




Inter-Cell Network Slicing With Transfer Learning Empowered Multi-Agent Deep Reinforcement Learning

TIANLUN HU^{1,2}  (Graduate Student Member, IEEE), QI LIAO¹  (Member, IEEE),
QIANG LIU³  (Member, IEEE), AND GEORG CARLE²

¹Network Automation Department, NSSR Lab, Nokia Bell Labs, 70435 Stuttgart, Germany

²Department of Computer Engineering, Technical University of Munich, 80333 Munich, Germany

³School of Computing, University of Nebraska–Lincoln, Lincoln, NE 68588, USA

CORRESPONDING AUTHOR: T. HU (e-mail: tianlun.hu@nokia.com)

This work was supported by the German Federal Ministry of Education and Research (BMBF) Project KICK under Grant 16KIS1102K. Partial contents of this paper appear in International Conference on Communications (ICC) 2022 [1] [DOI: 10.1109/MCOM.2021.9530474].

ABSTRACT Network slicing enables operators to cost-efficiently support diverse applications on a common physical infrastructure. The ever-increasing densification of network deployment leads to complex and non-trivial inter-cell interference, which requires more than inaccurate analytic models to dynamically optimize resource management for network slices. In this paper, we develop a DRIP algorithm with multiple deep reinforcement learning (DRL) agents to cooperatively optimize resource partition in individual cells to fulfill the requirements of each slice, based on two alternative reward functions with max-min fairness and logarithmic utility. Nevertheless, existing DRL approaches usually tie the pre-trained model parameters to specific network environments with poor transferability, which raises practical deployment concerns in large-scale mobile networks. Hence, we design a novel transfer learning-aided DIRP (TL-DIRP) algorithm to ease the transfer of DIRP agents across different network environments in terms of sample efficiency, model reproducibility, and algorithm scalability. The TL-DIRP algorithm first centrally trains a generalized model and then transfers the “generalist” to each local agent (a.k.a., the “specialist”) with distributed finetuning and execution. TL-DIRP consists of two steps: 1) centralized training of a generalized distributed model, and 2) transferring the “generalist” to each local agent with distributed finetuning and execution. We comprehensively investigate different types of transferable knowledge: model transfer, instance transfer, and combined model and instance transfer. We evaluate the proposed algorithms in a system-level network simulator with 12 cells. The numerical results show that not only DIRP outperforms existing baseline approaches in terms of faster convergence and higher reward, but more importantly, TL-DIRP significantly improves the service performance, with reduced exploration cost, accelerated convergence rate, and enhanced model reproducibility. As compared to a traffic-aware baseline, TL-DIRP provides about 15% less violation ratio of the quality of service (QoS) for the worst slice service and 8.8% less violation on the average service QoS.

INDEX TERMS Transfer learning, deep reinforcement learning, multi-agent coordination, network slicing, resource allocation.

I. INTRODUCTION

EMERGING technologies, e.g., autonomous driving, augmented and mixed reality, lead to increasingly volatile network dynamics in terms of traffic, mobility,

and demand. To cost-efficiently accommodate heterogeneous services with diverse performance requirements, communications service providers (CSPs) offer virtual end-to-end networks (a.k.a., slices) on common shared network physical

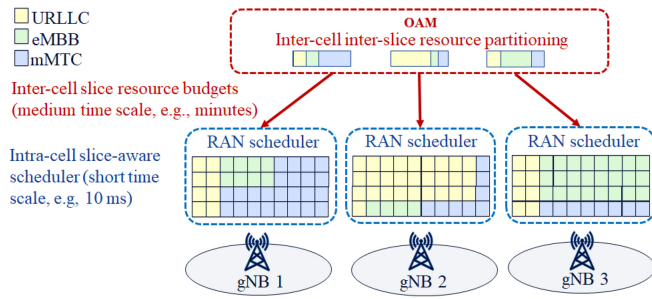


FIGURE 1. Dynamic multi-cell slicing resource allocation.

infrastructures, e.g., base stations and network switches. Network slicing enables performance and functional isolation, which guarantees that the slice performance is not affected by the operations in other slices, and assures the manageability for their slice tenants, respectively. To achieve dynamic network slicing under varying slice traffic, efficient resource management of virtual network resource is necessitated. For instance, a variety of slice-aware scheduling algorithms [2], [3] are proposed in radio access network (RAN) to dynamically allocate radio resource (e.g., physical resource blocks (PRBs)) of individual base stations, e.g., eNBs and gNBs, to different slices according to network conditions and service demands.

With the increasing spread of base station deployment in 5G and beyond, network slicing is becoming more complex. As a result, the lack of interference coordination in existing individualized approaches can degrade the slice performance in multi-cell scenarios [1]. Many works proposed model-based resource allocation and scheduling algorithms with inter-cell coordination, which rely on the approximated mathematical models towards the fast-changing interference and various optimization methods, e.g., linear programming [4], [5] and convex optimization [6], [7]. These algorithms are proposed to be implemented in RAN, and, to model the network capacity, they usually assume perfect channel state information (CSI) shared among all cells. In practical systems, however, such algorithms are extremely difficult to implement, because of two reasons at least: first, RAN scheduler makes decisions at a short time scale, e.g., every 10 ms, while such time constraint is very challenging for the model-based algorithms due to the high communication overhead (caused by CSI exchange) and the high computational cost (the “snapshot” approaches cannot well adapt to network dynamics and need to solve the problem for every time slot); second, these analytical solutions tend to fail in the real networks, because the approximated models cannot fully and accurately represent the complex network dynamics. Thus, in practical systems, as shown in Fig. 1, inter-cell inter-slice resource partitioning is introduced into network operations, administration, and maintenance (OAM) [8], which collects a limited set of key performance indicators (KPIs) from all cells at medium time scale (e.g., minutes or even a quarter

hour), performs inter-slice resource partitioning, and provides per-slice resource budgets to all cells. Then, each RAN receives the resource budgets computed by OAM periodically and uses them as resource constraints to guide the slice-aware scheduling and PRB allocation algorithms within RAN. In this paper, we focus on the inter-cell inter-slice resource partitioning problem in network OAM. Note that in OAM we tackle a problem different from the conventional coordinated interference mitigation problem at the media access control (MAC) layer in RAN, because we obtain only a limited set of cell-based KPIs, while the short-term physical layer measurements such as CSI are not available. Moreover, the mapping from the multi-cell network optimization parameters to these higher-layer KPIs is usually non-linear and non-convex, and the optimization goal is often multi-objective. Thus, OAM usually benefits from the model-free machine learning and deep learning approaches that can be implemented in a distributed, cloud-native manner.

Recent advances in model-free approaches, especially deep reinforcement learning (DRL) [9], [10], have shown promising potential in automatically learning to manage radio access networks without the need for prior models. In general, the resource management problem is formulated as a Markov Decision Process (MDP), which is then addressed by training and deriving a deep neural network parameterized policy. A variety of DRL algorithms, e.g., deep Q-network (DQN), Deep Deterministic Policy Gradient (DDPG), and proximal policy optimization (PPO) are exploited to achieve better policies in terms of performance, robustness, and convergence. In particular, the problems with constraints, e.g., performance requirements, are resolved by leveraging different methods, e.g., interior-point policy optimization [10] and Lagrangian primal-dual methods [11]. The inter-cell coordination problem is studied with distributed multi-agent deep reinforcement learning (MADRL) approaches, which create multiple DRL agents and train their policies in different schemes. The centralized scheme aims to train a common policy for all agents, where agents are distributedly executed with the shared model as the training completes. For example, Li et al. [12] proposed a centralized scheme for slicing resource management with a DRL-based algorithm, but it fails to address the model complexity of agents when the network scale grows. In contrast, the distributed scheme [13], [14], [15] independently trains agents with individualized policy, which shows promising performance improvement in terms of convergence speed and communication overhead. Zhao et al. [14] investigated the dynamic resource allocation problem in network slicing with distributed DRL, which lacks inter-agent coordination and thus suffers uncoordinated interference in multi-cell slicing management. Several efforts [16] have been made to address the issue of non-stationary environments from the perspective of individual agents, e.g., augmenting the state space of individual agents. However, these aforementioned approaches raise concerns about sample efficiency, lengthy exploration, and convergence speed,

which hinder their practical implementations in large-scale networks.

The emerging transfer learning (TL) techniques [17] have been increasingly studied to address this challenge regarding the algorithm scalability, model reproducibility, and sample efficiency in machine learning-based approaches [18], [19], [20]. The basic idea of TL is to utilize prior knowledge from pretrained models to benefit the learning process in target models. Although there are extensive TL works [21], [22], they are in the supervised learning domain, e.g., computer vision, and cannot be directly applied in reinforcement learning (RL) domain [23], [24]. A few works [25], [26] studied TL in resource allocation in mobile networks, e.g., spectrum sharing in vehicle-to-everything (V2X) [27] and parameter optimization in network slicing. However, TL-assisted MADRL in inter-cell network slicing scenarios is still an open problem.

In this paper, we focus on the inter-cell resource partition problem in network slicing with distributed MADRL by extending our previous work [1]. Our objective is to optimize the service qualities over all slices and cells while satisfying the constraints of the resource capacity. We first develop a distributed inter-cell inter-slice resource partition (DIRP) algorithm, which effectively solves the problem with an inter-agent coordination mechanism, allowing information sharing between cells. The optimization is based on two alternative designs of objectives: 1) *max-min fairness* over all slices, and 2) *maximizing the average logarithmic utility* over all slices. The former guarantees that all slice-specific requirements for throughput and delay are fulfilled. The motivation is to align with 3GPP specifications that the service provided by any network slice must comply with the service level agreement (SLA) [28]. Note that max-min fairness, known to provide the best fairness guarantees, is a special case of the general class of the well-known α -fair utility functions [29], [30]. The latter, as a classical concave utility function, also belonging to the α -fair utility functions, compromises the SLA fairness to improve resource efficiency [31]. Then, we design a transfer learning-aided DIRP (TL-DIRP) algorithm to further improve the sample efficiency, model reproducibility, and algorithm scalability. We investigate the effectiveness of the transferable knowledge in three schemes, i.e., *pretrained model transfer*, *instance transfer*, and *combined model and instance transfer*. We further observe several key insights from the simulation results when integrating TL in MADRL under these schemes. The contributions of this paper are summarized as follows:

- We formulate the dynamic inter-cell resource partitioning problem to meet the requirements of throughput and latency for all slices, under the inter-slice resource constraints. We study two alternative objectives: 1) maximizing the minimum service quality over all slices and cells, and 2) maximizing the average of logarithmic utilities over all slices.
- We design a multi-agent DRL algorithm to solve the problem with inter-agent coordination. We show that

inter-agent load sharing improves the performance of conventional distributed schemes while achieving a lower model complexity and a faster convergence in comparison with centralized single-agent schemes.

- We further design a novel TL-DIRP algorithm to ease the transfer of DRIP agents across different network environments and analyze its effectiveness in three schemes, i.e., *pretrained model transfer*, *instance transfer* and *combined model and instance transfer*.
- We implement the proposed solutions in a system-level simulator and evaluate by comparing them with three baselines, i.e., centralized DRL, distributed DRL, and a traffic-aware heuristic approach. The results show that DIRP outperforms all three baselines in terms of per-slice service quality, and the proposed TL-DIRP further improves the performance with much faster algorithm convergence and lower exploration cost.

The rest of the paper is organized as follows. In Section III, we define the system model and formulate the inter-cell inter-slice resource partitioning problem. In Section IV, we propose the DIRP algorithm to solve the problem with inter-agent coordination. In Section V, we enhance the DIRP algorithm with transfer learning and investigate different types of transferable knowledge. The numerical results are demonstrated in Section VI. Finally, we conclude this paper in Section VII.

II. RELATED WORK

This work relates to network resource management, deep reinforcement learning in mobile networks, and transfer learning in networking.

Model-Based Resource Management: There are extensive works that use model-based approaches to manage the resource allocation of RAN slices in 5G and beyond networks. Several works [6], [7] investigated the problem of network slice resource allocation by assuming the resource demands are known and static and leveraged the methods of convex optimization to solve the problem with different utility functions. The network slicing for machine-type communications is studied in [5], where a radio resource allocation method is proposed to dynamically select channel bandwidth according to the QoS requirements and traffic aggregation in machine-to-machine (M2M) gateways. Addad et al. [4] analyzed the virtual network function deployment in network slicing, formulated a mixed-integer linear programming model, and proposed a heuristic algorithm under different resource constraints. Cavalcante et al. [32] formulated a max-min fairness problem to handle load-coupled interference, then transformed it into a fixed point problem and solved it with low complexity iteration algorithm. Recently, an inter-cell coordinated scheme for dense cellular network resource scheduling was proposed [33], which tackled inter-cell interference and provided inspiring results. However, the approximated mathematical models cannot fully represent the characteristics of

complex networks. More importantly, applying these model-based solutions in OAM is challenging due to the lack of CSI measurements at fine time granularity.

Deep Reinforcement Learning in Mobile Networks: Liu et al. [10] proposed a constrained DRL-based on interior-point policy optimization (IPO) to solve the slicing resource allocation problem in the single base station scenario. Xu et al. [34], studied a DRL-based solution to extract per-slice users' behavior with traffic-aware exploration and allocate sufficient RAN resource accordingly. Liu et al. [11] proposed a DRL-based algorithm named DeepSlicing by decomposing RAN slicing optimization into a master problem and several slave problems, which are addressed with a joint coordinator and associated DRL agent for each slice respectively. However, these works are designed to address the resource allocation problem in single-cell scenarios. Several works [13], [14] studied the multi-cell scenarios and proposed several DRL solutions with discrete action space. Recent efforts [35], [36] extended the discrete action space into continuous action space, which showed improved performances in handling complex scenarios. However, none of them addressed the inter-cell dependencies and inter-slice resource constraints.

Transfer Learning in Networking: Xu et al. proposed an aggregation TL method applied to MADRL for real-time strategy games by transferring knowledge from small-scale to large-scale multi-agent systems, which improves the convergence speed of the algorithm [37]. Zafar et al. proposed to enhance the double Q-learning with TL for solving the decentralized spectrum sharing problem in the V2X communication networks [27]. By transferring the Q-values of the expert model to the learner model, the TL-assisted method accelerates the convergence rate of the learner model. Mai et al. [26] proposed to optimize the slice parameters, e.g., transmission power and spreading factor, with DDPG and TL. The TL was conducted by pretraining a model on a centralized controller and then using it as the initial model on local slice optimization tasks. Nagib et al. [25] studied TL to accelerate the DRL algorithms for dynamic RAN slicing resource allocation in single-cell scenarios, by transferring the model pretrained from an expert base station to a learner base station. Nevertheless, none of the above-mentioned works studied TL in coordinated MADRL for inter-cell slicing resource partition.

III. SYSTEM MODEL AND PROBLEM FORMULATION

In this section, we first describe the MDP-based system model in Section III-A. Then, we formulate the optimization problem based on the MDP model in Section III-B. Table 1 summarizes the notations used in this work.

A. SYSTEM MODEL

We consider a network system consisting of a set of cells $\mathcal{K} := \{1, 2, \dots, K\}$ and a set of slices $\mathcal{N} := \{1, 2, \dots, N\}$. Each slice $n \in \mathcal{N}$ has predefined throughput and delay requirements, denoted by ϕ_n^* and d_n^* , respectively. The

TABLE 1. Table of notations.

Symbol	Meaning
\mathbf{s}	Global state in \mathcal{S}
\mathbf{a}	Global action in \mathcal{A}
r	Global reward
\mathbf{s}_k	Local state in \mathcal{S}_k of Agent k
\mathbf{a}_k	Local action in \mathcal{A}_k of Agent k
r_k	Local reward of agent k
\tilde{r}_k	Approximated local reward of Agent k
\mathbf{m}_k	Message sent from Agent k to neighbors
$\bar{\mathbf{m}}_k$	Received messages from all neighbors of Agent k
\mathbf{c}_k	Extracted information from $\bar{\mathbf{m}}_k$ of Agent k
Q_θ	Current critic network with parameter θ
π_ϕ	Current actor network with parameter ϕ
$Q_{\theta'}$	Target critic network with parameter θ'
$\pi_{\phi'}$	Target actor network with parameter ϕ'
$\pi^{(G)}$	Generalist's policy learned by central controller
π_k	Specialist's policy learned by Agent k
\mathcal{D}_S	Source domain $\mathcal{D}_S := \mathcal{D}^{(G)}$, i.e., generalist's domain
\mathcal{T}_S	Source task $\mathcal{T}_S := \mathcal{T}^{(G)}$, i.e., generalist's task
\mathcal{D}_T	Target domain $\mathcal{D}_T := \mathcal{D}_k^{(S)}, k \in \mathcal{K}$, i.e., specialist's domain
\mathcal{T}_T	Target task $\mathcal{T}_T := \mathcal{T}_k^{(S)}, k \in \mathcal{K}$, i.e., specialist's task

network system runs on discrete time slots $t \in \mathbb{N}_0$. OAM adapts the inter-slice resource partitioning for all cells periodically to meet their performance requirements, as illustrated in Fig. 1.

To capture the temporal and inter-cell dependencies, we model the multi-cell resource partition as an MDP defined by $\mathcal{M} := \{\mathcal{S}, \mathcal{A}, P(\cdot), r(\cdot), \gamma\}$, where $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ denotes the transition probability distribution over state space \mathcal{S} and action space \mathcal{A} . $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, which evaluates the per-slice QoS for all cells and $\gamma \in [0, 1]$ denotes the discount factor for cumulative reward calculation.

Assuming that at each time step t , the network observes the global **state** $\mathbf{s}(t) := [\mathbf{s}_1(t), \dots, \mathbf{s}_K(t)] \in \mathcal{S}$, where $\mathbf{s}_k(t)$ is the local state observed from cell k . The **action** at slot t denoted by $\mathbf{a}(t) := [\mathbf{a}_1(t), \dots, \mathbf{a}_K(t)] \in \mathcal{A}$, includes the RAN slice resource budget, where the local action $\mathbf{a}_k(t) \in \mathcal{A}_k$ indicates the partitioning ratio $a_{k,n}(t) \in [0, 1]$ to each slice for $n \in \mathcal{N}$ aligning with intra-cell resource constraints. Thus, the local action space \mathcal{A}_k and the global action space \mathcal{A} yield

$$\mathcal{A}_k := \left\{ \mathbf{a}_k \mid a_{k,n} \in [0, 1], \forall n \in \mathcal{N}; \sum_{n=1}^N a_{k,n} = 1 \right\}. \quad (1)$$

$$\mathcal{A} := \{ \mathbf{a} \mid \mathbf{a}_k \in \mathcal{A}_k, \forall k \in \mathcal{K} \}. \quad (2)$$

The goal is to maximize the satisfaction level of QoS in terms of throughput and delay requirements (ϕ_n^* , d_n^*) for every slice $n \in \mathcal{N}$ in each cell $k \in \mathcal{K}$. Thus, we design two alternative **reward** functions for the two alternative objective designs: *max-min fairness* and *maximizing the average logarithmic utilities*. The former provides the best fairness that guarantees overall slice requirements by giving the maximum protection to the most critical and resource-demanding slice. While the latter, although taking fairness into account, still tries to achieve a good fairness-efficiency tradeoff.

The global **reward** function $r(t)$, based on the two alternative objectives, respectively, is defined as follows:

- 1) *Max-Min Fairness*: we define $r(t)$ as the minimum per-slice QoS satisfaction level based on the observed average throughput $\phi_{k,n}(t)$ and average delay $d_{k,n}(t)$ at time step t for each slice n in cell k , as

$$r(t) := \min_{k \in \mathcal{K}, n \in \mathcal{N}} \min \left\{ \frac{\phi_{k,n}(t)}{\phi_n^*}, \frac{d_n^*}{d_{k,n}(t)}, 1 \right\}. \quad (3)$$

The reward formulation drops below 1 when the actual average throughput or delay of any slices fails to fulfill the requirements. Note that the reward is upper bounded by 1 even if all slices achieve better performances than the requirements, to achieve more efficient resource utilization. The second item in (3) is **inversely proportional** to the actual delay, namely, if the delay is longer than required, this term is lower than 1.

- 2) *Maximizing the Average Logarithmic Utilities*: we define $r(t)$ as the average logarithmic utilities over the service satisfaction levels of all slices, given by

$$r(t) := \frac{1}{K \cdot N} \cdot \sum_{k \in \mathcal{K}, n \in \mathcal{N}} \log \left(\min \left\{ \frac{\phi_{k,n}(t)}{\phi_n^*}, \frac{d_n^*}{d_{k,n}(t)} \right\} + 1 \right) \quad (4)$$

where the *service satisfaction level* per slice per cell $\min \left\{ \frac{\phi_{k,n}(t)}{\phi_n^*}, \frac{d_n^*}{d_{k,n}(t)} \right\} \geq 0$ is defined as the minimum between the throughput and delay satisfaction levels. Thus, if either throughput or delay does not meet the requirement, this term is below 1. By adding an offset 1 within the log function with base 2, the per-slice logarithmic utility function is always non-negative. Note that unlike (3), the reward in (4) is not upper bounded by 1, because the service satisfaction level is not upper bounded. However, if all slices' requirements are exactly met, then we have $r(t) = 1$.

B. PROBLEM FORMULATION

The problem is to find the optimal policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$, which decides the inter-cell inter-slice resource partitioning $\mathbf{a} \in \mathcal{A}$ based on the observation of network state $\mathbf{s} \in \mathcal{S}$, to maximize the expectation of the cumulative discounted reward defined in Eq. (3) or Eq. (4) of a trajectory for a finite time horizon T . The problem is given by:

Problem 1:

$$\max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=0}^T \gamma^t r(\mathbf{s}(t), \mathbf{a}(t)) \right], \text{ s.t. } \mathbf{a} \in \mathcal{A}, \quad (5)$$

where \mathcal{A} is defined by Eq. (1) and Eq. (2), r is given by Eq. (3) or Eq. (4).

The challenges of solving the aforementioned problem are two-fold. First, the global reward functions depend on high-dimensional state and action spaces, and involve complex inter-cell dependencies, which are difficult to be accurately

obtained in practical network systems. For example, increasing resource partition in one slice n and cell k improves its own service performance, however, it decreases the available resource allocated to other slices in the same cell and may aggravate the interference received in neighboring cells. Besides, because we aim at solving the inter-cell inter-slice resource partitioning problem in OAM, only a limited set of KPIs (e.g., averaged cell throughput and delay) at a medium time scale (e.g., every 15 minutes) is available. It is extremely difficult to derive closed-form expressions for the multi-cell network with the extracted data at the higher layers (above MAC layer) of the network system. Second, the dynamic of network systems, e.g., additional cell deployments, changes the properties of the problem, e.g., leading to expanded state and action space. This requires the solution of this problem to be efficient and scalable in terms of fast convergence speed, high sample efficiency, and low computational efforts.

IV. DISTRIBUTED INTER-CELL RESOURCE PARTITION

In this section, we propose the distributed inter-cell inter-slice resource partition (DIRP) algorithm based on the MADRL approach with an inter-agent coordination scheme. Then, we briefly introduce the actor-critic method to solve the DRL problem. Next, we propose the method to tackle the intra-cell resource constraint with modified DRL network architecture.

A. PROPOSED DIRP ALGORITHM

In this part, we propose the DIRP algorithm with inter-agent coordination, which allows each agent to learn an individualized policy and make its own decision on the local action, based on local observations and neighboring information. In contrast to conventional centralized DRL [12], which collects global observation from all slices and cells of the network system, the DIRP algorithm may not achieve the global performance as good as the centralized one due to the limited observation on the entire network. However, it may converge much faster and be more sample efficient by using a less complex model based on lower dimensional state and action spaces, and the coordination mechanism could improve the performance of distributed agents with additional side information about the environment.

To capture local network observations, each agent k observes its **local state** \mathbf{s}_k . In particular, we include the following measurements and performance metrics:

- Average per-slice user throughput $\{\phi_{k,n} : n \in \mathcal{N}\}$;
- Per-slice load $\{l_{k,n} : n \in \mathcal{N}\}$;
- Per-slice number of active users $\{u_{k,n} : n \in \mathcal{N}\}$;
- Per-slice throughput requirement $\{\phi_{k,n}^* : n \in \mathcal{N}\}$;
- Per-slice delay requirement $\{d_{k,n}^* : n \in \mathcal{N}\}$.

In conventional distributed DRL approach, each agent k in the k -th cell computes a **local reward** r_k , and makes decision on the **local action** $\mathbf{a}_k \in \mathcal{A}_k \subset [0, 1]^N$. The local reward for max-min fairness or maximizing average logarithmic utilities, based on the local observations, is given by

1) *Max-Min Fairness:*

$$r_k(t) := \min_{n \in \mathcal{N}} \min \left\{ \frac{\phi_{k,n}(t)}{\phi_n^*}, \frac{d_n^*}{d_{k,n}(t)}, 1 \right\}, \quad (6)$$

2) *Maximizing Average Logarithmic Utilities:*

$$r_k(t) := \frac{1}{N} \cdot \sum_{n \in \mathcal{N}} \log \left(\min \left\{ \frac{\phi_{k,n}(t)}{\phi_n^*}, \frac{d_n^*}{d_{k,n}(t)} \right\} + 1 \right). \quad (7)$$

Note that r_k not only depends on the local state-action pair but also on the states and actions of other agents. The global reward yields $r(t) = \min_{k \in \mathcal{K}} r_k(\mathbf{a}(t), \mathbf{s}(t))$ with local reward (6) or $r(t) = \frac{1}{K} \sum_{k \in \mathcal{K}} r_k(\mathbf{a}(t), \mathbf{s}(t))$ with local reward (7). We can approximate $r_k(\mathbf{s}, \mathbf{a})$ based on the local observations $(\mathbf{s}_k, \mathbf{a}_k)$, denoted by $\tilde{r}_k(\mathbf{s}_k(t), \mathbf{a}_k(t))$. However, the estimation can be inaccurate because it neglects the inter-cell dependencies and estimates local reward independently.

Thus, to capture the inter-agent dependencies, in DIRP algorithm we let the agents communicate and exchange additional information with neighboring cells. Let each agent k send a message \mathbf{m}_k to a set of its neighboring agents, denoted by \mathcal{K}_k . Then, each agent k holds the following information: local state and action pair $(\mathbf{s}_k, \mathbf{a}_k)$ and received messages $\bar{\mathbf{m}}_k := \{\mathbf{m}_i : i \in \mathcal{K}_k\}$. One option is to directly use all received messages $\bar{\mathbf{m}}_k$ along with $(\mathbf{s}_k, \mathbf{a}_k)$ to estimate $r_k(\mathbf{s}, \mathbf{a})$ with $\tilde{r}_k(\mathbf{s}_k, \bar{\mathbf{m}}_k, \mathbf{a}_k)$. However, if the dimension of the exchanged message is high, this increases the complexity of the local model.

An alternative is to extract the useful information $\mathbf{c}_k \in \mathbb{R}^{Z^{(c)}}$ from the received messages $\bar{\mathbf{m}}_k \in \mathbb{R}^{Z^{(m)}}$ with $g : \mathbb{R}^{Z^{(m)}} \rightarrow \mathbb{R}^{Z^{(c)}} : \bar{\mathbf{m}}_k \mapsto \mathbf{c}_k$, such that $Z^{(c)} \ll Z^{(m)}$, where $Z^{(m)}$ and $Z^{(c)}$ stand for the corresponding dimensions. We can then use $\tilde{r}_k(\mathbf{s}_k, \mathbf{c}_k, \mathbf{a}_k)$ to approximate r_k , by capturing the hidden information in the global state, while remaining the low model complexity. Pioneer works such as [38] proposed to learn the extraction of the communication messages by jointly optimizing the communication action with the reinforcement learning model. However, the joint training of multiple interacting models usually leads to extended convergence time and even diverged training. To provide a robust and efficient practical solution, we leverage domain knowledge to extract the information. Knowing that the inter-agent dependencies are mainly caused by the load-coupling inter-cell interference, we propose to let each agent k communicate with its neighboring agents the slice-specific load information $l_{i,n}, \forall n \in \mathcal{N}$. Then, based on the exchanged load information, we compute the average per-slice neighboring load as the extracted information $\mathbf{c}_k(t)$. Namely, we define a deterministic function

$$g_k : \mathbb{R}^{N|\mathcal{K}_k|} \rightarrow \mathbb{R}^N : [l_{i,n} : n \in \mathcal{N}, i \in \mathcal{K}_k] \mapsto \mathbf{c}_k(t)$$

with
$$\mathbf{c}_k(t) := \left[\frac{1}{|\mathcal{K}_k|} \sum_{i \in \mathcal{K}_k} l_{i,n}(t) : n \in \mathcal{N} \right]. \quad (8)$$

In this way, the DIRP algorithm solves Problem 1 with approximated local reward while considering the inter-cell

dependencies by including neighboring information. Thus, the DIRP algorithm approximates $r_k(\mathbf{s}, \mathbf{a})$ with $\tilde{r}_k(\mathbf{s}_k, \mathbf{c}_k, \mathbf{a}_k)$, decomposes Problem 1 with K independent subproblems, and finds the following local policies $\pi_k : \mathcal{S}_k \times \mathbb{R}^N \rightarrow \mathcal{A}_k$ for each DIRP agent $k \in \mathcal{K}$:

$$\pi_k^* = \arg \max_{\pi_k : \mathbf{a}_k \in \mathcal{A}_k} \mathbb{E}_{\pi_k} \left[\sum_{t=0}^T \gamma^t \tilde{r}_k(\mathbf{s}_k(t), \mathbf{c}_k(t), \mathbf{a}_k(t)) \right]. \quad (9)$$

B. THE TRAINING OF AGENTS

In this part, we follow the actor-critic method [39] to train the agents, which has proven effective when dealing with high dimensional and continuous state space. Such method solves the optimization problem by using critic function $Q(\mathbf{s}_t, \mathbf{a}_t | \theta)$ (in this subsection, we denote $\mathbf{s}(t)$ and $\mathbf{a}(t)$ by \mathbf{s}_t and \mathbf{a}_t respectively for brevity) to approximate the value function, i.e., $Q(\mathbf{s}_t, \mathbf{a}_t | \theta) \approx Q^\pi(\mathbf{s}_t, \mathbf{a}_t)$, and actor $\pi(\mathbf{s}_t | \phi)$ to update the policy π at every DRL step in the direction suggested by critic. For brevity, we denote the network with parameters in the form Q_θ and π_ϕ for critic and actor respectively.

In this work, we use Twin Delayed Deep Deterministic policy gradient (TD3) algorithm [40] as an off-policy DRL algorithm built on top of the actor-critic method. As an extension of DDPG [41], TD3 overcomes the DDPGs problem of overestimating Q-values by introducing a double critic structure for both current networks $Q_{\theta_1}, Q_{\theta_2}$ and target networks $Q_{\theta'_1}, Q_{\theta'_2}$. The minimum of the two Q-values is used to represent the approximated Q-value of the next state. Besides, the updates of the policy network are less frequent than the value network, which allows the value network to reduce errors before it is used to update the policy network. Moreover, TD3 uses target policy smoothing, i.e., adding noise to the target action, to make it harder for the policy to exploit Q-function errors by smoothing out Q along with changes in action. The target actions are computed based on the next state collected in the sample, given by

$$\mathbf{a}'(\mathbf{s}_{t+1}) = \text{clip}(\pi'_{\phi'}(\mathbf{s}_{t+1}) + \text{clip}(\epsilon, -c, c), \mathbf{a}_L, \mathbf{a}_H) \quad (10)$$

where the added noise $\epsilon \sim \mathcal{N}(0, \sigma)$ is clipped to keep the target close to the original action, and $\mathbf{a}_L, \mathbf{a}_H$ are the lower and upper bounds of the action, respectively.

The target update in TD3 is given by:

$$y_t = r_t + \gamma \min_{i=1,2} Q_{\theta'_i}(\mathbf{s}_{t+1}, \mathbf{a}'(\mathbf{s}_{t+1})). \quad (11)$$

The critic parameters $\theta_i, i \in \{1, 2\}$ are updated with temporal difference (TD) learning, given by:

$$L(\theta_i) = \mathbb{E} \left[(y_t - Q_{\theta_i}(\mathbf{s}_t, \mathbf{a}_t))^2 \right]. \quad (12)$$

The actor is updated by policy gradient based on the expected cumulative reward J with respect to the actor parameter θ^π with:

$$\begin{aligned} \nabla_{\phi} J &\approx \mathbb{E} \left[\nabla_{\phi} Q_{\theta_1}(\mathbf{s}, \mathbf{a}) |_{\mathbf{s}=\mathbf{s}_t, \mathbf{a}=\pi_{\phi}(\mathbf{s}_t)} \right] \\ &= \mathbb{E} \left[\nabla_{\mathbf{a}} Q_{\theta_1}(\mathbf{s}, \mathbf{a}) |_{\mathbf{s}=\mathbf{s}_t, \mathbf{a}=\pi_{\phi}(\mathbf{s}_t)} \nabla_{\phi} \pi_{\phi}(\mathbf{s}_t) \right]. \end{aligned} \quad (13)$$

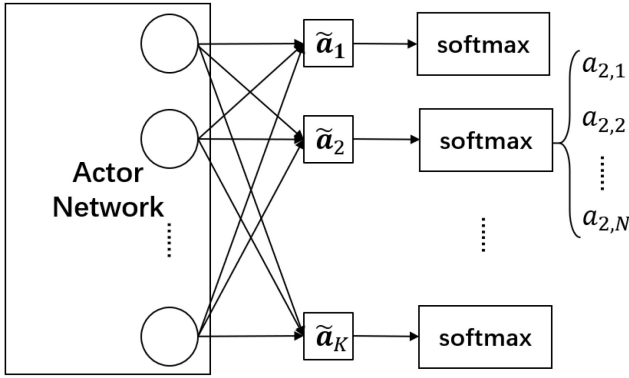


FIGURE 2. Actor's output layer with decoupled softmax activation.

The parameters of the target networks are updated with the soft update to ensure that the TD-error remains small:

$$\begin{aligned} \theta'_i &\leftarrow \tau\theta_i + (1 - \tau)\theta'_i, i = 1, 2; \\ \phi' &\leftarrow \tau\phi + (1 - \tau)\phi'. \end{aligned} \quad (14)$$

C. DEALING WITH RESOURCE CONSTRAINTS

To address the inter-slice resource constraints in Eq. (1), we propose a method by reconstructing the network architecture of DRL model with an additional regularization layer.

In this method, we embed a decoupled regularization layer into the output layer of the actor network, such that this layer becomes part of the end-to-end back propagation training of the neural network. Since the softmax function realizes for each \mathbf{a}_k the following projection

$$\sigma : \mathbb{R}^N \rightarrow \left\{ \mathbf{a}_k \in \mathbb{R}^N \mid a_{k,n} \geq 0, \sum_{n=1}^N a_{k,n} = 1 \right\},$$

the decoupled softmax layer well addresses the intra-cell inter-slice resource constraints $\sum_{n=1}^N a_{k,n} = 1, \forall k \in \mathcal{K}$ as shown in Fig. 2.

In summary, we provide the TD3-based DIRP algorithm with inter-agent coordination in Algorithm 1.

V. TRANSFER LEARNING-AIDED DIRP ALGORITHM

As discussed in Section IV-A, the DIRP algorithm achieves a good trade-off between reducing model complexity and capturing the inter-cell dependencies. However, each agent needs to learn the local policy from scratch and still faces the well-known challenge of the exploration-exploitation dilemma. The environment dynamics and state transitions are usually unknown at the early stage of training, and the agent cannot exploit its knowledge until the state-action space is exhaustively explored. Moreover, because the local model is trained on a specific data domain, the learned model is sensitive to domain shift (a change in the data distribution between an algorithm's training dataset, and the dataset that it encounters when deployed). This means, even a slight change in the environment may result in deteriorated performance, and the agent may face a long period of retraining time.

Algorithm 1 The DIRP Algorithm

```

1: Initialize parameters for critics  $Q_{\theta_1^k}, Q_{\theta_2^k}$  and actor  $\pi_{\phi^k}$ , with
   random parameters  $\theta_1^k, \theta_2^k, \phi^k, \forall k \in \mathcal{K}$ 
2: Initialize target networks  $\theta_1'^k \leftarrow \theta_1^k, \theta_2'^k \leftarrow \theta_2^k, \phi'^k \leftarrow \phi^k$ 
3: Initialize empty replay buffer  $\mathcal{B}_k$ 
4: Initialize  $\epsilon \in [0, 1]$  and decay  $d \in [0, 1]$  for  $\epsilon$ -greedy
   exploration
5: Define time periods  $\mathcal{H}^{(\text{Expl})}, \mathcal{H}^{(\text{Train})}, \mathcal{H}^{(\text{Eval})}$  for exploration,
   training, and evaluation phases, respectively
6: Repeat
7: for local agent  $k \in \mathcal{K}$  do
8:   Observe local state  $\mathbf{s}_k(t)$  and information  $\mathbf{c}_k(t)$ 
9:   Select and execute action:
10:  if  $t \in \mathcal{H}^{(\text{Expl})}$  then
11:     $\mathbf{a}_k(t) \leftarrow \text{random choice}$ 
12:  else if  $t \in \mathcal{H}^{(\text{Train})}$  then
13:     $\mathbf{a}_k(t) \leftarrow \begin{cases} \pi_k(\mathbf{s}_k(t), \mathbf{c}_k(t)) + \epsilon, & \text{if } U[0, 1] > \epsilon \\ \text{random choice}, & \text{otherwise} \end{cases}$ 
14:    where  $U[0, 1]$  is the generated random value
15:    following uniform distribution in  $[0, 1]$ .
16:     $\epsilon \leftarrow d\epsilon$ 
17:  else if  $t \in \mathcal{H}^{(\text{Eval})}$  then
18:     $\mathbf{a}_k(t) = \pi_k(\mathbf{s}_k(t), \mathbf{c}_k(t))$ 
19:  end if
20:  Observe next state  $\mathbf{s}_k(t+1)$ , received information
21:   $\mathbf{c}_k(t+1)$ , and compute  $r_k(t)$ 
22:  Store instance in  $\mathcal{B}_k$ :
23:   $((\mathbf{s}_k(t), \mathbf{c}_k(t)), \mathbf{a}(t), (\mathbf{s}_k(t+1), \mathbf{c}_k(t+1)), r_k(t))$ 
24:  if time to update networks then
25:    Sample mini-batch of  $B$  instances from  $\mathcal{B}_k$ 
26:    Compute target actions and targets using (10) and
27:    (11) respectively
28:    Update critic and actor based on (12) and (13)
29:    if  $t \bmod \text{policy\_delay}$  then
30:      Update target networks using (14)
31:    end if
32:  end if
33: end for

```

To overcome the above-addressed challenges, we raise a hypothesis that some common hidden pattern may exist in the critic and actor networks across different agents, and propose to enhance the developed coordinated MADRL algorithm with TL. We expect the TL to improve the model reproducibility and speed up the learning convergence by performing the following two major steps as demonstrated in Fig. 3:

- 1) *Centralized Training of a "Generalist"*: A centralized controller collects the samples from all local agents $((\mathbf{s}_k(t), \mathbf{c}_k(t)), \mathbf{a}_k(t), (\mathbf{s}_k(t+1), \mathbf{c}_k(t+1)), r_k(t)), \forall k \in \mathcal{K}$ for a time period $t = 0, \dots, T^{(\text{G})}$ and trains a generalized model by interacting with the environment based on the same model in training for all agents.
- 2) *Distributed Transfer Learning and Finetuning to the "Specialists"*: After time slot $T^{(\text{G})}$, we transfer the learned knowledge in the "generalist" to each local agent (i.e., the "specialists"), and finetune the customized model locally. The details of different types of transferable knowledge are provided later in Section V-B.

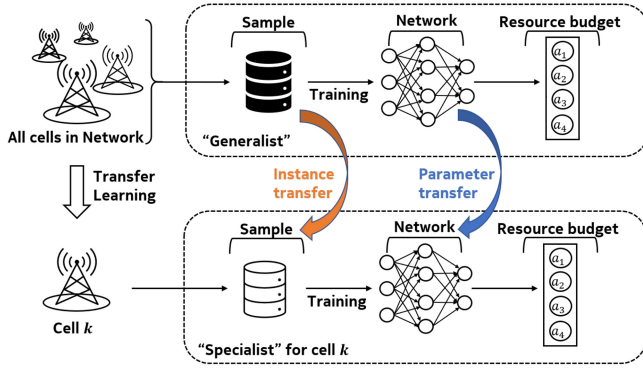


FIGURE 3. Generalist-to-Specialist transfer learning scheme.

A. TRANSFER LEARNING PROBLEM FORMULATION

Before introducing the TL problem in the context of MADRL, let us first introduce a general definition of transfer learning.

A domain $\mathcal{D} := \{\mathcal{X}, P(X)\}$ consists of a feature space \mathcal{X} and its probability distribution $P(X), X \in \mathcal{X}$. A task $\mathcal{T} := \{\mathcal{Y}, f(\cdot)\}$ consists of a label space \mathcal{Y} and a predictive function $f(\cdot)$, where $f(\cdot)$ can be written as $P(Y|X), Y \in \mathcal{Y}$ and $X \in \mathcal{X}$. Formally, the general definition of the TL is given below.

Definition 1 (Transfer Learning [17]): Given a source domain \mathcal{D}_S and a source learning task \mathcal{T}_S , a target domain \mathcal{D}_T and a target learning task \mathcal{T}_T , TL aims to improve the learning of the target predictive function $f_T(\cdot)$ in \mathcal{D}_T using the knowledge in \mathcal{D}_S and \mathcal{T}_S , where $\mathcal{D}_S \neq \mathcal{D}_T$, or $\mathcal{T}_S \neq \mathcal{T}_T$.

In the context of DRL, a domain $\mathcal{D} := \{\mathcal{S}, P(\mathbf{s})\}$ consists of the state space \mathcal{S} and its probability distribution $P(\mathbf{s}), \mathbf{s} \in \mathcal{S}$, while the task $\mathcal{T} := \{\mathcal{A}, \pi(\cdot)\}$ consists of the action space \mathcal{A} and a policy function $\pi(\cdot)$. In general, the policy π is a mapping from states to a probability distribution over actions. With the actor-critic method introduced in Section IV-B, the policy directly maps the state space to optimized action, thus, we have $\pi : \mathcal{S} \rightarrow \mathcal{A}$.

In the scope of our proposed generalist-to-specialist TL-DIRP algorithm, we introduce the following definitions of the source domain, source task, target domain, and target task.

- **Source Domain:** $\mathcal{D}_S := \mathcal{D}^{(G)}$ consists of the joint state and communicated message space $\mathcal{S}^{(G)} \times \mathbb{R}^N$ and its probability distribution $P(\mathbf{s}^{(G)}, \mathbf{c}^{(G)})$, where $\mathbf{s}^{(G)} \in \mathcal{S}^{(G)} := \cup_{k \in \mathcal{K}} \mathcal{S}_k$ and $\mathbf{c}^{(G)} \in \mathbb{R}^N$. The state $\mathbf{s}^{(G)}$ and message $\mathbf{c}^{(G)}$ are collected by the centralized controller from all local agents.
- **Source Task:** $\mathcal{T}_S := \mathcal{T}^{(G)}$ consists of the general action space $\mathcal{A}^{(G)}$ and the policy function $\pi^{(G)} : \mathcal{S}^{(G)} \times \mathbb{R}^N \rightarrow \mathcal{A}^{(G)}$. The general policy $\pi^{(G)}$ is trained on the instances collected by all agents.
- **Target Domain:** $\mathcal{D}_T := \mathcal{D}_k^{(S)}, k \in \mathcal{K}$ consists of the joint local state and communication message space $\mathcal{S}_k \times \mathbb{R}^N$ and its probability distribution $P(\mathbf{s}_k, \mathbf{c}_k)$, where $\mathbf{s}_k \in \mathcal{S}_k$ and $\mathbf{c}_k \in \mathbb{R}^N$.

- **Target Task:** $\mathcal{T}_T := \mathcal{T}_k^{(S)}, k \in \mathcal{K}$ consists of the local action space \mathcal{A}_k and local policy $\pi_k : \mathcal{S}_k \times \mathbb{R}^N \rightarrow \mathcal{A}_k$.

The problem of TL from a source DRL agent as a “generalist” to a set of target DRL agents, i.e., the local “specialists”, is formulated in Problem 2.

Problem 2: Given source domain $\mathcal{D}^{(G)} := \{\mathcal{S}^{(G)} \times \mathbb{R}^N, P(\mathbf{s}^{(G)}, \mathbf{c}^{(G)})\}$ and pretrained source task $\mathcal{T}^{(G)} := \{\mathcal{A}^{(G)}, \pi^{(G)}(\cdot)\}$, transfer learning aims to learn an optimal local policy for the target domain $\mathcal{D}_k^{(S)} := \{\mathcal{S}_k \times \mathbb{R}^N, P(\mathbf{s}_k, \mathbf{c}_k)\}$, $\forall k \in \mathcal{K}$ by leveraging the knowledge extracted from $(\mathcal{D}^{(G)}, \mathcal{T}^{(G)})$, as well as the knowledge exploited in the target domain $\mathcal{D}_k^{(S)}$. The problem is given by

$$\begin{aligned} \max_{\pi_k | \pi_k^{(0)} = \Lambda(\pi^{(G)})} \quad & \mathbb{E}_{\pi_k} \left[\sum_{t=0}^T \gamma_k^t \tilde{r}_k(\mathbf{s}_k(t), \mathbf{c}_k(t), \mathbf{a}_k(t)) \right] \\ \text{s.t.} \quad & (\mathbf{s}_k, \mathbf{c}_k, \mathbf{a}_k) \in \Omega(\mathcal{D}^{(G)}, \mathcal{D}_k^{(S)}, \mathcal{A}^{(G)}, \mathcal{A}_k). \end{aligned} \quad (15)$$

where $\Lambda(\pi^{(G)})$ is the *policy transfer strategy* which maps the pretrained source policy $\pi^{(G)}$ to an initial local policy $\pi_k^{(0)}$, while $\Omega(\mathcal{D}^{(G)}, \mathcal{D}_k^{(S)}, \mathcal{A}^{(G)}, \mathcal{A}_k)$ is the *instance transfer strategy* which extracts the instances from the source domain and combines them with the experienced instances from the target domain.

B. TRANSFER LEARNING APPROACHES

The problem defined in Eq. (15) offers various options for transferable knowledge:

- **Pretrained Model Transfer:** The policy transfer strategy $\Lambda(\cdot)$ simply maps the pretrained source policy to itself, i.e., the local agent uses the pretrained general policy $\pi^{(G)}$ as the initial policy $\pi_k^{(0)}$ and finetunes it by further interacting with the environment with locally made decisions.
- **Feature Extraction:** $\Lambda(\cdot)$ keeps partial knowledge of $\pi^{(G)}$. In DRL, the policy $\pi^{(G)}(\mathbf{s}^{(G)}, \mathbf{c}^{(G)} | \boldsymbol{\phi}^{(G)})$ is characterized by the pretrained parameters (weights) of the neural networks. Feature extraction freezes partial of the layers (usually the lower layers) of the pretrained neural networks while leaving the rest of them to be randomly initialized.
- **Instance Transfer:** Except for the instances from the target domain, the agent also trains its policy using the extracted instances from the source domain. The instance transfer strategy $\Omega(\cdot)$ decides which instances are chosen from the source domain to be combined with the instances from the target domain in the local replay buffer.

The above-mentioned knowledge from the source domain and task can be transferred separately or in a combined manner. In this paper, we focus on studying the following three TL schemes:

- **Pretrained Model Transfer Only:** Each local agent k uses the pretrained general policy $\pi^{(G)}$ to initialize the local policy $\pi_k^{(0)}$. With the actor-critic method

described in Section IV-B, we simply load the pretrained parameters of the actor and critic networks from the generalist to the local agents. However, when the difference between the source and target domain is large, the local agent still needs extensive exploration to finetune the general policy to a customized local policy.

- *Instance Transfer Only*: Each local agent offloads a set of selected instances in the source domain from the centralized controller to the local replay buffer. Then, the local agent trains a policy from scratch with the replay buffer containing mixed offline instances from the source domain and the experienced online instances in the target domain. In this paper, we select the instances collected from the exact same local agent. Future work includes the similarity analysis between agents and instance selection from similar agents, which falls into the subject of *domain adaptation* [42].
- *Combined Model and Instance Transfer*: To fully exploit the transferable knowledge, we combine the pretrained model transfer and instance transfer. Firstly, each local agent retrieves $\pi^{(G)}$ from the centralized controller and uses it to initialize the local policy $\pi_k^{(0)}$. Then, we further investigate two options for local finetuning:
 - *Online Finetuning With Mixed Replay Buffer*: The local agent further online finetunes the policy with the replay buffer containing both the offloaded instances from the source domain and the locally experienced instances from the target domain.
 - *Offline Finetuning With Offloaded Instances & Online Finetuning With Experienced Instances*: The local agent first offline finetunes $\pi^{(G)}$ with the offloaded instances. Then, the offline finetuned model is used to initialize $\pi_k^{(0)}$ and further finetuned online with the locally experienced instances in the target domain.

Note that our experiments focus on the pretrained model transfer and instance transfer, while do not include the feature extraction. This is because, feature exaction usually performs well when the target domain is highly similar to the source domain. However, in general, the similarity between the generalist’s domain and the specialist’s domain is not sufficiently high. Thus, the feature exaction method may better suit the scenario of inter-agent TL, while it may not be appropriate for generalist-to-specialist knowledge transfer.

We illustrate the TL-DIRP algorithm with a combined model and instance transfer in Algorithm 2.

VI. PERFORMANCE EVALUATION

In this section, we evaluate the performance of the proposed methods for inter-cell slicing resource partitioning introduced in Sections IV and V with a system-level simulator [43], which mimics real-life network scenarios with customized network slicing traffic, user mobility, and network topology.

To implement our proposed DRL solution, we build in the simulator a network with 4 sites (12 cells) covering an urban area of Helsinki city, as demonstrated in Fig. 4, consisting

Algorithm 2 Transfer Learning-Aided DIRP Algorithm

- 1: **I. Generalist training in centralized controller**
- 2: Initialize generalist’s critics $Q_{\theta_1^{(G)}}, Q_{\theta_2^{(G)}}$ and actor $\pi_{\phi^{(G)}}$ with random parameters $\theta_1^{(G)}, \theta_2^{(G)}, \phi^{(G)}$
- 3: Initialize target networks $\theta_1'^{(G)} \leftarrow \theta_1^{(G)}, \theta_2'^{(G)} \leftarrow \theta_2^{(G)}, \phi'^{(G)} \leftarrow \phi^{(G)}$
- 4: Initialize empty replay buffer $\mathcal{B}^{(G)}$
- 5: Define time periods $\mathcal{H}^{(G)}, \mathcal{H}^{(S)}$ for generalist training and specialist finetuning respectively
- 6: **for** $t \in \mathcal{H}^{(G)}$ **do**
- 7: Collect observations of local states $\mathbf{s}_k(t)$ and
- 8: received information $\mathbf{c}_k(t), \forall k \in \mathcal{K}$
- 9: Use general policy $\pi^{(G)}$ to select and execute action
- 10: $\mathbf{a}_k(t), \forall k \in \mathcal{K}$
- 11: Observe the next local states $\mathbf{s}_k(t+1)$ and information
- 12: $\mathbf{c}_k(t+1)$, compute local rewards $r_k(t), \forall k \in \mathcal{K}$
- 13: Store K instances in replay buffer $\mathcal{B}^{(G)}$
- 14: Train and update the general critics $Q_{\theta_i^{(G)}}, i = 1, 2,$
- 15: actor $\pi^{(G)}$, and target critics $Q_{\theta_i'^{(G)}}, i = 1, 2$ and actor
- 16: $\pi_{\phi'^{(G)}}$ using the TD3 algorithm in Section IV-B
- 17: **end for**
- 18: **II. Specialist finetuning in local agents**
- 19: Initialize parameters for critics $Q_{\theta_1^k}, Q_{\theta_2^k}$ and actor π_{ϕ^k} with
- 20: $\theta_1^k \leftarrow \theta_1^{(G)}, \theta_2^k \leftarrow \theta_2^{(G)}, \phi^k \leftarrow \phi^{(G)}, \forall k \in \mathcal{K}$
- 21: Initialize target networks $\theta_1'^k \leftarrow \theta_1^k, \theta_2'^k \leftarrow \theta_2^k, \phi'^k \leftarrow \phi^k$
- 22: **for** $t \in \mathcal{H}^{(S)}$ **do**
- 23: **for** Local agent $k \in \mathcal{K}$ **do**
- 24: Finetune local policy with Algorithm 1 (except for
- 25: the initialization steps)
- 26: **end for**
- 27: **end for**

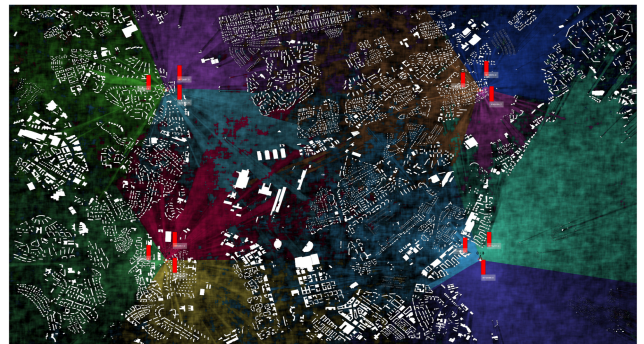


FIGURE 4. Network environment setup with 12 cells.

of 4 three-sector macro sites. All cells are deployed using LTE radio technology with 2.6 GHz. We use the realistic radio propagation model Winner+ [44].

The network is built up with $N = 4$ network slices, with per-slice throughput requirements of $\phi_1^* = 4$ MBit/s, $\phi_2^* = 1$ MBit/s, $\phi_3^* = 3$ MBit/s, and $\phi_4^* = 0.5$ MBit/s and per-slice delay requirements of $d_1^* = 1$ ms, $d_2^* = 1.5$ ms, $d_3^* = 2$ ms, and $d_4^* = 1$ ms respectively. All cells in the network have a fixed bandwidth of 20 MHz.

We define four groups of user equipments (UEs) associated with each defined slice respectively, i.e., 16 groups of

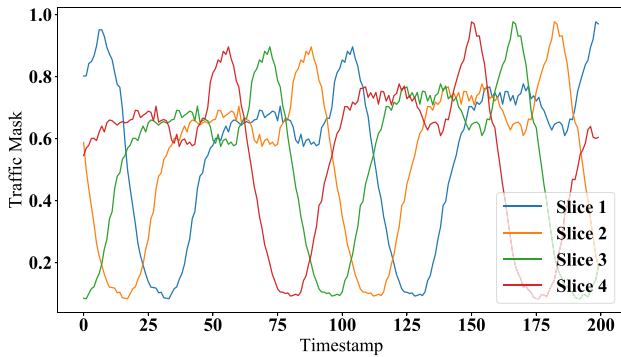


FIGURE 5. The first two days of a three-week traffic mask.

UEs in total, all with the maximum group size of 10. UEs are moving uniformly randomly within the defined moving sphere of each group. The positions and moving radius of UEs groups are defined heterogeneously to ensure that each site can serve UE from all slices. To imitate the time-varying traffic pattern, we also apply a time-dependent traffic mask $\tau_n(t) \in [0, 1]$ for each slice $n \in \mathcal{N}$ to scale the total number of UEs in the scenario. In Fig. 5, we demonstrate the changes of the first 2 days of a three-week traffic mask. The UE traffic volume is updated every timestamp, which corresponds to 15 minutes in real time, also known as the typical KPI reporting time in OAM. In the experiments, the entire traffic mask is extended and periodically repeated after every 2016 timestamps corresponding to the three-week time period (96 timestamps per day).

A. SCHEMES AND BASELINES TO COMPARE

For performance evaluation, we compare the proposed DIRP and TL-DIRP algorithms with the following three baselines:

- *BL-Cen*: centralized DRL approach solving Eq. (5) referring to [12]. We assume that a single agent has full observation of the global state $\mathbf{s} \in \mathcal{S}$, computes the global reward and makes the decision of the slicing resource partitioning for all agents $\mathbf{a} \in \mathcal{A}$.
- *BL-Dist*: distributed DRL approach without inter-agent coordination referring to [14].
- *BL-Heur*: a traffic-aware heuristic approach that assumes perfect knowledge about per-slice traffic demand, and dynamically adapts to the current per-slice traffic amount. It is implemented by dividing the resource in each cell $k \in \mathcal{K}$ to each slice proportionally to the amount of traffic demand per slice.

The DRL-based schemes to evaluate and compare are summarized in Table 2.

Similarly, to evaluate the TL-DIRP algorithm and compare between different types of knowledge to transfer, we implement the proposed TL method in Section V, i.e., centralized training of a generalist and then distributed finetuning to specialist. We compare different transferable knowledge: instances, pretrained model, and combined instances and pretrained model. In addition, to ensure a safer exploration

and better performance during online training, we perform the offline finetuning using the transferred instance before the online training in each local agent.

- *Gen*: centralized training of a general policy in the centralized controller based on the collected samples from all local agents, as described in Algorithm 2.
- *Spec*: distributed finetuning of the specialists with full knowledge transfer. Each local agent initializes its critic and actor networks with the generalist's model parameters. It also initializes the local replay buffer with the offloaded selected instances from the generalist's buffer.
- *Spec-Instance*: distributed finetuning of the specialists with instance transfer only. The model parameters in each local agent are randomly initialized.
- *Spec-Model*: distributed finetuning of the specialists with model transfer only. Each local agent initializes its critic and actor networks by loading the generalist's model parameters, while the local buffer is initialized as an empty queue.
- *TL-DIRP*: In addition to Spec (full knowledge transfer), we apply the offline finetuning based on the transferred instances before the online training.

Note that for “generalist-to-specialist” TL schemes with complete knowledge we apply both max-min fairness and logarithmic utilities as local reward r_k for $k \in \mathcal{K}$ respectively, as:

- *TL-DIRP-Maxmin*: TL-DIRP approach with max-min fairness reward bases on Eq. (6).
- *TL-DIRP-Log*: TL-DIRP approach with on logarithmic utility reward based on Eq. (7).

B. HYPERPARAMETERS USED FOR LEARNING

As for DRL training, we use multi-layer perception (MLP) architecture for actor-critic networks of TD3 algorithm. In BL-Cen scheme, the models of the actor and critic networks are both built up with 3 hidden layers, with the number of neurons (384, 192, 64) and (324, 144, 64), respectively. While for BL-Dist and DIRP schemes, both actor-critic networks only have 2 hidden layers, with the number of neurons (48, 24) and (64, 24), respectively. In all schemes, the learning rate of actor and critic are 0.0005 and 0.001 respectively with Adam optimizer and training batch size of 32. We choose a small DRL discount factor $\gamma = 0.1$, since the current action has a strong impact on the instantaneous reward while a weaker impact on the future reward. For the distributed DRL approaches, we only apply 100 steps for exploration, while for the centralized approaches we apply 500 steps of exploration, since the centralized agent has much higher dimensions of state and action. After the exploration phase, we apply 5000 steps for training, and the final 500 steps for evaluation of all approaches.

In TL training, we apply the same DRL settings. For TL training setup, we set 100 steps for exploration, 5000 steps for learning, and 500 steps for evaluation in Gen and Spec-Model schemes, while in other TL execution schemes, we

TABLE 2. Comparison of dimensions of DRL models used in simulation.

	BL-Cen	BL-Dist	DIRP & TL-DIRP
State	Global state $\mathbf{s} \in \mathbb{R}^{240}$	Local state $\mathbf{s}_k \in \mathbb{R}^{20}$	Local state with extracted message $[\mathbf{s}_k, \mathbf{c}_k] \in \mathbb{R}^{24}$
Action	Global action $\mathbf{a} \in [0, 1]^{48}$	Local action $\mathbf{a}_k \in [0, 1]^4$	Local action $\mathbf{a}_k \in [0, 1]^4$
Reward	Global reward r_G^m in Eq. (3)	Local reward r_k^m in Eq. (6)	Local reward r_k^m in Eq. (6)

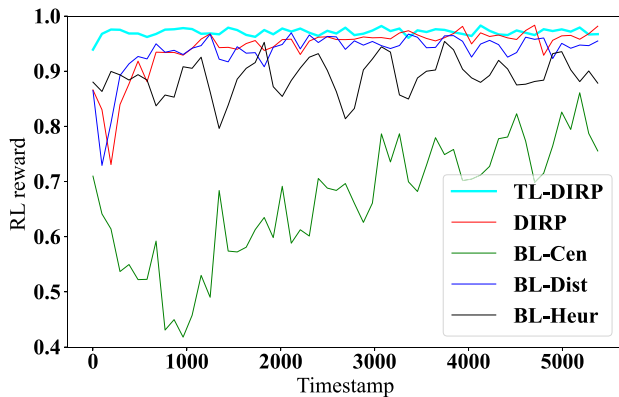


FIGURE 6. Comparison of reward among schemes.

skip the exploration phase. The result of each process is derived from the average of 3 times of experiments.

In this work, we apply an orientated exploration strategy that chooses the new action under the recommendation of the traffic-aware heuristic policy, namely, the heuristic baseline BL-Heur. The reason is that we observe that BL-Heur provides sub-optimal performance without any training process. At the beginning of the exploration phase, the probability of using traffic-aware exploration is 0.5, and that of random exploration is also 0.5. Then, during the exploration, the probability of traffic-aware exploration gradually increases, and that of random exploration decreases.

C. PERFORMANCE COMPARISON

1) COMPARISON OF THE DISTRIBUTED MADRL SCHEMES

In this comparison, we apply the reward design for max-min fairness to all approaches, i.e., global reward based on Eq. (3) for BL-Cen and local reward based on Eq. (4) for BL-Dist and DIRP. While for comparison between the different reward functions, we implement DIRP algorithm with both types of local reward r_k based on Eq. (6) and Eq. (7).

Fig. 6 demonstrates the comparison of max-min fairness reward Eq. (3) during the training process among the baseline schemes BL-Cen, BL-Dist, BL-Heur, and the proposed DIRP and TL-DIRP algorithms.

As shown in Fig. 6, TL-DIRP provides the best performance among all approaches in terms of faster convergence, higher start point, and higher robustness after convergence.

While in comparison to baselines, DIRP algorithm achieves significantly better global reward than BL-Heur after convergence. Note that BL-Heur is already a well-performed baseline because it assumes perfect traffic awareness and offers all resources to the UEs. On the other

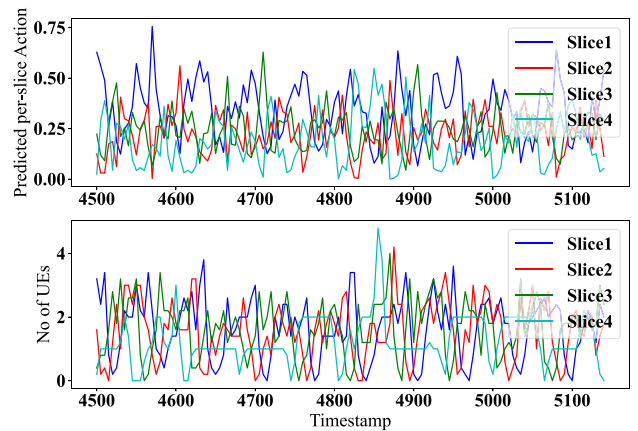


FIGURE 7. Adaptive action to traffic mask after training.

hand, BL-Cen fails to achieve performance as good as DIRP within the same training time. As Table 2 indicates, the dimensions of the state and action spaces of BL-Cen are much higher than the distributed approaches, making the training process more difficult for large-scale networks. Not only converges BL-Cen slower, but it also often experiences poor performance at the early stage of training. The training curves are turbulent, corresponding to the time-varying traffic demand in Fig. 5, while DIRP is more robust compared to BL-Heur and BL-Cen.

In comparison between the two distributed schemes, according to Fig. 6, DIRP outperforms BL-Dist scheme within the same training time period in terms of both converged global reward and convergence rate, which verifies the advantage of inter-agent coordination.

Fig. 7 shows the predicted action, i.e., per-slice resource partitioning as the ratio, and the actual traffic amount of DIRP in cell $k = 5$ after convergence. It verifies that the DRL approach well adapts its predicted actions to the dynamic network traffic demand with respect to different slice-specific QoS requirements.

Although DIRP shows better performance than baselines, it still faces two major challenges: slow convergence and oscillation. In Fig. 6, we show that TL-DIRP overcomes these challenges by transferring prelearned knowledge. In particular, TL-DIRP achieves a much higher reward from the beginning of the learning process and quickly converges after a few hundred timestamps, while DIRP converges much slower because each local agent needs to learn from scratch. TL-DIRP outperforms DIRP in terms of both convergence rate and converged performance within the same time period.

Fig. 6 shows the evolving algorithms' performance during the training and testing process, while in the following, let us take a deeper look into the distributions of the converged

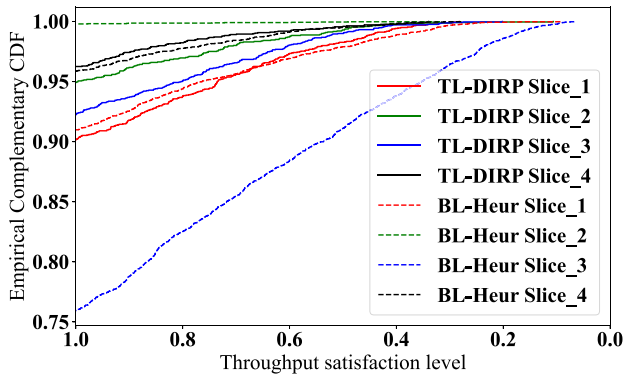


FIGURE 8. Comparing throughput QoS between TL-DIRP and BL-Heur.

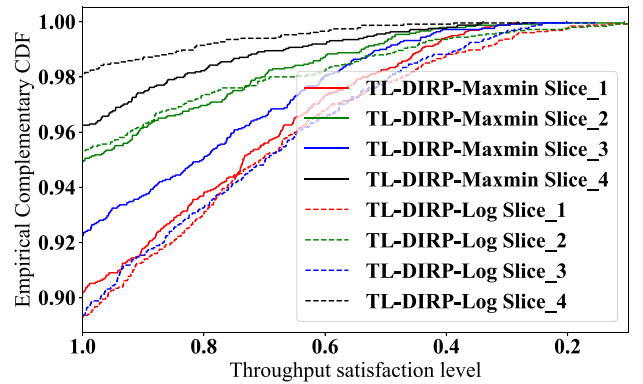


FIGURE 10. Comparing throughput QoS between utilities.

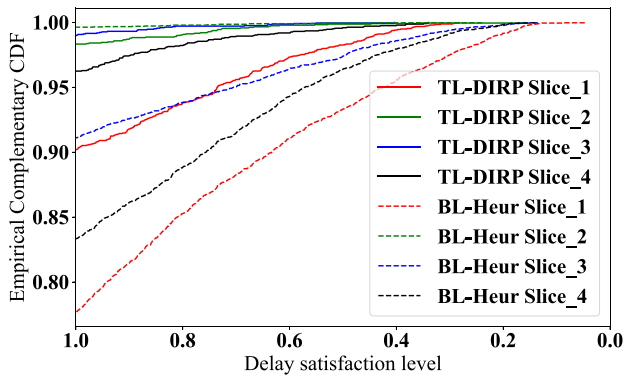


FIGURE 9. Comparing delay QoS between TL-DIRP and BL-Heur.

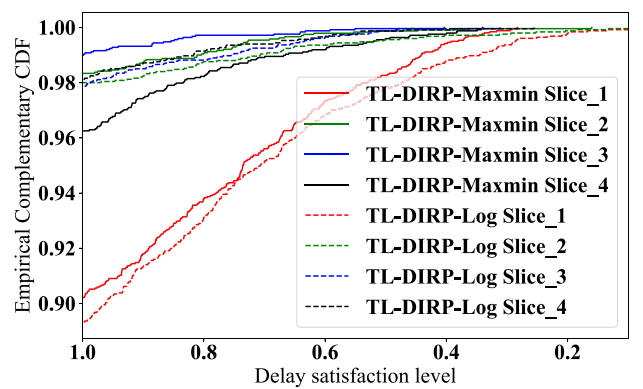


FIGURE 11. Comparing delay QoS between utilities.

service quality in terms of throughput and delay satisfaction level for each slice. Fig. 8 and Fig. 9 illustrate the empirical complementary cumulative distribution function (CDF) (or called survival function) which equals $1 - F_X(x)$, where $F_X(x)$ denotes the CDF of per-slice throughput and delay satisfaction level between TL-DIRP and BL-Heur schemes, respectively.

Fig. 8 shows that TL-DIRP achieves 14% higher the worst-case throughput QoS among all slices than the traffic-aware baseline BL-Heur. It also guarantees that all the slices achieve a throughput satisfaction level above 90%, while BL-Heur serves Slice 3 with only 75% throughput satisfaction level.

Similar observation can be made for the delay satisfaction level in Fig. 9. TL-DIRP provides over 90% of the delay satisfaction level for all slices, while BL-Heur serves Slice 1 and 3 with only 77% and 83% respectively. In terms of the average delay satisfaction level over all slices, TL-DIRP achieves over 96% while BL-Heur only 88%. We observe that TL-DIRP attempts to fulfill more critical requirements by compromising resources from the less demanding slices while remaining sufficient satisfaction levels in others.

2) COMPARISON BETWEEN REWARD FUNCTION WITH TWO UTILITIES

Fig. 10 and Fig. 11 compare the two designs of the reward function, corresponding to max-min fairness Eq. (6) and

maximizing average logarithmic utilities Eq. (7), respectively. They demonstrate the empirical complementary CDF of QoS in terms of throughput and delay satisfaction level for TL-DIRP with both reward functions. The results show that max-min fairness gives the maximum protection to the slice with the weakest performance, such that the minimum per-slice satisfaction level over all slices achieves 90% for both throughput and delay, while maximizing average logarithmic utilities provides slightly lower satisfaction levels, about 89% for both throughput and delay, but higher maximum per-slice throughput satisfaction levels. This is because the logarithmic utility tends to distribute the resource more efficiently than max-min fairness, i.e., allocating more resources to the slice that can improve the averaged performance over all slices.

From an engineering perspective, max-min fairness is preferred for scenarios that require sufficiently good performance for all slices, especially those highly demanding ones. While the logarithmic utility is more suitable for cases that desire higher resource efficiency.

3) COMPARISON OF THE TRANSFER LEARNING METHODS

Fig. 12 illustrates the evolving rewards during the training and testing processes with different TL methods. Note that this comparison is based on max-min fairness in all TL methods. Here we aligned the training process with

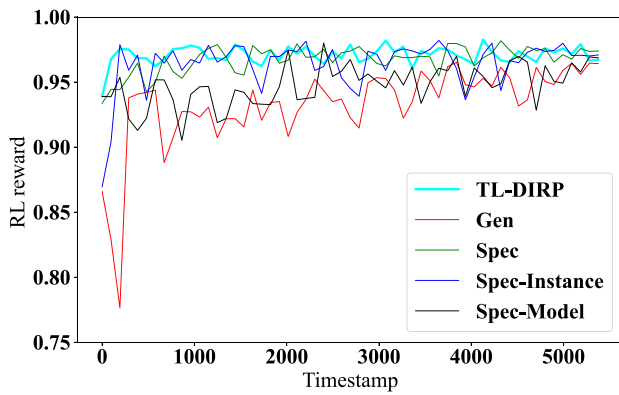


FIGURE 12. Comparing of reward among TL schemes.

TABLE 3. Performance comparison among different schemes.

	RL Reward	Min Slice Average Throughput Satisfy	Min Slice Average Delay Satisfy
BL-Cen	0.801	0.738	0.739
BL-Dist	0.948	0.952	0.962
BL-Heur	0.891	0.903	0.902
DIRP	0.968	0.967	0.967
Gen	0.961	0.960	0.961
Spec	0.972	0.968	0.968
Spec-Instance	0.971	0.970	0.974
Spec-Model	0.962	0.965	0.966
TL-DIRP	0.973	0.971	0.971

the Spec scheme for comparison. The results are derived from the average of 3 independent instances of experiments. Spec with complete knowledge transfer leads to higher reward and robustness compared to Spec-Instance and Spec-Model schemes with partial knowledge at the early stage of the training process, while in the latter two schemes, TL also helps in terms of convergence rate, compared with Gen. Furthermore, with offline finetuning Spec-Finetune provides better performance with faster convergence and higher reward within the same training time. In most of the TL schemes, we observe that each specialist agent improves its performance with local finetuning from a higher starting point, which helps avoid risky action choices during exploration. With Spec-Instance, the agents behave the worst at the beginning of the training but converge fast later. On the other hand, Spec-Model also suffers from a weaker performance at the beginning and takes a longer time to learn. Eventually, Spec-Instance converges to a similar performance as Spec while Spec-Model achieves a slightly worse performance. Our guess is that there is still a substantial difference between the source domain and the target domain. Without transferring sufficient instances in the source domain (instances following similar distribution to the target domain), the initialized general policy cannot quickly adapt to the target task. Moreover, introducing offline finetuning with the transferred instances to the TL scheme further improves the performance by providing even faster convergence and more robust training.

In Fig. 13, we plot the change of local reward in each cell during the complete TL procedure from generalist training

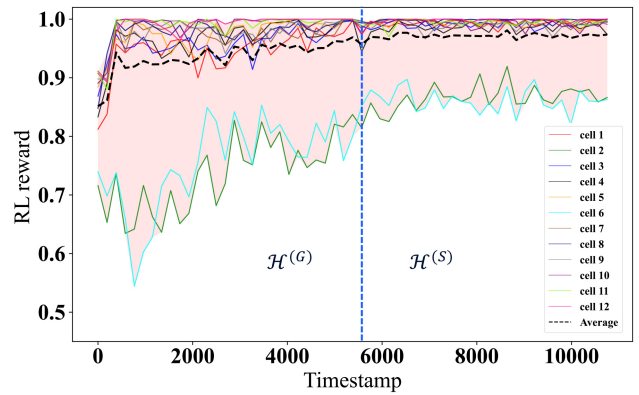


FIGURE 13. Change of local reward during TL scheme.

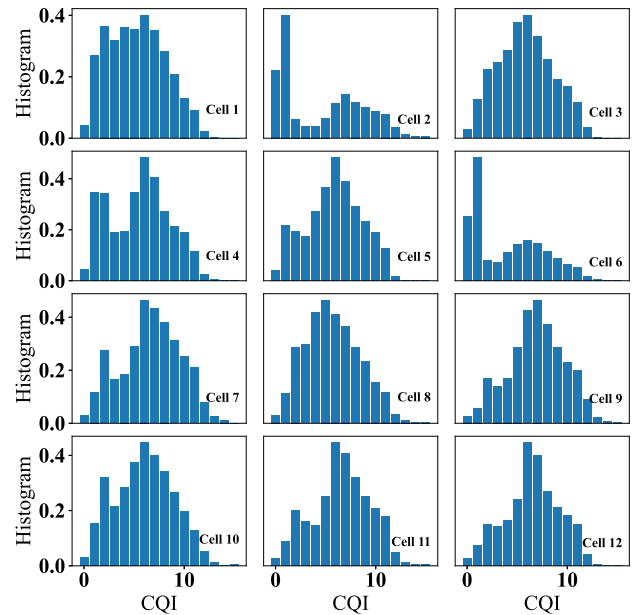


FIGURE 14. Comparison of channel quality indicator (CQI) distribution between cells.

to specialist finetuning as described in Algorithm 2. During the time in $\mathcal{H}^{(G)}$, the local rewards achieved by the generalist agent converge to a generally good reward over all cells. Later, in the local finetuning period $\mathcal{H}^{(S)}$, the Spec scheme further finetunes the general agent locally and concludes better performance in each cell. The averaged local reward in $\mathcal{H}^{(S)}$ also indicates better robustness under time-varying traffic demand. We can also observe that the rewards from two cells are always lower compared to others during $\mathcal{H}^{(G)}$, and achieve relatively poor performance after knowledge transfer in $\mathcal{H}^{(S)}$. Fig. 14 shows the comparison of channel quality indicator (CQI) distributions from all cells, it is clear to see that in cell 2 and cell 6 which derive poorer performance as shown in Fig. 13 correspondingly, the CQI histograms are significantly different compared to others. The difference in user distribution or radio propagation can make the “generalist” ambiguous on learning a general policy for all cells, and the derived policy is better for handling the samples from others. Thus, during

$\mathcal{H}^{(G)}$, the rewards in these two cells are lower than others, while in $\mathcal{H}^{(S)}$, the performances in these two cells get better with local finetuning yet still worse than the others.

Summarized comparisons of the average performance metrics among all schemes in the testing phase are listed in Table 3. We can see that TL-DIRP as offline finetuned Spec provides the best performance in terms of the desired RL reward and minimum (worst-case) per-slice throughput satisfaction level among the schemes, while Spec-Instance provides a slightly better minimum per-slice delay satisfaction level. Moreover, TL-DIRP encourages a more balanced service quality between all slices in comparison to TL-DIRP-Log. It is also worth noting that the inference time for a pretrained distributed DRL model to make a local decision is less than 4 milliseconds due to the small sizes of our defined neural networks.

D. KEY TAKEAWAYS

In the following, we summarize the takeaways from our numerical analysis:

- *Distributed vs. Centralized:* For conventional DRL algorithms, the distributed scheme demonstrates good learning capability for adapting to slice-aware traffic and providing good service quality in the defined network scenario with 12 cells, while the centralized scheme fails to converge to a good reward within the same training time because of its high model complexity and high dimensional state and action spaces. In fact, the larger the scale the network has, the higher the gain the distributed schemes achieve when compared with the centralized approach.
- *Inter-Agent Coordination:* The DIRP algorithm with inter-agent coordination and letting the multiple agents share load information provides better performance than the distributed DRL in terms of converged reward and convergence rate while maintaining lower model complexity.
- *The Advantages of Transfer Learning:* Our proposed TL-DIRP algorithm further improves the converged reward of DIRP with about 11.5% higher start point, 87.5% faster convergence, and lower exploration cost. It is worth noting that, the converged performance of the TL-DIRP algorithm has higher robustness than DIRP without TL. It also provides about 15% higher QoS satisfaction level for the most critical slice and an 8.8% higher average slice QoS fully satisfaction level than the traffic-aware baseline.
- *The Needs of “Generalist-to-Specialist” Transfer:* During the “generalist” training process of TL-DIRP, the difference in CQI between cells make the learning of general policy ambiguous, and the agents from different cell CQI derive poorer performance than others. Later in “specialist” all agents grant higher reward and robustness with local finetuning.
- *Comparison Between Two Reward Functions:* The proposed TL-DIRP approach with reward based on

max-min fairness and logarithmic utility can both provide sufficiently good performance among all slice QoS. However, max-min fairness reward achieves better QoS for critical slice requirements by occupying resources from the slices with less critical requirements, while logarithmic utility provides higher resource efficiency. From the engineering perspective, different reward definitions can be chosen for variant use cases.

- *How to Transfer:* As for the transferable knowledge, TL scheme with combined model and instance transfer enhanced by offline finetuning provides the best performance, in terms of both the starting point and the convergence rate. As expected, when transferring instances only, the local agents still need to train from scratch and suffer from the low performance at the beginning. When transferring the pretrained model, the performance at the beginning is slightly better but requires a longer time to converge. Our guess is that there is substantial difference between the source and target domains according to the CQI distribution of cells. Without transferring sufficient instances from the source domain (instances following similar distribution to the target domain), the initialized general policy cannot adapt quickly to the target task. Moreover, by introducing an offline finetuning with the transferred instances, TL-DIRP provides a further performance improvement to the TL scheme without offline finetuning in terms of higher start point and faster convergence.

VII. CONCLUSION

In this paper, we formulated the dynamic inter-cell resource partitioning problem to meet the slice-aware service requirements by jointly optimizing the inter-cell inter-slice resource partitioning. First, we proposed the DIRP algorithm to solve the problem with inter-agent coordination. To further improve the algorithm transferability, we designed the TL-DIRP algorithm by introducing a generalist-to-specialist TL framework with different types of transferable knowledge. We evaluated the proposed solutions with a 12 cells network scenario in a system-level simulator. The evaluation results showed that the TL-DIRP algorithm provides better slice-aware service performance than the existing baseline approaches. Besides, using TL in MADRL improves the training performances in different aspects, e.g., higher start point, faster convergence speed, and higher asymptote. We also investigated two reward definitions with max-min fairness and logarithmic utility in TL-DIRP and found that different rewards should be chosen for variant purposes in practical use cases.

As an extension to the “generalist-to-specialist” TL scheme, future works include inter-agent TL, which enables knowledge transfer from a pretrained DRL agent to another, e.g., transferring knowledge from a pretrained cell to a newly deployed cell. However, as we observed in numerical experiments, transferring knowledge between agents with different

domains and tasks may deteriorate the performance at the early training phase of TL, or, sometimes even cause negative transfer. Thus, quantitative analysis needs to be developed to detect similar DRL agents for efficient knowledge transfer.

REFERENCES

- [1] T. Hu, Q. Liao, Q. Liu, D. Wellington, and G. Carle, "Inter-cell slicing resource partitioning via coordinated multi-agent deep reinforcement learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2022, pp. 3202–3207.
- [2] A. Ksentini and N. Nikaein, "Toward enforcing network slicing on RAN: Flexibility and resources abstraction," *IEEE Commun. Mag.*, vol. 55, no. 6, pp. 102–108, Jun. 2017.
- [3] P. L. Vo, M. N. Nguyen, T. A. Le, and N. H. Tran, "Slicing the edge: Resource allocation for RAN network slicing," *IEEE Wireless Commun. Lett.*, vol. 7, no. 6, pp. 970–973, Dec. 2018.
- [4] R. A. Addad, M. Bagaa, T. Taleb, D. Dutra, and H. Flink, "Optimization model for cross-domain network slices in 5G networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 5, pp. 1156–1169, May 2020.
- [5] H. Beshley, M. Beshley, M. Medvetzkyi, and J. Pyrih, "QoS-aware optimal radio resource allocation method for machine-type communications in 5G LTE and beyond cellular networks," *Wireless Commun. Mobile Comput.*, vol. 2021, May 2021, Art. no. 9966366.
- [6] F. Fossati, S. Moretti, P. Perny, and S. Secci, "Multi-resource allocation for network slicing," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1311–1324, Jun. 2020.
- [7] T. Ma, Y. Zhang, F. Wang, D. Wang, and D. Guo, "Slicing resource allocation for eMBB and URLLC in 5G RAN," *Wireless Commun. Mobile Comput.*, vol. 2020, Jan. 2020, Art. no. 6290375.
- [8] "System architecture for the 5G system (5GS), V17.4.0," 3GPP, Sophia Antipolis, France, 3GPP Rep. TS 23.501, Mar. 2022.
- [9] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proc. 15th ACM Workshop Hot Topics Netw.*, 2016, pp. 50–56.
- [10] Y. Liu, J. Ding, and X. Liu, "A constrained reinforcement learning based approach for network slicing," in *Proc. IEEE 28th Int. Conf. Netw. Protocols (ICNP)*, 2020, pp. 1–6.
- [11] Q. Liu, T. Han, N. Zhang, and Y. Wang, "DeepSlicing: Deep reinforcement learning assisted resource allocation for network slicing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2020, pp. 1–6.
- [12] R. Li et al., "Deep reinforcement learning for resource management in network slicing," *IEEE Access*, vol. 6, pp. 74429–74441, 2018.
- [13] I. Alqerm and B. Shihada, "A cooperative Online learning scheme for resource allocation in 5G systems," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2016, pp. 1–7.
- [14] N. Zhao, Y.-C. Liang, D. T. 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.
- [15] Y. Shao, R. Li, Z. Zhao, and H. Zhang, "Graph attention network-based DRL for network slicing management in dense cellular networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2021, pp. 1–6.
- [16] H. Nie, S. Li, and Y. Liu, "Multi-agent deep reinforcement learning for resource allocation in the multi-objective HetNet," in *Proc. Int. Wireless Commun. Mobile Comput. (IWCMC)*, 2021, pp. 116–121.
- [17] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.
- [18] C. T. Nguyen et al., "Transfer learning for future wireless networks: A comprehensive survey," 2021, *arXiv:2102.07572*.
- [19] M. Wang, Y. Lin, Q. Tian, and G. Si, "Transfer learning promotes 6G wireless communications: Recent advances and future challenges," *IEEE Trans. Rel.*, vol. 70, no. 2, pp. 790–807, Jun. 2021.
- [20] C. Parera, Q. Liao, I. Malanchini, C. Tatino, A. E. C. Redondi, and M. Cesana, "Transfer learning for tilt-dependent radio map prediction," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 2, pp. 829–843, Jun. 2020.
- [21] M. E. Taylor, P. Stone, and Y. Liu, "Transfer learning via inter-task mappings for temporal difference learning," *J. Mach. Learn. Res.*, vol. 8, no. 1, pp. 2125–2167, 2007.
- [22] F. Zhuang et al., "A comprehensive survey on transfer learning," *Proc. IEEE*, vol. 109, no. 1, pp. 43–76, Jan. 2021.
- [23] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *J. Mach. Learn. Res.*, vol. 10, no. 56, pp. 1633–1685, 2009.
- [24] Z. Zhu, K. Lin, A. K. Jain, and J. Zhou, "Transfer learning in deep reinforcement learning: A survey," 2020, *arXiv:2009.07888*.
- [25] A. M. Nagib, H. Abou-Zeid, and H. S. Hassanein, "Transfer learning-based accelerated deep reinforcement learning for 5G RAN slicing," in *Proc. IEEE 46th Conf. Local Comput. Netw. (LCN)*, 2021, pp. 249–256.
- [26] T. Mai, H. Yao, N. Zhang, W. He, D. Guo, and M. Guizani, "Transfer reinforcement learning aided distributed network slicing optimization in industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 18, no. 6, pp. 4308–4316, Jun. 2022.
- [27] H. Zafar, Z. Utkovski, M. Kasparick, and S. Stańczak, "Transfer learning in multi-agent reinforcement learning with double Q-networks for distributed resource sharing in V2X communication," 2021, *arXiv:2107.06195*.
- [28] "Technical specification group services and system aspects; management and orchestration; concepts, use cases and requirements, V17.2.0," 3GPP, Sophia Antipolis, France, 3GPP Rep. TS 28.530, Dec. 2021.
- [29] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Trans. Netw.*, vol. 8, no. 5, pp. 556–567, Oct. 2000.
- [30] T. Bonald, L. Massoulié, A. Proutiere, and J. Virtamo, "A queueing analysis of max-min fairness, proportional fairness and balanced fairness," *Queueing Syst.*, vol. 53, no. 1, pp. 65–84, 2006.
- [31] J. Ewing, "Autonomic performance optimization with application to self-Architecting software systems," Ph.D. dissertation, Dept. Comput. Sci., George Mason Univ., Fairfax, VA, USA, Apr. 2015.
- [32] R. L. G. Cavalcante, Q. Liao, and S. Stańczak, "Connections between spectral properties of asymptotic mappings and solutions to wireless network problems," *IEEE Trans. Signal Process.*, vol. 67, no. 10, pp. 2747–2760, May 2019.
- [33] V. Sciancalepore, I. Filippini, V. Mancuso, A. Capone, and A. Banchs, "A multi-traffic inter-cell interference coordination scheme in dense cellular networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 5, pp. 2361–2375, Oct. 2018.
- [34] Z. Xu et al., "Experience-driven networking: A deep reinforcement learning based approach," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2018, pp. 1871–1879.
- [35] H. Song, L. Liu, J. D. Ashdown, and Y. C. Yi, "A deep reinforcement learning framework for spectrum management in dynamic spectrum access," *IEEE Internet Things J.*, vol. 8, no. 14, pp. 11208–11218, Jul. 2021.
- [36] H. Peng and X. S. Shen, "Deep reinforcement learning based resource management for multi-access edge computing in vehicular networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2416–2428, Oct.-Dec. 2020.
- [37] D. Xu, P. Qiao, and Y. Dou, "Aggregation transfer learning for multi-agent reinforcement learning," in *Proc. 2nd Int. Conf. Big Data Artif. Intell. Softw. Eng. (ICBASE)*, 2021, pp. 547–551.
- [38] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Proc. NIPS*, 2016, pp. 2137–2145.
- [39] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," in *Proc. NIPS*, 1999, pp. 1008–1014.
- [40] S. Fujimoto, H. V. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," 2018, *arXiv:1802.09477*.
- [41] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. A. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. ICML*, 2014, pp. 387–395.
- [42] G. Kang, L. Jiang, Y. Yang, and A. G. Hauptmann, "Contrastive adaptation network for unsupervised domain adaptation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4893–4902.
- [43] White paper: Self-organizing network (SON): Introducing the nokia siemens networks SON suite-an efficient, future-proof platform for SON, Nokia Siemens Netw. Espoo, Finland, Rep. C401-00445-WP-200909-1-EN, Oct. 2009.
- [44] J. Meinilä et al. "Wireless world initiative new radio-winner +". Accessed: May 16, 2023. [Online]. Available: https://www.researchgate.net/publication/261467821_CP5-026_WINNER_D53_v10_WINNER_Final_Channel_Models