Network Slicing via Transfer Learning aided Distributed Deep Reinforcement Learning

Tianlun Hu*‡, Qi Liao*, Qiang Liu†, and Georg Carle‡

*Nokia Bell Labs, Stuttgart, Germany
†University of Nebraska Lincoln, United States
‡Technical University of Munich, Germany

Email: *‡tianlun.hu@nokia.com, *qi.liao@nokia-bell-labs.com, †qiang.liu@unl.edu, ‡carle@net.in.tum.de

Abstract—Deep reinforcement learning (DRL) has been increasingly employed to handle the dynamic and complex resource management in network slicing. The deployment of DRL policies in real networks, however, is complicated by heterogeneous cell conditions. In this paper, we propose a novel transfer learning (TL) aided multi-agent deep reinforcement learning (MADRL) approach with inter-agent similarity analysis for inter-cell inter-slice resource partitioning. First, we design a coordinated MADRL method with information sharing to intelligently partition resource to slices and manage inter-cell interference. Second, we propose an integrated TL method to transfer the learned DRL policies among different local agents for accelerating the policy deployment. The method is composed of a new domain and task similarity measurement approach and a new knowledge transfer approach, which resolves the problem of *from whom to transfer* and *how to transfer*. We evaluated the proposed solution with extensive simulations in a system-level simulator and show that our approach outperforms the state-of-the-art solutions in terms of performance, convergence speed and sample efficiency. Moreover, by applying TL, we achieve an additional gain over 27% higher than the coordinated MADRL approach without TL.

I. INTRODUCTION

Network slicing is the key technique in 5G and beyond which enables network operators to support a variety of emerging network services and applications, e.g., autonomous driving, metaverse, and machine learning. The virtual networks (aka. network slices) are dynamically created on the common network infrastructures, e.g., base stations, which are highly customized in different aspects to meet the diverse performance requirement of these applications and services. As the ever-increasing network deployment, e.g., small cells, the traffic of slices and inter-cell interference in radio access networks become more dynamic and complex. Conventional model-based solutions, e.g., linear programming or convex optimization, can hardly handle the ever-complicating resource management problem.

Recent advances in machine learning, especially deep reinforcement learning (DRL) [1], [2], has shown a promising capability to deal with the dynamic and high-dimensional networking problems. The machine learning techniques, as model-free approaches, learn from historical interactions with the network, which require no prior knowledge, e.g., mathematical models. Several works studied to formulate resource management problems as Markov decision process (MDP)s, which are then solved by using DRL to derive a centralized policy with global observations of the network. As the network scale grows, the action and state space of the centralized problem increases exponentially, which challenges the convergence and sample efficiency of DRL. Multi-agent deep reinforcement learning (MADRL) [3], [4] has been exploited to address this issue, which creates and trains multiple cooperative DRL agents, where each DRL agent focuses on an individual site or cell. However, training all individual DRL agents from scratch can still be costly and time-consuming, e.g., expensive queries with real networks, and unstable environments from the perspective of individual DRL agents.

Recently, transfer learning (TL) [5] based methods have been increasingly studied to improve the sample efficiency and model reproducibility in the broad machine learning fields [6]–[8]. The basic idea of TL is to utilize prior knowledge from prelearned tasks to benefit the training process in new tasks. For example, the resource partitioning policy of a cell can be transferred to another cell when they share similar network settings, e.g., bandwidth, transmit power, and traffic pattern. Generally, there are several questions to be answered before using TL methods, i.e., *what to transfer*, *from whom to transfer*, and *how to transfer*. Existing TL methods are mostly focused on supervised machine learning, e.g., computer vision and natural language processing [9], which provide limited insights on applying in DRL tasks [10]–[13]. Therefore, it is imperative to study how TL improves the performance of MADRL in terms of sample efficiency and fine-tune costs, in the inter-cell resource partitioning problem.

In this paper, we proposed a novel TL aided MADRL approach with domain similarity analysis for inter-slice resource partitioning. First, we design a coordinated MADRL method for inter-cell resource partitioning problems in network slicing, where DRL agents share local information with each other to mitigate inter-cell interference. The objective of MADRL is to maximize the satisfaction level of per-slice service requirements in terms of average user throughput and delay in each cell. Second, we design an integrated TL method to transfer the learned DRL policies among different agents for accelerating the policy deployment, where the new method consists of two parts. On the one hand, we propose a feature-based inter-agent similarity analysis approach, which measures the domain and task difference by extracting representative feature distributions in latent space. On the other hand, we propose a new knowledge transfer approach with the combined model (policy) and instance transfer. The main contributions of this paper are summarized as follows:

- We design a coordinated MADRL method for the inter-cell resource partitioning problem in network slicing.
- We design a novel inter-agent similarity analysis approach, based on the features extracted by variational auto-encoder (VAE) to evaluate both domain and task similarity between two reinforcement learning agents.
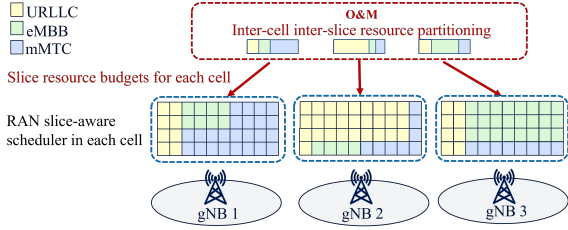- We design a new knowledge transfer approach that combines the model (policy) and instance transfer from

Figure 1: Dynamic multi-cell slicing resource partitioning

the selected source agent to the target agent.

- We evaluate the performance of the proposed solution with extensive simulations in a system-level simulator. The results show that, by applying TL, we achieve an additional gain over 27% higher than the coordinated MADRL approach without TL. Moreover, the performance gain achieved by TL is more significant in the low-data regime.

II. SYSTEM MODEL AND DEFINITIONS

We consider a network consisting of a set of cells $\mathcal{K} := \{1, 2, \ldots, K\}$ and a set of slices $\mathcal{N} := \{1, 2, \ldots, N\}$. Each slice $n \in \mathcal{N}$ has predefined average user throughput and delay requirements, denoted as ϕ_n^* and d_n^* respectively. The network system runs on discrete time slots $t \in \mathbb{N}_0$. As illustrated in Fig. 1, network operation and maintenance (O&M) adapts the inter-slice resource partitioning for all cells to provide per-slice resource budgets to each cell periodically. Then, within each cell, the radio access network (RAN) scheduler uses the provided resource budgets as constraints and performs resource scheduling and physical resource block (PRB) allocation. In this paper, we focus on the inter-cell inter-slice resource partitioning problem in network O&M.

Considering the diverse slice requirements and dynamic network conditions, we model the multi-cell resource partitioning system as a set of K distributed MDPs $\boldsymbol{\mathcal{M}} := \{\mathcal{M}_1, \ldots, \mathcal{M}_K\}$, with $\mathcal{M}_k := \{\mathcal{S}_k, \mathcal{A}_k, P_k(\cdot), r_k(\cdot), \gamma_k\}$ defined for each agent $k \in \mathcal{K}$ (with a slight abuse of notation, hereafter we use k for cell and agent interchangeably). $\mathcal{S}_k$ and $\mathcal{A}_k$ denote the state space and action space respectively. $P_k(\cdot) : \mathcal{S}_k \times \mathcal{A}_k \times \mathcal{S}_k \to [0, 1]$ is the transition probability over $\mathcal{S}_k$ and $\mathcal{A}_k$ for cell k. $r_k : \mathcal{S}_k \times \mathcal{A}_k \to \mathbb{R}$ is defined as the reward function which evaluates the network service of all slices in cell k and γ_k denotes the discount factor for cumulative reward calculation.

At each time step t, agent k collects state $\mathbf{s}_k(t) \in \mathcal{S}_k$ and decides an action $\mathbf{a}_k(t) \in \mathcal{A}_k$ according to policy $\pi_k : \mathcal{S}_k \to \mathcal{A}_k$, which indicates the per-slice resource partitioning ratio $a_{k,n} \in [0, 1]$ for $n \in \mathcal{N}$ while aligning with inter-slice resource constraints. Thus, the local action space $\mathcal{A}_k$ yields

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

For each cell $k \in \mathcal{K}$, our objective is to maximize the minimum service satisfaction level in terms of average user throughput and delay (ϕ_n^*, d_n^*) over all slices. Thus, for each agent k, we define the local reward function based on the observed per-slice average user throughput $\phi_{k,n}(t)$ and delay $d_{k,n}(t)$ at time t as

$$r_k(t) := \min_{n \in \mathcal{N}} \min \left\{ \frac{\phi_{k,n}(t)}{\phi_{k,n}^*}, \frac{d_{k,n}^*}{d_{k,n}(t)}, 1 \right\}. \quad (2)$$

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 (2) is **inversely proportional** to the actual delay, namely, if the delay is longer than required this term is lower than 1.

III. PROBLEM FORMULATION

The Reinforcement Learning Problem: The problem is to find a policy $\pi_k : \mathcal{S}_k \to \mathcal{A}_k$ for each $k \in \mathcal{K}$ that predicts optimal inter-slice resource partitioning $\mathbf{a}_k(t) \in \mathcal{A}_k$ base on the local state $\mathbf{s}_k(t) \in \mathcal{S}_k$ dynamically, to maximize the expectation of the cumulative discounted reward $r_k(t)$ defined in (2), in a finite time horizon T. The problem is given by:

$$\max_{\pi_k; \mathbf{a}_k(t) \in \mathcal{A}_k} \mathbb{E}_{\pi_k} \left[\sum_{t=0}^{T} \gamma_k^t r_k \big(\mathbf{s}_k(t), \mathbf{a}_k(t) \big) \right], \ \forall k \in \mathcal{K}, \quad (3)$$

where $\mathcal{A}_k$ is defined in (1).

In our previous work [14], we proposed a coordinated multi-agent DRL approach to transform an MADRL problem to the distributed DRL problem similar to (3), where the extracted information from neighboring cells is included into the state observation to better capture the inter-agent dependency. However, training all local agents in parallel from scratch can be costly and time-consuming. Moreover, the trained models are sensitive to environment changes and the retraining cost can be high.

Thus, in this paper, we raise the following new questions: *Can we reuse the knowledge in a pretrained model? When is the knowledge transferable? And, most importantly, how to transfer the gained knowledge from one agent to another?*

The Transfer Learning Problem: To tackle the transfer learning problem, let us first introduce two definitions *domain* and *task* in the context of reinforcement learning.

A *domain* $\mathcal{D} := \{\mathcal{S}, P(\mathbf{s})\}$ consists of a state feature space $\mathcal{S}$ and its probability distribution $P(\mathbf{s})$, for $\mathbf{s} \in \mathcal{S}$. A *task* $\mathcal{T} := \{\mathcal{A}, \pi(\cdot)\}$ consists of the action space $\mathcal{A}$ and a policy function $\pi : \mathcal{S} \to \mathcal{A}$.

Thus, our inter-agent transfer learning problem is to find the optimal source agent among a set of pretrained agents, and transfer its knowledge (pretrained model and collected instances) to the target agent, such that problem (3) can be solved in the target agent with fast convergence and limited amount of samples. In particular, the problem is defined in Problem 1.

Problem 1. *Given a set of pretrained source agents $\overline{\mathcal{K}} \subset \mathcal{K}$ with source domains $\boldsymbol{\mathcal{D}}^{(S)} := \left\{ \mathcal{D}_i^{(S)} : i \in \overline{\mathcal{K}} \right\}$ and pretrained tasks $\boldsymbol{\mathcal{T}}^{(S)} := \left\{ \mathcal{T}_i^{(S)} : i \in \overline{\mathcal{K}} \right\}$, also given any target agent $k \notin \overline{\mathcal{K}}$ with target domain $\mathcal{D}_k^{(T)}$ and untrained task $\mathcal{T}_k^{(T)}$, find the optimal source agent $i_k^* \in \overline{\mathcal{K}}$ for target agent k to transfer knowledge such that*

$$i_k^* := \underset{\substack{\pi_k | \pi_k^{(0)} = \Lambda\left(\pi_i^{(S)}\right); \\ i \in \overline{\mathcal{K}}}}{\arg \max} \mathbb{E}_{\pi_k} \left[\sum_{t=0}^{T} \gamma_k^t r_k \big(\mathbf{s}_k(t), \mathbf{a}_k(t) \big) \right] \quad (4)$$

$$s.t. \ (\mathbf{s}_k, \mathbf{a}_k) \in \Gamma \left(\mathcal{D}_i^{(S)}, \mathcal{D}_k^{(T)}, \mathcal{A}_i^{(S)}, \mathcal{A}_k^{(T)} \right),$$

where $\Lambda\left(\pi_i^{(S)}\right)$ is the policy transfer *strategy which maps a pretrained source policy $\pi_i^{(S)}$ to the initial target policy $\pi_k^{(0)}$,*

while $\Gamma\left(\mathcal{D}_i^{(S)}, \mathcal{D}_k^{(T)}, \mathcal{A}_i^{(S)}, \mathcal{A}_k^{(T)}\right)$ is the instance transfer strategy which selects the instances from the source agent, combines them with the experienced instances from the target agent, and saves them in the replay buffer for model training or fine-tuning in the target agent. More details about the transfer learning strategies will be given in Section IV-C.

IV. PROPOSED SOLUTIONS

In this section, we first present a distributed MADRL approach to solve the slicing resource partitioning problem in (3). Then, to solve problem (4) to find the optimal source agent, we propose a novel approach to inter-agent similarity analysis based on the extracted features using VAE. Finally, for inter-agent transfer learning, we introduce transfer learning strategy which combines the model (policy) transfer and instance transfer.

A. Coordinated MADRL Approach

As stated in (3), the distributed DRL approach allows each agent to learn a local policy and makes its own decision on inter-slice resource partitioning based on local observation. Compared with the centralized DRL approaches, distributed approaches reduce the state and action spaces and significantly accelerate the training progress. However, local observation alone cannot capture the inter-cell dependencies and provide sufficient information to achieve the globally optimal solution. Thus, we proposed in [14] a distributed DRL approach with inter-agent coordination which keeps the low model complexity while including the extracted information from neighboring cells to capture the inter-cell interference. We briefly summarize the coordinated distributed DRL approach below, because we would like to focus on the main contribution, namely, the inter-agent transfer learning, in this paper. For more details, readers are referred to our previous work [14].

Each local agent k observes a local state $\mathbf{s}_k'$, which contains the following network measurements:

- Per-slice average user throughput $\{\phi_{k,n} : n \in \mathcal{N}\}$;
- Per-slice network load $\{l_{k,n} : n \in \mathcal{N}\}$;
- Per-slice number of users $\{u_{k,n} : n \in \mathcal{N}\}$.

Thus, with the above-defined three slice-specific features, the local state $\mathbf{s}_k'$ has the dimension of $3N$. Additionally, to better capture the inter-cell dependencies and estimate the global network performance, we introduce an inter-agent coordination mechanism through network information sharing among agents. Let each agent k broadcast a message $\mathbf{m}_k$ to its neighboring group of agents, denoted by $\mathcal{K}_k$, which means, each agent k receives a collection of messages $\overline{\mathbf{m}}_k := [\mathbf{m}_i : i \in \mathcal{K}_k] \in \mathbb{R}^{Z^{(m)}}$. Instead of using all received messages in $\overline{\mathbf{m}}_k$, we propose to to extract useful information $\mathbf{c}_k \in \mathbb{R}^{Z^{(c)}}$ to remain the low model complexity. We aim to find an feature extractor $g : \mathbb{R}^{Z^{(m)}} \to \mathbb{R}^{Z^{(c)}} : \overline{\mathbf{m}}_k \to \mathbf{c}_k$, such that $Z^{(c)} \ll Z^{(m)}$. Then, we include the extracted features from the shared messages into the local state: $\mathbf{s}_k := [\mathbf{s}_k', \mathbf{c}_k]$.

Knowing that the inter-agent dependencies are mainly caused by inter-cell interference based on cell load coupling [15], we propose to let each cell k share its per-slice load $l_{k,n}, \forall n \in \mathcal{N}$ to its neighboring cell. Then, we compute the extracted information $\mathbf{c}_k$ as the average per-slice neighboring load. Namely, we define a deterministic feature extractor, given by:
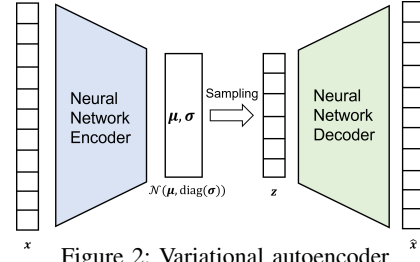

Figure 2: Variational autoencoder

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

$$\text{with } \mathbf{c}_k(t) := \left[\frac{1}{|\mathcal{K}_k|} \sum_{i \in \mathcal{K}_k} l_{i,n}(t) : n \in \mathcal{N}\right]. \quad (5)$$

With the extended local state including the inter-agent shared information, we can use classical DRL approaches, e.g., the actor-critic algorithms such as Twin Delayed Deep Deterministic policy gradient (TD3) [16] to solve (3).

B. Integrated TL with Similarity Analysis

The distributed DRL approach introduced in Section IV-A allows us to derive a set of pretrained local agents. Still, given a target cell k, e.g., a newly deployed cell, or an existing cell but with changed environment, more questions need to be answered: Can we transfer the prelearned knowledge from at least one of the pretrained agents? Which source cell provides the most transferable information? How to transfer the knowledge?

To solve the transfer learning problem in (4), we develop a distance measure $\mathfrak{D}_{i,k}$ to quantify the inter-agent similarity between a source agent i and a target agent k. We aim to transfer the knowledge from the source agent with the highest similarity (reflected by the lowest distance measure).

The ideal approach to analyze the domain and task similarity between two agents is to obtain their probability distributions of the state $P(\mathbf{s})$ and derive the conditional probability distribution $P(\mathbf{a}|\mathbf{s})$. However, the major challenge here lies in the limited samples in the target agent. Considering that the target agent is a newly deployed agent, there is no information available about its policy $P(\mathbf{a}|\mathbf{s})$, and $P(\mathbf{s})$ is very biased, because all samples are collected under the default configurations (i.e., constant actions).

Thus, we need to design a distance measure constrained by very limited and bias samples in the target agent, without any information about its policy $P(\mathbf{a}|\mathbf{s})$. Our idea is to derive and compare the **joint state and reward distribution** under the same default action $\mathbf{a}'$, $P(\mathbf{s}, r|\mathbf{a} = \mathbf{a}')$, in both source and target agent. The rationale behind this is that, when applying the actor-critic-based DRL architecture, the critic function estimates the Q value $Q_\pi(\mathbf{a}, \mathbf{s})$ based on action and state. Hence, the conditional probability $P(r|\mathbf{s}, \mathbf{a})$ should provide useful information of the policy. With $\mathbf{a} = \mathbf{a}'$, we can consider to estimate $P(r|\mathbf{s}, \mathbf{a} = \mathbf{a}')$. To efficiently capture the information for both domain similarity (based on $P(\mathbf{s}|\mathbf{a} = \mathbf{a}')$) and task/policy similarity (based on $P(r|\mathbf{s}, \mathbf{a} = \mathbf{a}')$), we propose to estimate the joint probability $P(\mathbf{s}, r|\mathbf{a} = \mathbf{a}') = P(r|\mathbf{s}, \mathbf{a} = \mathbf{a}')P(\mathbf{s}|\mathbf{a} = \mathbf{a}')$.

Sample collection: To estimate the distance between $P(\mathbf{s}, r|\mathbf{a} = \mathbf{a}')$ of both the source and target agents, we use all available samples from the target agent k under the default action $\mathbf{a}'$, $\mathcal{X}_k = \{(\mathbf{s}_k(n), r_k(n))_{\mathbf{a}_k(n) = \mathbf{a}'} : n = 1, \ldots, N_k\}$, and select a subset of the samples from the source agent i with

the same default action $\mathcal{X}_i = \{(\mathbf{s}_i(n), r_i(n))_{\mathbf{a}_i(n) = \mathbf{a}'} : n = 1, \ldots, N_i\}$. Note that in this subsection we slightly abuse the notation by using n as index of samples, and N_k as number of samples with default action collected from agent k.

Feature extraction with VAE: To extract the representative features from the high-dimension vector $[\mathbf{s}, r]$, we propose to apply VAE [17] to map the samples into a low dimensional latent space. As Fig. 2 illustrates, for each sample $\mathbf{x} := [\mathbf{s}, r] \in \mathcal{X}$, the encoder of VAE estimates an approximated distribution $P(\mathbf{z})$ in latent space $\mathcal{Z}$ as a multi-variate Gaussian distribution with $\mathcal{N}(\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}))$, where diag denotes the diagonal matrix. The decoder samples a latent variable $\mathbf{z} \in \mathcal{Z}$ from the approximated distribution $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}))$ and outputs a reconstructed sample $\hat{\mathbf{x}}$ by training on the following loss function:

$$\mathcal{L} := \|\mathbf{x} - \hat{\mathbf{x}}\|^2 + \alpha \cdot D_{KL}\left(\mathcal{N}(\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma})) \| \mathcal{N}(\mathbf{0}, \mathrm{diag}(\mathbf{1}))\right), \quad (6)$$

where α is the weight factor and D_{KL} denotes the Kullback-Leibler (KL) divergence.

Inter-agent similarity analysis: Since VAE does not directly provide the probability distribution function $P(\mathbf{x})$, we propose to utilize the extracted features in the latent space to evaluate the inter-agent similarity. Considering the limited amount of samples (only those under default action), we propose to train a general VAE model based on the samples from all candidate source agents and the target agent, e.g., $\mathcal{X} = \bigcup_{j \in \overline{\mathcal{K}} \cup \{k\}} \mathcal{X}_j$. The idea is to extract the latent features from samples from all relevant agents with a general encoder and to distinguish the agents within a common latent space.

Thus, for each sample $\mathbf{x}_n \in \mathcal{X}$, we can derive its extracted features, i.e., the posterior distribution $P(\mathbf{z}_n | \mathbf{x}_n) = \mathcal{N}(\boldsymbol{\mu}_n, \mathrm{diag}(\boldsymbol{\sigma}_n))$. We denote the extracted latent space for agent k by $\mathcal{Z}_k$. Next, we can measure the inter-agent distance between an arbitrary source agent i and target agent k by calculating the KL divergence based on the extracted latent variables from their collected samples:

$$\mathfrak{D}_{i,k} := \frac{1}{N_i N_k} \cdot \sum_{\substack{(\boldsymbol{\mu}_n, \boldsymbol{\sigma}_n) \in \mathcal{Z}_i; \\ (\boldsymbol{\mu}_m, \boldsymbol{\sigma}_m) \in \mathcal{Z}_k}} D_{KL}\left(\mathcal{N}(\boldsymbol{\mu}_n, \mathrm{diag}(\boldsymbol{\sigma}_n)) \| \mathcal{N}(\boldsymbol{\mu}_m, \mathrm{diag}(\boldsymbol{\sigma}_m))\right).$$

$$(7)$$

This requires to compute the KL divergence of every pair of samples (n, m) for $n \in \mathcal{X}_i$ and $m \in \mathcal{X}_k$, which could be computing intensive.

Note that they are both Gaussian distributions, we can efficiently compute them with closed-form expression (as will be shown later in (8)). Besides, from our experiment, we observed that $\boldsymbol{\sigma}_n \to \mathbf{0}$ for nearly all the collected samples $\mathbf{x}_n \in \mathcal{X}$, i.e., their variances are extremely small (to the level below $10e - 5$ from our observation). Thus, for our problem, we can use a trick to evaluate the distance measure more efficiently based on the following lemma.

Lemma 1. *Given two multi-variate Gaussian distributions $p = \mathcal{N}(\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)$ and $q = \mathcal{N}(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$, where $\boldsymbol{\mu}_n, \boldsymbol{\mu}_m \in \mathbb{R}^L$, $\boldsymbol{\Sigma}_n = \boldsymbol{\Sigma}_m = \mathrm{diag}(\boldsymbol{\sigma}) \in \mathbb{R}^{L \times L}$ and every entry of $\boldsymbol{\sigma}$ is equal to a small positive constant $\sigma \ll 1$, the KL divergence $D_{KL}(p\|q)$ is proportional to $\sum_{l=1}^{L}(\mu_{n,l} - \mu_{m,l})^2$.*

Proof. It is easy to derive that

$$\begin{aligned} D_{KL}(p\|q) =& \frac{1}{2}\Big[\log \frac{|\boldsymbol{\Sigma_n}|}{|\boldsymbol{\Sigma_m}|} - L + \\ & (\boldsymbol{\mu}_n - \boldsymbol{\mu}_m)^T \boldsymbol{\Sigma_m}^{-1}(\boldsymbol{\mu}_n - \boldsymbol{\mu}_m) + \quad (8) \\ & \mathrm{Tr}\left\{\boldsymbol{\Sigma_m}^{-1}\boldsymbol{\Sigma_n}\right\}\Big]. \end{aligned}$$

Because $\boldsymbol{\Sigma}_n = \boldsymbol{\Sigma}_m = \mathrm{diag}([\sigma^2, ..., \sigma^2])$, we have the first term in (8) equals to 0, and the last term equals to L. Thus, we obtain

$$D_{KL}(p\|q) = \frac{1}{2\sigma^2}\sum_{l=1}^{L}(\mu_{n,l} - \mu_{m,l})^2. \quad (9)$$
$\square$

With Lemma 1, we can measure the distance between two agents more efficiently, based on the extracted $\boldsymbol{\mu}_n$ and $\boldsymbol{\mu}_m$ in the source and target latent spaces. Thus, to solve Problem (III.1), we propose to choose the source agent:

$$i_k^* := \arg\min_{i \in \overline{K}} \mathfrak{D}_{i,k}, \quad (10)$$

where $\mathfrak{D}_{i,k}$ is computed based on (7) and (9).

C. Integrated Transfer Learning Approach

In general, the prelearned knowledge can be transferred from a source agent i to the target agent k with various policy transfer strategies $\Lambda(\cdot)$ and instance transfer strategy $\Gamma(\cdot)$:

- **Model transfer**: The policy transfer strategy $\Lambda(\cdot)$ simply initializes the target agent's policy $\pi_k^{(0)}$ by loading the parameters (e.g., weights of the pretrained neural networks) of the pretrained policy $\pi_i^{(S)}$ from the source agent i.
- **Feature transfer**: The policy transfer strategy $\Lambda(\cdot)$ keeps partial information extracted from the source agent's pretrained policy $\pi_i^{(S)}$. In particular, the target agent loads partial of the layers (usually the lower layers) of the pretrained neural networks of $\pi_i^{(S)}$, while leaving the rest of them to be randomly initialized. Then, during training, the loaded layers are frozen and only the randomly initialized layers are fine-tuned with the instances newly collected by the target agent.
- **Instance transfer**: The instance transfer strategy $\Gamma(\cdot)$ transfers the collected instances from the source agent i to the target agent k and saves them in the target agent's replay buffer. Then, the target agent trains a policy from scratch with randomly initialized parameters and mixed instances collected from both source and target agents.

The above-mentioned knowledge from the source domain and source task can be transferred separately or in a combined manner. In this paper, we propose the **integrated transfer method** with both model and instance transfer. Specifically, the target agent k initializes its local policy $\pi_k^{(0)}$ by loading the pretrained policy of the source agent $\pi_i^{(S)}$ and fine-tunes the policy by sampling from the replay buffer containing both types of instances: the instances transferred from the source agent and those locally experienced. Here, we skip the feature transfer because it practically performs well only when the similarity between the source domain/task and target domain/task is very high. Although this assumption may hold for some regression and classification tasks, we empirically find that it fails in this context of MADRL.

V. Performance Evaluation

In this section, we evaluate the performance of the proposed solution within a system-level simulator [18]. The simulator
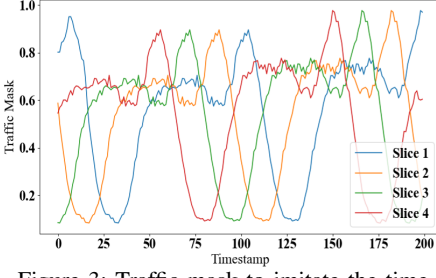
Figure 3: Traffic mask to imitate the time varying network traffic
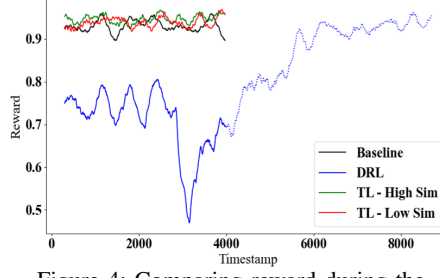

Figure 4: Comparing reward during the training process
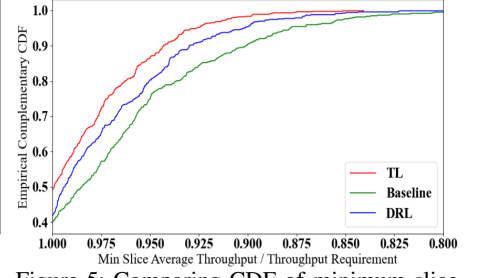

Figure 5: Comparing CDF of minimum slice throughput satisfaction

achieves a great accuracy in imitating the real network systems with configurable user mobility, network slicing traffic and topology. In addition, we introduce a traffic-aware baseline which allocates resource proportionally to the data traffic demand per slice. Note that the baseline assumes perfect information about per-cell per-slice traffic demands, which provides already very good results.

1) Network settings: We build a radio access network with 4 three-sector sites (i.e., $K = 12$ cells). All cells are deployed using LTE radio technology with 2.6 GHz under a realistic radio propagation model Winner+ [19]. Each cell has $N = 4$ slices with diverse per-slice requirements in terms of average user throughput and delay. In the cells with label $1, 2, 3, 7, 8, 9$, we define per-slice average throughput requirements of $\phi_1^* = 4$ MBit/s, $\phi_2^* = 3$ MBit/s, $\phi_3^* = 2$ MBit/s, and $\phi_4^* = 1$ MBit/s respectively, and per-slice delay requirements of $d_1^* = 3$ ms, $d_2^* = 2$ ms, $d_3^* = d_4^* = 1$ ms. In the cells with label $4, 5, 6, 10, 11, 12$, we define per-slice throughput requirements as $\phi_1^* = 2.5$ MBit/s, $\phi_2^* = 2$ MBit/s, $\phi_3^* = 1.5$ MBit/s, and $\phi_4^* = 1$ MBit/s, and delay requirements of $d_n^* = 1$ ms, $\forall n \in \mathcal{N}$. All cells have the same radio bandwidth of 20 MHz.

We define four groups of user equipment (UE) associated to four slices in each cell respectively, each UE group has the maximum size of 32 and moves randomly among the defined network scenario. To mimic dynamic behavior of real user traffic, we apply a varying traffic mask $\tau_n(t) \in [0, 1]$ to each slice to scale the total number of UEs in each cell, Fig. 3 shows the traffic mask in first 200 steps.

2) DRL training configuration: For MADRL training, we implemented TD3 algorithm at each local agent using multi-layer perception (MLP) architecture for actor-critic networks. In each TD3 model, both actor and critic neural works consist of two layers with the number of neurons as $(48, 24)$ and $(64, 24)$ respectively. The learning rates of actor and critic are 0.0005 and 0.001 accordingly with Adam optimizer and training batch size of 32. We set the discount factor as $\gamma = 0.1$, since the current action has stronger impact on instant network performance than future observation. As for the training, for distributed DRL agents we applied 3000 steps for exploration, 5500 steps for training, and final 250 steps for evaluation. For TL training process, we apply the same model setups as DRL approaches, while only setting 4000 steps for training and 250 for evaluation since knowledge transfer save the time for exploration.

3) Comparing DRL to TL aided approach: In Fig. 4 we compare the evolution of reward during the training processes among the baseline, DRL approach (proposed in Section IV-A), and TL approaches when transferred from source agent with low and high similarity (proposed in Section IV-B and

IV-C), respectively. For DRL, we present the first 4000 step, i.e., the same training time as TL approaches with solid line and the rest training curve with dashed line.

As shown in Fig. 4, the distributed DRL approach learns to achieve similar reward as baseline after a lengthy exploration phase, while both TL approaches start with much higher start compared to DRL. After a short fine-tuning period, the TL approaches outperform the baseline with higher robustness, especially during the period with higher traffic demands and strong inter-cell interference where baseline has sharp performance degradation. Besides, in comparison between the TL from agents with different similarity measure, we observe that with higher similarity, TL provides higher start at the early stage of training, while both of them converge to similar performance after the training converges.

For performance evaluation, we compare the statistical results on minimum per slice throughput satisfaction level and maximum per slice delay, respectively, among all cells among the methods baseline, distributed DRL and the proposed TL approach after convergence. Fig. 5 illustrated the empirical complementary cumulative distribution function (CDF) which equals $1 - F_X(x)$ where $F_X(x)$ is the CDF of minimum per slice throughput satisfaction level. We observe that the TL approach provides the best performance comparing to others by achieving only about 12% fail to satisfy 0.95 of the requirement, while converged DRL and baseline conclude 19% and 25% failure rate respectively. By average satisfaction level, the TL approach conclude 0.92 while DRL and baseline only provide 0.90 and 0.87. Similar observation can be made from Fig. 6, which illustrates the CDF of maximum slice delay in ms. The TL approach provides 1.5 ms maximum average per-slice delay, while DRL achieves 1.7 ms and baseline achieves 1.8 ms.

4) Inter-agent similarity analysis: We implemented the similarity analysis method introduced in Section IV-B with a VAE model in MLP architecture, both networks of encoder and decoder consist of 3 layers with number of neurons as $(64, 24, 4)$ and $(4, 24, 64)$ respectively. To achieve a good trade-off between low dimensional latency space and accurate reconstruction with VAE, we map the original sample $x \in \mathbb{R}^{17}$ to the latent variable $\mathbf{z} \in \mathbb{R}^4$.

Fig. 7 illustrates the results of inter-agent similarity analysis as a metric of distance measure proposed in (7). It shows that our proposed method can distinguish cells with different per-slice service quality requirements and gather the cells with similar joint state-reward distribution.

5) Dependence of TL performance on distance measure: In Fig. 8 we compare the benefits of TL in training process by transferring knowledge from source agents with different average inter-agent distance measures. The TL gains are
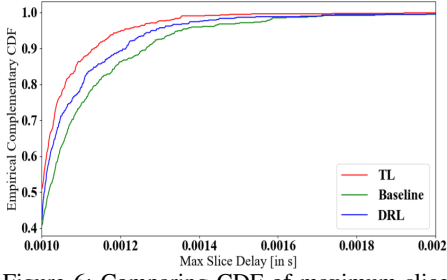
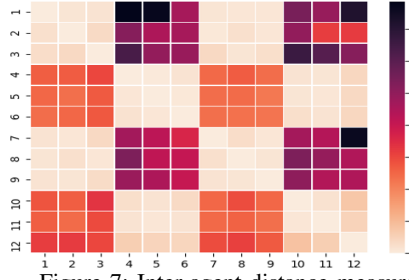Figure 6: Comparing CDF of maximum slice delay
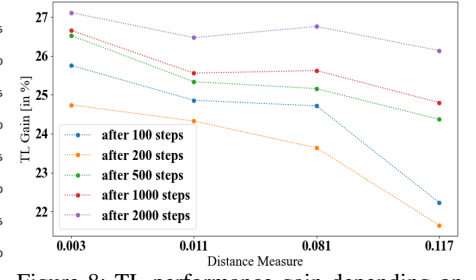

Figure 7: Inter-agent distance measure

Figure 8: TL performance gain depending on distance measure

derived by comparing the reward to DRL approach at the same training steps. The results show that before 200 steps of TL training, the TL approaches with the lowest distance measure provides about 3% higher gain than the one with the largest distance. As the training process continues, the gains in all TL approaches increase with local fine-tuning and the difference between transferring from highly similar and less similar agents is getting smaller. However, TL from the most similar agent proyvides higher gains for all training steps.

6) Key Takeaways: : We summarized the takeaways from numerical results as follows:

- All distributed DRL-based approaches achieve better per-slice network service than the traffic-aware baseline after convergence. However, the TL schemes outperform the conventional DRL approach in terms of convergence rate, initial and converged performance.
- Our propose VAE-based similarity measure well quantifies the distance between agents and can be used to suggest a mapping from the defined distance measure to the transfer learning performance gain.
- The difference between the gains achieved by TL from the highly similar and the less similar agents is more significant when the number of training steps is low (i.e., with limited online training samples). Although the advantage of transferring from a highly similar agent over a less similar agent decreases when the number of online training steps increases, a slight performance gain is always achieved by transferring knowledge from the most similar source agent.

VI. CONCLUSION

In this paper, we formulated the dynamic inter-slice resource partitioning problem to optimize the network requirement satisfaction level of all slices in each cell. To tackle the inter-cell interference, we proposed a coordinated MADRL method with the coordination scheme of information sharing. We proposed a novel integrated TL method to transfer the learned DRL policies among different local agents for accelerating the policy deployment. The method is accomplished by a new inter-agent similarity measurement approach and a new knowledge transfer approach. We evaluated the proposed solution with extensive simulations in a system-level simulator, where the results show our approach outperforms conventional DRL solutions.

ACKNOWLEDGMENT

REFERENCES

[1] Y. Liu, J. Ding, and X. Liu, "A constrained reinforcement learning based approach for network slicing," in *2020 IEEE 28th International Conference on Network Protocols (ICNP)*, 2020, pp. 1–6.

[2] Q. Liu, T. Han, N. Zhang, and Y. Wang, "DeepSlicing: Deep reinforcement learning assisted resource allocation for network slicing," in *IEEE GLOBECOM*, 2020, pp. 1–6.

[3] 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 Transactions on Wireless Communications*, vol. 18, pp. 5141–5152, 2019.

[4] Y. Shao, R. Li, Z. Zhao, and H. Zhang, "Graph attention network-based drl for network slicing management in dense cellular networks," *IEEE WCNC*, pp. 1–6, 2021.

[5] 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, 2009.

[6] C. T. Nguyen *et al.*, "Transfer learning for future wireless networks: A comprehensive survey," *arXiv preprint arXiv:2102.07572*, 2021.

[7] M. Wang, Y. Lin, Q. Tian, and G. Si, "Transfer learning promotes 6g wireless communications: recent advances and future challenges," *IEEE Transactions on Reliability*, 2021.

[8] C. Parera, Q. Liao *et al.*, "Transfer learning for tilt-dependent radio map prediction," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 2, pp. 829–843, 2020.

[9] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

[10] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *J. Mach. Learn. Res.*, vol. 10, pp. 1633–1685, 2009.

[11] Z. Zhu, K. Lin, and J. Zhou, "Transfer learning in deep reinforcement learning: A survey," *CoRR*, vol. abs/2009.07888, 2020. [Online]. Available: https://arxiv.org/abs/2009.07888

[12] A. M. Nagib, H. Abou-zeid, and H. S. Hassanein, "Transfer learning-based accelerated deep reinforcement learning for 5G RAN slicing," *IEEE 46th LCN*, pp. 249–256, 2021.

[13] T. Mai, H. Yao *et al.*, "Transfer reinforcement learning aided distributed network slicing resource optimization in industrial IoT," *IEEE Transactions on Industrial Informatics*, 2021.

[14] T. Hu, Q. Liao, Q. Liu, D. Wellington, and G. Carle, "Inter-cell slicing resource partitioning via coordinated multi-agent deep reinforcement learning," in *IEEE ICC*, 2022.

[15] R. L. G. Cavalcante, Q. Liao, and S. Stańczak, "Connections between spectral properties of asymptotic mappings and solutions to wireless network problems," *IEEE Transactions on Signal Processing*, vol. 67, pp. 2747–2760, 2019.

[16] S. Fujimoto *et al.*, "Addressing function approximation error in Actor-Critic methods," *ArXiv*, vol. abs/1802.09477, 2018.

[17] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *NIPS*, 2015.

[18] Nokia Siemens Networks, *White paper: Self-organizing network (SON): Introducing the Nokia Siemens networks SON suite-an efficient, future-proof platform for SON.* Technical report, October, 2009.

[19] J. Meinilä, P. Kyösti, L. Hentilä, T. Jämsä, E. Suikkanen, E. Kunnari, and M. Narandžić, *Wireless World Initiative New Radio - Winner+.* Technical report, 2010.