ACuTE: Automatic Curriculum Transfer from Simple to Complex Environments

Yash Shukla
Tufts University
Medford, USA
yash.shukla@tufts.edu

Christopher Thierauf
Tufts University
Medford, USA
christopher.thierauf@tufts.edu

Ramtin Hosseini
Tufts University
Medford, USA
ramtin.hosseini@tufts.edu

Gyan Tatiya
Tufts University
Medford, USA
gyan.tatiya@tufts.edu

Jivko Sinapov
Tufts University
Medford, USA
jivko.sinapov@tufts.edu

ABSTRACT

Despite recent advances in Reinforcement Learning (RL), many problems, especially real-world tasks, remain prohibitively expensive to learn. To address this issue, several lines of research have explored how tasks, or data samples themselves, can be sequenced into a curriculum to learn a problem that may otherwise be too difficult to learn from scratch. However, generating and optimizing a curriculum in a realistic scenario still requires extensive interactions with the environment. To address this challenge, we formulate the *curriculum transfer* problem, in which the schema of a curriculum optimized in a simpler, easy-to-solve environment (e.g., a grid world) is transferred to a complex, realistic scenario (e.g., a physics-based robotics simulation or the real world). We present "ACuTE", Automatic Curriculum Transfer from Simple to Complex Environments, a novel framework to solve this problem, and evaluate our proposed method by comparing it to other baseline approaches (e.g., domain adaptation) designed to speed up learning. We observe that our approach produces improved jumpstart and time-to-threshold performance even when adding task elements that further increase the difficulty of the realistic scenario. Finally, we demonstrate that our approach is independent of the learning algorithm used for curriculum generation, and is Sim2Real transferable to a real world scenario using a physical robot.

KEYWORDS

Curriculum Learning; Transfer Learning; Reinforcement Learning

ACM Reference Format:
Yash Shukla, Christopher Thierauf, Ramtin Hosseini, Gyan Tatiya, and Jivko Sinapov. 2022. ACuTE: Automatic Curriculum Transfer from Simple to Complex Environments. In *Proc.of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 9 pages.

1 INTRODUCTION

Curriculum Learning (CL) attempts to optimize the order in which an agent accumulates experience, increasing performance while reducing training time for complex tasks [8, 21, 22]. The core of CL is to generalize the experience and knowledge acquired in simple tasks and leverage it to learn complex tasks. Viable results are achieved in simulation, where the system dynamics can be easily modeled and the environment is predictable. One major limitation of many curriculum learning approaches is that the time to generate the curriculum is greater than the time to learn the target task from scratch, which prohibits the use of such methods in complex, real-world, high-fidelity domains [21]. The useful scenario of transfer to the real world remains challenging: System identification, domain adaptation, and domain randomization have performed Sim2Real transfer by attempting to match simulation with the physical environment (see Sim2Real Transfer), but these methods are elaborate and time-consuming if the simulation dynamics are expensive.

Since the dynamics of high-fidelity (HF) environments may not lend themselves to optimize a curriculum [35, 36], we propose learning the curriculum in a simplified version of the HF environment, which we call the low-fidelity (LF) environment. Parameters from each LF task can then be transferred, generating a corresponding task in the HF environment. This curriculum transfer problem is an open question (see [21]), and to our knowledge, our novel approach is the first to address this problem. We refer to this as transferring the *schema* of the curriculum: only task parameters are transferred as to address situations where policies and value functions cannot be directly transferred due to differences in the observation and action spaces. High-level task descriptions can be used to model inter-task relationships, and tasks with similar task-descriptors are shown to aid positive transfer [25, 29]. We show that our curriculum transfer approach leads to a quicker convergence even in cases where the dynamics of the LF and HF environments are different enough such that traditional domain adaptation methods do not produce a sufficient boost in learning.

An overview of our approach is shown in Fig 1. We consider a complex task and call it the HF target task, and map it to its simplified LF representation. The simplified dynamics of the LF environment allows curricula generation and experimentation to avoid costly setup and expensive data collection associated with the HF environment. Once an optimized curriculum is generated in the LF environment, the task parameters are mapped to obtain their respective HF counterparts. We learn these source tasks iteratively, transferring skills, before learning the target task. Finally, we perform a demonstration with Sim2Real transfer from the HF environment to test our effectiveness on a physical TurtleBot.

Proc.of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online.

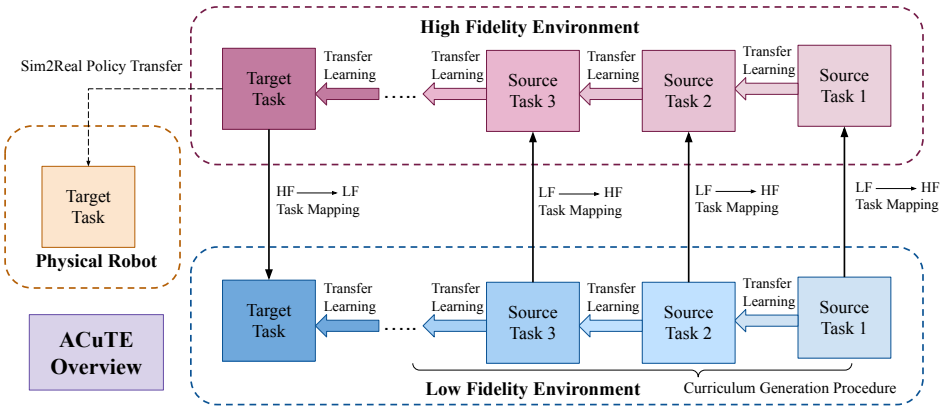


Figure 1: Overview of the proposed curriculum transfer approach ACuTE. The curriculum is generated and optimized in the low-fidelity environment, which is then mapped to generate a curriculum of the high-fidelity environment and learned before learning the final target task. The policy is then transferred to a physical robot.

In this work, we perform extensive experimental evaluation to demonstrate that curriculum transfer enables the agent to reduce the overall target task time compared to baselines. Through ACuTE, we propose an autonomous curriculum transfer method, which we refer to as "Automated Curriculum Transfer", that parameterizes the target task to generate and optimize the sequence of source tasks. We notice quick and efficient learning compared to baseline approaches present in literature such as Domain Adaptation [3], Self-Play [30] and Teacher-Student curriculum learning [18]. Additionally, we observe an improved jumpstart and time to threshold performance even when we add elements that make the HF target task too difficult to learn without a curriculum. We demonstrate that our approach is independent of the learning algorithm by showing improved performance in the HF environment when using a different learning algorithm from the one used when optimizing the curriculum, and also demonstrate positive transfer with imperfect mapping between the two environments. We observe that the Sim2Real transfer achieves successful task completion performance, equivalent to the HF agent's performance, on a physical TurtleBot.

2 RELATED WORK

Transfer Learning uses knowledge from learned tasks and transfers it to a complex target task [31]. Policy transfer is one such approach, in which the policy learned in a source task is used to initialize the policy for the next task in the curriculum [6, 14, 20, 31]. One popular transfer learning technique is to transfer the value function parameters learned in one task to initialize the value function of the next task in curriculum [1, 14–16].

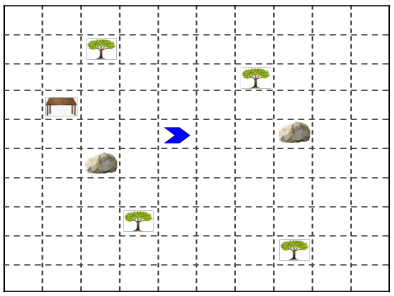
Sim2Real Transfer allows a model to be trained in simulation before deploying onto hardware, reducing time, cost, and safety issues. However, it encounters what Tobin *et. al.* [32] describe as the "Reality Gap" where a model does not allow an agent to train on realistic details. The same authors introduce "domain randomization" as a solution, later expanded upon by Peng *et. al.* [23]. Continual learning on incremental simulations can help tackle the sample inefficiency problem of domain randomization [10]. In contrast, Operational Space Control [12] avoids domain randomization while speeding up training with fewer hyperparameters. Carr *et. al.* [3] proposed a domain adaptation strategy approach, where state knowledge is transferred across semantically related games. Unlike aforementioned works, in this paper, we transfer the curriculum and not the policy to handle situations where Sim2Real fails, e.g., when the observation and action spaces of the simulation and real environment are different.

Curriculum Learning was introduced in the early 1990's, where it was used for grammar learning [7], control problems [26] and in supervised classification tasks [2]. CL has been extensively used for RL, with applications ranging from complex games [9, 34], to robotics [11], and self-driving cars [24]. In [21], the authors propose a framework for curriculum learning (CL) in RL, and use it to classify and survey existing CL algorithms. The three main elements of CL are task generation, task sequencing, and transfer learning. Task generation produces a set of source tasks that can be learned before learning the target task [13, 28]. Task sequencing orders the generated tasks to facilitate efficient transfer from the source to the target task [18, 22, 30]. Metaheuristic search methods are a popular tool to evaluate the performance of the task sequencing optimization framework [8]. In our work, we propose a framework to generate the source tasks and optimize their sequence, while evaluating performance against three baseline approaches. In most existing methods, generating and optimizing the curricula to learn a complex task is still time-consuming and sometimes takes longer compared to learning from scratch. Our proposed framework addresses this concern by generating, optimizing, and then transferring the schema of the curriculum from a simple and easy-to-learn environment to a complex and realistic environment.

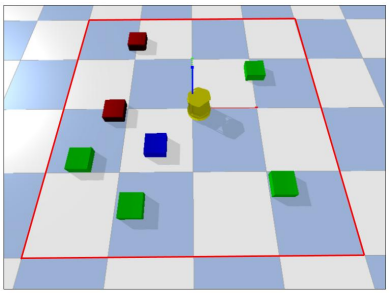
3 THEORETICAL FRAMEWORK

3.1 Markov Decision Processes

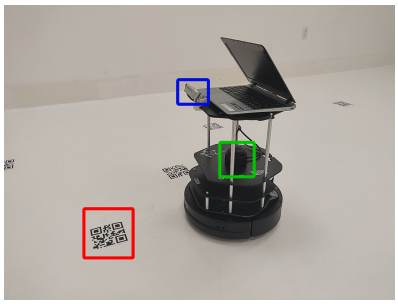
An episodic Markov Decision Process (MDP) M is defined as a tuple $(\mathcal{S},\mathcal{A},p,r,\gamma)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $p(s'|s,a)$ is the transition function, $r(s',a,s)$ is the reward function and $\gamma \in [0,1]$ is the discount factor. For each timestep t, the agent observes a state s and performs an action a given by its policy function $\pi_\theta(a|s)$, with parameters θ. The agent's goal is to learn an *optimal policy* π^*, maximizing its discounted return $G_0 = \sum_{k=0}^{K} \gamma^k r(s'_k, a_k, s_k)$ until the end of the episode at timestep K.



(a) Target task in Low Fidelity Environment



(b) Target task in High Fidelity Environment



(c) Target task in Physical Environment, using a camera (blue) to interact with fiducials (red). LIDAR (green) is also visible.

Figure 2: Illustration of the final target task in LF, HF and physical environment. The agent performs navigation, breaking action on 2 trees and 1 rock and then crafts a stone-axe at the crafting table to successfully terminate the task.

3.2 Curriculum Learning (CL)

We define a task-level curriculum as:
Let $\mathcal{T}$ be a set of tasks, where $M_i = (\mathcal{S}_i, \mathcal{A}_i, p_i, r_i)$ is a task in $\mathcal{T}$. Let $\mathcal{D}^{\mathcal{T}}$ be the set of all possible transition samples from tasks in $\mathcal{T}$: $\mathcal{D}^{\mathcal{T}} = \{(s, a, r, s') \mid \exists\, m_i \in \mathcal{T}\ s.t.\ s \in \mathcal{S}_i,\ a \in \mathcal{A}_i,\ s' \sim p_i(\cdot|s, a),\ r \leftarrow r_i(s, a, s')\}$. A curriculum $C = [M_1, M_2, \ldots, M_n]$ is an ordered list of tasks, where M_i is the i^{th} task in the curriculum. The ordered list signifies that samples from M_i must be used for training a policy before samples from M_{i+1} are used. The sequence of tasks terminates on the target task M_n.

3.3 Problem Formulation

The aim of CL is to generate a curriculum and train an agent on a sequence of tasks $\{M_1, M_2, ..., M_U\}$, such that the agent's performance on the final target task (M_U) improves relative to learning from scratch. The domains $\mathcal{T}^{HF}$ and $\mathcal{T}^{LF}$ of possible tasks are sets of MDPs in the high-fidelity (HF) and the low-fidelity (LF) environments, respectively. An individual task can be realized by varying a set of *parametric variables* and subjecting the task to a set of *constraints*. The *parametric variables* P are a set of attribute-value pair features $[P_1, ..., P_n]$ that parameterize the environment to produce a specific task. Each $P_i \in P$ has a range of possible values that the feature can take while the *constraints* of a domain are a set of tasks attained by determining the goal condition P_G.

Let $C_U^{\mathcal{T}_{LF}}$ be the set of all curricula over tasks $\mathcal{T}^{LF}$ of length U in the LF environment. Similarly, let $C_U^{\mathcal{T}_{HF}}$ be the set of all curricula over tasks $\mathcal{T}^{HF}$ of length U in the HF environment. The goal is to find a curriculum $c_U{}^{\mathcal{T}_{LF}}$ in the LF environment that can be transferred through a set of mapping functions $\mathcal{F} := \{f_1, f_2, \ldots, f_n\}$ to attain the curriculum in the HF environment $c_U{}^{\mathcal{T}_{HF}}$. A mapping function maps the parametric variables in the LF environment (P^{LF}) to the parametric variables in the HF environment (P^{HF}). We characterize the mapping as an affine transformation given by:

$$P^{HF} = A \odot P^{LF} + B$$

where $A = [a_1, \ldots, a_n]^T \in \mathbb{R}^n$ and $B = [b_1, \ldots, b_n]^T \in \mathbb{R}^n$ denote linear mapping and translation vector and $\odot$ is the Hadamard product. Thus, a parameter mapping ($f_i : P_i^{LF} \rightarrow P_i^{HF}$) is given by:

$$P_i^{HF} = a_i P_i^{LF} + b_i$$

The source tasks of the curriculum in the HF environment are learned before final target task as described in Section 3.2.

3.4 Running Example

For the physical environment shown in Fig 2c, we generate Crafter-TurtleBot (Fig 2b), a realistic simulation of the physical environment. The aim is to learn a policy in this high-fidelity (HF) environment, through an automated curriculum transfer from the low-fidelity (LF) environment (Fig 2a), and perform Sim2Real Policy transfer from the HF environment to execute the task in the physical environment.

The agent's goal is to break 2 trees to collect 2 pieces of wood, break a rock to collect a stone and craft a stone axe at the crafting table. The agent needs to navigate, face the object and perform the break action to collect it in inventory. The parametric variables for this task are the width (P_W) and height (P_H) of the navigable area, the number of trees ($P_{T,e}$), rocks ($P_{R,e}$), and crafting table (P_{CT}) present, the number of wood ($P_{T,i}$) and stones ($P_{R,i}$) present in the inventory of the agent when the episode starts, and the goal (P_G) of the task. The goal is drawn from a discrete set, which can be navigating to an item, breaking a subset of the items present in the environment or crafting the stone axe.

As described in Fig 1, the HF target task is mapped to its LF equivalent. The simplified LF dynamics allow efficient curriculum optimization. Once the curriculum is generated, each task of the LF is mapped back to generate an equivalent HF task, which are learned through a curriculum to develop a successful task policy for the target HF task. This policy is then transferred to the Physical Robot through a Sim2Real Transfer.

3.5 Curriculum Transfer Approach

ACuTE consists of three parts: Generating a LF target task, Curriculum generation in LF, and Task sequencing and learning in HF. Algorithm 1 presents our approach. The first step entails generating the LF target task from the HF target task. To obtain the task parameters for the LF task, we pass the HF target task parameters (P^{HF_U}) through the inverse of the affine mapping functions $f^{-1}(P^{HF_U})\ \forall f \in \mathcal{F}$ followed by `Generate_Env` (line 1), to obtain the parameters for the target task in the LF environment. The two requirements for obtaining a corresponding mapping are as follows:

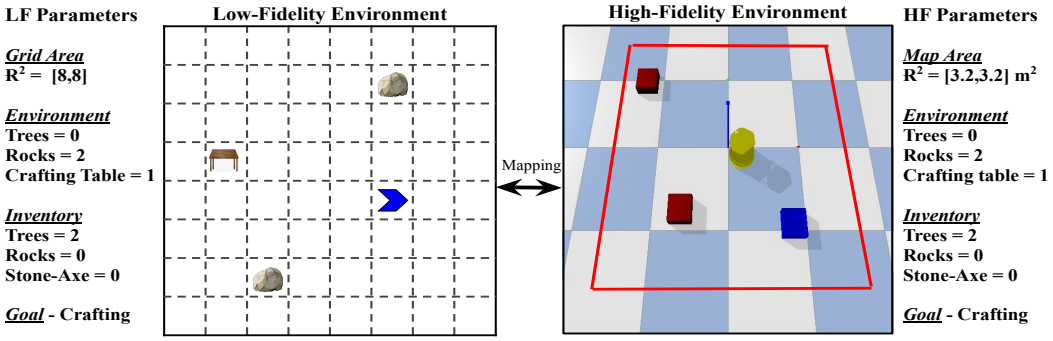


Figure 3: Illustrative example of the low-fidelity to high-fidelity mapping in the crafting task.

- Each task parameter in the HF environment (P^{HF}) needs a corresponding task parameter in the LF environment (P^{LF}). $\forall\, P_i^{HF} \in P^{HF}\ \exists\, P_i^{LF}$ s.t. $f_i(P_i^{LF}) = P_i^{HF}$. Varying these task parameters yields different tasks that are sequenced to form a curriculum.
- The final task in the HF environment must be mapped to the final task in the LF environment. $\forall f \in \mathcal{F}\ \exists\, f(P^{LF_U}) = P^{HF_U}$ Thus, we can guarantee that each source task obtained through the curriculum in the LF can be mapped to a corresponding task for the HF environment.

The actions in the HF can be complex, but each action can be simplified to an LF action that need not be correspondingly equivalent. Our approach does not assume an equivalency between the state or the action spaces between the LF and the HF environments, but only on the two requirements listed above. To generate and sequence the source tasks in the LF environment, we compared two approaches. The first approach, called Handcrafted Curriculum Transfer (HC), involves a human expert deciding the parametric variables P^{HC} for the tasks for the curriculum (line 3). The second approach, Automated Curriculum Transfer (AC), automatically generates and optimizes a sequence of source tasks from the parametric variables P^{AC} for the agent to learn the final task with the fewest number of episodes (line 5).

3.6 Handcrafted and Automated Curriculum Transfer Generation

To optimize curricula through Handcrafted Curriculum Transfer (`Generate_HC`) (HC), a human expert determines the parametric variables P^{HC} and the task sequence for the source tasks in the curriculum. Whereas, for optimizing curriculum using the Automated (AC) procedure (`Generate_AC`), we use the approach given in Algorithm 2. We start from an empty sequence of source task parameters P_W. The algorithm calls a parameterizing function (`Init_Src`) that assigns random values to the parametric variables for the first source task P^{LF_1} from the range of values P^{AC} can attain while simultaneously initializing an RL agent (`Init_Agent`). Based on P^{LF_1}, the algorithm generates the first task for the agent, M_1^{LF}, using the function (`Generate_Env`). The agent attempts learning this source task (`Learn`) with the initial policy $\pi_{1,w,init}$, until the *stopping criterion* is met. The *stopping criterion* determines if the agent's goal rate (δ) is $\geq \delta_G$ in the last s episodes (Algorithm 1 line 12). Failure to meet the *stopping criterion* implies that the agent has reached maximum permitted episodes (termed *budget* (b)) signifying $\delta < \delta_G$. The value for δ_G is set at 0.85 for our experiments.

In AC, the first task of the curriculum is randomly initialized N times and learned until *stopping criterion* is met. The algorithm then finds the W most promising solutions (`Best_Candidates`), based on fewest interactions to meet the *stopping criterion*. To optimize the sequence of the curriculum, we use beam search [17]. Beam search is a greedy search algorithm that uses a breadth-first search approach to formulate a tree. At each level of the tree, the algorithm sorts all the $(N \times W)$ successors of the tree (N successor tasks for each task of W) at the current level in increasing order of number of episodes required to learn the task $M_{u,w,n}^{LF}$. Then, it selects (W) number of best tasks at each level, given by fewest interactions to reach *stopping criterion*, and performs the same step until the tree reaches the desired number of levels (U).

Now, using the parametric variables P^{LF_1} for each task in the beam ($w \in W$), the algorithm generates parametric variables P^{LF_2} (`Init_Inter`) for the next task M_2^{LF} in the curriculum. This is done by choosing a goal P_G not encountered by the agent until the current level u in the beam w, and randomly initializing parametric variables $\geq$ the minimum required to accomplish this goal. The

Algorithm 1 `ACuTE`(N,W,U,P^{HF_U},f,s,b)

Output: HF target task policy: π_U^{HF}
Algorithm:

1: $M_U^{LF} \leftarrow$ `Generate_Env`$(f^{-1}(P^{HF_U}))$
2: **if** `curriculum = HC` **then**
3: C^{LF} =`Generate_HC`(M_U^{LF}, U)
4: **else if** `curriculum = AC` **then**
5: C^{LF}=`Generate_AC`$(M_U^{LF}, U, N, W, seeds)$
6: **end if**
7: $\pi_0^{HF} \leftarrow \emptyset$
8: **for** $u \in U$ **do**
9: $M_u^{HF} \leftarrow f(C_u^{LF})$
10: **while** *episode* $< b$ **do**
11: π_u^{HF} = `Learn`$(M_u^{HF}, \pi_{u-1}^{HF})$
12: **if** $\mathbb{E}[\pi_u^{HF}[episode{-}s{:}]] \geq \delta_G$ **then**
13: **break**
14: **end if**
15: **end while**
16: **end for**
17: **return** π_U^{HF}

Algorithm 2 `Generate_AC`$(N, W, U, M_U^{LF}, seeds)$

Output: LF Curriculum Parameters: P_W
Placeholder Initialization: Timesteps: $T_1, \ldots, T_U \leftarrow \emptyset$
LF task params for all tasks at each beam level $\xi_1, \ldots, \xi_U \leftarrow \emptyset$
LF task params at each width and level $\xi_{1,W}, \xi_{2,W}, \ldots, \xi_{U,W} \leftarrow \emptyset$
LF curriculum params $P_W \leftarrow \emptyset$
LF task policies for all tasks at each beam level: $\Pi_1, \ldots, \Pi_U \leftarrow \emptyset$
LF task policies for each width and level: $\Pi_{1,W}, \ldots, \Pi_{U,W} \leftarrow \emptyset$
Algorithm:

1: **for** $u \in U$ **do**
2: **for** $w \in W$ **do**
3: **for** $n \in N$ **do**
4: **if** $u = 1$ **then**
5: $P^{LF_u} \leftarrow$ `Init_Src`(M_U^{LF})
6: $\pi_{1,w,init}$ = `Init_Agent`$(seeds)$
7: **else**
8: $P^{LF_u} \leftarrow$ `Init_Inter`$(\xi_{u-1,W}[w], M_U^{LF})$
9: $\pi_{u,w,init}$ = `Load_Agent`$(\Pi_{u-1,W}[w])$
10: **end if**
11: $\xi_u\ [w, n] \leftarrow P^{LF_u}$
12: $M_{u,w,n}^{LF}$ = `Generate_Env`(P^{LF_u})
13: $(t_{u,w,n}, \pi_{u,w,,n,fin})$ = `Learn`$(M_{u,w,n}^{LF}, \pi_{u,w,n,init})$
14: $T_u\ [w, n] \leftarrow t_{u,w,n}$, $\Pi_u\ [w, n] \leftarrow \pi_{u,w,n,fin}$
15: **end for**
16: **end for**
17: $T_{u,W}, \Pi_{u,W}, \xi_{u,W}$ = `Best_Candidates`(T_u, Π_u, W, ξ_u)
18: **end for**
19: $P_W \leftarrow$ `Best_LF_Params`$(\xi_{1,W}, \xi_{2,W}, \ldots, \xi_{U,W})$
20: **return** P_W

agent then attempts learning M_2^{LF} with the final policy of the previous source task in the beam $\pi_{1,w,fin}$ (`Load_Agent`). The task terminates when the agent meets the *stopping criterion*. The algorithm finds the W most promising solutions, given by fewest interactions to reach *stopping criterion* (`Best_Candidates`) and carries out this procedure iteratively, until the final target task M_U^{LF} is learned. The parameters of the curriculum with the lowest number of episodes to reach the *stopping criterion* is selected as the most promising solution (`Best_LF_Params`) P_W for learning the target task. The curriculum generation procedure requires the length of the curriculum U to be >= the number of goals attainable. This ensures all the goals available (P_G) are encountered by the agent in the curriculum.

3.6.1 Task Sequencing and Learning in HF. Once the LF curriculum parametric variables (P_W) are obtained, they are passed through the set of mapping functions ($\mathcal{F}$) to attain the task parameters in HF, generating the curriculum source tasks from these parameters. The agent attempts learning the first source task M_1^{HF} with an initial policy $\pi_{1,init}^{HF}$. The task terminates when the agent meets the *stopping criterion*, generating the final policy $\pi_{1,fin}^{HF}$. This learned policy is used as an initial policy for the next source task M_2^{HF} in the curriculum. This procedure is carried out iteratively, culminating at the HF target task, returning π_U^{HF} (Algorithm 1, line 8-15).

4 EXPERIMENTAL RESULTS

We aim to answer the following questions: (1) Does the automated curriculum transfer yield sample efficient learning? (2) Does it scale to environments that are too difficult to learn from scratch? (3) Is the curriculum transfer framework independent of the RL algorithm used to generate the curriculum? (4) Can we perform a Sim2Real transfer to a physical robot? (5) Can the curriculum transfer framework yield successful convergence with imperfect (e.g., noisy) mappings between the HF and LF environments? [1]

To answer the first question, we evaluate our curriculum transfer method on grid-world as low-fidelity (LF) and Crafter-TurtleBot as the high-fidelity (HF) environments. In the LF environment, the agent can move 1 cell forward if the cell ahead is clear or rotate $\pi/2$ clockwise or counter-clockwise. In the target task, the agent receives a reward of $+1 \times 10^3$ upon crafting a stone axe, and -1 reward for all other steps. In the source tasks of the curriculum, the agent also receives +50 reward for breaking an item that is needed for crafting. This reward shaping is absent in the final target task. The agent's sensor emits a beam at incremental angles of $\pi/4$ to determine the closest object in the angle of the beam (i.e., the agents received a local-view of its environment). Two additional sensors provide information on the amount of wood and stones in the agent's inventory (See Appendix Section A.1 for further details).

The HF environment, Crafter-TurtleBot, is structurally similar to the grid-world domain, but differs in that objects are placed in continuous locations to more closely model the real-world. The agent is a TurtleBot, rendered in PyBullet [5]. An example of the LF $\leftrightarrow$ HF mapping between the tasks in LF and HF environments is shown in Fig 3. Here, the task mapping is demonstrated on an intermediate task of the curriculum, whose goal is to break a rock and craft a stone axe at the crafting table. The task mapping function ensures the LF and HF tasks have the same number and types of objects in the environments and the inventory. The mapping considers increased navigable area in the HF and does not assume the positions of the objects are preserved. In each episode, objects are positioned randomly within the boundaries of the environment. Refer to Appendix A.6 for details on mapping function set $\mathcal{F}$.

In the HF environment, the agent's navigation actions are moving forward 0.25 units and rotating by $\frac{\pi}{9}$ radians. The break and craft actions and the reward shaping in source tasks is identical to the LF environment. The HF agent's sensor is similar to the LF agent's sensor; however, it emits beams at incremental angles of $\frac{\pi}{10}$, accounting for the large state space of the location of objects.

To evaluate the performance of curriculum transfer, we used the *jumpstart* [8, 14] metric. *Jumpstart* evaluates the performance increase over D episodes after transfer from a source task as compared to a baseline approach. *Jumpstart* is defined as:

$$\eta_j := \frac{1}{D} \sum_{i=1}^{D} (G^i_{M_f^c} - G^i_{M_f^b})$$

where $G^i_{M_f^c}$ and $G^i_{M_f^b}$ are the returns obtained during episode i in task M_f^c (learning through automated curriculum transfer) and the baseline task M_f^b respectively. Another metric we used is the *time to threshold* metric [21, 31], which computes how faster an agent can learn a policy that achieves expected return $G \geq \delta$ on the target task if it transfers knowledge, as opposed to learning from another approach, where δ is desired performance threshold.

[1] Code available at: https://github.com/tufts-ai-robotics-group/ACuTE

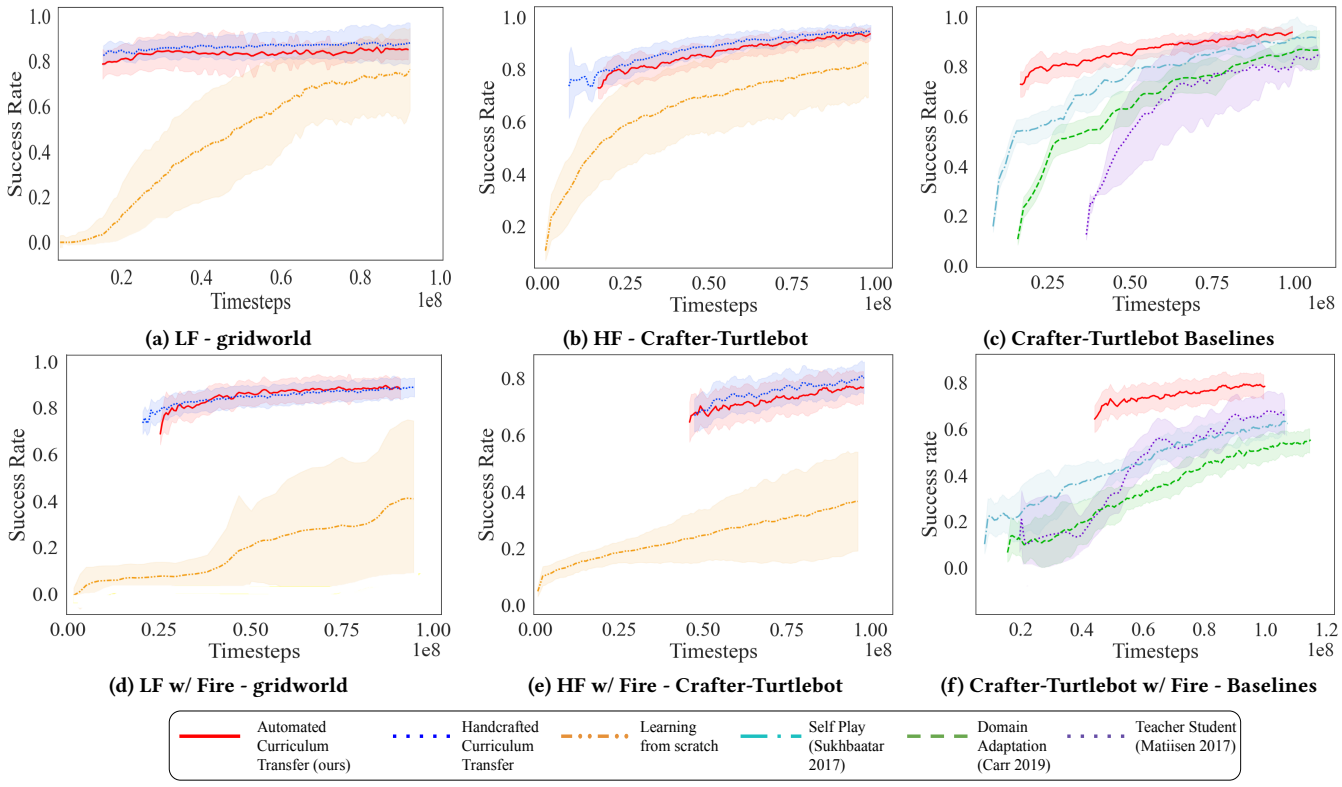


Figure 4: Learning Curves for low fidelity (LF) and high fidelity (HF) environments with and without Fire.

4.1 Curriculum Generation in High Fidelity

We used the algorithm presented in Algorithm 1 to generate and sequence the source tasks using the handcrafted curriculum transfer (HC) and the automated curriculum transfer (AC) approach. The navigable area in target task M_U^{LF} in low-fidelity (LF) environment is a grid of $\mathbb{R}^2_{LF} \rightarrow [10\times10]$, as seen in Fig 2a, and the high-fidelity (HF) target task area is a continuous $\mathbb{R}^2_{HF} \rightarrow [4m\times4m]$ space, as seen in Fig 2b. Both these environments contain 4 trees, 2 rocks and 1 crafting table, placed at random locations.

The RL algorithm was a Policy Gradient [33] network with ϵ-greedy action selection for learning the optimal policy. The episode terminates when the agent successfully crafts a stone axe or exceeds the total number of timesteps permitted, which is 10^2 in the LF environment and 6×10^2 in the HF environment. All experiments are averaged over 10 trials. (See Appendix Section A.2).

For generating the AC in the LF environment, we set the width of the beam search algorithm at $W = 4$, and the length of the beam at $N = 20$, the curriculum length was $U = 4$, and *budget* $b = 5 \times 10^3$. We obtained the parameters after performing a heuristic grid search over the space of the search algorithm. We evaluated our curriculum transfer framework with four other baselines. *Learning from scratch* trains the final HF target task without any curriculum. We also adopted three approaches from the literature designed for speeding-up RL agents: Asymmetric Self-play [30], Teacher-Student Curriculum learning [18] and Domain Adaptation for RL [3]. The first two baselines do not make use of the LF environment while the third uses the LF environment as the source domain. All baseline approaches involve reward shaping similar to the source task of the automated curriculum transfer approach.

The results in Figs 4a and 4b show that the AC approach results in a substantial improvement in learning speed, and is comparable to the curriculum proposed by a human expert. Furthermore, as shown in Fig 4f, the curriculum transfer method outperforms the three baseline approaches[2] in terms of learning speed. The learning curve for our curriculum transfer approaches has an offset on the x-axis to account for the time steps used to go through the curriculum before moving on to the target task, signifying *strong transfer* [31]. The other three baseline approaches perform better than learning from scratch, but do not outperform the curriculum transfer approach. Self-Play requires training a goal-proposing agent, which contributes to the sunk cost for learning. Whereas, Domain Adaptation relies on the similarity between the tasks. Teacher-Student curriculum learning requires defining the source tasks of the curriculum beforehand, and the teacher attempts to optimize the order of the tasks. All baselines involve interactions in the costly high-fidelity domain for generating/optimizing the curriculum, which proves to be costly. This sunk cost has been accounted in the learning curves by having an offset on the x-axis. See Appendix Section A.3 for details on our adaptation and tuning of these three baselines.

Table 1 compares the *jumpstart* values for AC with the baselines. The higher *jumpstart*, the better performance of the curriculum.

[2]Refer Appendix Section A.5 for learning curves for reward

Env	Methods	ΔJumpstart (Return)	ΔTime-to-threshold
HF	AC ⟶ Learning from scratch	304 ± 242	5.4 $\times 10^7$
	AC ⟶ Carr *et. al.*	231±160	3.5 $\times 10^7$
	AC ⟶ Sukhbaatar *et. al.*	85±130	2.1 $\times 10^7$
	AC ⟶ Matiisen *et. al.*	568 ±205	6.3 $\times 10^7$
HF w/ Fire	AC ⟶ Learning from scratch	628±284	1.02 $\times 10^8$
	AC ⟶ Carr *et. al.*	519±126	8.4 $\times 10^7$
	AC ⟶ Sukhbaatar *et. al.*	288±87	7.8 $\times 10^7$
	AC ⟶ Matiisen *et. al.*	346±121	4.6 $\times 10^7$

Table 1: Table comparing *jumpstart* (mean±SD), and *time to threshold* for learning the final target task. Here, HF and AC refer to high-fidelity and automated curriculum transfer respectively. Time-to-threshold measured in timesteps.

AC achieves high positive *jumpstart* and *time to threshold* performance in comparison to baseline approaches, denoting improved performance and quicker learning (magnitude $> 10^7$ timesteps).

4.2 Results with Added Complexity

To answer the second question, we evaluated our framework in a situation where the target task in the HF environment is too difficult to learn from scratch. To do this, both the LF and HF environments were modified by adding a new type of object, "fire", such that when the agent comes in contact with it, the episode terminates instantly with a reward of -1×10^3.

The parameters listed in Section 4.1 are used to optimize AC in the LF environment. From learning curves in Figs. 4d, 4e, 4f and Table 1, we observe our curriculum transfer approach, AC, consistently achieving higher average reward, better *jumpstart* and *time to threshold* performance compared to baselines. Learning from scratch fails to converge to a successful policy in 10^8 interactions, while the other baseline achieves marginally better performance than learning from scratch. Table 1 summarizes the jumpstart and the time-to-threshold between the approaches. Our approach still achieves a high jumpstart, and converges much quicker than other approaches. Through this experiment, we see our AC approach extrapolates to challenging environments, producing quicker and efficient convergence. Refer Appendix section A.1.1 for trends observed in different runs in our AC approach.

4.3 Results with Different RL Algorithms

To answer the third question, we conducted experiments by making the HF learning algorithm different (PPO [27] and DQN [19]) from the RL algorithm used for generating the curriculum in the LF environment (Policy gradient). Fig 5 shows the result of this test

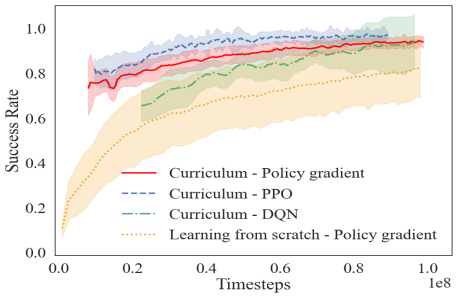


Figure 5: Learning HF target task through different RL algorithms.

(See Appendix A.4 for hyperparameters). In all the cases, learning through a curriculum is quicker and more efficient than learning from scratch. Here, we do not intend to find the best RL algorithm to solve the task, but demonstrate that actor-critic networks, policy gradients and value function based approaches learn the HF target task through curriculum, irrespective of the RL algorithm used to optimize the LF curriculum.

4.4 Noisy Mappings

In the above sections, we evaluated the curriculum transfer schema on accurate mappings between the two environments. In certain partially observable environments, it might not be possible to obtain such an accurate mapping between the two environments. To demonstrate the efficacy of our approach in imperfect mappings, we evaluate the experiments by incorporating noise in the mapping function. The noisy mapping function involves a multivariate noise over the range of the parametric variables. While obtaining the noisy parameters, we do not incorporate any noise in the parameter for the goal condition, as we can safely assume that the goals have been mapped accurately. The noisy mappings are given by:

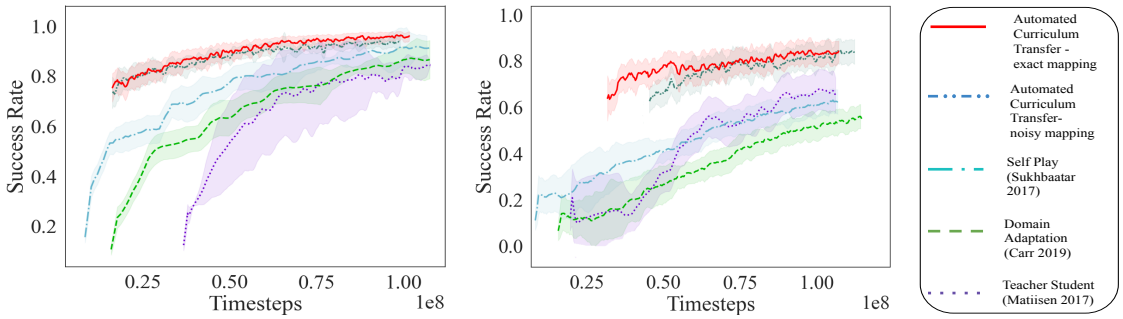
$$P_{noisy} = P_{exact} + \mathcal{N}(0, \Sigma)$$

where $\Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 & \dots \\ 0 & \sigma_2 & 0 & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \sigma_n \end{bmatrix}_{n \times n}$ is the covariance matrix, which is symmetric and positive semi-definite, and $\sigma_i = (max(P_i) - min(P_i))/6$, covers the entire range of the parametric variable in six standard deviations. We verify whether the noisy parameters meet the minimum requirements of reaching the goal for the sub-task. If the requirements are not met, we generate a new set of noisy parameters until the requirements are met.

The learning curves for the automated curriculum transfer generated through noisy parameters are shown in Fig 6. We compare its performance with learning curves for automated curriculum transfer generated through exact mappings, and with other baseline approaches present in the literature. Even with noisy mappings, the automated curriculum transfer outperforms other curriculum approaches, and performs comparable to the automated curriculum transfer with exact mappings. On the complicated task (HF with Fire), the automated curriculum transfer with noisy mappings takes longer to converge on the source tasks of the curriculum, yet it achieves a significant performance advantage over other baselines.

4.5 Runtime Comparison

Since our approach relies on the low-fidelity environment for curriculum generation, its sunk cost in computational runtime is significantly lesser than baseline approaches, in which curriculum generation requires extensive interactions in the costly high-fidelity environment. Fig 7 compares the computational runtime (in CPU Hours) required to run one trial (with each episode having a maximum of 6×10^2 timesteps) of the high-fidelity task until the *stopping criterion* is met. The experiments were conducted using a 64-bit Linux Machine, having Intel(R) Core(TM) i9-9940X CPU @ 3.30GHz processor and 126GB RAM memory. The sunk cost of our automated curriculum transfer approach involves the interactions required to



(a) Noisy mappings - HF **(b) Noisy mappings - HF w/ Fire**

Figure 6: Comparison of Learning Curves for high fidelity (HF) environments generated using noisy mappings.

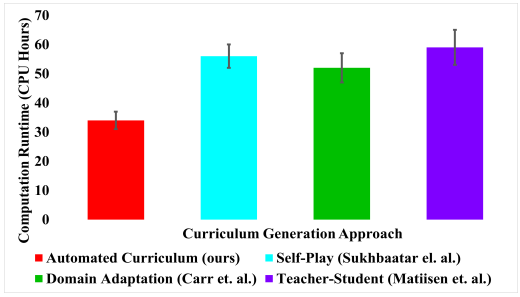


Figure 7: Runtime comparison of different approaches.

optimize the curriculum in the LF environment, and the interactions required to learn the source tasks in the HF environment.

5 TRANSFER TO A PHYSICAL ROBOT

To answer the fourth question and demonstrate the efficacy of this approach in the real-world, we performed a Sim2Real transfer on a TurtleBot after the simulated robot had learned the target task through the curriculum. We demonstrate object breaking and item retrieval, followed by crafting, through a TurtleBot which reads fiducials scattered throughout the environment.

We made use of the TurtleBot 2, modified to have an on-board laptop (running Ubuntu 16.04 and ROS Kinetic) which interfaces with a camera (the Intel D435i) and a LIDAR unit (the Slamtec RPLIDAR-A2). These additional sensors were used to provide more stable odometry via the Canonical Scan Matcher [4], as the default odometry stack was found to have too much drift to be relied upon. The full platform is visible in Fig 2c. Making use of the modified odometry stack, movement was provided as actions that attempt movement in the approximate units expected by the high-fidelity environment.

In this setting, the target task policy learned through our automated curriculum transfer approach in the high-fidelity Crafter-TurtleBot environment is transferred to run in this physical setting. No learning is taking place on the agent as the physical experimentation continues. These fiducials are QR codes corresponding to the different possible objects in the known environment: trees, rocks, and the crafting table. When demonstrated, the agent controls the TurtleBot to navigate to the Tree locations before calling the break action, and is then visible navigating to the Rock locations before again calling the break action. The agent completes the sequence by navigating to the crafting table before calling the craft action. The policy is run without any modifications.

It is important to note that breaking and crafting actions are successful only in the event of reading the fiducial. In this way, we restrict the agent to being successful only in the cases where it has operated successfully in the physical environment, allowing the agent to proceed only in the event of a sensible Sim2Real transfer.

6 CONCLUSION AND FUTURE WORK

We proposed a framework for automated curriculum transfer from an easy-to-solve environment to a real-world scenario. Our curriculum transfer approach generated results comparable to a curriculum generated by a human expert, exceeding the baseline performances. Our experimental evaluations show improved learning time and jumpstart performance on the final target task, even when additional challenging elements are introduced. We demonstrated ACuTE is independent of the RL algorithm used to generate the curriculum and is easily Sim2Real transferable to a physical robot setting and is also scalable to environments with inexact mappings.

An extension of our approach will be to scale the algorithm to multi-agent settings, with inter-agent curriculum transfer. A limitation of this work is that the task mapping is generated heuristically, a future work would involve automating the mapping generation. Future work can investigate how a LF version of the environment can be created autonomously, and providing a theoretical guarantee for the curriculum transfer approach. Second, while in this work we only transferred the schema of the curriculum, our baseline comparison with Domain Adaptation suggests that DA can be combined with curriculum transfer such that the agent can learn the tasks in the curriculum even faster. Finally, our algorithms for optimizing the curriculum in the LF environment did not make use of any data from the HF domain, and in future work, we plan to modify our framework to use interaction experience from both domains when constructing the curriculum.

ACKNOWLEDGMENTS

The research presented in this paper was conducted with partial support from DARPA (contract W911NF-20-2-0006) and AFRL (contract FA8750-22-C-0501).

REFERENCES

[1] David Abel, Yuu Jinnai, Sophie Yue Guo, George Konidaris, and Michael Littman. 2018. Policy and value transfer in lifelong reinforcement learning. In *International Conference on Machine Learning*. PMLR, 20–29.

[2] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proc.of the 26th annual Intl Conf. on machine learning*. 41–48.

[3] Thomas Carr, Maria Chli, and George Vogiatzis. 2019. Domain Adaptation for Reinforcement Learning on the Atari. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. 1859–1861.

[4] Andrea Censi. 2008. An ICP variant using a point-to-line metric. In *2008 IEEE International Conference on Robotics and Automation*. 19–25. https://doi.org/10.1109/ROBOT.2008.4543181

[5] Erwin Coumans and Yunfei Bai. 2016–2019. PyBullet, a Python module for physics simulation for games, robotics and machine learning. http://pybullet.org.

[6] Felipe Leno Da Silva and Anna Helena Reali Costa. 2019. A survey on transfer learning for multiagent reinforcement learning systems. *Journal of Artificial Intelligence Research* 64 (2019), 645–703.

[7] Jeffrey L Elman. 1993. Learning and development in neural networks: The importance of starting small. *Cognition* 48, 1 (1993), 71–99.

[8] Francesco Foglino, Christiano Coletto Christakou, and Matteo Leonetti. 2019. An optimization framework for task sequencing in curriculum learning. In *2019 Joint IEEE 9th Intl Conf. on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*. IEEE, 207–214.

[9] Yifan Gao and Lezhou Wu. 2021. Efficiently Mastering the Game of NoGo with Deep Reinforcement Learning Supported by Domain Knowledge. *Electronics* 10, 13 (2021), 1533.

[10] Josip Josifovski, Mohammadhossein Malmir, Noah Klarmann, and Alois and Knoll. 2020. Continual Learning on Incremental Simulations for Real-World Robotic Manipulation Tasks. In *2nd Workshop on Closing the Reality Gap in Sim2Real Transfer for Robotics at Robotics: Science and Systems (R:SS) 2020*. Nicht veröffentlichter Vortrag. https://sim2real.github.io/assets/papers/2020/josifovski.pdf

[11] Andrej Karpathy and Michiel Van De Panne. 2012. Curriculum learning for motor skills. In *Canadian Conference on Artificial Intelligence*. Springer, 325–330.

[12] Manuel Kaspar, Juan David Munoz Osorio, and Jürgen Bock. 2020. Sim2Real Transfer for Reinforcement Learning without Dynamics Randomization. *arXiv preprint arXiv:2002.11635* (2020).

[13] Anil Kurkcu, Domenico Campolo, and Keng Peng Tee. 2020. Autonomous Curriculum Generation for Self-Learning Agents. In *2020 16th Intl Conf. on Control, Automation, Robotics and Vision (ICARCV)*. IEEE, 1104–1111.

[14] Alessandro Lazaric. 2012. Transfer in reinforcement learning: a framework and a survey. In *Reinforcement Learning*. Springer, 143–173.

[15] Yong Liu, Yujing Hu, Yang Gao, Yingfeng Chen, and Changjie Fan. 2019. Value Function Transfer for Deep Multi-Agent Reinforcement Learning Based on N-Step Returns.. In *IJCAI*. 457–463.

[16] Yaxin Liu and Peter Stone. 2006. Value-function-based transfer for reinforcement learning using structure mapping. In *AAAI*. 415–420.

[17] Bruce T Lowerre. 1976. *The HARPY speech recognition system*. Technical Report. Carnegie Mellon University, PA, Department of Computer Science.

[18] Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. 2020. Teacher-Student Curriculum Learning. *IEEE Trans. Neural Networks Learn. Syst.* 31, 9 (2020), 3732–3740. https://doi.org/10.1109/TNNLS.2019.2934906

[19] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (Feb. 2015), 529–533. https://doi.org/10.1038/nature14236

[20] Akshay Narayan, Zhuoru Li, and Tze-Yun Leong. 2017. SEAPoT-RL: selective exploration algorithm for policy transfer in RL. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31.

[21] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. 2020. Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey. *J. of Machine Learning Research* 21 (2020), 1–50.

[22] Sanmit Narvekar, Jivko Sinapov, and Peter Stone. 2017. Autonomous Task Sequencing for Customized Curriculum Design in Reinforcement Learning.. In *IJCAI*. 2536–2542.

[23] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. 2018. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE Intl. Conf. on robotics and automation (ICRA)*. IEEE, 1–8.

[24] Zhiqian Qiao, Katharina Muelling, John M Dolan, Praveen Palanisamy, and Priyantha Mudalige. 2018. Automatically generated curriculum based reinforcement learning for autonomous vehicles in urban environment. In *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 1233–1238.

[25] Mohammad Rostami, David Isele, and Eric Eaton. 2020. Using task descriptions in lifelong machine learning for improved performance and zero-shot transfer. *Journal of Artificial Intelligence Research* 67 (2020), 673–704.

[26] Terence D Sanger. 1994. Neural network learning control of robot manipulators using gradually increasing task difficulty. *IEEE transactions on Robotics and Automation* 10, 3 (1994), 323–333.

[27] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *CoRR* abs/1707.06347 (2017). arXiv:1707.06347 http://arxiv.org/abs/1707.06347

[28] Felipe Leno Da Silva and Anna Helena Reali Costa. 2018. Object-oriented curriculum generation for reinforcement learning. In *Proc.of the 17th Intl Conf. on Autonomous Agents and MultiAgent Systems*. 1026–1034.

[29] Jivko Sinapov, Sanmit Narvekar, Matteo Leonetti, and Peter Stone. 2015. Learning inter-task transferability in the absence of target task samples. In *Proc.of the 2015 Intl Conf. on Autonomous Agents and Multiagent Systems*. 725–733.

[30] Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. 2018. Intrinsic Motivation and Automatic Curricula via Asymmetric Self-Play. In *International Conference on Learning Representations*.

[31] Matthew E Taylor and Peter Stone. 2009. Transfer learning for reinforcement learning domains: A survey. *J. of Machine Learning Research* 10, 7 (2009).

[32] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. 2017. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 23–30.

[33] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3-4 (1992), 229–256.

[34] Yuechen Wu, Wei Zhang, and Ke Song. 2018. Master-Slave Curriculum Design for Reinforcement Learning.. In *IJCAI*. 1523–1529.

[35] Yunzhi Zhang, Pieter Abbeel, and Lerrel Pinto. 2020. Asymmetric self-play for automatic goal discovery in robotic manipulation. *Advances in Neural Information Processing Systems Deep Reinforcement Learning Workshop* (2020).

[36] Yunzhi Zhang, Pieter Abbeel, and Lerrel Pinto. 2020. Automatic curriculum learning through value disagreement. *Advances in Neural Information Processing Systems* 33 (2020).