Cross-Domain DRL Agents for Efficient Job Placement in the Cloud-Edge Continuum

Theodoros Aslanidis
theodoros.aslanidis@ucdconnect.ie
University College Dublin
Dublin, Ireland

Sokol Kosta
sok@es.aau.dk
Aalborg University
Copenhagen, Denmark

Spyros Lalis
lalis@uth.gr
University of Thessaly
Volos, Greece

Dimitris Chatzopoulos
dimitris.chatzopoulos@ucd.ie
University College Dublin
Dublin, Ireland

ABSTRACT

The growing computational demands of modern applications call for resource management strategies that effectively utilize the strengths of both cloud and edge computing. Deep Reinforcement Learning (DRL) has shown great promise in addressing these challenges, offering advanced decision-making capabilities that optimize resource allocation and system performance. However, deploying DRL agents in cloud-edge continuum infrastructures remains a significant challenge due to their dependence on infrastructure-specific state-action representations. This paper presents a novel architectural framework for DRL agents that incorporates feature extraction and adaptation mechanisms to enable their seamless operation across diverse environments. By transforming state features into an infrastructure-agnostic representation, our approach reduces the need for extensive retraining when system configurations change. Experimental results show that our method outperforms both a heuristic method and a DRL baseline algorithm while achieving faster convergence when infrastructure and workloads change. This work is an important step forward in developing transferable and adaptable DRL solutions for real-world cloud-edge resource management challenges.

CCS CONCEPTS

• **Computing methodologies → Reinforcement learning**; **Dimensionality reduction and manifold learning**; • **Computer systems organization → Cloud computing**.

KEYWORDS

Deep Reinforcement Learning, Cloud-Edge Continuum, Resource Management, Job Placement, State Abstractions, Cross-Domain Transfer, Domain Adaptation, Transfer Reinforcement Learning

ACM Reference Format:
Theodoros Aslanidis, Sokol Kosta, Spyros Lalis, and Dimitris Chatzopoulos. 2025. Cross-Domain DRL Agents for Efficient Job Placement in the Cloud-Edge Continuum. In *The 5th Workshop on Machine Learning and Systems (EuroMLSys '25), March 30–April 3, 2025, Rotterdam, Netherlands.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3721146.3721934

1 INTRODUCTION

Motivation. Cloud computing has revolutionized the way computing resources are provisioned, offering centralized, on-demand access to storage, processing power, and applications. This paradigm enables flexible, scalable, and collaborative service deployment, making it a cornerstone of modern computing infrastructures [29]. However, the rapid proliferation of Internet of Things devices and the resulting explosion of data streams have exposed the limitations of centralized cloud infrastructures, which increasingly struggle to handle the computational load [1]. Edge computing addresses these challenges by distributing processing and storage closer to data sources, reducing latency and bandwidth demands for real-time applications [31]. Despite its advantages, edge computing often faces constraints in computational power, which caused the emergence of the cloud-edge continuum. This is a hybrid model that combines the strengths of both paradigms. The cloud-edge continuum combines robust, scalable resource management with local, real-time processing, making it particularly suited for applications in smart cities, healthcare, and other domains where both centralized control and fast responsiveness are essential [30]. Job placement in cloud-edge continuum environments is a critical and recurring challenge, as continuous fluctuations in workload, resource availability, and network conditions demand repeated, optimized decisions to maintain high service quality and efficient resource utilization [29, 31]. Moreover, while simple heuristics or rule-based approaches may offer an initial solution, they are generally inadequate for these complex systems because they cannot dynamically capture and adapt to the rapidly changing operational conditions inherent in heterogeneous cloud-edge infrastructures [2, 30].

The inherent complexity and heterogeneity of the cloud-edge continuum have motivated the integration of machine learning techniques to optimize decision-making. Among these, Reinforcement Learning (RL), and more specifically, Deep Reinforcement Learning (DRL) has proven particularly effective for managing dynamic, non-periodic user patterns and making long-term, strategic decisions in such environments [8, 21, 28, 43, 45]. Various DRL paradigms, including hierarchical [21], multi-agent [47], and multi-objective RL [14, 17], have been explored to improve scalability and adaptability. Moreover, recent advances in meta-learning [9] and continual reinforcement learning [16] have demonstrated promising approaches for reusing and retaining knowledge, enabling the development of more generalizable DRL agents in real-world scenarios [26]. These frameworks expose DRL models to a wide range of variances during training, promoting better generalization to new tasks while applying previously acquired knowledge without suffering from catastrophic forgetting [18, 25].

State of the art. These approaches mainly address variability in workload distributions, ensuring robust performance across different data inputs. While adapting to input distributions is crucial for mitigating model degradation due to domain shift and concept drift [20, 22], DRL solutions in real-world scenarios face an even more pressing challenge: the tight coupling between the state-action space of an agent and the specific environment in which it is trained [23, 38, 40, 42]. When the underlying infrastructure changes due to hardware upgrades, network reconfigurations, or resource allocation adjustments, the state-action space may also change, rendering the agent unable to interact with the environment.

This issue, known as state-action dimension mismatch [42], is a fundamental problem in cross-domain transfer and domain adaptation. It requires redesigning and retraining agents from scratch whenever the infrastructure changes, a prohibitively expensive process in terms of time and resources.

Recent works have focused on transferring knowledge across tasks with mismatched state-action spaces. For example, [7, 38] propose methods that learn compact latent representations— via autoencoders or mutual information objectives—to disentangle task-specific details from generalizable features. Similarly, [32, 42] construct embedding spaces for policy transfer across domains. While these methods have shown promise in structured environments such as robotics (e.g., MuJoCo [36] locomotion tasks), they rely on well-defined state-action representations. It is important to note that although dynamic conditions—including fluctuating constraints and communication delays—are also present in robotics, the nature and scale of these challenges in cloud-edge infrastructures are fundamentally different. In robotics, environmental variations typically stem from controlled physical interactions and predictable task dynamics, whereas cloud-edge systems must contend with highly volatile resource availability, heterogeneous hardware configurations, and complex, multi-hop network topologies. Consequently, while cross-domain RL approaches from robotics provide valuable insights, they do not fully address the unique state-action mismatches and the extensive variability inherent in cloud-edge resource management.

Other works in cross-domain RL have explored policy representation and transfer in more heterogeneous settings. For example, [15, 44] propose techniques for adapting policies across domains, while [6, 42] extend these ideas by integrating learned state abstractions and knowledge transfer mechanisms. However, these methods often assume that source and target tasks share some underlying structure or similarity, which may not hold in practical scenarios like cloud-edge resource management. Additionally, while disentangled representation learning [13, 39] and autoencoder-based state representation learning [5, 19, 41], have been proposed to capture domain-invariant features, they primarily focus on robotics or simulation settings. These approaches do not address the unique challenges of cloud-edge infrastructures, such as fluctuating resource constraints, dynamic network topologies, and hardware heterogeneity, which significantly impact RL adaptation.

Some works have explored transfer learning in the context of cloud-edge environments but do not address state-action mismatches caused by infrastructure changes. For example, [46] proposes a framework for cloud-edge collaborative DRL, focusing on knowledge distillation between heterogeneous agents. Similarly, [33] introduces a transfer RL framework for adaptive task offloading, using domain adaptation to align heterogeneous characteristics of mobile devices. Finally, [35] proposes a hybrid cloud-edge control strategy using transfer DRL, relying on fine-tuning and domain adaptation networks. While these methods improve convergence and adaptability, they assume that state-action spaces remain consistent across tasks, limiting their applicability when infrastructure changes.

This challenge is particularly acute in modern cloud-edge computing infrastructures, which are highly dynamic and heterogeneous. Variations in datacenter configurations driven by differences in hardware, network topologies, and resource allocation policies are common, as are changes caused by equipment upgrades or failures. These variations pose significant challenges for DRL agents, as those trained on one infrastructure often struggle to generalize to others, or may become entirely incompatible. Without mechanisms for efficient adaptation, DRL approaches that do not handle cross-domain transfers and state-action space mismatches become obsolete, requiring new state-action space designs and retraining from scratch with every system change.

Contributions. To address the challenge of state-action dimension mismatch, we propose a novel framework that decouples DRL agents from domain-specific features, enabling them to operate effectively across diverse cloud-edge environments. Our approach employs state abstraction techniques to create a unified, fixed-dimension, and infrastructure-invariant representation of the state space. The discrete state features are mapped into fixed-size embeddings that capture underlying relationships in a dense, continuous space. Then, the continuous and discrete features are passed through two distinct two-layer MLP networks and then concatenated and passed through an adaptive residual layer, which further enhances generalization and adaptation during transfers. This transformation abstracts infrastructure-specific details, enabling effective knowledge transfer across different cloud-edge environments without the need for agent redesign.

Our work is orthogonal to meta-learning and continual learning approaches. Once the agent is made compatible with multiple infrastructures, our framework can be combined with meta-learning or continual learning techniques to further enhance adaptability to changes in input workloads. This integration would create DRL agents capable of generalizing not only to changes in the underlying infrastructure but also to variations in input distributions.

This work bridges the gap between DRL models and their practical deployment in dynamic, heterogeneous cloud-edge systems. Our contributions can be summarized as follows:
1) We formulate the job placement problem in multi-datacenter infrastructures as an RL task and propose a state-action space and reward design for multi-datacenter cloud-edge resource management, enabling seamless cross-domain transfer and adaptation across environments with varying scales.
2) We integrate a custom feature extractor into the internal architecture of the DRL agent, using state abstraction and adaptation techniques to decouple the agent from infrastructure-specific details. This approach enables rapid adaptation to infrastructure changes.
3) We develop a framework for job-to-datacenter placement and evaluate it through extensive simulations. We compare it against a heuristic and a baseline DRL approach, demonstrating significant improvements in terms of total reward and convergence speed.

To the best of our knowledge, no prior work addresses the challenge of state-action dimension mismatch in the cloud-edge continuum environments for resource management tasks. Existing methods are either highly theoretical and impractical for real-world deployment or tailored to specific domains like robotics or games. This challenge is particularly critical in cloud-edge environments, where infrastructure changes are common. Our work fills this gap by proposing a transfer learning framework that uses state abstraction to create an infrastructure-invariant representation, enabling seamless policy transfer across diverse environments. By decoupling the agent from domain-specific features, our framework minimizes the need for extensive retraining, ensuring robust performance across varying system configurations.

2 BACKGROUND

RL. The RL paradigm includes an *agent* and an *environment*. The agent observes the *state* s of the *environment*, interacts with it by taking *actions* a based on a learned behavior, formally called a *policy*, and receives feedback in the form of a *reward* signal r. Each action affects the environment, causing a transition to a new state. These interactions occur over discrete time steps t, which together form an *episode*. The goal of the agent is to maximize its cumulative episodic reward by discovering an optimal policy. RL has been widely applied in robotics, game playing, and autonomous decision-making, where sequential decision processes are crucial.

DRL. DRL extends standard RL by incorporating deep neural networks as function approximators to learn complex policies, enabling agents to handle large, high-dimensional state-action spaces found in modern systems. However, this comes at a cost, as training deep networks requires substantial computational resources, extensive environment interaction, and often suffers from instability during learning. The challenge of *sample inefficiency* arises, where an agent may require a large number of interactions to converge to an effective policy. Additionally, a fundamental aspect of RL is the *exploration-exploitation tradeoff*, where an agent must balance between exploring new actions to discover better strategies and exploiting known actions to maximize rewards. Poor exploration can lead to suboptimal policies, excessive training times, or convergence to local optima.

Actor-Critic Algorithms. These algorithms consist of two components: the *actor*, which determines the agent's actions, and the *critic*, which evaluates the quality of those actions by providing feedback. This is achieved using two separate neural networks—one for each component. The state serves as input to both the actor and the critic. The *policy network* (actor) maps states to a probability distribution over possible actions. It is responsible for decision-making, selecting actions based on learned policies that maximize expected rewards. The policy is continuously refined using feedback from the critic, allowing the agent to explore and exploit effectively. The *value network* (critic) estimates a scalar *state value*, which represents the expected cumulative reward from a given state. It evaluates the agent's performance and computes the *advantage*, defined as the difference between the predicted state value and the actual received reward. This advantage function helps guide policy updates, improving decision-making over time. By training both networks simultaneously, actor-critic algorithms enable more stable learning, balancing long-term planning with immediate reward feedback, making them effective for complex reinforcement learning tasks.

Proximal Policy Optimization (PPO). PPO is a widely used actor-critic algorithm that optimizes the policy using a clipped surrogate objective, which prevents excessively large updates that could destabilize training. By constraining policy updates, PPO ensures a more stable learning process while effectively balancing exploration and exploitation. These properties make PPO well-suited for complex DRL tasks that require reliable performance and sample efficiency.

Transfer RL & Cross-Domain Transfer. Transfer RL leverages pretrained agent knowledge from a *source* task to a *target* task to reduce the required interactions, significantly accelerating convergence and improving sample efficiency. In cross-domain transfers, the source task that the agent initially trained on and the target task that the agent is later deployed on have different state-action spaces. The key challenge in cross-domain transfer is enabling the agent to effectively reuse knowledge from tasks that do not share identical state-action representations, moving beyond conventional transfer learning that assumes a common structure. Various approaches in the literature aim to bridge this gap, facilitating knowledge transfer across environments with different state-action spaces.

State Abstraction and Dimensionality Reduction. In reinforcement learning, the state space often contains high-dimensional, redundant, or irrelevant information that can hinder learning and generalization. State abstraction and dimensionality reduction techniques address this challenge by transforming raw state features into a fixed, low-dimensional, and unified representation that captures the essential aspects of the environment. This abstraction process is critical for enabling domain-invariant and environment-agnostic representations, which allow agents to transfer knowledge across environments with different state-action dimensionalities. By decoupling the agent's policy from environment-specific details, state abstraction ensures that the agent can interact with and adapt to new environments efficiently, even when the underlying infrastructure or task configuration changes. For example, in cloud-edge resource management, where infrastructure configurations vary widely, a unified representation of resource states (e.g., CPU usage, memory availability) enables the agent to generalize across different datacenters without requiring extensive retraining. This approach not only improves sample efficiency but also enhances the agent's ability to handle dynamic and heterogeneous environments, making it a cornerstone of effective cross-domain transfer learning.

Embeddings in Policy Networks of DRL Agents. Embeddings (first introduced in [4], popularized in [24]) are *low-dimensional representations of high-dimensional data* that capture essential features while preserving meaningful relationships. They replace raw one-hot or categorical encodings by mapping discrete variables into *continuous vector spaces*, allowing models to generalize more effectively. In DRL, embeddings play a crucial role in policy networks by enabling agents to efficiently process large and complex state-action spaces. By learning compact, transferable representations, embeddings facilitate cross-domain policy adaptation, ensuring smoother transfer learning and better alignment of state-action spaces in heterogeneous environments. This ability to encode meaningful feature relationships is especially valuable in dynamic systems where traditional representations struggle to adapt efficiently.

Residual Connections in Policy Networks of DRL Agents. Residual connections in neural networks, introduced in [12], enable deeper architectures by allowing gradients to flow more effectively through layers, mitigating vanishing gradient issues. In DRL, residual connections in the policy network facilitate smoother policy updates by preserving useful representations while enabling adaptation to new tasks. For transfer learning, residual connections help retain previously learned knowledge while integrating new information, preventing catastrophic forgetting. When adapting to new policies across different environments, residual layers allow for *incremental modifications to the policy*, ensuring stability while enabling flexibility in handling state-action mismatches during cross-domain transfers.

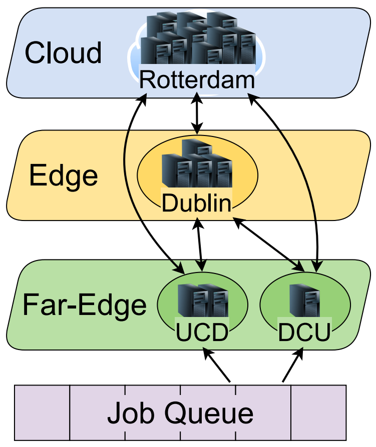

Figure 1: A multi-datacenter infrastructure consisting of one cloud, one edge, and two far-edge datacenters (Env B).

Datacenter Type	Job Latency Tolerance	QoS
Far-Edge	Tolerant	1.0
Edge	Tolerant	1.0
Cloud	Tolerant	1.0
Far-Edge	Moderate	1.0
Edge	Moderate	1.0
Cloud	Moderate	0.0
Far-edge	Critical	1.0
Edge	Critical	0.5
Cloud	Critical	0.0

Table 1: Job QoS achieved based on datacenter type and job latency tolerance.

3 PROBLEM FORMULATION & SOLUTION

System Model. We consider a hierarchical infrastructure composed of n datacenters, as illustrated in Figure 1. Datacenters are classified into three types: cloud, edge, and far-edge. For instance, far-edge (on-premise) datacenters may be located at two different universities in Dublin (e.g., UCD and DCU), an edge datacenter elsewhere in Dublin, and a cloud datacenter in Rotterdam. Each datacenter consists of multiple hosts. In our scenario, all datacenters are interconnected except for the two far-edge datacenters, which remain isolated from one another.

Cloud datacenters typically offer a larger number of hosts with more powerful machines, whereas edge and far-edge datacenters have fewer and less capable hosts but are accessible with a lower latency since they are closer to the users. Although each datacenter maintains its own job queue, we preprocess these individual queues to create an aggregated view that forms part of the RL agent's state space.

In our model, jobs arrive exclusively at far-edge locations and may be placed in the same far-edge datacenter, in an edge datacenter connected to it, or, as a last resort, in a cloud datacenter. Notably, jobs arriving at a particular far-edge (e.g., UCD) cannot be deployed to another far-edge (e.g., DCU). Each job is characterized by its CPU core requirements, execution length, arrival location (corresponding to one of the datacenters), a soft deadline, and a latency tolerance classified as tolerant, moderate, or critical.

The soft deadline defines a preferred timeframe for job placement, whereas the latency tolerance determines how far from its arrival location the job can be placed without significant degradation (e.g., closer at the far-edge, at the edge, or further away in the cloud). Consequently, job placement directly impacts QoS, as jobs processed farther from their arrival location experience higher communication delays (e.g., longer round-trip times). To clarify how these parameters impact service quality, Table 1 summarizes the expected quality-of-service (QoS) for each combination of datacenter type and job latency tolerance. The soft deadline represents the maximum allowable waiting time before placement. If the deadline is exceeded, the job is still executed, but this outcome is considered suboptimal as it results in a degradation of the job's QoS.

In our RL setup, each episode is defined by a fixed workload size. Specifically, we introduce a predefined number of jobs (e.g., 50) into the system. The episode terminates once all these jobs have been successfully executed, after which the simulation resets, and the experiment is repeated.

The primary objective of our DRL agent is to serve as a broker that maps incoming jobs to suitable datacenters for execution. The job-to-host assignment within a datacenter is determined by a heuristic that allocates the job to the host with the maximum available resources. Furthermore, jobs are executed within virtual machines (VMs). We assume that each host has an always-on VM that is available provided sufficient resources exist. The agent has full visibility of all datacenters and the current job queue. This visibility is based on a lightweight state representation, where each host is described by three numerical values and each job in the queue by four numerical values (see Figure 2). The agent does not have insight into future job arrivals.

The agent operates in a slotted manner (with one slot corresponding to one second). It receives the current infrastructure and job queue states at each slot and returns an action vector. When no jobs are pending, all actions default to no-ops. Since the number of arriving jobs is variable, we define a maximum queue length to accommodate fluctuations in workload. We need to set a maximum job queue length to define the agent's action space, which corresponds

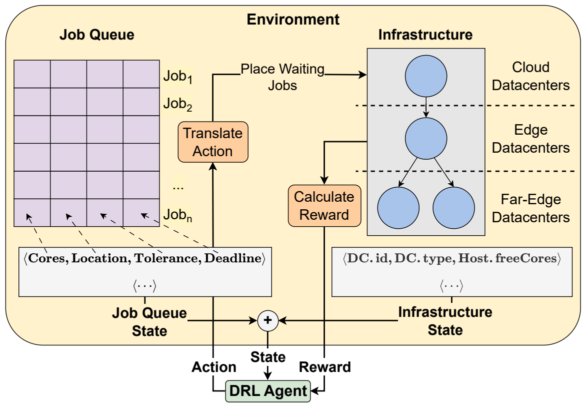

Figure 2: The DRL agent takes the concatenated states as input, selects an action, and receives a reward based on the environment's job placement response. This loop repeats until the episode ends.

to placement decisions for each job. This fixed limit is essential for the DRL agent's architecture, as it affects the output layer size of the neural network. The queue does not need to reach this maximum; if fewer jobs are present, slots are filled with zeros.

State Space. The state space consists of two subspaces. The first represents the infrastructure load, where each host in a datacenter is described by the tuple $\langle DC_{id}, DC_{type}, Host_{freeCores} \rangle$. Here, DC_{id} uniquely identifies a datacenter, DC_{type} indicates whether it is cloud, edge, or far-edge, and $\text{Host}_{\text{freeCores}}$ indicate the number of available cores at the this slot. This subspace informs the agent about the currently available computing resources. The second subspace captures the job queue, with each job represented by the tuple $\langle cores, location, tolerance, deadline \rangle$. The overall state at each timestep is formed by concatenating the infrastructure state with the job queue information, as depicted in Figure 2.

Action Space. The agent's action is expressed as a vector whose length equals the maximum number of jobs that can be queued. Each element of this vector specifies the datacenter (via its id) to which the corresponding job should be assigned. To handle DRL agent transfers across infrastructures with different numbers of datacenters, the action vector is defined based on a user-specified maximum number of datacenters. If an action is infeasible—such as assigning a job to a datacenter with insufficient free resources or a nonexistent datacenter—the environment treats that action as a no-op. Similarly, if the agent issues an action for a job that is not present (due to variable queue lengths), it is ignored.

Reward Function. Our reward function integrates multiple components to balance system performance metrics. First, to encourage prompt job placement and prevent queue congestion, we define a placement reward,

$$R_p = \frac{\text{jobsPlaced}}{\text{jobsWaiting}}.$$

Second, to account for QoS based on the datacenter type and the job's latency tolerance, we incorporate a QoS reward,

$$R_q = \frac{\text{QoS}}{\text{jobsPlaced}},$$

with quality values as detailed in Table 1. Third, to penalize deadline violations, we compute a deadline violation ratio,

$$R_d = \frac{\text{deadlineViolationsCount}}{\text{jobsWaiting}}.$$

The overall reward is given by

$$R = c_p R_p + c_q R_q - c_d R_d,$$

where c_p, c_q, and c_d are coefficients (with $0 \leq c_p, c_q, c_d \leq 1$ and $c_p + c_q + c_d = 1$) that adjust the relative importance of each component.

Policy Network Architecture. With increasing system complexity, designing DRL solutions often leads to incorporating ever more features into the state space in an effort to capture every nuance of the infrastructure. However, this expansion has two major drawbacks. First, it slows training and convergence due to the curse of dimensionality phenomenon [3]. Second, it tightly couples the agent to a specific infrastructure, forcing a complete redesign of the state space, action space, and reward function when transitioning to a new environment—even when the underlying tasks are conceptually similar. Although feature importance analysis helps identify which features contribute positively, these methods remain time-consuming and do not fully decouple the agent from environment-specific details. Moreover, new infrastructures may introduce additional features, and those previously considered important might change, further exacerbating the state dimension mismatch issue inherent in cross-domain transfers.

To address these challenges, we adopt a more efficient technique based on state abstraction. Our DRL agent incorporates a custom feature extractor within its policy network that transforms the raw state into a compressed, fixed-dimension latent representation as shown in Figure 3. This unified latent representation is infrastructure-agnostic, enabling the agent not only to learn effectively in a single environment but also to retain and transfer knowledge to similar environments.

The state abstraction process operates as follows. First, all features in the raw observation are padded with zeros up to their maximum user-defined values. Within the policy network, continuous and categorical features are processed separately. Categorical features, typically represented via one-hot encoding, are passed through an embedding layer to

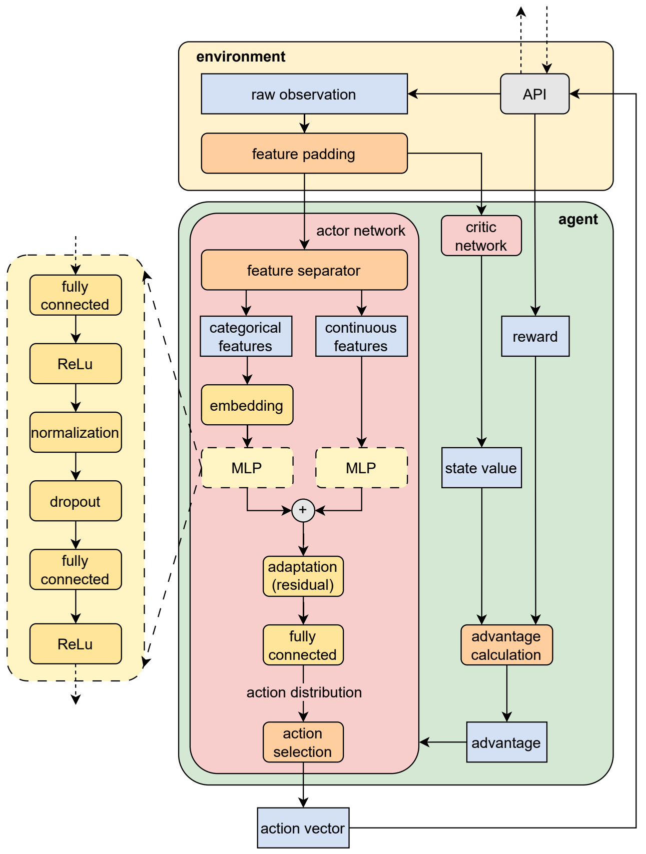

Figure 3: DRL agent architecture for handling state-action mismatches in transfers between different environments.

compactly capture their underlying relationships. Each feature, whether continuous or categorical, is then processed by its own dedicated MLP with an identical architecture. These MLPs use ReLU activation functions and dropout for regularization. Their outputs are then concatenated and passed through an adaptation layer—implemented as a residual connection—before action selection. This layer allows the agent to preserve previously learned knowledge while adapting to new information, mitigating catastrophic forgetting. By converting raw observations into a unified, domain-invariant representation, the agent can transfer knowledge across diverse cloud-edge environments without requiring extensive retraining. Finally, all neural network weights are initialized using Xavier initialization [11], with biases set to zero, ensuring efficient learning and stable convergence.

4 PERFORMANCE EVALUATION

Experimental Setup. We evaluate our approach using the CloudSim Plus simulator [34], and a custom DRL environment built on the Gymnasium API [37]. To bridge the Java-based CloudSim Plus with Python DRL agents, we used the Py4J gateway [10], adding the functionality to dynamically support job-to-datacenter placement decisions.

For the calculation of rewards, we assign equal weights to placement, quality of service, and deadline compliance metrics (coefficients $c_p = c_q = c_d = 0.33$). The experiments used the PPO algorithm from the stable-baselines3 (SB3) library [27], with two variants: *(i)* the baseline *PPO-Base* (vanilla SB3 implementation), and *(ii)* our extended *PPO-X*, which integrates a custom feature extractor (see Figure 3) to enhance adaptability to infrastructure changes. Training was conducted on NVIDIA RTX 4080/4090 GPUs with Intel i7-14700KF/i9-14900KF CPUs over 600k timesteps (about 5 to 7 hours per run), with a simulation timestep of 1 second. Each experiment was repeated using five different random seeds, with the final results averaged across these runs.

Test Scenarios. We compared our *PPO-X* algorithm against two baselines, a heuristic and a vanilla DRL algorithm:

(1) **Heuristic**: Prioritizes jobs according to their deadline and criticality, placing them in the closest available datacenter. To enhance its effectiveness, we intentionally designed it to allow multiple placement attempts per job, unlike DRL agents, which make a single decision per step. This gives the heuristic an advantage, as repeated placement attempts increase the chances of finding available resources.

(2) **PPO-Base**: The vanilla PPO provided by the SB3 library.

Note that observations are always zero-padded before being passed to the agent, ensuring a consistent state dimension and allowing even the vanilla PPO algorithm to handle them without mismatches.

The algorithms were trained in the topology of Figure 1. We created two additional environments for transfer learning experiments. The environment *A* removes the cloud datacenter from *B* to test the adaptability of the agent to a reduced infrastructure. The environment *C* extends *B* with two new datacenters: an edge datacenter in Copenhagen and a far-edge datacenter at the AAU CPH university, both connected to the Rotterman cloud datacenter. The jobs in *C* arrive at three far-edge locations instead of two in *B*, and the job trace is different, with unseen job descriptions. The datacenter characteristics are as follows: 16 hosts with 64 CPU cores each for the cloud datacenter; 8 hosts with 16 CPU cores each for the edge datacenters; 2-3 hosts with 6 CPU cores each for the far-edge datacenters. All cores have the same processing speed. Synthetic datasets were generated with job durations uniformly distributed between $3 - 5$ seconds, requested CPU cores ranging from 1 to 20, soft deadlines between $0 - 5$ seconds, and arrival rate of $1 - 4$ jobs per timestep, totaling 50 jobs per episode.

Observations. In Figure 4, we see that the converged rewards of both PPO variants (at 600k steps) exceed the heuristic's performance in all three environments. We also observe

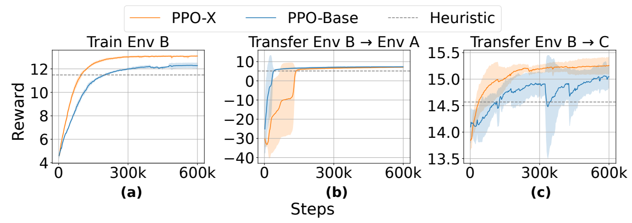

Figure 4: Reward comparison of our method vs. baselines.

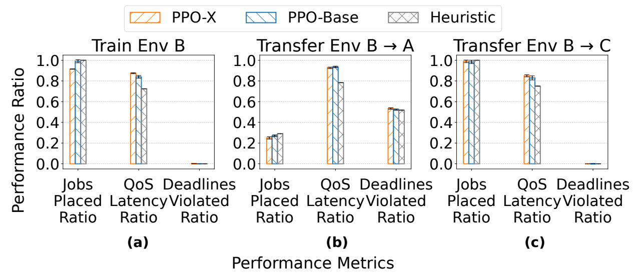

Figure 5: Performance metrics: our method vs. baselines.

that during training, our PPO extension consistently outperforms the standard PPO algorithm (Figure 4a). However, it is important to note that our extension is not always necessary and may converge more slowly than the simpler version of PPO (Figure 4b). For example, in environments such as *A*, where the state-action space is relatively small and the optimal solution is easier to identify, our extension may not provide significant benefits. Due to the larger model size, adaptation can be slower as more parameters need to be updated. In such cases, it is crucial to evaluate whether the added complexity of our extension justifies its use over the simpler PPO version. At the same time, in Figure 5, we see how total rewards translate into practical system performance. Specifically, the jobs placed and QoS latency ratios should be maximized, while the deadlines violated ratio should be minimized. We observe that the performance metrics in environment *A* are actually worse compared to environment *B*, which may seem counterintuitive given that environment *A* is less complex and thus presumably easier to solve. The reason for the bad performance is that while environment *A* has a much smaller state-action space, making it computationally simpler to solve, it also has far fewer hosts available. As a result, the workload pattern becomes highly stressful, leading to worse job placement and deadline violation ratios compared to the environment *B*, which is more resource rich.

The advantage of our extension is evident in Figure 4c, where the environment presents greater challenges compared to what the agent encountered during training. To better understand the impact of workload variability, we designed a test with a different workload trace, evaluating how both PPO architectures respond to sudden and critical changes. As shown in Figure 4c, our extension not only converges to a superior overall solution, but also exhibits greater stability, especially in the second half of the run, whereas the *PPO-base* algorithm remains highly unstable, as indicated by the shaded regions representing variability across different seeds. This stability suggests that *PPO-X* is better suited for environments where conditions fluctuate significantly, making it a more robust choice.

5 CONCLUSIONS & FUTURE WORK

We present a novel approach to enhance the adaptability and transferability of DRL agents in dynamic cloud-edge environments. Our architecture minimizes infrastructure-specific dependencies, enabling cross-domain transfers. By learning infrastructure-agnostic state representations, DRL agents can generalize effectively despite resource availability or workload pattern changes. This reduces the need for extensive retraining, simplifying real-world deployment.

Moving forward, we aim to evaluate our approach in real-world cloud infrastructures, focusing on performance, cost efficiency, and energy savings. A key goal is quantifying the benefits of reusing agent knowledge across different infrastructures. We also plan to explore multi-agent and hierarchical RL, where specialized agents manage datacenter selection, host allocation, VM scaling, and job migration, enabling scalable, decentralized decision-making. To enhance resilience, we will investigate drift detection and adaptation by introducing host failures or workload surges and ensuring agents can detect and quickly respond to drift. Additionally, online learning and adaptive reward shaping will refine policies in real time, improving stability and convergence. Another direction is integrating uncertainty-aware RL and contrastive learning to improve decision-making in unpredictable environments. Continual and meta-learning could enhance long-term adaptability while preventing catastrophic forgetting. Finally, using graph neural networks in the policy network could better capture relationships between cloud-edge resources, improving robustness in large-scale deployments.

ACKNOWLEDGEMENTS

This work has been supported by (i) the Horizon Europe research and innovation program of the European Union, MLSysOps (grant agreement number 101092912), and (ii) partly supported by the CLEVER project (grant agreement number 101097560). CLEVER is supported by the EU Chips Joint Undertaking (Chips JU) and its members (including top-up funding by the Innovation Fund Denmark (IFD)).

REFERENCES

[1] Ala I. Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. 2015. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Commun. Surv. Tutorials* 17, 4 (2015), 2347–2376. https://doi.org/10.1109/COMST.2015.2444095

[2] Theodoros Aslanidis, Andreas Chouliaras, and Dimitris Chatzopoulos. 2023. Reinforcement Learning Techniques for Optimizing System Configuration on the Cloud: A Taxonomy and Open Problems. In *Proceedings of the 2023 International Conference on embedded Wireless Systems and Networks, EWSN 2023, Rende, Italy, September 25-27, 2023*, Giancarlo Fortino, Valeria Loscrì, Fabrizio Granelli, Tarek F. Abdelzaher, Antonella Guzzo, and Claudio Savaglio (Eds.). ACM, 345–350. https://doi.org/10.5555/3639940.3639995

[3] Richard Bellman. 1957. *Dynamic Programming*. Dover Publications.

[4] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A Neural Probabilistic Language Model. *J. Mach. Learn. Res.* 3 (2003), 1137–1155. https://jmlr.org/papers/v3/bengio03a.html

[5] Nicolò Botteghi, Mannes Poel, and Christoph Brune. 2022. Unsupervised Representation Learning in Deep Reinforcement Learning: A Review. https://doi.org/10.48550/ARXIV.2208.14226 arXiv:2208.14226

[6] Haitham Bou-Ammar, Eric Eaton, Paul Ruvolo, and Matthew E. Taylor. 2015. Unsupervised Cross-Domain Transfer in Policy Gradient Reinforcement Learning via Manifold Alignment. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, Blai Bonet and Sven Koenig (Eds.). AAAI Press, 2504–2510. https://doi.org/10.1609/AAAI.V29I1.9631

[7] Yu Chen, Yingfeng Chen, Yu Yang, Ying Li, Jianwei Yin, and Changjie Fan. 2019. Learning Action-Transferable Policy with Action Embedding. *CoRR* abs/1909.02291 (2019). arXiv:1909.02291 http://arxiv.org/abs/1909.02291

[8] Mieszko Ferens, Diego Hortelano, Ignacio De Miguel, Ramón J Durán Barroso, and Sokol Kosta. 2024. STEROCEN: Simulation and Training Environment for Resource Orchestration in Cloud-Edge Networks. In *2024 15th International Conference on Network of the Future (NoF)*. IEEE, 133–141.

[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017 (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 1126–1135. http://proceedings.mlr.press/v70/finn17a.html

[10] Włodzimierz Funika, Paweł Koperek, and Jacek Kitowski. 2018. Repeatable experiments in the cloud resources management domain with use of reinforcement learning. In *Cracow Grid Workshop*. 31–32.

[11] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 249–256.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 770–778. https://doi.org/10.1109/CVPR.2016.90

[13] Irina Higgins, Arka Pal, Andrei A. Rusu, Loïc Matthey, Christopher P. Burgess, Alexander Pritzel, Matthew M. Botvinick, Charles Blundell, and Alexander Lerchner. 2017. DARLA: Improving Zero-Shot Transfer in Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017 (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 1480–1490. http://proceedings.mlr.press/v70/higgins17a.html

[14] Menglan Hu, Hao Wang, Xiaohui Xu, Jianwen He, Yi Hu, Tianping Deng, and Kai Peng. 2024. Joint Optimization of Microservice Deployment and Routing in Edge via Multi-Objective Deep Reinforcement Learning. *IEEE Trans. Netw. Serv. Manag.* 21, 6 (2024), 6364–6381. https://doi.org/10.1109/TNSM.2024.3443872

[15] Girish Joshi and Girish Chowdhary. 2018. Cross-Domain Transfer in Reinforcement Learning Using Target Apprentice. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*. IEEE, 7525–7532. https://doi.org/10.1109/ICRA.2018.8462977

[16] Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. 2022. Towards Continual Reinforcement Learning: A Review and Perspectives. *J. Artif. Intell. Res.* 75 (2022), 1401–1476. https://doi.org/10.1613/JAIR.1.13673

[17] Boonhatai Kruekaew and Warangkhana Kimpan. 2022. Multi-Objective Task Scheduling Optimization for Load Balancing in Cloud Computing Environment Using Hybrid Artificial Bee Colony Algorithm With Reinforcement Learning. *IEEE Access* 10 (2022), 17803–17818. https://doi.org/10.1109/ACCESS.2022.3149955

[18] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory G. Slabaugh, and Tinne Tuytelaars. 2022. A Continual Learning Survey: Defying Forgetting in Classification Tasks. *IEEE Trans. Pattern Anal. Mach. Intell.* 44, 7 (2022), 3366–3385. https://doi.org/10.1109/TPAMI.2021.3057446

[19] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. 2020. CURL: Contrastive Unsupervised Representations for Reinforcement Learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 5639–5650. http://proceedings.mlr.press/v119/laskin20a.html

[20] Jeng-Lin Li, Chih-Fan Hsu, Ming-Ching Chang, and Wei-Chao Chen. 2024. A Comprehensive Review of Machine Learning Advances on Data Change: A Cross-Field Perspective. *CoRR* abs/2402.12627 (2024). https://doi.org/10.48550/ARXIV.2402.12627 arXiv:2402.12627

[21] Ning Liu, Zhe Li, Jielong Xu, Zhiyuan Xu, Sheng Lin, Qinru Qiu, Jian Tang, and Yanzhi Wang. 2017. A Hierarchical Framework of Cloud Resource Allocation and Power Management Using Deep Reinforcement Learning. In *37th IEEE International Conference on Distributed Computing Systems, ICDCS 2017, Atlanta, GA, USA, June 5-8, 2017*, Kisung Lee and Ling Liu (Eds.). IEEE Computer Society, 372–382. https://doi.org/10.1109/ICDCS.2017.123

[22] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, João Gama, and Guangquan Zhang. 2019. Learning under Concept Drift: A Review. *IEEE Trans. Knowl. Data Eng.* 31, 12 (2019), 2346–2363. https://doi.org/10.1109/TKDE.2018.2876857

[23] Jiafei Lyu, Chenjia Bai, Jingwen Yang, Zongqing Lu, and Xiu Li. 2024. Cross-Domain Policy Adaptation by Capturing Representation Mismatch. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net. https://openreview.net/forum?id=3uPSQmjXzd

[24] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1301.3781

[25] German Ignacio Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. 2019. Continual lifelong learning with neural networks: A review. *Neural Networks* 113 (2019), 54–71. https://doi.org/10.1016/J.NEUNET.2019.01.012

[26] Haoran Qiu, Weichao Mao, Chen Wang, Hubertus Franke, Alaa Youssef, Zbigniew T. Kalbarczyk, Tamer Basar, and Ravishankar K. Iyer. 2023.

AWARE: Automate Workload Autoscaling with Reinforcement Learning in Production Cloud Systems. In *Proceedings of the 2023 USENIX Annual Technical Conference, USENIX ATC 2023, Boston, MA, USA, July 10-12, 2023*, Julia Lawall and Dan Williams (Eds.). USENIX Association, 387–402. https://www.usenix.org/conference/atc23/presentation/qiu-haoran

[27] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. 2021. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research* 22, 268 (2021), 1–8. http://jmlr.org/papers/v22/20-1364.html

[28] Yongyi Ran, Han Hu, Xin Zhou, and Yonggang Wen. 2019. DeepEE: Joint Optimization of Job Scheduling and Cooling Control for Data Center Energy Efficiency Using Deep Reinforcement Learning. In *39th IEEE International Conference on Distributed Computing Systems, ICDCS 2019, Dallas, TX, USA, July 7-10, 2019*. IEEE, 645–655. https://doi.org/10.1109/ICDCS.2019.00070

[29] Bhaskar Prasad Rimal, Eunmi Choi, and Ian Lumb. 2009. A Taxonomy and Survey of Cloud Computing Systems. In *International Conference on Networked Computing and Advanced Information Management, NCM 2009, Fifth International Joint Conference on INC, IMS and IDC: INC 2009: International Conference on Networked Computing, IMS 2009: International Conference on Advanced Information Management and Service, IDC 2009: International Conference on Digital Content, Multimedia Technology and its Applications, Seoul, Korea, August 25-27, 2009*, Jinhwa Kim, Dursun Delen, Jinsoo Park, Franz Ko, Chen Rui, Jong Hyung Lee, Jian Wang, and Gang Kou (Eds.). IEEE Computer Society, 44–51. https://doi.org/10.1109/NCM.2009.218

[30] Daniel Rosendo, Alexandru Costan, Patrick Valduriez, and Gabriel Antoniu. 2022. Distributed intelligence on the Edge-to-Cloud Continuum: A systematic literature review. *J. Parallel Distributed Comput.* 166 (2022), 71–94. https://doi.org/10.1016/J.JPDC.2022.04.004

[31] Mahadev Satyanarayanan. 2017. The Emergence of Edge Computing. *Computer* 50, 1 (2017), 30–39. https://doi.org/10.1109/MC.2017.9

[32] Sergio A. Serrano, José Martínez-Carranza, and Luis Enrique Sucar. 2023. Similarity-based Knowledge Transfer for Cross-Domain Reinforcement Learning. *CoRR* abs/2312.03764 (2023). https://doi.org/10.48550/ARXIV.2312.03764 arXiv:2312.03764

[33] Kefan Shuai, Yiming Miao, Kai Hwang, and Zhengdao Li. 2023. Transfer Reinforcement Learning for Adaptive Task Offloading Over Distributed Edge Clouds. *IEEE Trans. Cloud Comput.* 11, 2 (2023), 2175–2187. https://doi.org/10.1109/TCC.2022.3192560

[34] Manoel C Silva Filho, Raysa L Oliveira, Claudio C Monteiro, Pedro RM Inácio, and Mário M Freire. 2017. CloudSim plus: a cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness. In *2017 IFIP/IEEE symposium on integrated network and service management (IM)*. IEEE, 400–406.

[35] Yuechuan Tao, Jing Qiu, and Shuying Lai. 2022. A Hybrid Cloud and Edge Control Strategy for Demand Responses Using Deep Reinforcement Learning and Transfer Learning. *IEEE Trans. Cloud Comput.* 10, 1 (2022), 56–71. https://doi.org/10.1109/TCC.2021.3117580

[36] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*. IEEE, 5026–5033. https://doi.org/10.1109/IROS.2012.6386109

[37] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. 2024. Gymnasium: A Standard Interface for Reinforcement Learning Environments. *arXiv preprint arXiv:2407.17032* (2024).

[38] Michael Wan, Tanmay Gangwani, and Jian Peng. 2020. Mutual Information Based Knowledge Transfer Under State-Action Dimension Mismatch. In *Proceedings of the Thirty-Sixth Conference on Uncertainty in Artificial Intelligence, UAI 2020, virtual online, August 3-6, 2020 (Proceedings of Machine Learning Research, Vol. 124)*, Ryan P. Adams and Vibhav Gogate (Eds.). AUAI Press, 1218–1227. http://proceedings.mlr.press/v124/wan20a.html

[39] Xin Wang, Hong Chen, Yuwei Zhou, Jianxin Ma, and Wenwu Zhu. 2023. Disentangled Representation Learning for Recommendation. *IEEE Trans. Pattern Anal. Mach. Intell.* 45, 1 (2023), 408–424. https://doi.org/10.1109/TPAMI.2022.3153112

[40] Xiaoyu Wen, Chenjia Bai, Kang Xu, Xudong Yu, Yang Zhang, Xuelong Li, and Zhen Wang. 2024. Contrastive Representation for Data Filtering in Cross-Domain Offline Reinforcement Learning. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net. https://openreview.net/forum?id=rReWhol66R

[41] Jinwei Xing, Takashi Nagata, Kexin Chen, Xinyun Zou, Emre Neftci, and Jeffrey L. Krichmar. 2021. Domain Adaptation In Reinforcement Learning Via Latent Unified State Representation. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 10452–10459. https://doi.org/10.1609/AAAI.V35I12.17251

[42] Tianpei Yang, Heng You, Jianye Hao, Yan Zheng, and Matthew E. Taylor. 2024. A Transfer Approach Using Graph Neural Networks in Deep Reinforcement Learning. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (Eds.). AAAI Press, 16352–16360. https://doi.org/10.1609/AAAI.V38I15.29571

[43] Zhe Yang, Phuong Nguyen, Haiming Jin, and Klara Nahrstedt. 2019. MIRAS: Model-based Reinforcement Learning for Microservice Resource Allocation over Scientific Workflows. In *39th IEEE International Conference on Distributed Computing Systems, ICDCS 2019, Dallas, TX, USA, July 7-10, 2019*. IEEE, 122–132. https://doi.org/10.1109/ICDCS.2019.00021

[44] Heng You, Tianpei Yang, Yan Zheng, Jianye Hao, and Matthew E. Taylor. 2022. Cross-domain adaptive transfer reinforcement learning based on state-action correspondence. In *Uncertainty in Artificial Intelligence, Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence, UAI 2022, 1-5 August 2022, Eindhoven, The Netherlands (Proceedings of Machine Learning Research, Vol. 180)*, James Cussens and Kun Zhang (Eds.). PMLR, 2299–2309. https://proceedings.mlr.press/v180/you22a.html

[45] Jing Zeng, Ding Ding, Kaixuan Kang, Huamao Xie, and Qian Yin. 2022. Adaptive DRL-Based Virtual Machine Consolidation in Energy-Efficient Cloud Data Center. *IEEE Trans. Parallel Distributed Syst.* 33, 11 (2022), 2991–3002. https://doi.org/10.1109/TPDS.2022.3147851

[46] Tianyu Zeng, Xiaoxi Zhang, Jingpu Duan, Chao Yu, Chuan Wu, and Xu Chen. 2024. An Offline-Transfer-Online Framework for Cloud-Edge Collaborative Distributed Reinforcement Learning. *IEEE Trans. Parallel Distributed Syst.* 35, 5 (2024), 720–731. https://doi.org/10.1109/TPDS.2024.3360438

[47] Yutong Zhang, Boya Di, Zijie Zheng, Jinlong Lin, and Lingyang Song. 2021. Distributed Multi-Cloud Multi-Access Edge Computing by Multi-Agent Reinforcement Learning. *IEEE Trans. Wirel. Commun.* 20, 4 (2021), 2565–2578. https://doi.org/10.1109/TWC.2020.3043038