MULTIPOLAR: Multi-Source Policy Aggregation for Transfer Reinforcement Learning between Diverse Environmental Dynamics

Mohammadamin Barekatain[1,2*] , **Ryo Yonetani**[1] and **Masashi Hamaya**[1]
[1]OMRON SINIC X
[2]Technical University of Munich
m.barekatain@tum.de, {ryo.yonetani, masashi.hamaya}@sinicx.com,

Abstract

Transfer reinforcement learning (RL) aims at improving the learning efficiency of an agent by exploiting knowledge from other source agents trained on relevant tasks. However, it remains challenging to transfer knowledge between different environmental dynamics without having access to the source environments. In this work, we explore a new challenge in transfer RL, where only a set of source policies collected under diverse unknown dynamics is available for learning a target task efficiently. To address this problem, the proposed approach, MULTI-source POLicy AggRegation (MULTIPOLAR), comprises two key techniques. We learn to aggregate the actions provided by the source policies adaptively to maximize the target task performance. Meanwhile, we learn an auxiliary network that predicts residuals around the aggregated actions, which ensures the target policy's expressiveness even when some of the source policies perform poorly. We demonstrated the effectiveness of MULTIPOLAR through an extensive experimental evaluation across six simulated environments ranging from classic control problems to challenging robotics simulations, under both continuous and discrete action spaces. The demo videos and code are available on the project webpage: https://omron-sinicx.github.io/multipolar/.

1 Introduction

We envision a future scenario where a variety of robotic systems, which are each trained or manually engineered to solve a similar task, provide their policies for a new robot to learn a relevant task quickly. For example, imagine various pick-and-place robots working in factories all over the world. Depending on the manufacturer, these robots will differ in their kinematics (*e.g.*, link length, joint orientation) and dynamics (*e.g.*, link mass, joint damping, friction, inertia). They could provide their policies to a new robot [Devin *et al.*, 2017], even though their dynamics factors, on which the policies are implicitly conditioned, are not typically available [Chen *et al.*, 2018]. Moreover, we cannot rely on a history of their individual experiences, as they may be unavailable due to a lack of communication between factories or prohibitively large dataset sizes. In such scenarios, we argue that a key technique to develop is the ability to transfer knowledge from a collection of robots to a new robot quickly *only by exploiting their policies while being agnostic to their different kinematics and dynamics*, rather than collecting a vast amount of samples to train the new robot from scratch.

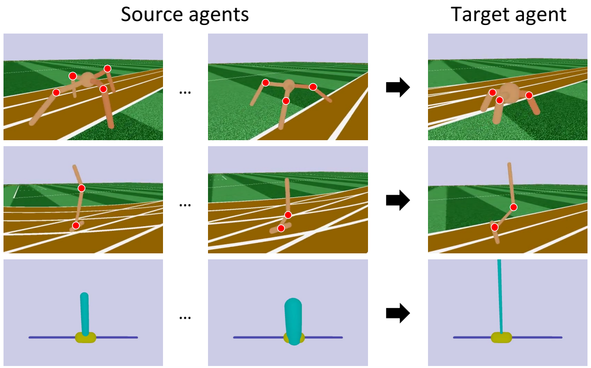

Figure 1: Transfer RL between diverse environmental dynamics. We aim to train a policy of a target agent efficiently *by utilizing the policies of other source agents* under different unknown environmental dynamics. Some joints are annotated by red circles to highlight kinematic diversity.

The scenario illustrated above poses a new challenge in the transfer learning for reinforcement learning (RL) domains. Formally, consider multiple instances of a single environment with diverse state transition dynamics, *e.g.*, independent robots presented in Figure 1, which reach different states by executing the same actions due to the differences in their kinematics and dynamics designs. Some source agents interacting with one of the environment instances provide their deterministic policy to a new target agent in another environment instance. Then, our problem is: *can we efficiently learn the policy of a target agent given only the collection of source policies?* Note that information about source environmental dynamics, such as the exact state transition distributions and the history of environmental states, will not be visible to the target agent as mentioned above. Also, the

[*] Work done as an intern at OMRON SINIC X

source policies are neither trained nor hand-engineered for the target environment instance, and therefore not guaranteed to work optimally and may even fail [Chen *et al.*, 2018]. Importantly, these conditions prevent us from adopting existing works on transfer RL between different environmental dynamics, as they require access to source environment instances or their dynamics for training a target policy (*e.g.*, [Lazaric *et al.*, 2008; Chen *et al.*, 2018; Yu *et al.*, 2019; Tirinzoni *et al.*, 2018; Parisotto *et al.*, 2016]). Similarly, meta-learning approaches such as [Vanschoren, 2018; Clavera *et al.*, 2019], cannot be used here because they typically train an agent on a diverse set of tasks (*i.e.*, environment instances). Also, existing techniques that utilize a collection of source policies, *e.g.*, policy reuse frameworks [Fernández and Veloso, 2006; Rosman *et al.*, 2016; Zheng *et al.*, 2018] and option frameworks [Sutton *et al.*, 1999; Bacon *et al.*, 2017; Mankowitz *et al.*, 2018], are not a promising solution because, to our knowledge, they assume that source policies have the same environmental dynamics but different goals. The most relevant work is “attend, adapt, and transfer” (A2T) method [Rajendran *et al.*, 2017] that learns to attend multiple source policies to enable selective transfer. However, this method assumes access to the action probability distribution of source policies (*i.e.*, stochastic source policies), making it hard to address our problem where only deterministic actions of source policies are available (*i.e.*, deterministic source policies). Besides, their presented method is tested only in environments with a discrete action space.

As a solution to the problem, we propose a new transfer RL approach named **MULTI-source POLicy AggRegation (MULTIPOLAR)**. As shown in Figure 2, our key idea is twofold; 1) In a target policy, we adaptively aggregate the deterministic actions produced by a collection of source policies. By learning aggregation parameters to maximize the expected return at a target environment instance, we can better adapt the aggregated actions to unseen environmental dynamics of the target instance without knowing source environmental dynamics nor source policy performances. 2) We also train an auxiliary network that predicts a residual around the aggregated actions, which is crucial for ensuring the expressiveness of the target policy even when some source policies are not useful. As another notable advantage, the proposed MULTIPOLAR can be used for both continuous and discrete action spaces with few modifications while allowing a target policy to be trained in a principled fashion.

We evaluate MULTIPOLAR in a variety of environments ranging from classic control problems to challenging robotics simulations. Our experimental results demonstrate the significant improvement of sample efficiency with the proposed approach, compared to baselines learning from scratch, as well as leveraging a single source policy and A2T. We also conducted a detailed analysis of our approach and found it worked well even when some of the source policies performed poorly in their original environment instance.

Main contributions: (1) a new transfer RL problem that leverages multiple source policies collected under diverse unknown environmental dynamics to train a target policy in another dynamics, and (2) MULTIPOLAR, a principled and effective solution verified in our extensive experiments.

2 Preliminaries

Reinforcement Learning. We formulate our problem under the standard RL framework [Sutton and Barto, 1998], where an agent interacts with its environment modeled by a Markov decision process (MDP). An MDP is represented by the tuple $\mathcal{M} = (\rho_0, \gamma, \mathcal{S}, \mathcal{A}, R, T)$ where ρ_0 is the initial state distribution and γ is a discount factor. At each timestep t, given the current state $s_t \in \mathcal{S}$, the agent executes an action $a_t \in \mathcal{A}$ based on its policy $\pi(a_t \mid s_t; \theta)$ parameterized by θ. Importantly, in this work, we consider both cases of continuous and discrete for action space $\mathcal{A}$. The environment returns a reward $R(s_t, a_t) \in \mathbb{R}$ and transitions to the next state s_{t+1} based on the state transition distribution $T(s_{t+1} \mid s_t, a_t)$. In this framework, RL aims to maximize the expected return with respect to the policy parameters θ.

Environment Instances. Similar to some prior works on transfer RL [Song *et al.*, 2016; Tirinzoni *et al.*, 2018; Yu *et al.*, 2019], we consider K instances of the same environment which differ only in their state transition dynamics. Namely, we model each environment instance by an indexed MDP: $\mathcal{M}_i = (\rho_0, \gamma, \mathcal{S}, \mathcal{A}, R, T_i)$ where no two state transition distributions $T_i, T_j; i \neq j$ are identical. Unlike the prior works, we assume that *each T_i is unknown when training a target policy*, *i.e.*, agents cannot access the exact form of T_i nor a collection of states sampled from T_i.

Source Policies. For each of the K environment instances, we are given a deterministic source policy $\mu_i : \mathcal{S} \to \mathcal{A}$ that only maps states to actions. Each source policy μ_i can be either parameterized (*e.g.*, learned by interacting with its environment instance $\mathcal{M}_i$) or non-parameterized (*e.g.*, heuristically designed by humans). Either way, we assume that *no prior knowledge about the source policies μ_i's is available for a target agent, such as their representations or original performances, except that they were acquired from a source environment instance $\mathcal{M}_i$ with an unknown T_i*. This is one of the assumptions that make our problem unique from others, such as policy reuse and option frameworks.

Problem Statement. Given the set of source policies $L = \{\mu_1, \ldots, \mu_K\}$, our goal is to train a new target agent's policy $\pi_{\text{target}}(a_t \mid s_t; L, \theta)$ in a sample efficient fashion, where the target agent interacts with another environment instance $\mathcal{M}_{\text{target}} = (\rho_0, \mathcal{S}, \mathcal{A}, R, T_{\text{target}})$ and T_{target} is not identical to the source T_i $(i = 1 \ldots, K)$ due to their distinct dynamics.

3 Multi-Source Policy Aggregation

As shown in Figure 2, with the Multi-Source Policy Aggregation (MULTIPOLAR), we formulate a target policy π_{target} using a) the adaptive aggregation of deterministic actions from the set of source policies L, and b) the auxiliary network predicting residuals around the aggregated actions. We first present our method for the continuous action space, and then extend it to the discrete space.

Adaptive Aggregation of Source Policies. Let us denote by $a_t^{(i)} = \mu_i(s_t)$ the action predicted deterministically by source policy μ_i given the current state s_t. For the continuous

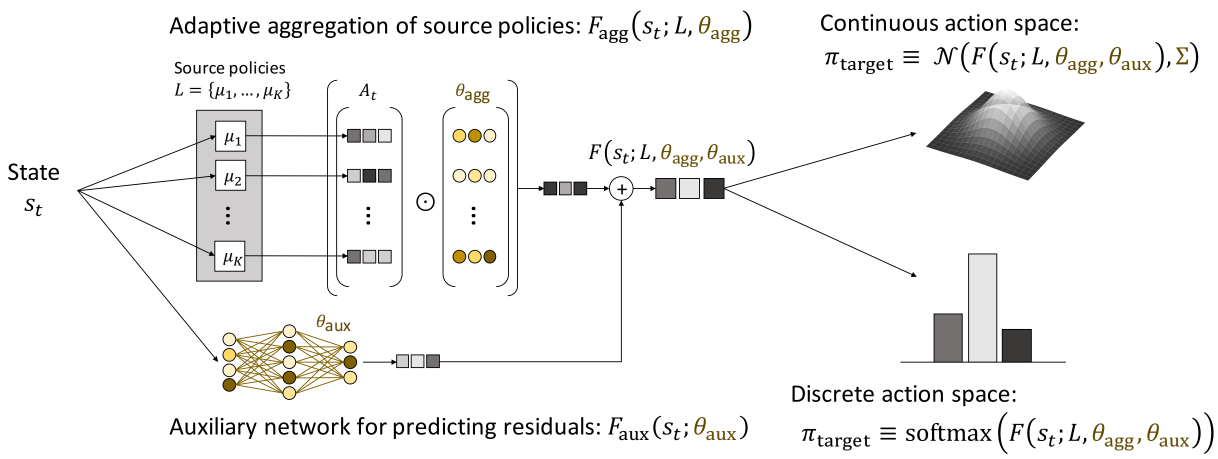
Adaptive aggregation of source policies: $F_{\text{agg}}(s_t; L, \theta_{\text{agg}})$
Continuous action space:
$\pi_{\text{target}} \equiv \mathcal{N}(F(s_t; L, \theta_{\text{agg}}, \theta_{\text{aux}}), \Sigma)$
Source policies
$L = \{\mu_1, \ldots, \mu_K\}$
A_t
θ_{agg}
State
s_t
μ_1
μ_2
μ_K
$\odot$
$F(s_t; L, \theta_{\text{agg}}, \theta_{\text{aux}})$
θ_{aux}
Discrete action space:
Auxiliary network for predicting residuals: $F_{\text{aux}}(s_t; \theta_{\text{aux}})$
$\pi_{\text{target}} \equiv \text{softmax}(F(s_t; L, \theta_{\text{agg}}, \theta_{\text{aux}}))$

Figure 2: Overview of MULTIPOLAR. We formulate a target policy π_{target} with the sum of 1) the adaptive aggregation F_{agg} of deterministic actions from source policies L and 2) the auxiliary network F_{aux} for predicting residuals around F_{agg}.

action space, $a_t^{(i)} \in \mathbb{R}^D$ is a D-dimensional real-valued vector representing D actions performed jointly in each timestep. For the collection of source policies L, we derive the matrix of their deterministic actions:

$$A_t = \left[(a_t^{(1)})^\top, \ldots, (a_t^{(K)})^\top \right] \in \mathbb{R}^{K \times D}. \tag{1}$$

The key idea of this work is to aggregate A_t adaptively in an RL loop, *i.e.*, to maximize the expected return. This adaptive aggregation gives us a "baseline" action that could introduce a strong inductive bias in the training of a target policy *even without knowing each source environmental dynamics* T_i. As mentioned earlier, other recent methods that enable transfer between different dynamics, by contrast, require access to the source environmental dynamics or related physical parameters [Chen *et al.*, 2018; Yu *et al.*, 2019].

More specifically, we define the adaptive aggregation function $F_{\text{agg}} : \mathcal{S} \rightarrow \mathcal{A}$ that produces the baseline action based on the current state s_t as follows:

$$F_{\text{agg}}(s_t; L, \theta_{\text{agg}}) = \frac{1}{K} \mathbb{1}^K \left(\theta_{\text{agg}} \odot A_t \right), \tag{2}$$

where $\theta_{\text{agg}} \in \mathbb{R}^{K \times D}$ is a matrix of trainable parameters, $\odot$ is the element-wise multiplication, and $\mathbb{1}^K$ is the all-ones vector of length K. θ_{agg} is neither normalized nor regularized, and can scale each action of each policy independently. This means that unlike A2T [Rajendran *et al.*, 2017] method, we do not merely adaptively interpolate action spaces, but more flexibly emphasize informative source actions while suppressing irrelevant ones. Furthermore, in contrast to A2T method that trains a new network from scratch to map states to aggregation weights, our approach directly estimates θ_{agg} in a state-independent fashion, as differences in environmental dynamics would affect optimal actions in the entire state space rather than a part of it.

Predicting Residuals around Aggregated Actions. Moreover, we learn auxiliary network $F_{\text{aux}} : \mathcal{S} \rightarrow \mathcal{A}$ jointly with F_{agg}, to predict residuals around the aggregated actions. F_{aux} is used to improve the target policy training in two ways. 1) If the aggregated actions from F_{agg} are already useful in the target environment instance, F_{aux} will correct them for a higher expected return. 2) Otherwise, F_{aux} learns the target task while leveraging F_{agg} as a prior to have a guided exploration process. Any network could be used for F_{aux} as long as it is parameterized and fully differentiable. Finally, the MULTIPOLAR function is formulated as:

$$F(s_t; L, \theta_{\text{agg}}, \theta_{\text{aux}}) = F_{\text{agg}}(s_t; L, \theta_{\text{agg}}) + F_{\text{aux}}(s_t; \theta_{\text{aux}}), \tag{3}$$

where θ_{aux} denotes the set of trainable parameters of F_{aux}. Note that the idea of predicting residuals for a source policy has also been presented in [Silver *et al.*, 2018; Johannink *et al.*, 2019]. The main difference here is that, while these works just *add* raw action outputs provided from a *single* hand-engineered source policy, we adaptively aggregate actions from multiple source policies to exploit them flexibly.

Target Policy. Target policy π_{target} can be modeled by reparameterizing the MULTIPOLAR function as a Gaussian distribution, *i.e.*, $\mathcal{N}(F(s_t; L, \theta_{\text{agg}}, \theta_{\text{aux}}), \Sigma)$, where Σ is a covariance matrix estimated based on what the used RL algorithm requires. Since we regard $\mu_i \in L$ as fixed functions mapping states to actions, this Gaussian policy π_{target} is differentiable with respect to θ_{agg} and θ_{aux}, and hence could be trained with any RL algorithm that explicitly updates policy parameters. Unlike [Silver *et al.*, 2018; Johannink *et al.*, 2019], we can formulate the target policy in a principled fashion for actions in a discrete space. Specifically, instead of a D-dimensional real-valued vector, here we have a D-dimensional one-hot vector $a_t^{(i)} \in \{0, 1\}^D$, $\sum_j (a_t^{(i)})_j = 1$ as outputs of μ_i, where $(a_t^{(i)})_j = 1$ indicates that the j-th action is to be executed. Following Eqs. (2) and (3), the output of $F(s_t; L, \theta_{\text{agg}}, \theta_{\text{aux}})$ can be viewed as D-dimensional un-normalized action scores, from which we can sample a discrete action after normalizing it by the softmax function.

4 Experimental Evaluation

We aim to empirically demonstrate the sample efficiency of a target policy trained with MULTIPOLAR (denoted by

"MULTIPOLAR policy"). To complete the experiments in a reasonable amount of time, we set the number of source policies to be $K = 4$ unless mentioned otherwise. Moreover, we investigate the factors that affect the performance of MULTIPOLAR. To ensure fair comparisons and reproducibility of experiments, we followed the guidelines introduced by [Henderson *et al.*, 2018] and [François-Lavet *et al.*, 2018] for conducting and evaluating all of our experiments.

4.1 Experimental Setup

Baseline Methods. To show the benefits of leveraging source policies, we compared our MULTIPOLAR policy to the standard multi-layer perceptron (MLP) trained from scratch, which is typically used in RL literature [Schulman *et al.*, 2017; François-Lavet *et al.*, 2018]. We also used MULTIPOLAR with $K = 1$, which is an extension of residual policy learning [Silver *et al.*, 2018; Johannink *et al.*, 2019] (denoted by "RPL") with adaptive residuals as well as the ability to deal with both continuous and discrete action spaces. As another baseline, we used A2T [Rajendran *et al.*, 2017] approach with the modification of replacing the actions sampled from stochastic source policies with deterministic source actions. We also made A2T employable in continuous action spaces by formulating the target policy in the same way as MULTIPOLAR explained in Section 3. We stress here that the existing transfer RL or meta RL approaches that train a universal policy network agnostic to the environmental dynamics, such as [Frans *et al.*, 2018; Chen *et al.*, 2018], cannot be used as a baseline since they require a policy to be trained on a distribution of environment instances, which is not possible in our problem setting. Also, other techniques using multiple source policies, such as policy reuse frameworks [Fernández and Veloso, 2006], [Parisotto *et al.*, 2016], and [Yu *et al.*, 2019], are not applicable because they require source policies to be collected under the target environmental dynamics or to be trained simultaneously under known source dynamics.

Environments. To show the general effectiveness of the MULTIPOLAR policy, we conducted comparative evaluations of MULTIPOLAR on the following six OpenAI Gym environments: Roboschool Hopper, Roboschool Ant, Roboschool InvertedPendulumSwingUp, Acrobot, CartPole, and LunarLander. We chose these six environments because 1) the parameterization of their dynamics and kinematics is flexible enough, 2) they cover discrete action space (Acrobot and CartPole) as well as continuous action space, and 3) they are samples of three distinct categories of OpenAI Gym environments, namely Box2d, Classic Control, and Roboschool. We used the Roboschool implementation of Hopper, Ant, and InvertedPendulumSwingup since they are based on an open-source engine, which makes it possible for every researcher to reproduce our experiments.

Experimental Procedure. For each of the six environments, we first created 100 environment instances by randomly sampling the dynamics and kinematics parameters from a specific range. For example, these parameters in the Hopper environment were link lengths, damping, friction, armature, and link mass with the sampling range defined similar to [Chen *et al.*, 2018]. Details of sampling ranges for dynamics and kinematics parameters of all six environments are presented in Tables 1, 2, 3, 4, 5, and 6. Note that we defined the sampling ranges for each environment such that the resulting environment instances are significantly different in dynamics. For this reason, a plain use of a source policy usually performed poorly in the target instance. Also, these parameters were used only for simulating environment instances and were not available when training a target policy. Then, for each environment instance, we trained an MLP policy that was used in two ways: a) the baseline MLP policy for each environment instance, and b) one of the 100 members of the source policy candidate pool from which we sample K of them to train MULTIPOLAR policies as well as A2T policies, and one of them to train RPL policies[1]. Specifically, for each environment instance, we trained three sets of MULTIPOLAR, A2T, and RPL policies each with distinct source policy sets selected randomly from the candidate pool. The learning procedure explained above was done three times with fixed different random seeds to reduce variance in results due to stochasticity. As a result, for each of the six environments, we had 100 environment instances × 3 random seeds = 300 experiments for MLP and 100 environment instances × 3 choices of source policies × 3 random seeds = 900 experiments for RPL, A2T, and MULTIPOLAR. The aim of this large number of experiments is to obtain correct insights into the distribution of performances [Henderson *et al.*, 2018]. Due to the large number of experiments for all the environments, our detailed analysis and ablation study of MULTIPOLAR components were conducted only in Hopper environment, as its sophisticated second-order dynamics plays a crucial role in agent performance [Chen *et al.*, 2018].

Kinematics		**Dynamics**	
Links	Length Range	Factors	Value Range
Pole	[0.1, 3]	Force	[6, 13]
		Gravity	[-14, -6]
		Poll mass	[0.1, 3]
		Cart mass	[0.3, 4]

Table 1: Sampling range for CartPole environment parameters.

Kinematics		**Dynamics**	
Links	Length Range	Factors	Value Range
Link 1	[0.3, 1.3]	Mass	[0.5, 1.5]
Link 2	[0.3, 1.3]	Center mass	[0.05, 0.95] × default length
		Inertia moments	[0.25, 1.5]

Table 2: Sampling ranges for Acrobot environment parameters.

Implementation Details. All the experiments were done using the Stable Baselines [Hill *et al.*, 2018] implementation of learning algorithms as well as its default hyperparameters and MLP network architecture for each environment. Based

[1] Although we used trained MLPs as source policies for reducing experiment times, any type of policies including hand-engineered ones could be used for MULTIPOLAR in principle.

Kinematics	
Links	Length Range
Side engine height	[10, 20]

Dynamics	
Factors	Value Range
Scale	[25, 50]
Initial Random	[500, 1500]
Main engine power	[10, 40]
Side engine power	[0.5, 2]
Side engine away	[8, 18]

Table 3: Sampling ranges for LunarLander environment parameters.

Kinematics	
Links	Length Range
Leg	[0.35, 0.65]
Foot	[0.29, 0.49]
Thigh	[0.35, 0.55]
Torso	[0.3, 0.5]

Dynamics	
Factors	Value Range
Damping	[0.5, 4]
Friction	[0.5, 2]
Armature	[0.5, 2]
Links mass	[0.7, 1.1] × default mass

Table 4: Sampling ranges for Hopper environment parameters.

on the performance of learning algorithms reported in [Hill *et al.*, 2018], all the policies were trained with Soft Actor-Critic [Haarnoja *et al.*, 2018] in the LunarLander environment and with Proximal Policy Optimization [Schulman *et al.*, 2017] in the rest of the environments. For fair comparisons, in all experiments, auxiliary network F_{aux} had an identical architecture to that of the MLP. Therefore, the only difference between MLP and MULTIPOLAR was the aggregation part F_{agg}, which made it possible to evaluate the contribution of transfer learning based on adaptive aggregation of source policies. In the same way as [Rajendran *et al.*, 2017], to implement the attention network branch of A2T, we used the same MLP architecture except for the last layer, which is activated with a softmax function. In all the experiments, source policies are the same for A2T and MULTIPOLAR to ensure an unbiased evaluation. We avoided any random seed optimization since it has been shown to alter the policies' performance [Henderson *et al.*, 2018]. As done by [Hill *et al.*, 2018], to have a successful training, we normalized rewards and input observations using their running average and standard deviation for all the environments except CartPole and LunarLander. Furthur, in all of the experiments, θ_{agg} is initialized to be the all-ones matrix. To run our experiments in parallel, we used GNU Parallel tool [Tange, 2018]. Finally, all the hyperparameters used for experiments on each environment can be found in our codebase, which is available on the project website: https://omron-sinicx.github.io/multipolar/.

Evaluation Metric. Following the guidelines of [Henderson *et al.*, 2018], to measure sampling efficiency of training policies, *i.e.*, how quick the training progresses, we used the average episodic reward over training samples. Also, to ensure that higher average episodic reward is representative of better performance and to estimate the variation of it, we used the sample bootstrap method to estimate statistically relevant 95% confidence bounds of the results of our experiments. Across all the experiments, we used 10K bootstrap iterations and the pivotal method[2].

Kinematics	
Links	Length Range
Legs	[0.4, 1.4] × default length

Dynamics	
Factors	Value Range
Damping	[0.1, 5]
Friction	[0.4, 2.5]
Armature	[0.25, 3]
Links mass	[0.7, 1.1] × default mass

Table 5: Sampling range for Ant environment parameters.

Kinematics	
Links	Length Range
Pole	[0.2, 2]

Dynamics	
Factors	Value Range
Damping	[0.1, 5]
Friction	[0.5, 2]
Armature	[0.5, 3]
Gravity	[-11, -7]
Links mass	[0.4, 3] × default mass

Table 6: Sampling ranges for InvertedPendulumSwingup environment parameters.

4.2 Results

Sample Efficiency. Figure 3 and Table 7 clearly show that on average, in all the environments, MULTIPOLAR outperformed baseline policies in terms of sample efficiency and sometimes the final episodic reward[3]. For example, in Hopper over 2M training samples, MULTIPOLAR with $K = 4$ achieved a mean of average episodic reward about three times higher than MLP (*i.e.*, training from scratch) and about twice higher than A2T and RPL. It is also noteworthy that MULTIPOLAR had always on par or better performance than RPL, which indicates the effectiveness of leveraging multiple source policies. Table 7 further suggests that although A2T is considerably more data efficient than learning from scratch in 3 out of 6 environments, it is substantially outperformed by MULTIPOLAR in all of the environments. We believe that the A2T performance limitation stems from 1) requiring training an extra network branch (attention) with the same size as the auxiliary network which almost doubles the number of parameters to be trained and 2) soft-attention mechanism that assigns a single weight for each policy resulting in merely *interpolation* of source policies and the auxiliary network, unlike our approach that flexibly aggregates each action of each source policy.

Ablation Study. To demonstrate the importance of each component of MULTIPOLAR, we evaluated the following degraded versions: (1) θ_{agg} *fixed to 1*, which just averages the deterministic actions from the source policies without adaptive weights (similar to the residual policy learning methods that use raw action outputs of a source policy), and (2) F_{aux}

[2] We used the Facebook Boostrapped implementation: github.com/facebookincubator/bootstrapped.

[3] Episodic rewards in Figure 3 are averaged over 3 random seeds and 3 random source policy sets on 100 environment instances. Table 7 reports the mean of this average over training samples.

Methods	CartPole	
	50K	100K
MLP	229 (220,237)	291 (282,300)
RPL	238 (231,245)	289 (282,296)
A2T (K=4)	230 (224,235)	281 (275,287)
MULTIPOLAR (K=4)	**252 (245,260)**	**299 (292,306)**
	Acrobot	
	100K	200K
MLP	-164 (-172,-156)	-111 (-117,-106)
RPL	-120 (-124,-116)	-98 (-101,-95)
A2T (K=4)	-201 (-208,-195)	-120 (-123, -116)
MULTIPOLAR (K=4)	**-117 (-121,-113)**	**-96 (-99,-93)**
	LunarLander	
	250K	500K
MLP	112 (104,121)	216 (210,221)
RPL	178 (174,182)	**246 (243,248)**
A2T (K=4)	128 (123,133)	226 (223,229)
MULTIPOLAR (K=4)	**181 (177,185)**	**246 (244,249)**

Methods	Roboschool Hopper	
	1M	2M
MLP	43 (42,45)	92 (88,96)
RPL	75 (70,79)	152 (142,160)
A2T (K=4)	58 (56,59)	126 (122,129)
MULTIPOLAR (K=4)	**138 (132,143)**	**283 (273,292)**
	Roboschool Ant	
	1M	2M
MLP	1088 (1030,1146)	1500 (1430,1572)
RPL	1120 (1088,1152)	1432 (1391,1473)
A2T (K=4)	1025 (990,1059)	1361 (1320,1402)
MULTIPOLAR (K=4)	**1397 (1361,1432)**	**1744 (1705,1783)**
	Roboschool InvertedPendulumSwingup	
	1M	2M
MLP	267 (260,273)	409 (401,417)
RPL	195 (192,198)	322 (317,326)
A2T (K=4)	310 (305,315)	458 (452,464)
MULTIPOLAR (K=4)	**476 (456,495)**	**588 (571,605)**

Table 7: Bootstrap mean and 95% confidence bounds of average episodic rewards over various training samples across six environments.

MULTIPOLAR (K=4)	1M	2M
Full version	**138 (132,143)**	**283 (273,292)**
θ_{agg} fixed to 1	118 (111,126)	237 (222,250)
F_{aux} learned independent of s_t	101 (95,108)	187 (175,200)

Table 8: Ablation study in Hopper.

MULTIPOLAR (K=4)	1M	2M
Random	138 (132,143)	283 (273,292)
4 high performance	**214 (208,220)**	**420 (409,430)**
2 high & 2 low performance	98 (94,102)	208 (200,215)
4 low performance	45 (44,47)	92 (88,95)

Table 9: Results with different source policy sampling in Hopper.

MULTIPOLAR	1M	2M
K=4	138 (132,143)	283 (273,292)
K=8	160 (154,167)	323 (312,335)
K=16	**177 (172,182)**	**357 (348,367)**

Table 10: Results with different number of source policies in Hopper.

learned independent of s_t, which replaces the state-dependent MLP with an adaptive "placeholder" parameter vector making actions a linear combination of source policy outputs. As shown in Table 8, the full version of MULTIPOLAR significantly outperformed the degraded ones, suggesting that adaptive aggregation and predicting residuals are both critical.

Effect of Source Policy Performances. Figure 4 shows the histograms of final episodic rewards (average rewards of the last 100 training episodes) for the source policy candidates obtained in their own original environment instances. As shown in the figure, the source policies were diverse in terms of the performance. In this setup, we investigate the effect of source policies performances on MULTIPOLAR sample efficiency. We created two separate pools of source policies, where one contained only high-performing and the other only low-performing ones[4]. Table 9 summarizes the results of sampling source policies from these pools (4 high, 2 high & 2 low, and 4 low performances) and compares them to the original MULTIPOLAR (shown as 'Random') also reported in Table 7. Not surprisingly, MULTIPOLAR performed the best when all the source policies were sampled from the high-performance pool. However, we emphasize that such high-quality policies are not always available in practice, due to the variability of how they are learned or hand-crafted under their own environment instance. Interestingly, by comparing the reported results in Table 2 (MLP) and Table 4 (4 low performance), we can observe that even in the worst case scenario of having only low performing source policies, the sample efficiency of MULTIPOLAR is on par with that of learning from scratch. This suggests that MULTIPOLAR avoids negative transfer, which occurs when transfer degrades the learning efficiency instead of helping it. Further, Figure 5 shows an example where MULTIPOLAR successfully learned to suppress the useless low-performing sources to maximize the expected return in a target task, indicating the mechanism of source policy rejection.

Effect of Number of Source Policies. Finally, we show how the number of source policies contributes to MULTIPOLAR's sample efficiency in Table 10. Specifically, we trained MULTIPOLAR policies up to $K = 16$ to study how the mean

[4] Here, policies with final episodic reward over 2K are regarded as high-performing and below 1K are as low-performing.

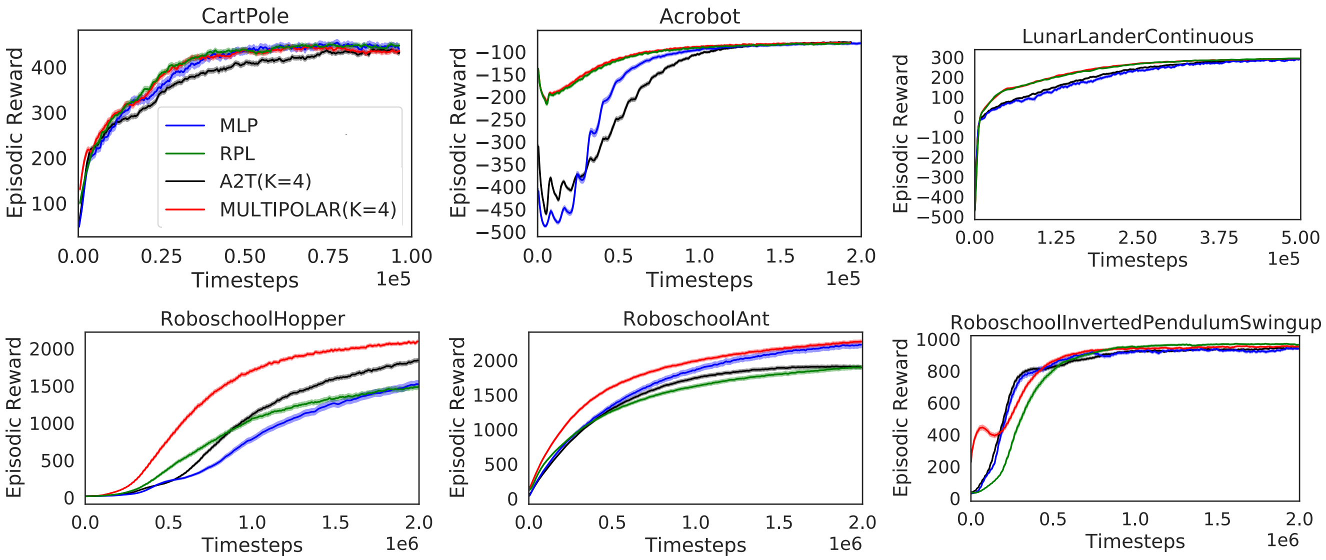

Figure 3: Average learning curves of MLP, RPL, A2T ($K = 4$), and MULTIPOLAR ($K = 4$) over all the experiments for each environment. The shaded area represents 1 standard error.

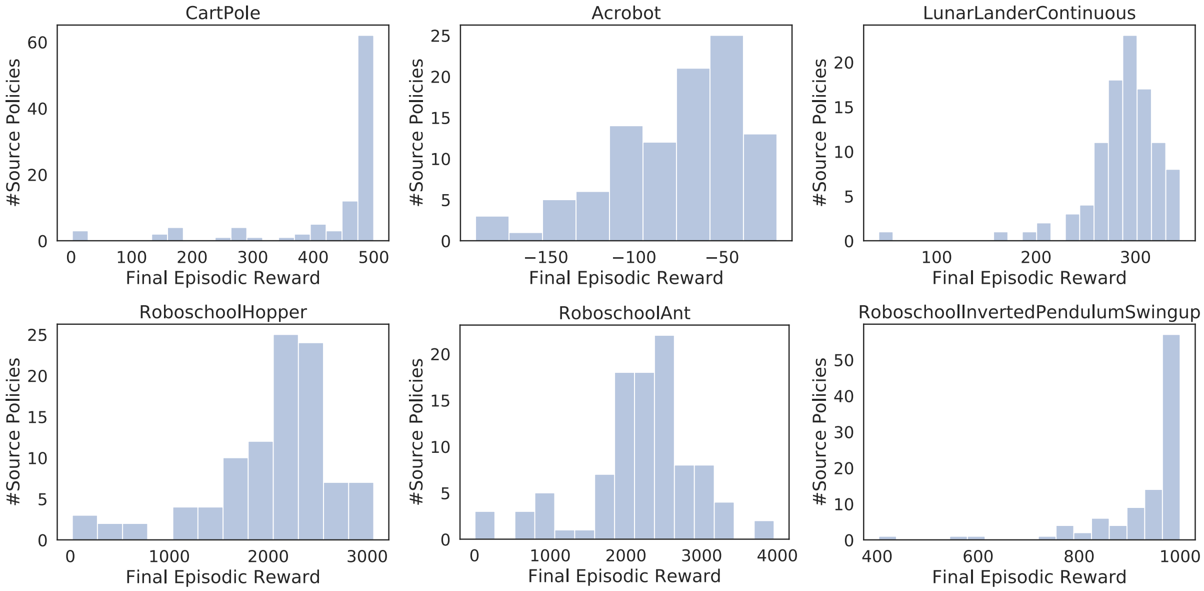

Figure 4: Histogram of final episodic rewards obtained by source policies in their original environment instances.

of average episodic rewards changes. The monotonic performance improvement over K (for $K \leq 16$), is achieved at the cost of increased training and inference time. In practice, we suggest balancing this speed-performance trade-off by using as many source policies as possible before reaching the inference time limit required by the application.

5 Discussion and Related Work

In this section, we highlight how our work is different from the existing approaches and also discuss the future directions.

Transfer between Different Dynamics. Our work is broadly categorized as an instance of transfer RL between different environmental dynamics, in which a policy for a target task is trained using information collected from source tasks. Much related work requires training samples collected from source tasks, which are then used for measuring the similarity between source and target environment instances [Lazaric *et al.*, 2008; Tirinzoni *et al.*, 2018] or conditioning a target policy to predict actions [Chen *et al.*, 2018]. Alternative means to quantify the similarity is to use a full specification of MDPs [Song *et al.*, 2016; Wang *et al.*, 2019] or environmental dynamics [Yu *et al.*, 2019]. The work of [Parisotto *et al.*,

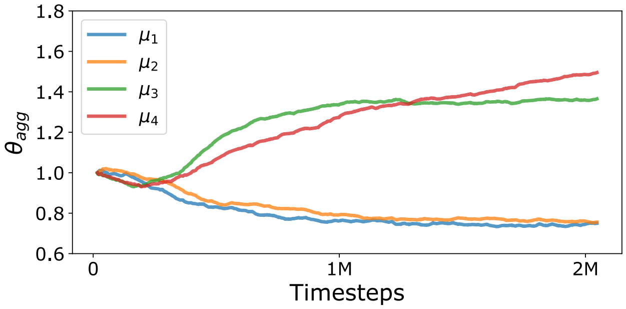

Figure 5: Aggregation parameters θ_{agg} averaged over the three actions of Hopper environment during the training, where source policies μ_1, μ_2 are low-performing and μ_3, μ_4 are high-performing.

2016] presented a multi-task learning framework to enable transfer between different games. However, it requires a target policy to be pre-trained with a variety of source tasks. In contrast, the proposed MULTIPOLAR allows the knowledge transfer only through the policies acquired from source environment instances with diverse unknown dynamics, which is beneficial when source and target environments are not always connected to exchange information about their dynamics and training samples.

Leveraging Multiple Policies. The idea of utilizing multiple source policies can be found in the literature of policy reuse frameworks [Fernández and Veloso, 2006; Rosman *et al.*, 2016; Li and Zhang, 2018; Zheng *et al.*, 2018; Li *et al.*, 2019]. The basic motivation behind these works is to provide “nearly-optimal solutions” [Rosman *et al.*, 2016] for short-duration tasks by reusing one of the source policies, where each source would perform well on environment instances with different rewards (*e.g.*, different goals in maze tasks). In our problem setting, where environmental dynamics behind each source policy are different, reusing a single policy is not the right approach as described in [Chen *et al.*, 2018]. Even with leveraging multiple source policies by a convex combination where the weights are predicted from an additional network learned from scratch [Rajendran *et al.*, 2017], the performance is still limited without effective and flexible source policy aggregation, as demonstrated in our experiments. Another relevant idea is hierarchical RL [Kulkarni *et al.*, 2016; Osa *et al.*, 2019] that involves a hierarchy of policies (or action-value functions) to enable temporal abstraction. In particular, option frameworks [Sutton *et al.*, 1999; Bacon *et al.*, 2017; Mankowitz *et al.*, 2018] make use of a collection of policies as a part of “options”. However, they assumed all the policies in the hierarchy to be learned in a single environment instance. Another relevant work along this line of research is [Frans *et al.*, 2018], which meta-learns a hierarchy of multiple sub-policies by training a master policy over the distribution of tasks. Nevertheless, hierarchical RL approaches are not useful for leveraging multiple source policies each acquired under diverse environmental dynamics.

Learning Residuals in RL. Some recent works adopt residual learning to mitigate the limited performance of hand-engineered policies [Silver *et al.*, 2018; Johannink *et al.*, 2019]. We are interested in a more extended scenario where various source policies with unknown performances are provided instead of a single sub-optimal policy. Also, these approaches focus only on robotic tasks in the continuous action space, while our approach could work on both of continuous and discrete action spaces in a broad range of environments.

Future Directions. In our experiments, we confirmed the effectiveness of our proposed MULTIPOLAR with up to 16 source policies. An important question is up to how many source policies will eventually saturate the performance gain of MULTIPOLAR? Another interesting direction is adapting MULTIPOLAR to combinatorial optimization problems [Bello *et al.*, 2016] (with heuristic approaches as source policies) as well as involving other types of environmental differences, such as dissimilar reward functions and state/action spaces.

6 Conclusion

We presented a new problem setting of transfer RL which aims to train a policy efficiently using a collection of source policies acquired under diverse environmental dynamics. To this end, we proposed MULTIPOLAR that adaptively aggregates a set of actions provided by the source policies while learning a residual around the aggregated actions. This approach can be adopted for both continuous and discrete action spaces and is particularly advantageous when one does not have access to a distribution of source environment instances with diverse dynamics. We confirmed the high training sample efficiency of our approach on a variety of environments. Future work seeks to extend MULTIPOLAR to other challenging problems such as sim-to-real transfer [Tan *et al.*, 2018] and real-world robotics tasks.

Acknowledgments

The authors would like to thank Robert Lee, Felix von Drigalski, Yoshihisa Ijiri, Tatsunori Taniai, and Daniel Plop, for the insightful discussions and helpful feedback on the manuscript.

References

[Bacon *et al.*, 2017] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The Option-Critic Architecture. In *AAAI*, 2017.

[Bello *et al.*, 2016] Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

[Chen *et al.*, 2018] Tao Chen, Adithyavairavan Murali, and Abhinav Gupta. Hardware Conditioned Policies for Multi-Robot Transfer Learning. In *NeurIPS*, 2018.

[Clavera *et al.*, 2019] Ignasi Clavera, Anusha Nagabandi, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to Adapt in Dynamic, Real-World Environments through Meta-Reinforcement Learning. In *ICLR*, 2019.

[Devin *et al.*, 2017] Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning Modular Neural Network Policies for Multi-Task and Multi-Robot Transfer. In *ICRA*, 2017.

[Fernández and Veloso, 2006] Fernando Fernández and Manuela Veloso. Probabilistic Policy Reuse in a Reinforcement Learning Agent. In *AAMAS*, 2006.

[François-Lavet *et al.*, 2018] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, and Joelle Pineau. An Introduction to Deep Reinforcement Learning. *Foundations and Trends in Machine Learning*, 11(3-4):219–354, 2018.

[Frans *et al.*, 2018] Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. Meta Learning Shared Hierarchies. In *ICLR*, 2018.

[Haarnoja *et al.*, 2018] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *ICML*, 2018.

[Henderson *et al.*, 2018] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep Reinforcement Learning That Matters. In *AAAI*, 2018.

[Hill *et al.*, 2018] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Stable Baselines. https://github.com/hill-a/stable-baselines, 2018.

[Johannink *et al.*, 2019] Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual Reinforcement Learning for Robot Control. In *ICRA*, 2019.

[Kulkarni *et al.*, 2016] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation. In *NeurIPS*, 2016.

[Lazaric *et al.*, 2008] Alessandro Lazaric, Marcello Restelli, and Andrea Bonarini. Transfer of Samples in Batch Reinforcement Learning. In *ICML*, 2008.

[Li and Zhang, 2018] Siyuan Li and Chongjie Zhang. An Optimal Online Method of Selecting Source Policies for Reinforcement Learning. In *AAAI*, 2018.

[Li *et al.*, 2019] Siyuan Li, Fangda Gu, Guangxiang Zhu, and Chongjie Zhang. Context-Aware Policy Reuse. In *AAMAS*, 2019.

[Mankowitz *et al.*, 2018] Daniel J Mankowitz, Timothy A Mann, Pierre-Luc Bacon, Doina Precup, and Shie Mannor. Learning Robust Options. In *AAAI*, 2018.

[Osa *et al.*, 2019] Takayuki Osa, Voot Tangkaratt, and Masashi Sugiyama. Hierarchical Reinforcement Learning via Advantage-Weighted Information Maximization. In *ICLR*, 2019.

[Parisotto *et al.*, 2016] Emilio Parisotto, Jimmy Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. In *ICLR*, 2016.

[Rajendran *et al.*, 2017] Janarthanan Rajendran, Aravind S Lakshminarayanan, Mitesh M Khapra, P Prasanna, and Balaraman Ravindran. Attend, adapt and transfer: Attentive deep architecture for adaptive transfer from multiple sources in the same domain. In *ICLR*, 2017.

[Rosman *et al.*, 2016] Benjamin Rosman, Majd Hawasly, and Subramanian Ramamoorthy. Bayesian Policy Reuse. *Machine Learning*, 104(1):99–127, 2016.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[Silver *et al.*, 2018] Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual Policy Learning. *arXiv preprint arXiv:1812.06298*, 2018.

[Song *et al.*, 2016] Jinhua Song, Yang Gao, Hao Wang, and Bo An. Measuring the Distance Between Finite Markov Decision Processes. In *AAMAS*, 2016.

[Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, 1st edition, 1998.

[Sutton *et al.*, 1999] Richard S Sutton, Doina Precup, and Satinder Singh. Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[Tan *et al.*, 2018] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-Real: Learning Agile Locomotion For Quadruped Robots. In *RSS*, 2018.

[Tange, 2018] Ole Tange. *GNU Parallel 2018*. Ole Tange, 2018.

[Tirinzoni *et al.*, 2018] Andrea Tirinzoni, Andrea Sessa, Matteo Pirotta, and Marcello Restelli. Importance Weighted Transfer of Samples in Reinforcement Learning. In *ICML*, 2018.

[Vanschoren, 2018] Joaquin Vanschoren. Meta-Learning: A Survey. *arXiv preprint arXiv:1810.03548*, 2018.

[Wang *et al.*, 2019] Hao Wang, Shaokang Dong, and Ling Shao. Measuring Structural Similarities in Finite MDPs. In *IJCAI*, 2019.

[Yu *et al.*, 2019] Wenhao Yu, C. Karen Liu, and Greg Turk. Policy Transfer with Strategy Optimization. In *ICLR*, 2019.

[Zheng *et al.*, 2018] Yan Zheng, Zhaopeng Meng, Jianye Hao, Zongzhang Zhang, Tianpei Yang, and Changjie Fan. A Deep Bayesian Policy Reuse Approach against Non-Stationary Agents. In *NeurIPS*, 2018.