# Reinforcement Learning With Adaptive Policy Gradient Transfer Across Heterogeneous Problems

Gengzhi Zhang, Liang Feng, Yu Wang, Min Li, Hong Xie, *Member, IEEE*, and Kay Chen Tan, *Fellow, IEEE*

***Abstract*—To date, transfer learning (TL) has been successfully applied for enhancing the learning performance of reinforcement learning (RL), and many transfer RL (TRL) approaches have been proposed in the literature. However, most of the existing TRL approaches consider knowledge transfer between RL tasks sharing the same state-action space. These methods thus may fail in cases where the RL tasks available for conducting knowledge transfer possess heterogeneous state-action spaces, which is common in many real-world applications. TRL across heterogeneous problem domains is challenging since the differences lie in the state-action spaces of the RL tasks are natural barriers in the knowledge transfer across tasks. This becomes more difficult if multiple heterogeneous source tasks are available when conducting knowledge transfer for a target RL task, as we have to identify the appropriate source task adaptively before performing knowledge transfer towards enhanced RL performance. In this article, we propose a new TRL algorithm with adaptive policy gradient transfer for the cases having multiple heterogeneous source RL tasks. The core ingredients of the proposed algorithm contain a *source task selection module* to select an appropriate task from a set of heterogeneous source tasks and a *knowledge transfer module* for conducting knowledge transfer across heterogeneous RL tasks. To investigate the performance of the proposed algorithm, we have conducted comprehensive empirical studies based on the well-known continuous robotic RL task with heterogeneous settings in the number of robot arms (links). The obtained results show that the proposed algorithm is effective and efficient in conducting knowledge transfer across heterogeneous problems for enhanced RL performance, over both the RL algorithm having no knowledge transfer in the learning process and the existing state-of-the-art TRL method.**

***Index Terms*—Reinforcement learning, knowledge transfer, cross-domain transfer.**

Manuscript received 29 May 2023; revised 7 December 2023; accepted 21 December 2023. Date of publication 26 February 2024; date of current version 27 May 2024. This work was supported in part by the National Key R&D Program of China under Grant 2022YFC3801700, in part by the China Postdoctoral Science Foundation under Grant 2022M710517, in part by the Joint Project JD User Growth Engine under Grant H20211431, in part by the National Natural Science Foundation of China (NSFC) under Grant U21A20512, in part by the Research Grants Council of the Hong Kong SAR under Grant PolyU11211521, Grant PolyU15218622, and Grant PolyU15215623, and in part by the The Hong Kong Polytechnic University under Grants P0039734 and P0035379. *(Corresponding author: Liang Feng.)*

Gengzhi Zhang, Liang Feng, and Hong Xie are with the College of Computer Science, Chongqing University, Chongqing 400044, China (e-mail: gzzhang@cqu.edu.cn; liangf@cqu.edu.cn; xiehong2018@foxmail.com).

Yu Wang and Min Li are with the Department of User Growth and Operations, Jing Dong Retail, Beijing 100176, China (e-mail: wangyu1393@jd.com; limin606@jd.com).

Kay Chen Tan is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong SAR (e-mail: kctan@polyu.edu.hk).

Recommended for acceptance by Z. Zhu.

Digital Object Identifier 10.1109/TETCI.2024.3361860

## I. Introduction

REINFORCEMENT learning (RL) is commonly applied for solving the sequential decision-making problems (SDPs), in which an RL agent learns optimal behaviors for the detected states by interacting with the environment through a trial-error manner [1], [2], [3]. Recent advances of RL have demonstrated its great success in solving complex sequential decision-making tasks, such as robotic control [4], [5], complex games [6], [7], [8], autonomous driving [9], [10], and combinatorial optimization [11], [12]. However, despite the success enjoyed by the RL algorithms, it is worth noting that traditional RL method often learns new problems from scratch or zero knowledge state, which requires a lot of interactions with the environment, and is thus very slow in problem-solving. To date, as problems seldom exist in isolation [13], [14], [15], to improve the learning efficiency of RL, transfer RL (TRL) which contends to leverage the learning experiences from solved RL tasks (source tasks) to enhance the RL performance in an unseen task (target task), has been proposed in the literature. Over the years, many TRL algorithms have been proposed and successfully applied to improve the learning efficiency of RL in different practical applications [8], [16], [17], [18], [19].

Based on a recent survey [19], the existing TRL algorithms can be categorized as homogeneous TRL and heterogeneous TRL based on the differences of the state-action spaces between the source and target RL tasks. In particular, homogeneous TRL considers the scenarios that the source and the target tasks share the same state-action space, or the RL tasks only differ in the reward distributions [20], so that the learned knowledge in a RL task can be directly transferred and reused in another RL task, in terms of policy and demonstration. For instance, Fernández et al. [21] proposed a policy reuse method, which improves a RL agent by reusing past learned policy as a probabilistic bias in the learning of the target task. Hester et al. [22] proposed to transfer the demonstrations from the source agents to the target agent to guide the exploration of the target policy and thus accelerate the learning process. Other approaches also applied many TL techniques to achieve knowledge transfer in homogeneous TRL scenarios, such as policy distillation and representation sharing. For instance, Czarnecki et al. [23] used policy distillation to transfer the learned policies, in which the target agent is learned by matching the probability distribution of the policy provided by the source agent using Kullback-Leibler (KL) divergence. Andreas et al. [24] proposed to use modular network to reuse

representation in target task by decomposing the policy network into a task-specific module and agent-specific module. More reviews of homogeneous TRL can be referred to [16], [19].

Moreover, the heterogeneous TRL considers knowledge transfer across RL tasks which are with heterogeneous state-action spaces (and even different reward functions). In this case, knowledge learned in the source tasks such as policies and optimal trajectories, cannot be directly transferred to the target task. In the literature, in contrast to homogeneous TRL, only a few studies have been conducted for heterogeneous TRL. In particular, Taylor et al. [25] defined the rules of transfer manually to enable the knowledge transfer between agents which have different state vectors and actions. Moreover, to automate the knowledge transfer process, Ammar et al. [26] proposed an unsupervised manifold alignment approach to learn the inter-task mappings of state spaces autonomously, and then transferred samples across heterogeneous RL tasks via the learned mappings. Gupta et al. [27] tried to learn the invariant feature spaces morphologically between different RL agents by training them on common auxiliary tasks, and then transfer knowledge across two heterogeneous RL agents through the learned invariant feature spaces. However, despite the success obtained by these methods in conducting knowledge transfer across heterogeneous RL tasks, it is worth noting that these methods rely on the learning of inter-task mapping or common representation of the encountered two RL tasks. This will become impractical in cases having multiple heterogeneous RL tasks, since the number of required inter-task mappings or common representations increases dramatically with the number of RL tasks. To our knowledge, there is no study considering TRL with multiple heterogeneous tasks, which possess different state-action spaces and reward functions, in the literature so far, and this article thus presents an attempt to fill this gap.

In this article, we present a study on heterogeneous TRL with adaptive policy gradient transfer considering the cases of having multiple source tasks with heterogeneous state-action spaces and different reward functions. The core ingredients of the proposed algorithm contain a *source task selection module* to select an appropriate task from multiple heterogeneous source tasks and a *knowledge transfer module* for conducting knowledge transfer from the selected source task to the target task. Considering the differences lie in the state-action spaces and reward functions across RL tasks, the proposed *source task selection module* computes the similarity based on the local feature of the value function distributions between every source task to the target task to select the appropriate task for knowledge transfer. As RL in a target task often has no prior knowledge about the environment, the proposed *source task selection module* performs online source task selection and adjust the selection bias during the learning of the target task. Moreover, the *knowledge transfer module* considers the advantage value as knowledge to be transferred across heterogeneous RL tasks. As advantage value can be utilized easily in heterogeneous target task without requirement of learning inter-task mapping between the source and target RL tasks, the proposed *knowledge transfer module* would not bring much extra computational cost to the learning process in the target task and thus is capable of being applied in multi-source scenarios. To demonstrate the efficacy of the proposed algorithm, comprehensive empirical studies have been conducted on the continuous robotic control tasks with heterogeneous RL settings, which are built on the MuJoCo platform[1]. The RL tasks are differing with respect to the state space, the action space, the environment dynamics and the reward functions.

The rest of this article is organized as follows: Section II gives the background of RL. The related works are also briefly reviewed therein. Section III presents the details of our proposed method, which include the source task selection and knowledge transfer across RL tasks, considering multiple heterogeneous source tasks are available. Section IV discusses the empirical studies conducted on the MuJoCo platform, to evaluate the proposed TRL algorithm. Finally, Section V concludes this article with a few remarks.

## II. Preliminaries

This section first introduces the background of RL. Next, we present a brief review of the existing TRL algorithms in the literature.

### A. Reinforcement Learning

Generally, the RL process can be defined as a discounted Markov decision process (MDP) [30], which is often represented by $\mathcal{M} =< \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \rho_0, \gamma >$, where $\mathcal{S}$ gives the set of states that could be detected by the RL agent from the environment, $\mathcal{A}$ denotes the set of actions available for the RL agent, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the state transition function of the RL agent that maps a state-action pair to a probability distribution over the states of the agent, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function that gives the reward received by the RL agent from the environment, $\rho_0$ denotes the distribution of the initial state $s_0$ of the RL agent, and $\gamma \in (0, 1)$ is a discount factor that is used to obtain the discounted long-term objective. Based on this, the learning process of a RL algorithm can be summarized as: a RL agent starts by interacting with the environment based on the behavior policy $\pi$ which maps a state to a probability distribution over the actions of the agent; the RL agent observes the current state $s_t$ of the environment and performs an action $a_t$ predicted by $\pi$; it then observes the next state $s_{t+1}$ and receives the reward $r(s_t, a_t)$ which is given by the environment after performing action $a_t$; this process will be proceeded iteratively until a certain stopping criteria are satisfied. In RL, the sequences of the states and actions is often denoted as trajectory $\tau$, which is with length $H$. The ultimate goal of a RL agent is thus to learn the optimal behavior policy $\pi$, which maximizes the expected discounted reward denoted by $J(\pi)$, that is defined as:

$$J(\pi) = \mathbb{E}_{\tau \sim p_\pi(\tau)} \left[ \sum_{t=0}^{H} \gamma^t r(s_t, a_t) \right], \tag{1}$$

[1]A common empirical study platform for RL [28], [29], which is available at: https://www.roboti.us/.

where $p_\pi$ is the trajectory distribution for a given MDP $\mathcal{M}$ and policy $\pi$, which is given by:

$$p_\pi(\tau) = \rho_0(s_0) \prod_{t=0}^{H} \pi(a_t|s_t)\mathcal{T}(s_{t+1}|s_t, a_t) \tag{2}$$

To maximize $J(\pi)$ in (1), according to [31], existing approaches in the literature can be generally categorized as value-based methods and policy search-based methods, according to the way of finding the optimized policy $\pi$. In particular, value-based methods build a value function model to estimate the expected reward of being in a given state. Let $V^\pi(s_t)$ be the value function, it can be represented by the expected returns when starting at the state $s_t$ following policy $\pi$, which is given by:

$$V^\pi(s_t) = \mathbb{E}_{\tau\sim p_\pi(\tau|s_t)}\left[\sum_{t'=t}^{H} \gamma^{t'-t} r(s_t, a_t)\right] \tag{3}$$

Moreover, the state-action value function $Q^\pi(s_t, a_t)$ is also a common value function used in value-based methods, which is similar to $V^\pi(s_t)$ except that the action $a_t$ is included in the estimation of the expected reward, which is defined as:

$$Q^\pi(s_t, a_t) = \mathbb{E}_{\tau\sim p_\pi(\tau|s_t,a_t)}\left[\sum_{t'=t}^{H} \gamma^{t'-t} r(s_t, a_t)\right] \tag{4}$$

where the optimal policy $\pi^*$ can then be recovered from the accurate value function $V^*(s_{t+1})$ or $Q^*(s_t, a_t)$ by choosing action at every given state to maximize the value function.

Besides the absolute value function introduced above, the advantage function $A^\pi(s_t, a_t)$ is another value function often considered in RL, which gives the relative state-action values, that is defined by:

$$A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t) \tag{5}$$

Intuitively, an advantage function gives the success of the current action against the current state, rather than learning the actual value of a state-action pair. As the learning of the relative value from a state to the next state is often easier than the learning of the actual value, the idea of using advantage function has been proposed in many recent deep RL algorithms [32], [33], [34].

Next, in contrast to the value-based RL methods, the policy search-based RL algorithms represent the policy $\pi$ as a parameterized model, which aims to search for the optimal policy to maximize the expected reward in (1) via either gradient-free or gradient-based methods. As gradient-based method is more sample-efficient when the policy possesses a large number of parameters, policy gradient methods are the workhorse of policy search-based RL methods in the literature. Let $\theta$ denote the weight of a parameterized policy, the parameterized policy $\pi_\theta$ in a policy gradient method is often optimized by the gradient ascent, which can be expressed as:

$$\nabla_\theta L(\pi_\theta) = \mathbb{E}_{\tau\sim p_{\pi_\theta}(\tau)}\left[\sum_{t=0}^{H} \nabla_\theta \log \pi_\theta(a_t|s_t) J(\pi_\theta)\right] \tag{6}$$

where $J(\pi_\theta)$ is the expected discounted reward in (1), and the gradient update is weighted by the empirical return.

Furthermore, as the calculation of gradient is relied on the empirical return of the trajectory of the states, which may have a high variance, the computed gradient is thus with a high variance, and could deteriorate the performance of the policy gradient. To address this, a common practice is to make the empirical return subtract a baseline $b(\pi_\theta)$ [35] which is often the averaged value of several trajectories starting from the current state: $\hat{J}(\pi_\theta) = J(\pi_\theta) - b(\pi_\theta)$. Therefore, the function $\hat{J}(\pi_\theta)$ becomes an advantage value rather than a return value $J(\pi_\theta)$.

Moreover, a learned value function (i.e., actor-critic [34]) has also been proposed in the literature, to estimate the empirical return, which can provide a better estimation for policy gradient update. In actor-critic, the actor learns the policy, while the critic estimates the value function $V^\pi$. In particular, given a $V^\pi$ by the critic, the temporal difference (TD) error $\delta^\pi$ is often employed as an unbiased estimation of the advantage function to update the policy [36], which is given by:

$$\delta^\pi(s_t, a_t) = r(s_t, a_t) + \gamma V^\pi(s_{t+1}) - V^\pi(s_t) \tag{7}$$

In this way, the gradient of the policy $\pi_\theta$ is given by:

$$\nabla_\theta L(\pi_\theta) = \mathbb{E}_{\tau\sim p_{\pi_\theta}(\tau)}\left[\sum_{t=0}^{H} \nabla_\theta \log \pi_\theta(a_t|s_t) \delta^{\pi_\theta}(s_t, a_t)\right] \tag{8}$$

The critic estimates the value function $V^\pi$, which is parameterized by $\eta$ and is then updated by minimizing the squared loss of the TD error:

$$L(\eta) = \mathbb{E}_{\tau\sim p_{\pi_\theta}(\tau)}\left[\sum_{t=0}^{H} ||r(s_t, a_t) + \gamma V_\eta^{\pi_\theta}(s_{t+1}) - V_\eta^{\pi_\theta}(s_t)||^2\right] \tag{9}$$

By doing so, actor-critic combines the benefits of policy gradient methods and the learned value function, which can reduce both the variance of policy gradient and the bias introduced by the value function. Therefore, actor-critic has attached lots of attention in recent years, and a number of actor-critic methods have been proposed in the literature [29], [33], [37], [38], [39]. In this article, we consider the actor-critic as the RL algorithm in the proposed method, which will be detailed in Section III.

### B. Transfer Reinforcement Learning

TRL contends to enhance the learning performance of the RL agents in a target task by using the knowledge learned from other related source tasks. As aforementioned, according to the differences between the source and target tasks, existing TRL algorithms can be categorized into two categories, i.e., homogeneous TRL and heterogeneous TRL.

In homogeneous TRL, the source and target tasks are the same or only differ in reward distribution, which means that the state-action spaces and environment dynamics of source and target tasks are the same. In this setting, homogeneous TRL methods can transfer knowledge through transfer of policies or demonstrations. For policy transfer, existing methods often transfer knowledge through policy reuse. For instance, Fernández et al. [21] proposed the policy reuse method based on Q-learning, in which the behavior policy of target task was chosen from either the source policies or itself under a reuse

probability. Rosman et al. [40] proposed an approach based on Bayesian optimisation, which can choose the suitable policy from a policy library, and then reuse the selected policy for efficiently responding to an unseen target task. Barreto et al. [41] proposed a Q-value based TRL method, which extends policy reuse to be generalized across multiple source tasks. Moreover, some methods leverage the policy distillation to transfer knowledge, in which the target agent is trained to match the probability distribution over actions with the source task. For instance, Rusu et al. [42] proposed a policy distillation method based on Deep Q-Network (DQN) to transfer the well-learned policy to a new RL network. Parisotto et al. [43] proposed to distill the knowledge of expert agents into the target agent by minimizing the cross entropy between the expert and target policies at the pre-training phrase. Czarnecki et al. [23] explored the landscape of policy distillation and discussed several variants of policy distillation with theoretical and empirical analysis. Moreover, Teh et al. [44] proposed to distill a shared policy from multiple source tasks, and then transfer the shared policy network to the target agent using Kullback-Leibler (KL) divergence to accelerate the learning of the target task.

For demonstration transfer, it contends to transfer the demonstration of a source RL agent to the target RL agent with the aim of providing high-quality training data. In particular, Kang et al. [45] and Nair et al. [46] leveraged the demonstration data from the source tasks to guide the agent in a target task to explore meaningful states when the environmental feedback is sparse, which brought remarkable benefits for the policy learning in target task. Hester et al. [22] proposed to transfer the demonstration at pre-training stage to improve the initial performance of the target agent. More reviews of homogeneous TRL can refer to [16], [19]. From the discussions above, we can see that the homogeneous TRL methods transfer knowledge through reuse of policies or demonstrations, which require the source and target tasks share common state and action representations, they thus cannot be applied in heterogeneous TRL problems, in which the source and target tasks are differing in state-action spaces.

In contrast to the tremendous attempts in homogeneous TRL, only a few studies have been conducted in the literature for heterogeneous TRL. This might be because the differences lying in the state-action spaces between source and target tasks bring great challenges in conducting knowledge transfer between them. In order to enable the knowledge transfer between heterogeneous RL problems, some TRL approaches try to first build or learn the bridge to connect the source and target tasks for knowledge transfer [25], [26], [27]. In particular, as introduced in Section I, existing heterogeneous TRL methods either try to manually design or automatically learn the mapping between source and target tasks, or learn a common space of the source and target tasks. Besides the methods aforementioned, Joshi et al. [47] adapted and reused the source task policy in a target task, after learning a mutual mapping between the state-action spaces of the source and target tasks. Although these methods are able to conduct knowledge transfer across heterogeneous RL tasks, it is worth noting here that they are often computationally expensive in the learning inter-task mapping or common space across RL tasks. It is thus impractical to apply these methods in more complex scenarios where multiple heterogeneous tasks are available for knowledge transfer, since computational cost rises dramatically when the number of source tasks increases.

In the setting of having multiple heterogeneous source tasks, towards positive knowledge transfer across RL tasks, the target agent has to identify the appropriate source task or a set of source tasks automatically and then transfer knowledge across tasks while the RL progresses in the target task online. To the best of our knowledge, although this setting is common in practice, it has not been explored in heterogeneous TRL yet. In the literature, there are only a few methods considering multiple source RL tasks in the setting of homogeneous problem tasks. In particular, Rajendran et al. [48] developed an attentive deep architecture to perform selective transfer from multiple source tasks in the same RL domain. The attention network is learned to make a choice on the source policy selection for the target agent. Li et al. [49] utilized option framework to learn the selection of a source policy for reuse, which considered the contexts (i.e., a subset of states) in the selection process. Furthermore, Li et al. [50] presented an online source policy selection algorithm which considers the source policy selection task as a multi-armed bandit problem and regards different policies of the source RL tasks as bandits for selection. More recently, Yang et al. [51] modeled the problem of source task selection as the option learning problem, and proposed to alternatively select the appropriate source tasks online without measuring the similarity between tasks. Cheng et al. [52] proposed an attention-based mixture of experts multitask RL approach to select the proper group of expert source tasks to improve the performance of the target task. However, it is worth noting here that as these methods rely on the assumption that the RL tasks for knowledge transfer share a common state-action space and possess high relatedness, they cannot be directly applied in TRL with multiple heterogeneous problem tasks.

In the next section, the details of the proposed TRL algorithm with adaptive policy gradient transfer across multiple heterogeneous problem tasks are presented.

## III. Proposed Algorithm

In this section, we present the proposed policy gradient-based TRL algorithm for the scenario when multiple heterogeneous source RL tasks are available for knowledge transfer across tasks. In particular, we first introduce the outline of the proposed TRL algorithm. Next, the two core components of the proposed TRL algorithm, i.e., source task selection and knowledge transfer across heterogeneous RL tasks, are detailed.

### A. Outline of the Proposed TRL Algorithm

Fig. 1 gives the outline of the proposed TRL approach, which includes two main components. One is *source task selection module*, which is to select the appropriate source task from multiple heterogeneous source tasks. As the environment of the target task is often unknown beforehand and the target agent has no prior knowledge about the source tasks, we propose to perform source task selection while the RL process in target task

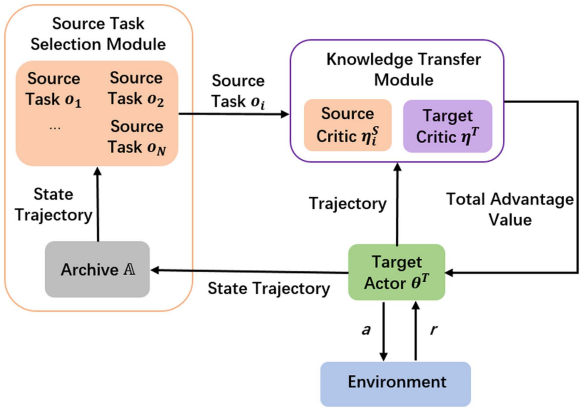



Fig. 1. Illustration of the proposed TRL across multiple heterogeneous problems.

progresses online, towards positive knowledge transfer in TRL. In this manner, the proposed selection module is able to automatically adapt the selection of source task in different learning stages of the target task. The other core component is *knowledge transfer module*, which is responsible for conducting knowledge transfer across heterogeneous RL tasks. In our proposed TRL method, we consider the advantage value as useful knowledge to be transferred across RL tasks. It can be regarded as a bias provided by source task in the update of the target policy (see (8)) during the learning process. In this way, useful traits learned in a source RL task can be easily transferred to a heterogeneous target task efficiently. Moreover, by considering the transferred knowledge as an update bias in the learning of policy, the target agent is able to learn from both the environment and the selected source tasks simultaneously, which makes the target agent robust to the negative effect brought by the transferred knowledge. Then, as the proposed *knowledge transfer module* does not need the learning of mapping across tasks, it can be easily applied in cases having multiple heterogeneous source RL tasks without incurring much extra computational burden in the RL process.

Moreover, Algorithm 1 summarizes the pseudo code of the proposed TRL algorithm. In particular, let $O = \{o_1, o_2, \ldots, o_N\}$ and $o_T$ denote a set of source tasks and the target task, respectively. The proposed algorithm follows the following learning loop: in an episode, the RL agent of the target task starts by sampling trajectories $\tau$ (i.e., state-action pairs) from the environment until reaching a predefined batch size $M$ (line 2–8 in Algorithm 1). Next, the agent transfers the trajectory samples to the *source task selection module* to do online source task selection (line 9 in Algorithm 1). After selecting the appropriate source task $o_i$, the *knowledge transfer module* then kicks in to transfer the advantage value of the selected source task $o_i$ to the target task $o_T$ to provide a bias in the update of the target policy (line 10 in Algorithm 1). This learning loop will terminate until the target policy converges or a certain predefined stopping criteria are satisfied.

The details of the *source task selection module* and the *knowledge transfer module* will be presented in Section III-B and Section III-C, respectively.

**Algorithm 1:** Pseudo Code of the Proposed TRL Algorithm.

**Parameter:** Batch size $M$; Number of episode $E$;
**Input:** Source tasks $O = \{o_1, o_2, \ldots, o_N\}$; Number of source task $N$; Target task $o_T$; Episode counter $e = 0$;
**Output:** Target actor $\theta^T$; Target critic $\eta^T$;
1: **begin**
2: **while** $e \in \{0, \ldots, E\}$ **do**
3: $m \leftarrow 0$;
4: **while** $m < M$ or the terminal state is not reached **do**
5: Perform an action $a \sim \pi_{\theta^T}$;
6: Observe the new state $s'$ and receive the reward $r$;
7: $m \leftarrow m + 1$;
8: **end while**
9: Execute the *source task selection module* to choose the appropriate source task $o_i$;
10: Execute the *transfer knowledge module*, and update the target model $\theta^T, \eta^T$;
11: $e \leftarrow e + 1$;
12: **end while**
13: **end**

### B. Source Task Selection Module

As introduced in Section II-A, the problem of source task selection has been explored in the homogeneous TRL algorithm designs [48], [49], [50], [51]. From the existing homogeneous TRL methods, it can be observed that computing similarity between the source and target tasks is a common way for selecting the appropriate source tasks for knowledge transfer. In particular, these methods are either based on the calculations of the Markov decision process (MDPs) [53], [54] between source and target tasks, which include the similarities of state vectors, actions and reward functions, or using the benefits that the target agent could be obtained by selecting a source task [55], which are often approximated by the similarity of policies or Q value (see (4)) distributions between the source and target tasks. However, these methods cannot be directly applied in heterogeneous TRL. First of all, for the MDP components, such as state, action and reward, due to the differences of state-action representations and reward functions across heterogeneous RL tasks, they may be similar in value but with different practical meanings in different RL tasks. Therefore, it is not reliable to calculate the similarities based on the MDP components. For the policies or Q values, as the mapping between state and action spaces may be different across heterogeneous RL tasks, given a particular state-action pair in target task, a similar state-action pair may not exist in source tasks. Thus, the Q values and policy depended on state-action pairs are not suitable to measure the similarity in heterogeneous TRL as well.

From the discussions above, we can see that it is hard to evaluate the similarity between the source and target tasks directly based on the MDP components, policy and Q function distribution for heterogeneous TRL. However, since the optimal policy can be recovered from a state value distribution by moving from a low value area to a high value area, the state value distributions can be used to measure if a common policy

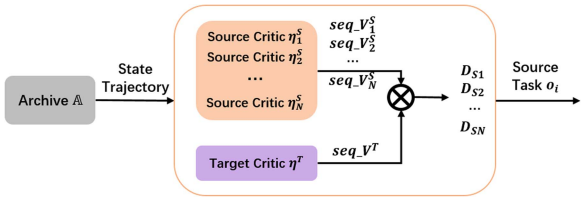



Fig. 2. Illustration of the proposed source task selection module.

exists in source and target tasks. Taking this cue, we propose a new method to compute similarity based on the local feature of the state-value function $V(s)$ depended on sampled state trajectories in the target task. Compared to the Q value function and policy, the V value function depends on only the state vector, so that the way of computing V value can avoid the dilemma brought by the coupling of state and action spaces as discussed above. Moreover, as the transferred knowledge is only related to the current target policy and the global value distribution is not available during the learning process, the local state value distributions depended on the latest sampled trajectory in the target task is used to compute the similarity between source and target tasks. Fig. 2 illustrates the outline of the *source task selection module*.

Particularly, first of all, We devise an archive to house the state sequences in trajectories sampled by target agent during the learning process. Once these trajectories are used to update the similarity between source and target tasks, they will be removed from this archive. Therefore, the archive size can be configured to accommodate the maximal number of samples (states) collected by the target agent. Next, the sequences of V-values of the state trajectories in the archive are computed by every source and target RL models, which are regarded as the local state value distributions of every source and target tasks. Subsequently, we normalize these V value sequences to adjust these V values to a common scale. The normalized value sequences reflect the change regularity of local state value distributions. If the source and target tasks are similar in the local area under the current policy, the V value sequences computed by source and target RL models should possess the similar change regularity. Considering above, Euclidean distance is employed to measure the discrepancy between source and target tasks, which is given by:

$$D_S = ||\mathcal{N}orm(seq\_V^S), \mathcal{N}orm(seq\_V^T))|| \tag{10}$$

where $seq\_V^S$ and $seq\_V^T$ denote the sequence of V-values computed by the source and the target RL models. $\mathcal{N}orm(seq\_V^S)$ and $\mathcal{N}orm(seq\_V^T)$ are the normalized values of $seq\_V^S$ and $seq\_V^T$. Here, we use min-max normalization to scale $seq\_V^S$ and $seq\_V^T$ into the range [0,1].

Moreover, Algorithm 2 gives the pseudo code of the proposed *source task selection module*. Specifically, given a set of heterogeneous source tasks, which are denoted by $O = \{o_1, \ldots, o_N\}$. The well-learned critic networks of each source task are parameterized by $\eta^S_{n=1,2\ldots N}$. The target task is denoted by $o_T$, and the critic network of the target task is parameterized by $\eta^T$.

**Algorithm 2:** Pseudo Code of the Proposed Source Task Selection Module.

**Parameter:** State sequence archive $\mathbb{A}$; Archive size $K$;
**Input:** Number of source tasks $N$: Set of the source tasks $L = \{o_1, o_2, \ldots, o_N\}$; Target task $o_T$; Source Critic $\eta^S_{n=1,2,\ldots,N}$; Target Critic $\eta^T$; Trajectories $\tau$;
**Output:** The selected source task $o_i$;
1: **begin**
2: Preserve the state sequence of trajectories $\tau$ into archive $\mathbb{A}$;
3: **for** each task $o_n$ **do**
4: Computes V-value of source task $seq\_V^S$ by the source critic $\eta_n^S$ according to the state sequence in the archive $\mathbb{A}$;
5: Computes V-value of target task $seq\_V^T$ by the target critic $\eta^T$ according to the state sequence in the archive $\mathbb{A}$;
6: Compute the distance $D_{Sn}$ between the source task $o_n$ and the target task $o_T$ using (10);
7: **end for**
8: Select the source task $o_i$ with the smallest distance $D_{Si}$ for the *knowledge transfer module*;
9: **end**

The process of source task selection can then be summarized as follows: first of all, the RL agent of the target task collects the trajectories $\tau$ in the target task $o_T$; next, the state vectors in trajectories $\tau$ are saved in the archive $\mathbb{A}$; if the number of states archived in $\mathbb{A}$ is larger than the size of the archive $K$, the earliest saved state vectors will be discarded (line 2 in Algorithm 2); subsequently, every source critic calculates the source V-value sequences $seq\_V^S_{1,2,\ldots,N}$ of each source task according to the state sequence in the archive $\mathbb{A}$ (line 4 in Algorithm 2); meanwhile, the target critic calculates the target V-value sequence $seq\_V^T$ using the same state sequence in the archive $\mathbb{A}$ (line 5 in Algorithm 2); after that, the discrepancies between every source task and the target task are obtained by computing the distance between $seq\_V^S_{1,2,\ldots,N}$ and $seq\_V^T$ based on (10) (Line 6 in Algorithm 2); lastly, the source task with the smallest discrepancy, which is represented by $o_i$, will be selected to serve as the input of the *knowledge transfer module* (line 8 in Algorithm 2).

Furthermore, in heterogeneous TRL, the dimension of the state in the target task may be different from that in the source task. In order to allow the source critic to calculate the V-values using the state sequence in the archive $\mathbb{A}$, we have to reconstruct the state vectors to make the state trajectories have a common dimension with the source tasks. In this study, we propose to simply select the dimensions of the source and target tasks which share the common or similar physical meanings to reconstruct the state vectors. Taking the robot arm with different links as an example, as shown in Fig. 3, the state of the robot arm contains the coordinates of the target point, the angle of each joint, as well as the distance from the fingertip to the target point. If a two-link arm robot and a three-link arm robot serve as the agents in the source and target RL tasks, respectively, the two digits in

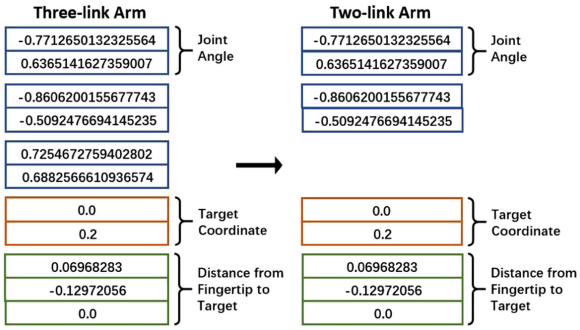



Fig. 3. Illustration of state vector reconstruction.

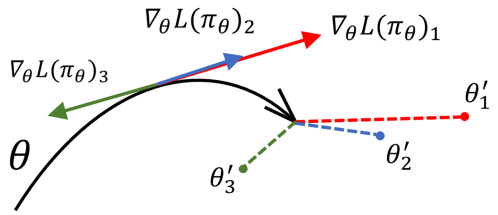



Fig. 4. Illustration of the gradient update of the policy which is parameterized by $\theta$.

the state of the three-link arm robot, which represents the angle of the additional joint with respect to the two-link arm, will be discarded to make the state of the three-link arm robot has equal dimensionality with that of the two-link arm robot.

### C. Knowledge Transfer Module

Besides the source task selection, the other core ingredient of the proposed TRL algorithm is the *knowledge transfer module*, which is to capture useful knowledge from the selected source task and transfer to the target task with heterogeneous state-action space for enhanced RL performance.

As discussed in Section II-A, the learning of the policy in an actor-critic agent is based on the estimation of the advantage function. In particular, as illustrated in Fig. 4, where $\bigtriangledown_\theta L(\pi_\theta)_1$, $\bigtriangledown_\theta L(\pi_\theta)_2$ and $\bigtriangledown_\theta L(\pi_\theta)_3$ denote the gradients of the loss functions possessing different advantage values, the policy which is parameterized by $\theta$ is updated based on the gradient in different directions with various step sizes, e.g., $\theta'_1$, $\theta'_2$ and $\theta'_3$ in Fig. 4. However, when the agent has not learned the proper policy from sufficient sampled data during learning process, the V value computed by critic is almost random and cannot represent the actual cumulative discounted expected value. According to TD-error function (see (7)), advantage value is thus not able to facilitate the learning of the policy at early stage of the learning process. However, a better advantage value is expected to be computed depended on the critic of the selected source task, which is well-trained and has similar state value distribution to the target task. Taking this cue, in this study, we propose to transfer the advantage value as useful knowledge from the selected source task (source advantage value) to the target task to guide the step size and direction of the policy update of

**Algorithm 3:** Pseudo Code of the Proposed Knowledge Transfer Module

**Input**: Number of source task $N$: Source critic $\eta^S_{n=1,2...N}$; Target critic $\eta^T$; Weight of knowledge transfer $\alpha$; Trajectories $\tau$; The selected source task $o_i$;
**Output**: Target critic $\eta^T$; Target actor $\theta^T$;

1: **begin**
2: **for** each sample in trajectories $\tau$ **do**
3: Compute the self-learning advantage value via (11)
4: Compute the transferred advantage value via (12)
5: Compute the total target advantage value via (13)
6: **end for**
7: Compute the gradient of the target actor using (14)
8: Update the parameters of the target actor $\theta^T$:

$$\theta^T \longleftarrow \theta^T + \beta \nabla_{\theta^T} L(\pi_{\theta^T})$$

9: Compute the loss of the target critic using (15)
10: Update the parameters of the target critic $\eta^T$:

$$\eta^T \longleftarrow \eta^T - \beta \nabla_{\eta^T} L(\eta^T)$$

11: **end**

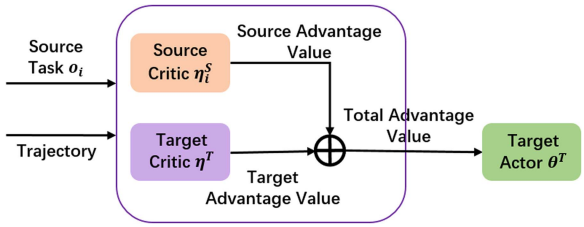



Fig. 5. Illustration of the proposed knowledge transfer module.

the actor-critic in the target task. With the guidance of source advantage value, the target agent is able to reduce the random exploration and collect high-quality trajectories efficiently.

In particular, the proposed knowledge transfer across heterogeneous RL task is summarized in Algorithm 3, and Fig. 5 shows the outline of the proposed *knowledge transfer module*. First of all, according to the trajectories $\tau$ sampled from the target task, the target agent computes the self-learning advantage value $A_T$ (Line 3 in Algorithm 3), which is given by:

$$A^T(s_t, a_t) = r(s_t, a_t) + \gamma V^T(s_{t+1}) - V^T(s_t) \tag{11}$$

where $V_T$ is the V-value computed by the critic network of target task, $r(s_t, a_t)$ is the reward received by the target agent from the environment.

Next, after selecting the appropriate source task $o_i$, the critic network of the source task $o_i$ calculates the transferred advantage value $A^S$ according to the trajectories $\tau$ (Line 4 in Algorithm 3), which is given by:

$$A^S(s_t, a_t) = r(s_t, a_t) + \gamma V^S(s_{t+1}) - V^S(s_t) \tag{12}$$

where $V^S$ is the V-value of the selected source task, and $r(s_t, a_t)$ is the reward received by the target agent from the environment.

It is noted that, if the state vector of the target task has different dimensions with the selected source task, the state vectors in the trajectories $\tau$ need to be reconstructed as described in Section III-B, to make the source and target states have equal dimensionality.

Subsequently, the total advantage value $A^{total}$ for the policy learning in the target task, is then determined by the self-learning advantage value $A^T$ and the transferred source advantage value $A^S$ (Line 5 in Algorithm 3), which is given by:

$$A^{total}(s_t, a_t) = \alpha A^S(s_t, a_t) + (1-\alpha)A^T(s_t, a_t) \quad (13)$$

where $\alpha$ is to balance the self-learning from the environment and transfer learning from the source tasks of the agent in the target task. As the target policy is updated based on both the self-learning advantage value and the transfer advantage value, when the transferred advantage is not proper, the target agents can still learn from the self-learning advantage. Besides, when the transfer advantage value guide the target agent explore in the wrong direction, the target agent can self-correct from negative feedback from environment. Therefore, the proposed knowledge transfer method is able to reduce the effects brought by the negatively transferred knowledge to some extent.

Moreover, the actor network of the agent in the target task, which is parameterized by $\theta^T$, is then updated according to the total advantage value (Line 7-8 in Algorithm 3), in which the gradient of the loss is given by:

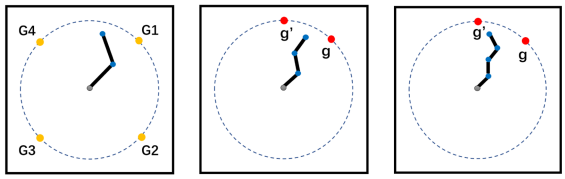
$$\nabla_{\theta^T} L(\pi_{\theta^T}) = \sum_{t=0}^{H} \nabla_{\theta^T} \log \pi_{\theta^T}(a_t|s_t) A^{total}(s_t, a_t)) \quad (14)$$

As both the target advantage value and the transfer advantage value work on the update of the target policy, the proposed knowledge transfer method is robust to the negative transferred knowledge. It is because that when the transferred advantage is not proper, it can still be fixed by the self-learning advantage in a degree.

The optimization of the target critic network, which is parameterized by $\eta^T$, is proceeded as routine (Line 9-10 in Algorithm 3), which is described in Section II-A, that is given by:

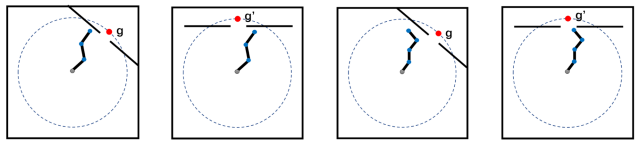
$$L(\eta^T) = \sum_{t=0}^{H} ||r(s_t, a_t) + V_{\eta^T}^T(s_{t+1}) - V_{\eta^T}^T(s_t))||^2 \quad (15)$$

From the discussion of the proposed method, we can see that the computational cost incurred by knowledge transfer is mainly on three procedures. First of all, in the *source task selection module*, the forward propagation of the source critic for every source task is performed to compute the V value. Next, Euclidean distances between the V value sequences of source and target tasks are computed to estimate the similarity between source and target tasks. In these two procedures, with N source tasks, the time complexity is $O(N)$, where N denotes the source task number. Moreover, in the *knowledge transfer module*, the source advantage value is computed for the selected source task, of which computational cost is a constant. Therefore, the time complexity of the proposed knowledge transfer is $O(N)$.




(a) The 2-link reacher tasks with goal position G1, G2, G3, and G4. (b) The 3-link reacher tasks with goal position g' and g. (c) The 4-link reacher tasks with goal position g' and g.

Fig. 6. Illustration of the reacher tasks.




(a) The 3-link insert task with goal position g. (b) The 3-link insert task with goal position g'. (c) The 4-link insert task with goal position g. (d) The 4-link insert task with goal position g'.

Fig. 7. Illustration of the insert tasks.

## IV. Experimental Study

This section presents the empirical studies conducted to investigate and analyze the performance of the proposed TRL algorithm based on the well-known robot RL tasks over both the RL algorithm without knowledge transfer and the existing state-of-the-art TRL algorithm.

### A. Experimental Setup

In this study, the proposed TRL algorithm is evaluated on two well-known robot tasks, which are the Reacher and the Insert task built on the MUJOCO physics simulator.[2] In order to construct multiple heterogeneous RL scenarios, according to [27], we build different types of robots with different number of arms (links), i.e. two links, three links and four links, in both the reacher and insert tasks, which are illustrated in Figs. 6 and 7, respectively. Arms of the robots are all with the same length. In both the reacher and insert tasks, the state of the RL robot is consist of the joints angles, the joint position as well as the goal position. The action of the robot is the set of driving forces of each arm, which changes the angular velocities of the arm. Therefore, the state-action spaces of the robots with different number of arms are heterogeneous. Moreover, Table I summarizes the dimension property of the state spaces and the action spaces of all the RL tasks considered in this study. In the reacher and insert RL tasks, the initial position of the robot arms are randomly generated at the beginning of a learning episode. The objective of the RL robot or agent is to reach at a pre-defined position (goal position) which is often set as the farthest position that the robot arms are able to reach. During the RL process, a negative reward that is proportional to the distance between

[2]https://www.roboti.us/

TABLE I
SUMMARY OF THE DIMENSION PROPERTIES OF THE STATE SPACES AND THE ACTION SPACES OF ALL THE RL TASKS

| RL tasks | Dimension of the state space | Dimension of the action space |
|---|---|---|
| 2-link reacher | 9 | 2 |
| 3-link reacher | 11 | 3 |
| 4-link reacher | 13 | 4 |
| 2-link insert | 9 | 2 |
| 3-link insert | 11 | 3 |
| 4-link insert | 13 | 4 |

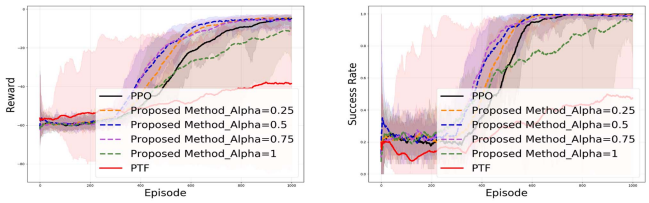
TABLE II
SUMMARY OF THE HYPER-PARAMETERS IN PPO

| Parameter | Default |
|---|---|
| Number of layer | 3 |
| Node number in hidden layer | 64 |
| Reward decay | 0.99 |
| Learning rate | 0.001 |

the end effector of the robot and the goal position, is given to the agent in each learning episode. Once the agent achieved the objective of the given task successfully, it will be given a positive reward (i.e., $+1$). The RL process of an agent will terminate when its moving steps exceed a fixed number of episodes or it reaches the target position.

Next, for the compared baseline algorithms, both the RL algorithm without knowledge transfer and a state-of-the-art TRL algorithm are considered for comparison. In particular, the proximal policy optimization (PPO) [29], which is one of the state-of-the-art policy gradient-based RL algorithms, is employed as the basic RL method in this study. Moreover, as discussed in Section II, since there is no existing study working on TRL in cases having multiple heterogeneous source RL tasks, the Policy Transfer Framework (PTF) [51], which is the recently proposed TRL method for multiple source tasks in homogeneous TRL, is compared in this study. To enable the learning of policy in heterogeneous TRL, the operation of reconstructing state vector used in the proposed method is also applied in PTF to ensure a fair comparison. Furthermore, in order to investigate the impact of the amount of knowledge transfer in the proposed algorithm, we investigate the proposed TRL algorithm with different transfer weights $\alpha$ (see (13)), i.e., $\alpha = 0.25$, 0.5, 0.75, 1.

Lastly, to ensure a fair comparison, all the compared algorithms employ the PPO [29] with a common configuration as the basic RL algorithm. Further, the policy and value networks of the actor-critic in all the compared algorithms are configured as neural networks involving three layers and 64 hidden units. The common RELU activation is adopted in the neural networks. According to [56], the standard back propagation method is employed to train these neural networks via the ADAM optimizer with learning rate 0.001. The hyper-parameters of basic RL algorithm are summarized in Table II. Moreover, in this study, all the compared algorithms are executed for 20 independent runs having different randomly generated starting state to obtain the averaged results for comparison.




(a) Averaged reward (b) Averaged success rate

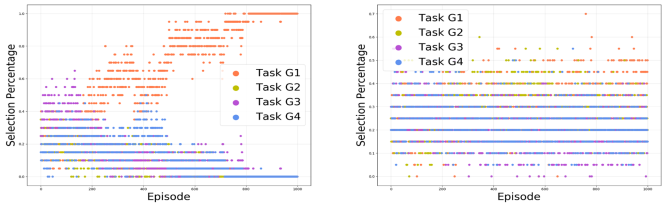
Fig. 8. Comparison on the 3-link reacher task with goal position $g$.

### B. TRL Across the Reacher Tasks

Fig. 6 illustrates the reacher tasks with 2-link, 3-link and 4-link robot arms, which serve as the source and target RL tasks in this study. In particular, here we consider the four 2-link reacher tasks with different goal positions (see Fig. 6(a)), i.e., $G1$, $G2$, $G3$ and $G4$, as the source tasks, and the 3-link reacher and 4-link reacher with goal positions $g$ and $g'$ (see Fig. 6(b) and (c)) as the target tasks. As can be observed in Fig. 6, different goal positions of source and target tasks are configured to investigate the efficacy of the proposed algorithm. Specifically, the goal position $g$ of the target tasks is the same as the goal position $G1$ in the source task, while goal position $g'$ of the target tasks lies in between the goal positions $G1$ and $G4$ of the source tasks.

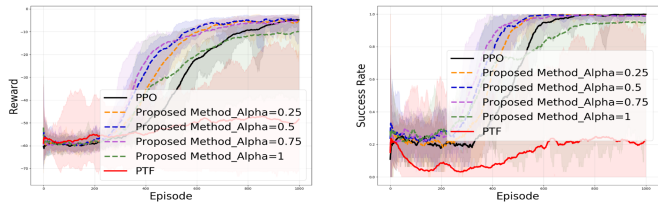
*1) 3-Link Reacher With Goal Position $g$:* Fig. 8 presents the averaged results in terms of averaged reward or averaged success rate[3] which are obtained by the proposed algorithm and the compared approaches on the 3-link reacher task with goal position $g$, over 20 independent runs. In Fig. 8, the Y-axis gives the obtained averaged reward or averaged success rate, and the X-axis denotes the number of learning episode incurred by the compared algorithms so far. From the figure, we can see that the proposed method with $\alpha = 0.25, 0.5, 0.75$ has obtained superior learning performance over the baseline algorithm PPO which is with no knowledge transfer from the source tasks, with respect to both the averaged reward and averaged success rate. As the proposed method shares the same basic RL algorithm with PPO, the obtained results confirmed that the effectiveness of the proposed TRL algorithm in conducting knowledge transfer for enhanced RL performance across heterogeneous RL tasks. Moreover, when $\alpha = 1$, we noted that the performance of the proposed algorithm is worse than the PPO. This is because that the agent in the target task has no self-learning from the environment when $\alpha = 1$ (see (13)). As the source and target tasks are two heterogeneous RL tasks, only learning from the source task is not able to obtain a good policy for the target task. Further, for the compared TRL algorithm, i.e., the PTF, it is observed that both the obtained averaged reward and averaged success rate are much worse than that of the PPO. This shows that the knowledge transfer conducted by the PTF deteriorated the learning of policy in the target task, which indicates that the PTF is not suitable for heterogeneous TRL.

[3]In this study, the success rate is given by the ratio of the total number of times that the agent successfully achieves the goal position of the task and the total number of times that the agent performed on this task.



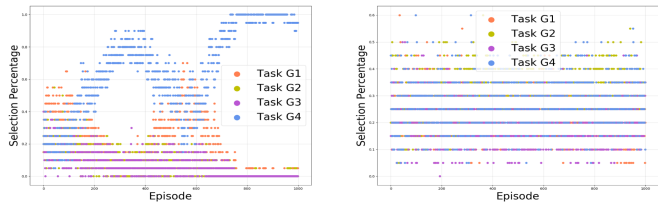

(a) Proposed algorithm (b) PTF

Fig. 9. Selection percentage of each source task in the proposed algorithm and PTF on the 3-link reacher task with goal position $g$, over 20 independent runs.




(a) Averaged reward (b) Averaged success rate

Fig. 10. Comparison on the 3-link reacher task with goal position $g'$.




(a) Proposed algorithm (b) PTF

Fig. 11. Selection percentage of each source task in the proposed algorithm and PTF on the 3-link reacher task with goal position $g'$, over 20 independent runs.

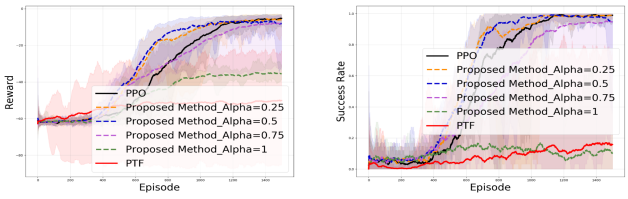
To provide a deeper insight of the proposed algorithm in source task selection from multiple heterogeneous RL tasks, Fig. 9 presents the percentage of each source task (i.e., the 2-link reacher tasks with goal positions $G1$, $G2$, $G3$, and $G4$) being selected by the proposed algorithm with $\alpha = 0.5$ and by the compared PTF method in every learning episode over the 20 independent runs. In Fig. 9, the Y-axis gives the percentage that a source task is selected in a particular episode over 20 independent runs, while the X-axis denotes the number of learning episode incurred by the compared algorithms so far. It can be observed from the figure that, at the beginning of the learning process, every source task is likely to be selected with equal probability in both the proposed algorithm and the PTF. This is because the target task has not been learned yet, and the selection of the source tasks is thus almost random. As the target agent learns from the environment over time, for the proposed method, the percentage of selecting the source task with goal position $G1$ increases rapidly, and maintains nearly 100% in the later stages of the learning process. This again confirmed the effectiveness of the proposed algorithm in identifying proper source tasks while the RL progresses in the target task online, since the goal position $G1$ of the source task is the same as the goal position of the target task, and the task $G1$ is most likely to provide useful knowledge for the target agent to reach the goal position over the other source tasks. In contrast to the proposed algorithm, the PTF has not learned the proper strategies for selecting source tasks. This is because that PTF learns to select the source tasks according to the environmental feedback, which requires a longer learning period, that will be discussed in the following experiments in Section IV-B3, using a larger number of learning episodes.

*2) 3-Link Reacher With Goal Position $g'$:* Next, to further investigate whether the proposed algorithm is able to learn and transfer knowledge alternatively from multiple source tasks towards enhanced RL performance in the target task, here the 3-link reacher with a different goal position $g'$ is considered as the target task. In this case, none of the source tasks shares a common goal position with the target task.

Fig. 10 presents the results obtained by the proposed algorithm and the compared methods on the 3-link reacher task with goal position $g'$, over 20 independent runs. As can be observed, the obtained results are similar to that as discussed in Section IV-B1. In particular, when $\alpha = 0.25, 0.5, 0.75$, the proposed algorithm achieved superior performance with respect to both the averaged reward and the averaged success rate over all the compared algorithms, although the target task has a different goal position with all the heterogeneous source tasks. When $\alpha = 1$, the performance of the proposed algorithm is worse than that of the baseline PPO, since there is no self-learning from the environment of the proposed algorithm in the target task if $\alpha = 1$. However, with knowledge transferred from the proper source tasks, we also observed faster learning speed obtained by the proposed algorithm in the early learning stages over the PPO when $\alpha = 1$. Moreover, similar to the observations of the PTF in Section IV-B1, the PTF obtained the worst averaged reward and averaged success rate among all the compared algorithms, which is even worse than that obtained in Section IV-B1. This is because the target task is now more different from all the source tasks, which indicates that the inappropriate knowledge transfer across RL tasks cannot only improve the learning of the policy, but also mislead the learning process.

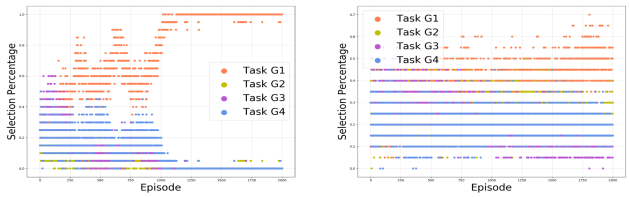
Next, Fig. 11 presents the percentage of every source task being selected along the RL process in both the proposed algorithm and the PTF to further explore the rationales behind the performance obtained as discussed above. $\alpha$ in the proposed algorithm is kept as 0.5 which is consistent with that in Section IV-B1. It can be observed from the figure that in the proposed algorithm, the target agent learns to select the source task $G1$ and $G4$ alternatively during the learning process. This gives the reason behind the superior performance obtained by the proposed algorithm above, since the goal positions of the source tasks $G1$ and $G4$ are more closer to the goal position $g'$ of the target task, and thus are more likely to provide useful knowledge for the learning of policy in the target task. Moreover, for the compared PTF, it again fails to learn the proper source task selection strategy in the RL learning process, which is due to the same reason as discussed in Section IV-B1.

*3) 4-Link Reacher With Goal Positions $g$ and $g'$:* Moreover, the more heterogeneous tasks are considered as the target tasks to further evaluate the performance of the proposed TRL algorithm.



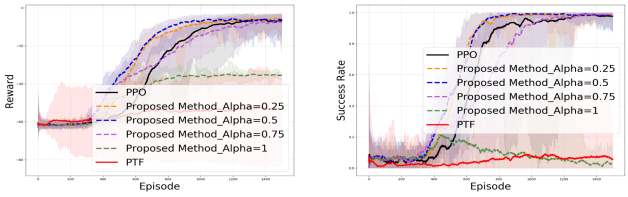

(a) Averaged reward (b) Averaged success rate

Fig. 12. Comparison on the 4-link reacher task with goal position $g$.




(a) Proposed algorithm (b) PTF

Fig. 13. Selection percentage of each source task in the proposed algorithm and PTF on the 4-link reacher task with goal position $g$, over 20 independent runs.



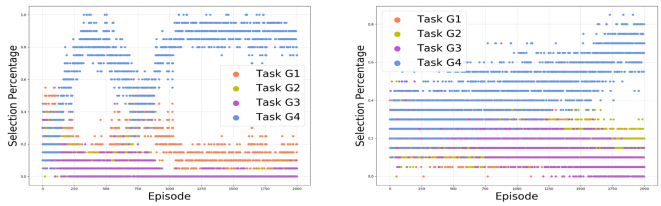

(a) Averaged reward (b) Averaged success rate

Fig. 14. Comparison on the 4-link reacher task with goal position $g'$.

In particular, in this section, we consider the 4-link reacher tasks with goal positions $g$ and $g'$ as the target tasks, and keep the 2-link reacher tasks with goal positions $G1$, $G2$, $G3$, and $G4$, as the source tasks.

Figs. 12 and 14 present the averaged reward and the averaged success rate of all the compared algorithms obtained on the 4-link reacher tasks with goal positions $g$ and $g'$, respectively, over 20 independent runs. As can be observed in the figures, when $\alpha = 0.25$ and 0.5 the proposed algorithm obtained faster learning speed than both the baseline algorithms PPO and PTF. For the compared PTF, as the difference between the source and target tasks is greater than the above experiments, the learning performance of the PTF becomes worse than that obtained when considering the 3-link reacher as the target task. Moreover, we also observed negative knowledge transfer of the proposed algorithm when configuring $\alpha = 0.75$ and 1. This is also due to the increases of differences between the source and target tasks, which require sufficient self-learning from the target environment to learn a good behavior policy.

Next, Figs. 13 and 15 present the selection percentages of each task in every learning episode of the proposed algorithm and the PTF, over 20 independent runs on the 4-link reacher tasks with goal positions $g$ and $g'$, respectively. $\alpha$ is kept as $\alpha = 0.5$ in the proposed algorithm. As can be observed from these figures,




(a) Proposed algorithm (b) PTF

Fig. 15. Selection percentage of each source task in the proposed algorithm and PTF on the 4-link reacher task with goal position $g'$, over 20 independent runs.

similar results have been obtained by the proposed algorithm and the PTF as discussed in Section IV-B1 and Section IV-B2 when considering different tasks as the target task. As the proposed algorithm and the PTF share the same basic RL algorithm, the obtained results again confirmed the effectiveness of the proposed algorithm in conducting TRL across heterogeneous RL tasks. Moreover, for the PTF, we can observe from the figures that it starts to select the useful tasks after around 1500 learning episode, which is much slower than the proposed algorithm. This is due to the learning of selection is based on the samples from the environment, which is of much lower efficiency.

Moreover, in order to facilitate a clearer comparison, we summarize the results in tables as well. We divide the learning process into five phrases, which are denoted by the percentages from 0% to 100%. Within each phrase, the averaged reward and success rate are computed and presented in two columns, labeled as "Averaged reward" and "Averaged success rate", respectively in the tables. The horizontal table header indicates the percentage of learning process (20%, 40%, 60%, 80% and 100%). The vertical table header includes the proposed method with diverse $\alpha$ and baseline algorithms. Table III show the averaged results for all the reacher tasks.

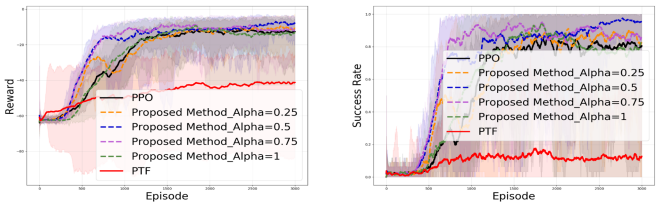
### C. TRL Across the Reacher and Insert Tasks

In this section, the more complex RL task, i.e., insert, is considered as the target task to further investigate the performance of the proposed algorithm for TRL across heterogeneous tasks. In contrast to the reacher task, the robot arm in an insert task, has to learn how to go through a narrow opening to reach a pre-defined goal position. Moreover, the insert task contains more actions, which thus requires a more sophisticated policy in action prediction to reach the goal position. In this study, we kept the 2-link reacher tasks with different goal positions $G1$, $G2$, $G3$ and $G4$, as the source tasks, and consider the 3-link insert and 4-link insert tasks as the target tasks. The target tasks have two position goals, i.e. $g$ and $g'$, which are illustrated in Fig. 7. As described in Section IV-B, the goal position $g$ of the target task is the same as one of the source tasks, and the goal position $g'$ of the target task lies in between the goal positions of two source tasks.

*1) 3-Link Insert With Goal Positions g and g′:* Figs. 16 and 18 present the averaged reward and averaged success rate obtained by the proposed algorithm and the compared algorithms on the 3-link insert task with goal positions $g$ and $g'$ over 20 independent runs, respectively. As the insert task is more complex

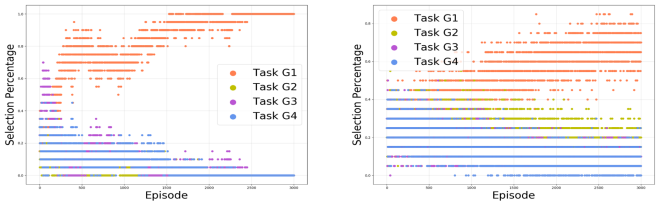
TABLE III
COMPARISON ON 4 REACHER TASKS

| | Averaged reward | | | | | Averaged success rate | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **20%** | **40%** | **60%** | **80%** | **100%** | **20%** | **40%** | **60%** | **80%** | **100%** |
| PPO | -60.28 | -51.0 | -26.14 | -10.67 | -6.3 | 0.13 | 0.33 | 0.79 | 0.98 | **0.99** |
| Proposed method $\alpha$ = 0.25 | -59.89 | -42.89 | -16.89 | -6.98 | **-5.28** | 0.14 | 0.45 | 0.9 | **0.99** | **0.99** |
| Proposed method $\alpha$ = 0.5 | **-59.57** | **-39.67** | **-13.49** | **-6.78** | -6.15 | **0.16** | **0.5** | **0.94** | 0.98 | 0.98 |
| Proposed method $\alpha$ = 0.75 | -60.08 | -43.76 | -17.92 | -8.66 | -6.82 | 0.16 | 0.4 | 0.88 | 0.97 | 0.97 |
| Proposed method $\alpha$ = 1 | -60.32 | -50.93 | -34.84 | -27.32 | -23.61 | 0.16 | 0.26 | 0.4 | 0.46 | 0.5 |
| PTF | -57.72 | -53.97 | -51.68 | -49.12 | -46.2 | 0.07 | 0.08 | 0.15 | 0.2 | 0.24 |



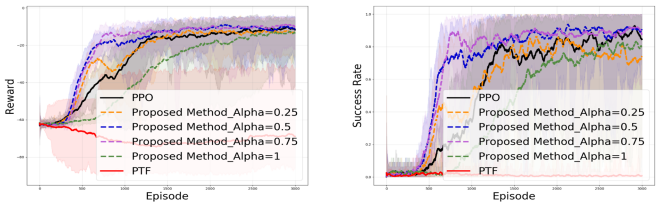

(a) Averaged reward (b) Averaged success rate

Fig. 16. Comparison on the 3-link insert task with goal position $g$.




(a) Proposed algorithm (b) PTF

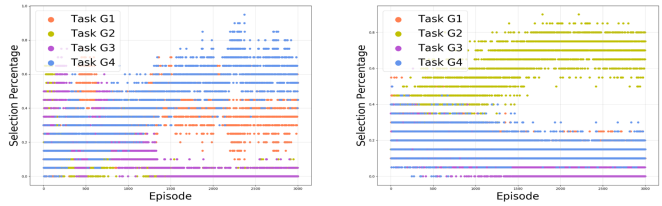
Fig. 17. Selection percentage of each source task in the proposed algorithm and PTF on the 3-link insert task with goal position $g$, over 20 independent runs.




(a) Averaged reward (b) Averaged success rate

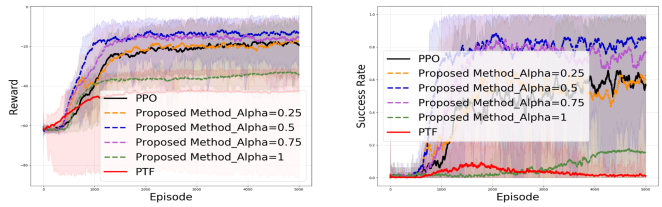
Fig. 18. Comparison on the 3-link insert task with goal position $g'$.

than the reacher task, it can be observed from the figures that the success rates obtained by all the compared algorithms are with higher variances than those obtained by these algorithms on the reacher tasks. Moreover, due to the complexity of the target task, proper guidance in the learning process is important for efficient policy learning. As can be observed, in both Figs. 16 and 18, with the increase of $\alpha$ from 0.25 to 0.75, the learning performance of the proposed algorithm has been improved, in terms of both averaged reward and averaged success rate. In the case of $\alpha = 0.5$ and 0.75, the proposed algorithm achieved the best learning performance over both the baseline algorithms PPO and PTF. Furthermore, for the compared TRL




(a) Proposed algorithm (b) PTF

Fig. 19. Selection percentage of each source task in the proposed algorithm and PTF on the 3-link insert task with goal position $g'$, over 20 independent runs.




(a) Averaged reward (b) Averaged success rate

Fig. 20. Comparison on the 4-link insert task with goal position $g$.

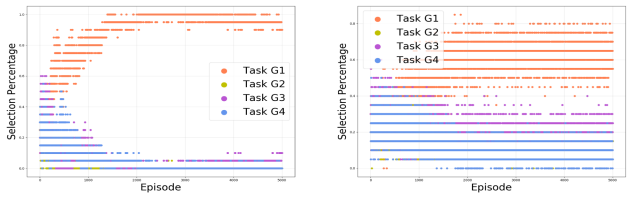
algorithm, i.e., the PTF, it obtained the worst learning performance among all the compared algorithms due to the improper knowledge transfer across RL tasks.

Next, Figs. 17 and 19 depict the selection percentages of the source tasks in each learning episode of the proposed algorithm with $\alpha = 0.5$ and the PTF, over 20 independent runs, respectively. It can be observed in Fig. 17 that, on the 3-link insert task with goal position $g$, the proposed algorithm is able to select the proper source task (task $G1$ that shares the same goal position with the target task) quickly less than 500 learning episode. The compared PTF can also identify the proper source task using around 1000 learning episode. However, due to the knowledge transfer method in PTF is not suitable for heterogeneous TRL, it also fails to enhance the learning performance in the target task as depicted in Fig. 16. Moreover, on the 3-link insert task with goal position $g'$, as can be observed in Fig. 19, the proposed algorithm again selected the proper source tasks (i.e., tasks $G1$ and $G4$ which have goal positions close to the goal position $g'$ of the target task) around 800 learning episode. However, in this case, the PTF failed to select the proper source tasks.

*2) 4-Link Insert With Goal Positions $g$ and $g'$:* Figs. 20 and 22 present the averaged reward and the averaged success rate, which are obtained by the proposed TRL algorithm and the

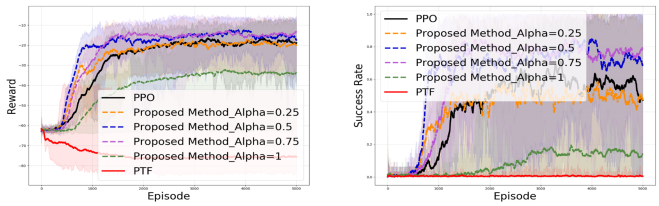
TABLE IV
COMPARISON ON 4 INSERT TASKS

| | Averaged reward | | | | | Averaged success rate | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **20%** | **40%** | **60%** | **80%** | **100%** | **20%** | **40%** | **60%** | **80%** | **100%** |
| PPO | -56.07 | -29.22 | -18.06 | -16.23 | -15.21 | 0.04 | 0.37 | 0.64 | 0.69 | 0.71 |
| Proposed method $\alpha$ = 0.25 | -51.87 | -27.0 | -17.7 | -16.13 | -15.58 | 0.08 | 0.44 | 0.66 | 0.68 | 0.66 |
| Proposed method $\alpha$ = 0.5 | **-48.48** | -17.57 | -13.88 | **-12.38** | **-12.18** | **0.14** | **0.73** | **0.81** | **0.84** | **0.85** |
| Proposed method $\alpha$ = 0.75 | -51.33 | **-16.7** | **-13.32** | -12.78 | -13.12 | 0.08 | 0.75 | 0.83 | 0.82 | 0.81 |
| Proposed method $\alpha$ = 1 | -58.57 | -39.79 | -28.62 | -24.99 | -23.98 | 0.02 | 0.15 | 0.36 | 0.45 | 0.47 |
| PTF | -60.94 | -59.96 | -59.32 | -58.32 | -57.29 | 0.01 | 0.05 | 0.05 | 0.04 | 0.04 |




(a) Proposed algorithm (b) PTF

Fig. 21. Selection percentage of each source task in the proposed algorithm and PTF on the 4-link insert task with goal position $g$, over 20 independent runs.

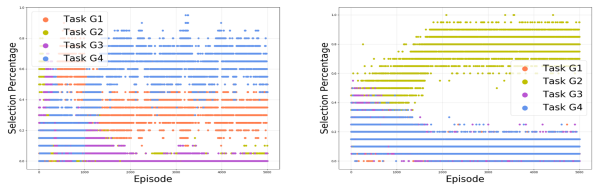



(a) Averaged reward (b) Averaged success rate

Fig. 22. Comparison on the 4-link insert task with goal position $g'$.

compared algorithms on the 4-link insert task with goal positions $g$ and $g'$ over 20 independent runs, respectively. Compared to the 3-link insert task, with the increase of number of arms (links), the RL tasks become more complex here. In this case, we thus observed that the basic RL algorithm PPO without knowledge transfer can only obtain around 60% averaged success rate. However, with useful traits transferred across tasks via the proposed algorithm, the averaged success rates have been improved significantly, and arrived over 80% (see the proposed algorithm with $\alpha = 0.5$ and 0.75). When $\alpha = 0.25$, the proposed algorithm performed competitively with the PPO, since there is only a few knowledge transfer occurred across tasks. Moreover, we also compared the obtained results in this section against the results obtained in Sections IV-B and IV-C1, where the target tasks are simpler. As can be observed, the proposed algorithm is able to bring more significant improvements of the RL performance when the target task becomes more complex. As the basic RL algorithm is kept the same in all these experiments, the improvements achieved can be fully attributed to the proposed algorithm for TRL across heterogeneous RL tasks.

Moreover, Figs. 21 and 23 present the selection percentages of the source tasks in each learning episode of the proposed algorithm with $\alpha = 0.5$ and the PTF, over 20 independent runs, respectively. The results of the proposed algorithm and PTF




(a) Proposed algorithm (b) PTF

Fig. 23. Selection percentage of each source task in the proposed algorithm and PTF on the 4-link insert task with goal position $g'$, over 20 independent runs.

are similar as that discussed in Section IV-C1, which further demonstrated that the proposed algorithm is able to identify the proper source tasks in heterogeneous TRL while the RL progresses online in the target task.

Lastly, the averaged results on all the insert tasks are summarized in Table IV, which are similar as that described in Section IV-B.

## V. CONCLUSION

In this article, considering the scenario of knowledge transfer across multiple heterogeneous RL tasks for enhanced RL performance, we have proposed a new heterogeneous TRL algorithm with adaptive policy gradient transfer. In particular, the proposed method includes two modules. One is the *source task selection module*, which is responsible to select the useful source tasks from multiple heterogeneous source tasks. The other is the *knowledge transfer module*, which transfers knowledge from the selected source task to a heterogeneous target task to accelerate the learning process of the RL agent. In contrast to existing methods, the proposed method is able to perform online source task selection efficiently even the source and target tasks are with heterogeneous state-action spaces. Moreover, the proposed knowledge transfer across heterogeneous RL tasks does not rely on the learning of common representation or task mapping between RL tasks, which thus will not bring much computational burden in the RL process. To evaluate the performance of the proposed heterogeneous TRL algorithm, comprehensive experimental studies have been conducted on the well-known continuous and heterogeneous robotic RL tasks, i.e., Reacher and Insert with different number of arms, against both the RL algorithm without knowledge transfer and the existing state-of-the-art TRL algorithm. The obtained results confirmed the efficacy of the proposed TRL algorithm in enhancing the RL performance across multiple heterogeneous RL tasks.

In the future, we would like to explore the heterogeneous TRL approach with adaptive control between knowledge transfer

and self-learning from the environment to further improve the efficacy of the proposed algorithm in complex problem-solving.

## REFERENCES


[1] R.S. Sutton and A.G. Barto, "Reinforcement learning: An introduction," *Robotica 17*, no. 2,pp. 229–235.

[2] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," 2017, *arXiv:1708.05866*.

[3] Y. Li, "Deep reinforcement learning: An overview," 2017, *arXiv:1701.07274*.

[4] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, "Optimal and autonomous control using reinforcement learning: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2042–2062, Jun. 2018.

[5] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: A survey," in *Proc. IEEE Symp. Ser. Comput. Intell.*, 2020, pp. 737–744.

[6] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[7] Z. Wang et al., "Sample efficient actor-critic with experience replay," 2016, *arXiv:1611.01224*.

[8] K. Shao, Y. Zhu, and D. Zhao, "Starcraft micromanagement with reinforcement learning and curriculum transfer learning," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 3, no. 1, pp. 73–84, Feb. 2019.

[9] A. E. L. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electron. Imag.*, vol. 2017, no. 19, pp. 70–76, 2017.

[10] B. R. Kiran et al., "Deep reinforcement learning for autonomous driving: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 4909–4926, Jun. 2022.

[11] Z. Pan, L. Wang, J. Wang, and J. Lu, "Deep reinforcement learning based optimization algorithm for permutation flow-shop scheduling," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 7, no. 4, pp. 983–994, Aug. 2023.

[12] Y. Du, J.-Q. Li, X.-L. Chen, P.-Y. Duan, and Q.-K. Pan, "Knowledge-based reinforcement learning and estimation of distribution algorithm for flexible job shop scheduling problem," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 7, no. 4, pp. 1036–1050, Aug. 2023.

[13] K. C. Tan, L. Feng, and M. Jiang, "Evolutionary transfer optimization - a new frontier in evolutionary computation research," *IEEE Comput. Intell. Mag.*, vol. 16, no. 1, pp. 22–33, Feb. 2021.

[14] M. Jiang, Z. Wang, L. Qiu, S. Guo, X. Gao, and K. C. Tan, "A fast dynamic evolutionary multiobjective algorithm via manifold transfer learning," *IEEE Trans. Cybern.*, vol. 51, no. 7, pp. 3417–3428, Jul. 2021.

[15] L. Feng et al., "Evolutionary multitasking via explicit autoencoding," *IEEE Trans. Cybern.*, vol. 49, no. 9, pp. 3457–3470, Sep. 2019.

[16] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *J. Mach. Learn. Res.*, vol. 10, no. 7, pp. 1633–1685,2009.

[17] A. Lazaric, "Transfer in reinforcement learning: A framework and a survey," in *Reinforcement Learning*. Berlin, Germany: Springer, 2012, pp. 143–173.

[18] T. Zhang, Z. Liu, Z. Pu, and J. Yi, "Automatic curriculum learning for large-scale cooperative multiagent systems," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 7, no. 3, pp. 912–930, Jun. 2023.

[19] Z. Zhu, K. Lin, A. K. Jain, and J. Zhou, "Transfer learning in deep reinforcement learning: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 11, pp. 13344–13362, Nov. 2023.

[20] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1312–1320.

[21] F. Fernández and M. Veloso, "Probabilistic policy reuse in a reinforcement learning agent," in *Proc. 5th Int. Joint Conf. Auton. Agents Multiagent Syst.*, 2006, pp. 720–727.

[22] T. Hester et al., "Deep Q-learning from demonstrations," in *Proc. AAAI Conf. Artif. Intell. 32nd AAAI Conf. Arti. Intell. 30th Innov. Appl. Arti. Intell. Conf. 8th AAAI Symp. Educ. Adv. Artif. Intell.*, pp. 3223–3230.

[23] W. M. Czarnecki, R. Pascanu, S. Osindero, S. Jayakumar, G. Swirszcz, and M. Jaderberg, "Distilling policy distillation," in *Proc. 22nd Int. Conf. Artif. Intell. Statist.*, 2019, pp. 1331–1340.

[24] J. Andreas, D. Klein, and S. Levine, "Modular multitask reinforcement learning with policy sketches," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 166–175.

[25] M. E. Taylor and P. Stone, "Cross-domain transfer for reinforcement learning," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 879–886.

[26] T. Hester et al., "Deep Q-learning from demonstrations," in *Proc. 32nd AAAI Conf. Artif. Intell. 30th Innov. Appl. Artif. Intell. Conf. 8th AAAI Symp. Educ. Adv. Artif. Intell.*, Feb. 2018.

[27] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine, "Learning invariant feature spaces to trans-fer skills with reinforcement learning," 2017, *arXiv:1703.02949*.

[28] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

[29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[30] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, 1996.

[31] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[32] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep Q-learning with model-based acceleration," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2829–2838.

[33] V. Mnih et al., "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.

[34] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2015, *arXiv:1506.02438*.

[35] Ronald J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. 3, pp. 229–256, 1992.

[36] G. Tesauro et al., "Temporal difference learning and TD-Gammon," *Commun. ACM*, vol. 38, no. 3, pp. 58–68, 1995.

[37] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 1008–1014.

[38] T. Degris, M. White, and R. S. Sutton, "Off-policy actor-critic," 2012.

[39] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.

[40] B. Rosman, M. Hawasly, and S. Ramamoorthy, "Bayesian policy reuse," *Mach. Learn.*, vol. 104, no. 1, pp. 99–127, 2016.

[41] A. Kai, M.P Deisenroth, M. Brundage, and A.A Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag. 34*, no. 6, pp. 26–38, 2017.

[42] A. A. Rusu et al., "Policy distillation," 2015, *arXiv:1511.06295*.

[43] E. Parisotto, J. L. Ba, and R. Salakhutdinov, "Actor-mimic: Deep multitask and transfer reinforcement learning," 2015, *arXiv:1511.06342*.

[44] Y. Teh et al., "Distral: Robust multitask reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.31st Int. Conf. Neural Inf. Process. Syst.*, Dec. 2017, pp. 4499–4509.

[45] B. Kang, Z. Jie, and J. Feng, "Policy optimization with demonstrations," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 2469–2478.

[46] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 6292–6299.

[47] G. Joshi and G. Chowdhary, "Cross-domain transfer in reinforcement learning using target apprentice," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 7525–7532.

[48] J. Rajendran, A. Srinivas, M. M. Khapra, P. Prasanna, and B. Ravindran, "Attend, adapt and transfer: Attentive deep architecture for adaptive transfer from multiple sources in the same domain," 2015, *arXiv:1510.02879*.

[49] S. Li, F. Gu, G. Zhu, and C. Zhang, "Context-aware policy reuse," 2018, *arXiv:1806.03793*.

[50] S. Li and C. Zhang, "An optimal online method of selecting source policies for reinforcement learning," in *Proc. 32nd AAAI Conf. Artif. Intell. 30th Innov. Appl. Artif. Intell. Conf. 8th AAAI Symp. Educ. Adv. Artif. Intell.*, Feb. 2018, pp. 3562–3570.

[51] T. Yang et al., "Efficient deep reinforcement learning via adaptive policy transfer," in *Proc. 29th Int. Conf. Int. Joint Conf. Artif. Intell.*,2021.

[52] G. Cheng, L. Dong, W. Cai, and C. Sun, "Multi-task reinforcement learning with attention-based mixture of experts," *IEEE Robot. Automat. Lett.*, vol. 8, no. 6, pp. 3812–3819, Jun. 2023.

[53] H. B. Ammar et al., "An automated measure of MDP similarity for transfer in reinforcement learning," in *Proc. Workshops 28th AAAI Conf. Artif. Intell.*, 2014, pp. 31–37.

[54] Álvaro Visús, J. García, and F. Fernández, "A taxonomy of similarity metrics for Markov decision processes," 2021, *arXiv:2103.04706*.

[55] James L. Carroll and K. Seppi, "Task similarity measures for transfer in reinforcement learning task libraries," in *Proc IEEE Int. Joint Conf. Neural Netw.*, 2005, vol. 2, pp. 803–808.
[56] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

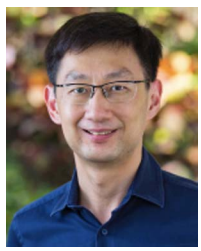
**Gengzhi Zhang** received the B.E. degree in 2018 from the school of Big Data & Software Engineering, Chongqing University, Chongqing, China, where she is currently working toward the Ph.D. degree in College of Computer Science, Chongqing University. Her current research interests include reinforcement learning, transfer reinforcement learning, and multitask reinforcement learning.

**Liang Feng** received the Ph.D. degree from the School of Computer Engineering, Nanyang Technological University, Singapore, in 2014. He is currently a Professor with the College of Computer Science, Chongqing University, Chongqing, China. His research interests mainly include computational and artificial intelligence, memetic computing, Big Data optimization and learning, as well as transfer learning and optimization. He has been honored with the 2019 IEEE TEVC Outstanding Article Award, 2023 IEEE TETCI Outstanding Article Award, and 2024 IEEE CIM Outstanding Article Award. He is an Associate Editor for IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTATIONAL INTELLIGENCE, IEEE CIM, Memetic Computing. He is also the founding Chair of the IEEE CIS Intelligent Systems Applications Technical Committee Task Force on Transfer Learning & Transfer Optimization.

**Yu Wang** received the Ph.D. degree from the School of Information Science and Technology, University of Science and Technology of China, Hefei, China. He is currently an Algorithm Scientist and Senior Director with Jingdong Retail, Beijing. He has been deeply engaged in the field of general AI for many years. He has authored or coauthored more than 50 academic papers and more than 170 patents till now. His research interests include operations optimization, machine learning, and data mining. He has been dedicated to the industrialization of artificial intelligence for many years, and received significant achievements in e-commerce, intelligent logistics, automotive services, and Big Data platforms.

**Min Li** received the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, China. He is currently a Retail Algorithm Scientist with Jingdong Retail and responsible for the construction of intelligent user growth system. He has authored or coauthored more than 20 international conference and journal articles in the field of artificial intelligence, more than 70 patents and eight U.S. patents. After graduated in 2010, he was a Cognitive Computing Researcher of IBM China Research Institute, Algorithm Expert of Alibaba Search Division, and Chief Algorithm Engineer of Didi. He has been deeply engaged in general AI field for many years, including large-scale optimization, machine learning, image recognition and other fields.

**Hong Xie** (Member, IEEE) received the B.Eng. degree from the School of Computer Science and Technology from The University of Science and Technology of China, Hefei, China, in 2010, and the Ph.D. degree from the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, in 2015, proudly under the supervision of Prof. John C.S. Lui. He is currently a Researcher with the Big Data Research Center, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Beijing, China. He was was a Postdoctoral Research Fellow with the Department of Computing Science and Engineering, The Chinese University of Hong Kong (CUHK) hosted by Prof. John C.S. Lui, and a Postdoctoral Research Fellow with the School of Computing, Nationial University of Singapore, Singapore, hosted by Prof. Richard T.B. Ma. He was also a Faculty Member with Chongqing University. He is a member of CCF, a member of ACM.



**Kay Chen Tan** (Fellow, IEEE) received the B.Eng. (first class Hons.) and the Ph.D. degrees from the University of Glasgow, Glasgow, U.K., in 1994 and 1997, respectively. He is currently a Chair Professor of the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. He was an Editor-in-Chief of IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION from 2015 to 2020, and *IEEE Computational Intelligence Magazine* from 2010 to 2013, and currently is an Editorial Board member of more than ten journals. He is currently the Vice-President (Publications) of IEEE Computational Intelligence Society, an Honorary Professor with the University of Nottingham, Nottingham, U.K., and the Chief Co-Editor of Springer Book Series on Machine Learning: Foundations, Methodologies, and Applications.