2022 American Control Conference (ACC)
Atlanta, USA, June 8-10, 2022

Sample efficient transfer in reinforcement learning for high variable cost environments with an inaccurate source reward model

Md Ferdous Alam[1], Max Shtein[2], Kira Barton[3], David J. Hoelzle[1]

***Abstract*— Here we propose an algorithm that combines two classic ideas, transfer learning and temporal abstraction, to accelerate learning in high variable cost environments (HVC-envs). In an HVC-env, each sampling of the environment incurs a high cost, thus methods to accelerate learning are sought to reduce the incurred cost. Transfer learning can be useful for such environments by using prior knowledge from a source environment. As only a small number of samples can be collected from an HVC-env due to high sampling cost, learning becomes challenging when the source environment provides inaccurate rewards. To overcome this challenge we propose a simple but effective way of creating useful temporally extended actions from an inaccurate physics guided model (PGM) that acts as the source task. At first we address this issue theoretically by providing performance bounds between two semi-Markov Decision Processes (SMDPs) with different reward functions. Later we develop two benchmark HVC-envs where learning must happen using a small number of real samples (often on the order of $\sim 10^2$ or 10^3). Finally we show that it is possible to obtain sequential high rewards in both of these environments using $\sim 10^3$ real samples by leveraging knowledge from PGMs with inaccurate reward models.**

I. Introduction

Online learning through sequential decision making is usually achieved by reinforcement learning (RL). Typically RL is formalized as a Markov Decision Process (MDP) [1], where an agent learns to find an optimal policy in an environment by observing the feedback, often known as the 'reward'. This trial and error based learning approach has shown exceptional capabilities in mastering complex games like 'go' [2], surpassing human level performance in Atari games [3] and robotic manipulation [4], to name a few. Although RL is promising for numerous real-world applications, most RL algorithms usually require a large number of samples, on the order of thousands or hundreds of thousands, from the environment to learn a task [5]. This approach is not feasible in many real-world applications where each sample incurs a high variable cost, thus making extensive sampling not economical. For example, an additive manufacturing system where manufacturing a sample incurs a variable cost in the form of machine time, materials consumption, and labor [6]. In these environments the learning must happen with a small number of samples, often on the order of 10^2 or 10^3. We coin such environments as "High Variable Cost Environments (HVC-env)". Recently, a learning algorithm has been proposed for autonomous manufacturing systems [7] that builds upon two popular ideas, transfer learning [8], [9] and temporal abstraction [10]. The basic idea of transfer learning is to learn in a source task and then transfer the knowledge in a similar target task so that the number of samples needed to learn in the target task can be reduced. Another way of accelerating the learning is the use of temporal abstraction that extends an MDP to a semi-Markov Decision Process (SMDP) and makes it possible to implement temporally extended actions, often known as 'options'. The intuition behind combining these ideas in [7] is to extract temporally extended actions or options from learning in a first-principles based cheap source task and then use those options to interact with the real system which is the target task. Transfer learning directs our actions to avoid low-reward region of the state space. Temporal abstraction permits us to jump large swaths of the low reward state space to high reward regions. Here, we extend their work so that it can be applicable to a wide variety of HVC-envs. We provide necessary theoretical performance bounds for creating temporal abstractions. Later we provide simple yet effective ways to create temporal abstractions that can

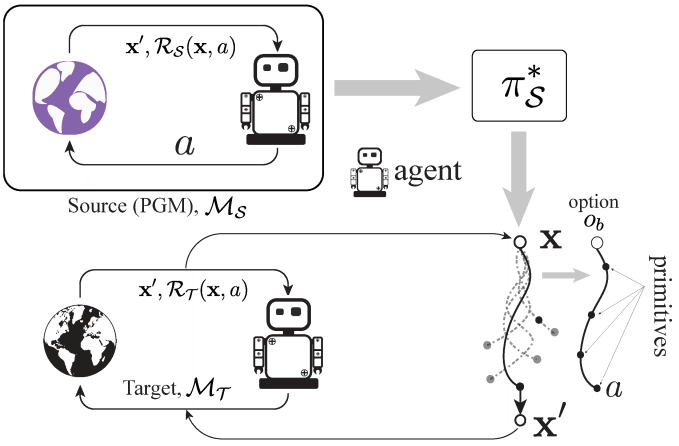


Fig. 1: TAPRL algorithm demonstrating how a PGM source environment can be used to create useful temporal abstractions: source optimal policy $\pi^*_{\mathcal{S}}$, extracted from learning in a PGM $\mathcal{M}_{\mathcal{S}}$, is used to create temporal abstraction o_b with primitives. The last primitive is the only primitive action that interacts with the real system, obtains reward R_t, and then updates a modified policy π^*_{TAPRL}.

This work was supported in part by NSF Award CMMI-1727894

[1]Department of Mechanical and Aerospace Engineering, The Ohio State University, Columbus, OH, USA, 43210. {alam.92,hoelzle.1}@osu.edu

[2]Department of Materials Science and Engineering, University of Michigan, Ann Arbor, Michigan, USA, 48109. {mshtein}@umich.edu

[3]Department of Mechanical Engineering, University of Michigan, Ann Arbor, Michigan, USA, 48109.{bartonkl}@umich.edu

leverage the source knowledge even when the source reward function is a biased estimate of the target reward function. Fig. 1 shows the basic idea of this paper where an RL agent has access to a Physics Guided Model (PGM) that acts as the source task and provides an estimated reward function $\mathcal{R}_{\mathcal{S}}$. The actual reward function $\mathcal{R}_{\mathcal{T}}$ in the target task is unknown *apriori*. The agent learns to behave optimally in the PGM through sequential interactions and finds the source optimal policy $\pi^*_{\mathcal{S}}$. Next multiple temporal abstractions are created from $\pi^*_{\mathcal{S}}$ where each of this abstraction consists of multiple single timestep primitive actions suggested by the PGM. We call this approach 'Temporal abstraction in Physics guided RL (TAPRL)' and denote π^*_{TAPRL} as the optimal policy obtained from this method that can overcome the bias induced by $\pi^*_{\mathcal{S}}$.

The contributions of this paper are the following: (1) Theoretical performance bounds on SMDPs when the reward function is different between two environments, (2) Development of two benchmark problems with HVC-envs, (3) Demonstration of the performance of TAPRL in the benchmark problems. This paper is organized as follows: Section II provides important background on MDPs, SMDPs and temporal abstraction, Section III provides the definitions and theoretical bounds along with the proposed algorithm, Section IV provides the benchmark examples, and Section V discusses important results.

II. Background

A. MDP and value function based RL

The standard RL framework is formalized as a Markov Decision Process (MDP) [11] while temporal abstraction is formalized by semi-Markov Decision Process (SMDP) in the context of MDPs [10]. A finite discrete time Markov decision process is usually defined as a tuple $\mathcal{M} = \langle \mathcal{X}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where $\mathcal{X}$ is the state-space, $\mathcal{A}$ is the action-space, $\mathcal{P}$ is the transition probability ($p(\mathbf{x}_{t+1}|\mathbf{x}_t, a_t)$) that describes the dynamics of the system, $\mathcal{R} : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ defines a reward function and $\gamma \in [0, 1)$ is a scalar discount factor. At each timestep, the learning agent at state $\mathbf{x}_t$ interacts with the environment using an action a_t, obtains a reward $R_t = \mathcal{R}(\mathbf{x}_t, a_t)$ as the feedback and transitions to the next state $\mathbf{x}_{t+1}$ according to the conditional probability $p(\mathbf{x}_{t+1}|\mathbf{x}_t, a_t)$. We use the following notations to describe the transition dynamics in Eq. 1 and expected reward in Eq. 2 for this single timestep learning mechanism in an MDP.

$$p^a_{\mathbf{x}\mathbf{x}'} = p(\mathbf{x}_{t+1}|\mathbf{x}_t, a_t) \tag{1}$$

$$r^a_{\mathbf{x}} = \mathbb{E}[R_t|\mathbf{x}_t, a_t] \tag{2}$$

Formally, the goal of an agent is to learn the policy $\pi : \mathcal{X} \to \mathcal{A}$ that maximizes the expected discounted future reward from state $\mathbf{x}_t$ in the case of an state-value function $V^\pi(\mathbf{x}_t)$ or from state-action tuple $(\mathbf{x}_t, a_t)$ in the case of an action-value function $Q(\mathbf{x}_t, a_t)$. The action-value function while following a policy π is defined as $Q^\pi(\mathbf{x}, a) = \mathbb{E}[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots | \mathbf{x}, a] = r^a_{\mathbf{x}} + \gamma \sum_{\mathbf{x}' \in \mathcal{X}} p^a_{\mathbf{x}\mathbf{x}'} \sum_{a' \in \mathcal{A}} \pi(\mathbf{x}', a') Q^\pi(\mathbf{x}', a')$. In value-based RL, the optimal policy π^* is extracted from the optimal value functions $Q^*(\mathbf{x}, a) = r^a_{\mathbf{x}} + \gamma \sum_{\mathbf{x}' \in \mathcal{X}} p^a_{\mathbf{x}\mathbf{x}'} \max_{a' \in \mathcal{A}} Q^*(\mathbf{x}', a')$. In contrast, policy based RL methods are also popular for directly optimizing the policy to obtain the optimal policy [1], but are not leveraged in this work.

B. SMDP and temporal abstraction

To extend the MDP framework for temporal abstraction, we adopt the "option framework" [10], [12] where option refers to temporally extended action, also known as 'multi-time actions'. Unlike single timestep actions, options can be executed for multiple timesteps. If options are terminated after one timestep then the framework is the same as the traditional single-step action RL framework.

Definition 1. (Option) An option is a three element tuple $\langle \mathcal{I}, \pi, \beta \rangle$ where the option starts from current state $\mathbf{x}_t$ if and only if $\mathbf{x}_t \in \mathcal{I}$ where $\mathcal{I} \subseteq \mathcal{X}$, follows intra-option policy π and terminates according to the probability $\beta : \mathcal{X} \to [0, 1]$.

In the presence of a set of options, the MDP is elevated to a new decision making process known as semi-Markov Decision Process (SMDP) [10]. In the above formulation it is also imperative to identify whether an option is markov or semi-markov. Let $\tau^{o_t}_{\Delta} = \{\mathbf{x}_t, a_t, \mathbf{x}_{t+1}, a_{t+1}, \dots, a_{t+\Delta-1}, \mathbf{x}_{t+\Delta}\}$ be the trajectory created from the execution of option o_t starting at timestep t and continuing for Δ timesteps before termination. Then o_t is Markov if for any $\tau = \{\tau : t \leq \tau \leq \Delta\}$, policies and termination conditions depend entirely on $\mathbf{x}_\tau$. In contrast, o_t is semi-Markov if policies and termination conditions depend on the history of the trajectory up to τ. Just as MDPs have a policy π that chooses actions from action-space $\mathcal{A}$, SMDPs also have policy-over-options $\mu : \mathcal{X} \to \mathcal{O}$ that chooses option from the option-space $\mathcal{O}$. To finalize the option framework for an SMDP, we can make the following definitions of the transition dynamics, Eq. 3, and expected reward, Eq. 4, which are analogous to Eq. 1 and Eq. 2 respectively for an MDP.

$$p^o_{\mathbf{x}\mathbf{x}'} = \sum_{k=1}^{\infty} \gamma^k p(\mathbf{x}', k) \tag{3}$$

$$r^o_{\mathbf{x}} = \mathbb{E}[\sum_{k=0}^{\Delta-1} \gamma^k R_{t+k} | \mathbf{x}_t, a_t] \tag{4}$$

where $p(\mathbf{x}', k)$ is the probability of terminating the option at state $\mathbf{x}'$ after k timesteps. Value functions, option-value functions and their corresponding optimal counterparts for an SMDP can be defined as we did for an MDP. For example, the optimal option-value function can be defined in a way similar to optimal action-value in an MDP, $Q^*_{\mathcal{O}}(\mathbf{x}, o) = r^o_{\mathbf{x}} + \sum_{\mathbf{x}' \in \mathcal{A}} p^o_{\mathbf{x}\mathbf{x}'} Q^*_{\mathcal{O}}(\mathbf{x}', o')$. SMDPs have a key limitation in the sense that each option is treated as one single unit and there is hardly any way to investigate how each option is made. A more efficient way might be to interrupt options before completing the whole sequence or in case of markov options, to use "intra-option learning" [10], [12]. Note that when the options are semi-Markov, intra-option learning might not be

feasible as in that case we need to wait until the semi-Markov option terminates before we can evaluate it.

III. Temporal Abstraction for Transfer Learning

A. Mechanism and performance bound

Using the formalization from the previous section, we establish the framework for using temporal abstractions in transfer learning in the context of RL. We argue that temporal abstraction provides guided exploration in the target environment while also keeping the number of interactions feasible. Intuitively our idea is to execute the optimal policy learned in the source environment in small segments and observe the feedback at the end of each execution to develop sample efficient exploration strategies in the target environment. Informally this idea can be visualized as in Fig. 2. Here we elevate the source MDP to an SMDP and then create 'useful' temporal abstractions that are transferred to the target SMDP which is elevated from the target MDP.

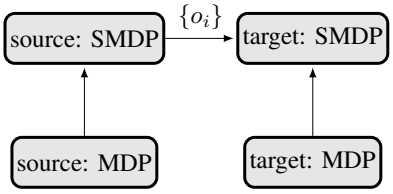


Fig. 2: Transferring temporal abstraction from a source to target environment where $\{o_i\}$ is the set of temporal abstractions

One key challenge in transferring temporal abstractions is that their usefulness decrease substantially if the source and target SMDP are not similar even in only one of the MDP elements. In our problem setting the source MDP is a possibly inaccurate estimation of the target MDP which makes decision making challenging due to the changes in the elements of the MDPs. To this end we restrict ourselves to the case where the source reward function $\mathcal{R}_{\mathcal{S}}$ is an inaccurate estimation of the target reward function $\mathcal{R}_{\mathcal{T}}$. Note that the transition probabilities may also vary between these two environments but this consideration is out of the scope of this paper. We provide two value-function based performance bounds for this transfer learning strategy. We draw motivations from "changing MDPs" [13] and transfer RL [14] and extend their results to SMDPs in theorem 1 and corollary 1.1. Specifically, theorem 1 provides the worst-case performance bound when each environment uses its own optimal temporal abstractions, meaning each environment uses its own optimal option-policy. Corollary 1.1 provides the worst-case performance bound when the optimal temporal abstractions from the source task is directly implemented on the target task without any modification, meaning we execute the optimal option-policy from the source environment in an offline setting in the target environment. Note that in this study we are interested in semi-Markov options only as we focus more on creating useful options and executing them on the target environment rather than interrupting them. Each of our semi-Markov option consists of primitive actions from the action space $\mathcal{A}$.

Theorem 1. *Let two MDPs, $\mathcal{M}_{\mathcal{S}} = \langle \mathcal{X}, \mathcal{A}, \mathcal{P}, \mathcal{R}_{\mathcal{S}}, \gamma \rangle$ and $\mathcal{M}_{\mathcal{T}} = \langle \mathcal{X}, \mathcal{A}, \mathcal{P}, \mathcal{R}_{\mathcal{T}}, \gamma \rangle$ where $\mathcal{R}_{\mathcal{T}} \neq \mathcal{R}_{\mathcal{S}}$, $\mathcal{O}$ is the same set of options defined on both MDPs, $\mu^*_{\mathcal{S}}$ and $\mu^*_{\mathcal{T}}$ are the optimal policy over options in $\mathcal{M}_{\mathcal{S}}$ and $\mathcal{M}_{\mathcal{T}}$ respectively, then the option value bound is*

$$\|Q^{\mu^*_{\mathcal{T}}}_{\mathcal{O},\mathcal{T}}(\mathbf{x}, o) - Q^{\mu^*_{\mathcal{S}}}_{\mathcal{O},\mathcal{S}}(\mathbf{x}, o)\|_\infty \leq \frac{\|r^o_{\mathbf{x},\mathcal{T}} - r^o_{\mathbf{x},\mathcal{S}}\|_\infty}{1-\gamma^\Delta}$$

where, option o is executed for at least Δ timesteps and $r^o_{\mathbf{x},\mathcal{T}}, r^o_{\mathbf{x}.\mathcal{S}}$ are rewards obtained from execution of o starting from $\mathbf{x}$ in $\mathcal{M}_{\mathcal{T}}$ and $\mathcal{M}_{\mathcal{S}}$ respectively.

Proof: Let's define the supremum norm $||r^o_{\mathbf{x},\mathcal{T}} - r^o_{\mathbf{x},\mathcal{S}}||_\infty = \max_{\mathbf{x},o} |r^o_{\mathbf{x},\mathcal{T}} - r^o_{\mathbf{x},\mathcal{S}}|$ and $||Q^{\mu^*_{\mathcal{T}}}_{\mathcal{O},\mathcal{T}}(\mathbf{x}, o) - Q^{\mu^*_{\mathcal{S}}}_{\mathcal{O},\mathcal{S}}(\mathbf{x}, o)||_\infty = \max_{\mathbf{x},o} |Q^{\mu^*_{\mathcal{T}}}_{\mathcal{O},\mathcal{T}}(\mathbf{x}, o) - Q^{\mu^*_{\mathcal{S}}}_{\mathcal{O},\mathcal{S}}(\mathbf{x}, o)|$. Also let's simplify the notations $Q^{\mathcal{T}}_{\mathcal{T}} \equiv Q^{\mu^*_{\mathcal{T}}}_{\mathcal{O},\mathcal{T}}(\mathbf{x}, o), Q^{\mathcal{S}}_{S} \equiv Q^{\mu^*_{\mathcal{S}}}_{\mathcal{O},\mathcal{S}}(\mathbf{x}, o), Q^{\mathcal{T}'}_{T} \equiv Q^{\mu^*_{\mathcal{T}}}_{\mathcal{O}}(\mathbf{x}', o'), r^o_{\mathcal{T}} \equiv r^o_{\mathbf{x},\mathcal{T}}, r_{\mathcal{S}} \equiv r^o_{\mathbf{x},\mathcal{S}}$. From the definition of optimal option-value function,

$$\begin{aligned}
&|Q^{\mathcal{T}}_{\mathcal{T}} - Q^{\mathcal{S}}_{\mathcal{S}}| \\
&= |r^o_{\mathcal{T}} + \sum_{\mathbf{x}'} p^o_{\mathbf{x}\mathbf{x}'} \max_{o \in \mathcal{O}_{\mathbf{x}'}} Q^{\mathcal{T}'}_{\mathcal{T}} - r^o_{S} + \sum_{\mathbf{x}'} p^o_{\mathbf{x}\mathbf{x}'} \max_{o \in \mathcal{O}_{\mathbf{x}'}} Q^{\mathcal{S}'}_{\mathcal{S}}| \\
&\leq |r^o_{\mathcal{T}} - r^o_{S}| + |\sum_{\mathbf{x}'} p^o_{\mathbf{x}\mathbf{x}'} \max_{o \in \mathcal{O}_{\mathbf{x}'}} Q^{\mathcal{T}'}_{\mathcal{T}} - \sum_{\mathbf{x}'} p^o_{\mathbf{x}\mathbf{x}'} \max_{o \in \mathcal{O}_{\mathbf{x}'}} Q^{\mathcal{S}'}_{\mathcal{S}}| \\
&\leq |r^o_{\mathcal{T}} - r^o_{S}| + \sum_{\mathbf{x}'} p^o_{\mathbf{x}\mathbf{x}'} | \max_{o \in \mathcal{O}_{\mathbf{x}'}} Q^{\mathcal{T}'}_{\mathcal{T}} - \max_{o \in \mathcal{O}_{\mathbf{x}'}} Q^{\mathcal{S}'}_{\mathcal{S}}| \\
&\leq |r^o_{\mathcal{T}} - r^o_{S}| + \sum_{\mathbf{x}'} \sum_{k=1}^{\infty} \gamma^k p(\mathbf{x}', k) \max_{o \in \mathcal{O}_{\mathbf{x}'}} |Q^{T'}_{\mathcal{T}} - Q^{S'}_{S}| \\
&\leq |r^o_{\mathcal{T}} - r^o_{S}| + \gamma^\Delta \sum_{k=\Delta}^{\infty} \sum_{\mathbf{x}'} p(\mathbf{x}', k) \max_{o \in \mathcal{O}_{\mathbf{x}'}} |Q^{T'}_{\mathcal{T}} - Q^{S'}_{S}| \\
&\leq ||r^o_{\mathcal{T}} - r^o_{S}||_\infty + \gamma^\Delta ||Q^{\mathcal{T}}_{\mathcal{T}} - Q^{\mathcal{S}}_{\mathcal{S}}||_\infty
\end{aligned}$$

As the above inequality is true for all $\mathbf{x} \in \mathcal{X}, o \in \mathcal{O}$ the following is true as well

$$||Q^{\mu^*_{\mathcal{T}}}_{\mathcal{O},T} - Q^{\mu^*_{\mathcal{S}}}_{\mathcal{O},\mathcal{S}}||_\infty \leq ||r^o_{T} - r^o_{S}||_\infty + \gamma^\Delta ||Q^{\mu^*_{\mathcal{T}}}_{\mathcal{O},T} - Q^{\mu^*_{\mathcal{S}}}_{\mathcal{O},\mathcal{S}}||_\infty . \square$$

In the following, we provide a straight forward extension of the above theorem which gives the option-value bound while following the same option-policy in both environments. Here we establish a useful performance bound in corollary 1.1 that compares the optimal option-value function in $\mathcal{M}_{\mathcal{T}}$ while following $\mu^*_{\mathcal{S}}$ obtained from $\mathcal{M}_{\mathcal{S}}$.

Corollary 1.1. *If $\mu^*_{\mathcal{T}}$ and $\mu^*_{\mathcal{S}}$ are the optimal option-policy in $\mathcal{M}_{\mathcal{T}}$ and $\mathcal{M}_{\mathcal{S}}$ respectively, then*

$$\|Q^{\mu^*_{\mathcal{T}}}_{\mathcal{O},T}(\mathbf{x}, o) - Q^{\mu^*_{\mathcal{S}}}_{\mathcal{O},\mathcal{T}}(\mathbf{x}, o)\|_\infty \leq \frac{2\|r^o_{\mathbf{x},\mathcal{T}} - r^o_{\mathbf{x},\mathcal{S}}\|_\infty}{1-\gamma^\Delta}$$

Proof: using the simplified notations $Q^{\mathcal{S}}_{\mathcal{T}} \equiv Q^{\mu^*_{\mathcal{S}}}_{\mathcal{O},\mathcal{T}}(\mathbf{x}, o), Q^{\mathcal{S}}_{\mathcal{S}} \equiv Q^{\mu^*_{\mathcal{S}}}_{\mathcal{O},S}(\mathbf{x}, o), Q^{\mathcal{T}}_{\mathcal{T}} \equiv Q^{\mu^*_{\mathcal{T}}}_{\mathcal{O},\mathcal{T}}(\mathbf{x}, o)$,

$$
\begin{aligned}
&|Q^{\mathcal{T}}_{\mathcal{T}} - Q^{\mathcal{S}}_{\mathcal{T}}| = |Q^{\mathcal{T}}_{\mathcal{T}} - Q^{\mathcal{S}}_{\mathcal{S}} + Q^{\mathcal{S}}_{\mathcal{S}} - Q^{\mathcal{S}}_{\mathcal{T}}| \\
&\leq |Q^{\mathcal{T}}_{\mathcal{T}} - Q^{\mathcal{S}}_{\mathcal{S}}| + |Q^{\mathcal{S}}_{\mathcal{S}} - Q^{\mathcal{S}}_{\mathcal{T}}| \\
&\leq ||Q^{\mathcal{T}}_{\mathcal{T}} - Q^{\mathcal{S}}_{\mathcal{S}}||_\infty + ||Q^{\mathcal{S}}_{\mathcal{S}} - Q^{\mathcal{S}}_{\mathcal{T}}||_\infty
\end{aligned}
$$

Using results from theorem 1 we get that $||Q^{\mathcal{T}}_{\mathcal{T}} - Q^{\mathcal{S}}_{\mathcal{S}}||_\infty = ||Q^{\mathcal{S}}_{\mathcal{S}} - Q^{\mathcal{S}}_{\mathcal{T}}||_\infty \leq \frac{||r^o_{\mathcal{T}} - r^o_{\mathcal{S}}||_\infty}{1-\gamma^\Delta}$. □

B. Generating useful temporal abstractions

Initially, we train the agent in $\mathcal{M}_{\mathcal{S}}$ and extract optimal policy $\pi^*_{\mathcal{S}}$. As we have already assumed that $\mathcal{M}_{\mathcal{S}}$ is a physics guided model (PGM) that provides cheap data, any RL algorithm may be used to learn in $\mathcal{M}_{\mathcal{S}}$. Next we are interested in transferring $\pi^*_{\mathcal{S}}$ from $\mathcal{M}_{\mathcal{S}}$ to $\mathcal{M}_{\mathcal{T}}$ and create useful temporal abstractions or options. To achieve this we execute $\pi^*_{\mathcal{S}}$ for a fixed horizon length Δ from current state $\mathbf{x} \equiv \mathbf{x}_t$ to create a trajectory $\tau^{\pi^*_{\mathcal{S}}}_{\mathbf{x}} = \{\mathbf{x}_t, a_t, x_{t+1} \ldots, a_{t+\Delta-1}, \mathbf{x}_{t+\Delta}\}$ and reach state $\mathbf{x}' \equiv \mathbf{x}_{t+\Delta}$. Following this procedure we have created an option, o_s, from the source optimal policy $\pi^*_{\mathcal{S}}$. Directly executing o_s in $\mathcal{M}_{\mathcal{T}}$ will result in poor performance if $\mathcal{R}_S \neq \mathcal{R}_{\mathcal{T}}$. To account for the changes between $\mathcal{R}_{\mathcal{T}}$ and $\mathcal{R}_{\mathcal{S}}$, we propose to create multiple options instead of a single one and perform an option optimization procedure to choose the best option. We perform this procedure in two steps. First, we perform probabilistic policy reuse [15] to create a set of options $\{o_i\}$ from o_s. This means that, each option o_i follows o_s probabilistically with occasional exploration. In this context each option can be defined as follows:

$$o_i \equiv \langle \mathbf{x}, \epsilon\text{-greedy}(\pi^*_{\mathcal{S}}), \mathbf{x}' \rangle \tag{5}$$

We summarize the approach for creating a set of options $\{o_i\}$ from o_s in algorithm 1.

Second, we execute the set of options in $\mathcal{M}_{\mathcal{T}}$, observe the feedback and optimize the set of options based on some utility function $\mathcal{U}(\cdot)$ to choose the best option, o_b. The utility function may be defined in a wide variety of ways. It can directly be the final reward obtained after reaching state $\mathbf{x}'$ from the previous state, or it can be the cumulative sum of all the rewards obtained from the execution of the option in $\mathcal{M}_{\mathcal{T}}$. Alternatively the utility function may also describe whether we have reached a certain preferred state.

$$o_b = \underset{o \in \{o_i\}}{\arg\max}\ \mathcal{U}(o) \tag{6}$$

Each time we observe a reward from $\mathcal{M}_{\mathcal{T}}$ we also fine tune $\pi^*_{\mathcal{S}}$ by updating the optimal Q-functions from the source, $Q^*_{\mathcal{S}}$. Finally, we choose the state $\mathbf{x}'$ obtained using o_b to repeat this whole procedure. We keep repeating this procedure as long as we are allowed to interact with $\mathcal{M}_{\mathcal{T}}$.

Algorithm 1 Creating temporal abstractions

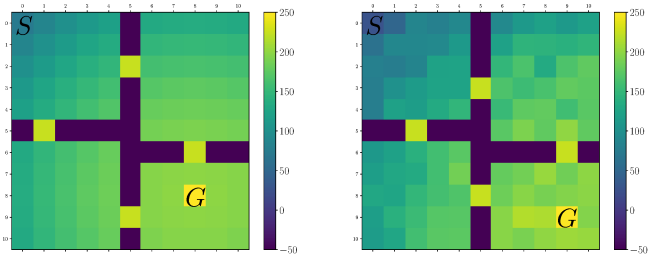
function CREATE OPTION($\mathbf{x}_0$, π^*_S, ϵ, Δ)
  $\tau^{\pi^*_S}_{\mathbf{x}} \leftarrow (\mathbf{x}_0, a_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\Delta-1}, a_{\Delta-1}, \mathbf{x}_\Delta)$
  for $t = 0$ to $\Delta - 1$ **do**
    if random_number $< \epsilon$ **then**
      $a_t \sim$ Uniform($\{1, 2, \ldots, |\mathcal{A}|\}$)
    else
      $a_t = \pi^*_S(\mathbf{x}_t)$
    $\mathbf{x}_{t+1}, R_t \leftarrow$ env($\mathbf{x}_t, a_t$)
    save a_t
    $\mathbf{x}_t \leftarrow \mathbf{x}_{t+1}$
  $option, o_i = \{a_0, a_1, \ldots, a_{\Delta-1}\}$
return $option, o_i$

IV. Simulation Experiments

Here we implement the method proposed in the previous section in two different tasks. The first one is a modified version of the classic four-room problem in RL that acts as a toy problem for demonstrating the effectiveness of temporal abstraction. The second one is motivated by real-world autonomous applications that can be benefited by the proposed method. In the second one, we implement our method in a case study of 'autonomous manufacturing systems'. In both tasks we consider learning in the target environment where source reward function $\mathcal{R}_{\mathcal{S}}(\mathbf{x}, a)$ is a biased deterministic representation of unknown stochastic target reward function $\mathcal{R}_{\mathcal{T}}(\mathbf{x}, a)$.

A. Modified four room problem

There are four rooms separated by walls, four hallways between these rooms and a goal state in one of the rooms. This benchmark problem is different from the classic four room problem in the sense that rewards are not sparse in our case, rather each state provides a positive reward that comes from a reward distribution. We also assume that the walls are penetrable and they provide negative rewards (-50 in this case) except the hallway state. Each hallway has a high positive reward, $+200$, compared to the adjacent states and the goal state, G, has the highest positive reward, $+250$. The goal of the agent is to reach G starting from the initial state, S, while also accumulating high rewards. For the source reward function $\mathcal{R}_{\mathcal{S}}$ we use certain positions of the hallways and goal state, while for the target reward function, $\mathcal{R}_{\mathcal{T}}$, we shift these hallways and goal state with added stochastic noise, $\xi \sim \mathcal{N}(0, 5)$. This stochasticity makes the reward uncertain and thus makes learning in the target task challenging.



(a) $\mathcal{R}_{\mathcal{S}}$ (b) $\mathcal{R}_{\mathcal{T}}$

Fig. 3: Reward distribution for modified four room problem, S = starting state, G = goal state.

B. Autonomous manufacturing of metamaterials

As the second benchmark problem we consider the challenging task of autonomous manufacturing of metamaterials [7], [16]. Here the goal of the agent is to autonomously learn the process parameters of an additive manufacturing system that can manufacture an acoustic metamaterial artifact with desired spectral response without human intervention. The cost of data collection in this system is extremely high due to materials cost, prolonged manufacturing time etc. As a result, vanilla RL approaches can not be implemented for this application. In this problem setting, each state can be represented by two process parameters, x_1 and x_2 that control the the filament spacing, l_{xy} and filament diameter, d of an artifact respectively. By controlling these two parameters, it is possible to manufacture an artifact with desired spectral response $\mathbf{g}_d$. Conceptually, we start with some random values of the parameters, manufacture an artifact, observe the reward and take an action accordingly.

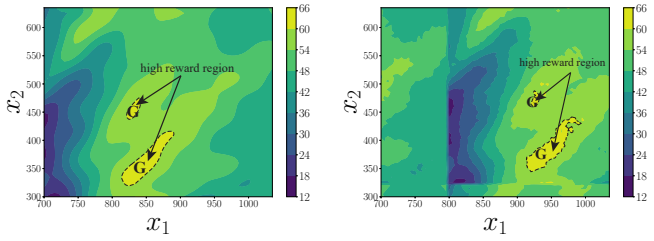
1) Dataset: For this case study we have developed a dataset of 4624 Finite Element Method (FEM) simulations where each simulation is performed for a distinct set of values of x_1 and x_2. The output of each simulation provides spectral response, $\mathbf{g}_a$, which is used in a custom similarity metric to compare against a user defined desired spectral response, $\mathbf{g}_d$. Next we use a baseline value to convert the losses into non-negative rewards.

$$R = B - \mathcal{L}(\mathbf{g}_a, \mathbf{g}_d) \tag{7}$$

where $\mathcal{L}(\cdot,\cdot)$ is a custom similarity metric similar to [16], B is the baseline value. To perform computational experiments we create two environments from this dataset. For the source environment $\mathcal{M}_\mathcal{S}$, we create a model of the rewards using the non-parametric Gaussian process [17] regression method. Here $\mathcal{M}_\mathcal{S}$ is the physics guided model, PGM. This reward model acts as $\mathcal{R}_\mathcal{S}$ (Fig. 4a). For the target environment $\mathcal{M}_\mathcal{T}$, a circular shift of the source rewards is performed such that the highest reward region of $\mathcal{R}_\mathcal{S}$ becomes the lowest reward region of $\mathcal{R}_\mathcal{T}$ (Fig. 4b). A small amount of stochastic noise is also added, $\xi \sim \mathcal{N}(0, 0.5)$, to the reward values to make the target reward values uncertain. This makes the learning problem challenging because the learned offline policy $\pi^*_\mathcal{S}$ will perform poorly in $\mathcal{M}_\mathcal{T}$. Our goal is to overcome this bias and accumulate high rewards while converging to a good terminal state $\mathbf{x}_T$, which should be in the high reward region, after T timesteps; meaning that the corresponding final artifact produces a spectral response that closely matches the desired, g_d.

C. Implementation details

To obtain the optimal policy in $\mathcal{M}_\mathcal{S}$, we train the agent in $\mathcal{M}_\mathcal{S}$ for sufficiently long time (10^5 and 10 million timesteps respectively). As traditional RL algorithms use sequential updates, only certain action values get updated. To break this correlation, we use random sample Q-learning as the training algorithm in both source environments. This off-policy algorithm uses the Q-learning update rule for learning, but collects data according to a random policy. Next we



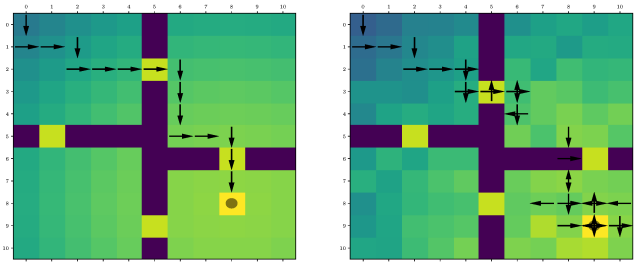
(a) $\mathcal{R}_\mathcal{S}$ (b) $\mathcal{R}_\mathcal{T}$

Fig. 4: Reward distribution for autonomous manufacturing problem, G = goal states with high rewards.

extract $\pi^*_\mathcal{S}$ from the trained Q-values. Using the method described earlier, 5 options are created for both benchmark problems. In each experiment the length of the options are kept constant, $\Delta = 7$ and $\Delta = 5$ for the four room problem and the autonomous manufacturing problem respectively. We optimize the options according to Eq. 6 where $\mathcal{U}(\cdot)$ is the reward obtained only from the final state of the trajectory, meaning $\mathcal{U}(o) = \mathcal{R}_\mathcal{T}(\mathbf{x}_{t+\Delta-1}, a_{t+\Delta-1})$. We allow the agent to learn and update π^*_{TAPRL} for 25 and 100 timesteps in the four room problem and the autonomous manufacturing problem respectively. This allows the agent to utilize a total of 125 and 500 samples collected from $\mathcal{M}_\mathcal{T}$ to learn π^*_{TAPRL} respectively. In this way we can obtain the empirical performance of π^*_{TAPRL}. For the modified four room problem we demonstrate the theoretically derived option-value bound in theorem 1 along with the empirical performance of the learned policy π^*_{TAPRL}. Here, we use value iteration for training in $\mathcal{M}_\mathcal{S}$. To demonstrate the performance of TAPRL for the autonomous manufacturing problem, three different initial condition tests are applied: a poor initial condition that is far from the high reward region and having the lowest reward region in-between ($\mathbf{x}_{0,b} = [725, 525]^T \mu m$), a good initial condition that is close to the high reward region ($\mathbf{x}_{0,g} = [925, 525]^T \mu m$) and finally 100 randomly chosen initial conditions. Each experiment is repeated 100 times to obtain the statistics of the collected rewards i.e. mean and standard deviation. Finally the performance of the learned policy π^*_{TAPRL} is evaluated by comparing the rewards accumulated by π^*_{TAPRL} against $\pi^*_\mathcal{S}$ and $\pi^*_\mathcal{T}$ for 100 timesteps. Note that $\pi^*_\mathcal{T}$ is the benchmark policy obtained by training the agent for 10 million timesteps in the target environment. The effect of exploration in creating temporal abstractions and the number of total samples used to learn π^*_{TAPRL} are also investigated.

V. Results

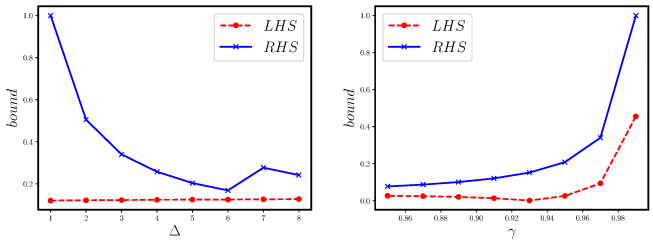
For the four room problem, actions chosen by π^*_{TAPRL} is shown in Fig. 5b. Although π^*_{TAPRL} explores more states than $\pi^*_\mathcal{S}$, the algorithm can still can reach the goal state while an offline implementation of $\pi^*_\mathcal{S}$ obtained from $\mathcal{M}_\mathcal{S}$ will converge to a poor final state as $\pi^*_\mathcal{S}$ does not take into consideration any feedback obtained from $\mathcal{M}_\mathcal{T}$. The theoretical bounds obtained for this toy problem are shown in Fig. 6. Interestingly, lowest option-value bound for this



(a) $\pi^*_{\mathcal{S}}$ in $\mathcal{M}_{\mathcal{S}}$ (b) π^*_{TAPRL} in $\mathcal{M}_{\mathcal{T}}$

Fig. 5: π^*_{TAPRL} obtained in the target task by leveraging knowledge from $\pi^*_{\mathcal{S}}$ and creating temporal abstractions to overcome bias from source rewards

specific problem is obtained when the length of the temporal abstraction is $\Delta = 6$ timesteps, see Fig. 6a. As expected, the option-value bound increases with the increase in the discount factor value, see Fig. 6b.

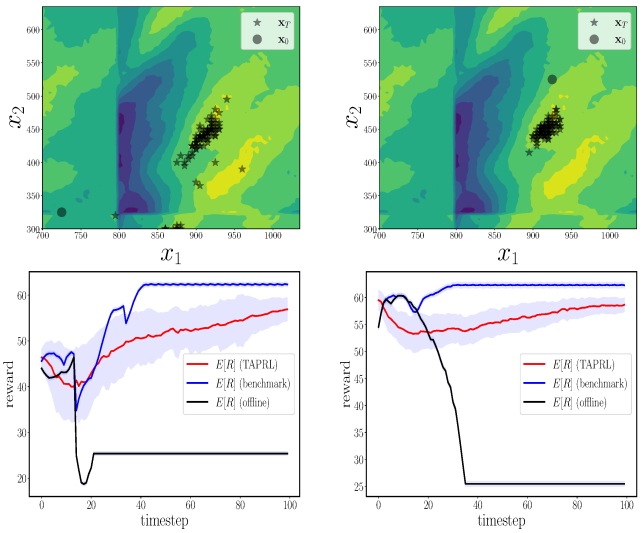


(a) temporal abstraction length (b) discount factor

Fig. 6: Option-value bound derived from theorem 1 in the four-room problem against various values of Δ and γ respectively

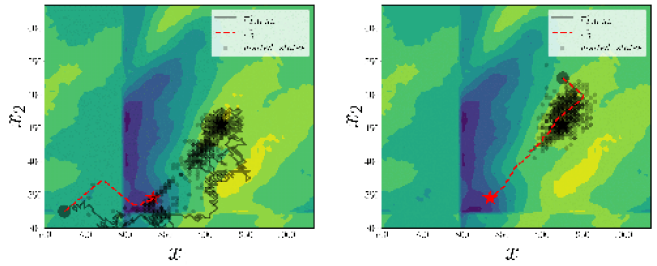
The effectiveness of TAPRL for the autonomous manufacturing problem is shown in Figs. 7 and 8. Figs. 8b and 8a demonstrates how π^*_{TAPRL} explores the state space from two different initial positions $\mathbf{x}_{0,b}$ and $\mathbf{x}_{0,g}$ described in Section IV-C. Eventually π^*_{TAPRL} overcomes the bias of the source rewards and finds the 'high reward region' in $\mathcal{M}_{\mathcal{T}}$ while $\pi^*_{\mathcal{S}}$ fails to do so. We also show the accumulated rewards using π^*_{TAPRL} in $\mathcal{M}_{\mathcal{T}}$ and corresponding final states for 100 experiments in Fig. 7. First, we show the final states and accumulated rewards obtained using π^*_{TAPRL} from the initial position $\mathbf{x}_{0,b}$ in Fig. 7(a). Despite this poor initialization π^*_{TAPRL} is capable of finding the 'high reward region' almost 80% of the times. Second, we show similar results for the initial position $\mathbf{x}_{0,g}$ in Fig. 7(b) and observe that π^*_{TAPRL} finds the 'high reward region' in almost all of the experiments. Finally we show similar results for 100 experiments with random initial positions in Fig. 10a and it is easy to see that π^*_{TAPRL} achieves high rewards almost 90% of the times. The corresponding reward plot in Fig. 10b shows that rewards accumulated using π^*_{TAPRL} is very close to the rewards accumulated using benchmark optimal policy $\pi^*_{\mathcal{T}}$ while the offline source optimal policy $\pi^*_{\mathcal{S}}$ performs worst in every case.

Fig. 11a shows the effects of exploration in creating temporal



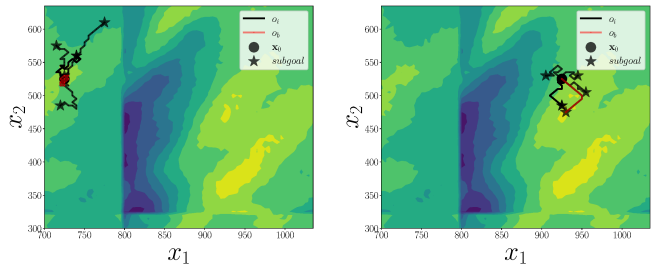
(a) (b)

Fig. 7: Top: Final states of 100 experiments obtained by running TAPRL for 100 timesteps, bottom: rewards collected by $\pi^*_{\text{TAPRL}}, \pi^*_{\mathcal{S}}, \pi^*_{\mathcal{T}}$, from (a) $\mathbf{x}_{0,b}$, (b) $\mathbf{x}_{0,g}$



(a) π^*_{TAPRL} from $\mathbf{x}_{0,b}$ (b) π^*_{TAPRL} from $\mathbf{x}_{0,g}$

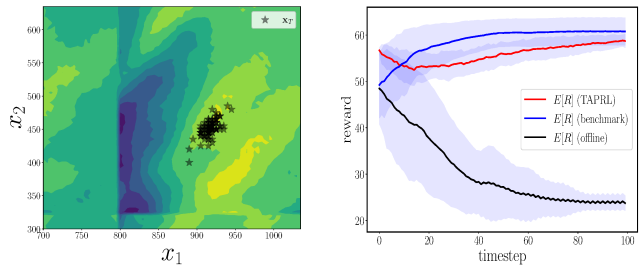
Fig. 8: π^*_{TAPRL} obtained from two different initial conditions, the color intensity of the small circles represent the number of times a state has been visited



(a) (b)

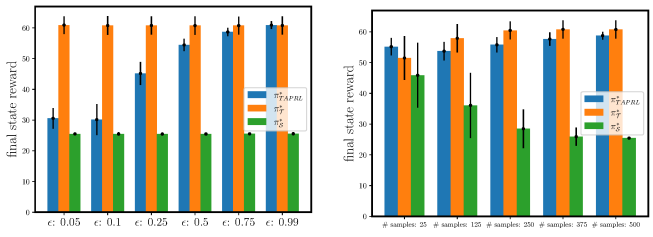
Fig. 9: Two examples of 5 temporal abstractions created in $\mathcal{M}_{\mathcal{T}}$ from two different states

abstractions and a clear pattern shows that reward from the final state gets higher as we increase the exploration value. This makes sense because high exploration leads to options that are not necessarily very close to the optimal option suggested by $\pi^*_{\mathcal{S}}$ and thus encouraging exploration. Fig. 11b shows the effect of number of samples used to obtain π^*_{TAPRL};



(a) Final states, $\mathbf{x}_T$ (b) rewards

Fig. 10: Final states and rewards obtained from 100 randomly chosen initial conditions, for (b) each experiment is repeated 100 times and rewards are shown within one standard deviation limit



(a) (b)

Fig. 11: Rewards obtained from the final state using $\pi^*_{\text{TAPRL}}, \pi^*_{\mathcal{S}}, \pi^*_{\mathcal{T}}$ for 100 experiments with random initial conditions (a) value of exploration in creating options (b) total number of used samples from $\mathcal{M}_{\mathcal{T}}$, rewards are shown within one standard deviation limit

final state with higher reward can be obtained by using a larger number of samples from $\mathcal{M}_{\mathcal{T}}$ although in all of the cases, the number of samples are reasonable.

VI. Conclusion

Implementing RL algorithms in physical system is challenging due to lack of sample efficiency and we believe that our algorithm can provide interesting perspectives for systems where data collection is expensive. This study presents a simple, but effective transfer approach in RL that creates temporal abstractions from a learned physics guided source model. We empirically show that this method can achieve and maintain high rewards in a target environment even if the source provides different rewards and with a small number of samples of the environment, in comparison to vanilla RL. The demonstration has been provided by developing two benchmark problems with two distinct applications. We provided an extreme scenario in one of the benchmark problems where the highest reward region in the source is shifted by the lowest reward region in the target and still the agent is capable of getting high rewards. We believe there are two important limitations of the proposed algorithm. First, creating too many temporally extended actions will increase the number of samples. Second, scalability might be a challenge in high dimensional problems. Additional sampling techniques may be used for creating temporal abstractions to improve the performance by incorporating application specific noise models in the source reward function.

References

[1] R. S. Sutton, A. G. Barto, *et al.*, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 135.

[2] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[4] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[5] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on machine learning (ICML-11)*. Citeseer, 2011, pp. 465–472.

[6] D. J. Corbin, A. R. Nassar, E. W. Reutzel, A. M. Beese, and N. A. Kistler, "Effect of directed energy deposition processing parameters on laser deposited inconel® 718: External morphology," *Journal of Laser Applications*, vol. 29, no. 2, p. 022001, 2017.

[7] M. F. A. Max Shtein, Kira Barton and D. J. Hoelzle, "A physics guided reinforcement learning framework for an autonomous manufacturing system with expensive data," in *2021 American Control Conference (ACC)*, 2021.

[8] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey." *Journal of Machine Learning Research*, vol. 10, no. 7, 2009.

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

[10] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[11] M. L. Puterman, "Markov decision processes," *Handbooks in operations research and management science*, vol. 2, pp. 331–434, 1990.

[12] D. Precup, "Temporal abstraction in reinforcement learning." 2001.

[13] B. C. Csáji and L. Monostori, "Value function based reinforcement learning in changing markovian environments." *Journal of Machine Learning Research*, vol. 9, no. 8, 2008.

[14] A. Barreto, W. Dabney, R. Munos, J. J. Hunt, T. Schaul, H. Van Hasselt, and D. Silver, "Successor features for transfer in reinforcement learning," *arXiv preprint arXiv:1606.05312*, 2016.

[15] F. Fernández and M. Veloso, "Probabilistic policy reuse in a reinforcement learning agent," in *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 2006, pp. 720–727.

[16] M. F. Alam, M. Shtein, K. Barton, and D. J. Hoelzle, "Autonomous manufacturing using machine learning: A computational case study with a limited manufacturing budget," in *ASME 2020 15th International Manufacturing Science and Engineering Conference*. American Society of Mechanical Engineers Digital Collection, 2020.

[17] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer School on Machine Learning*. Springer, 2003, pp. 63–71.