Optimizing Resource Scheduling for Multi-Scenario Mixed Service Groups under Edge Cloud-Native Environments Using Simulation Learning

Wei Xiong[1], Xinying Wang[1], Franz Wotawa[2], Qiaozhi Hua[1]*

[1] School of Computer Engineering, HuBei University of Arts and Science, China
[2] Institute for Software Technology, Graz University of Technology, Austria
xwei9093@126.com, wxy200888@126.com, wotawa@ist.tugraz.at, alex2441@163.com

Abstract

The evolution of cloud and edge computing technologies has brought about resource management challenges. Traditional resource scheduling strategies fall short in dynamic cloud-edge environments, one of the challenges is identifying system state changes in multi-scenario edge cloud-native environments. The dynamic orchestration and deployment of container resources are crucial. To address this issue, we introduce a virtual environment, which generates interactions of multi-scenario mixed service groups. Furthermore, we proposed a multi-agent adversarial imitation learning approach, which is trained in the virtual environment. Experiments reveal that our approach, which is fully trained in the virtual mixed-service environment, results in no physical sampling costs and significantly outperforms traditional supervised approaches.

Keywords: Edge cloud-native, Resource scheduling, Imitation learning

1 Introduction

As cloud-native technology matures, it's being explored by academia and business communities for practical implementation [1]. The integrated development of AI, IoT, and edge computing is leading to a growing variety, scale, and complexity of business operations in edge computing scenarios [2-4], and building a new generation of edge computing platforms using cloud-native technology is becoming an industry focus [5]. Therefore, studying how to integrate cloud-native technology and edge computing to assist developers in managing large-scale applications on wide-ranging cloud-edge resources is of significant importance [6].

Edge cloud-native applications have a broad spectrum of use cases, including live video [7], cloud gaming [8], logistics and transportation [9], intelligent manufacturing [10], and urban brain [11], etc. These applications can be categorized into mobile broadband services [12], large-scale IoT services with fixed sensors [13], and mission-critical IoT services like the Internet of Vehicles, based on factors such as mobility, billing, security, policy control, latency, and reliability [14-16]. However, deploying ultra-large-scale edge cloud-native services can be challenging due to issues like decentralized computing power, heterogeneous resources, and weak network connectivity. The goal of edge cloud-native technology is to consolidate scattered computing power into a larger resource pool and optimize resource scheduling for maximum energy efficiency, effectively balancing peak and valley filling. When users deploy mixed services (a combination of online and offline services) and request necessary computing resources (such as CPU, memory, disk) from the management node, the scheduler selects physical machines that meet the specifications to deploy the containers. Since there are often multiple fitting physical machines, each with varying resource capacities, different allocation approaches can lead to different allocation rates. The crucial role of the scheduler is to select the most suitable physical machine out of numerous possibilities, adhering to a specific strategy [17-18]. Implementing mixed services in multiple scenarios within a native edge cloud environment is a challenge due to the varying movement characteristics and communication capabilities among different service communities. These discrepancies can delay the formation of a stable cluster. Issues arise when the data flow and control flow are separated into different mixed service groups, obstructing the collaboration among multi-scenario mixed service groups. In the current edge cloud environment, resources for data and control planes are scheduled separately within distinct service groups. This situation prompts several questions. How can we maintain stability in multi-scenario mixed-service groups given their rapidly changing topologies and complex interaction? Additionally, how can we effectively manage service group resources?

Reinforcement learning (RL) can effectively manage the scheduling of mixed-service resources. However, its direct application can be challenging due to the need for numerous interactions with the environment, which can be expensive. As a solution, simulators are often used for RL training. A prime example is Google's data center cooling system [19], where a neural network estimates system dynamics. The policy is subsequently trained in a virtual environment using advanced reinforcement learning algorithms.

Applying reinforcement learning to real-world tasks can be challenging. In physical environments, conducting a large number of experiments as required by current approaches is often impractical. To enhance service resource scheduling in edge-native multi-application scenarios, we propose an

*Corresponding Author: Xinying Wang; E-mail: wxy200888@126.com
DOI: https://doi.org/10.70003/160792642024122507011

approach involving offline training using reinforcement learning algorithms in a virtual environment. The goal is to maximize long-term rewards in a simulator. The resulting policy is expected to perform well in the actual environment or serve as a starting point for online tuning.

Simulating the behavior of multi-scenario mixed service groups in a dynamic environment is more challenging than approximating the dynamics of a data center. We generate the behavior data for the mixed service groups from some policies. Current imitation learning approaches can form policies from data [20-21]. Behavior cloning (BC) approaches [22] primarily learn policies from state-behavior data using supervised approaches. However, BC requires an i.i.d. assumption on demonstration data, which isn't met in RL tasks. On the other hand, Inverse Reinforcement Learning (IRL) approaches [23] learn a reward function from data and then train a policy based on this reward function. Unlike BC, IRL relaxes the i.i.d. assumption on the data but still presumes a static environment. If the environment changes, the learned policy may fail. These shortcomings make these approaches less practical for building virtual environments.

Reinforcement learning approaches come with several challenges: 1) they are time-consuming and require substantial engineering; 2) if the model is not trained sufficiently before deployment, it may perform poorly on real-time data, compromising the reinforcement learning phase; 3) large amounts of data are necessary to ensure the model's robustness. To mitigate these challenges, some researchers recommend using a virtual environment for training and assessing the system [24-25]. Integrating a virtual environment with reinforcement learning enables intelligent scheduling based on runtime resources. It simplifies the unified resource scheduler, enhances runtime stability, and reduces resource costs.

However, there is a compound error issue in the environment model of offline reinforcement learning. In 2002, Kearns and Singh demonstrated [26] that the environment model obtained through supervised learning is prone to a significant compound error, with an error coefficient of T^2 after T steps of action. For example, if there are 100 steps, the final single-step error will be magnified by a factor of 10,000. This quadratic compound error has had a substantial impact on the field, leading to the abandonment of high-precision environment models and the avoidance of relying on them. As an alternative, the model-based policy optimization (MBPO) approach, proposed by the University of California, Berkeley in 2021 [27], recognizes the difficulty in reducing compound error and instead focuses on minimizing it within a few steps in the environment model. However, this approach faces challenges in fully evaluating the policy if the complete action decision trajectory cannot be executed in the environment model, greatly reducing its applicability in real-world tasks.

In 2020, Tian Xu et al. proposed a novel proof approach for cumulative errors [28], which identified that compound error consists of two components. The first one is the conversion from single-step error to state distribution error, while the second one is the conversion from state distribution error to overall policy return error. By bypassing the first part and utilizing distribution matching targets instead of single-step prediction targets, the coefficient of compound error can be reduced from T^2 to T, reaching the theoretical lower bound. This breakthrough effectively resolves the theoretical problem of excessive compound error. Distribution matching targets involve matching the overall distribution of data generated through policy interactions in the environment model with historical data. The approaches for constructing an environment model based on distribution matching targets can effectively leverage reinforcement learning techniques to train efficient policy models in the environment model.

In this study, we build a virtual environment through the interaction between multi-agent policy and the environment. This is particularly relevant for multi-scenario mixed service groups under edge cloud-native environments, where the movement characteristics and communication capabilities of multi-scenarios vary greatly. Their distribution is complex and extensive, but database sampling cannot generate additional data, leading to a low degree of generalization in the final model. To address this, we propose the GAN (Generative Adversarial Network) [29] Simulating Multi-scenario Mixed Service groups (GAN-SMMSG) approach to create virtual agents, as traditional approaches like GMM [30] and GAN struggle with such high-dimensional data. To generate interactions, a fundamental aspect of virtual environments, we introduce a Multi-agent Adversarial imitation Learning of Multi-Scenario Mixed Service Groups (MAIL-MMSG). This approach follows the idea of GAIL [31] and learns the policies of the client and the platform simultaneously using the generative adversarial framework [32]. MAIL-MMSG trains a discriminator to distinguish between simulated and real interactions, using signal feedback as a reward to generate more realistic interactions. After the agent and environmental interactions are generated, the virtual environment is complete and can be used for platform policy training. In our experiments, we created a virtual environment using records from a real mixed ministry service environment and compared it with the actual environment. The results showed that the virtual environment accurately replicates properties close to the real environment. We then used the virtual environment to train a resource scheduling policy for the multi-scenario mixed service to maximize revenue. The policy trained in the virtual environment outperformed traditional supervised learning approaches, improving revenue in the real environment, with no physical trial costs.

The contributions of this paper are:

(1) For the construction of a multi-scenario virtual environment of mixed service groups, it's necessary to sample the perception-decision-execution behavior of the mixed service groups. The distribution of the group's characteristics should closely map the real distribution.

(2) The environment and policy models are trained simultaneously through the dual and adversarial learning of the virtual environment and the agent, enabling more realistic interactions.

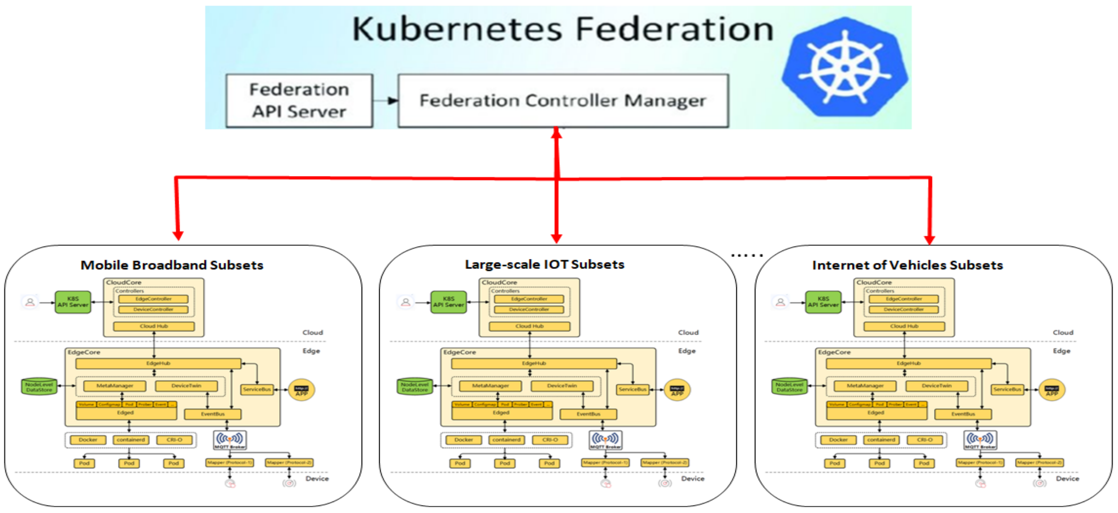


Figure 1. Multi-scenario under edge cloud-native environment

2 Motivating Scenario

Implementing reinforcement learning in a real user environment can be costly due to trial and error, even with minimal traffic. Simply transferring the reinforcement learning algorithm to the real world is impractical. However, by learning virtual environment models from historical data in real-world scenarios, we can achieve significant advantages in terms of generalization ability.

To ensure the disaster tolerance and other runtime state requirements of services in the scheduled container, the scheduling system allows business applications to set unique type requirements, exclusive requirements, and mutual exclusion and affinity when scheduling. These firm and flexible rules undoubtedly add complexity to the knapsack problem.

We delved into scenarios where online and offline tasks coexisted. Suppose the online task chooses to drop some containers to allocate resources for the offline task, based on current business service requirements. Which instances would be most sensible to reduce? While scaling is a factor to consider, should it also be accounted for in capacity allocation? Another factor to consider is time constraints. While it's not critical, adhering to previously outlined conditions is important. Typically, each request should be completed within a maximum of 180 seconds. The scale of the controlled host usually falls around the 10,000 mark. The high potential concurrency level of requests should also be taken into account.

All of our previous simulations were static, such as the Sigma-cerebro scheduling simulator [33], which was a tool platform, using 1:1 production data to perform scheduling and distribution simulations. These simulations are completely data-driven, based on static data for dynamic prediction. This approach is due to the challenge of simulating the online conditions of tens of thousands of hosts, which will require a lot of resources. This brings two problems:

Firstly, even if static requirements are met, it's unclear whether various microservices will coexist harmoniously. We need to determine the most effective combination of applications and explore whether cutting peak load and using resources efficiently through approaches like CPU share is preferable.

Secondly, our current static simulations can't answer these questions.

This paper primarily focuses on the unified scheduling approach for mixed-service resources across multiple application scenarios as shown in Figure 1. We utilize a virtual environment to simulate business Scenarios and utilize the Reinforcement Learning (RL) algorithm to train strategies. This exploration helps address the challenge of unified scheduling for mixed service groups under multiple scenarios.

3 Our Approach

In this section, we give a formal description of the problem in Section 3.1, propose the framework in Section 3.2, build a multi-scenario virtual environment in Section 3.3, and train the environment and policy models simultaneously through the dual and adversarial learning in Section 3.4.

3.1 Problem Formulation

In the native edge cloud, there are numerous sub-scenarios. Each is optimized independently, resulting in a competitive dynamic between them. Unfortunately, enhancing the performance of individual sub-scenarios doesn't necessarily lead to overall improvement. To address this, the sorting problem of multiple sub-scenarios was treated as a series of fully cooperative and partially observable multi-agent sequential decision issues. This approach allowed for the exploration of optimization strategies, shifting the deployment strategy of each scenario from being independent to a cooperative, win-win approach.

3.1.1 Markov Decision Process Formulation

In a classical reinforcement learning problem, there will be a formulation $(o_1, r_1, a_1, \ldots, a_t, o_t, r_t)$, where $o/r/a$ denotes observation, payoff, and action, respectively. As mentioned before, the environment in our problem is partially observable, which means that the state St represents experience, i.e. $s_t = f(o_1, r_1, a_1, ..., a_{t-1}, o_t, r_t)$, we are considering a problem with N agents $\{A^1, A^1, ..., A^N\}$, each of which corresponds to an optimization scenario with a feature (e.g. mobile broadband, Internet of things, Internet of vehicles, etc.). In this multi-agent setting, the state of the environment (St) is global and shared by multiple agents; But observations ($o_t = (o_t^1, o_t^2, ..., o_t^N)$), actions ($a_t = (a_t^1, a_t^2, ..., a_t^N)$), and memories of short-term rewards ($r_t = (r(s_t, a_t^1), r(s_t, a_t^2), ..., r(s_t, a_t^N))$) are all owned by the individual agents.

More specifically, each agent A^i will take each decision action a_t^i based on its policy $\pi^i(s_t)$ and state St, and then it will receive a temporary reward $r_t^i = r(s_t, a_t^i)$ from the environment while the state is updated from St to St+1. In our task, multiple agents will cooperate to achieve the overall maximum payoff. We have a global "action-value" function (critic) $Q(s_t, a_t^1, a_t^2, ..., a_t^N)$ that estimates the global payoff for the whole when taking an action $(a_t^1, a_t^2, ..., a_t^N)$ under the current state. We also have a global state representation that each agent will perform a local action after getting a local observation.

3.1.2 Simulator-based RL for Optimization

In this paper, we adhere to the general process of simulator-based RL optimization. We first define an environment simulator M: $S \times A \times S \rightarrow Y$. Specifically, the goal of the user simulator can be formally defined as follows: Given a state-action pair (s, a), imitate the user's feedback y on action a based on state s. For each time step t, given the prediction $\hat{y}_{t+1}$, we first update s_{t+1}^{hist} and s_{t+1}^{stat} then load $s_{t+1}^{agent,r}$ and $s_{t+1}^{group,r}$ according to $\hat{y}_{t+1}$, from a real trajectory dataset $D\tau^r$,where $\tau^r = [s_0^r, a_0^r, s_1^r, a_1^r, ..., s_T^r, a_T^r]$. Finally, we have $s_{t+1} = [s_{t+1}^{hist}, s_{t+1}^{stat}, s_{t+1}^{agent,r}, s_{t+1}^{group,r}]$ and rewards $r_t = R(s_t, a_t, s_{t+1})$. We define a symbol $P_{M,\tau^r}(s'|s, a)$ as the above transition process based on M and τ^r. Note that instead of directly predicting the entire next state s_{t+1}, the simulator just predicts the past y and builds other states from historical τ^r.

The overall goal of simulator-based RL is to find an optimal policy $\hat{\pi}^*$ that maximizes the cumulative reward for all users. In particular, the goal is written as follows:

$$\max E_{g\sim p(g),u\sim p(u),\tau^r\sim D(u,g)}[E_{r\sim p(\tau|\pi,P_M)}[\sum_{t=0}^{T}\gamma^t r_t]] \quad \textbf{(1)}$$

where $p(g)$ and $p(u)$ are the prior distributions of multi-scenario mixed service groups and sub-scenarios, $\tau^r \sim D(u,g)$ represent the true trajectories of sub-scenarios u sampled from the log dataset D of multi-scenario mixed service groups g, and $p(\tau \mid \pi, P_M)$ is the probability of generating trajectories $\tau = [s_0, a_0, r_0, ..., s_T, a_T, r_T]$ under the policy π and transition functions P_M. In particular,

$$p(\tau \mid \pi, P) = d_0(s_0)\prod\nolimits_{t=0}^{T} P(s_{t+1} \mid s_t, a_t)\pi(a_t \mid s_t) \quad \textbf{(2)}$$

where $d_0(s_0)$ is the initial state distribution.

3.2 Outline

Our Approach is a multi-agent training strategy that can be used to simultaneously train the policies of multiple agents and the policies of virtual environments. In this way, the multi-agent policy obtained can include different scheduling strategies. By training both the agent and the virtual environment together, only historical data is required, without the need to access the real environment, and Nash equilibrium can be achieved among multiple agents.

Figure 2 shows the process of our approach, which is introduced as follows:

(1) First, we obtained the features of mixed-services under multiple scenarios, and then utilized a generative adversarial network that simulates the distribution of mixed services to construct multiple-scenario virtual environments.

(2) A multi-agent adversarial imitation learning approach is proposed, which allows agents to interact with the virtual environment during the training process, while continuously optimizing the reward function.

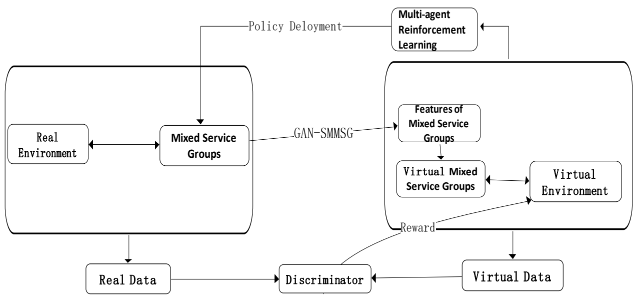


Figure 2. Framework of our approach

3.3 Building a Multi-Scenario Virtual Environment

To build a multi-scenario simulator, it's necessary to generate characteristics of multi-scenario mixed service groups. This involves sampling the perception-decision-execution behaviors of these groups from multiple scenarios to initiate their interaction process. The distribution of the generated characteristics for a single scenario should resemble the actual distribution. However, learning the feature distribution of a single scenario mixed service group in high-dimensional space can be challenging. Traditional approaches like the Gaussian mixture model struggle to achieve this similarity, whereas the Generative Adversarial Network (GAN) framework can effectively generate samples that closely align with the original data. Yet, the traditional GAN discriminator can only determine if the behavior data of a single service instance is real, without capturing the characteristics of the service group in a single scenario. To generate the distribution of the mixed service groups, as opposed to a single instance, we use the Generative Adversarial Network Simulating Multi-scenario mixed service groups (GAN-SMMSG), as illustrated in Algorithm 1. GAN-SMMSG maintains a generator (G) and a discriminator (D). The discriminator aims to accurately distinguish the generated data from the training data by maximizing the following objective function

$$E_{P_{x\sim D}}[\log D(x)] + E_{P_{x\sim G}}[\log(1 - D(G(z))] \quad \textbf{(3)}$$

The generator is updated to maximize the following objective function

$$E_{\mathrm{p}_G,\,\mathrm{p}_D}\left[D(G(z)) + aH(V(G(z))) - \beta KL(V(x) \| V(G(z)))\right] \quad \textbf{(4)}$$

We shorten $\mathrm{p}_{x\sim G}$ to p_G, $\mathrm{p}_{x\sim D}$ to p_D. $G(z)$ is the instance generated by the noise sample z, $V(\cdot)$ represents some variable associated with internal value. $H(V(G(z))$ denote the entropy of the variable derived from the generated data, used to create a broader distribution. $KL(V(x)\|V(G(z)))$ is the KL divergence between the training data and the variables that generated the data, guiding the generated distribution towards the training data's distribution. Under the constraints of KL divergence and entropy, GAN-MMSG can learn from the generator more effectively. It uses guiding information from actual data and can generate superior distributions compared to the traditional GAN.

Algorithm 1. Generative Adversarial Network Simulating Multi-scenario Mixed Service Groups (GAN-SMMSG)

```
Input: Real data distribution $\mathrm{p}_D$
Body
Initialize training variables $\theta_D$, $\theta_G$
for i = 0, 1, 2, ...do
for k steps do
sample mini-batch from $\mathrm{p}_G$
sample mini-batch from $\mathrm{p}_D$
 update the generator by gradient:
$\forall_{\theta_G} E_{\mathrm{p}_G,\,\mathrm{p}_D}\left[D(G(z)) + aH(V(G(z))) - \beta KL(V(x)\|V(G(z)))\right]$ end for
sample mini-batch from $\mathrm{p}_G$
sample mini-batch from $\mathrm{p}_D$
update the discriminator by gradient:
$\forall_{\theta_G} E_{\mathrm{p}_{x\sim D}}[\log D(x)] + E_{\mathrm{p}_{x\sim G}}[\log(1 - D(G(z))]$
end for
Output: agent generator $G$
```

3.4 Multi-agent Adversarial Imitation Learning

We propose an approach called Multi-agent Adversarial Imitation Learning of Multi-scenarios Mixed Service Groups (MAIL-MMSG), inspired by GAIL. GAIL enables the agent to engage with the environment during training, while simultaneously optimizing the reward function. It's important to note that the environment needs to be accessible during GAIL training. However, when training the client policy, the environment is expected to be an unknown or dynamic one.

The GAIL algorithm consists of a discriminator and a policy. The policy functions as a generator in a generative adversarial network. Given a state, the policy dictates the action to be taken, and the discriminator D processes the state-action pair (s, a) as input. It then outputs a real number between 0 and 1, which represents the probability that the state-action pair originated from the agent's policy, not an expert. The discriminator aims to bring the output for expert data closer to 0 and the output for the imitator policy closer to 1, enabling a clear distinction between the two sets of data. Consequently, the loss function of the discriminator is defined as follows:

$$L(\phi) = -E_{\mathrm{p}_\pi}[\log D_\phi(s,a)] + E_{\mathrm{p}_\pi}[\log(1 - D_\phi(G(s,a))] \quad \textbf{(5)}$$

where ϕ are the parameters of the discriminator. With discriminator D, the goal of the imitator strategy is that the trajectories produced by its interactions can be mistaken for expert trajectories by the discriminator. Therefore, we can use the output of Discriminator D as the reward function to train the imitator policy. Specifically, if the imitator policy samples the state s of the environment and takes action a, the state-action pair (s, a) is fed into the discriminator D, outputs $D_\phi(s, a)$, and then the reward is set to $r(s, a) = -\log D_\phi(s, a)$. We can then use this data with any reinforcement learning algorithm to continue training the imitator policy. Finally, after the adversarial process continues, the data distribution generated by the imitator strategy will be close to the real expert data distribution, achieving the goal of imitation learning.

MAIL-MMSG is a multi-agent approach that simultaneously trains the client and engine policies, unlike GAIL which trains a single-agent policy in a static environment. This allows the learned client policies to generalize across different environment policies. Because MAIL-MMSG trains both policies concurrently, it requires only historical data and doesn't need access to the real environment.

However, learning the agent policy and the environment policy iteratively can result in a large search space, leading to poor performance. Luckily, we can optimize both simultaneously. To model the agent policy π^c, we parameterize customer policy π^c_k by k, environment policy π_σ by σ, and reward function R^c_θ by θ. If the customer observation $s^c =< s, a, n >$ depends on the action a, we have:

$$\pi^c(s^c, a^c) = \pi^c(< s, a, n >, a^c) = \pi^c(< s, \pi(s,\cdot), n >, a^c) \quad \textbf{(6)}$$

This shows that the joint policy $\pi^c_{k,\sigma}$ can be seen as a mapping from S×N to A^c given the environment policy. In other words, given the parameters of the environment agent, the customer agent can make the decision directly. Since $S^c = S\times A\times N$, for convenience, we still consider $\pi^c_{k,\sigma}$ is the mapping from S^c to A^c. Joint policies π and π^c together provide the opportunity to learn agent policies and environments simultaneously. The reward function R^c is designed to be non-discriminative in generating data and historical state-action pairs. Adopting a reinforcement learning algorithm will maximize the reward, which means generating indistinguishable data.

Algorithm 2 shows the processing of MAIL-MMSG. we need historical traces τ_e and agent distribution P_c to run MAIL. In this paper, GAN-SMMSG is used for pre-learning P_c. After initializing the variables, we start the main process of MAIL-MMSG: in each iteration, we collect trajectories during the interactions between the agent and the environment. We then sample from the generated trajectories and optimize the reward function via a gradient approach. Then, k and τ is updated by a joint policy $\pi^c_{k,\sigma}$ from M^c by RL optimization. When the iteration ends, MAIL-MMSG returns the agent policy π^c.

After simulating P_c and π^c, we know how the agent behaviors are generated and how they react to the environment, and the virtual environment has been set up. We can generate interactions by deploying engine policies to the virtual environment.

Algorithm 2. Multi-agent Adversarial Imitation Learning of Multi-scenario Mixed Service Groups (MAIL-MMSG)

```
Input: Expert trajectories $\tau_e$, agent distribution $p_D$
Body
Initialize variables $k, \sigma, \theta$
for i = 0,1,2,…I do
for j = 0,1,2,…J do
$\tau_j = \varnothing, s \sim P^c, a \sim \pi_\sigma(S^c, \cdot), S^c = < s, a >$
while NOT TERMINATED do
        sample $a^c \sim \pi_\sigma(s^c, \cdot)$
        add $< s^c, a^c >$ to $\tau_j$
        generate $s^c \sim \tau_\sigma^c(s^c, a^c \mid P^c)$
end while
end for
sample trajectories $\tau_g$ from $\tau_{0\sim J}$
update $\theta_{0\sim J}$ to in the direction to minimize:
$\sum_{0\sim J} E_{r_\theta}[\log D(R_\theta^c(s,a))] + E_{p_{x\sim g}}[\log(1 - D(R_\theta^c(s,a))]$
 update $k, \sigma$ by optimizing $\pi_{k,\sigma}^c$ with RL in $M^c$
end for
Output : agent policy $\pi^c$
```

4 Experiments

In this paper, a Kubernetes-based edge cloud-native platform is utilized to transition the Multi-agent Adversarial Imitation Learning (MAIL) [34] to an edge cloud-native scenario. By leveraging the duality and adversarial learning of the environment and agent, the environment model and policy model are concurrently trained. This paper concludes with a verification of the process.

4.1 Experimental Design

We utilized an edge-cloud network consisting of cloud servers, edge nodes, and terminal devices to support multi-scenario mixed service groups as our experimental environment. The cloud server is a physical machine with a 16-core Intel Xeon E5-2620 v4 CPU, 64GB of memory, a 1TB hard drive, and Ubuntu 18.04 operating system, providing powerful computing and storage capabilities as the cloud computing center. The connection between the cloud server, edge nodes, and terminal devices is established through wired or wireless networks, with network bandwidth and latency dynamically changing according to the actual situation.

On the software side, we have built a mixed architecture based on Karmada, RunD, and Koordinator. We use their APIs and tools to implement various resource scheduling strategies. The training process includes real-time acquisition of user behavior logs to provide training samples for the MA-PPO algorithm. These samples are then stored in a replay buffer. The model is updated, and the revised model is applied online. This process repeats, allowing the online model to be dynamically updated to capture changes in agent behavior.

In terms of parameter settings, for each agent (scene) using PPO, the local observations are a 52-dimensional vector, and the actions correspond to 7-dimensional and 3-dimensional vectors. For simplicity, we output a vector of length 10 (filled with 0s to account for the vacant part) from the evaluation network, as both the communication module and the evaluation network require behavior from each different scene.

In PPO, the actor network has 2 hidden layers, the number of neurons in each layer is 32, and ReLU is used as the activation function. The gain attenuation coefficient in the Bellman formula is set to γ = 0.95. In our experiments, we use RMSProp to learn the parameters of the network; The learning rate is $10^{-2\text{-}3}$and 10, and the hidden layer of the network is 128 layers, corresponding to the actor network and the critic network, respectively.

The discriminator uses a 4-layer fully connected network. It takes state-action pairs as inputs and outputs a probability scalar. The replay buffer size is 104, while the minibatch size is 100.

To assess the virtual environment's impact, we use metrics such as average task completion time, convergence time, and offloading ratio (defined as the ratio of offloaded tasks to the total number of tasks generated in the network).

All measurements were obtained from online experiments. To compare these metrics between the real and virtual environments, we implement the random engine strategy in the real environment and collect the corresponding trajectories as historical data. Please note that we do not assume the engine strategy that generated the data. When building the virtual environment, the engine strategy could be an unknown complex model.

We simulated the customer distribution P_c with GAN-SMMSG, where α = β = 1. Then we construct the virtual environment using MAIL-MMSG, implementing the PPO RL approach. All function approximates in this study utilize multi-layer perception. Due to resource constraints, we could only compare both strategies concurrently in the online experiment.

We demonstrate that the virtual agent's distribution is similar to that of the real agent using Algorithm 1. This is achieved by deploying the history engine strategy in the virtual environment and comparing the agent's offloading ratio over time and their characteristics in both the virtual and real environments. We ran algorithm 2 in the virtual environment.

4.2 Analysis

4.2.1 Virtual Agent Distributions and Behaviors

To evaluate the ability of simulating agent distribution, we compare our approach with the following approaches:

(1) GMM (Gaussian Mixture Model) [30]: This approach combines multiple Gaussian models to form a mixture model. It is used to calculate the probability distribution of the data.

(2) GAN (Generative Adversarial Networks) [29]: Generative Adversarial Networks are composed of two networks: the generator network generates simulated data, and the discriminator network determines whether the input data is real or generated. The generator network continuously optimizes the generated data to make it indistinguishable from the discriminator network, while the discriminator network also optimizes itself to make more accurate judgments. The interaction between the two networks creates an adversarial scenario.

(3) WGAN (Wasserstein generative adversarial network)

[35]: In the original GAN, the discriminator measures the JS divergence between two distributions. However, using JS divergence can lead to unstable training. To address this issue, this approach uses Wasserstein Distance to measure the distance between two probability distributions. WGAN is proposed to improve the stability of GAN model training and avoid mode collapse.

The ratio of different agents is a fundamental criterion for evaluating the virtual environment. We simulate agent distribution using GAN-SMMSG, generating 100,000 agents, the agents are divided into 3 levels, corresponding to 3 sub-scenes. The proportions are calculated in four dimensions: average task time, convergence time, offloading ratio, and agent. These results are compared with the ground truth. Figure 3 illustrates that the GAN-SMMSG distribution closely matches the ground truth. Table 1 details the specific KL difference between the generated and real agents. In our experiments, we carefully selected the number of GMM clusters to be 10 to fit the data, finding that traditional GANs struggled to capture the distribution structure.

Virtual agents are generated by GAN-SMMSG. The agents have varying feature preferences, which affect the offloading ratio. To assess if the virtual environment accurately simulates reality, we analyze the impact of agent characteristics on the offloading ratio within this environment and compare these findings with real-world results. As depicted in Figure 3, the results from the virtual environment closely resemble the ground truth.

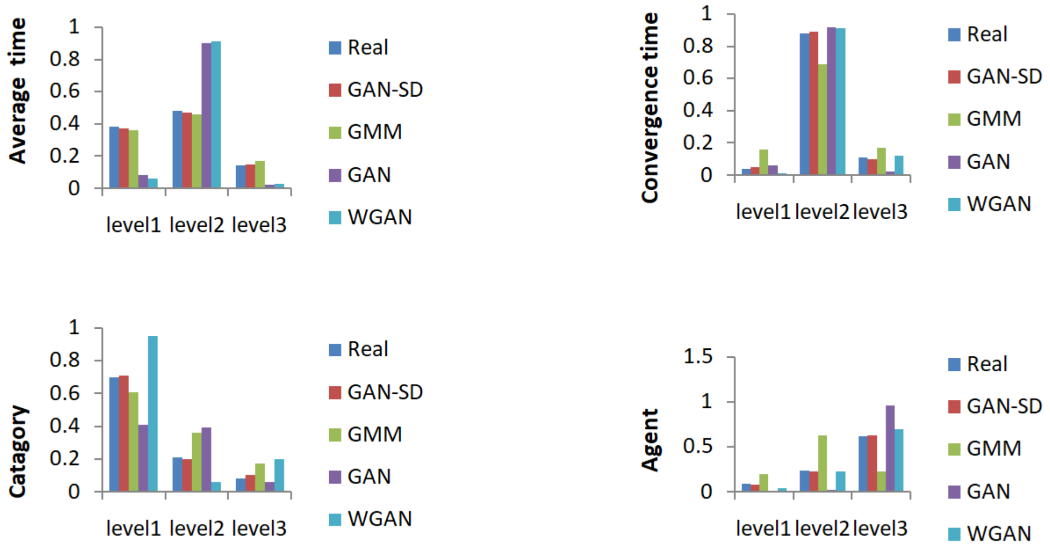


Figure 3. Comparisons of the virtual agent distributions

Table 1. KL divergence between virtual agents and real agents

KL	Average task time	Convergence time	Offloading ratio	Agent
GAN-SMMSG	0.00	0.00	0.01	0.00
GMM	0.03	0.03	0.07	0.32
GAN	0.70	0.11	0.18	0.90
WGAN	0.68	0.03	0.28	0.02

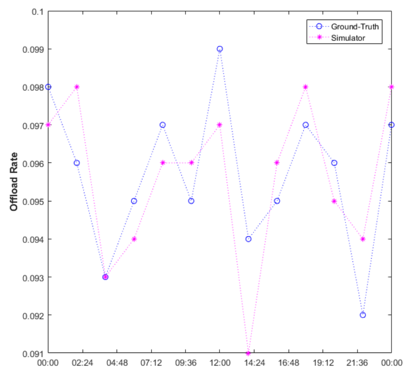


Figure 4. Comparisons of offload-ratio between the reality and virtual environment

The offload ratio of the agent changes over time, suggesting that the virtual environment should have similar characteristics. However, because our agent model doesn't account for time, we split the historical data of a day into 12 parts in chronological sequence to simulate the offload ratio's temporal changes. Each partitioned dataset is used to independently train a virtual environment. Each virtual environment then deploys the same history engine policy, specifically the random policy. We present theoffload ratio in both the virtual and real environments. As illustrated in Figure 4, the virtual environment can mirror the offload ratio's temporal trends.

4.2.2 Generalization Ability of Our Approach

Next, we will assess whether the proposed approach's strategy has superior generalization capabilities. Given there is no prior work on constructing virtual environments, we compare our approach with a traditional behavior cloning (BC) algorithm. This algorithm learns a mapping from customer states to actions, assuming the data is independently and identically distributed, which isn't accurate in practice. As a result, the BC algorithm can't discern the agent's true intent, and minor environmental changes can significantly decrease accuracy. In contrast, the MAIL approach learns long-term behavior without assuming independent and identical distribution, thus potentially offering better generalization.

Since the goal of building the virtual environment is to train RL algorithms offline, we use the RL approach to learn policies on the virtual environment S and use Behavior Cloning (BC) approach to learn policies on historical data S_1. Please note that the virtual environment is constructed solely based on historical data.

The first baseline approach adopts Behavior Cloning (BC), which fits a model based on historical data to generate actions that are close to the "correct" ones, using the following loss function:

$$\pi^*_{BC} = \arg\min_{\pi} \frac{1}{|S_1|} \sum_{(s,a)\in S_1} |\pi(s)-a|^2 \quad \textbf{(7)}$$

The second baseline adopts MAIL-MMSG, using the following loss function:

$$\pi^*_{MAIL\text{-}MMSG} = \arg\min_{\pi} \frac{1}{|S_1|} \sum_{(s,a)\in S_1} |\pi(s)-a|^2 + \frac{\lambda}{|S|} \sum_{(s,a)\in S} |\pi(s)|^2 \quad \textbf{(8)}$$

where λ is set to 0.4.

We utilize one day's data to construct a virtual environment using MAIL-MMSG, and then use one day, one week, and one month's data to create three more environments. We initially run our approach in the first environment and implement the resulting policies in the others. As environments grow more distinct over time, the offloading ratio is anticipated to decrease, reflecting the ability to generalize to new environments. We replicated this process, substituting MAIL with the behavior cloning approach (BC), which uses the same network structure as MAIL-MMSG. Table 2 shows the offload ratio improvement for the random strategy. Offloading decreases faster in the BC environment than in others. After a month, the BC environment strategy performs worse than the random strategy.

Table 2. Offloading ratios improved from two simulators by BC and MMSG-MAIL

	0 day	1 day	1 week	1 month
BC	21.45%	8.36%	1.05%	0.54%
MAIL-MMSG	18.57%	16.68%	8.44%	8.47%

5 Related Work and Discussion

This paper studies the multi-scenario unified resource scheduling problem in an edgecloud-native environment. It involves edge computing, cloud-native computing, resource allocation, and task scheduling, presenting both high theoretical complexity and practical value.

Several issues arise in the unified scheduling of mixed-scenario services: How can we evaluate the effectiveness of scheduling and allocation results? How can we accurately assess resources when numerous applications are generated simultaneously? How can we recreate a real-world scheduling problem in the virtual environment?

There are two potential solutions to these problems: one is using a supervised learning approach; the other involves learning the environment model from historical data first, and then deriving the scheduling policy from that model.

Supervised learning approaches include:

(1) Behavior Cloning (BC) primarily learns policies from state-behavior data using supervised approaches. However, BC requires an independent and identically distributed assumption on demonstration data, which is not met in reinforcement learning (RL) tasks [22].

(2) Inverse Reinforcement Learning (IRL) learns a reward function from data and then trains a policy based on this function. IRL relaxes the independent and identically distributed data assumption, but still assumes that the environment is static. IRL relaxes the independent and identically distributed assumption of data, but still assumes that the environment is static.

Given the highly dynamic nature of real-world environments, these approaches have limitations. They make it unfeasible to learn policies directly from data. Moreover, building a unified resource scheduling policy for multiple scenarios is challenging.

Reinforcement learning training often uses simulators to avoid physical costs. A prime example is Google's application for data center cooling. Here, a neural network models the system dynamics, and the policy is trained in the simulated environment using a state-of-the-art RL algorithm [36]. However, simulating the behavior of mixed service groups across multiple scenarios in the native environment of the edge cloud is challenging. Current RL algorithms typically require abundant interactions with the environment. Any errors in the environment model can increase the square error of the policy. Despite this, successful environment learning can offer unparalleled advantages in terms of policy generalization and application.

This paper aims to understand the virtual environment from historical data to achieve reinforcement learning with "zero cost" training. However, the behavior of the highly dynamic mixed-service community is complex. Can it be successfully simulated? Is this technical approach feasible?

Sigma, Alibaba's container scheduling system, has delved into this topic. Its current simulator, Cerebro, is a tool that uses 1:1 production data to perform scheduling and allocation simulations [33]. Currently, simulations are purely data-level, with dynamic predictions also based on static data. However, several questions remain unanswered by these static simulations. Can various microservices harmonize once static requirements are met? What combination of microservices is the most effective? Is it more efficient to reduce peak load and utilize resources, such as by sharing CPUs? These are all inquiries that current static simulations cannot address.

Generative Adversarial Imitation Learning (GAIL) [31] was recently proposed to overcome the fragility of behavior cloning using GAN frameworks as well as the costality of inverse reinforcement learning. GAIL allows the agent to interact with the environment and learn a policy through reinforcement learning approach, while improving the reward function during training. Hence, RL approaches are generators in the GAN framework. GAIL employs a discriminator D to measure the similarity between policy-generated trajectories and expert trajectories. Practice shows that the theoretical and empirical results of GAIL are similar to those of IRL, and GAIL is more efficient. GAIL has become a popular choice for imitation learning [37], and there already exist model-based extensions [38] and third-person extensions [39]. Inspired by the work of Yang et al. [40]. we will migrate MAIL (Multi-agent Adversarial Imitation Learning) to edge cloud-native scenarios.

Generative Adversarial Imitation Learning (GAIL) was recently proposed to address the limitations of behavior cloning and the complexity of inverse reinforcement learning. GAIL allows the agent to interact with the environment, learning a policy through reinforcement approaches while refining the reward function during training. In this framework, reinforcement learning approaches function as generators. GAIL employs a discriminator to compare policy-generated trajectories and expert trajectories. Empirical evidence shows that GAIL's theoretical and practical results are similar to those of inverse reinforcement learning, but GAIL is more efficient. As such, it has become a popular choice for imitation learning [37], with existing model-based extensions [38] and third-person extensions [39]. Inspired by the work of Yu Yang et al. [34] and Fan Yang et al. [40], we adapted Multi-agent Adversarial Imitation Learning (MAIL) to edge cloud-native scenarios.

6 Conclusion and Future Work

In conclusion, this paper presents an approach to resource scheduling in edge-cloud-native environments using Multi-agent Adversarial Imitation Learning. Our approach has effectively demonstrated its capability to replicate a real-world scheduling problem within the virtual environment, and it stands out for its superior generalization capabilities when compared to traditional approaches. This superiority is particularly notable given the dynamic and complex nature of real-world environments.

Our approach offers a promising solution to the challenges of resource scheduling in mixed-service scenarios, bringing both high theoretical complexity and practical value. It addresses the inherent issues in the unified scheduling of mixed-scene services such as evaluating the effectiveness of scheduling and allocation results, accurately assessing resources when numerous applications are generated simultaneously, and recreating a real-world scheduling problem in the virtual environment.

Future work will concentrate on further optimizing this approach, refining the algorithms, and enhancing the ability of the virtual environment to map the temporal trends of the offload ratio in real-world environments. We also aim to explore the potential applications of this approach in other complex scheduling scenarios. Given the growing complexities of edgecloud-native environments, the need for effective and efficient resource scheduling strategies is only projected to increase. Therefore, the lessons learned and the methodologies developed in this study could have implications for future research and developments in this field.

Acknowledgment

The work has been supported by the Natural Science Foundation of Hubei Province in China (No. 2021CFB482), the Basic Research Science and Technology Project of Xiangyang (High-tech Domain 2022ABH007013), and Hubei Superior and Distinctive Discipline Group of "New Energy Vehicle and Smart Transportation".

References

[1] A. Damiani, G. Fiscaletti, M. Bacis, R. Brondolin, M. D. Santambrogio, BlastFunction: A Full-stack Framework Bringing FPGA Hardware Acceleration to Cloud-native Applications, *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, Vol. 17, pp. 1-27, June, 2022.

[2] A. Boudi, M. Bagaa, P. Poyhonen, T. Taleb, H. Flinck, AI-Based Resource Management in Beyond 5G Cloud Native Environment, *IEEE Network*, Vol. 35, No. 2, pp. 128-135, March/April, 2021.

[3] S. Pardeshi, C. Khairnar, K. Alfatmi, Analysis of Data Handling Challenges in Edge Computing, *International Journal of Performability Engineering*, Vol. 18, No. 3, pp. 176-187, March, 2022.

[4] H. Deng, X. Zhang, J. Jiang, J. Wang, H. Huang, Privacy Protection of Personal Education Information on Blockchain, *International Journal of Performability Engineering*, Vol. 18, No. 5, pp. 317-328, May, 2022.

[5] G. A. Jimenez-Maggiora, S. Bruschi, H. Qiu, J. So, P. S. Aisen, Corrigendum to: ATRI EDC: a novel cloud-native remote data capture system for large multicenter Alzheimer's disease and Alzheimer's disease-related dementias clinical trials, *JAMIA Open*, Vol. 5, No. 1, pp. 1-8, April, 2022.

[6] M. Al-Quraan, L. Mohjazi, L. Bariah, A. Centeno, A. Zoha, K. Arshad, K. Assaleh, S. Muhaidat, M. Debbah, M. Imran, Edge-Native Intelligence for 6G Communications Driven by Federated Learning: A Survey of Trends and Challenges, *IEEE Transactions on Emerging Topics in Computational Intelligence*, Vol. 7, No. 3, pp. 957-979, June, 2023.

[7] H. Wang, G. Tang, K. Wu, J. Wang, PLVER: Joint Stable Allocation and Content Replication for Edge-assisted Live Video Delivery, *IEEE Transactions on*

Parallel and Distributed Systems, Vol. 33, No. 1, pp. 218-230, January, 2022.

[8] S. Kassir, G. Veciana, N. Wang, X. Wang, P. Palacharla, Joint Update Rate Adaptation in Multiplayer Cloud-Edge Gaming Services: Spatial Geometry and Performance Tradeoffs, *Twenty-second International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, Shanghai, China, 2021, pp. 191-200.

[9] N. Sreekumar, A. Chandra, J. Weissman, Position Paper: Towards a Robust Edge-Native Storage System, *IEEE/ACM Symposium on Edge Computing (SEC)*, San Jose, CA, USA, 2020, pp. 285-292.

[10] G. Nain, K. K. Pattanaik, G. K. Sharma, Towards edge computing in intelligent manufacturing: Past, present and future, *Journal of Manufacturing Systems*, Vol. 62, pp. 588-611, January, 2022.

[11] Q. Huang, Y. Huang, The Significance of Urban Cockpit for Urban Brain Construction, *11th International Conference on E-business, Management and Economics*, Virtual Conference, 2020, pp. 70-73.

[12] L. Loven, T. Leppanen, E. Peltonen, J. Partala, E. Harjula, P. Porambage, M. Ylianttila, J. Riekki, EdgeAI: A Vision for Distributed, Edge-native Artificial Intelligence in Future 6G Networks, *6G Wireless Summit*, Levi, Finland, 2019, pp. 10-18.

[13] J. Okwuibe, J. Haavisto, E. Harjula, I. Ahmad, M. Ylianttila, SDN Enhanced Resource Orchestration of Containerized Edge Applications for Industrial IoT, *IEEE Access*, Vol. 8, pp. 229117-229131, December, 2020.

[14] X. He, H. Lu, Y. Mao, K. Wang, QoE-driven Task Offloading with Deep Reinforcement Learning in Edge intelligent IoV, *IEEE Global Communications Conference*, Taipei, Taiwan, 2020, pp. 1-6.

[15] Y. Zhai, W. Sun, J. Wu, L. Zhu, J. Shen, X. Du, M. Guizani, An Energy Aware Offloading Scheme for Interdependent Applications in Software-Defined IoV With Fog Computing Architecture, *IEEE Transactions on Intelligent Transportation Systems*, Vol. 22, No. 6, pp. 3813-3823, June, 2021.

[16] M. Liwang, R. Chen, X. Wang, Resource Trading in Edge Computing-enabled IoV: An Efficient Futures-based Approach, *IEEE Transactions on Services Computing*, Vol. 15, No. 5, pp. 2994-3007, September-October, 2022.

[17] M. C. Ogbuachi, A. Reale, P. Suskovics, B. Kovacs, Context-Aware Kubernetes Scheduler for Edge-native Applications on 5G, *Journal of Communications Software and Systems*, Vol. 16, No. 1, pp. 85-94, March, 2020.

[18] S. H. VanderLeest, ARINC 653 hypervisor, *29th Digital Avionics Systems Conference*, Salt Lake City, UT, USA, 2010, pp. 5.E.2-1-5.E.2-20.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing Atari with Deep Reinforcement Learning, *arXiv*, arXiv: 1312.5602, December, 2013. https://arxiv.org/abs/1312.5602

[20] S. Schaal, Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, Vol. 3, No. 6, pp. 233-242, June, 1999.

[21] B. D. Argall, S. Chernova, M. Veloso, B. Browning, A survey of robot learning from demonstration, *Robotics and autonomous systems*, Vol. 57, No. 5, pp. 469-483, May, 2009.

[22] V. G. Goecks, G. M. Gremillion, V. J. Lawhern, J. Valasek, N. R. Waytowich, Integrating Behavior Cloning and Reinforcement Learning for Improved Performance in Dense and Sparse Reward Environments, *19th International Conference on Autonomous Agents and MultiAgent Systems*, Auckland, New Zealand, 2020, pp. 465-473.

[23] P. Wang, D. Liu, J. Chen, H. Li, C. Chan, Human-like Decision Making for Autonomous Driving via Adversarial Inverse Reinforcement Learning, *arXiv*, arXiv: 1911.08044, February, 2020. https://arxiv.org/abs/1911.08044v2

[24] X. Zhao, L. Zhang, L. Xia, Z. Ding, D. Yin, J. Tang, Deep Reinforcement Learning for List-wise Recommendations, *arXiv*, arXiv: 1801.00209, June, 2019. https://arxiv.org/abs/1801.00209v3

[25] X. Zhao, L. Xia, L. Zou, D. Yin, J. Tang, Toward Simulating Environments in Reinforcement Learning Based Recommendations, *arXiv*, arXiv: 1906.11462, September, 2019. https://arxiv.org/abs/1906.11462

[26] M. Kearns, S. Singh, Near-Optimal Reinforcement Learning in Polynomial Time, *Machine Learning*, Vol. 49, No. 2/3, pp. 209-232, November, 2002.

[27] M. Janner, J. Fu, M. Zhang, S. Levine, When to trust your model: Model-based policy optimization, *arXiv*, arXiv: 1906.08253, November, 2021. https://arxiv.org/abs/1906.08253

[28] T. Xu, Z. Li, Y. Yu, Error bounds of imitating policies and environments, *arXiv*, arXiv: 2010.11876, October, 2020. https://arxiv.org/abs/2010.11876

[29] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, D. Krishnan, Unsupervised Pixel-Level Domain Adaptation with Generative Adversarial Networks, *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 2017, pp. 95-104.

[30] T. Toda, H. Saruwatari, K. Shikano, Voice conversion algorithm based on Gaussian mixture model with dynamic frequency warping of STRAIGHT spectrum, *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, Salt Lake City, UT, USA, 2001, pp. 841-844.

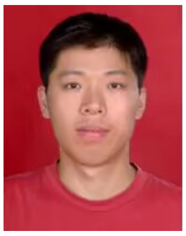
[31] X. Wang, J. Zhou, T. Song, D. Liu, Q. Wang, FlotGAIL: An operational adjustment framework for flotation circuits using generative adversarial imitation learning, *Minerals Engineering*, Vol. 183, Article No. 107598, June, 2022.

[32] H. Zhao, H. Li, S. Maurer-Stroh, L. Cheng, Synthesizing retinal and neuronal images with generative adversarial nets, *Medical Image Analysis*, Vol. 49, pp. 14-26, October, 2018.

[33] Ali container scheduling system Sigma simulation platform Cerebro was revealed, 2023. Available online: https://developer.aliyun.com/article/448315 (accessed on 13 October 2023).
[34] J. C. Shi, Y. Yu, Q. Da, S. Y. Chen, A. X. Zeng, Virtual-Taobao: Virtualizing Real-World Online Retail Environment for Reinforcement Learning, *arXiv*, arXiv: 1805.10000, May, 2018. https://arxiv.org/abs/1805.10000
[35] Z. Hu, H. Xue, Q. Zhang, J. Gao, N. Zhang, S. Zou, Y. Teng, X. Liu, Y. Yang, D. Liang, X. Zhu, H. Zheng, DPIR-Net: Direct PET Image Reconstruction Based on the Wasserstein Generative Adversarial Network, *IEEE Transactions on Radiation and Plasma Medical Sciences*, Vol. 5, No. 1, pp. 35-43, January, 2021.
[36] R. S. Sutton, A. G. Barto, Reinforcement Learning: An Introduction, *IEEE Transactions on Neural Networks*, Vol. 9, No. 5, pp. 1054-1054, September, 1998.
[37] A. Kuefler, J. Morton, T. Wheeler, M. Kochenderfer, Imitating driver behavior with generative adversarial networks, *2017 IEEE Intelligent Vehicles Symposium (IV)*, Los Angeles, CA, USA, 2017, pp. 204-211.
[38] N. Baram, O. Anschel, S. Mannor, Model-based adversarial imitation learning, *arXiv*, arXiv: 1612.02179, December, 2016. https://arxiv.org/abs/1612.02179
[39] B. C. Stadie, P. Abbeel, I. Sutskever, Third-person imitation learning, *arXiv*, arXiv: 1703.01703, September, 2019. https://arxiv.org/abs/1703.01703
[40] F. Yang, A. Vereshchaka, C. Chen, W. Dong, Bayesian Multi-type Mean Field Multi-agent Imitation Learning, *34th Conference on Neural Information Processing Systems*, Vancouver, Canada, 2020, pp. 1-10.

Biographies

Wei Xiong is an associate professor at Hubei University of Arts and Science, China. He received his Ph.D. degree in computer science from Wuhan University, China in 2015. His research interest includes edge computing and AI, as well as autonomous driving vehicles.



Xinying Wang is an associate professor at Hubei University of Arts and Science, China. He received his B.S. in education technology from Central China Normal University in 2000, M.S. degree in automatic control from Wuhan University of Technology in 2008. His research interest includes service computing and software engineering.

Franz Wotawa is a PhD graduate from the Vienna University of Technology in Austria. He currently holds a professorship at Graz University of Technology. Professor Wotawa's research focuses on intelligent systems, software verification, system testing, and autonomous driving vehicles.

Qiaozhi Hua is an associate professor at Hubei University of Arts and Science, China. He received the PhD from Waseda University, Tokyo, Japan in 2019. His main fields of research interests include mobile communications, wireless sensor networks, intelligent transportation systems and optical communications.