TempLe: Learning Template of Transitions for Sample Efficient Multi-task RL

Yanchao Sun, [1] **Xiangyu Yin,** [2] **Furong Huang** [1]

[1] University of Maryland, College Park, MD, 20742
[2] Beijing University of Posts and Telecommunications, China
ycs@umd.edu, yinxiangyu@bupt.edu.cn, furongh@umd.edu

Abstract

Transferring knowledge among various environments is important for efficiently learning multiple tasks online. Most existing methods directly use the previously learned models or previously learned optimal policies to learn new tasks. However, these methods may be inefficient when the underlying models or optimal policies are substantially different across tasks. In this paper, we propose Template Learning (TempLe), a PAC-MDP method for multi-task reinforcement learning that could be applied to tasks with varying state/action space without prior knowledge of inter-task mappings. TempLe gains sample efficiency by extracting similarities of the transition dynamics across tasks even when their underlying models or optimal policies have limited commonalities. We present two algorithms for an "online" and a "finite-model" setting respectively. We prove that our proposed TempLe algorithms achieve much lower sample complexity than single-task learners or state-of-the-art multi-task methods. We show via systematically designed experiments that our TempLe method universally outperforms the state-of-the-art multi-task methods (PAC-MDP or not) in various settings and regimes.

1 Introduction

Multi-task reinforcement learning (MTRL) (Wilson et al. 2007; Brunskill and Li 2013; Modi et al. 2018) requires the agent to efficiently tackle a series of tasks. A key goal of MTRL is to improve per-task learning efficiency compared against single-task learners, by using the knowledge obtained from previous tasks to learn new tasks. Despite the recent rapid progress in MTRL, some issues remain unsettled. *(1) Guaranteed sample efficiency.* Only a few existing methods have guarantees on sample efficiency, the most common bottleneck of RL algorithms. *(2) Correctness v.s. efficiency.* An overly aggressive application of previous knowledge may transfer incorrect knowledge and deteriorate the performance on new tasks, resulting in a "negative transfer" (Taylor and Stone 2009). However, if an agent is overly conservative in applying previously learned knowledge, much of the similarities between tasks will be ignored, resulting in an "inefficient transfer". It is nontrivial to balance between the correctness and efficiency or achieve both. *(3) Varying state/action space across tasks.* In practice, transferring knowledge learned from smaller environments to learning in larger environments is extremely useful. However, most existing works on MTRL assume the state/action space is shared across tasks.

In an effort to provide guaranteed sample efficiency for MTRL, Brunskill and Li (2013) propose an algorithm that clusters the underlying Markov Decision Processes (MDPs) of tasks into groups and identifies new tasks as learned groups. However, transferring knowledge from the clustered MDP models could be an "inefficient transfer" if the underlying models are too different to be clustered into a small number of groups. Similarly, most existing model-based approaches (Liu, Guo, and Brunskill 2016; Modi et al. 2018) only exploit model-level similarities, which also makes it difficult to transfer knowledge among different-sized tasks.

We remedy the aforementioned three issues by extraction of more commonalities in tasks without suffering from "negative transfer". A motivating example is the navigation problem in mazes with slippery floors which result in stochastic transitions. For instance, the agent taking an action of going *up* on ice could slip to the *left*, *right* or *down* (instead of *up*) with a certain probability determined by the slipperiness of ice. The slipperiness of the floor depends on the landform of the location, such as sand, marble and ice. We show some examples of different combinations/distributions of the landforms in the maze in Figure 1; the MDP models are drastically different across different mazes, therefore transferring knowledge using similarity of models is inefficient.

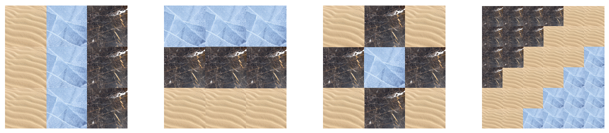


Figure 1: Examples of landform combinations in Maze, where 🟫 stands for sand, ⬛ stands for marble and 🟦 stands for ice. Different landforms have different slippery probability, thus different transition dynamics. Consider a $\sqrt{S} \times \sqrt{S}$ maze with G types of landforms. There could be up to G^S different MDP models, making it prohibitive to extract similarities from the models. However, the types of underlying transition dynamics associated with each state/location are governed by the number of distinct landforms G.

However, our key observation is that the same landforms share the transition dynamics, and knowledge could be transferred from sand to sand, marble to marble, and ice to ice. More importantly, we can extend the knowledge learned from a maze to any-sized mazes consisting of these same types of landforms (e.g., the 4th example in Figure 1). With this idea, we achieve more effective and efficient knowledge transfer by exploiting similarities at the level of *state-action transition dynamics* **instead of** *MDP model dynamics*, allowing knowledge transfer between tasks with **varying state/action space without prior knowledge of inter-task mappings.** The challenge of learning is now reduced to extracting such "landforms" without prior knowledge of the tasks.

We propose a novel method called *Template Learning (TempLe)* for MTRL, which provably guarantees sample efficiency and achieves efficient transfer learning for multi-task reinforcement learning with varying state/action space. We extract templates for similar state-action transition dynamics (landforms in the example above), called *Transition Templates*, and confidently improve the efficiency of transition dynamics estimation in new tasks. By sharing experience among state-action pairs associated with similar templates, the learning process is expedited. We introduce two versions of TempLe: one is for online MTRL without prior knowledge about models, named *Online Template Learning (O-TempLe)*, the other further improves the learning efficiency based on a finite-model assumption, named *Finite-Model Template Learning (FM-TempLe)*.

Summary of Contributions: (1) TempLe achieves a significant *reduction of sample complexity* compared with state-of-the-art PAC-MDP (Probably Approximately Correct in Markov Decision Processes) algorithms. **(2)** TempLe covers two realistic settings, solving MTRL problems in different regimes – *with or without prior knowledge of models*. **(3)** To the best of our knowledge, TempLe is the *first PAC-MDP algorithm* that is able to learn tasks with *varying state/action spaces* without any prior knowledge of inter-task mappings.

2 Related Work

PAC-MDP MTRL Algorithms. Brunskill and Li (2013) present the first formal analysis of the sample complexity for MTRL. They propose a two-phase algorithm and prove that per-task sample complexity is reduced compared with single-task learners. However, they require all tasks coming from a small number of models, and when the number of distinct models is large, their algorithm becomes similar to single-task learning. In this paper, we show our proposed methods outperform the method provided by Brunskill and Li (2013) both in theory and in experiments. There are other PAC-MDP algorithms for multi-task RL, considering the problem from different perspectives. For example, Brunskill and Li (2014) discuss lifelong learning in semi-Markov decision processes (SMDPs), where options are involved. Liu, Guo, and Brunskill (2016) extend the finite-model method (Brunskill and Li 2013) to continuous state space. Feng, Yin, and Yang (2019) and Tirinzoni, Poiani, and Restelli (2020) significantly reduce the sample complexity, but are under the assumption of generative models. Modi et al. (2018) improve the learning efficiency through the assistance of side informations. Abel et al. (2018b) propose MaxQInit, which transfers the maximum Q values across tasks. We empirically compare with MaxQInit in this paper.

Reducing MDPs to Compact Ones. There is a line of research that reduces the original MDPs to compact ones to achieve sample efficiency, including Relocatable Action Model (RAM) (Leffler, Littman, and Edmunds 2007), homomorphism (Ravindran and Barto 2003), and ϵ-equivalent MDP (Even-Dar and Mansour 2003). However, since learning such compact structures is usually difficult (e.g., learning homomorphism is NP-hard as noted by Soni and Singh (2006)), most of the previous works require some prior knowledge. To give a detailed comparison, **our algorithm (1) requires no prior knowledge about the MDP structure.** RAM (Leffler, Littman, and Edmunds 2007) requires knowledge of the "type" of all states (walls, pits, etc) and the next-state function of all states and type-action outcomes. Its continuous extension (Brunskill et al. 2008) also needs knowledge of the types. Homomorphism works (Ravindran and Barto 2004, 2003; Soni and Singh 2006) require knowledge of (candidate) homomorphisms to compress an SMDP or transfer knowledge between MDPs. **(2) works for general RL problems with PAC guarantee.** Although Leffler et al. (2005) (learns latent structure by clustering) and Sorg and Singh (2009) (learns soft homomorphisms) provide methods that do not require knowledge of the structure, Leffler et al. (2005) study a simplified non-MDP problem where actions do not influence state transitions, and Sorg and Singh (2009) do not provide theoretical guarantees when the target model is not known in advance.

Overall, our method is different from the above works, as we do not pre-define the compact structure. Instead, we observe that the transition dynamics, if permuted into descending order, could be naturally grouped to some template. Notably, we *learn* the similarities rather than assuming knowledge of them. Our method could be more practical than the above works (Leffler, Littman, and Edmunds 2007; Leffler et al. 2005; Brunskill et al. 2008; Ravindran and Barto 2004, 2003; Soni and Singh 2006; Sorg and Singh 2009) in multi-task RL, since a new task is often drawn randomly and knowing its structure in advance could be unrealistic.

Comparison with C-UCRL (Asadi et al. 2019). C-UCRL learns a single task by leveraging a state-action equivalence structure that is similar with our proposed templates. They provide an improved regret bound in the case of a known equivalence structure. However, in the more challenging case of an unknown equivalence structure, as is the setting of our paper, no regret bound is provided. In contrast, our work provides a sample complexity guarantee under the unknown equivalence structure scenario. In addition, C-UCRL does not extend trivially to multi-task setting since it find a coarse partition of all state-action pairs at every step, while in MTRL, new state-action pairs come with new tasks, and negative transfer problem may exist when the equivalence structure is unknown.

3 Preliminaries and Notations

Standard RL Notations. An MDP is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, p(\cdot|\cdot,\cdot), r(\cdot,\cdot), \mu, \gamma \rangle$, where $\mathcal{S}$ is the state space (with cardinality S); $\mathcal{A}$ is the action space (with cardinality A); $p(\cdot|\cdot,\cdot)$ is the transition probability function with $p(s'|s,a)$ representing the probability of transiting to state s' from state s by taking action a; $r(\cdot,\cdot)$ is the reward function with $r(s,a)$ recording the reward achieved by taking action a in state s; μ is the initial state distribution; γ is the discount factor. Denote the maximum value of r as R_{max}. Without loss of generality, suppose $0 \le r(s,a) \le 1$ for all (s,a), so $R_{\max} = 1$. Here $p(\cdot|\cdot,\cdot)$ and $r(\cdot,\cdot)$ together are the **model dynamics** of the MDP.

At every step, the agent selects an action based on the current *policy* π. The value function of a policy $V^\pi(s)$, which evaluates the performance of a policy π , is the expected future reward gained by following π starting from s. Similarly, the action value $Q^\pi(s,a)$ is the expected future reward starting from pair (s,a). In an RL task, an agent searches for the optimal policy by interacting with the MDP. We use $V_{\max}$ to denote the upper bound of V. In the discounted setting $V_{\max} = \frac{R_{\max}}{1-\gamma} = \frac{1}{1-\gamma}$.

Sample Complexity. The general goal of RL algorithms is to learn an optimal policy for an MDP with as few interactions as possible. For any $\epsilon > 0$ and any step $h > 0$, if the policy π_h generated by an RL algorithm L satisfies $V^* - V^{\pi_h} \le \epsilon$, we say L is near-optimal at step h. If for any $0 < \delta < 1$, the total number of steps that L is not near-optimal is upper bounded by a function $\zeta(\epsilon, \delta)$ with probability at least $1-\delta$, then ζ is called the *sample complexity* (Kakade et al. 2003) of L.

4 Learning with Templates

As motivated in the example described in Section 1, the main idea of this work is to boost the learning process by aggregating similar state-action transition dynamics (see Definition 1). We permute the elements of transition dynamics/probability vectors to be in descending order, and aggregate these permuted transition probabilities to obtain "templates of transition" defined in Definition 2. We show that the templates are effective abstractions of the environment.

4.1 Transition Template: An Abstraction of Dynamics

In this section, we introduce a more compact way to represent the model dynamics of an MDP. We first formally define the transition dynamics of a state-action (s-a) pair.

Definition 1 (State-Action (s-a) Transition Dynamics)**.** *For any state-action pair (s,a), its* ***transition dynamics*** *is defined as a length-$(S+1)$ vector $\theta(s,a) = [p(s_1|s,a), p(s_2|s,a), \cdots, p(s_S|s,a), r(s,a)]$, where S is the number of states.*

Note that s-a transition dynamics are different from the model dynamics, which characterize the transitions for all s-a pairs. In s-a transition dynamics, the first S elements form the transition probability vector $p(\cdot|s,a)$. As defined in most RL literatures (Kakade et al. 2003; Brunskill and Li 2013), the order of elements in $p(\cdot|s,a)$ is the natural order of the states. In contrast, we re-order the elements of $p(\cdot|s,a)$ by their values, and obtain a more compact representation of the transition dynamics called *Transition Template*.

Definition 2 (Transition Template)**.** *A* ***Transition Template (TT)*** $\mathbf{g}$ *is defined as a tuple $(\mathbf{g}^{(p)}, g^{(r)})$, where $\mathbf{g}^{(p)} \in \mathbb{R}^S$ is a transition probability vector with non-increasingly ordered elements, i.e., $\sum_{i=1}^S g_i^{(p)} = 1$ and $g_i^{(p)} \ge g_j^{(p)} \ge 0, \forall 1 \le i \le j \le S$; $0 \le g^{(r)} \le 1$ is a scalar representing the reward.*

Any s-a transition dynamics can be permuted to an unique TT by re-arranging the transition probability vector $p(\cdot|s,a)$ in a decreasing order and maintaining the reward $r(s,a)$ to $g^{(r)}$, i.e., $\mathbf{g}_{(s,a)} = (\mathrm{desc}(p(\cdot|s,a)), r(s,a))$, where desc orders the elements of $p(\cdot|s,a)$ from the largest value to the smallest value. For example, if $\theta(s_1,a_1) = [0.3, 0.7, 0, 1]$, and $\theta(s_2,a_2) = [0, 0.3, 0.7, 1]$, then (s_1,a_1) and (s_2,a_2) have the same TT $([0.7, 0.3, 0], 1)$, although their s-a transition dynamics are different.

A TT is a representation of multiple s-a transition dynamics with some similarities. It ignores how the s-a pair transits to a specific next state, but only considers the patterns of transition probabilities, allowing more efficient exploitation of similarities. An intuitive example is given in Figure 4 in Appendix A[1]., where there are 100 distinct s-a transition dynamics, but only 2 distinct TTs. Appendix F.5 further discusses the universal existence of such similarities.

4.2 Empirical Estimation of Transition Templates

Section 4.1 defines TT based on the underlying s-a transition dynamics. However, in reality, we do not have access to the underlying dynamics. In model-based RL, a key step is to estimate the dynamics and to build a model of the environment. We now illustrate the estimation of TTs, as well as how TTs augments the learning process.

The conventional estimation of s-a transition dynamics. A direct estimate of $\theta(s,a)$ is obtained through experience, $\hat{\theta}(s,a) = [\frac{n(s,a,s_1)}{n(s,a)}, \frac{n(s,a,s_2)}{n(s,a)}, \cdots, \frac{n(s,a,s_S)}{n(s,a)}, \frac{R(s,a)}{n(s,a)}]$, where $n(s,a,s')$ is the number of observations of transitioning from s to s' by taking action a, $n(s,a)$ is the total number of observations of (s,a), and $R(s,a)$ is the cumulative rewards obtained by (s,a). An accurate estimate of the transition dynamics $\theta(s,a)$ requires a large enough number of observations $n(s,a)$ according to the theory of concentration bounds. Therefore, it is sample-consuming to accurately estimate the transition dynamics of each s-a pair in this way.

Augmented estimation of s-a transition dynamics. As discussed in Section 4.1, different s-a pairs may share the same TTs. Our goal is then to aggregate the estimations of s-a transition dynamics associated with the same TTs. We introduce the following process to obtain estimates of all s-a transition dynamics:

(1) rough estimation: obtain $\widehat{\theta}(s,a) = [\frac{\mathbf{n}(s,a,\cdot); R(s,a)}{n(s,a)}]$ for each (s,a) with a small n;

[1]Appendix can be found on https://arxiv.org/abs/2002.06659

Algorithm 1 Online Template Learning (O-TempLe)

```
    Input: user-specified TT gap $\hat{\tau}$; error tolerance $\epsilon$; discount factor $\gamma$; regular known threshold $m$; small known threshold $m_s$
    Output Near-optimal policies $\{\pi_t\}_{t=1,2,\cdots}$
 1: Initialize an empty TT group set $\mathcal{G}$ and TT visit set $\mathcal{O}$
 2: for $t \leftarrow 1, 2, \cdots$ do
 3:     Receive a task $M_t$
 4:     Initialize visits $\boldsymbol{n}(s, a, \cdot) \leftarrow \mathbf{0}$, accumulative rewards $R(s, a) \leftarrow 0, \forall (s, a) \in (\mathcal{S}, \mathcal{A})$, an empty known state-action set $\mathcal{K}$, and an initial policy $\pi$
 5:     for $h \leftarrow 1, 2, \cdots, H$ do
 6:         Take action $a_h \leftarrow \pi(s_h)$, get $s_{h+1}$ and $r_h$
 7:         Update visits $n(s_h, a_h, s_{h+1})$ and $R(s_h, a_h)$
 8:         if $(s_h, a_h) \notin \mathcal{K}$ and $\|\boldsymbol{n}(s_h, a_h, \cdot)\|_{\ell_1} = m_s$ then
            ▷ TT identification with the small threshold
 9:             $\tilde{\mathbf{g}}, \mathbf{o}_{\tilde{\mathbf{g}}}, \sigma \leftarrow$ GEN-TT$(\boldsymbol{n}(s_h, a_h, \cdot), R(s_h, a_h))$
10:             if no $\mathbf{g} \in \mathcal{G}$ is $\hat{\tau}$-close to $\tilde{\mathbf{g}}$ then
11:                 Add $\tilde{\mathbf{g}}$ to $\mathcal{G}$, $\mathbf{o}_{\tilde{\mathbf{g}}}$ to $\mathcal{O}$
12:             else
13:                 Find the closest TT $\mathbf{g}^*$ to $\tilde{\mathbf{g}}$
14:                 TT-UPDATE$(\mathbf{g}^*, \mathbf{o}_{\mathbf{g}^*}, \boldsymbol{n}(s_h, a_h, \cdot), R(s_h, a_h))$
15:                 AUGMENT$(\mathbf{o}^*_{\mathbf{g}}, \boldsymbol{n}(s_h, a_h, \cdot), R(s_h, a_h), \sigma)$
16:         if $(s_h, a_h) \notin \mathcal{K}$ and $\|\boldsymbol{n}(s_h, a_h, \cdot)\|_{\ell_1} \geq m$ then
            ▷ policy update with the regular threshold
17:             Update $\pi$ using visits $\boldsymbol{n}$ and $R$ by RMax
18:             add $(s_h, a_h)$ to $\mathcal{K}$
19:     for all $(s, a) \in (\mathcal{S}, \mathcal{A})$ with identified TT $\mathbf{g}_{(s,a)}$ do
20:         TT-UPDATE$(\mathbf{g}_{(s,a)}, \mathbf{o}_{\mathbf{g}_{(s,a)}}, \boldsymbol{n}(s, a, \cdot), R(s, a))$
```

(2) permutation: permute each $\widehat{\theta}(s, a)$ to its corresponding permuted estimates $\tilde{\boldsymbol{g}}_{(s,a)}$;
(3) template identification: identify the group of the permuted estimate $\tilde{\boldsymbol{g}}_{(s,a)}$ such that permuted estimates are similar within the group, and obtain a more confident estimate of TT $\widehat{\mathbf{g}}$ aggregating within-group statistics.
(4) augmentation: for every (s, a), obtain a more confident estimate of the transition dynamics by permuting back its corresponding TT with accumulated knowledge.

The noisy estimate of transition dynamics will not render error other than the smaller amount of noise in estimated transition templates if it is identified into the right group. To guarantee accurate identification, the ordering of the elements in the noisy estimate should be consistent with the ground truth. Therefore, the consistency of our estimation depends on TT gap as defined in Definition 5 and "ranking gap" as defined in Definition 8 (see Appendix D for details). An example in Appendix A.1 shows how augmented estimation helps save a large number of samples compared against the conventional estimation.

Now we are ready to formally introduce our algorithms in two settings, Online MTRL and Finite-Model MTRL.

4.3 O-TempLe: Online Template Learning

In the *online MTRL setting*, an agent interacts with multiple tasks streaming-in, each of which corresponding to

Algorithm 2 TT Functions

```
 1: function GEN-TT$(\boldsymbol{n}, R)$                          ▷ generate TT
 2:     find permutation $\sigma$ s.t. $\sigma(\boldsymbol{n})$ is in descending order
 3:     ordered visits $\mathbf{o}_{\mathbf{g}}^{(N)} \leftarrow \sigma(\boldsymbol{n}), o_{\mathbf{g}}^{(R)} \leftarrow R, \mathbf{o}_{\mathbf{g}} \leftarrow (\mathbf{o}_{\mathbf{g}}^{(N)}, o_{\mathbf{g}}^{(R)})$
 4:     transition template $\mathbf{g} \leftarrow (\frac{\mathbf{o}_{\mathbf{g}}^{(N)}}{\|\boldsymbol{n}\|_{\ell_1}}, \frac{o_{\mathbf{g}}^{(R)}}{\|\boldsymbol{n}\|_{\ell_1}})$
 5:     return $\mathbf{g}, \mathbf{o}_{\mathbf{g}}, \sigma$
 6: function TT-UPDATE$(\mathbf{g}, \mathbf{o}_{\mathbf{g}}, \boldsymbol{n}, R)$          ▷ add visits to TT
 7:     $\mathbf{o}_{\mathbf{g}} \leftarrow \mathbf{o}_{\mathbf{g}} + (descending(\boldsymbol{n}), R)$
 8:     $\mathbf{g} \leftarrow (\frac{\mathbf{o}_{\mathbf{g}}^{(N)}}{\|\mathbf{o}_{\mathbf{g}}^{(N)}\|_{\ell_1}}, \frac{o_{\mathbf{g}}^{(R)}}{\|\mathbf{o}_{\mathbf{g}}^{(N)}\|_{\ell_1}})$
 9: function AUGMENT$(\mathbf{o}_{\mathbf{g}}, \boldsymbol{n}, R, \sigma)$  ▷ augment visits by TT
10:     $\boldsymbol{n} \leftarrow \boldsymbol{n} + \sigma^{-1}(\mathbf{o}_{\mathbf{g}}^{(N)})$
11:     $R \leftarrow R + o_{\mathbf{g}}^{(R)}$
```

a specific MDP. The tasks are i.i.d. drawn from a set $\mathcal{M}$ of MDPs (models). MDPs in $\mathcal{M}$ may have different state/action spaces. The number of MDPs $|\mathcal{M}|$ can be arbitrarily large.

We introduce Online Template Learning (O-TempLe) for the online MTRL setting. O-TempLe is a meta-learning algorithm with model-based "*base learners*" which compute policies for the current task. We use RMax (Brafman and Tennenholtz 2003) as the base learner, and it can be replaced by other model-based methods such as E^3 (Kearns and Singh 2002) and MBIE (Strehl and Littman 2005). The principle of RMax algorithm on an MDP M is to build an induced MDP based on a known threshold m. A state-action pair is said to be m-known if the number of visits/observations $n(s, a) \geq m$. A state is m-known if $n(s, a) \geq m, \forall a \in \mathcal{A}$. The set of all m-known states induces an MDP M_k, where for any m-known state s, $p(s'|s, a) = \frac{n(s,a,s')}{n(s,a)}$, $r(s, a) = \frac{R(s,a)}{n(s,a)}$ and for any non-m-known state s, $p(s'|s, a) = \mathbb{I}\{s' = s\}$, $r(s, a) = R_{\max}$. Then, RMax computes an optimal policy based on the optimistic model by dynamic programming.

In contrast, O-TempLe uses augmented estimation introduced in Section 4.2. to reduce the required number of visits to every single s-a pair. Instead of aggregating the estimates of all s-a transition dynamics at once, O-TempLe asynchronously identifies the TTs of s-a pairs and updates the template groups in an online manner, through measuring the distances among TTs.

Algorithm 1 illustrates how O-TempLe works. In addition to the regular known threshold m used in RMax, we design a smaller known threshold m_s, which is the smallest number of visits to ensure identifying the TTs of all s-a pairs. If for any (s, a), the total number of visits ($\|\boldsymbol{n}(s, a, \cdot)\|_{\ell_1}$) reaches m_s, then the estimated TT $\tilde{\mathbf{g}}$ of (s, a) will be generated by function GEN-TT. If $\tilde{\mathbf{g}}$ has at least $\hat{\tau}$-distance with all existing TTs, we regard it as a new TT and append it to set $\mathcal{G}$ (Line 10-11); otherwise (Line 12-15), we find the closest TT to $\tilde{\mathbf{g}}$, then synchronize the experience of (s, a) in the current task and the accumulated experience that its TT holds by calling functions TT-UPDATE and AUGMENT, which respectively send the current visits of (s, a) to the corresponding TT, and feed the accumulative visits of the TT to the current (s, a). GEN-TT, TT-UPDATE and AUGMENT involve the permutation operations, and are given by Algorithm 2. Accumulated experience of each TT is stored in a tuple $\mathbf{o_g} = (\mathbf{o_g}^{(N)}, o_\mathbf{g}^{(R)})$, where $\mathbf{o_g}^{(N)}$ is the total visits accumulated by permuted $\mathbf{n}(s, a, s')$ of all (s, a)'s with TT $\mathbf{g}$. When (s, a) is m-known, the policy is updated (Line 16-18). Overall, our O-TempLe allows grouped s-a transition dynamics to share their visit counts, making it much easier for them to reach m visits than in regular RMax.

Note that Algorithm 1 also works for tasks with varying state/action space, since the comparison of TTs considers the non-zero elements of the transition vectors only. One can compute the difference between two different-sized TTs by simply padding zeros to the end of the shorter TT.

4.4 FM-TempLe: Finite-Model Template Learning

Online MTRL setting requires no prior knowledge of the types of underlying MDPs and improves the sample efficiency by accumulating knowledge with TT groups. However, under a more restrictive assumption that the number of possible MDPs $C = |\mathcal{M}|$ is known and small, it is possible to get rid of the dependence on the size of state-action space and achieve *more efficient learning*.

We propose Finite-Model Template Learning (FM-TempLe), an extension of our O-TempLe, under the *finite-model MTRL setting*, where the agent still interacts with streaming-in tasks drawn from a set $\mathcal{M}$ of MDPs, but the number of MDPs in the set $\mathcal{M}$ is small and known.

In contrast with O-TempLe, FM-TempLe is able to correctly identify the TTs of some s-a pairs before they are visited for m_s times. This is because the number of underlying models is small, and thus identifying the model is easy and inexpensive. It is possible to obtain the TTs for all s-a pairs immediately after identifying the model, since the way how TTs are distributed over all s-a pairs is fixed for each MDP model.

The main steps of FM-TempLe are stated below, and the details are illustrated in Algorithm 3 in Appendix C. *(1) Collecting Models:* for the first T_1 tasks, the agent acts in the same way as O-TempLe, but also stores the TT structure of each model. *(2) Grouping Models:* the first T_1 tasks are clustered into finite groups of models based on their TT structures. *(3) Identifying Models:* for any new task, the agent still follows O-TempLe, but also seeks the true model for the current task from all the model groups, by ruling out the groups of models that have different TT structures.

Brunskill and Li (2013) make the same finite-model assumption and propose an algorithm FMRL which extracts model similarities. However, FMRL can not transfer knowledge between two models which are the same except for one state-action pair. In contrast, our FM-TempLe extracts state-action dynamics similarities and thus transferring happens among any state-action pairs that have similar dynamics. Compared with FMRL, FM-TempLe not only has lower sample complexity as proved in Section 5, but also saves computations due to the direct comparison of TTs.

5 Theoretical Analysis

This section provides sample complexity analysis of the proposed two algorithms O-TempLe and FM-TempLe. Although O-TempLe and FM-TempLe can be applied to tasks with varying state/action spaces, we assume all tasks have the same $\mathcal{S}$ and $\mathcal{A}$ for simplicity of notations, and the analysis extends to varying state/action spaces trivially.

We first assume there is a diameter D such that any state s' is reachable from any states s in at most D steps on average. This assumption is commonly used in RL (Jaksch, Ortner, and Auer 2010), and it ensures the reachability of all state from any state on average.

We further define the underlying minimal ℓ_2-distance among TTs as τ, namely TT gap. We also define ν as the ranking gap; a large ranking gap implies that for any s-a pair, the probabilities of transitioning to any two states are substantially different. For any $\mathbf{g} \in \mathcal{G}$, if $\mathbf{g}_i^{(p)} > \mathbf{g}_j^{(p)}$ are two adjacent elements in $\mathbf{g}^{(p)}$, then either $\mathbf{g}_i^{(p)} - \mathbf{g}_j^{(p)} \geq \nu$, or $\mathbf{g}_i^{(p)} - \mathbf{g}_j^{(p)} \leq \tilde{\mathcal{O}}(\frac{\epsilon(1-\gamma)}{\sqrt{S}V_{\max}})$(logarithmic terms are hided in $\tilde{\mathcal{O}}(\cdot)$). The ranking gap implies that for any s-a pair, the probabilities of transitioning to any two states are either very close, or substantially different. Note that the algorithms take a user-specified τ, but do not require input of ν. See Appendix D for formal definitions of TT gap and ranking gap. For notation simplicity, let ω denote $\max\{\min(\tau, \nu), \mathcal{O}(\frac{\epsilon(1-\gamma)}{\sqrt{S}V_{\max}})\}$.

Theorem 3 (**Sample Complexity of O-TempLe**). *For any given $\epsilon > 0$, $1 > \delta > 0$, running Algorithm 1 on T tasks, each for at least $\mathcal{O}(\frac{DSA}{\omega^2} \ln \frac{1}{\delta})$ steps, generates at most $\tilde{\mathcal{O}}\left(\frac{SGV_{\max}^3}{\epsilon^3(1-\gamma)^3} + \frac{TSAV_{\max}}{\omega^2\epsilon(1-\gamma)}\right)$ non-ϵ-optimal steps, with probability at least $1 - \delta$, where G is the total number of TTs.*

Remark. (1) Our provided bound achieves *state-of-the-art* dependence on the environment size T, S, A for general MTRL, given that G is independent of T, S, A. (2) When ϵ is small, the sample complexity only has a linear dependence on the number of states S and the number of templates G, because the first term dominates. By definition, G is always no larger than TSA, the number of all s-a pairs. And in most environments, we have $G \ll TSA$, as discussed in Appendix F.5. (3) When ϵ is not small or T is very large, the sample complexity has linear dependences on T, S and A since the second term dominates.

O-TempLe *does not necessarily require the number of templates G to be small*. A large G suggests the environment is highly stochastic, e.g., the slipping probabilities of every grid in maze is sampled from a Gaussian distribution. In this case, we can still cluster s-a pairs with adequately close templates, as verified in experiments (see Section 6.3).

Proof Sketch. We first show that for any s-a pair, $m_s = \tilde{\mathcal{O}}(\frac{1}{\omega^2})$ samples would guarantee correct template identification and aggregation, and $m = \tilde{\mathcal{O}}(\frac{SV_{\max}^2}{\epsilon^2(1-\gamma)^2})$ samples are sufficient for estimating the s-a transition dynamics. Then we prove that all s-a pairs reach m_s within finite steps. Finally, by computing the number of visits to unknown s-a pairs and applying the PAC-MDP theorem proposed by Strehl, Li, and Littman (2012), we get the sample complexity result. Proof details are in Appendix E.

Comparison with a single-task learner. If RMax is sequentially run for every task, the total sample complexity for T tasks is $\tilde{\mathcal{O}}\left(\frac{TS^2AV_{\max}^3}{\epsilon^3(1-\gamma)^3}\right)$.

(1) When precision is high, i.e., ϵ is small, a significant improvement is achieved, if $\mathcal{O}(SG) \ll \mathcal{O}(TS^2A)$.

(2) When T is large, as long as $\tilde{\mathcal{O}}(\frac{SV_{\max}^2}{\epsilon^2(1-\gamma)^2}) \gg \tilde{\mathcal{O}}(\frac{1}{\omega^2})$, our O-TempLe gains improved sample efficiency.

(3) O-TempLe will not cause negative transfer among tasks. In the worst case, $G = TSA$ (there is no similarity among all s-a transition dynamics) or $\omega^2 = \tilde{\mathcal{O}}(\frac{SV_{\max}^2}{\epsilon^2(1-\gamma)^2})$, O-TempLe has the same-order sample complexity with RMax.

Theorem 4 (**Sample Complexity of FM-TempLe**). *Under the finite-model assumption of there are at most C MDPs for all tasks, for any given $\epsilon > 0, 1 > \delta > 0$, Algorithm 3 on T tasks follows ϵ-optimal policies for all but*

$$\tilde{\mathcal{O}}\Big(\frac{SGV_{\max}^3}{\epsilon^3(1-\gamma)^3} + \frac{T_1SAV_{\max}}{\omega^2\epsilon(1-\gamma)} + \frac{(T-T_1)DC^2V_{\max}}{\omega^2\epsilon(1-\gamma)}\Big) \quad (1)$$

steps with probability at least $1-\delta$, where G is the total number of TTs, $T_1 = \Omega(\frac{1}{p_{\min}}\ln\frac{C}{\delta})$ is the number of tasks in the first phase, where $p_{\min}$ is the minimal probability for a task to be drawn from $\mathcal{M}$.

Remark. (1) When C is very large, or $p_{\min}$ is very small, $T_1 \to T$ and FM-TempLe degenerates to O-TempLe. (2) If $DC^2 < SA$ and $T \gg T_1$, FM-TempLe requires fewer samples than O-TempLe.

Comparison with FMRL (Brunskill and Li 2013) FM-TempLe has a large improvement over FMRL in most cases. The sample complexity of FMRL for T tasks in our notation is

$$\tilde{\mathcal{O}}\Big(\frac{CS^2AV_{\max}^3}{\epsilon^3(1-\gamma)^3} + \frac{T_1S^2AV_{\max}^3}{\epsilon^3(1-\gamma)^3} + (T-T_1)\big(\frac{DC^2V_{\max}}{\Gamma^2\epsilon(1-\gamma)} + \frac{SCV_{\max}^3}{\epsilon^3(1-\gamma)^3}\big)\Big). \quad (2)$$

where $T_1 = \Omega(\frac{1}{p_{\min}}\ln\frac{C}{\delta})$, and Γ is the model difference gap defined by Brunskill and Li (2013). We organize Equation 1 and Equation 2 both as three-term forms. The first term is for learning of all TTs or all models, where FM-TempLe reduces the dependence on S and gets rid of the dependence on A. The second term is for the first phase, where FMRL performs the same with a single-task RMax learner, while FM-TempLe requires much fewer samples to get optimal policies. Finally, the last term is for the second phase. FMRL needs an additional model elimination step for each task, while FM-TempLe does not. FM-TempLe is worse than FMRL only in extreme cases where there are few MDP models with large model gaps, and a large number of TTs with small TT gaps or ranking gaps.

6 Experiments

In this section, we demonstrate empirical results to show O-TempLe and FM-TempLe outperform existing state-of-the-art algorithms both in the finite-model setting and in the more realistic online setting. TempLe is able to transfer knowledge between tasks with different sized environments. More importantly, TempLe has a high tolerance to model perturbations; it implements efficient transfer even when the underlying number of TTs is infinite. Our code is available at https://github.com/umd-huang-lab/template-reinforcement-learning.

Baselines. We choose the state-of-the-art MTRL algorithms, *Abstraction RL (Abs-RL)* (Abel et al. 2018a), MaxQInit (Abel et al. 2018b) and *FMRL* (Brunskill and Li 2013) as baselines. For Abs-RL and MaxQInit, we use the code provided by authors. Note that Abs-RL and MaxQInit have multiple versions due to the selection of different base learners, we show the ones with their best performance in this section, and other versions in Appendix F.3. Abs-RL works for both the online and finite-model setting, whereas MaxQInit and FMRL work for the finite-model setting only, since they both require the number of tasks to be small and known. Meanwhile, to show the effectiveness of our proposed algorithms and other MTRL algorithms, we also run RMax and Q-learning (Watkins and Dayan 1992) for every single task without knowledge transfer.

6.1 Finite-Model MTRL

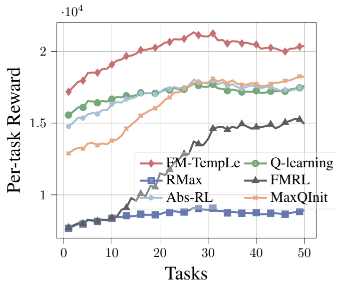
Environment. All the baselines including FMRL, Abs-RL, MaxQInit are designed for the finite-model setting (note that Abs-RL also works in the online setting), where the number of models C is small. We use a similar maze environment as in FMRL, where MDPs only differ at the goal state.

Performance. We generate two 4×4 maze tasks with different goal states as the underlying models, and then randomly sample 50 tasks from the two underlying models. Figure 2a shows the comparison of per-task rewards. FMRL has the same performance with RMax in the model-collecting phase, and then achieves increasing rewards in the following tasks after it successfully identifies the underlying two types of MDPs. After 30 tasks, all state-actions pairs in the models become known, so the per-task reward converges. Similarly, MaxQInit gains more rewards when it collects adequate knowledge of the Q values. In contrast, FM-TempLe has a better start as it learns TTs from the beginning. And model identification further helps with efficient learning. Over all tasks, FM-TempLe substantially outperforms other agents, despite that baselines are designed for the finite-model case.

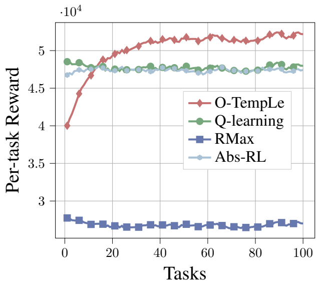
6.2 Online MTRL

Environment. For the more realistic Online MTRL which allows the *number of MDP models to be extremely large*, we generalize the traditional maze environment to have arbitrary combinations of landforms, as shown in Figure 1. We use 3 types of landforms, sand, marble and ice, respectively with slipping probabilities 0, 0.2, and 0.4. In this scenario, under a certain number of states S, the number of possible tasks is exponential in S.

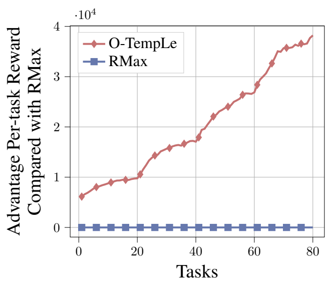
Performance. In the online setting, we consider 4×4 mazes with different arrangements of landforms streaming in. The per-task rewards of each agent are displayed in Figure 2b. Among all agents, our O-TempLe obtains the highest average reward. We see during the first 40 tasks, the performance



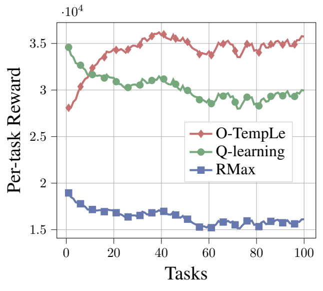
(a) Finite-Model MTRL



(b) Online MTRL



(c) Varying-sized MTRL



(d) Infinite number of TTs

Figure 2: Performance of O-TempLe and FM-TempLe compared against state-of-the-art baselines in **(a)** Online MTRL (to show TempLe's ability to efficiently transfer knowledge), **(b)** Finite-Model MTRL (to show TempLe outperforms baselines even under environments that the baselines are designed for),**(c)** varying sized MTRL (to show TempLe extends to varying sized state space) and **(d)** Online MTRL with Mixture-of-Gaussians distributed landforms (to show TempLe's robustness against noise and model-perturbation). All results are averaged over 20 different random sequences of tasks. Confidence intervals are omitted to reduce overlapping.

of O-TempLe continuously and rapidly grows by transferring previous knowledge. In contrast, the performance of Abs-RL does not increase as more tasks come in and keeps the same with single-task Q-learning, because the maze environment is not efficiently abstracted by Abs-RL.

Performance on Varying State Space. To show the feasibility of TempLe for varying-sized environment tasks, and its ability to generalize knowledge learned in small tasks to speed up learning in larger tasks, we vary the size of the mazes across tasks. More specifically, the first 20 tasks are 3×3 mazes, followed by 20 4×4 mazes, 20 5×5 mazes and 20 6×6 mazes. We show O-TempLe's per-task advantage rewards over single task RMax in Figure 2c, since other MTRL baselines are not feasible in this setting. The performance advantage over RMax increases over more observed tasks, verifying that O-TempLe transfers knowledge among different-sized mazes. Experiments on varying action spaces are shown in Appendix F.4.

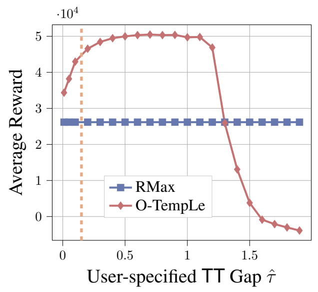
6.3 MTRL with Infinite TTs

Environment. We also conduct experiments to show TempLe's robustness to noise and model perturbations. which is crucial for its application to real-world settings where "landforms" could vary continuously. We draw the landforms (slipping probabilities) of each grid from a mixture of Gaussian distributions, which are centered at 0.2, 0.4, and 0.6 with standard derivation 0.05. In this case, the number of TTs could be infinitely large.

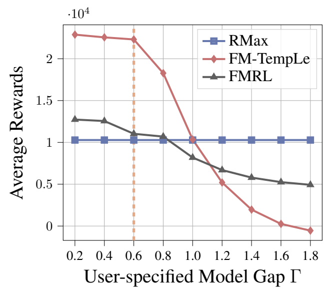
Performance. We show O-TempLe's per-task advantage rewards over single task RMax and Q-learning in Figure 2d, in which O-TempLe still achieves successful multi-task learning. This result demonstrates O-TempLe's ability of tolerating noise and generalizing to real-life applications.

6.4 Robustness to Hyper-parameters

TempLe requires a user-specified TT gap $\hat{\tau}$ as input. Also, both FMRL and FM-TempLe require a user-specified model gap Γ. We test various hyper-parameters to understand how significantly the performance of the algorithms could be affected by inaccurate guesses of $\hat{\tau}$ and Γ, shown in Figure 3.



(a) TT gap $\hat{\tau}$



(b) Model gap Γ

Figure 3: Hyper-parameter test of TT gap $\hat{\tau}$ and model gap Γ(the vertical dashed line shows the underlying true value).

According to Figure 3a, the performance of O-TempLe drops when $\hat{\tau}$ is too large. However, the rewards remain high for relatively small $\hat{\tau}$. Figure 3b shows that FM-TempLe gets higher rewards than RMax when setting $\Gamma \leq 1$, although Γ has a larger influence on FM-TempLe compared to FMRL, potentially because the failure of model clustering will cause more inaccurate TT identification. Note that by definition, both $\hat{\tau}$ and Γ would not exceed 2 (see Lemma 6). So we still have a large chance to get higher rewards than RMax by making an educated guess. The results in Figure 3 guide the users to specify hyper-parameters when using TempLe.

The results with confidence intervals, comparison of cumulative rewards, and experiments on additional environments are shown in Appendix F. We also provide an extension of our work to deep RL is discussed in Appendix F.6.

7 Conclusion and Discussion

In this work, we propose TempLe, the first PAC-MDP MTRL algorithm that works for tasks with varying state/action space without any inter-task mappings or prior knowledge of the MDP structures. This work can be extended in many directions. For example, one may benefit from investigating transition probability and reward separately. The idea of extracting modular similarities can also be extended to continuous MDP and deep model-based RL.

Acknowledgements

Huang is supported by startup fund from Department of Computer Science of University of Maryland, National Science Foundation IIS-1850220 CRII Award 030742-00001, DOD-DARPA-Defense Advanced Research Projects Agency Guaranteeing AI Robustness against Deception (GARD), Laboratory for Physical Sciences at University of Maryland, and Adobe, Capital One and JP Morgan faculty fellowships.

Ethical Impact

Our presented algorithms on multi-task reinforcement learning facilitate the learning of new tasks using knowledge accumulated from previously learned tasks. In scenarios where an RL agent needs to sequentially interact with a series of environments, e.g., navigation in various places, our proposed algorithm could be applied to improve the learning efficiency without loss of accuracy. More importantly, our algorithms are guaranteed to learn near-optimal policies and avoid negative transfer, which are crucial for high-stakes applications, such as autonomous driving, market making, and health-care systems.

Nowadays, Deep Reinforcement Learning (DRL) has achieved great success in many applications. However, problems like high variance and instability restrict the use of DRL in real-life problems. Thus, it is important to study tabular RL with guarantees, which could potentially benefit DRL and applications involving DRL. Our proposed algorithms, although not in the scope of DRL, could be potentially extended to DRL in the following ways. **(1)** Our idea of extracting "relative" transition probability similarity could be directly used in model-based DRL. For example, the next-state prediction model usually outputs a Gaussian distribution for every s-a pair, and one can augment the learned derivation by averaging over predicts with close derivations, assuming some similarity about the uncertainty among different states. **(2)** It is possible to discretize state space and apply count-based methods, as suggested in by Tang et al. (2016).

Our work on multi-task reinforcement learning also has the potential to be applied to other transfer learning tasks within and outside of the Reinforcement Learning community. Any learning in systems that share modular similarities could potentially benefit from our algorithms to speed up the training process.

References

Abel, D.; Arumugam, D.; Lehnert, L.; and Littman, M. 2018a. State abstractions for lifelong reinforcement learning. In *International Conference on Machine Learning*, 10–19.

Abel, D.; Jinnai, Y.; Guo, S. Y.; Konidaris, G.; and Littman, M. 2018b. Policy and value transfer in lifelong reinforcement learning. In *International Conference on Machine Learning*, 20–29.

Asadi, M.; Talebi, M. S.; Bourel, H.; and Maillard, O.-A. 2019. Model-Based Reinforcement Learning Exploiting State-Action Equivalence. *arXiv preprint arXiv:1910.04077* .

Brafman, R. I.; and Tennenholtz, M. 2003. R-max - a General Polynomial Time Algorithm for Near-optimal Reinforcement Learning. *J. Mach. Learn. Res.* 3: 213–231. ISSN 1532-4435.

Brunskill, E.; Leffler, B. R.; Li, L.; Littman, M. L.; and Roy, N. 2008. CORL: A Continuous-State Offset-Dynamics Reinforcement Learner. UAI'08, 53–61. Arlington, Virginia, USA: AUAI Press. ISBN 0974903949.

Brunskill, E.; and Li, L. 2013. Sample Complexity of Multi-Task Reinforcement Learning. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13, 122–131. Arlington, Virginia, USA: AUAI Press.

Brunskill, E.; and Li, L. 2014. PAC-Inspired Option Discovery in Lifelong Reinforcement Learning. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, II–316–II–324. JMLR.org.

Even-Dar, E.; and Mansour, Y. 2003. Approximate equivalence of Markov decision processes. In *Learning Theory and Kernel Machines*, 581–594. Springer.

Feng, F.; Yin, W.; and Yang, L. F. 2019. How Does an Approximate Model Help in Reinforcement Learning? *arXiv e-prints* arXiv:1912.02986.

Jaksch, T.; Ortner, R.; and Auer, P. 2010. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research* 11(Apr): 1563–1600.

Kakade, S. M.; et al. 2003. *On the sample complexity of reinforcement learning*. Ph.D. thesis, University of London London, England.

Kearns, M.; and Singh, S. 2002. Near-optimal reinforcement learning in polynomial time. *Machine learning* 49(2-3): 209–232.

Leffler, B. R.; Littman, M. L.; and Edmunds, T. 2007. Efficient reinforcement learning with relocatable action models. In *AAAI*, volume 7, 572–577.

Leffler, B. R.; Littman, M. L.; Strehl, A. L.; and Walsh, T. J. 2005. Efficient Exploration With Latent Structure. In *Robotics: Science and Systems*, 81–88.

Liu, Y.; Guo, Z.; and Brunskill, E. 2016. PAC continuous state online multitask reinforcement learning with identification. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, 438–446.

Modi, A.; Jiang, N.; Singh, S.; and Tewari, A. 2018. Markov decision processes with continuous side information. In *Algorithmic Learning Theory*, 597–618.

Ravindran, B.; and Barto, A. G. 2003. SMDP Homomorphisms: An Algebraic Approach to Abstraction in Semi-Markov Decision Processes. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, 1011–1016. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Ravindran, B.; and Barto, A. G. 2004. *An algebraic approach to abstraction in reinforcement learning*. Ph.D. thesis, University of Massachusetts at Amherst.

Soni, V.; and Singh, S. 2006. Using homomorphisms to transfer options across continuous reinforcement learning domains. In *AAAI*, volume 6, 494–499.

Sorg, J.; and Singh, S. 2009. Transfer via soft homomorphisms. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, 741–748.

Strehl, A. L.; Li, L.; and Littman, M. L. 2012. Incremental model-based learners with formal learning-time guarantees. *arXiv preprint arXiv:1206.6870* .

Strehl, A. L.; and Littman, M. L. 2005. A Theoretical Analysis of Model-Based Interval Estimation. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, 856–863. New York, NY, USA: ACM. ISBN 1-59593-180-5. doi:10.1145/1102351.1102459.

Tang, H.; Houthooft, R.; Foote, D.; Stooke, A.; Chen, X.; Duan, Y.; Schulman, J.; De Turck, F.; and Abbeel, P. 2016. #Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning. *arXiv e-prints* arXiv:1611.04717.

Taylor, M. E.; and Stone, P. 2009. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research* 10(Jul): 1633–1685.

Tirinzoni, A.; Poiani, R.; and Restelli, M. 2020. Sequential Transfer in Reinforcement Learning with a Generative Model. In III, H. D.; and Singh, A., eds., *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, 9481–9492. PMLR.

Watkins, C. J.; and Dayan, P. 1992. Q-learning. *Machine learning* 8(3-4): 279–292.

Wilson, A.; Fern, A.; Ray, S.; and Tadepalli, P. 2007. Multitask reinforcement learning: a hierarchical Bayesian approach. In *Proceedings of the 24th international conference on Machine learning*, 1015–1022. ACM.