band (eMBB) and Ultra Reliable Low Latency Communications (URLLC), network slicing has been proposed to enable flexibility and customization of 5G networks. Based on software defined network and network function virtualization techniques, physical network devices are split into multiple logical network slices [2]. Each slice may include its own controller to manage available resources.

This work was supported by Natural Sciences and Engineering Research Council of Canada (NSERC), Collaborative Research and Training Experience Program (CREATE) under Grant 497981 and Canada Research Chairs Program.

H. Zhou and M. Erol-Kantarci are with the School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, ON K1N 6N5, Canada. (emails: {hzhou098, melike.erolkantarci}@uottawa.ca).

H. V. Poor is with the Department of Electrical and Computer Engineering, Princeton University, Princeton, NJ 08544 USA (e-mail: poor@princeton.edu).

As an important part of network slicing, radio access network (RAN) slicing is more complicated than core and transport network slicing due to limited bandwidth resources and fluctuating radio channels. For instance, an earliest-deadline-first scheduling method is proposed in [3] to support 5G RAN slicing, and it shows an improved customization and resource utilization performance. A two-layer slicing approach is introduced in [4] for efficient and low complexity RAN slicing, which aims to find a suitable trade-off between slice isolation and efficiency in dynamic radio resource allocation. Whereas [5] defines a novel online convex optimization framework for inter-slice radio resource allocation, and the proposed method learns resource allocation from previously observed data.

The aforementioned works mainly concentrate on the RAN slicing. While the bandwidth of a base station (BS) is indisputably the most critical resource of RAN network, other resources are equally important to guarantee the network performance, especially cache resource [6]. Incorporating caching into the RAN has attracted interest from both academia and industry, and some novel network architectures have been proposed to harvest the potential advantages of edge caching [7], [8]. Indeed, edge caching allows storing data closer to the users, by utilizing the storage capacity available at the network devices, and reduces the traffic to the core network. Moreover, caching can save the backhaul capacity without affecting the network delay. Cache placement and caching strategies have been extensively studied in numerous works, but the problem is usually addressed disjointly, without considering the RAN. For example, a location customized caching method is presented in [9] to maximize the caching hit rate, and a linear model is deployed to estimate the future content hit rate. The content caching locations are optimized in [10], where both cloud-centric and edge-centric caching are considered for optimization.

Integrating RAN with caching will bring significant improvements for 5G networks. However, it leads to great complexity for network management. Network slicing is expected to allocate limited resources such as bandwidth or caching capacity between slices, and fulfill the QoS requirements of slices. The complex network dynamics, especially the stochastic arrival requests of slices, make the underlying network optimization challenging. Fortunately, machine learning (ML) techniques offer promising solutions [11]. For instance, by applying a reinforcement learning (RL) scheme, the potential

complexity of defining a dedicated optimization model is avoided. Combined with a low-complexity heuristic algorithm, RL is applied in [12] to service eMBB and vehicle-to-everything slices. Communication and computation resources are jointly considered in [13] for 5G RAN slicing, and Q-learning is deployed to maximize the network utility. Moreover, deep Q-learning is used in [14] for end-to-end network slicing, and double deep Q-learning is deployed in [15] for computation offloading within sliced RANs.

Although learning-based methods such as RL and deep reinforcement learning (DRL), have been generally applied for network resource allocation, most existing works do not consider the possibility of knowledge transfer [12]–[15]. Specifically, an agent is designed for a specific task in these works, and it interacts with its environment from scratch, which usually leads to a lower exploration efficiency and longer convergence time. Whenever a new task is assigned, the agent needs to be retrained, even though similar tasks have been completed before. The poor generalization capability of straightforward RL methods motivate us to find a learning method with better generalization and knowledge transfer capability.

In this work, we propose two transfer reinforcement learning (TRL) based solutions for joint allocation of radio and cache resources. The first method is Q-value transfer reinforcement learning (QTRL) and the second technique is called action selection transfer reinforcement learning (ASTRL). Using these schemes, agents can utilize the knowledge of experts to improve their performance on current tasks. Furthermore, the current network optimization schemes are usually defined in a centralized way [12]–[14]. This leads to excessive control overhead where processing the specific requests of all devices can be a heavy burden for the central controller. To this end, we propose a hierarchical architecture for the joint resource allocation. The global resource manager (GRM) is responsible for inter-slice radio and caching capacity allocation. Then the slice resource managers (SRMs) will distribute radio resources among the attached user equipment (UEs) and replace the cached contents based on its own need.

The main contributions of this work are: (1) We define a hierarchical architecture for joint radio and cache resources allocation of cellular networks. The GRM will intelligently allocate resources between slices, then SRMs will distribute radio and cache resources within each slice based on specific rules. The proposed hierarchical architecture can reduce the control overhead and achieves a higher flexibility.

(2) We propose two TRL-based solutions for the inter-slice resource allocation, namely QTRL and ASTRL. QTRL can utilize the Q-values of experts as prior knowledge for an improved learning process, while ASTRL focuses on action selection knowledge. Compared with RL or DRL, the proposed TRL solutions have a better knowledge transfer capability.

(3) Finally, the TRL solutions are compared with the model-free Q-learning, the model-based priority proportional fairness and time-to-live (PPF-TTL) algorithms via simulations. The results demonstrate that TRL-based algorithms achieve better

performance in both network and ML metrics. In particular, QTRL and ASTRL have 23.9% lower URLLC delay and 41.6% higher eMBB throughput than Q-learning. The simulations also show 40.3% lower URLLC delay and almost doubled eMBB throughput than the PPF-TTL method. QTRL and ASTRL also outperform Q-learning with a significantly faster convergence.

The rest of this work is organized as follows. Section II presents related work, and Section IV shows the system model and problem formulation. Section V defines the TRL-based resource allocation scheme. The simulations are shown in Section VI, and Section VII conclude this work.

II. RELATED WORK

Recently there have been numerous studies to apply artificial intelligence (AI) techniques for the resource allocation of 5G networks. Content caching and mode selection are jointly considered in [16] to minimize the system power consumption, and two RL algorithms are developed to solve the high complexity optimization problem. [17] also jointly considers the content caching and mode selection problem, which applies DRL for decision making of the cloud server to satisfy the requirements of slices. The multi-tenant radio resource allocation problem is investigated in [18], and DRL is leveraged to optimize the computation offloading and packet scheduling. A multi-agent DRL algorithm is proposed in [19] for the cooperative content caching of small base stations, and each agent can intelligently cache contents under time-varying popularity. The radio, caching and computation resources are jointly considered in [20], and an actor-critic DRL algorithm is used to minimize the end-to-end delay.

The aforementioned works show that various ML methods have been applied for the resource management of wireless networks, including RL [12], [13], [16], [17], DRL [14], double deep Q-learning [15], multi-agent RL [18], multi-agent DRL [19], actor-critic DRL [20], and so on. In our former work [21], we proposed a correlated Q-learning based method for the radio resource allocation of 5G RAN, but the knowledge transfer was still not considered. In these works, the main motivations of deploying ML algorithms are the increasing complexity of wireless networks and the difficulties to build dedicated optimization models. Indeed, the evolving network architecture, emerging new network performance requirements and increasing device numbers make the traditional methods such as convex optimization more and more complicated. ML, especially RL, offers a good opportunity to reduce the optimization complexity and make data-driven decisions. However, a large number of samples are needed for the learning process, which means long training time and low system efficiency. Furthermore, after the long learning process, the agent can only handle one specific task without any generalization. This learning process has to be repeated when a new task is assigned. DRL can be considered as a solution to reduce the number of samples and improve convergence. However, it is still limited in a

TABLE I
COMPARISON OF RL, TL, DRL AND TRL

Algorithms	Features	Difficulties	Applications
RL	Agent has no prior knowledge about tasks. It explores new tasks from scratch.	Long convergence time, low exploration efficiency.	Tasks with limited state-action space and no prior knowledge.
TL	Improving generalization across different distributions between source and target tasks.	Negative transfer, automation of task mapping.	It is mainly designed for the supervised learning domain, e.g., classification, regression.
DRL	Combining artificial neural networks with RL architecture. Applying neural networks to estimate state-action values.	Time-consuming network training and tuning; Low sample and exploration efficiency; Training stability.	Large state-action space or continuous-action problem.
TRL	Utilizing prior knowledge from experts to improve performance of learners such as higher average reward or faster convergence.	Transfer function needs to be defined to digest prior knowledge.	Optimization of tasks with existing prior knowledge.

localized domain, and the neural network training can be time-consuming because of the hyperparameter tuning. To this end, we propose TRL-based solutions for the joint radio and cache resources allocation of 5G RAN. The TRL has been applied in [22] for intra-beam interference management of 5G mm-Wave, but here, the resource allocation problem is more complicated due to a much bigger state-action space. The proposed scheme has a satisfying knowledge transfer capability by utilizing the knowledge of expert agents, and it outperforms RL by faster convergence.

III. BACKGROUND

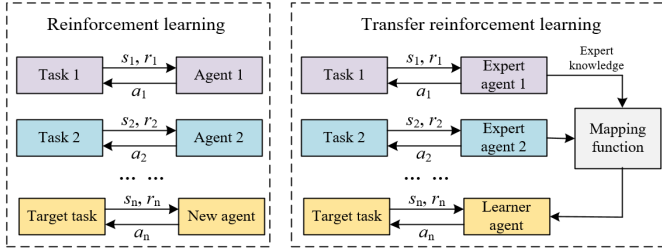


Fig. 1. Comparison of TRL and RL.

In this section, we introduce the background of TRL to distinguish it from RL. As shown in Fig.1, given a task pool, the interactions between tasks and agents can be described by the Markov decision process (MDP) $\langle S, A, R, T \rangle$, where S is the state space, A is the action space, R is the reward function, and T is the transition probability. In RL, each agent works independently to try different actions, then it arrives new states and receives rewards. The learning phase of RL can be defined as:

$$\mathcal{L}_{RL} : s \times \mathcal{K} \rightarrow a, r (a \in A), \quad (1)$$

where the \mathcal{K} is the knowledge of this agent, s is the current state, a is the selected action, and r is the reward. Equation (1) indicates that RL agent utilizes the collected knowledge to select action a and receive reward r under state s .

On the contrary, TRL includes two phases: knowledge transfer phase and learning phase. In the knowledge transfer phase, as shown in Fig.1, considering task differences, a

mapping function is defined to make the knowledge of expert agents digestible for the learner agent. Then the learner agent explores target tasks on its own and forms its own knowledge. The whole process of TRL can be defined as:

$$\mathcal{L}_{TRL} : s \times \mathcal{M}(\mathcal{K}_{expert}) \times \mathcal{K}_{learner} \rightarrow a, r (a \in A), \quad (2)$$

where the \mathcal{M} is the mapping function, \mathcal{K}_{expert} is the knowledge from experts, $\mathcal{K}_{learner}$ is the knowledge of learner agent. In equation (2), knowledge from experts are utilized in the action selection of learner agent, and it is expected to accelerate the learning process \mathcal{L}_{TRL} . In TRL, new tasks can be better handled based on knowledge of experts.

The RL, transfer learning (TL), DRL and TRL techniques are compared in Table I. Compared with RL, TRL has higher exploration efficiency and better generalization capability [23]. On the other hand, although DRL is a breakthrough approach by combining neural networks with RL scheme, the time-consuming network training is a well known issue, and the training stability as well as generalization capability of DRL can cause problems [24]. Finally, although TL has been extensively studied in the ML literatures, it mainly focuses on the supervised learning domain such as classification and regression [25]. Compared with TL, TRL can be more complicated because the knowledge needs to be transferred in the context of MDP scheme. Moreover, due to the dedicated components of MDP, the knowledge may exist in different forms, which needs to be transferred in different ways [26].

IV. SYSTEM MODEL AND PROBLEM FORMULATION

A. Overall Architecture

Fig.2 presents the proposed radio and cache resources allocation scheme, and a hierarchical architecture is defined for the resource allocation of cache-enabled macro cellular networks. We assume the BS has the caching capability to store contents. Firstly, the SRMs collect the QoS requirements of attached UEs, and the collected information is sent to GRM. Then the GRM will intelligently implement the inter-slice allocation to divide the radio and cache resources between eMBB and URLLC SRMs. Finally, SRMs distribute resource blocks (RBs) to attached UEs, and update cached contents

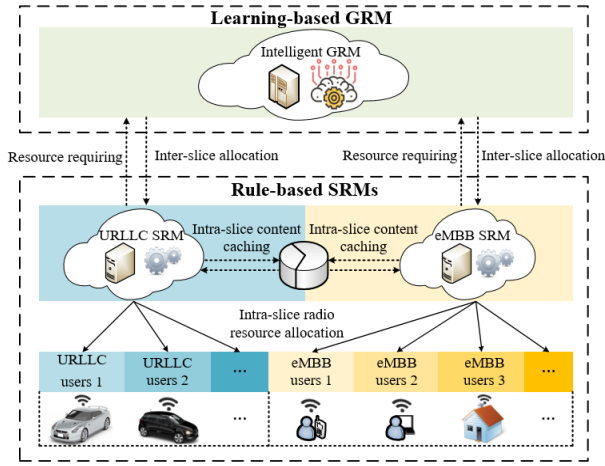


Fig. 2. Proposed radio and cache resources allocation scheme

within the allocated caching capacity. We apply learning-based methods to realize an intelligent GRM, while rule-based methods are deployed for SRMs. This hierarchical architecture can alleviate the burden of GRM, since the GRM only accounts for slice-level allocation, instead of handling all the UEs directly.

B. Communication Model

In this section, we introduce the communication model. Firstly, the total delay consists of:

$$d_{j,m,g} = d_{j,m,g}^{tx} + d_{j,m,g}^{que} + d_{j,m,g}^{rtx} + (1 - \beta_{j,g})d^{back}, \quad (3)$$

where $d_{j,m,g}^{tx}$ is the transmission delay of content g from the j^{th} BS to the m^{th} UE, $d_{j,m,g}^{que}$ and $d_{j,m,g}^{rtx}$ are the queuing delay and the re-transmission delay, respectively. $d_{j,m,g}^{back}$ is the backhaul delay of fetching contents from the core network. $\beta_{j,g}$ is a binary variable. $\beta_{j,g} = 1$ when the content g is cached at the j^{th} BS, and $\beta_{j,g} = 0$ otherwise. Equation (3) shows that the total delay is affected by both communication and cache resources. The transmission delay depends on radio resource allocation, and edge caching can prevent the backhaul delay. The scheduling efficiency will affect the queuing delay.

The transmission delay depends on the link capacity between the BS and UE:

$$d^{tx} = \frac{L_m}{C_{j,m}}, \quad (4)$$

where L_m is the content size required by the m^{th} UE. $C_{j,m}$ is the link capacity between the j^{th} BS and the m^{th} UE, which is calculated as follows:

$$C_{m,j} = \sum_{r \in N_m^R} b_r \log \left(1 + \frac{p_{j,r} x_{j,r,m} g_{j,r,m}}{b_r N_0 + \sum_{j' \in J-j} p_{j',r'} x_{j',r',m'} g_{j',r',m'}} \right), \quad (5)$$

where N_m^R is the set of RBs that are allocated to the m^{th} UE, b_r is the bandwidth of the r^{th} RB, N_0 is the noise power density, $p_{j,r}$ is the transmission power of the r^{th} RB of j^{th} BS, $x_{j,m,r}$ is a binary indicator to denote whether the r^{th} RB is allocated to the m^{th} UE in the j^{th} BS, $g_{j,m,r}$ is the

channel gain between the BS and UE, and $j' \in J-j$ is the BS set except j^{th} BS.

C. Slicing-based Caching Model

We will introduce the slicing-based caching model in this section, in which the time-to-live (TTL) method is used as the content replacement strategy. The TTL indicates the time that a content is stored in a caching system before it is deleted or replaced. TTL will be reset if this content is required again, and thus popular contents will live longer. Although there have been many content replacement strategies, TTL is selected because: i) this work mainly focus on the inter-slice level resource allocation, and it is reasonable to apply a well-known caching method for intra-slice caching; ii) TTL requires no prior knowledge for content popularity, which is more realistic. Nevertheless, our proposed architecture is compatible with any other caching methods without loss of generality. The complexity of slicing-based caching model lies in how to effectively divide the limited caching capacity between slices, which is far more complicated than original TTL model.

We use \mathcal{N} to represent the slice set, and each slice n contains $|\mathcal{M}_n|$ UEs with each UE is denoted by m ($m \in \mathcal{M}_n$). We assume each slice has its own content library \mathcal{G}_n , and g represent the content ($g \in \mathcal{G}_n$). The content requiring rate of UE m in slice n is $\phi_{n,m}$, and $\phi_{n,m,g}$ is the content requiring rate for content g . Then we have $\phi_{n,m,g} = p_{n,m,g} \phi_{n,m}$, in which $p_{n,m,g}$ is content require distribution of UE m in slice n for content g ($\sum_{g \in \mathcal{G}_n} p_{n,m,g} = 1$). The caching hit probability of content g in slice n is [27]:

$$h_{n,g} = 1 - e^{-\sum_{m \in \mathcal{M}_n} \phi_{n,m,g} T_{n,m,g}}, \quad (6)$$

where the TTL of content g will be reset to $T_{n,m,g}$ when this content is demanded. Note that the number of contents is large and the requiring rate of each content is relatively small. Based on $\lim_{x \rightarrow 0} e^x = x + 1$, we approximately have:

$$h_{n,g} = \sum_{m \in \mathcal{M}_n} \phi_{n,m,g} T_{n,m,g}, \quad (7)$$

Meanwhile, the total caching hit probability is related with the allocated storing capacity of slice n :

$$\sum_{g \in \mathcal{G}_n} \sum_{m \in \mathcal{M}_n} h_{n,m,g} = \frac{C_n}{C_T}, \quad (8)$$

where C_n is the allocated storing capacity for slice n , C_T is the total storing capacity of this BS, $h_{n,m,g}$ is the hit probability of UE m for content g . We assume the content g has the same popularity for different UEs and thus $T_{n,m,g} = T_{n,g}$. Given equation (7) and (8), we have:

$$h_{n,g} = \frac{C_n}{C_T} \frac{\sum_{m \in \mathcal{M}_n} \phi_{n,m,g}}{\sum_{m \in \mathcal{M}_n} \phi_{n,m}}, \quad (9)$$

Equation (9) indicates that a higher storing capacity C_n will lead to a higher caching hit probability $h_{n,g}$ [28]. The popular contents, which is indicated by a higher required rate $\phi_{n,m,g}$

has a higher caching hit probability. Then the hit rate can be calculated by:

$$\phi_{n,m}^{hit} = \sum_{g \in \mathcal{G}_n} \phi_{n,m,g} h_{n,m,g}, \quad (10)$$

and the cache miss rate is: $\phi_{n,m}^{miss} = \phi_{n,m} - \phi_{n,m}^{hit}$.

Finally, the backhaul delay is only applied when a content is not cached at the BS. The backhaul service is assumed to obey the M/M/1 queue, the average delay is [29]:

$$d^{back} = \frac{1}{\frac{B}{L_s} - \sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{M}_n} (\phi_{n,m}^{miss})}, \quad (11)$$

where B is the backhaul capacity, B/L_s denotes the service rate, and $\sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{M}_n} (\phi_{n,m}^{miss})$ is the total miss rate. Note that d^{back} will be infinite if miss rate is higher than service rate.

D. Global Resource Manager

In this work, we consider two typical slices, namely eMBB slice and URLLC slice. It is assumed that the contents required by two slices are not overlapping. For the eMBB slice, the objective is to maximize the total throughput, while the URLLC slice aims to minimize the delay. To balance the requirements of two slices, the GRM needs to allocate the radio and cache resources between two slices by:

$$\max \sum_{m \in \mathcal{M}_j^{embb}} b_{j,m} \quad \text{and} \quad \min \sum_{u \in \mathcal{M}_j^{urllc}} d_{j,u} \quad (12)$$

$$\text{s.t.} \quad (3) \ (5) \ (9) \ (10) \ \text{and} \ (11) \quad (12a)$$

$$N_j^{embb} + N_j^{urllc} \leq N_j^T \quad (12b)$$

$$C_j^{embb} + C_j^{urllc} \leq C_j^T \quad (12c)$$

$$\phi^{embb} + \phi^{urllc} \leq \frac{L_s}{B} \quad (12d)$$

where $b_{j,m}$ is the throughput of UE m in the eMBB slice of j^{th} BS, and $d_{j,u}$ is the delay of UE u of the URLLC slice. \mathcal{M}_j^{embb} and \mathcal{M}_j^{urllc} are the set of UEs in eMBB and URLLC slices, respectively. N_j^{embb} and N_j^{urllc} are the number of RBs that GRM allocated to eMBB and URLLC slices, respectively. N_j^T is the total number of RBs of j^{th} BS. C_j^{embb} and C_j^{urllc} are the content caching capacity of eMBB and URLLC slices, respectively, and C_j^T is the total caching capacity in j^{th} BS. ϕ^{embb} and ϕ^{urllc} are the cache miss rate of eMBB and URLLC slices, respectively.

Equation (12) shows that GRM will jointly consider the objectives of eMBB and URLLC slices; (12b) is the radio resources constraint; (12c) ensures the allocated caching capacity cannot exceed the capacity limit; (12d) denotes that the total miss rate should not exceed the backhaul service rate. In the GRM, control variables are N_j^{embb} , N_j^{urllc} , C_j^{embb} and C_j^{urllc} , which means GRM only accounts for the inter-slice radio and cache resources allocation. Note that no specific UE is involved, which will reduce the control overhead and computation burden of GRM.

E. Slice Resource Manager

The problem formulations of SRMs are defined in this section. Specifically, SRMs distribute radio resources to UEs, and replaces cached contents within the slice. Considering all UEs in the same slice are equally important, we apply a proportional fairness algorithm for the radio resource allocation [32]. Meanwhile, the TTL method is still used for content replacement. The rule-based intra-slice resource allocation of eMBB slice can be described by:

$$\arg \max_{m \in \mathcal{H}} \frac{o_m}{(\sum_{t-\Delta t}^t b_{t,m})/\Delta t} \quad (13)$$

$$\text{s.t.} \quad \sum_{m \in \mathcal{M}_j^{embb}} x_{j,r,m} \leq 1 \quad (13a)$$

$$\sum_{r \in N_j} \left(\sum_{m \in \mathcal{M}_j^{embb}} x_{j,r,m} \right) \leq N_j^{embb} \quad (13b)$$

$$\sum_{g \in \mathcal{G}^{embb}} \beta_{j,g} \leq C_j^{embb} \quad (13c)$$

$$\sum_{m \in \mathcal{M}_j^{embb}} \phi_m^{miss} = \phi^{embb} \quad (13d)$$

where \mathcal{H} is the transmission queue and o_m is the estimated transmission rate of UE m , and $(\sum_{t-\Delta t}^t b_{t,m})/\Delta t$ is the average throughput in the past Δt time interval. N_j is the set of the total RBs in the j^{th} BS, \mathcal{G}^{embb} is the content library of the eMBB slice, ϕ_m^{miss} is the miss rate of the UE m .

Equation (13) means each RB will choose a transmitting UE by proportional fairness. Constraint (13a) indicates that each RB can only be allocated to at most one UE; (13b) and (13c) show that available radio and cache resources cannot exceed the capacity allocated by GRM (indicated by N_j^{embb} and C_j^{embb}). $\beta_{j,g}$ is a binary variable that is defined in equation (3) to represent whether a content is cached. Constraint (13d) means the total miss rate of eMBB slice consists of the miss rate of all UEs in this slice.

Similarly, for URLLC slice, we have:

$$\arg \max_{u \in \mathcal{H}} \frac{o_u}{(\sum_{t-\Delta t}^t b_{t,u})/\Delta t} \quad (14)$$

$$\text{s.t.} \quad \sum_{u \in \mathcal{M}_j^{urllc}} x_{j,r,u} \leq 1 \quad (14a)$$

$$\sum_{r \in N_j} \left(\sum_{u \in \mathcal{M}_j^{urllc}} x_{j,r,u} \right) \leq N_j^{urllc} \quad (14b)$$

$$\sum_{g \in \mathcal{G}^{urllc}} \beta_{j,g} = C_j^{urllc} \quad (14c)$$

$$\sum_{u \in \mathcal{M}_j^{urllc}} \phi_u^{miss} = \phi^{urllc} \quad (14d)$$

where \mathcal{G}^{urllc} is the content library of URLLC slice. Equation (13) still denotes the proportional fairness principle; constraint (14a) and (14b) are radio resource constraints; constraint (14c) shows that the total number of cached content will not exceed the capacity limit; (14d) is the miss rate constraint.

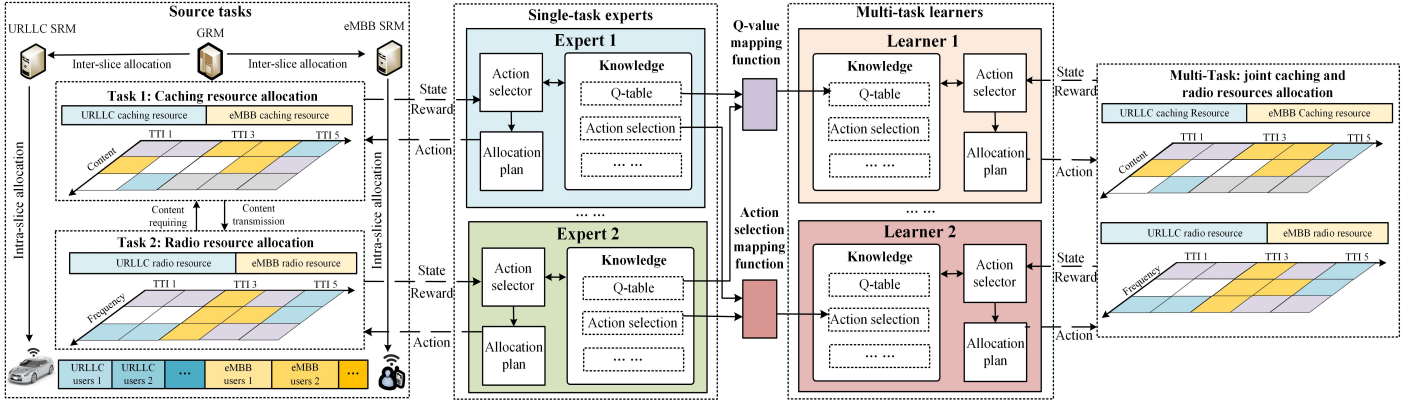


Fig. 3. Proposed transfer reinforcement learning architecture for resource management.

TABLE II
STRATEGIES AND MDP DEFINITIONS OF ALGORITHMS

Indices	Strategies	States	Actions	Instant Reward
Expert 1	Q-learning based radio resource allocation ; No caching capability.	(s^{embb}, s^{urllc}) , where s^{embb} denotes the number of eMBB contents in the queue, and s^{urllc} is defined similarly.	(n^{embb}, n^{urllc})	$r = wr^{embb} + (1-w)r^{urllc}$, where $r^{embb} = \begin{cases} \frac{2}{\pi} \tan^{-1}(\sum_{m \in \mathcal{M}^{embb}} b_m), & \mathcal{H}^{embb} \neq 0 \\ 0, & \mathcal{H}^{embb} = 0 \end{cases}$ $r^{urllc} = \begin{cases} 1 - \sum_{u \in \mathcal{M}^{urllc}} d_u, & \mathcal{H}^{urllc} \neq 0, \\ 0, & \mathcal{H}^{urllc} = 0, \end{cases}$ where w is the weighting factor, \mathcal{H}^{embb} and \mathcal{H}^{urllc} are the queues of eMBB and URLLC slices, respectively.
Expert 2	Fixed radio resource allocation; Q-learning based cache resource allocation.		(c^{embb}, c^{urllc})	
Learner 1	TRL-based joint radio and cache resources allocation with the prior knowledge of Q-tables of experts.		$(n^{embb}, n^{urllc}, c^{embb}, c^{urllc})$, where n^{embb} and c^{embb} denote the number of RBs and caching capacity allocated to the eMBB slice, respectively. n^{urllc} and c^{urllc} are defined similarly for the URLLC slice.	
Learner 2	TRL-based joint radio and cache resources allocation with prior knowledge of action selections of experts.			
Baseline	Q-learning for joint radio and cache resources allocation without any prior knowledge.			

In the SRMs, we combine the proportional fairness algorithm with TTL method for the intra-slice resource allocation, and SRMs are only responsible for intra-slice allocation by these rules.

V. TRANSFER REINFORCEMENT LEARNING BASED RESOURCE ALLOCATION

In this section, we will introduce the TRL based inter-slice resource allocation, which is designed for the intelligent GRM. Fig.3 shows the proposed TRL architecture for joint radio and cache resources allocation. We assume there are two single-task experts, and each expert has specific knowledge for a single task only, e.g., radio or cache resource allocation. The learners can utilize the experts' knowledge of single task to improve their multi-task performance. For example, the learner 1 in Fig.3 can utilize the Q-tables knowledge of expert 1 and 2 to improve its own learning performance on joint resources allocation, while learner 2 prefers the action selection knowledge of experts. However, due to different state and action definitions, the knowledge cannot be applied directly, and mapping functions are necessary to make the original knowledge digestible for learners.

As shown in Table II, five different strategies are deployed for the radio and cache resources allocation. Experts are only

good at one of radio or cache resource allocation. The learners are presumed to utilize various knowledge from experts to improve their own performance on joint radio and cache resources allocation. The classic Q-learning is implemented as a baseline without any prior knowledge. In the following, we will introduce the experts, learners and baselines.

A. Experts Definition

For a complicated resource allocation problem, it is reasonable to deploy the learning-based methods partially to reduce the potential complexity of the system. These experts have some learning experience of one specific task, but they have no knowledge of other tasks. For expert 1, it uses Q-learning for radio resource allocation, and there is no caching capability. For expert 2, RBs are allocated by the UE numbers of each slice, and Q-learning is used for caching capacity allocation. The MDP definitions of experts are:

- **State:** The states of both experts are defined as (s^{embb}, s^{urllc}) , which indicates the number of required contents waiting in the queues of the eMBB and URLLC slices, respectively.
- **Action:** In expert 1, considering the Q-learning is only applied for radio resource allocation, the action (n^{embb}, n^{urllc}) means the number of RBs allocated to

the eMBB and URLLC slices. Meanwhile, the action of expert 2 is (c^{emb}, c^{urllc}) , which indicates the caching capacity allocation.

- **Reward:** The reward is defined by the objectives of slices. For eMBB slice, obtaining higher throughput leads to a higher reward. For URLLC slice, lower delay means a higher reward. A weighting factor is used to balance two rewards and form an overall reward. We also apply a penalty when packets are dropped to guarantee the packet drop rate (PDR).

With Q-learning, the agent aims to maximize the long-term expected reward:

$$V(s) = \mathbb{E}_\pi \left(\sum_{n=0}^{\infty} \gamma^n r(s_n, a_n) | s = s_0 \right), \quad (15)$$

where $V(s)$ is the long-term expected accumulated reward at state s , s_0 is the initial state, $r(s_n, a_n)$ denotes the reward of selecting action a_n at state s_n , and γ is the discount factor ($0 < \gamma < 1$).

Then state-action value is updated by:

$$Q^{new}(s_e, a_e) = (1 - \alpha)Q^{old}(s_e, a_e) + \alpha(r + \gamma \max_{a_e'} Q(s_e', a_e')), \quad (16)$$

where $Q^{old}(s_e, a_e)$ and $Q^{new}(s_e, a_e)$ are old and new Q-values, s_e and s_e' are current and next states of experts, a_e is the action, r is the reward, and α is the learning rate ($0 < \alpha < 1$). By updating the Q-values iteratively, the experts will learn the optimal action selections to achieve the best accumulated reward.

B. Learner: Transfer Reinforcement Learning

Based on the prior knowledge, learners are expected to solve more complicated cross domain problems with higher state-action space. Compared with allocating one single resource, the joint resources allocation problem is much more complicated, especially when multiple slices are involved. Considering the Q-tables of learners are much bigger than experts, it will unavoidably lead to a longer convergence time, even sub-optimal results. To this end, we introduce QTRL and ASTRL. The MDP definitions of two algorithms are presented in Table II.

1) Learner 1: Q-value Transfer Reinforcement Learning

In QTRL, the experts' Q-values are presumed to be the prior knowledge of learner 1, which can guide the exploration of learner agent and improve the exploration efficiency [30]. Q-values in QTRL are updated by:

$$Q^{new}(s_{l,1}, a_{l,1}) = \sigma Q^T(\mathcal{F}(s_{l,1}), \mathcal{F}'(a_{l,1})) + (1 - \alpha)Q^{old}(s_{l,1}, a_{l,1}) + \alpha(r + \gamma \max_{a_{l,1}'} Q(s_{l,1}', a_{l,1}')), \quad (17)$$

where $s_{l,1}$ and $s_{l,1}'$ are current and next states of learner agent, respectively, and $a_{l,1}$ is the action. Compared with equation (16), the main difference is that $\sigma Q^T(\mathcal{F}(s_{l,1}), \mathcal{F}'(a_{l,1}))$ is involved as an extra reward of selecting action $a_{l,1}$ under state $s_{l,1}$. \mathcal{F} and \mathcal{F}' are state and action mapping functions, respectively. σ is the transfer factor, which describes the importance of prior knowledge ($0 < \sigma < 1$). A higher transfer

factor means the prior knowledge utilization is more important than its own learning, while a lower value indicates the reverse.

In equation (17), we apply $\sigma Q^T(\mathcal{F}(s_{l,1}), \mathcal{F}'(a_{l,1}))$ to guide the action selection of learner. The main idea is that learner agent is designed to prefer actions that have higher Q-values in experts, which is indicated by a higher extra reward in equation (17). Considering the similarities of tasks, these actions are expected to bring satisfying rewards in learner agent. However, due to different state-action space, the Q-values of experts cannot be directly utilized by the learner, thus a function is needed to map the experts' Q-values to the learner's Q-table. Q^T term in equation (17) is generated by:

$$Q^T(\mathcal{F}(s_{l,1}), \mathcal{F}'(a_{l,1})) = Q_{e,1}(s_{e,1}, a_{e,1}) + Q_{e,2}(s_{e,2}, a_{e,2}), \quad (18)$$

where $Q_{e,1}$, $s_{e,1}$ and $a_{e,1}$ are Q-value, state and action of expert 1, respectively. $Q_{e,2}$, $s_{e,2}$ and $a_{e,2}$ are defined similarly for expert 2. The idea behind the mapping functions \mathcal{F} and \mathcal{F}' is to find the states and actions of the experts that is similar with $s_{l,1}$ and $a_{l,1}$, then we can get the corresponding Q-values of experts. \mathcal{F} and \mathcal{F}' are defined by:

- **State mapping \mathcal{F} :** For a given state $s_{l,1}$, considering both experts and learners have the same state definition, we can always find $s_{l,1} = s_{e,1} = s_{e,2}$. Thus \mathcal{F} can be easily defined.
- **Action mapping \mathcal{F}' :** The goal of \mathcal{F}' is to find $(a_{e,1}, a_{e,2}) = \mathcal{F}'(a_{l,1})$. For any action $a_{l,1}$, which is defined as $a_{l,1} = (n^{emb}, n^{urllc}, c^{emb}, c^{urllc})$, it can be decomposed into the combination of $a_{e,1} = (n^{emb}, n^{urllc})$ and $a_{e,2} = (c^{emb}, c^{urllc})$. Then \mathcal{F}' can be defined accordingly.

Based on the state and action mapping relationships, if given $s_{l,1}$ and $a_{l,1}$, we can always find specific $Q_{e,1}(s_{e,1}, a_{e,1})$ and $Q_{e,2}(s_{e,2}, a_{e,2})$. Then $Q^T(\mathcal{F}(s_{l,1}), \mathcal{F}'(a_{l,1}))$ can be directly used by the learner 1.

2) Learner 2: Action Selection Transfer Reinforcement Learning

Learner 2 has the same MDP definition with learner 1, but the prior knowledge is different. Learners have bigger state-action space than experts because more actions are involved for joint allocation. Indeed, huge action space will lower the exploration efficiency and result in longer convergence time. We propose an ASTRL method to extract the action selection knowledge of experts, which will reduce the action space of learner 2 to speed up the learning process.

The original action space of learner 2 is described by:

$$A_{l,2} = \{(n^{emb}, n^{urllc}, c^{emb}, c^{urllc}) | 0 \leq n^{emb}, n^{urllc} \leq N^T; 0 \leq c^{emb}, c^{urllc} \leq C^T\}, \quad (19)$$

where N^T and C^T are the total number of RBs and caching capacity, respectively.

When reducing the action space in ASTRL, we need to assure that the potential optimal actions are preserved. To this end, we only select the actions that lead to a higher

accumulated reward in the experts [31]. For a given state (s^{emb}, s^{urllc}) , the reduced action space of learner 2 is:

$$A_{l,2} = \{(n^{emb}, n^{urllc}, c^{emb}, c^{urllc}) | (n^{emb}, n^{urllc}) \in A_{e,1}; (c^{emb}, c^{urllc}) \in A_{e,2}\}, \quad (20)$$

where $A_{e,1}$ and $A_{e,2}$ are defined as:

$$A_{e,1} = \{(n^{emb}, n^{urllc}) | Q_{e,1}(s^{emb}, s^{urllc}, n^{emb}, n^{urllc}) \geq \lambda \frac{\sum_{0 \leq n^{emb}, n^{urllc} \leq N^T} Q_{e,1}(s^{emb}, s^{urllc}, n^{emb}, n^{urllc})}{(N^T)^2}\} \quad (21)$$

$$A_{e,2} = \{(c^{emb}, c^{urllc}) | Q_{e,2}(s^{emb}, s^{urllc}, c^{emb}, c^{urllc}) \geq \lambda \frac{\sum_{0 \leq c^{emb}, c^{urllc} \leq C^T} Q_{e,2}(s^{emb}, s^{urllc}, c^{emb}, c^{urllc})}{(C^T)^2}\} \quad (22)$$

Equation (21) and (22) indicate that only actions with higher Q-values in experts will be included in the action space of learner 2. λ is the transfer factor for ASTRL ($0 < \lambda < 1$). A higher λ value means the learner mainly focuses on the prior knowledge utilization, which is indicated by a smaller action space. On the contrary, a lower transfer factor means the learner intends to explore the tasks by itself. Similarly, the Q-value is updated by equation (16). QTRL and ASTRL are summarized in Algorithm 1.

C. Baseline: Q-learning and PPF-TTL Algorithm

The basic Q-learning scheme is implemented as a baseline algorithm for the joint radio and cache resources allocation. The MDP definition is shown in Table II. Q-learning is similar with QTRL, but no prior knowledge is involved. Q-learning based resource allocation is given in Algorithm 2.

On the other hand, to evaluate the performance of the learning-based methods, we apply a model-based PPF-TTL algorithm. The well-known priority proportional fairness (PPF) algorithm is applied for radio resource allocation, in which URLLC packets has a higher priority than eMBB packets [32]. The RBs will first serve URLLC traffic, then eMBB traffic will be processed. We still apply the TTL method for caching, but no slicing and learning are included. The PPF-TTL method is shown in Algorithm 3.

VI. PERFORMANCE EVALUATION

A. Parameter Settings

In this section, we consider six different cases: expert 1, expert 2, QTRL, ASTRL, Q-learning and the PPF-TTL. The experience of experts is presumed to be the prior knowledge of learners. Each cases contains 5 gNBs shown as Fig.4, and each gNB contains an eMBB slice and a URLLC slice. The eMBB slice has 5 UEs, and URLLC slice has 10 UEs. A specific resource allocation strategy is implemented in each case. For example, in QTRL case, every gNB works as an independent agent to implement QTRL with the prior knowledge of experts' Q-tables. There are inter-cell interference in each scenario, but no intra-cell interference.

For each gNB, there are 100 RBs in total, which is divided into 13 resource block groups (RBGs) [33]. The first 12

Algorithm 1 TRL-based joint resources allocation

```

1: Initialize: Wireless network parameters
2: For QTRL, initializing Q-table; For ASTRL, initializing
   Q-table, and setting action space as equation (20).
3: for  $TTI = 1$  to  $T$  do
4:   for every BS do
5:     With probability  $\epsilon$ , choosing actions randomly, otherwise selecting actions by  $\arg \max_a Q(s, a)$ ;
6:     GRM implements inter-slice resource allocation.
7:     SRMs distribute radio resources to UEs by proportional fairness, and replace cached contents.
8:     GRM calculates reward based on received QoS.
9:     Updating system state.
10:    For QTRL, updating Q-values by:
         $Q^{new}(s_{l,1}, a_{l,1}) = \sigma Q^T(\mathcal{F}(s_{l,1}), \mathcal{F}'(a_{l,1})) +$ 
11:         $(1 - \alpha)Q^{old}(s_{l,1}, a_{l,1}) + \alpha(r + \gamma \max_{a_{l,1}} Q(s'_{l,1}, a_{l,1})),$ 
12:    For ASTRL, updating Q-values by:
         $Q^{new}(s_{l,2}, a_{l,2}) = (1 - \alpha)Q^{old}(s_{l,2}, a_{l,2}) + \alpha(r + \gamma \max_{a_{l,2}} Q(s'_{l,2}, a_{l,2})),$ 
13:
14:   end for
15: end for
```

Algorithm 2 Q-learning based joint resources allocation

```

1: Initialize: Wireless network parameters
2: Initialize Q-table to zero;
3: for  $TTI = 1$  to  $T$  do
4:   for every BS do
5:     With probability  $\epsilon$ , choosing actions randomly, otherwise selecting actions by  $\arg \max_a Q(s, a)$ ;
6:     GRM implements inter-slice resource allocation.
7:     SRMs distribute radio resources to UEs by proportional fairness, and replace cached contents.
8:     GRM calculates reward based on received QoS.
9:     Updating system state, and updating Q-values by:
10:     $Q^{new}(s, a) = (1 - \alpha)Q^{old}(s, a) + \alpha(r + \gamma \max_a Q(s', a)),$ 
11:   end for
12: end for
```

Algorithm 3 PPF-TTL based joint resources allocation

```

1: Initialize: Wireless network parameters
2: for  $TTI = 1$  to  $T$  do
3:   for every BS do
4:     for every RB do
5:       Calculating the estimated transmission rate of UEs in the queue.
6:       Calculating proportional fairness metric by equation (13).
7:       Transmitting URLLC packets with the highest proportional fairness. If no URLLC packet, then transmitting eMBB packets.
8:     end for
9:     BS replaces cached contents by time-to-live rule.
10:   end for
11: end for
```

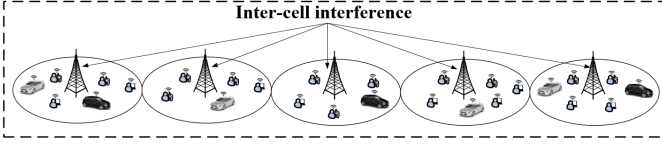


Fig. 4. Base station distribution.

TABLE III
PARAMETERS SETTINGS

5G settings	Cache settings
Bandwidth: 20MHz	Caching capacity: 20 contents
3GPP urban macro network	TTL value: 50 TTIs
Number of RBs: 100	Contents library size: 40/slice
Subcarriers in each RB: 12	Content popularity: Zipf
Subcarrier bandwidth: 15kHz	Traffic model
Transmission power: 40 dBm (uniform distributed) [34]	URLLC/ eMBB traffic: poisson distribution
TTI size: 2 OFDM symbols	Packet size: 36 Bytes
Tx/Rx antenna gain: 15 dB.	Propagation model
Retransmission settings	$128.1 + 37.6 \log(\text{distance}(\text{km}))$
Max number of retransmissions: 1	Log-Normal shadowing: 8 dB.
Round trip delay: 4 TTIs	Learning settings
Hybrid automatic repeat request.	Learning rate: 0.9
UE and gNBs	Discount factor: 0.5
25 eMBB UE, 50 URLLC UE	Epsilon value: 0.05
UE random distribution	Reward weight: 0.5
Number of gNBs: 5	Learner 1 transfer factor: 0.5
Inter-gNB distance: 500m	Learner 2 transfer factor: 0.5

RBGs contain 8 RBs each, and the last RBG has 4 RBs. We assume the caching capacity is reallocated every 50 TTIs because it takes time to replace the cached contents. The simulations include 3000 TTIs, where the first 1500 TTIs are exploration period, and the rest TTIs are exploitation period. The simulations are implemented in Matlab 5G library with 10 runs and 95% confidence interval. Other 5G and learning parameters are shown in Table III.

B. Comparison under Different Traffic Load

In this section, we compare the network performance of different algorithms under various traffic loads. The eMBB traffic is fixed to 1 Mbps per cell, and the URLLC traffic ranges from 1 to 6 Mbps. We first present the results, then explain the performance of each algorithm.

Fig.5 (a) shows the empirical cumulative distribution function (ECDF) of URLLC latency with 2 Mbps URLLC traffic, which presents the empirical distributions of packet delay. We zoom the area that the ECDF value is higher than 0.1 and the URLLC delay is lower than 1 ms to better show the results. The highest delay distribution is observed for expert 1, which is indicated by the high delay distribution in 0.1-1 interval of ECDF axis. On the contrary, expert 2 has the best delay distribution. The delay of Q-learning and PPF-TTL methods are relatively higher. Some packets in expert 1 and Q-learning method suffer more than 3 ms delay. The QTRL and

ASTRL take advantage of the TRL methods, and they achieve comparable latency performance. Fig.5 (b) and (c) present the average URLLC delay and eMBB throughput against traffic load. It shows that QTRL and ASTRL maintain lower delay and higher throughput than expert 1, Q-learning, and PPF-TTL under various traffic loads. Expert 2 shows a comparable URLLC delay, but its eMBB throughput performance is much lower than QTRL and ASTRL. In Fig.5 (d), all algorithms have a comparable PDR except the PPF-TTL. Following we will explain the results of each algorithm.

We first analyze the experts' performance. The expert 1 shows high delay and low throughput because it has no caching capability. All packets required by UEs in expert 1 has to be processed by the core network, and the constant backhaul delay leads to high URLLC delay and low eMBB throughput. On the contrary, expert 2 has lower delay because we apply a fixed RB allocation strategy. In particular, RBs are distributed according to UE numbers, thus the URLLC slice always has more RBs, which leads to low URLLC delay. But the eMBB slice is affected, and a low eMBB throughput is observed.

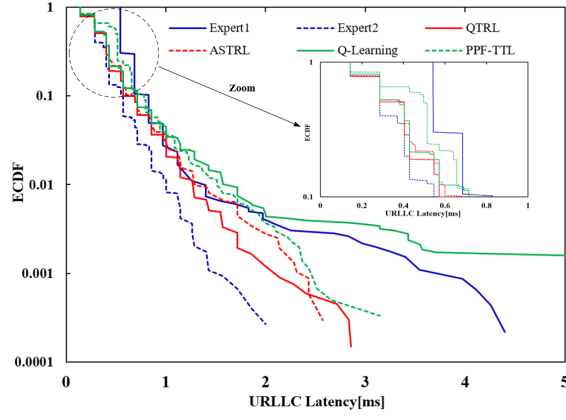
For the baseline algorithms, Q-learning outperforms PPF-TTL method and experts because it jointly learns the radio and cache resources allocation. The eMBB throughput of the PPF-TTL method decreases significantly, which results from the priority settings in this method. In the PPF method, whenever a new URLLC packet arrives, it is directly scheduled over the eMBB packets, which will unavoidably affect the eMBB performance.

Finally, the proposed QTRL and ASTRL have the best overall performance. Due to the novel TRL-based scheme, they can leverage the knowledge of experts and further improve their own performance on new tasks. When the URLLC traffic is 6 Mbps, QTRL and ASTRL present 23.9% lower URLLC delay and 41.6% higher eMBB throughput than Q-learning. 40.3% lower URLLC delay and almost doubled eMBB throughput are also observed compared with PPF-TTL method. The simulations show that the proposed TRL-based solutions achieve promising results than the baseline algorithms.

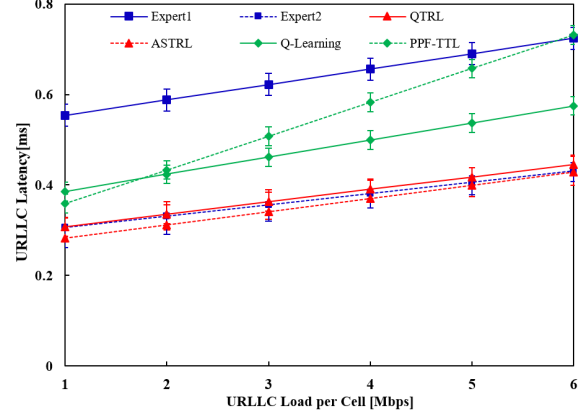
C. Comparison under Different Backhaul Capacity

Backhaul capacity is one of the main bottlenecks of 5G RAN. Here we investigate the network performance under various backhaul capacities, and the results are shown in Fig.6 (a) and (b). Considering the basic performance of experts have been shown in former results, we mainly compare the TRL-based solutions with baseline algorithms.

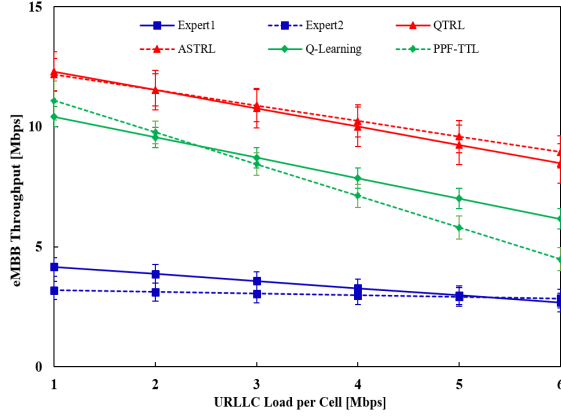
As expected, all algorithms have lower delay and higher throughput with increasing backhaul capacity, because higher capacity means lower backhaul delay. The improvements are less obvious when backhaul capacity is higher than 15 Mbps, which means the backhaul service rate is already huge enough for current traffic load. Meanwhile, QTRL and ASTRL achieve lower delay for URLLC slice and higher throughput for eMBB slice. When the backhaul capacity is 30 Mbps, QTRL and ASTRL present 37.8% lower URLLC delay and 13.6% higher



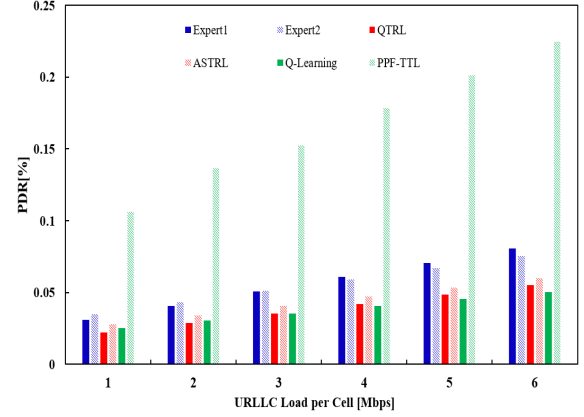
(a) ECDF of URLLC latency [ms] (1 Mbps eMBB traffic, 2 Mbps URLLC traffic)



(b) URLLC latency [ms] against traffic load [Mbps]

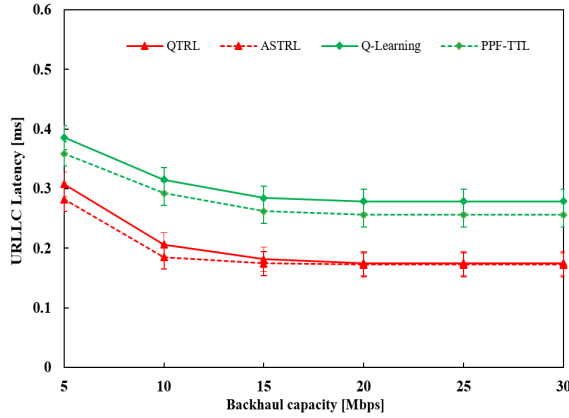


(c) eMBB throughput [Mbps] against traffic load [Mbps]

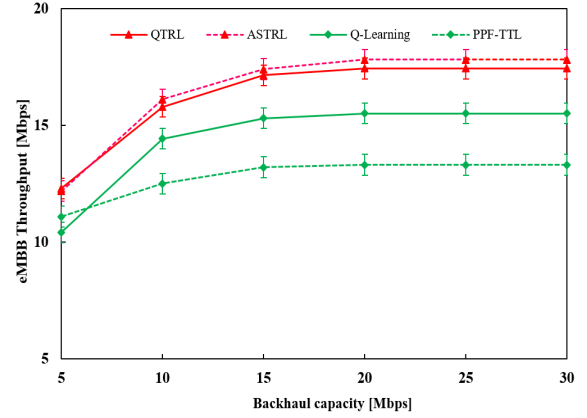


(d) PDR [%] comparison against traffic load [Mbps]

Fig. 5. Performance comparison under various traffic load.



(a) URLLC latency [ms] against backhaul capacity [Mbps]



(b) eMBB throughput [Mbps] against backhaul capacity [Mbps]

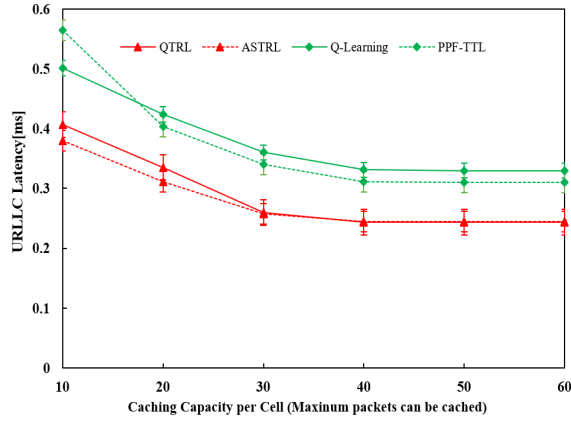
Fig. 6. Performance comparison under various backhaul capacities.

eMBB throughput than Q-learning. Compared with the PPF-TTL method, 32.3% lower URLLC delay and 32.4% higher eMBB throughput are observed.

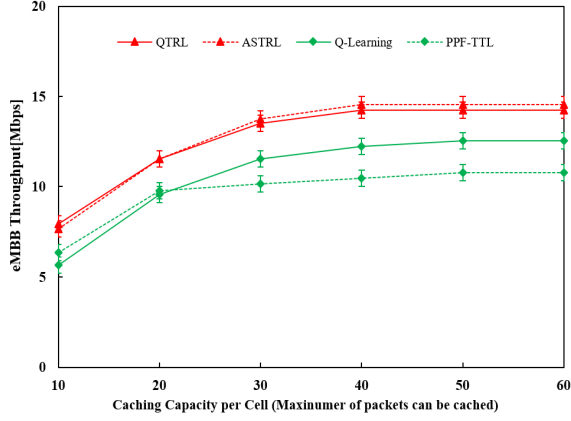
D. Comparison under Different Content Caching Capacity

In this section, we compare the network performance of different algorithms under various content caching capacities, which is indicated by the maximum number of contents can be stored. As shown in Fig.7 (a) and (b), a higher caching

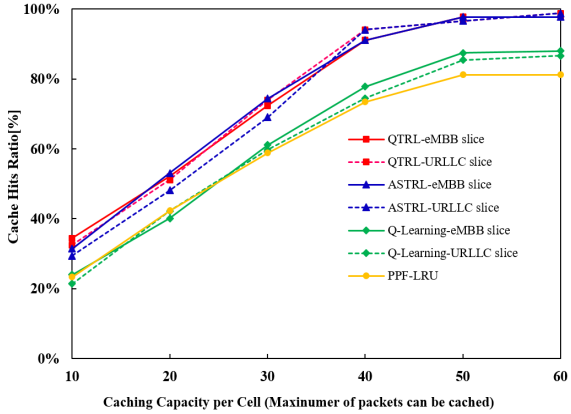
capacity will reduce the URLLC latency and increase the eMBB throughput. It is because a higher caching capacity means that more contents can be cached in the BS, and the average backhaul delay will be reduced. The Q-learning and PPF-TTL methods show comparable URLLC delay, but the eMBB throughput of the PPF-TTL method is 16.4% lower than Q-learning under the maximum caching capacity. QTRL and ASTRL still maintain lower URLLC delay and higher



(a) URLLC latency [ms] against caching capacity



(b) eMBB throughput [Mbps] against caching capacity

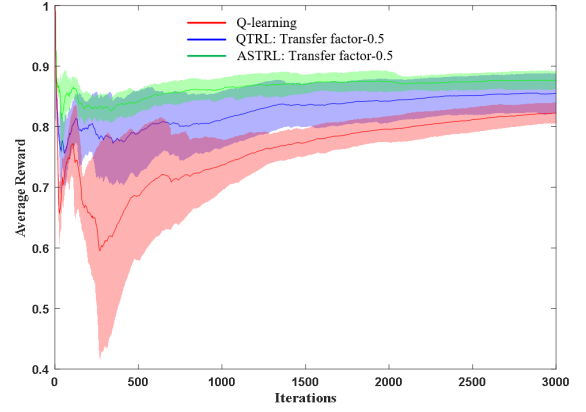


(c) Cached hit ratio [%] against caching capacity

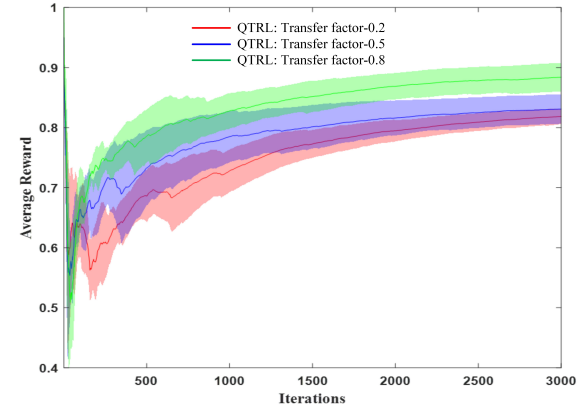
Fig. 7. Network performance comparison under various caching capacities.

eMBB throughput than other algorithms.

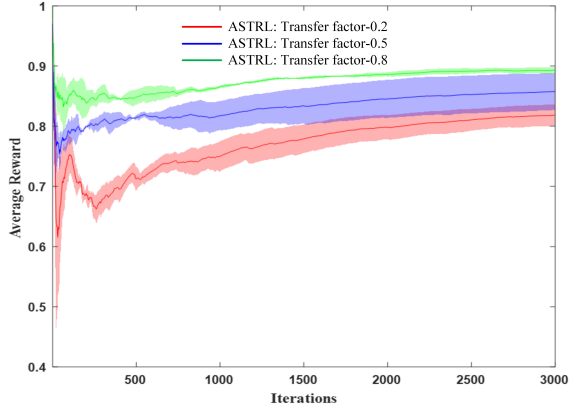
Furthermore, we present the cache hit ratio of eMBB and URLLC slices in Fig.7 (c). Cache hit ratio represents the proportion of packets that can be found in the cache server when they are demanded. A higher cache hit ratio indicates a better network performance, which is affected by both caching capacity and content replacement strategy. With the increasing caching capacity, cache hit ratios naturally increase for all algorithms. Here the cache hit ratio of the eMBB and URLLC slices are well maintained in QTRL and ASTRL, while the ratios in Q-learning and PPF-TTL method are 9.7% and 16.5%



(a) Convergence performance of Q-Learning, QTRL and ASTRL



(b) QTRL convergence performance under various transfer factor



(c) ASTRL convergence performance under various transfer factor

Fig. 8. Convergence performance of Q-learning, QTRL and ASTRL.

lower. Note that PPF-TTL method only has one curve because we assume no slicing in this method.

E. Convergence Performance Comparison

Here we will compare the convergence performance of QTRL, ASTRL and Q-learning, which is a critical metric for learning algorithms. In Fig.8 (a), ASTRL has the fastest convergence, which can be explained by the reduced action space. By extracting the action selection experiences of experts, ASTRL only selects actions that have higher Q-values in experts, which represents actions with higher potential

rewards. It proves that ASTRL applies a more efficient exploration strategy and achieves a better performance. QTRL also presents a better convergence performance than Q-learning. In QTRL, the Q-values of experts are extracted as extra rewards for action selections. It assumes that actions with higher Q-values in experts can also bring higher rewards to learners, and the exploration period is accelerated. However, other actions can still be selected with some probability, which will lower the exploration efficiency. On the contrary, in Q-learning, the agent has no prior knowledge about target tasks. The Q-learning agent starts from scratch to explore its tasks, which leads to longer convergence time and lower average reward.

Finally, note that we define transfer factors when introducing QTRL and ASTRL, which represents the importance of transferred knowledge. The convergence performance of QTRL and ASTRL under different transfer factors are investigated here. In QTRL, the transfer factor is indicated by the variable σ in equation (18). Fig.8 (b) shows that a higher σ value leads to faster convergence and higher average reward for QTRL. Because a higher σ indicates higher extra reward, and consequently a faster convergence is observed. A similar trend can be observed in Fig.8 (c) for ASTRL, in which the transfer factor is indicated by variable λ in equation (21) and (22). It is obvious that a higher ϕ value will lead to a smaller action space, and the exploration complexity is greatly reduced. In summary, both QTRL and ASTRL benefit from a higher transfer factor to shorten the exploration period.

VII. CONCLUSION

Evolving 5G networks require more efficient network management methods where the new ML techniques offer promising solutions. Although widely used reinforcement learning techniques have yield to improved performance, they suffer from long convergence time and lack of generalization. In that sense, knowledge transfer emerges as an important approach to improve the learning performance. Yet, transfer learning in wireless has been explored only very recently and in very few studies. In this work, we have presented two novel transfer reinforcement learning-based solutions for the joint radio and cache resources allocation. The proposed algorithms have been compared with Q-learning and the PPF-TTL algorithms via simulations. These results have shown that the proposed methods achieve better network metrics and faster convergence speeds than these other benchmarks. In the future, we plan to consider the knowledge transfer between tasks with different state definitions.

ACKNOWLEDGMENT

Mr. Zhou would like to thank Dr. Medhat Elsayed for initial discussions on transfer learning in this work.

REFERENCES

- [1] M. Shafi, A. Molisch, P. Smith, T. Haustein, P. Zhu, P. Silva, F. Tufvesson, A. Benjebbour, and G. Wunder, "5G: A Tutorial Overview of Standards, Trials, Challenges, Deployment, and Practice," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 6, pp. 1201-1221, Jun. 2017.
- [2] A. Ksentini, and N. Nikaein, "Toward Enforcing Network Slicing on RAN: Flexibility and Resources Abstraction," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 102-108, Jun. 2017.
- [3] T. Guo, and A. Suarez, "Enabling 5G RAN Slicing With EDF Slice Scheduling," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2865-2877, Mar. 2019.
- [4] D. Marabissi, and R. Fantacci, "Highly Flexible RAN Slicing Approach to Manage Isolation, Priority, Efficiency," *IEEE Access*, vol. 7, pp. 97130-97142, Jul. 2019.
- [5] T. Wang, and S. Wang, "Online Convex Optimization for Efficient and Robust Inter-Slice Radio Resource Management," *IEEE Transactions on Communications*, (Early Access), DOI: 10.1109/TCOMM.2021.3087127, Jun. 2021.
- [6] L. Li, G. Zhao, and R. S. Blum, "A Survey of Caching Techniques in Cellular Networks: Research Issues and Challenges in Content Placement and Delivery Strategies," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 1710-1732, Mar. 2018.
- [7] M. Erol-Kantarci, "Cache-At-Relay: Energy-Efficient Content Placement for Next-Generation Wireless Relays," *International Journal of Network Management*, vol. 25, no. 6, pp. 454-470, Nov./Dec. 2015.
- [8] Z. Zhao, M. Peng, Z. Ding, W. Wang, and H. V. Poor, "Cluster Content Caching: An Energy-Efficient Approach to Improve Quality of Service in Cloud Radio Access Networks," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 5, pp. 1207-1221, May 2016.
- [9] P. Yang, N. Zhang, S. Zhang, L. Yu, J. Zhang, and X. Shen, "Content Popularity Prediction Towards Location-Aware Mobile Edge Caching," *IEEE Transactions on Multimedia*, vol. 21, no. 4, pp. 915-929, Apr. 2019.
- [10] J. Kwak, Y. Kim, L. B. Le, and S. Chong, "Content Popularity Prediction Towards Location-Aware Mobile Edge Caching," *IEEE Transactions on Wireless Communications*, vol. 17, no. 5, pp. 3030-3045, May 2018.
- [11] M. Elsayed and M. Erol-Kantarci, "AI-Enabled Future Wireless Networks: Challenges, Opportunities, and Open Issues," *IEEE Vehicular Technology Magazine*, vol. 14, no. 3, pp. 70-77, Sep. 2019.
- [12] H. D. R. Albonda, and J. Perez-Romero, "An Efficient RAN Slicing Strategy for a Heterogeneous Network with eMBB and V2X Services," *IEEE Access*, vol. 7, pp. 44771-44782, Mar. 2019.
- [13] Y. Shi, Y. E. Sagduyu, and T. Erpek, "Reinforcement Learning for Dynamic Resource Optimization in 5G Radio Access Network Slicing," in *Proceedings of the 2020 IEEE 25th International Workshop on CAMAD*, pp. 1-6, Sep. 2020.
- [14] T. Li, X. Zhu, and X. Liu, "An End-to-End Network Slicing Algorithm Based on Deep Q-Learning for 5G Network," *IEEE Access*, vol. 8, pp. 122229-122240, Jul. 2020.
- [15] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized Computation Offloading Performance in Virtual Edge Computing Systems via Deep Reinforcement Learning," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4005-4018, Jun. 2019.
- [16] H. Xiang, M. Peng, Y. Sun, and S. Yan, "Mode Selection and Resource Allocation in Sliced Fog Radio Access Networks: A Reinforcement Learning Approach," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4271-4284, Apr. 2020.
- [17] H. Xiang, S. Yan, and M. Peng, "A Realization of Fog-RAN Slicing via Deep Reinforcement Learning," *IEEE Transactions on Wireless Communications*, vol. 19, no. 4, pp. 2515-2527, Apr. 2020.
- [18] X. Wu, J. Li, M. Xiao, P. C. Ching, H. V. Poor, "Multi-Agent Reinforcement Learning for Cooperative Coded Caching via Homotopy Optimization," *IEEE Transactions on Wireless Communications* (Early Access), DOI: 10.1109/TWC.2021.3066458, Mar. 2021.
- [19] X. Chen, Z. Zhao, C. Wu, M. Bennis, H. Liu, Y. Ji, and H. Zhang, "Multi-Tenant Cross-Slice Resource Orchestration: A Deep Reinforcement Learning Approach," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2377-2392, Oct. 2019.
- [20] Y. Wei, F. R. Yu, M. Song, and Z. Han, "Joint Optimization of Caching, Computing, and Radio Resources for Fog-Enabled IoT Using Natural Actor-Critic Deep Reinforcement Learning," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2061-2073, Apr. 2019.
- [21] H. Zhou, M. Elsayed, and M. Erol-Kantarci, "RAN Resource Slicing in 5G Using Multi-Agent Correlated Q-Learning," arXiv:2107.01018 [cs.NI], Jun. 2021.
- [22] M. Elsayed, M. Erol-Kantarci, and H. Yanikomeroglu, "Transfer Reinforcement Learning for 5G New Radio mmWave Networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 5, pp. 2838-2849, May. 2021.

- [23] M. E. Taylor and P. Stone, "Cross-Domain Transfer for Reinforcement Learning," in *Proceedings of the International Conference on Machine Learning*, pp. 879–886, Jun. 2007.
- [24] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.
- [25] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.
- [26] M. E. Taylor, P. Stone, and Y. Liu, "Transfer Learning for Reinforcement Learning Domains: A Survey," *Journal of Machine Learning Research*, vol. 10, pp. 1633–1685, Sep. 2009.
- [27] N. C. Fofack, P. Nain, G. Neglia, and D. Towsley, "Analysis of TTL-based cache networks," in *Proceedings of the 6th International ICST Conference on Performance Evaluation Methodologies and Tools*, pp. 1–10, Oct. 2012.
- [28] P. L. Vo, M. N. Nguyen, T. A. Le, and N. H. Tran, "Slicing the Edge: Resource Allocation for RAN Network Slicing," *IEEE Wireless Communications Letters*, vol. 7, no. 6, pp. 970–973, Dec. 2018.
- [29] T. Han, and N. Ansari, "Network Utility Aware Traffic Load Balancing in Backhaul-Constrained Cache-Enabled Small Cell Networks with Hybrid Power Supplies," *IEEE Transactions on Mobile Computing*, vol. 16, no. 10, pp. 2819–2832, Oct. 2017.
- [30] M. E. Taylor, P. Stone, and Y. Liu, "Transfer Learning via Inter-Task Mappings for Temporal Difference Learning," *Journal of Machine Learning Research*, vol. 8, no. 1, pp. 2125–2167, Sep. 2007.
- [31] L. Torrey, T. Walker, J. Shavlik, and R. Maclin, "Using Advice to Transfer Knowledge Acquired in One Reinforcement Learning Task to Another," in *Proceedings of the 2005 European Conference on Machine Learning*, pp. 412–424, Oct. 2005.
- [32] G. Pocovi, K. Pedersen, P. Mogensen, "Joint Link Adaptation and Scheduling for 5G Ultra-Reliable Low-Latency Communications," *IEEE Access*, vol. 6, pp. 28912–28922, May 2018.
- [33] 3GPP, "NR; Physical Layer Procedures for Data (version 15.2.0.)," Technical Specification 38.214, 3rd Generation Partnership Project (3GPP), Jun. 2018.
- [34] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA); Base Station (BS) Radio Transmission and Reception (version 12.5.0.)," Technical Specification 36.104, 3rd Generation Partnership Project (3GPP), Oct. 2014.

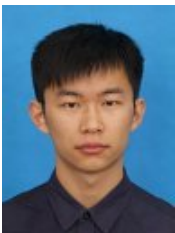


Melike Erol-Kantarci is Tier 2 Canada Research Chair in AI-enabled Next-Generation Wireless Networks and Associate Professor at the School of Electrical Engineering and Computer Science at the University of Ottawa. She is the founding director of the Networked Systems and Communications Research (NETCORE) laboratory. She has received several awards and recognitions. She has delivered 60+ keynotes, plenary talks and tutorials around the globe. She is on the editorial board of the *IEEE Transactions on Cognitive Communications and Networking*, *IEEE Internet of Things Journal*, *IEEE Networking Letters*, *IEEE Vehicular Technology Magazine* and *IEEE Access*. She has acted as the general chair and technical program chair for many international conferences and workshops. Her main research interests are AI-enabled wireless networks, 5G and 6G wireless communications, smart grid, Internet of things and wireless sensor networks. She is an IEEE ComSoc Distinguished Lecturer, IEEE Senior member and ACM Senior Member.



H. Vincent Poor (S'72, M'77, SM'82, F'87) received the Ph.D. degree in EECS from Princeton University in 1977. From 1977 until 1990, he was on the faculty of the University of Illinois at Urbana-Champaign. Since 1990 he has been on the faculty at Princeton, where he is currently the Michael Henry Strater University Professor. During 2006 to 2016, he served as the dean of Princeton's School of Engineering and Applied Science. He has also held visiting appointments at several other universities, including most recently at Berkeley and Cambridge.

His research interests are in the areas of information theory, machine learning and network science, and their applications in wireless networks, energy systems and related fields. Among his publications in these areas is the forthcoming book *Machine Learning and Wireless Communications*. (Cambridge University Press). Dr. Poor is a member of the National Academy of Engineering and the National Academy of Sciences and is a foreign member of the Chinese Academy of Sciences, the Royal Society, and other national and international academies. He received the IEEE Alexander Graham Bell Medal in 2017.



Hao Zhou is a Phd candidate at the University of Ottawa. He got his B.Eng. and M.Eng degrees from Huazhong University of Science and Technology in 2016, and Tianjin University in 2019, respectively, in China. He is working towards his Phd degree at the University of Ottawa since Sep. 2019. His research interests include electric vehicles, microgrid energy trading, resource management and network slicing in 5G. He is devoted to applying machine learning techniques for smart grid and 5G applications.