

Enhancing Sample-Efficiency in Reinforcement Learning Agents through Symbolic Reasoning

A dissertation submitted by

Yash Shukla

In partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

in

Computer Science

TUFTS UNIVERSITY

May 2025

Chair: Prof. Jivko Sinapov, Tufts University

Abstract

Reinforcement Learning (RL) researchers have made significant advancements in recent years, helping artificial agents to solve increasingly complex sequential decision-making tasks. However, many of these tasks remain prohibitively expensive to learn due to the number of interactions required with the environment. In several applications, symbolic information is readily available that can help improve the sample efficiency of RL agents. However, one key challenge remains in effectively leveraging this symbolic information, either by reducing the search space for the RL agent, or by providing a curriculum of increasingly complex goals.

One way to incorporate symbolic information to improve the sample efficiency of RL agents is through symbolic abstraction. However, identifying effective symbolic abstraction techniques is still a challenging problem. For complex real-world tasks, such structured abstraction is not easily available, making the incorporation of symbolic information difficult.

In my work, I explore three distinct techniques of symbolic abstraction to aid the learning progress of RL agents and improve their sample efficiency. Specifically, I investigate Object Oriented MDPs (OOMDPs); Temporal Logical Languages such as Linear Temporal Logic (LTL); and Large Language Models (LLMs). I use these three techniques to come up with a curriculum that can aid the learning progress of RL agents, improving their sample efficiency. Furthermore, I demonstrate how a curriculum designed in a low-fidelity environment transfers to a high-fidelity one. I conclude with a novel technique aimed at transferring policy knowledge learned in a simulator to a physical robot.

Acknowledgements

The idea that I would be pursuing a PhD, contributing to science, and writing my dissertation 7,605 miles away from where I grew up would have seemed like an impossible dream, if not a complete absurdity, to my 10-year-old self. Completing this dissertation would have been impossible without the unwavering support of the incredible people who have stood by me for longer than I can even begin to fathom. No words can truly capture the depth of my gratitude, but in the next few paragraphs, I will try my best.

Above all, to my advisor, Jivko Sinapov, who believed in me when I was a Master's student traveling 2 hours just to attend lab meetings, hoping for nothing more than a chance to get my foot in the door. He went above and beyond to give me a chance to pursue my PhD, and I have never looked back since. Throughout the years, he has never stopped challenging me to grow - to be a better researcher, a better learner, a better communicator, and a better person. For his support, I am forever grateful. Without him, I would not be here writing this dissertation.

I am thankful to the rest of my committee for their valuable feedback and gracious amounts of time, which helped shape this dissertation into its current form: Elaine Short, Reuth Mirsky, Chris Rogers, and Shiqi Zhang. Their diverse expertise has broadened my perspective and deepened my understanding across multiple areas of research.

I want to thank my collaborators and all other members of the MuLIP lab for their technical help: Shivam Goel, Gyan Tatiya, Christopher Thierauf, Ramtin Hosseini, Bharat Kesari, Tanushree Burman, Wenchang Gao, James Staley, Pierrick Lorang, Panagiotis Lymperopoulos, Andre Cleaver, Brennan Miller-Klugman, Ani Nair, Si Liu and Julia Santaniello. Our discussions and collaborations have given me a broader perspective

and enriched my research in countless ways.

A special thanks to Evana Gizzi, whose support and encouragement led her to reach out to Jivko on my behalf when I was a master’s student, ensuring that he became aware of my interest in pursuing a PhD. To my friend and collaborator, Abhishek Kulkarni, for sparking my initial interest in pursuing a PhD and later providing invaluable feedback on my work. I am deeply grateful to Bob Wright, Alvaro Velasquez and Vasanth Sarathy, who stepped in as mentors during critical stages of my PhD, offering profound guidance and wisdom that have been instrumental to my growth.

I extend my heartfelt gratitude to the faculty and staff at the Department of Computer Science at Tufts University. Special thanks to Sarah Richmond and Johnny Redmond for helping me navigate various walks of graduate life at Tufts. I also appreciate Sophia Alfred and the entire team at the Tufts International Center for their invaluable assistance with CPT, OPT, and all the other aspects of international life at Tufts.

Special thanks to Berk Calli, Carolina Ruiz, Jie Fu, and Michael Gennert for their invaluable guidance and insights during my time at WPI. I am also grateful for the opportunity to collaborate with incredible colleagues during my internships—Siddarth Jain and Diego Romeres at MERL, and Srinidhi Srinivas and Sachin Chitta at Autodesk. These experiences not only broadened my research interests but also exposed me to groundbreaking ideas that continue to inspire my work.

I am deeply honored and privileged to be surrounded by my wonderful family and friends. To my late Aaji and Ajoba (grandparents), Sarita and Vasant Shukla, who ignited my passion for science and instilled in me the power of dedication from a young age. To my late Ajoba, Vilas Kulkarni, and my Aaji, Vishakha Kulkarni, for their unwavering love and care. To my Aai and Baba (parents), Sanika and Sandeep, for empowering me to pursue my dreams and ensuring I had access to the best resources—your commitment has been the foundation of my journey.

To my cousin Sarvesh, sister-in-law Shefali, aunt Sunita, and nephew Ishan, for the laughter and joy they bring into my life. To my in-laws, Sonali and Pradeep Mahajan, and my sister-in-law Sneha, for their unwavering support and encouragement. When I first

arrived in the U.S., Shaku Aaji, Naren Kaka, and Sadanand Kaka helped me navigate the many uncertainties of a new country, making the transition smoother and less daunting. I am deeply grateful for their kindness, guidance, and support.

I feel extremely lucky to be surrounded by a wonderful group of friends - Manas, Vinit, Harshal, Siddhesh, Krishna, Omkar, and Ameya. Your constant humor, deep wisdom and timely group vacations have made this journey infinitely more enjoyable and far less overwhelming. To my friends from WPI - Anand, Ashvini, Shweta, Kranti, Vamshi, for turning Worcester into a place of joy and unforgettable memories. To Ruta, Vrushali, Aishwarya, Apurva, and Rhujuta, for your unwavering words of encouragement. And to Rohit, Aniket, Eshwar, Siddharth, and Harshal whose presence has made my academic journey over the past ten years feel like mere moments.

Finally, my deepest gratitude goes to my incredible wife, Shruti, who has been the very foundation of everything I have achieved and hope to achieve. Her unwavering confidence in me provides the strength I need every single day.

YASH SHUKLA

TUFTS UNIVERSITY

FEBRUARY 2025

Contents

Abstract	ii
1 Introduction	1
1.1 Background	1
1.1.1 Motivation	3
1.2 Dissertation Overview, Research Questions and Contributions	5
1.2.1 How Can Logical Specifications Improve Reinforcement Learning?	5
1.2.2 How Can Large Language Models Enhance Curriculum Generation?	6
1.2.3 Can Curricula Be Transferred Between Different Task Complexities?	8
1.2.4 How Can We Perform Few-Shot Simulation to Real World Policy Transfer?	9
1.3 Dissertation Outline	10
2 Related Work	13
2.1 Curriculum Design in Education	13
2.2 Curriculum Design for AI agents	15
2.3 Enhancing Sample-Efficiency of RL Agents	16
2.4 Knowledge Transfer in AI agents	17
2.5 Simulation to Real World Knowledge Transfer for Robotic Applications .	18
3 Utilizing Temporal Logical Specifications for Efficient Curriculum Gen- eration in RL	21
3.1 Introduction	22

3.2	Related Work	24
3.3	Theoretical Framework	25
3.3.1	Markov Decision Processes	25
3.3.2	Curriculum	26
3.3.3	Object Oriented MDP	27
3.3.4	Linear Temporal Logic and Deterministic Finite Automata (DFA)	28
3.3.5	Running Example	29
3.3.6	Problem Formulation	30
3.4	Methodology	30
3.5	Experimental Results	36
3.5.1	AGCL - Gridworld Results	36
3.5.2	AGCL on Tasks with Distractor Objects	39
3.5.3	Tasks with Imperfect OOMDP Descriptions	40
3.5.4	AGCL on Subset of Curriculum Candidates	40
3.5.5	Baselines Implementation	41
3.5.6	Tasks with Continuous OOMDP Ranges	45
3.6	Summary	46
4	Curriculum as a Dynamic Sub-Task Selection Strategy Using Temporal	
	Logic and LLMs	47
4.1	Introduction	48
4.2	Related Work	54
4.3	Theoretical Framework	55
4.3.1	Problem Formulation	57
4.4	Methodology	58
4.5	Convergence criteria for a trajectory to satisfy the high-level objective in LSTS	63
4.6	Experiments - LgTS	65
4.6.1	LgTS - DoorKey Domain	65
4.6.2	LgTS with distractor entities	68

4.6.3	LgTS - modified prompt	69
4.6.4	LgTS - Search and Rescue task	70
4.7	Experiments - LSTS	71
4.7.1	LSTS - Gridworld Domain	71
4.7.2	(LSTS + Cont. Training) - Gridworld Domain	74
4.7.3	LSTS and LSTS (CT) - Robotic Domains	76
4.7.4	LSTS - Search and Rescue task	78
4.7.5	Discussion: Improving LLM Sub-Goal Generation with Type Hierarchies	79
4.8	Summary	80

5 Curriculum Schema Transfer between Fidelities Using Object-Oriented

	MDPs	82
5.1	Introduction	83
5.2	Related Work	85
5.3	Theoretical Framework	87
5.3.1	Markov Decision Processes	87
5.3.2	Curriculum Learning (CL)	87
5.3.3	Object-Oriented MDP	87
5.3.4	Problem Formulation	89
5.4	Curriculum Transfer Approach	90
5.4.1	Low-Fidelity Environment Generation	91
5.4.2	Handcrafted and Automated Curriculum Transfer Generation . .	93
5.5	Experimental Results	96
5.5.1	Curriculum Generation in High Fidelity	98
5.5.2	Results with Added Complexity	102
5.5.3	Baseline Adaptation	102
5.5.4	Curriculum Generation with Different RL Algorithms	107
5.5.5	Curriculum Transfer Evaluation in Presence of Noisy Mappings .	108
5.5.6	Runtime Comparison	109

5.6	Summary	109
6	Simulation to Real-World Policy Transfer Using Observation Mapping and Behavior Cloning	112
6.1	Introduction	113
6.2	Related Work	115
6.3	Theoretical Framework	116
6.3.1	Markov Decision Processes	116
6.3.2	CycleGAN	116
6.3.3	Behavior Cloning	117
6.4	Problem Formulation	118
6.5	Few-shot Policy Transfer Approach	118
6.6	Experiments and Results	121
6.6.1	Sim2Sim Transfer with Observation Mismatch	121
6.6.2	Comparison with Different Numbers of Interactions	123
6.6.3	Semantically Different Source and Target Environments	124
6.6.4	Sim2Real Transfer on a Physical TurtleBot	125
6.7	Summary	127
7	Conclusion and Future Work	128
7.1	Summary of Contributions	129
7.1.1	Utilizing Temporal Logical Specifications for Efficient Curriculum Generation in RL	129
7.1.2	Curriculum as a Dynamic Sub-Task Selection Strategy Using Temporal Logic and LLMs	129
7.1.3	Curriculum Schema Transfer between Fidelities Using Object-Oriented MDPs	130
7.1.4	Sim2Real Using Observation Mapping and Behavior Cloning	130
7.2	Interconnections among Proposed Frameworks	130
7.3	Applications	132

7.3.1	Requirement of Robust Policies	132
7.3.2	Expensive Real-World Data Collection	132
7.3.3	Goal Objectives Known <i>apriori</i>	132
7.3.4	Expensive Automated Curriculum Generation	133
7.4	Boundaries	133
7.4.1	Availability of Symbolic Information	133
7.4.2	Dynamic Task Generation	133
7.4.3	Generation of Low-Fidelity Environments	134
7.4.4	Availability of Real-World Data Samples	134
7.4.5	Guarantees on Optimality	134
7.4.6	Limited Scope of Robotic Policies	134
7.5	Future Work	135
7.5.1	Automated Generation of Low-Fidelity Environments	135
7.5.2	Curriculum Transfer across Multiple Fidelities	136
7.5.3	Seamless Integration of Neural and Symbolic Information	136
7.5.4	Multi-modal Sim2Real Knowledge Transfer	136
7.5.5	Developing Simulators that Closely Replicate the Physical World	137
7.5.6	Human-in-the-Loop Automated Curriculum Generation	137
7.5.7	Different Types of Human Feedback for Improving Sample-Efficiency of RL Algorithms	138
7.5.8	Employing Curriculum Generation and Transfer for Real-World Robotic Tasks	139

List of Figures

1.1	Overview of the different techniques aimed at improving the sample-efficiency of RL agents	2
1.2	An overview of the curriculum schema transfer approach. The task in high-fidelity environment is first mapped to a low-fidelity environment. The curriculum is optimized in the low-fidelity and then transferred to the high-fidelity.	8
3.1	Overview of the automaton-guided curriculum generation procedure. Given the high-level task description, we use the equivalent automaton representation to come up with a sequence-based and a graph-based curriculum.	23
3.2	Examples of sequence-based (left) and graph-based (right) curricula. In a sequence-based curriculum, one source task contributes (knowledge) to one target task. In a graph-based curriculum, multiple source tasks contribute simultaneously to a target task.	26
3.3	Learning curves (Averaged over 10 trials): (a) Gridworld domain; (b) Gridworld - with distractor objects in the environment; (c) Gridworld - with imperfect OOMDP description of the environment; (d) Gridworld - with subset of curriculum candidates.	38
3.4	(a) and (c) show the two robotic domains - Crafter-Turtlebot and Panda-pick-and-place respectively. Learning curves (Averaged over 10 trials): are shown for (b) Crafter-Turtebot domain; and (d) Panda-pick-and-place domain.	44

4.1	A rendering of the Gridworld domain. The agent (shown in red triangle) needs to collect 1 <i>Key</i> , unlock the <i>Door</i> and reach the <i>Goal</i> to execute the task successfully.	49
4.2	The SPECTRL formula for the task and its DAG. The path highlighted in red denotes the more promising ways to reach the goal state among the two paths. Formulas l , k_1 , k_2 , d and g correspond to <i>Lava</i> , <i>Key</i> ₁ , <i>Key</i> ₂ , <i>Door</i> and <i>Goal</i> respectively	50
4.3	The prompt to the LLM that contains information about the number of paths n expected from the LLM and the symbolic information such as the entities, predicates and high-level initial and goal states. The LLM output is converted to a graphical representation	51
4.4	Learning Curves for each sub-task using the dynamic teacher-student sampling strategy. The sub-task sampling promotes learning promising tasks and discards unpromising tasks.	51
4.5	Learning curves (Averaged over 10 trials) for the two robotic domains.	77
5.1	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.	84
5.2	Illustration of the final target task in LF, HF, and physical environments. 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.	88
5.3	Illustrative example of the low-fidelity to high-fidelity mapping in the crafting task. Each of the class parameters in the LF domain are mapped to generate corresponding parameter in the HF domain.	90

5.4	Learning Curves for low fidelity (LF) and high fidelity (HF) environments with and without Fire. A common trend observed from the learning curves is the improved performance of ACuTE (red) compared to the baseline appraoches.	99
5.5	An example of the Automated Curriculum procedure in the low-fidelity environments. The agent learns complex tasks incrementally, before learning the final target task.	101
5.6	Learning HF target task through different RL algorithms.	107
5.7	Comparison of Learning Curves for high fidelity (HF) environments. Here, the HF environments are generated by assuming an imperfect (noisy) LF to HF mapping.	108
5.8	Runtime comparison of different approaches. Since our curriculum transfer approach generates the curriculum in Low-Fidelity environment, the curriculum generation is faster as compared to baselines that generate curriculum purely in the High-Fidelity domains.	110
6.1	Overview of the Few-Shot Policy Transfer approach. During training, the agent learns a mapping between the observations of the source and target domain and clones the successful source task behavior policy. During the evaluation, the agent tests the cloned policy on the target domain. . . .	115
6.2	Examples of source and target domain observations. The observations are images retreived from an ego-centric camera attached to the robotic agent.	121
6.3	Success rate (Mean $\pm$ SD) comparison of the three TL approaches for different target dataset sizes. More target datapoints results in a better mapping between the observations of the two domains, resulting in a better knowledge transfer.	124
6.4	Examples of source and target domain observations.	124
6.5	Real-world domain (left) and egocentric observations from the robot (right). The robotic agent needs to navigate to the grey box placed randomly in the environment.	127

List of Tables

4.1	Table comparing #interactions & success rate for the DoorKey domain. Our proposed approach, LgTS (highlighted in grey) outperforms baselines and is comparable to the oracle approach- AgTS. Other approaches are not able to learn a set of successful policies because the task is either too complex or the approach falls into a local optima, preventing the agent from learning policies to reach the goal state.	67
4.2	Table comparing #interactions & success rate for the DoorKey domain with distractor objects. Our proposed approach, LgTS (highlighted in grey) outperforms baselines and is comparable to the oracle approach- AgTS. Other approaches are not able to learn a set of successful policies because the task is either too complex or the approach falls into a local optima, preventing the agent from learning policies to reach the goal state.	69
4.3	Table comparing #interactions & success rate for the DoorKey domain with modified prompt. Our proposed approach, LgTS (highlighted in grey) outperforms baselines and is comparable to the oracle approach- AgTS.	70
4.4	Table comparing #interactions & success rate for the Search and Rescue domain. Our proposed approach, LgTS (highlighted in grey) outperforms baselines and is comparable to the oracle approach- AgTS. Other approaches are not able to learn a set of successful policies because the task is either too complex or the approach falls into a local optima, preventing the agent from learning policies to reach the goal state.	71

4.5	Table comparing #interactions & success rate. <i>LSTS</i> (highlighted) outperformed all baselines	72
4.6	Table comparing #interactions & success rate for the Search and Rescue domain. Our proposed approach, LSTS outperforms the baselines.	78
4.7	Parameters used for training the Proximal Policy Optimization (PPO).	79
5.1	Parameters used for training the Policy Gradient on the High-Fidelity task	101
5.2	Table comparing <i>jumpstart</i> (mean $\pm$ SD), and <i>time to threshold</i> 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.	102
5.3	Parameters used for training the Proximal policy optimization on the High-Fidelity task	106
5.4	Parameters used for training the Deep Q-Learning on the High-Fidelity task	106
6.1	Success rate comparison of different Source-to-Target Transfer Approaches. For the GAN+Behavior Cloning, and for CycleGAN+Behavior Cloning, 4000 source and target task images were used for generating the mapping.	122
6.2	Success rate comparison of different Source-to-Target Transfer Approaches. For the GAN+Behavior Cloning, and for CycleGAN+Behavior Cloning, 4000 source and target task images were used for generating the mapping.	125

Chapter 1

Introduction

1.1 Background

Reinforcement Learning (RL) offers a strong framework for complex sequential decision-making problems. One of the chief advantage of RL is its ability to learn robust behavior policies without the requirement of hardcoded task-specific rules [82]. These advantages makes RL suitable for problems in games, robotics and autonomous systems, where environments can be dynamic and unpredictable. The behavior policies learned by RL agents can evolve and improve over time with more experience, making them robust to tasks with inherent non-stationarity or changing goal objectives [56].

A key challenge in RL is generalizing policies across different tasks, environments, and domains. Techniques like transfer learning [128] and meta-learning [31] help RL agents to leverage their previous experience, significantly reducing the time taken to learn a successful policy for a new task. Recent advances in deep reinforcement learning (DRL) have enabled agents to process high-dimensional inputs like images and videos, achieving human-level performance in visually rich environments [82]. Such use cases underscore RL as a tool that can be used for developing intelligent systems that can learn autonomously and adapt to a wide variety of real-world challenges.

RL has enabled agents to learn complex behavior policies in complex environments. However, many real-world sequential decision-making tasks remain prohibitively expen-

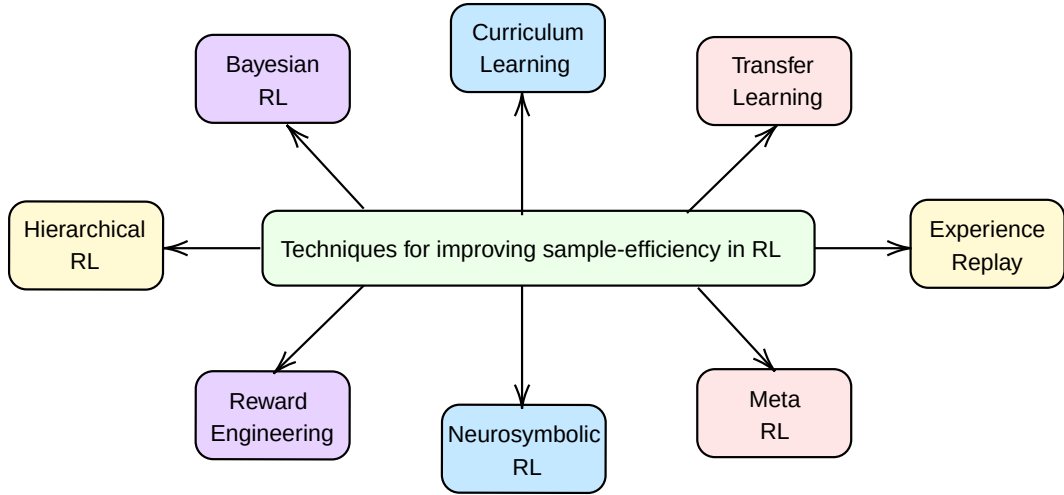


Figure 1.1: Overview of the different techniques aimed at improving the sample-efficiency of RL agents

sive to learn due to partially observable environments and sparse reward settings [85].

To address sample inefficiency issues, several lines of work have investigated techniques that help RL agents learn efficiently with fewer environmental interactions. One such approach is to use experience replay, where previous agent experiences (in the form of trajectories) are stored and reused to improve the sample efficiency of the agent [82]. Another common technique is dense reward shaping that helps guide the agent toward the goal quicker by supplementing the reward through predetermined heuristics [86]. However, reward shaping techniques assume that the human engineer is aware of which experiences are more informative than others. Incorrectly shaped rewards can drive the agent into learning partially useful behaviors [63]. An overview of the different techniques aimed at improving the sample-efficiency of RL agents is shown in Fig. 1.1.

Transfer learning and Meta-Learning are two techniques that help the agent learn a task efficiently. Transfer learning allows the agent to reuse knowledge from previously learned tasks to speed up learning in related tasks [128]. Similarly, meta-learning techniques (e.g., Model-Agnostic Meta-Learning (MAML) [31]) train the agent on a distribution of tasks that helps transferring policies to newer unseen tasks within that distribution. Transfer and meta-learning are most effective when tasks are related through similar structure or dynamics. As the tasks become more and more unrelated, the benefits of transfer and meta learning diminish, leading to a negative transfer.

Curriculum learning is another technique that aims to generate a progression of tasks of increasing difficulty, helping the agent to learn simpler tasks earlier and transferring that knowledge to complex tasks [8]. Curriculum in RL improves the sample efficiency of agent, making this technique suitable for complex tasks with high-dimensional state representations and long horizon goal objectives.

Curriculum learning involves three components: (1) task generation, (2) task sequencing, and (3) transfer learning. Task generation creates tasks of varying difficulty, task sequencing arranges these generated tasks in increasing order of complexity. The third component, transfer learning, helps the agent apply knowledge gained from one task to others thereby improving the overall efficiency [85].

A number of curriculum learning techniques exist in literature. In one such technique a human expert handcrafts the curriculum by deciding the sequence in which the RL agent attempts different tasks and gain experience [32]. In another approach, instead of relying on a human expert to decide the task order, the sequence is generated by an automated procedure that dynamically adjusts the task difficulty based on the agent’s performance [97]. Self-Play [125] is one technique that generates automated curricula by having agents compete against increasingly competent versions of themselves. Automatic Domain Randomization methods [133] automatically change the environmental parameters by monitoring the agent’s performance, which ultimately leads to better generalization on unseen tasks.

1.1.1 Motivation

Symbolic information obtained from Linear Temporal Logic (LTL) specifications or Large Language Models (LLMs) can help improving the sample efficiency of RL agents. Recent approaches aimed at incorporating symbolic information introduce reward machines [49, 67] as a technique to enhance the sample efficiency of RL agents. In approaches that use reward machines, a human engineer decides the relative importance as well as the order of sub-tasks, providing a dense reward shaping mechanism for the RL agent. For example, for an RL agent tasked with learning how to make a sandwich, a reward machine

would capture relative order of sub-tasks, such as slicing bread before toasting it. Such symbolic representations provide structured, high-level descriptions of task objectives and constraints which can enable agents to reason about their state and learn policies quickly without solely relying on environmental reward [14]. Symbolic information can provide information regarding the structure of the sub-goals as a finite state machine and help reduce the complexity of exploration for a complex long horizon task [122]. LLMs can also reason about the state of the agent in natural language. This reasoning helps bridge the gap between human specified instructions and RL policies that rely on a lower representation of the environment. This integration enables RL agents to better handle abstract reasoning in environments where reward signals are sparse or delayed.

Generating a curriculum in a simulator is less complex as compared to a real world scenario as the environment is easily configurable and interactions in a simulator are inexpensive. Generating automated curricula in a physical scenario is challenging as the environments are not easily configurable and the cost of resetting the environment is quite high. This dissertation takes steps that make it easier to generate curriculum in a low fidelity environment and transfer it to a high fidelity environment. We also propose a method for transferring knowledge from simulation to the real world that helps in learning policies in the real world efficiently.

In this dissertation, I introduce three novel curriculum learning techniques that leverages symbolic information derived from Linear Temporal Logic (LTL) specifications and Large Language Models (LLMs) to design adaptive task sequences for reinforcement learning agents. High level symbolic information provides meaningful sub-goal structure that guides the learning progress of the agent. Next, we propose a curriculum schema transfer technique, where a RL agent maps the schema of the curriculum from a low-fidelity environment to a high-fidelity environment. Finally, I present a method for transferring behavior policy from a simulation to a real-world (Sim2Real) environment without relying on hardcoded domain features. These three techniques together allow RL agents to generate automated curriculum in low fidelity simulators and transfer the schema of the curriculum along with behavior policy to a high fidelity setting.

1.2 Dissertation Overview, Research Questions and Contributions

In this dissertation, I emphasize on three aspects in training policies for high fidelity domains: (1) Curriculum generation, (2) curriculum schema transfer between different fidelities of environments and (3) Simulation to real world policy transfer. The primary hypothesis is that generating a curriculum in a low-fidelity setting is easier and quicker, and transferring this curriculum to a challenging high-fidelity environment along with policy transfer will aid faster policy adaption in the real world setting.

These techniques enable the agent in the high-fidelity setting to learn the task through a curriculum that was developed in a low-fidelity setting. Along with curriculum schema transfer, knowledge transfer in the form of reinforcement learning policies enables quick RL policy learning. The central research question addressed in this dissertation is:

How can symbolic information obtained from temporal logic, large language models, and human generated task guidance help in designing automated curricula in a low fidelity setting and how can the curricula and policy information from the low fidelity be transferred to a high fidelity setting to improve the sample efficiency of RL agents?

To address this question in depth, we decompose it into four supporting research questions where each contributes to the formulation of theoretical and practical concepts and provides valuable insights into the central research inquiry.

1.2.1 How Can Logical Specifications Improve Reinforcement Learning?

Temporal logic specifications have gained traction in reinforcement learning (RL) settings for specifying high-level task objectives in long horizon tasks where traditional reward shaping mechanisms fall short. Temporal logics (e.g., Linear Temporal Logic (LTL))

formulas provide us with a formal framework to express complex task specifications that enables the agent to reason about the sequence of sub-goals required to achieve a goal. By incorporating such logical constraints into the learning process of the RL agent, we can efficiently navigate sparse reward settings and improve the overall performance in sequential decision-making tasks [22].

Our first automated curriculum generation procedure, Automaton-guided Curriculum Learning (AGCL) [116] is a novel approach that employs LTL_f specifications (by using their equivalent Deterministic Finite Automata (DFA) representations) to automatically generate curricula for long horizon tasks. AGCL decomposes the high-level task objective into a sequence of sub-goals which are then used to design a sequence-based and a graph-based curriculum. The structure of the curriculum helps the agent to learn simpler tasks earlier and transferring knowledge to complex tasks. The graphical structure of the curriculum in AGCL improves sample efficiency of RL agents by promoting knowledge transfer to a target task in curriculum through multiple source tasks simultaneously [85].

We evaluate our approach through a series of experiments in gridworld and physics-based robotics environments, comparing it to state of the art baseline in curriculum learning and automaton-guided RL approaches. AGCL reduces the number of interactions with the target environment by orders of magnitude improving the learning progress of the agent. The curriculum developed by AGCL is robust in noisy environments or when distractor objects are present that are not modeled in the logical specification of the tasks. The results demonstrate that AGCL provides a practical solution for curriculum generation, particularly in long-horizon tasks, by efficiently combining symbolic information with RL to generate curricula that improve sample efficiency and performance in real-world scenarios.

1.2.2 How Can Large Language Models Enhance Curriculum Generation?

Along with the use of temporal logic for RL, recent advances in Large Language Models (LLMs) have increased the use of such models to design efficient AI agents. LLMs are

trained on a huge corpora of natural language, making them particularly useful for tasks requiring common sense human knowledge. Recent techniques using LLMs for RL tasks rely on hardcoded reward functions that guide the learning progress of the agent through a series of sub-goals obtained from LLMs. In our work, we intend to either query a Large Language Model to produce feasible sub-goals or have a human engineer specified temporal logic formula for generating sub-goal structure that guides the agent without relying on dense reward functions.

This dissertation proposes Logical Specification-guided Dynamic Task Sampling for RL Agents (LSTS) [117], a novel method that uses the sub-goal structure obtained from formal logical specifications like SPECTRL [55] formulas to guide the learning progress of RL agents by focusing on sub-tasks that show promise. By composing the sub-goal structure as a directed acyclic graph, LSTS promotes the RL agents to focus on tasks that show promise thereby reducing interactions on unpromising tasks. The teacher-student curriculum learning framework in LSTS assists the student agent by learning on sub-tasks that show promise, as determined by the teacher agent. This approach outperforms other RL methods such as RM-guided Q-learning and automaton-based RL by reducing the number of required interactions by one order of magnitude for minigrid [16] tasks.

Another approach, namely, LLM-guided Dynamic Task Sampling for RL Agents (LgTS) further enhances the sample efficiency and reduces the need for human-generated specifications by using LLMs to generate sub-goal structures. By querying an LLM with information about the environment, LgTS enables RL agents to generate and learn from dynamically crafted sub-goals, accelerating policy learning. Experimental results show that LgTS achieves faster convergence compared to curriculum learning approaches and performs comparably to oracle-guided sub-goals, demonstrating its applicability to complex, long-horizon tasks. Both LSTS and LgTS highlight the potential of integrating logical specifications and LLM-guided planning to enhance RL agents’ performance in environments with sparse rewards and unknown dynamics.

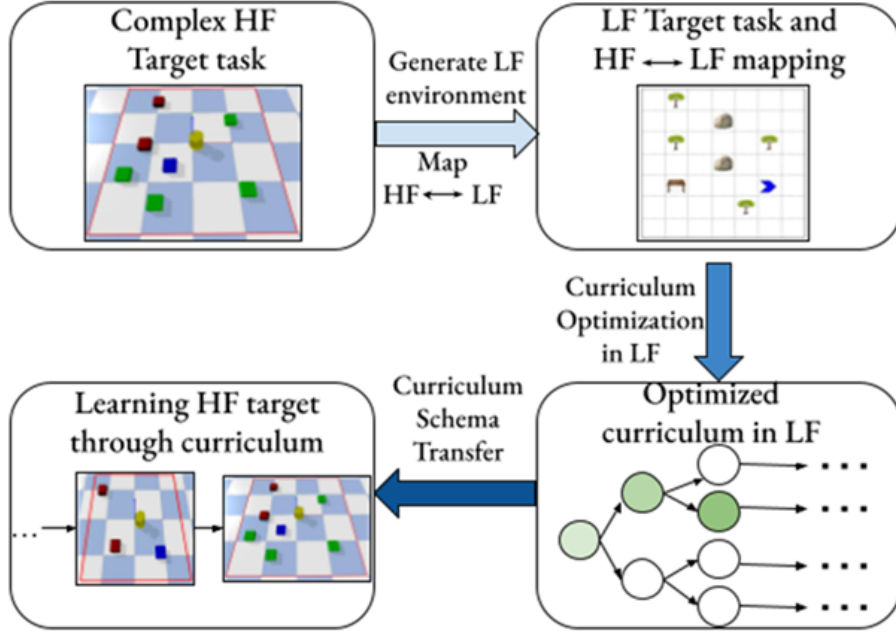


Figure 1.2: An overview of the curriculum schema transfer approach. The task in high-fidelity environment is first mapped to a low-fidelity environment. The curriculum is optimized in the low-fidelity and then transferred to the high-fidelity.

1.2.3 Can Curricula Be Transferred Between Different Task Complexities?

Curriculum Learning (CL) aims to optimize the learning process by sequencing simple tasks to help an agent gradually tackle more complex ones. However, applying CL to complex high-fidelity (HF) environments is a challenging endeavor due to complex system dynamics and the cost of environmental interactions. Traditional approaches for transferring knowledge from simplified low-fidelity (LF) environments to HF settings (e.g., domain adaptation, system identification) require extensive interactions and costly setup. Additionally, no known method exists that can transfer the schema of a curriculum between different fidelity of environments.

We propose a solution called Curriculum Schema Transfer 1.2, which uses Object-Oriented Markov Decision Processes (OOMDPs) to represent the tasks in the LF and HF environments. This method focuses on transferring only the schema of the curriculum through parameters for each task in the curriculum. This work does not transfer policy knowledge as it does not generalize across fidelities. By modeling tasks with OOMDPs, we can abstract and transfer task structures from simpler LF environments to more

complex HF environments. This approach allows the curriculum to be optimized in the LF environment, where computational costs are lower. As a next step, the schema of the curriculum is transferred to the HF environment to guide the agent’s learning process efficiently.

In this dissertation, we introduce ACuTE (Automatic Curriculum Transfer from Simple to Complex Environments), a novel method for efficient curriculum schema transfer from a LF environment to a HF environment. We demonstrate that curriculum schema transfer improves the sample efficiency of the RL agent in HF tasks. By testing our method on both simulated environments and physical robots (via Sim2Real transfer), we show that our approach leads to quicker convergence and higher jumpstart performance compared to other curriculum learning approaches. ACuTE is also independent of the RL algorithm and can handle imperfect mappings between LF and HF environments, making it a versatile solution for a wide range of RL applications.

1.2.4 How Can We Perform Few-Shot Simulation to Real World Policy Transfer?

Sim2Real transfer aims to transfer knowledge from a simulated environment to a physical one in order to improve the performance of the agent in the real environment. Even though simulation offers an easier and cheaper way to train agents, the differences between the simulated and real environments can make the knowledge transfer inefficient. The dynamics, observations, and actions in the simulation often do not match the complexities of the real world, making the policy learned in the simulation perform poorly in the target real-world task. Traditional Sim2Real methods often rely on domain adaptation techniques such as domain randomization, which tries to bridge the gap by modifying simulation parameters, but these approaches still face issues when real-world conditions are not accurately represented in the simulation.

The key challenge in Sim2Real transfer is to map the observations from the source (simulated) domain to the target (real) domain. In certain cases, the observations can be semantically different, making the mapping difficult. In this dissertation, we propose

a framework for Few-Shot Policy Transfer, which aims to transfer policies with minimal interactions in the target domain. The framework uses a set of CycleGANs to generate a mapping function that can efficiently map images between the simulation and the real domain. The cycle-consistency loss in CycleGAN helps ensure that the transformation from source to target observations is accurate, thereby enabling the policy learned in the source domain to be successfully adapted to the target domain using behavior cloning.

Our framework is effective in cases where there is limited data available from the target domain and even when the transition dynamics between the domains are different. We demonstrate the approach in situations where the action semantics in the target domain differ from the source domain. Through experiments, we show a successful Sim2Real transfer, where a policy trained in simulation performs well in the real-world target domain. Our experiments involve transferring a policy for a simulated turtlebot to reach a goal in simulated environment to a physical robot reaching a goal in a real environment. Our method achieves a high success rate of 90% for a goal-reaching task on a physical robot making a case for positive Sim2Real transfer, and reducing the reliance on policy learning in complex and expensive real world environments.

1.3 Dissertation Outline

The dissertation is structured as follows:

Chapter 2: Related Work

We begin with a study of curriculum design in education and cognitive science to understand how humans comprehend new concepts. Next, we explore the different techniques used for AI curriculum design and knowledge transfer. Finally, we survey techniques aimed at Sim2Real transfer for robotic agents that work on transfer of simulation-based representations to physical robotic systems.

Chapter 3: Utilizing Temporal Logical Specifications for Efficient Curriculum Generation in RL

In this chapter, we introduce Automaton-guided Curriculum Learning (AGCL), which uses LTL_f specifications and DFAs to automatically generate curricula for RL agents. Our approach improves sample efficiency by efficient knowledge transfer and parallel task learning through the graphical representation of sub-goals. Through extensive experiments in gridworld and robotics environments, we show that AGCL requires 6×10^6 fewer interactions than existing methods on a minecraft-like task in a gridworld setting, demonstrating improved sample efficiency.

Chapter 4: Curriculum as a Dynamic Sub-Task Selection Strategy Using Temporal Logic and LLMs

In this chapter, we introduce two approaches: LSTS, which uses formal logical specifications to guide RL agents through sub-goals via a Teacher-Student curriculum learning method, and LgTS, which uses LLMs to automatically generate these sub-goal structures. Our proposed methods do not rely on a dense reward function to guide the learning progress of the agent. Experimental results demonstrate that both approaches achieve faster convergence than traditional methods.

Chapter 5: Curriculum Schema Transfer between Fidelities Using Object-Oriented MDPs

In this chapter, 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). Our approach, “ACuTE” is a novel framework aimed to solve this problem. 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, and that our approach is also independent of the learning algorithm used for curriculum generation.

Chapter 6: Sim2Real Using Observation Mapping and Behavior Cloning

This chapter introduces a framework for Few-Shot Policy Transfer between simulated and real domains through Observation Mapping and Behavior Cloning. We use Generative Adversarial Networks (GANs) along with a cycle-consistency loss to map the observations between the source and target domains and later use this learned mapping to clone the successful source task behavior policy to the target domain. We observe successful behavior policy transfer with limited target task interactions and in cases where the source and target task are semantically dissimilar.

Chapter 7: Conclusion and Future Directions

Finally, I conclude with a synthesis of my main contributions and I outline promising directions for extending this research.

Chapter 2

Related Work

In this chapter, we delve into the body of literature that serves as a foundation for this dissertation. We first begin by surveying the field of curriculum design in education and cognitive science, which provides us with an understanding of how humans structure learning newer concepts. We then explore the curriculum design of AI agents that gives us an idea of how AI agents store information and use it to learn increasingly complex concepts. Next, we go over knowledge transfer techniques in AI agents, going over how AI agents transfer knowledge learned in one environment to another. Finally, we survey simulation to a physical robot knowledge transfer (Sim2Real) techniques - that aim to learn complex representations on physical robots using techniques learned in pure simulations.

2.1 Curriculum Design in Education

Curriculum design has been a central area of interest in education theory, providing structured frameworks that organize learning experiences in a way that best enhances systematic knowledge acquisition. In *Basic Principles of Curriculum and Instruction* [134], the author established a foundational model that emphasized learning and knowledge acquisition through clearly defined objectives, systematic content organization, and timely assessment strategies that evaluate the learning progress. Recent adaptations have focused on developing instructional methods that foster learning through predefined goal

objectives [87]. Such models encourage consistency across different educational systems and promote a structured knowledge acquisition from foundational concepts to intermediate and advanced concepts.

A parallel set of approaches promote constructivist ideas that have significantly influenced modern curriculum design by incorporating elements from learner-centered approaches. Specifically, active learning [95] is a tool where learners actively construct newer concepts by interacting with the environment. This approach has enhanced the application of different scaffolding techniques, where educators provide a large amount of support initially and gradually decreases as the learner becomes adept at solving the task at hand [139]. Recent research emphasizes the significance of adaptable learning environments that respond and change with the learners' skill in terms of knowledge and experience [11].

Another branch of literature supports competency-based education (CBE) as a framework for curriculum design. CBE as a tool, prioritizes mastery of specific skills instead of focusing on time-based learning goals. This methodology allows the learners to learn and progress at the pace they enjoy, and only advancing when they show competency in the required skill [38]. Several recent user studies have demonstrated that CBE enhances motivation and learning outcomes by allowing learner autonomy and personal preferences [66]. In such models, the competent skills are closely related to the an equivalent real world task, ensuring efficient knowledge transfer to real world scenarios [34].

Rapid advancements in the digital education space have transformed curriculum design by incorporating technologically enhanced learning platforms. These developments have increased the scope of flexible and personalized instruction [64]. Blended learning models, which combine a synchronous online instruction session with an asynchronous digital resource, are a good indicator of increasing engagement, accessibility and ease of learning [42]. Massive Open Online Courses (MOOCs), such as EdX, Coursera etc. now allow for modularized concepts that can be tailored to the students need, prior experience and goal objectives [78]. Digital learning sources have not only advanced access to education but also enabled practice of data-driven methods that can optimize the content,

promoting use of continuous evaluation for learning outcomes [119].

2.2 Curriculum Design for AI agents

Based on the promise of curriculum in human education systems, curriculum design for AI agents has become a key area of research. The fundamental work in Curriculum Learning (CL) proposed how machine learning models benefit (show positive transfer) when trained on easier tasks before progressing to challenging ones. This concept was later introduced for Reinforcement Learning, where the key premise was having agents learn simpler tasks initially, transferring knowledge to complex tasks as the curriculum progresses. Initial applications were centered in games and robotics, with more recent applications in large language models [57, 37].

In reinforcement learning, curriculum design has three components: task generation, task sequencing, and knowledge transfer [85]. Task generation focuses on creating a set of progressively difficult tasks culminating at the final target task [120, 61]. Recent work has investigated the different ways of generating source tasks by factorizing individual components of the target task [80]. The second component, task sequencing, involves ordering the generated tasks so that the learning progress is sample efficient [84, 125]. In environments where the optimal task sequencing is unknown, metaheuristic optimization methods are commonly applied [33]. The third component, knowledge transfer, deals with the approaches concerned with transferring valuable information (such as policy or value functions) between tasks of the curriculum to achieve target learning objectives [128].

More recently, human-generated as well as autonomous graph based curriculum generation approaches are being investigated for RL applications. In these methods, the curriculum is represented as a graph, with the nodes being the sub-goals and the edges denoting the transitions [126]. These methods assume prior knowledge about the state space of the environment, making them susceptible to errors introduced by sub-optimal engineering. Another promising direction uses Automaton guided RL. Automaton guided methods use high-level specifications (e.g., linear temporal logic (LTL) or reward ma-

chines) to define goals as a series of sub-tasks [132, 135]. This formulation allows agents to tie their low-level controller policies to a higher level symbolic reasoning agent, helping them make consistent progress toward the goal objective.

Recently, large language models (LLMs) are used for curriculum generation. LLMs are being used for task decomposition, which helps in generating smaller sub-goals given the complex objective. These sub-goals can later be used as part of the curricula for RL agents [68]. Since LLMs are trained on a huge corpora of human generated natural language, LLM based agents are capable of producing sub-goals that emulate human generated sub-goals. This process reduces the need to rely on a human to produce a curriculum, making the process efficient [62]. One of the key challenges of using LLM based approaches is the use of premeditated tools like chain-of-thought prompting or symbolic planners to ensure that the sub-goals generated by the LLMs are actually feasible [122].

Despite these advancements, many curriculum learning generation approaches either require extensive human involvement, or fail to generalize across different domains. Adaptive teacher-student learning frameworks serve as a promising direction to dynamically adjust the curriculum based on the agent’s performance [76]. Teacher-Student learning approaches along with modern advancements in neural network research enable us to solve complex tasks without relying on extensive hardcoding from a human engineer.

In our dissertation, we incorporate symbolic information either through logical specifications or LLMs to generate automated curricula. Unlike existing methods that rely heavily on hand-crafted curriculum generation, our work focuses on automated curriculum generation that can dynamically adapt to the learning progress of the agent.

2.3 Enhancing Sample-Efficiency of RL Agents

Along with curriculum learning, several other techniques are proposed that aim to improve the sample efficiency of RL agents. One common approach is experience replay, where past experiences are stored and reused during training to improve sample efficiency [82]. In this technique, agents reduce the need to gather new experiences by learn-

ing from previously recorded experiences. Techniques like prioritized experience replay [108] enable the agent to focus on learning from experiences that are more informative. Another common method is reward shaping, where the reward function is augmented with additional information that helps guide the agent toward desirable behaviors more quickly [86].

Transfer learning and meta-learning are two techniques aimed at improving sample efficiency of RL agents. Transfer learning allows agents to reuse knowledge from previously learned tasks to speed up learning in new, related tasks [128]. Similarly, meta-learning (e.g., Model-Agnostic Meta-Learning (MAML)) [31] focuses on training agents on tasks from a distribution so that they can quickly adapt to new tasks that belong in the same distribution with minimal additional training. Curriculum learning strategies [8, 85] structure the learning process by gradually increasing task difficulty, enabling agents to build on simpler skills before attempting complex tasks. Transfer learning, meta learning and curriculum learning are all aimed at enhancing the sample efficiency of RL algorithms. This attribute makes them suitable for real world tasks where sample efficiency is a key challenge.

2.4 Knowledge Transfer in AI agents

Knowledge transfer [128, 127] in AI agents promises quicker and efficient learning and also promotes reusing learned information across different tasks and domains. Transfer learning is one of the central components of knowledge transfer that aims to develop techniques designed to facilitate positive transfer across different environments and domains. Initial work in transfer learning is focused on fine-tuning general models that are pre-trained on large datasets such as ImageNet [23] to a smaller specific task [90]. This technique allows agents to learn general skills from the large dataset and apply it to specific tasks.

Another commonly used technique in knowledge transfer is domain adaptation [7], where an agent learns to apply knowledge learned on data coming from one distribution

to another task where data belongs to another distribution. Adversarial training [36] has been successful in aligning feature distributions across different domains. Knowledge transfer through domain adaptation reduces the need for large annotated task-specific datasets.

Knowledge distillation is another approach in knowledge transfer that deals with transferring knowledge from a complex, high-performing teacher agent to a simpler student agent [41]. Recent approaches have investigated knowledge distillation from several teacher agents, each with a specialized skill to a single student agent for efficient decision making [110].

In tasks where symbolic knowledge is readily available, knowledge transfer through symbolic information presents as a promising approach. Works in this area [21] have combined reinforcement learning with structured symbolic reasoning. The complementary interplay of symbolic reasoning with RL exploration enables agents to learn abstract concepts independently and allows for efficient transfer of these concepts across different domains.

While these methods have proven successful in various contexts, challenges remain in ensuring efficient knowledge transfer between heterogeneous environments and tasks. Our approaches leverage symbolic information to learn a mapping between environments of different fidelities, promoting cross environment knowledge transfer.

2.5 Simulation to Real World Knowledge Transfer for Robotic Applications

Simulation-to-reality knowledge transfer is one of the primary area of research for robotic applications. Learning skills in a simulated environment and transferring information to a real world allows for training policies in an inexpensive setting, reducing reliance on expensive real world data collection. However, the difference between the dynamics of the simulated environment and the real world (reality gap) presents several challenges for effective adaptation of simulated policies in real world applications.

One of the earliest examples of bridging the Sim-to-Real gap introduced Domain Randomization [129], a technique where a variety of simulation parameters (e.g., object textures, lighting conditions, camera noise) are randomized within a distribution during training. The idea behind domain randomization is to train a model over a distribution of these randomized parameters so that during evaluation, the real world task also lies in the distribution of parameters. This ultimately reduces the reality gap. However, domain randomization typically suffers from the combinatorial explosion problem during training, as it requires randomizing several parameters that are essential to training a robust policy.

Domain Adaptation, while frequently used for knowledge transfer, is also used for sim-to-real transfer. Recent work has explored aligning simulated visual representation to real world images using unsupervised learning [92]. Use of CycleGAN[144, 102] allows domain adaptation without the need of paired labeled images between simulated and real environments, making the process sample efficient [88].

Successful real world robotic navigation has been achieved through RL applications. Deep Q learning trained on a dataset that contains both simulated and real world datapoints achieves better generalization [106]. However, one of the key challenges in such approaches is data generation in real world environments, where collecting datapoints representative of the entire distribution is a challenge [77].

Behavior cloning [131] is a promising avenue for learning successful policies in the real world. Obtaining successful demonstrations from expert users and cloning them to learn an effective policy has shown more promise than pure reinforcement learning [102]. Behavior cloning has been successfully used for robotic driving [17], where expert demonstrations are relatively easier to collect and mimic using deep neural networks.

Another line of research has focused on enhancing features of physical simulators such as MuJoCo [130] and NVIDIA’s Isaac Sim [81] so that the dynamics of the simulated environments can closely match that of the real world. These simulators aim to reduce the sim-to-real gap by accurately modeling dynamics, friction, and sensor noise. Such high-fidelity simulators are getting increasingly commonplace for training RL policies for

robots.

Even with recent advancements, the gap between simulation and real environments is still a relevant challenge because of the unpredictable nature of the real world [25]. Deformable objects, fluids are harder to simulate and make the simulations computationally expensive. Therefore, Sim2Real fine-tuning methods [28] are also widely adopted, where policies trained in simulation are fine-tuned using small amounts of real-world data.

To summarize, efforts directed to closing the simulation to reality gap have evolved using various branched strategies, such as domain randomization, imitation learning, and high-fidelity simulators. Even with recent progress in Sim2Real transfer, consolidated efforts are needed to reduce the reliance on real world data samples.

Chapter 3

Utilizing Temporal Logical Specifications for Efficient Curriculum Generation in RL

Despite advances in Reinforcement Learning, many sequential decision making tasks remain prohibitively expensive and impractical to learn. Recently, approaches that automatically generate reward functions from logical task specifications have been proposed to mitigate this issue; however, they scale poorly on long-horizon tasks (i.e., tasks where the agent needs to perform a series of correct actions to reach the goal state, considering future transitions while choosing an action). Employing a curriculum (a sequence of increasingly complex tasks) further improves the learning speed of the agent by sequencing intermediate tasks suited to the learning capacity of the agent. However, generating curricula from the logical specification still remains an unsolved problem. To this end, we propose *AGCL*, Automaton-guided Curriculum Learning, a novel method for automatically generating curricula for the target task in the form of Directed Acyclic Graphs (DAGs). *AGCL* encodes the specification in the form of a deterministic finite automaton (DFA), and then uses the DFA along with the Object-Oriented MDP (OOMDP) representation to generate a curriculum as a DAG, where the vertices correspond to tasks, and edges correspond to the direction of knowledge transfer. Experiments in gridworld and

physics-based simulated robotics domains show that the curricula produced by *AGCL* achieve improved time-to-threshold performance (6×10^6 environmental interactions on a minecraft-like task in a gridworld setting) on a complex sequential decision-making problem relative to state-of-the-art curriculum learning (e.g., teacher-student, self-play) and automaton-guided reinforcement learning baselines (e.g, Q-Learning for Reward Machines). Further, we demonstrate that *AGCL* performs well even in the presence of noise in the task’s OOMDP description, and also when distractor objects are present that are not modeled in the logical specification of the tasks’ objectives ¹.

3.1 Introduction

Deep reinforcement learning utilizes neural networks to solve complex tasks ranging from Atari games to robot manipulation [37, 57]. Despite these advances, many tasks remain prohibitively expensive to learn from scratch, requiring large number of interactions. This problem worsens in long-horizon tasks where sparse reward settings and poor goal representations make the task difficult to solve. In many practical scenarios, the objective for a task is known before commencing learning the task, and thus it can be captured using high-level specifications that have an equivalent automaton representation [135, 55]. This formulation allows decomposing the task into sub-goals, where each sub-goal can be learned by a Reinforcement Learning (RL) agent. This approach of automaton-guided RL is beneficial in scenarios where the high-level task objective is known beforehand and can be expressed using an automaton, but the low-level transition dynamics are unavailable and hence motion and symbolic planners cannot be reliably used. Recently, many approaches have made use of richer language representations (such as finite-trace Linear Temporal Logic (LTL_f) [22]) to incorporate history in the task Markovian Decision Processes (MDPs). This technique helps the RL agent to shape reward effectively to achieve the goal in sparse reward settings, eliminating the need to rely on human guidance

¹This chapter is based on the following paper: Shukla, Yash, Abhishek Kulkarni, Robert Wright, Alvaro Velasquez, and Jivko Sinapov. “Automaton-guided curriculum generation for reinforcement learning agents.” *In Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 33, pp. 605-613. 2023. [116]

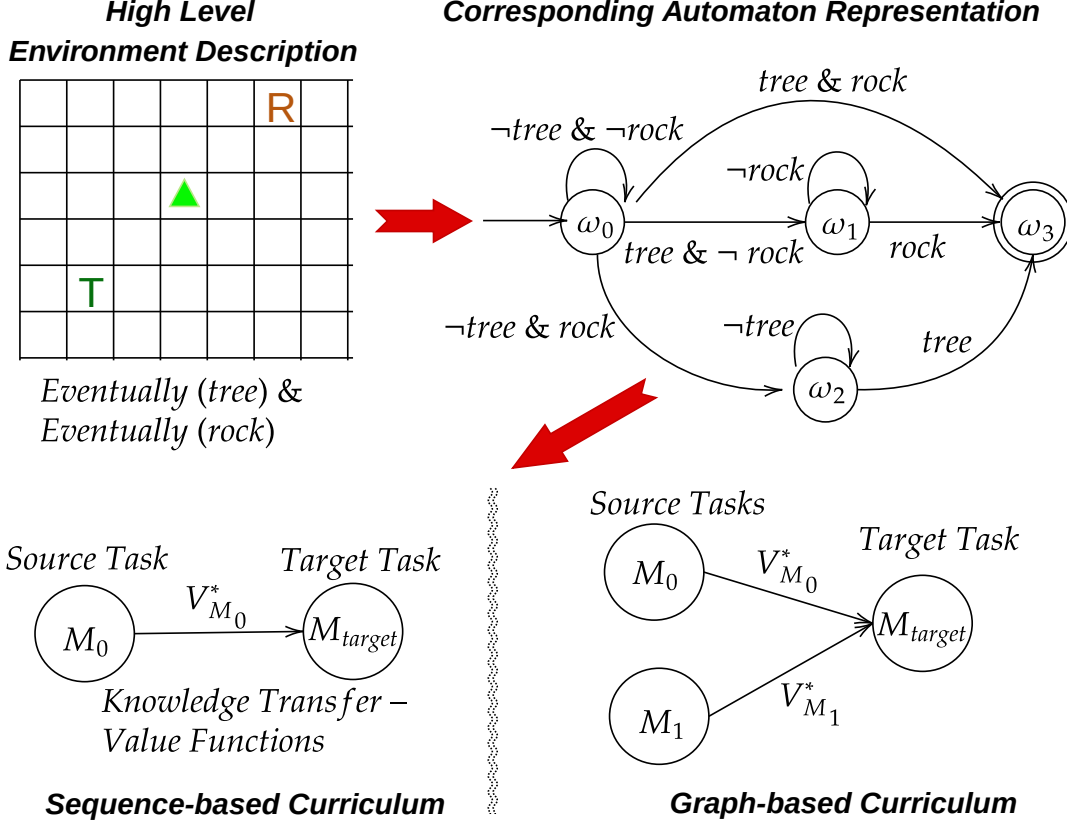


Figure 3.1: Overview of the automaton-guided curriculum generation procedure. Given the high-level task description, we use the equivalent automaton representation to come up with a sequence-based and a graph-based curriculum.

for shaping the reward function [14, 49, 135]. However, reward shaping falls short in complex sequential-decision making tasks as it does not generate simpler sub-tasks suited to the current knowledge of the agent, and incentivizes policies to reach local optima. In this work, we extend automaton-guided RL beyond reward shaping, by deriving a curriculum for the target objective. Curriculum Learning (CL) intends to optimize the order in which an agent attempts tasks, increasing the expected return while reducing training time for complex tasks [85, 33]. Nevertheless, automaton-guided curriculum generation still remains an unsolved problem in literature.

We explore representing the high-level task objective using finite-trace Linear Temporal Logic (LTL_f) formulas [22] which can be equivalently represented using Deterministic Finite Automaton (DFA). Using the structure of the DFA, we generate a curriculum for the target objective. While the DFA provides a graphical representation of the sequence in which the sub-goals must be achieved, it does not specify the individual sub-tasks

of the curriculum. Thus, generating a curriculum is non-trivial as it requires the agent to reason over multiple potential curriculum environment configurations for the same sub-goal objective. Given a high-level task objective in the form of a DFA, we propose *AGCL*, Automaton-guided Curriculum Learning (Fig. 3.1), that generates two types of curricula: 1) A sequence of sub-tasks that increase in order of difficulty for the agent, and 2) A directed acyclic graph (DAG), where agents can transfer knowledge concurrently learned in multiple source tasks to learn a common target task². Unlike previous graph-based curriculum approaches [126, 120], ours does not require access to all possible state configurations of the environment. Further, generating the curriculum does not require additional interactions with the environment, and learning through the curriculum yields quicker convergence to a desired performance compared to learning from scratch, automaton-guided reward shaping baselines - GSRS [14], QRM [49], and curriculum learning baselines - Teacher-Student [76] and self-play [125].

We further perform an extensive evaluation on a set of challenging robotic navigation and manipulation tasks and demonstrate that *AGCL* reduces the number of interactions with the target environment by orders-of-magnitude when compared to state-of-the-art curriculum learning and automaton-guided reinforcement learning baselines.

3.2 Related Work

Curriculum Learning (CL) in RL has been studied for games [37], robotic tasks [57], and self-driving cars [98]. A survey on CL [85] summarizes the three main elements of CL as task generation, task sequencing, and knowledge transfer. Task generation builds a set of source tasks given the target task [61]. Task sequencing optimizes the task sequence to enhance learning in the target task [84, 76], and knowledge transfer determines what information must be transferred from the source to the target to promote effective learning [20, 128]. Existing graph-based CL methods [126, 120] assume knowledge of the set of states in the task, and thus cannot be used for complex tasks that require function approximation. Metaheuristic search methods serve as a tool to evaluate the

²An in-depth analysis of Fig. 1 is provided in Section 3

CL frameworks [33]. In most methods, optimizing a curriculum is still computationally expensive and sometimes takes more interactions compared to learning from scratch. Our proposed framework addresses this concern by utilizing high-level logical specification to derive an optimized graphical representation of a curriculum, eliminating the sunk cost of interactions required to optimize the curriculum.

Automaton-guided RL approaches specify tasks using temporal logic-based high-level language specifications [132, 135, 51]. Most approaches generate a dense reward function to speed up learning. Learning separate policies for sub-goals of a task helps abstract knowledge which can be used in new tasks [49]. Another technique is to shape the reward inversely proportional to the distance from the accepting node in the automaton [14] however, it leads to inefficient reward settings. Augmenting the reward function with a dynamic Monte Carlo Tree Search helps mitigate this problem [135]. DIRM interleaves high-level planning with RL to learn a policy for each edge, overcoming challenges introduced by poor representations [55]. However, this approach becomes inefficient when there are multiple paths to the target in the graph, and other reward shaping techniques prove inefficient compared to curriculum learning [96]. In this work, we use the underlying DFA to develop a graphical representation of the curriculum, and implement a transfer learning mechanism compatible with a function-approximator, outperforming state-of-the-art baselines.

3.3 Theoretical Framework

3.3.1 Markov Decision Processes

An episodic Markov Decision Process (MDP) M is defined as a tuple

$(\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \mathcal{S}_0, \mathcal{S}_f, \gamma, L)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{P}(s'|s, a)$ is the transition function, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is the reward function, $\mathcal{S}_0$ and $\mathcal{S}_f$ are the sets of starting and final states respectively, and $\gamma \in [0, 1]$ is the discount factor. At 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^* ,

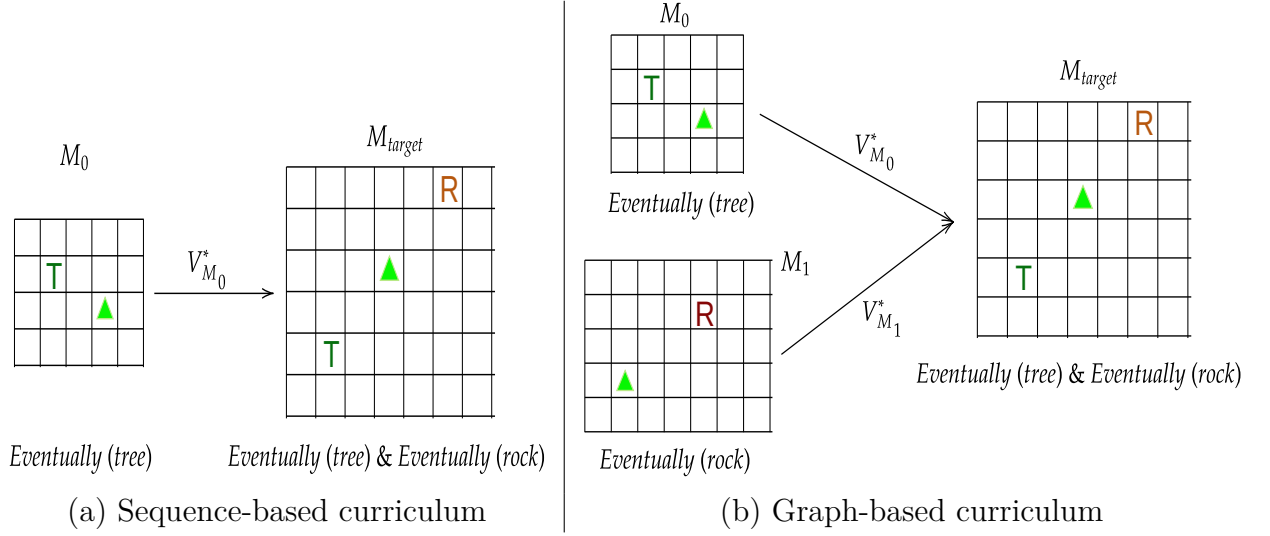


Figure 3.2: Examples of sequence-based (left) and graph-based (right) curricula. In a sequence-based curriculum, one source task contributes (knowledge) to one target task. In a graph-based curriculum, multiple source tasks contribute simultaneously to a target task.

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 . The labeling function $L : \mathcal{S} \rightarrow 2^{AP}$ maps a state in the MDP to a set of atomic propositions that hold true for that state. For example, in Fig. 3.1 the set of atomic propositions, $AP = \{tree, rock\}$.

3.3.2 Curriculum

Let $\mathcal{M}$ be a set of tasks, where $M_i = (\mathcal{S}_i, \mathcal{A}_i, P_i, R_i, \mathcal{S}_{0,i}, \mathcal{S}_{f,i}, \gamma_i, L_i)$ is a task in $\mathcal{M}$. Let $\mathcal{D}^{M_i}$ be the set of samples associated with task M_i : $\mathcal{D}^{M_i} = \{(s, a, r, s') | s \in \mathcal{S}_i, a \in \mathcal{A}_i, s' \sim P_i(\cdot | s, a), r \leftarrow R_i(s, a, s')\}$. A curriculum $T = (\mathcal{V}, \mathcal{E}, g, \mathcal{M})$ is a directed acyclic graph, where $\mathcal{V}$ is the set of vertices, $\mathcal{E} \subseteq \{(x, y) | (x, y) \in V \times V \wedge x \neq y\}$ is the set of directed edges, and $g : \mathcal{V} \rightarrow \{\mathcal{D}^{M_i} | M_i \in \mathcal{M}\}$ is a function that associates vertices to samples of a single task in $\mathcal{M}$. A directed edge (v_j, v_k) indicates that samples associated with $v_j \in \mathcal{V}$ should be trained on before samples associated with $v_k \in \mathcal{V}$. All paths terminate on a single sink node $v_t \in \mathcal{V}$ ³.

A graph-based curriculum is a general case of a curriculum, where the indegree and

³We find curriculum for single target task. Curriculum design for multiple target tasks is beyond the scope of this paper.

outdegree of each vertex $v \in \mathcal{V}$ can be greater than one, and there can be multiple source nodes but only one sink node. A sequence-based curriculum is a special case of a curriculum, where the indegree and outdegree of each vertex $v \in \mathcal{V}$ is at most 1, and there is only one source node and one sink node as shown in Fig. 3.1. Examples of a sequence-based curriculum and a graph-based curriculum are shown in Fig. 3.2(a) Fig. 3.2(b) respectively.

3.3.3 Object Oriented MDP

The Object-Oriented MDP (OOMDP) representation abstracts the task description, as to intuitively generalize the elements in the environment and their properties [27]. In OOMDPs, the task space is abstracted by a set of classes $\mathcal{C}$ where each class $C_i \in \mathcal{C}$ has a parameter set $Par(C_i) = \{C_i.p_1, \dots, C_i.p_{|p|}\}$. Each parameter $C_i.p_i$ has a range of values, given by $Range(C_i.p_i)$. At any given time, an environment consists of a set of objects $\mathcal{O} = \{o_1, \dots, o_{|\mathcal{O}|}\}$, where each object o_i is an instance of a class C_i and is defined by the parameters for that class $o_i = Par(C_i(o_i))$. For a task, the OOMDP state s_{oo} is given by the union of all object states $s_{oo} = \cup_{o_i \in \mathcal{O}} State(o_i)$, where each object state is the value of the parameters of that object: $State(o_i) = \{o_i.p_1, \dots, o_i.p_{|p|}\}$. Since the OOMDP representation is a high-level abstraction of the task space, we assume a many-to-one mapping between a subset of MDP states ($\mathcal{S}_{sub}$) and the OOMDP state (s_{oo}); $w : \mathcal{S}_{sub} \rightarrow s_{oo}$. For the task in Fig. 3.1, the OOMDP description consists of classes $\mathcal{C} = \{world_size, trees, rocks\}$, with the parameters: $\mathcal{C}(world_size) = \{width, height\}$ (width and height of the grid), $\mathcal{C}(trees) = \{trees_{env}, trees_{inv}\}$ (number of trees in the environment and inventory of the agent), and similarly, $\mathcal{C}(rocks) = \{rocks_{env}, rocks_{inv}\}$. In a task, classes have objects whose parameters are assigned values within a range. Example: $Par(world_size[height]) = [5, 10]$ denotes that the height of the grid can have any value within the range $[5, 10]$. The MDP transition $s \xrightarrow{a} s'$ and the OOMDP transition $s_{oo} \xrightarrow{a} s'_{oo}$ occur synchronously.

3.3.4 Linear Temporal Logic and Deterministic Finite Automata (DFA)

We define the high-level specification of our task using finite-trace Linear Temporal Logic (LTL_f) formulas [22]. LTL_f allows us to succinctly express complex temporal tasks such as reachability, safety, recurrence, persistence or a combination of these. Most importantly, the language of any LTL_f formula can equivalently be represented as a Deterministic Finite Automaton (DFA).

Given a set of atomic propositions, AP , a formula in LTL_f is constructed inductively using the operations $p, \neg\phi, \phi_1 \vee \phi_2, \mathbf{X}\phi, \mathbf{G}\phi, \mathbf{F}\phi, \phi_1 \mathbf{U} \phi_2$ where $p \in AP$ and ϕ, ϕ_1 and ϕ_2 are LTL_f formulas. The operator $\neg$ denotes negation, $\vee$ denotes disjunction, and the operators $\mathbf{X}, \mathbf{G}, \mathbf{F}, \mathbf{U}$ denote *Next*, *Always*, *Eventually*, and *Until* respectively. An LTL_f formula can be translated into a DFA $\mathcal{U} = \{\Omega, \omega_0, \Sigma, \delta, F\}$, where Ω is the set of nodes with initial node $\omega_0 \in \Omega$, $\Sigma = 2^{AP}$ is an alphabet defined over a set of atomic propositions AP , $\delta : \Omega \times \Sigma \rightarrow \Omega$ is a deterministic transition function, and $F \subseteq \Omega$ is the set of accepting nodes. Given any two states $\omega, \omega' \in \Omega$ and a symbol $\sigma \in \Sigma$, we denote the transition $\omega' = \delta(\omega, \sigma)$ as $\omega \xrightarrow{\sigma} \omega'$. A path $\zeta = \omega_0 \omega_1 \dots \omega_n$ is a finite-length sequence of states such that, for any $i = 0 \dots n-1$, we have $\omega_{i+1} = \delta(\omega_i, \sigma_i)$. The occurrence operator, $\text{Occ}(\zeta)$, defined as $\{\omega \in \Omega \mid \omega \text{ appears in } \zeta\}$, returns the set of nodes in the path ζ .

The set of atomic propositions in AP is shared within the MDP and the DFA. The labeled MDP and the automaton (DFA) operate synchronously. Specifically, a transition $s \xrightarrow{a} s'$ in the labeled MDP triggers a transition $\omega \xrightarrow{L(s')} \omega'$ in the DFA. Therefore, the high-level specification is achieved in the labeled MDP if and only if a final state is reached in the DFA. We define a one-to-many relationship using the function $P : \Omega \rightarrow \mathcal{M}$, that given a node $\omega \in \Omega$ in the DFA, returns the set of tasks that can reach the node ω from the start node ω_0 . The equivalent DFA representation of the LTL_f objective $\mathbf{F}(\text{tree}) \wedge \mathbf{F}(\text{rock})$ is shown in Fig. 3.1.

3.3.5 Running Example

Consider the environment description shown in Fig. 3.1. We start with the target task MDP M_{target} and its OOMDP description (described in Sec. 3.3.3) along with the LTL_f formula for the task $(\mathbf{F}(tree) \wedge \mathbf{F}(rock))$ which corresponds to the automaton shown in Fig. 3.1. When the agent follows a policy π that collects a tree followed by a rock, it satisfies the path $\zeta : \omega_0 \xrightarrow{tree} \omega_1 \xrightarrow{rock} \omega_3$. Since ω_3 is in the set of accepting nodes F , the policy produces a trajectory in the MDP that satisfies the LTL_f objective. Generating curriculum given the DFA involves reasoning over the trace paths as well as the set of MDPs that can generate trajectories to reach a node in the DFA. For example, to have a trajectory (of length > 0) that reaches the node ω_1 , the agent must have a *tree* in its inventory by the end of the episode. To achieve this, at least one *tree* must be present in the environment when the episode starts for the agent to collect. Using this information, a potential source task of the curriculum is generated by varying the parameter values for the objects within the range specified in the OOMDP representation. Thus, a query to the function $P(\omega_1)$ will generate the set of MDPs $\mathcal{M}_1$ that have at least 1 *tree* in the initial task configuration, while varying other object parameters. The set of initial and final OOMDP states for $\mathcal{M}_1$ will be given by $w(P(\omega_1)(\mathcal{S}_0))$ and $w(P(\omega_1)(\mathcal{S}_f))$ respectively as the function w maps the MDP state to an OOMDP state. An example of a sequence-based curriculum for the path $\zeta : \omega_0 \xrightarrow{tree} \omega_1 \xrightarrow{rock} \omega_3$ is shown in Fig. 3.2, where in the first task M_0 , the agent learns to collect a *tree* in a smaller environment, while transferring the knowledge to the final task. For the graph-based curriculum in Fig. 3.2, two agents simultaneously learn to collect a *tree* and a *rock* separately, and transfer their knowledge to learn the target task. The curriculum for the agent that learns to collect a *tree* in M_0 is derived from the set of MDPs produced using the path $\zeta : \omega_0 \xrightarrow{tree} \omega_1 \xrightarrow{rock} \omega_3$, whereas the curriculum for the agent that learns to collect a *rock* in M_1 is derived using the path $\zeta : \omega_0 \xrightarrow{rock} \omega_2 \xrightarrow{tree} \omega_3$. The agent’s trajectory in the final target task M_{target} can follow any of the paths in the DFA. The knowledge from the source tasks in the curriculum guides the agent in the final target task to make decisions based on its prior experiences. In the following sections, we discuss our automaton-guided curriculum generation approach.

3.3.6 Problem Formulation

CL aims to generate a curriculum, such that the agent’s convergence or its time-to-threshold performance on the final target task (M_{target}) improves relative to learning from scratch. Time-to-threshold (Δ) metric computes how much faster an agent can learn a policy that achieves expected return $G \geq \delta$ on the target task if it learns through a curriculum, as opposed to learning from another approach [33]. Here δ is desired performance threshold, in success rate or episodic reward. An optimal curriculum would converge to the expected return value in the target task the quickest. Formally:

$$T^* := \operatorname{argmin}_{T \in \mathcal{T}} [a_{\delta}^{M_{target}} + \sum_{M_i \in T} a^{M_i}]$$

where $\mathcal{T}$ is the set of all curricula, $a_{\delta}^{M_{target}}$ is the number of actions the agent took in the target task M_{target} to achieve threshold performance δ , and a^{M_i} is the number of actions the agent took in the source task M_i of the curriculum.

Our objective is to find a curriculum $T = \{\mathcal{V}, \mathcal{E}, g, \mathcal{M}\}$ that improves an agent’s time-to-threshold performance on the target task, given the LTL_f representation and the OOMDP and MDP description of the target task.

3.4 Methodology

The high-level curriculum generation algorithm is given in Algorithm 1. To begin, we assume the following:

1. We have access to a high-level target task (M_{tar}) objective which is expressed using LTL_f formulas, and can be equivalently represented using a DFA $\mathcal{U}$.
2. We have access to the OOMDP description of the target task. The OOMDP description contains the set of classes $\mathcal{C}$ of the environment, and for each class, the set of parameters for that class, along with the range of acceptable values for the parameters. Varying these parameters give us different tasks of the curriculum. We also have the initial and goal OOMDP states of the final task of the environment.

3. We have access to the functions P that given a node in the DFA, return the set of MDPs that can reach the node, and access to function w that performs mapping between MDP states and OOMDP states.

Given the DFA, we first generate a key-value pair dictionary K , where each key is a node of the automaton, and the value is the set of MDPs that can reach the node, with each MDP augmented with the initial and final OOMDP state configuration of the MDP, resulting in the MDP-OOMDP tuple $(M, s_{0,oo}, s_{f,oo})$, lines 1-2. The function **Get_Trace_Paths** returns the set of all paths (Z) that start from the initial node ω_0 and reach an accepting node $\omega_f \in F$. We consider only acyclic DFAs, ignoring paths that contain a cycle. A curriculum designed with this consideration will prune curriculum candidates that intend to learn the same task repeatedly, ensuring progress in the curriculum. Then, for each path $\zeta \in Z$, the **List_Candidates** function generates an ordered-list of sequence-based curriculum candidates by choosing a MDP-OOMDP tuple from each node of the path, and sequencing it according to the sequence of nodes in the path ζ . We repeat this until we attain a set of all possible combinations of MDP-OOMDP tuple sequences (lines 4-7). This yields an exhaustive set Ψ where each element in the set is an ordered list of MDP-OOMDP tuples, and the length of the list is equal to the number of transitions in the trace path. Thus, each list corresponds to a curriculum candidate T where the vertices of the graph are the MDPs in the list, and the edges are defined by the sequence of the MDPs in the list. Once we have a set of sequence-based curricula candidates, we choose an effective curriculum from this set. We introduce the jump score, where we assign a score for each pair of consecutive source tasks in the curriculum (lines 10-16).

Algorithm 1 AGCG($\mathcal{U}, M_{tar}, w, P, \eta, \text{CL-Type}$)

Require: $\mathcal{U}, M_{tar}, w, P, \eta, \text{CL-Type}$ **Ensure:** Curriculum: T

1: $Z \leftarrow \emptyset$ ▷ Placeholder Initialization
2: $K \leftarrow \emptyset$ ▷ Set of all acyclic trace paths
3: $\mathcal{J} \leftarrow \emptyset$ ▷ Node-MDP Dictionary
4: $\Psi \leftarrow \emptyset$ ▷ Curriculum - Jump Score dictionary
5: **for** $\omega \in \Omega$ **do** ▷ Set of curriculum candidates
6: $K \leftarrow K \cup \{\omega : (P(\omega), w(P(\omega)(\mathcal{S}_0)), w(P(\omega)(\mathcal{S}_f)))\}$ ▷ Algorithm
7: **end for**
8: $Z \leftarrow \text{GET_TRACE_PATHS}(\mathcal{U})$
9: **for** $\zeta \in Z$ **do**
10: **for** $\omega_i \in \text{Occ}(\zeta)$ **do**
11: $\Psi \leftarrow \Psi \cup \{\text{LIST_CANDIDATES}(K(\omega_i))\}$
12: **end for**
13: **end for**
14: **for** $\psi \in \Psi$ **do**
15: $\text{jump_sum} \leftarrow 0$
16: **for** $i = 1$ to $|\psi| - 1$ **do**
17: $\text{jump_sum} \leftarrow \text{jump_sum} + J_{M_i \rightarrow M_{i+1}}$
18: **end for**
19: $\bar{J}_\psi \leftarrow \text{jump_sum} / |\psi|$
20: $\mathcal{J} \leftarrow \mathcal{J} \cup \{\psi : \bar{J}_\psi\}$
21: **end for**
22: **if** CL-Type = Sequence-based **then**
23: $T \leftarrow \arg \min_{\psi \in \mathcal{J}} \mathcal{J}[\psi]$
24: **else if** CL-Type = Graph-Based **then**
25: $T' \leftarrow \{\psi \in \Psi \mid \mathcal{J}[\psi] \leq \eta\}$
26: $T \leftarrow \text{GRAPH}(T')$
27: **end if**
28: **return** T

Jump score: Inspired by the difficulty scores for supervised learning [137], we assign a jump score $J : (M_i, M_j) \rightarrow \mathbb{R} \in [0, 1]$ for each pair of consecutive source tasks M_i and M_j :

$$J_{M_i \rightarrow M_j} = 1/2 * (sim_t(M_j, M_{tar}) - sim_t(M_i, M_{tar}) + sim_g(M_j, M_{tar}) - sim_g(M_i, M_{tar})) \quad (3.1)$$

where, $sim_t(M_i, M_{tar})$ is the task configuration similarity between the task M_i and the final target task M_{tar} ⁴, while $sim_g(M_i, M_{tar})$ is the goal state similarity. The jump score $J_{M_i \rightarrow M_j}$ calculates how dissimilar two consecutive source tasks in the curriculum are by calculating the difference of the similarity of the task and goal configurations of each of the two consecutive tasks with the final target task.

Since we have the OOMDP initial state configuration $s_{0,oo}$ of each pair of consecutive source tasks of the curriculum (M_i, M_j) and the final target task M_{tar} , the task configuration similarity between a task and the final target task is measured as the parameter value overlap between the initial OOMDP state values of the two tasks.

$$sim_t(M_i, M_{tar}) = \frac{1}{\sum_{o \in \mathcal{O}_{M_{tar}}} |o.p^0|} \sum_{o \in \mathcal{O}_{M_{tar}}} \sum_{o.p \in o} \frac{o.p_{M_i}^0}{o.p_{M_{tar}}^0} \quad (3.2)$$

where $o.p_{M_i}^0$ is the value of the parameter of the object o in the OOMDP state $s_{0,oo}$ of task M_i . Likewise, we also have the OOMDP goal state $s_{f,oo}$ of each pair of consecutive source tasks and the final target task. We calculate the OOMDP goal state similarity of a task with respect to the final target task using:

$$sim_g(M_i, M_{tar}) = \frac{1}{\sum_{o \in \mathcal{O}_{M_{tar}}} |o.p^f|} \sum_{o \in \mathcal{O}_{M_{tar}}} \sum_{o.p \in o} \frac{o.p_{M_i}^f}{o.p_{M_{tar}}^f} \quad (3.3)$$

where $o.p_{M_i}^f$ is the value of the parameter of the object o in the OOMDP state $s_{f,oo}$ of task M_i .

A lower jump score corresponds to a pair of tasks that have similar initial OOMDP task configurations as well as similar OOMDP goal state configurations. Along any path ζ , the sum of the jump scores among all consecutive task pairs will always be equal to 1.

⁴More on MDP similarity: [136]

The set of OOMDP states the agent visits in an episode can be potentially large, and do not yield much information that might help in choosing a suitable curriculum candidate. Additionally, access to the set of OOMDP states that the agent might visit in an episode before the agent even attempts the episode requires knowledge of the transition dynamics, which is unavailable. Hence, only the initial and final states are used while calculating the jump score.

Next, we calculate the average jump score of each curriculum candidate $\psi \in \Psi$ and store it in a Curriculum - Jump Score dictionary $\mathcal{J}$ (line 16). For sequence-based curricula, the curriculum is determined by the candidate that yields the lowest average jump score (lines 17-18).

$$T = \underset{\psi \in \Psi}{\operatorname{argmin}} \mathcal{J}(\bar{\mathcal{J}}_{\psi}) \quad (3.4)$$

Intuitively, the curriculum T is an ordered list of MDP-OOMDP tuples that have the lowest cumulative average jump among any two consecutive source tasks, i.e., the curriculum has the lowest task and goal state dissimilarity between any two consecutive tasks. A lower dissimilarity will result in a stronger knowledge transfer.

The graph-based curriculum is determined by the set of elements in Ψ that yield the cumulative average jump score lower than a predetermined threshold value η (lines 19-21).

$$T' = \{\psi \in \Psi \ni \mathcal{J}(\bar{\mathcal{J}}_{\psi}) \leq \eta\} \quad (3.5)$$

Thus, T' is a set where each element is an ordered list of MDPs (each element is a sequence-based curriculum). A graph is generated by considering equivalent MDPs in T' as common nodes, and the edges are given by the sequence of MDPs in each element in T' (line 22). The weights on the edges denote the proportion of knowledge contribution of a source task and is discussed further below. The worst case time-complexity of the algorithm is $O(|\mathcal{V}|^3|\mathcal{E}||C|^2|p|^2)$ and where $|\mathcal{V}|$ and $|\mathcal{E}|$ are the number of vertices and edges in the DFA's DAG and $|C|$ and $|p|$ are the number of classes and maximum number of class parameters in the OOMDP.

Knowledge Transfer in curriculum learning involves leveraging learned knowledge from a source task and transferring relevant knowledge to the next task in the curriculum.

In our curriculum setting, each individual source task is learned using DQN [82], and hence we perform value function transfer, where the weights of the neural network of the learned value function of a source task are initialized as the weights of the value function of the next task, i.e., $V_{M_j}^i \leftarrow V_{M_i}^*$. Thus, instead of commencing the next task M_j with random $V(s)$ values, the learned value function of the source task biases action selection in the next task according to the experience already collected in the source task. In scenarios where the action spaces are continuous, the RL algorithm needs to be adapted to suit the needs of the domain. DDPG [69] and PPO [109] allow continuous action spaces and can replace the DQN as the base RL algorithm. In case of a sequence-based curriculum, where we transfer knowledge from only one source task M_i to the next task M_j in the curriculum, the value function of M_j is initialized to be the learned value function of M_i . This process is continued until we reach the end of curriculum, culminating in the final target task M_{tar} . In case of a graph-based curriculum, where we transfer knowledge from multiple source tasks $(M_i, M_{i+1}, \dots)$ to the next task in the curriculum (M_j), the value function of the next task (M_j) is initialized to be a weighted sum of the value functions of its source tasks, i.e.

$$V_j^i = \beta_0 \cdot V_i^* + \beta_1 \cdot V_{i+1}^* + \dots + \beta_l \cdot V_{i+l}^* \quad (3.6)$$

where V_j^i is the initial value function of the target task M_j , V_i^* is the learned value function of the source task M_i , l is the number of source tasks, and β is a scalar weighting factor. To calculate β , we use the following:

$$\beta_i \propto \frac{1}{J_{M_i \rightarrow M_j}} \quad \text{and} \quad \sum_{k=0}^{k=l} \beta_{i+k} = 1 \quad (3.7)$$

i.e., the lower the jump score between tasks M_i and M_j , the higher the similarity between their OOMDP task and goal configurations, thus, higher the weightage of the learned value function of the task M_i , and vice versa. A higher similarity will correspond to a stronger positive transfer. In case of a graph-based curriculum, we learn the tasks that correspond to the leaf nodes initially, transferring knowledge using Equations 3.6 and 3.7, culminating in the final target task.

3.5 Experimental Results

We aim to answer the following questions: (1) Does AGCL yield sample efficient learning? (2) How does it perform in environments that have distractor objects that are not modeled in the LTL_f specification? (3) How does it perform when the exact OOMDP description is unknown? (4) Does it yield sample efficient learning when the OOMDP parameter space is continuous? (5) How does it perform when generating all potential curricula candidates is intractable?

3.5.1 AGCL - Gridworld Results

To answer our first question, we evaluated *AGCL* on a grid-world domain with the LTL_f objective:

$$\mathbf{G}((t \rightarrow \neg r \wedge \neg p) \wedge (r \rightarrow \neg t \wedge \neg p) \wedge (p \rightarrow \neg r \wedge \neg t)) \wedge (\neg p \mathbf{U}(t \wedge \mathbf{X}(\neg p \mathbf{U}t))) \wedge (\neg p \mathbf{U}r) \wedge \mathbf{F}p \quad (3.8)$$

where t, r, p correspond to the atomic propositions *tree*, *rock* and *pogo-stick*.

Essentially, the agent needs to collect 2 *trees* and 1 *rock* (by navigating to the objects and *breaking* them) before approaching the crafting table to *craft* a *pogo-stick*. This is a complex sequential decision making task and requires $\sim 10^8$ interactions to reach convergence [114]. The OOMDP description is given by:

$\mathcal{C} = \{world_size, trees, rocks, crafting_table\}$, where the class parameters are:
 $C(world_size) = \{width, height\}$; $C(x) = \{x_{env}, x_{inv} | x \in \{trees, rocks\}\}$ and
 $C(crafting_table) = \{crafting_table_{env}\}$. The parameters *width* and *height* can assume values in the range $[6, 12]$ and the number of *trees*, *rocks* and *crafting_table* in the environment can assume a value in the ranges of $[0, 4]$, $[0, 2]$ and $[0, 1]$.

In this environment, the agent can move 1 cell forward if the cell ahead is clear or rotate $\pi/2$ left or right. In the target task, the agent receives a reward of 10^3 upon crafting a *pogo-stick*, and -1 reward for all other steps. 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 agent receives a local-view of its environment). Two additional sensors provide information on the amount of *trees* and *rocks* in the agent’s inventory.

In the gridworld and the Crafter-Turtlebot domains, the set of actions that the agent can take are: move forward, rotate left, rotate right, break and craft. Move forward makes the agent go forward by 1 cell in the gridworld domain and by $0.1m$ in the Crafter-Turtlebot domain. The rotate actions cause the agent to rotate $\pi/2$ in the gridworld domain and by $\pi/8$ in the Crafter-Turtlebot domain. The break action causes the object in front of the agent to disappear from the environment and appear in the inventory. It does not have any effect if there is no object in front of the agent (i.e. it is empty). Similarly, the agent can craft a pogo-stick only if it has 2 trees and 1 rock in the inventory and is facing the crafting-table.

The learning curves in Fig. 3.3 depict the performance of our proposed sequence and graph-based *AGCL* method against five baseline approaches namely learning from scratch, automaton-guided reward shaping baselines: GSRS [14], QRM [49], and curriculum learning baselines: Teacher-Student [76] and self-play [125], all implemented using the same RL algorithm DQN [82]. The automaton-guided reward shaping baselines do not employ a curriculum and build upon naive reward shaping by modifying the reward inversely proportional to the distance from the DFA goal state and by learning policies for each DFA state transition. The curriculum learning baseline approaches do not utilize a reward machine, and rely on optimizing the task sequence through agent’s experience.

The learning curve for the curriculum approaches have an offset on the x-axis to account for the interactions used to go through the curriculum before moving on to the target task, signifying *strong transfer* [128]. The results in Fig. 3.3a show that *AGCL* reaches a successful policy quicker, and the graph-based curriculum proposed by *AGCL* outperforms the sequence-based approach. Learning source tasks in a graph-based curriculum is also parallelized, reducing the overall wall time. Furthermore, our method outperforms the five baseline approaches in terms of learning speed. The other four baseline approaches perform better than learning from scratch, but do not outperform *AGCL*.

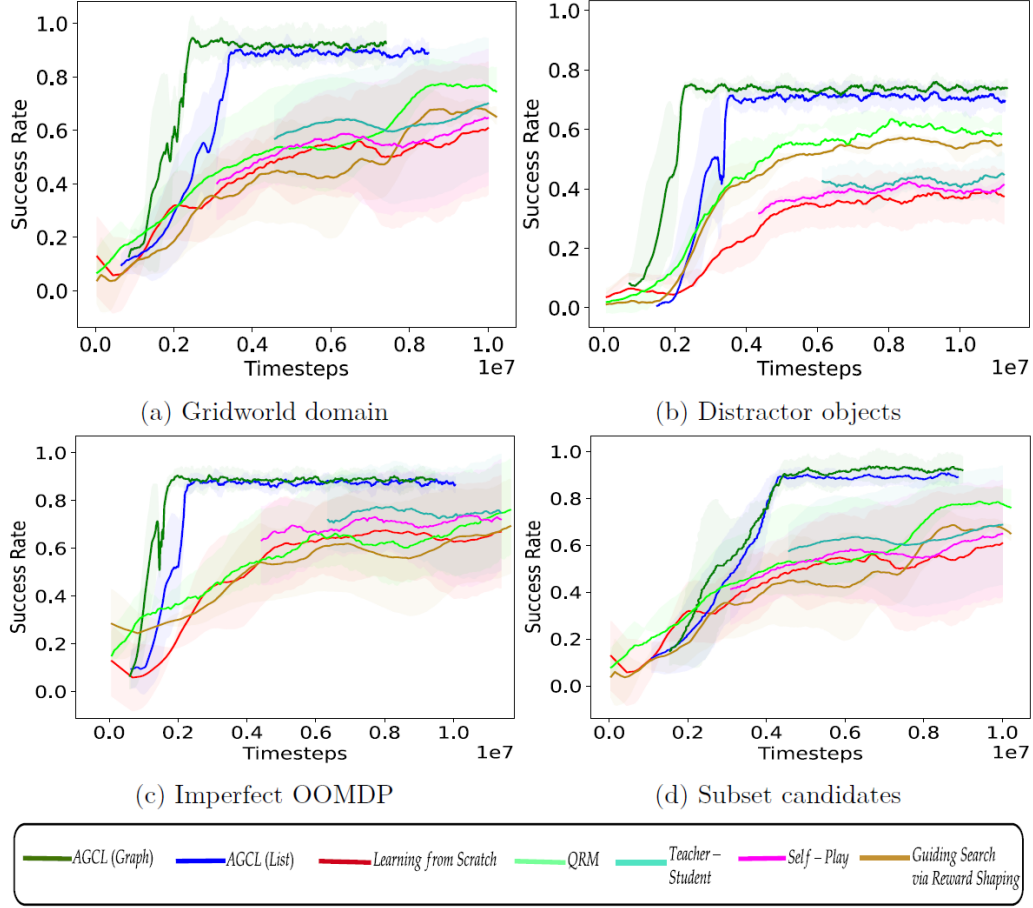


Figure 3.3: Learning curves (Averaged over 10 trials): (a) Gridworld domain; (b) Gridworld - with distractor objects in the environment; (c) Gridworld - with imperfect OOMDP description of the environment; (d) Gridworld - with subset of curriculum candidates.

All approaches may converge to a higher asymptote, but will need more interactions with the environment. For this task, the curriculum generated by *AGCL* has $|\mathcal{V}| = 4$ and $|\mathcal{V}| = 6$ for the sequence-based and graph-based curricula respectively.

To demonstrate that the average convergence rate of *AGCL* is consistently higher than the baseline approaches, we perform an unpaired t-test [59] to compare *AGCL* against the best performing baselines at the end of 10^7 training interactions and we observed statistically significant results (95% confidence). Thus, *AGCL* not only achieves a better success rate, but also converges faster. Further experiments are required to validate the effectiveness of *AGCL* on physical robotic domains.

3.5.2 AGCL on Tasks with Distractor Objects

Next, we test our *AGCL* approach (sequence and graph) on the *pogo-stick* task described in Sec. 3.5.1 but now the OOMDP description of the environment contains a new class $C(distractor) = \{distractor_{env}, distractor_{inv}\}$, whose object $distractor_{env}$ or $distractor_{inv}$ can assume values in the range $[0, 2]$, signifying that the initial task state or the inventory may contain 0 to 2 instances of the *distractor* object. The agent interacts with this object the same way it interacts with *trees* and *rocks*, i.e. by navigating to the object and collecting it in inventory. However, presence of *distractor* objects in the inventory of the agent is not necessary to achieve the target objectives and is not modeled in the LTL_f specification.

Fig. 3.3b demonstrates how our proposed *AGCL* performs against the baseline approaches. Since our curriculum generation approach models not only the goal configuration but also the initial task space configuration, it is successful in estimating the difficulty posed by the introduction of the distractor object as they are present in the initial state of the target task. The curriculum generated by *AGCL* guides the agent to a successful policy in fewer interactions compared to the other baselines. Approaches that design a curriculum purely from the LTL_f specification will not model the distractor objects, and will thus fail to determine the complexity of a source task that contains these distractor objects.

3.5.3 Tasks with Imperfect OOMDP Descriptions

In certain partially observable settings, it might not be possible to obtain an accurate OOMDP description of the environment. To demonstrate the efficacy of our approach in imperfect OOMDP descriptions, we incorporate a gaussian noise over the class parameter value ranges, i.e., if the parameter range is $Par_{given}(world_size[height]) = [a, b]$, we assume that this range is imperfect, and the true range is given by

$Par_{noisy}(world_size[height]) = [a - \mathcal{N}(0, \sigma), b + \mathcal{N}(0, \sigma)]$, where $\sigma = (b - a)/6$; covering the entire range of the parameter values in six standard deviations. We incorporate a gaussian noise over all the class parameters ranges.

3.5.4 AGCL on Subset of Curriculum Candidates

In cases where the range of parameter values or the DFA representation is too large, the number of curriculum candidates can grow exponentially in terms of the number of nodes in the DFA, making it infeasible to calculate the jump-score for each curriculum candidate. For this, we evaluate *AGCL* on the task described in Sec. 3.5.1 by sampling a subset Ψ_{sub} of the total curriculum candidates Ψ , such that $|\Psi_{sub}| = |\Psi|/4$, thus reducing the total number of curriculum candidates and the amount of computation required to determine a suitable curriculum. From the results in Fig. 3.3d, we observe that in this experiment, *AGCL* takes more interactions to converge to a desired policy as compared to Fig. 3.3a, as sampling a subset prunes some desirable curriculum candidates, however it outperforms the other curriculum learning and automaton-guided RL baselines in terms of learning speed.

We test *AGCL* (sequence and graph) on the same task of Sec. 3.5.1 but now the environment’s OOMDP description is noisy over the parameter values, i.e., the exact OOMDP description is unknown. We observe that even with imperfect OOMDP descriptions, *AGCL* converges faster than the baselines (Fig 3.3c), and the graph-based *AGCL* converges to a successful policy the quickest, in 2.5×10^6 interactions.

3.5.5 Baselines Implementation

As mentioned in the paper, we adapted four baseline approaches from the literature: Asymmetric Self-Play [125], Teacher-Student Curriculum learning [76], Q-learning for reward machines [49] and Guiding Search via Reward Shaping [14]

Asymmetric Self-Play

We adapted the asymmetric self-play approach proposed by [125] to our problem of crafting a pogo-stick. Alice, the goal-proposing policy, proposes a goal for Bob, whose goal is to reach the goal proposed by Alice. We used the approach in *Repeat* mode, where Bob repeats the task instead of reversing it, and tries to reach the goal proposed by Alice.

The maximum number of steps for Alice and Bob were set to 150, equal to the steps allowed for our *AGCL* approach. We tuned the hyperparameters through experimentation for our task, using a heuristic grid search. The parameters we used were: Self-Play percentage: 10%, Self-play reward scale γ : 0.01 and the self-play mode at *repeat*.

According to [125], Alice’s reward function is given by $r_A = \max(0, t_B - t_A)$ where t_B is the time taken for Bob to reach the goal state proposed by Alice, whereas t_A is the time taken by Alice to propose a goal (execute ‘stop’ action). In the target task, the goal is to craft a pogo-stick. We see that Alice’s rewards are sparse, and for Alice to keep receiving positive rewards, it needs to propose a goal that is just difficult for Bob to repeat. In our task, the large goal space makes it difficult for Alice to propose a task that is just difficult for Bob to repeat, making it difficult to optimize the curriculum according to the goals proposed by Alice, taking it longer to obtain a curriculum.

One difficulty is that the time (in interactions) spent by Alice in proposing a task is a sunk cost, and requires tuning the self-play percentage parameter to not increase this added cost. The self-play approach does help Bob explore the space, and is responsible for the jumpstart Bob achieves initially in the target task, evident from the baseline comparison learning curves. From our experiments, we observed that the exploration helped Bob in navigation, but was not effective in performing the sequential actions required for the target task.

Teacher-Student Curriculum Learning

The proposed teacher-student curriculum learning [76] assumes a teacher optimizing the sequence of the tasks in the curriculum for the student. The teacher proposes those tasks for which the student shows the highest potential in learning, i.e. the tasks for which the student learns the quickest. The ability of a student to learn a task is given by its slope of the curve of plotted reward function. The steeper the slope, the quicker the student is able to learn the task. One limitation of this approach is that it assumes the source tasks of the curriculum are defined. This is difficult in settings where the parameters of the task are continuous. We used the same DQN network for teacher-student curriculum learning approach as we did for *AGCL*. We initialized 4 source tasks and one target task, with varying parameters for each source task. The performance of the agent in the final target task is plotted. We followed the same reward shaping for the tasks in the curriculum as for the source tasks of the automated curriculum transfer approach.

In teacher-student curriculum learning, the sunk cost of the algorithm is the interactions in the source task. The sunk cost is very high when the interactions are costly, as in our high fidelity setup. This contributes to the high requirement of computational resources.

Q-Learning for Reward Machines

QRM [49] decomposes the task into sub-goals and intends to learn one q-value function for each state in the automaton. As an example, consider the automaton in Fig. 3.1, each of the nodes in the automaton will have a corresponding q-value function. At all times during the training, the agent keeps track of its state in the automaton. While choosing an action given the current observation, the agent chooses action depending on its current state in the reward machine. For example, if the agent is in node ω_1 , it will choose an action given by the q-value function q_{ω_1} corresponding to the node ω_1 . While updating the q-value function, we use the reward machine to determine the reward the agent would have received had it been in the reward state given by the action chosen, i.e. to update

$$q_\omega \stackrel{\alpha}{\leftarrow} r(s, a, s') + \gamma \max_{a'} q_{\omega'}(s', a')$$

Here, the maximization is over the $q_{\omega'}$ that determines the new reward state of the machine.

We implemented this approach for the grid-world domain where the LTL_f objective is given by:

$$\begin{aligned} \mathbf{G}((t \rightarrow \neg r \wedge \neg p) \wedge (r \rightarrow \neg t \wedge \neg p) \wedge (p \rightarrow \neg r \wedge \neg t)) \\ \wedge (\neg p \mathbf{U}(t \wedge \mathbf{X}(\neg p \mathbf{U} t))) \wedge (\neg p \mathbf{U} r) \wedge \mathbf{F} p \end{aligned} \quad (3.9)$$

where t, r, p are the atomic propositions corresponding to *tree*, *rock* and *pogo-stick* respectively.

Guiding Search via Reward Shaping

GSRS [14] shapes the reward inversely proportional to the distance from the accepting node in the automaton. The reward shaping is given by: $R'(s, a, s') = R(s, a, s') + F(s, a, s')$ where R is the original reward function and F is the shaping function. Here, F is given by:

$$F(s, a, s') = \gamma \phi(s') - \phi(s)$$

Here, F does not depend on the previous 2 states, but the last two states visited in the reward machine. This ensures that a positive shaping reward is given to transitions that make progress toward the goal state in the automaton. We shape the reward incrementally from the start state in the automaton until the accepting state, such that every time the agent takes an action that triggers a transition in the automaton to a different state, a positive reward is given that is higher than the previous reward. In the end, to preserve optimality, we subtract the shaped reward from the sum of rewards gained.

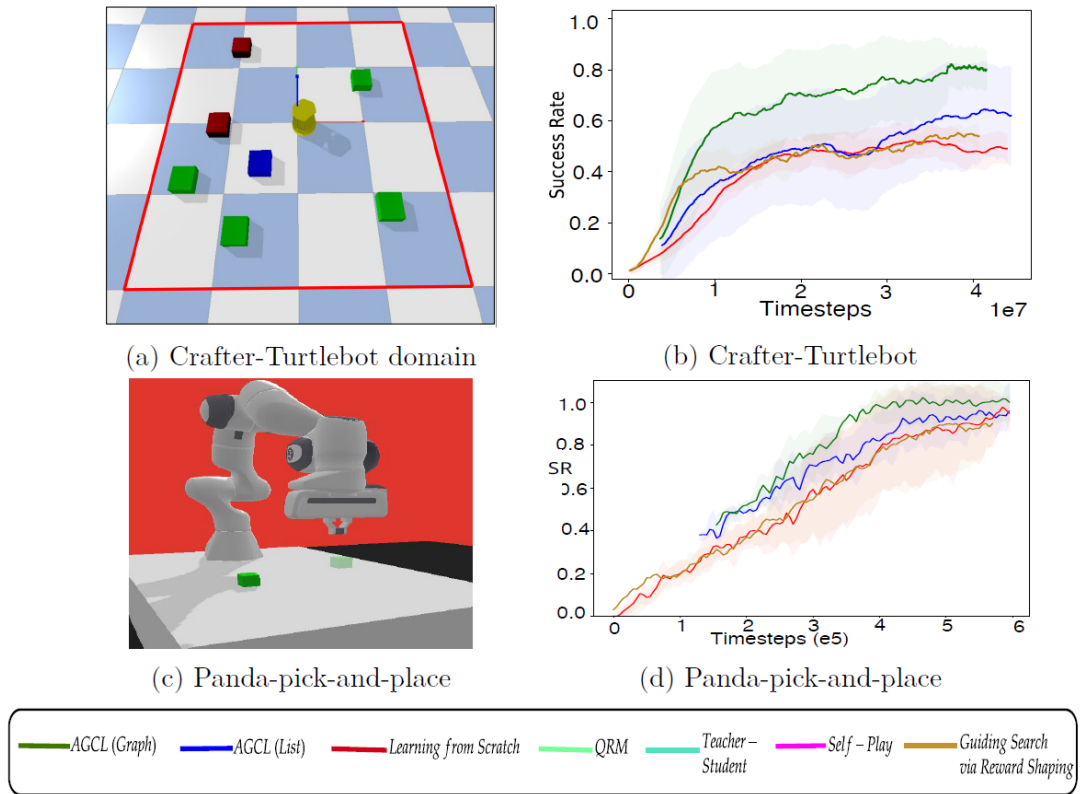


Figure 3.4: (a) and (c) show the two robotic domains - Crafter-Turtlebot and Panda-pick-and-place respectively. Learning curves (Averaged over 10 trials): are shown for (b) Crafter-Turtebot domain; and (d) Panda-pick-and-place domain.

3.5.6 Tasks with Continuous OOMDP Ranges

In the gridworld, the OOMDP parameter values were integers. In this experiment, we test how our approach performs in tasks that have continuous parameter ranges. We modify the function $P : \Omega \rightarrow \mathcal{M}$ that given a node in the DFA, samples a subset of MDPs $\mathcal{M}_{sub} \subseteq \mathcal{M}$ such that $|\mathcal{M}_{sub}| = b$. We test this on two challenging simulated robotic environments where the interaction cost is high.

Fig. 3.4a shows a robotic task with the same objective described in Sec. 3.5.1. Here the values for the parameters *width*, *height* for class *world_size* are in the continuous range $[2m, 4m]$. The move forward (backward) action causes the robot to move forward (backward) by $0.1m$ and the robot rotates by $\pi/8$ radians with each rotate action. Additionally, the objects can be placed at continuous locations in this robotic domain as compared to discrete grid locations for the gridworld task. These changes increase the number of MDP and OOMDP states the agent can attain.

The second simulated robotics environment (Fig. 3.4c) consists of a robotic arm performing a pick-and-place task [35] with the LTL_f objective: $\mathbf{F}(g \wedge \mathbf{F}(p \wedge \mathbf{F}q))$ where g, p, q are the atomic propositions for ‘reach-object’, ‘pick-up-object’ and ‘place-object’. The OOMDP is modeled as $\mathcal{C} = \{\text{world_size}, \text{objects}\}$ with $\text{Par}(\text{world_size}) = \{\text{length}, \text{width}, \text{height}\}$ and $\text{Par}(\text{objects}) = \{\text{no_objects}\}$. The continuous parameters $\{\text{length}, \text{width}, \text{height}\}$ assume values between $[10, 50]cm$. The robot has continuous action parameters for moving the arm and a binary gripper action (close/open). The robotic domains were modeled using PyBullet [19]. For each node, we sample $b = 25$ different MDPs and evaluate *AGCL*. The results in Figs.(3.4b,3.4d), show that *AGCL* outperforms all other baselines in both environments. Additionally, the graph-based *AGCL* converges to a successful policy the quickest.

However, these experiments were conducted on simulated robotics domains. Further experiments are required to validate the effectiveness of *AGCL* on physical robotic domains.

3.6 Summary

We proposed *AGCL*, a framework for curriculum generation using the high-level LTL_f specification and the OOMDP description of the environment. *AGCL* decomposes the target objective into sub-goals, and generates a sequence-based or a graph-based curriculum for the task. Through experiments, we demonstrated that *AGCL* accelerates learning, converging to a desired success rate quicker as compared to other curriculum learning and automaton-guided RL baselines. Moreover, our proposed approach improves learning performance even in the presence of distractor objects in the environment that are not modeled in the LTL_f specification. Finally, we evaluate our approach on long-horizon complex robotic tasks where the state space is large. *AGCL* reduces training time without relying on human-guided dense reward function nor does it require a perfect OOMDP description of the environment. The graph-based curricula produced by *AGCL* affords learning separate behaviors in parallel, reducing wall clock time for complex sequential decision making tasks. Thus, our proposed *AGCL* approach offers accelerated learning when the high-level task objective is available.

We assume that the high-level task objective is available, and can be characterized using a LTL_f formula. However, in certain cases, the target tasks can be completely novel, with no prior information on the task objective. Additionally, it might not be feasible to represent the domain using OOMDP representation, as the objects in the environment can be unknown, and the only way to get more information about the environment is through interaction with the environment. In scenarios where the LTL_f specification becomes complex, the DFA gets larger, and it might become intractable to come up with exact solutions of curriculum optimization. In the next chapter, we look at methods aimed at designing a curriculum without relying on the OOMDP representation of the environment or the LTL_f specification for a task.

Chapter 4

Curriculum as a Dynamic Sub-Task Selection Strategy Using Temporal Logic and LLMs

In the previous chapter, we introduced Automaton-Guided Curriculum Learning (AGCL). AGCL is a curriculum generation technique for reinforcement learning tasks using the environment’s object oriented markov decision process (OOMDP) representation and the linear temporal logic specification of the task. However, in certain situations, designing the OOMDP representation is infeasible or simply time consuming. In this chapter, we look at ways for curriculum generation without the requirement of the OOMDP representation of the environment. In this work, aim to improve sample efficiency by incorporating high-level task specifications, like Linear Temporal Logic (LTL) formulas, Reward Machines (RM), or Large Language Models (LLMs). This work presents a unified framework that combines logical specifications with LLM-guided sub-goal planning to enhance RL efficiency in environments with unknown dynamics.

We introduce two main contributions:

(1) Logical Specifications-guided Dynamic Task Sampling (LSTS): This method dynamically samples promising tasks based on logical specifications to train RL policies while

minimizing environmental interactions. Unlike prior approaches, LSTS does not require access to environment dynamics or a predefined RM.

(2) LLM-guided Teacher-Student learning (LgTS): In this approach, LLMs generate a graphical representation of sub-goals, helping the RL agent learn policies for sequential decision-making tasks through a Teacher-Student framework. LgTS does not require fine-tuned LLMs or pre-trained policies and can adapt to partially observable environments.

We evaluate both methods across two domains - a gridworld and a search-and-rescue task. Experimental results show that LSTS outperforms RM and Automaton-guided RL methods, such as Q-Learning for Reward Machines and Compositional RL from Logical Specifications (DIRL), in terms of sample efficiency. Similarly, LgTS demonstrates improved policy learning efficiency using LLM-generated sub-goals. These approaches highlight the potential of integrating logical specifications and LLM-guided planning to enhance RL agents’ performance in complex sequential decision-making tasks ¹.

4.1 Introduction

Advancements in Reinforcement Learning (RL) have enabled agents to learn optimal control behaviors across a wide range of tasks, from Minecraft games [114] to executing robotic manipulation [73]. However, model-free RL methods still face significant challenges, particularly in terms of sample inefficiency due to sparse rewards and unknown transition dynamics [116]. To address these challenges, prior work has proposed using intricately shaped reward functions to guide the agent toward its goal [43]. However, designing such reward functions is labor-intensive, requiring human engineers to determine the relative importance of different task components. Poorly designed reward functions

¹This chapter is based on the following two papers:

Shukla, Yash, Wenchang Gao, Vasanth Sarathy, Alvaro Velasquez, Robert Wright, and Jivko Sinapov. “LgTS: Dynamic Task Sampling using LLM-generated Sub-Goals for Reinforcement Learning Agents.” *In Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, pp. 1736-1744. 2024. [118]

Shukla, Yash, Tanushree Burman, Abhishek N. Kulkarni, Robert Wright, Alvaro Velasquez, and Jivko Sinapov. “Logical specifications-guided dynamic task sampling for reinforcement learning agents.” *In Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 34, pp. 532-540. 2024. [117].

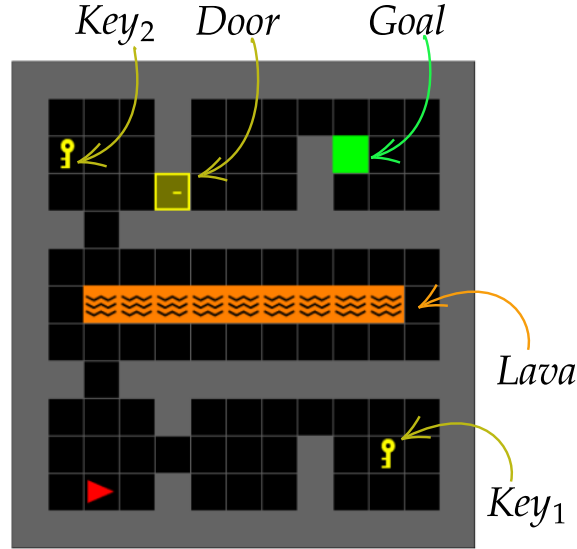


Figure 4.1: A rendering of the Gridworld domain. The agent (shown in red triangle) needs to collect 1 *Key*, unlock the *Door* and reach the *Goal* to execute the task successfully.

risk driving agents toward local optima, where they prioritize certain sub-goals while neglecting others. For example, in a task requiring a robot to first reach the kitchen and then prepare dinner, the graphical structure enables the agent to verify whether it has reached the kitchen before initiating the dinner-making policy. Designing the reward structure for such approaches assumes that the human engineer knows the appropriate reward allocation for achieving the sub-goals in a specific sequence. However, this assumption becomes impractical in situations where the environment’s structure is unknown beforehand.

Moreover, some tasks, such as ensuring a robot prepares dinner only if it purchased groceries earlier, demand that the agent *encode* and *remember* information (e.g., whether groceries were bought) to perform effectively. To address this challenge, recent research has explored alternative approaches that go beyond simple reward functions by defining task goals as a graphical structures of sub-tasks using high-level specification languages like finite-trace Linear Temporal Logic (LTL_f)[116], Reward Machines (RM)[50], and SPECTRL [54].

Such representations allow agents to first focus on simpler sub-goals and progressively build toward more complex objectives. Encoding tasks graphically also facilitates a Markovian approach, where the agent tracks its episodic history as part of the state space [2].

(achieve (Key_1) or achieve (Key_2) ;
achieve ($Door$) ; achieve ($Goal$)) ensuring ($\neg Lava$)

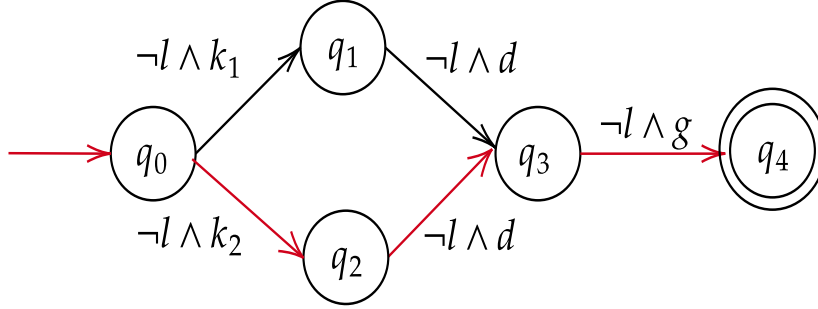


Figure 4.2: The SPECTRL formula for the task and its DAG. The path highlighted in red denotes the more promising ways to reach the goal state among the two paths. Formulas l , k_1 , k_2 , d and g correspond to $Lava$, Key_1 , Key_2 , $Door$ and $Goal$ respectively

To address the above challenge, we present two approaches aimed at improving the sample efficiency of RL agents using formal and natural language based goal-directed behavior. Our first approach, titled, Logical Specifications-guided Dynamic Task Sampling (*LSTS*) [117] uses SPECTRL specification formulas to encode the sub-goal structure for the agent. The second approach, LLM-guided Dynamic Task Sampling(*LgTS*) [118] goes one step further as it removes the dependency on the human to suggest the sub-goal structure, and queries an LLM to provide distinct ways of reaching the goal given information about the start state, goal state, objects present in the environment and the predicates available to the agent. The sub-goal structure obtained from both of the approaches is represented using a directed acyclic graphs (DAG) [54] helping the agent to encode memory. For the gridworld shown in Fig. 4.1, the DAG obtained from *LSTS* is shown in Fig. 4.2 and the graph from *LgTS* shown in Fig. 4.3.

Our key insight is to learn RL policies for sub-tasks defined using the edges of the DAG. Specifically, the agent transitions from the node q to p in the DAG when the propositional logic formula labeling the edge (q, p) evaluates to true. We use the set of propositional logic formulas labeling the outgoing edges from a given node in DAG to define sub-tasks. To learn the RL policies, we employ an adaptive Teacher-Student learning strategy, where, (1) the Teacher agent uses its high-level policy along with exploration techniques to actively sample a sub-task for the Student agent to learn. The high-level

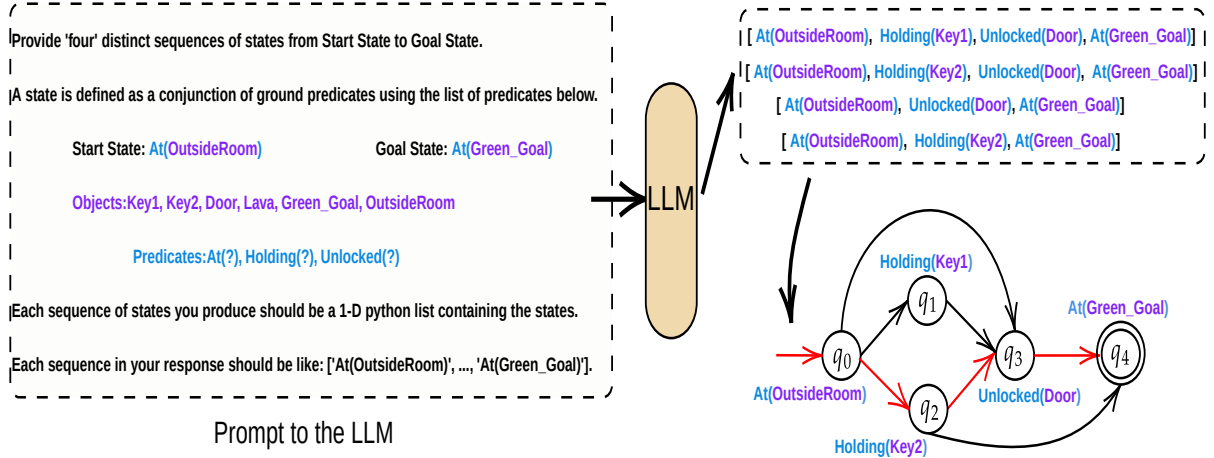


Figure 4.3: The prompt to the LLM that contains information about the number of paths n expected from the LLM and the symbolic information such as the entities, predicates and high-level initial and goal states. The LLM output is converted to a graphical representation

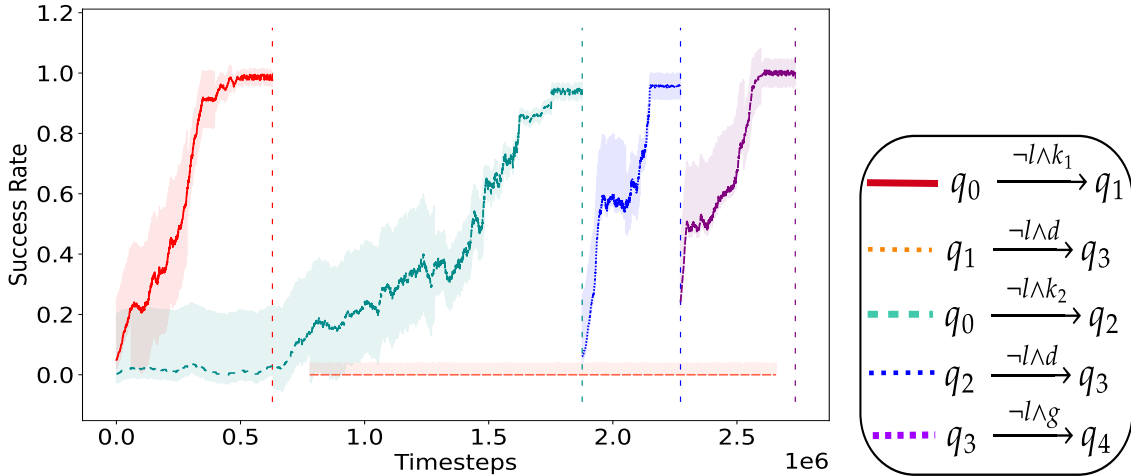


Figure 4.4: Learning Curves for each sub-task using the dynamic teacher-student sampling strategy. The sub-task sampling promotes learning promising tasks and discards unpromising tasks.

policy considers the DAG representation and the Student agent’s expected performance on all the sub-tasks, aiming to satisfy the high-level objective in the fewest number of interactions, and (2) the Student agent interacts with the environment for a few steps (much fewer than the interactions required to learn a successful policy for the sub-task) while updating its low-level RL policy for the sampled sub-task. The Teacher observes the Student’s performance on these interactions and updates its high-level policy. Steps (1) and (2) continue alternately until the Student agent learns a set of successful policies that guide the agent to reach a goal state.

Running example: As an example, let us look at the environment shown in Fig. 4.1. The goal for the agent is to collect *any* of the two *Keys*, followed by opening the *Door* and then reaching the *Goal* while avoiding the *Lava* at all times. The task’s high-level objective (ϕ) is represented using the SPECTRL formula and its corresponding DAG representation $\mathcal{G}_\phi$ in Fig. 4.2. A similar graphical structure is obtained using LLMs in LgTS approach as seen in Fig. 4.3. The DAG does not contain information about the environment configuration, such as: the optimal number of interactions required to reach *Door* from *Key*₁ are much higher compared to the interactions required to reach *Door* from *Key*₂, making the *Key*₁ to *Door* trajectory sub-optimal.

A naive approach to tackle this problem would involve learning individual RL policies for all the transitions of DAG, and then later identifying which path of the DAG leads to goal satisfaction in the fewest interactions. However, this approach will still explore all the paths of the DAG and prove sample-inefficient. It is crucial to prevent any additional interactions the agent spends in learning a policy for the sub-task defined by the edge $q_1 \xrightarrow{\neg l \wedge d} q_3$ as the path $q_0 \xrightarrow{\neg l \wedge k_1} q_1 \xrightarrow{\neg l \wedge d} q_3 \xrightarrow{\neg l \wedge g} q_4$ will always be sub-optimal. In our proposed approach *LSTS*, the Student agent begins with the *aim* of learning two distinct RL policies: π_1 for the task of visiting *Key*₁ and π_2 for the task of visiting *Key*₂, both avoiding *Lava*. The Teacher agent initially samples evenly from these two sub-tasks for the Student but later biases its sampling toward the sub-task on which the Student agent shows higher learning potential. Once the Student agent learns a successful policy for one of the sub-tasks (let’s say the learned policy π_1^* corresponding to the sub-task defined

by the transition $q_0 \xrightarrow{\neg l \wedge k_1} q_1$), the Teacher does not sample that task anymore, identifies the next task(s) for the Student using the DAG representation, and appends them to the set of tasks it is currently sampling (in this case, the only next task is: $q_1 \xrightarrow{\neg l \wedge d} q_3$) with the aim of learning a new policy for this new sub-task. Since the Student agent only has access to the state distribution over q_0 , it follows the trajectory given by π_1^* to reach a state that lies in the set of states where the proposition $\neg Lava \wedge Key_1$ holds true before commencing its learning for the policy (π_3) for $q_1 \xrightarrow{\neg l \wedge d} q_3$. If the Student agent learns the policies π_2^* for satisfying the sub-task defined by $q_0 \xrightarrow{\neg l \wedge k_2} q_2$ and π_4^* for $q_2 \xrightarrow{\neg l \wedge d} q_3$ before learning π_3 , it effectively has a set of policies to reach the node q_3 . Thus, the Teacher will now only sample the next task for the Student in the DAG representation $q_3 \xrightarrow{\neg l \wedge g} q_4$, as learning RL policies for paths that reach q_3 are effectively redundant. This process continues iteratively until the Student agent learns a set of policies that reach the goal node (q_4) from the start node (q_0). The learning curves in Fig. 4.4 empirically validate the running example. As evident from the learning curves, the Student agent learns policies for the path $q_0 \xrightarrow{\neg l \wedge k_2} q_2 \xrightarrow{\neg l \wedge d} q_3 \xrightarrow{\neg l \wedge g} q_4$ that produce trajectories to reach the goal node q_4 from the initial node q_0 , without excessively wasting interactions on the unpromising sub-task $q_1 \xrightarrow{\neg l \wedge d} q_3$. The dashed lines in Fig. 4.4 signify the interactions at which a task policy converged.

The dynamic task sampling strategy promotes *LSTS* and *LgTS* to achieve sample-efficient learning on complex tasks by identifying unpromising tasks and discarding them, saving costly interactions. Our empirical results show that our approaches reduce environmental interactions by orders of magnitude compared to Specifications-Guided RL Baseline D_{IRL}, Reward Machine-based baselines QRM [49], GSRS [14], and curriculum learning baseline TSCL [76]. We also evaluate *LSTS^{ct}*, a modified algorithm that further improves sample efficiency by continuing exploration on a new sub-task once the sub-task goal state is reached. We perform evaluation on two simulated robotic tasks and demonstrate that *LSTS* reduces the number of interactions by orders-of-magnitude when compared to automaton-guided RL baselines.

4.2 Related Work

Automaton-guided RL approaches leverage temporal logic-based language specifications to define tasks [132, 10, 141, 4]. By separating policies for sub-goals, these approaches help abstract knowledge that can be reused in novel tasks without relying on a dense reward function [49]. An alternative technique involves shaping the reward based on the distance from the accepting node in the automaton [14], though this can often lead to suboptimal reward settings. To address this, augmenting the reward function with Monte Carlo Tree Search has been proposed to mitigate the issue [135], but this requires the ability to plan ahead in the environment, which is not always feasible. Automaton-guided RL has been successfully applied to navigational exploration in robotic domains [13] and multi-agent settings [45]. Generating a curriculum based on high-level objectives [116] typically requires access to the Object-Oriented MDP [27], which is not always obtainable when environment details are unknown in advance.

DiRL (Dynamic RL) integrates high-level planning with RL to learn policies for each transition, helping to overcome issues from poor representations [55]. However, this method becomes inefficient in terms of interactions, requiring the agent to perform a fixed number of actions even when learning the task shows little promise. In contrast, our approach proposes a logical specifications-guided dynamic task sampling method that does not require access to the environment dynamics or the Reward Machine. This method efficiently samples tasks that are promising for achieving the high-level objective, thereby reducing unnecessary interactions with unpromising tasks.

Teacher-Student algorithms [76] have been explored in Curriculum Learning [85, 114] and Intrinsic Motivation literature [89]. In this framework, the Teacher proposes tasks to the Student on which the latter shows the most promise. This strategy helps the Student learn simpler tasks first and transfer knowledge to more complex tasks, ultimately reducing the number of interactions needed to learn an effective policy. While most existing approaches focus on optimizing a curriculum to learn a single policy, which is not ideal for temporally-extended tasks, we propose a Logical Specifications-guided Teacher-Student learning strategy that learns policies for promising automaton transitions, thereby pro-

moting sample-efficient training compared to the baselines.

Recent advancements in Large Language Models (LLMs)[79] have shown their potential in answering questions, reasoning, and engaging in dialogue. Several approaches have utilized LLMs for robot planning tasks, where the LLM decomposes natural language instructions into a form that can be directly fed to an artificial agent or robot for execution[12, 3, 26]. LLMs have been applied to embodied agents in tasks such as error correction during long-horizon planning [101], intrinsically shaping the reward to promote exploration [62], object rearrangement in robotics [124], and augmenting symbolic planners by providing workable symbolic plans [70]. These techniques often rely on well-engineered tools such as chain-of-thought prompting, fine-tuning datasets, and the use of a verifier (like a symbolic planner) to assess whether an LLM-generated plan is likely to succeed [122]. The absence of these techniques can significantly reduce the accuracy of LLM-generated plans. Moreover, without high-level operators to solve the plan, it becomes challenging to verify its correctness. In contrast to relying on a single LLM-generated plan, our approach queries multiple plans from an off-the-shelf LLM (without fine-tuning) that have the potential to satisfy the high-level goal objective. We then construct a graphical representation from the LLM output and employ an adaptive Teacher-Student learning algorithm to learn a set of policies that can effectively achieve the goal objective.

4.3 Theoretical Framework

Episodic Labeled Markov Decision Process

An episodic labeled Markov Decision Process (MDP) M is a tuple:

$(\mathcal{S}, \mathcal{A}, P, R, \mathcal{S}_0, \gamma, K, \mathcal{P}, L)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $P(s'|s, a)$ denotes the transition probability of reaching state $s' \in \mathcal{S}$ from $s \in \mathcal{S}$ using action $a \in \mathcal{A}$, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is the reward function, $\mathcal{S}_0$ is the initial state distribution, $\gamma \in [0, 1]$ is the discount factor, K is the maximum number of interactions in any episode, $\mathcal{P}$ is a set of predicates, and $L : \mathcal{S} \rightarrow 2^{\mathcal{P}}$ is a labeling function that maps a state $s \in \mathcal{S}$ to a

subset of predicates that are true in that state. In every interaction, the agent observes the current state s and selects an action a according to its policy function $\pi(a|s, \theta)$ with parameters θ . The MDP transitions to a new state $s' \in \mathcal{S}$ with probability $P(s' | s, a)$. The agent’s goal is to learn an *optimal policy* π^* that maximizes the discounted return $G_0 = \sum_{k=0}^K \gamma^k R(s'_k, a_k, s_k)$ until the end of the episode, which occurs after at-most K interactions.

High level specification language for *LSTS*

In our framework, we adopt the specification language SPECTRL to articulate reinforcement learning tasks [54]. A specification ϕ in SPECTRL is a logical formula applied to trajectories, determining whether a given trajectory $\zeta = (s_0, s_1, \dots)$ successfully accomplishes a desired task. Mathematically, ϕ can be depicted as a function $\phi : \mathcal{Z} \rightarrow \mathbb{B}$, where $\mathbb{B} = \{\text{TRUE}, \text{FALSE}\}$ and $\mathcal{Z}$ is the set of all trajectories.

Formally, a specification is defined over a set of *atomic predicates* $\mathcal{P}_0$. Each $p \in \mathcal{P}_0$ is associated with a function $f_p : S \rightarrow \mathbb{B}$. The agent’s MDP state s satisfies p (denoted by $s \models p$) when $f_p(s) = \text{True}$ (in other words, $p \subseteq L(s)$). A predicate $b \in \mathcal{P}$ follows the grammar $b ::= p \mid (b_1 \wedge b_2) \mid (b_1 \vee b_2)$, where $p \in \mathcal{P}_0$. Each predicate $b \in \mathcal{P}$ corresponds to a function $f_b : S \rightarrow \mathbb{B}$ defined naturally over Boolean logic. The syntax of SPECTRL specifications is given by:

$$\phi ::= \text{achieve } b \mid \phi_1 \text{ ensuring } b \mid \phi_1; \phi_2 \mid \phi_1 \text{ or } \phi_2,$$

where $b \in \mathcal{P}$. Each specification ϕ corresponds to a function $f_\phi : \mathcal{Z} \rightarrow \mathbb{B}$, and $\zeta \in \mathcal{Z}$ satisfies ϕ if $f_\phi(\zeta) := \text{True}$. Given a SPECTRL specification ϕ , we can construct an abstract graph $\mathcal{G}_\phi$ such that, for every trajectory $\zeta \in \mathcal{Z}$, we have $\zeta \models \phi$ if and only if $\zeta \models \mathcal{G}_\phi$. Thus, we can solve the reinforcement learning problem for ϕ by solving the reachability problem for $\mathcal{G}_\phi$. An *abstract graph* $\mathcal{G} = (Q, E, q_0, F, \beta, \mathcal{Z}_{safe}, \kappa)$ is a directed acyclic graph (DAG) with nodes Q , (directed) edges $E \subseteq Q \times Q$, initial node $q_0 \in Q$, final nodes $F \subseteq Q$, subgoal region map $\beta : Q \rightarrow 2^{\mathcal{S}}$ such that for each $q \in Q$, $\beta(q)$ is a subgoal region and *safe trajectories* $\mathcal{Z}_{safe} = \bigcup_{e \in E} \mathcal{Z}_{safe}^e$ where $\mathcal{Z}_{safe}^e \subseteq \mathcal{Z}_f$ denotes the

safe trajectories for edge $e \in E$. Intuitively, (Q, E) is a standard DAG, and q_0 and F define a graph reachability problem for (Q, E) . Furthermore, β and $\mathcal{Z}_{safe}$ connect (Q, E) back to the original MDP M ; in particular, for an edge $e = q \rightarrow q'$, $\mathcal{Z}_{safe}^e$ is the set of trajectories in the MDP M that can be used to transition from $\beta(q)$ to $\beta(q')$. The function κ labels each edge $e = q \rightarrow q'$ with the predicates b_e (labeled edge denoted as $e := q \rightarrow q'$). The agent transitions from q to q' when the states s_i, s_{i+j} in the agent's trajectory ζ satisfy $s_i \subseteq \beta(q)$ and $b_e \subseteq L(s_{i+j})$ and $j \geq 0$.

Prompt engineering for LLM-guided approach for *LgTS*

We assume access to symbolic information $\Sigma = \langle \mathcal{E}, \mathcal{F}, \mathcal{Q}, q_0, q_g \rangle$, where $\mathcal{E} = \{\varepsilon_1, \dots, \varepsilon_{|\mathcal{E}|}\}$ is a finite set of known environmental entities, and $\mathcal{F} = \{f_1(\odot), \dots, f_{|\mathcal{F}|}(\odot)\}$, where $\odot \subset \mathcal{E}$, is a finite set of known predicates with their negations. $\mathcal{Q} = \{q_1 \dots q_{|\mathcal{Q}|}\}$ is the set of symbolic states in the environment. A symbolic state $q \in \mathcal{Q}$ is defined as a complete assignment over all predicate-entity pairs. The starting state is given by $q_0 \in \mathcal{Q}$, and, $q_g \in \mathcal{Q}$ is the goal state. We assume access to a *single word* natural language description of the predicates and entities. We assume access to a prompt function $f_p : (\Sigma, n) \rightarrow x$ that takes in the symbolic information Σ along with the number of paths that the LLM should generate n and produces a natural language prompt x that serves as an input to the LLM. The prompt to the LLM x is designed in such a way that the output from the LLM $y = \text{LLM}_\phi(x)$ can be construed into a graph.

4.3.1 Problem Formulation

The aim of this work is to learn a set of policies π_i^* , $i = 0, \dots, n-1$, with the following properties: (i) Following π_0^* results in a trajectory in the MDP that induces a transition from q_0 to state $q_1 \in \mathcal{Q}$ in the DAG, following π_1^* results in a trajectory in MDP that induces a transition from q_1 to state $q_2 \in \mathcal{Q}$ in the DAG, and so on. (ii) The resulting path $q_0 q_1 \dots q_n$ in the DAG terminates at a final node, $q_n \in F$, with probability greater than a given threshold, $\eta \in (0, 1)$. (iii) The total number of environmental interactions spent in exploring and learning sub-task policies are minimized.

4.4 Methodology

Obtaining the DAG Structure

For *LSTS*, a human expert lists down the SPECTRL formula which gets subsequently converted to a DAG representation. For *LgTS*, given the symbolic information $\Sigma = \langle \mathcal{E}, \mathcal{F}, \mathcal{Q}, q_0, q_g \rangle$ and n , the number of ordered lists of sub-goal paths we expect from the LLM, we generate a natural language prompt $x = f_p(\Sigma, n)$. An example of a prompt is shown in Fig. 4.3. The prompt directs the LLM to produce outputs that is converted to a set of n ordered lists, where each element in the list is a high-level state, the first element of the list is initial high-level state q_0 , and the final element of the list is the goal high-level state q_g . For our work, we used LLAMA2-chat.

The next step is to convert the natural language output from the LLM y into a directed acyclic graph $\mathcal{U} = (V, E)$ such that each element $v \in V$ is a high-level state $q \in \mathcal{Q}$ and $E \subseteq V \times V$. If the output y from the LLM does not satisfy the high-level initial and goal state conditions, the LLM is re-prompted until the output y matches the correct syntax. We parse the output to get n distinct paths of reaching the high-level goal state q_g from the initial high-level state q_0 . These n distinct paths are in the form of an adjacency list for a graph. While constructing the graph, we omit self-loops and cycles, generating a directed acyclic graph (DAG) $\mathcal{G}_\phi$.

Sub-task definition:

Given the DAG $\mathcal{G}_\phi$, we define a set of sub-tasks based on the edges of the DAG. Intuitively, given any MDP state $s \in \mathcal{S}$ and a DAG node $q \in \mathcal{Q}$, a sub-task defined by an edge from node q to $p \in \mathcal{Q}$ defines a reach-avoid objective for the agent represented by the SPECTRL formula,

$$\mathbf{Task}(q, p) := \mathbf{achieve}(b_{(q,p)}) \wedge \mathbf{ensuring} \left(\bigwedge_{r \in \mathbf{Sc}(q), r \neq p} \neg b_{(q,r)} \right)$$

where $b_{(q,p)}$ is the propositional formula labeling the edge from q to p in the DAG and $\mathbf{Sc}(q)$ is the set of successors of node q in DAG. For example, in Fig. 4.2, the propositional formula labeling the edge from q_0 to q_1 is $b_{(q_0,q_1)} = \neg l \wedge k_1$. When $e = (q, p)$, we use $\mathbf{Task}(e)$ instead of $\mathbf{Task}(q, p)$ and b_e instead of $b_{(q,p)}$ for notational convenience.

Each sub-task $\mathbf{Task}(q, p)$ defines a problem to learn a policy $\pi_{(q,p)}^*$ such that, given any MDP state $s_0 \in \mathcal{S}$, following $\pi_{(q,p)}^*$ results in a trajectory $s_0 s_1 \dots s_n$ in MDP that induces the path $qq \dots qp$ in the DAG. That is, the agent’s high-level DAG state remains at q until it transitions to p . While constructing the set of sub-tasks, we omit transitions that lead to a ‘sink’ state (from which final states are unreachable).

We begin by initializing: (1) Set of: Active Tasks AT, Learned Tasks LT, Discarded Tasks DT; (2) A Dictionary Π that maps a sub-task $\mathbf{Task}(e)$ corresponding to edge e of DAG $\mathcal{G}_\phi$ to a RL policy π_e ; (3) A Dictionary representing the Teacher Q-Values Q by mapping $\mathbf{Task}(e)$ to a numerical value associated with $\mathbf{Task}(e)$.

Firstly, we convert $\mathcal{G}_\phi$ into an Adjacency Matrix $\mathcal{X}$, and find the tasks corresponding to the set all the outgoing edges $\overline{E}_{q_0} \subseteq E$ from the initial node q_0 . Satisfying the edge’s predicates $b_{(q_0,q_1)} \in \kappa(\overline{E}_{q_0})$ represent a reachability sub-task M' where each goal state $s \in \mathcal{S}_f^{M'}$ of M' satisfy the condition $b_{(q_0,q_1)} \subseteq L(s)$. The Student agent receives a positive reward for satisfying $b_{(q_0,q_1)}$ and a small negative reward at all other time steps. The state and action space, and the transition dynamics of M' are equivalent to M . To complete the sub-task, the Student agent must learn a RL policy $\pi_{(q,p)}^*$ that ensures a visit from q to p with probability greater than a predetermined threshold (η).

Algorithm 2 *LSTS and LgTS* ($\mathcal{G}_\phi, M, \eta, x$)

Output: Set of learned policies : Π^* , Edge-Policy Dictionary P

```
1: Placeholder Initialization:
2:
3: Sets of: Active Tasks ( $\mathcal{AT}$ )  $\leftarrow \emptyset$ ;
4: Learned Tasks ( $\mathcal{LT}$ )  $\leftarrow \emptyset$ ; Discarded Tasks ( $\mathcal{DT}$ )  $\leftarrow \emptyset$ 
5:
6: Edge-Policy Dictionary  $P : \text{Task}(e) \rightarrow \pi$ 
7: Teacher Q-Value Dictionary:  $Q : \text{Task}(e) \rightarrow -\infty$ 
8: Algorithm:
9:  $\mathcal{X} \leftarrow \text{Adjacency\_Matrix}(\mathcal{U})$ 
10:  $\mathcal{AT} \leftarrow \mathcal{AT} \cup \{\mathcal{X}[q_0]\}$ 
11:  $\forall \text{Task}(e) \in \mathcal{AT} : Q[e] = 0$ 
12: while True do
13:    $e \leftarrow \text{Sample}(Q)$ 
14:    $P[e], g \leftarrow \text{Learn}(M, \mathcal{U}, e, x, P)$ 
15:    $\text{Update\_Teacher}(Q, e, g)$ 
16:   if  $\text{Convergence}(Q, e, g, \eta)$  then
17:      $\Pi^* \leftarrow \Pi^* \cup P[e]$  ;  $\mathcal{LT} \leftarrow \mathcal{LT} \cup \{\text{Task}(e)\}$  ;  $\mathcal{AT} \leftarrow \mathcal{AT} \setminus \{\text{Task}(e)\}$ 
18:      $E_{\mathcal{DT}} \leftarrow \text{Discarded\_Tasks}(\mathcal{X}, e)$ 
19:      $\mathcal{DT} \leftarrow \mathcal{DT} \cup E_{\mathcal{DT}}$ 
20:      $E_{\mathcal{AT}} \leftarrow \text{Next\_Tasks}(\mathcal{X}, e, \mathcal{DT})$ 
21:     if  $|E_{\mathcal{AT}}| = 0$  then
22:       break
23:     end if
24:      $\forall \text{Task}(\bar{e}) \in E_{\mathcal{AT}} : Q[\bar{e}] = 0$ 
25:      $\mathcal{AT} \leftarrow \mathcal{AT} \cup E_{\mathcal{AT}}$ 
26:   end if
27: end while
28: return  $\Pi^*, P$ 
```

Teacher-Student learning:

We set the Teacher Q-Values for all the sub-tasks corresponding to edges in **AT** (i.e., tasks corresponding to $\overline{E}_{q_0}$) to zero. We formalize the Teacher’s goal of choosing the most promising sub-task as a *Partially Observable MDP*, where the Teacher does not have access to the entire Student agent state but only to the Student agent’s performance on a sub-task (e.g. success rate or average returns), and as an action, chooses a sub-task $\text{Task}(e) \in \text{AT}$ the Student agent should train on next. In this POMDP setting, each Teacher action has an Q-Value associated with it. Intuitively, higher Q-Values correspond to tasks on which the Student agent is more successful, and the Teacher should sample such tasks at a higher frequency to satisfy ϕ (reach a goal node) in fewest overall interactions.

(A) Given the Teacher Q-Values, we sample a sub-task $\text{Task}(e) \in \text{AT}$ using the ϵ -greedy exploration strategy, and (B) The Student agent trains on task $\text{Task}(e)$ using the RL policy $\Pi[e]$ for ‘ x ’ interactions. In one Teacher timestep, the Student trains for x environmental interactions. Here, $x \ll$ total number of environmental interactions required by the Student agent to learn a successful RL policy for $\text{Task}(e)$, since the aim is to keep switching to a sub-task that shows highest promise. (C) The Teacher observes the Student agent’s average return g_t on these x interactions, and updates its Q-Value for $\text{Task}(e)$ as: $Q[e] \leftarrow \alpha(g_t) + (1 - \alpha)Q[e]$: where α is the Teacher learning rate, g_t is the average Student agent return on $\text{Task}(e)$ at the Teacher timestep t . As the learning advances, g_t increases as t increases, and hence we use a constant parameter α to tackle the non-stationary problem of a moving return distribution. Steps (A), (B), (C) continue successively until the policy for *any* $\text{Task}(e) \in \text{AT}$ task converges.

Sub-task convergence criteria:

We define Student agent’s RL policy for $\text{Task}(q, p)$ to be converged if a trajectory ζ produced by the policy triggers the transition with probability $\Pr_{\zeta \in \mathcal{Z}}[\zeta \text{ satisfies } \text{Task}(q, p)] \geq \eta$ and $\Delta(g_t, g_{t-1}) < \tau$ where η is the expected performance and τ is a small numerical value. Once a policy for the $\text{Task}(q, p)$ converges, we append $\text{Task}(q, p)$ to the set of

Learned Tasks LT and remove it from the set of Active Tasks AT. In order to ensure that the learned sub-task does not get sampled any further, we set the Teacher Q-value for this sub-task to $-\infty$.

Intuitively, a converged policy attains an average success rate $\geq \eta$ and has not improved further by maintaining constant average returns. Like all other RM and automaton-based approaches, we assume access to the labeling function L to examine if the trajectory ζ satisfies the formula $b_{(q,p)}$ by checking if the final environmental state s of the trajectory satisfies the condition $b_{(q,p)} \subseteq L(s)$. The sub-goal regions need not be disjoint, i.e., the same state s can satisfy propositions for multiple DAG nodes.

Discarding unpromising sub-tasks:

Once we have a successful policy for the $\text{Task}(q, p)$ (the transition $q \xrightarrow{b_{(q,p)}} p$), we determine the sub-tasks that can be discarded. We find the sub-tasks corresponding to edges that: (1) lie before p in a path from q_0 to any $q \in F$, and, (2) do not lie in a path to $q \in F$ that does not contain p . Intuitively, if we already have a set of policies that can generate a successful trajectory to reach the node p , we do not need to learn policies for sub-tasks that ultimately lead *only* to p (e.g., in Fig. 4.2 if we have successful policies for $\text{Task}(q_0, q_2)$ and $\text{Task}(q_2, q_3)$, we can discard $\text{Task}(q_0, q_1)$ and $\text{Task}(q_1, q_3)$). We add all such sub-tasks to the set of Discarded Tasks DT, and set the Teacher Q-values for all the discarded tasks to $-\infty$ to prevent them from being sampled for the Student agent.

As an extension, in the limit, an optimal policy can be found by not completely discarding such sub-tasks, but rather biasing away from them so that they would still be explored.

Traversing in the DAG until ϕ satisfied:

Subsequently, we determine the next set of tasks $\text{Tasks}(\overline{E}_{AT})$ in the DAG to add to the AT set. This is calculated by identifying sub-tasks corresponding to all the outgoing edges from p . Since the edge $e_{q,p}$ corresponds to the transition $q \xrightarrow{b_{(q,p)}} p$, we have a successful policy that can produce a trajectory that reaches a state where $b_{(q,p)}$ hold true,

and $\text{Tasks}(\overline{E}_{AT})$ corresponds to $\mathcal{X}[p] \setminus \text{DT}$ i.e., sub-tasks corresponding to all the outgoing edges from p that do not lie in the DT set and set the Teacher Q-values for all these tasks to 0. The above steps go on iteratively until we reach a final node $q \in F$, i.e, $|\text{Tasks}(\overline{E}_{AT})| = \emptyset$. Thus, we **break** from the **while** loop and return the set of learned policies Π^* , and edge-policy dictionary Π . From Π and Π^* , we get an ordered list of policies $\Pi_{list}^* = [\pi_{(q_1, q_2)}, \dots, \pi_{(q_{k-1}, q_k)}]$ such that sequentially following $\pi \in \Pi_{list}^*$ generates trajectories that satisfy the objective ϕ .

4.5 Convergence criteria for a trajectory to satisfy the high-level objective in LSTS

Given the ordered list of policies Π_{list}^* , we can generate a trajectory ζ in the task M with $\Pr_{\zeta \in \mathcal{Z}}[\zeta \text{ satisfies } \phi] \geq \eta$. A trajectory ζ is defined as a sequence of state sequences $(s_0, s_1, \dots)$ with transitions defined using MDP transition function P .

We define that a trajectory ζ satisfies the high-level objective ϕ if the initial state of agent in the trajectory s_0 satisfies the propositions corresponding to the node q_0 of the DAG $\mathcal{G}_\phi$ that represents ϕ , and the agent reaches a state s_f that satisfies the propositions corresponding to a node that lies in the set of accepting nodes $q_f \in F$ of the DAG $\mathcal{G}_\phi$.

$\Pr_{\zeta \in \mathcal{Z}}[\zeta \text{ satisfies } \phi]$ denotes the probability with which a trajectory $\zeta \in \mathcal{Z}$ will satisfy ϕ . The set of all trajectories $\mathcal{Z}$ are generated by the agent starting in a state $s \sim \mathcal{S}_0$ and following trajectories given by the policies in Π_{list}^* .

Since $\Pi_{list}^* = [\pi_{(q_1, q_2)}, \pi_{(q_2, q_3)}, \dots, \pi_{(q_{f-1}, q_f)}]$, and the list of MDP states visited by the trajectory ζ is given by $\zeta_s : (s_0, s_1, \dots)$.

The trajectory ζ satisfies the DAG $\mathcal{G}_\phi$ if for the sequence of states in the trajectory $\zeta_s : (\zeta_s[0], \dots, \zeta_s[k_1], \dots, \zeta_s[k_2], \dots, \dots, \zeta_s[k_f])$:

- $\zeta_s[0] \sim \mathcal{S}_0$
- $\zeta_0 \xrightarrow{L(\zeta_s[k_1])} \zeta_1 \xrightarrow{L(\zeta_s[k_2])} \zeta_2, \dots, \zeta_{f-1} \xrightarrow{L(\zeta_s[k_f])} \zeta_f$
- $L(\zeta_s[k_f]) \in F$

Thus, ζ_s contains states $\zeta_s[k_i]$ that triggers transitions of the agent's high-level state in the DAG $\mathcal{G}_\phi$ from the node q_i to the node q_j . Additionally, the initial state of the agent $\zeta_s[0]$ lies in the set of starting states $\mathcal{S}_0$ of the MDP M corresponding to the node q_0 and final state of the agent $\zeta_s[k_f]$ lies in the set of final states $\mathcal{S}_f$ of the MDP M corresponding to the node $q_f \in F$.

As defined in Section 4, the *convergence criteria* for a task $\mathbf{Task}(q_i, q_{i+1})$ ensures that the successful policy $\pi_{(q_i, q_{i+1})} \in \Pi_{list}^*$ achieves $\Pr_{\zeta \in \mathcal{Z}}[\zeta \text{ satisfies } \mathbf{Task}(q_i, q_{i+1})] \geq \eta$ for all $i \in 1, \dots, f-1$.

Thus, convergence criteria for $\pi_{(q_0, q_1)}$ ensures that:

$$\Pr_{\zeta \in \mathcal{Z}}[\zeta_s[0 : i] \text{ satisfies } \mathbf{Task}(q_0, q_1)] \geq \eta$$

Next, the convergence criteria for $\pi_{(q_1, q_2)}$ ensures that:

$$\Pr_{\zeta \in \mathcal{Z}}[\zeta_s[0 : i] \text{ satisfies } \mathbf{Task}(q_0, q_1) \wedge \zeta_s[i : i+j] \text{ satisfies } \mathbf{Task}(q_1, q_2)] \geq \eta \quad (4.1)$$

Following this procedure inductively. the convergence criteria for $\pi_{q_{f-1}, \sigma_{f-1}}$ ensures:

$$\Pr_{\zeta \in \mathcal{Z}}[\zeta_s[0 : i] \text{ satisfies } \mathbf{Task}(q_0, q_1) \wedge \zeta_s[i : f] \text{ satisfies } \mathbf{Task}(q_1, q_2), \dots, \mathbf{Task}(q_{f-1}, q_f)] \geq \eta \quad (4.2)$$

We also know that:

$$\Pr_{\zeta \in \mathcal{Z}}[\zeta \text{ satisfies } \mathcal{G}_\phi] := \Pr_{\zeta \in \mathcal{Z}}[\zeta_s[0 : i] \text{ satisfies } \mathbf{Task}(q_0, q_1) \wedge \zeta_s[i : k] \text{ satisfies } \mathbf{Task}(q_1, q_2), \dots, \mathbf{Task}(q_{f-1}, q_f)] \quad (4.3)$$

Combining the above two equations:

$$\Pr_{\zeta \in \mathcal{Z}}[\zeta \text{ satisfies } \mathcal{G}_\phi] \geq \eta$$

Thus,

$$\Pr_{\zeta \in \mathcal{Z}}[\zeta \text{ satisfies } \phi] \geq \eta$$

4.6 Experiments - LgTS

We aim to answer the following questions: (Q1) Does *LgTS* yield sample efficient learning compared to other baseline approaches? (Q2) How does *LgTS* perform when distractor objects are present in the environment that are not essential for satisfying the high-level goal? (Q3) Does *LgTS* yield sample efficient learning even when the prompt to the LLM is modified by using synonyms for objects and predicates? (Q4) How does *LgTS* scale when the environment is complex and the optimal plan is longer than the DoorKey task?

4.6.1 LgTS - DoorKey Domain

To answer Q1, we evaluated *LgTS* on a Minigrid [16] inspired domain. The environment configuration is shown in Fig. 4.1. Essentially, the agent needs to collect *any* of the two available *Keys* before heading to the *Door*. After *toggling* the *Door* open, the agent needs to visit the *Green_Goal*. At all times, the agent needs to avoid the *Lava* object. We assume an episodic setting where an episode ends if the agent touches the *Lava* object, reaches the *Green_Goal* or exhausts the number of allocated interactions.

This is a complex sequential decision making problem for a reinforcement learning agent as the agent needs to perform a series of correct actions to satisfy the high-level objective, which is to navigate to any of the two keys, pick a key and then unlock the door. Then, navigate to reach the green-goal state. In this environment, the agent has access to three navigation actions: *move forward*, *rotate left* and *rotate right*. The agent can also perform: *pick-up* action, which adds a *Key* to the agent’s inventory if it is facing the *Key*, *drop* places the *Key* in the next grid if *Key* is present in the inventory, and, *toggle* that toggles the *Door* (closed $\leftrightarrow$ open) only if the agent is holding the *Key*. The agent can hold only one *Key* in its inventory. Hence, it needs to perform the *drop* action to drop a key present in its inventory before picking up another key. For this environment, we assume a fully-observable setting where the environmental state is a low-level encoding of the image. For each cell in the grid, the low-level encoding returns an integer that

describes the item occupying the grid, along with additional information, if any (e.g., the *Door* state can be open or closed).

The prompt to the LLM contains information about the high-level start state $At(OutsideRoom)$, the high-level goal state $At(Green_Goal)$, the entities present in the environment - Key_1 , Key_2 , $Door$, $OutsideRoom$, $Green_Goal$, $Lava$, the predicates that the agent can detect through its sensors - $Holding(?)$, $At(?)$, $Unlocked(?)$, and a hyperparameter that defines the number of feasible high-level sequences given by the LLM n . We performed grid-search to find the value of n . For our experiments, $n = 4$.

with the LTL_f objective:

$$\varphi_f^{gridworld} := \mathbf{G} \neg Lava \wedge \mathbf{F}((Key_1|Key_2) \wedge \mathbf{F}(Door \ \& \ \mathbf{F}(Goal))) \quad (4.4)$$

For the RL pipeline, we use PPO [109], which works for discrete and continuous action spaces. We consider a standard actor-critic architecture with 2 convolutional layers followed by 2 fully connected layers. For *LgTS*, the reward function is sparse. The agent gets a reward of $(1 - 0.9 \frac{(interactions\ taken)}{(interactions\ allocated)})$ if it achieves the goal in the sub-task, and a reward of 0 otherwise. For individual tasks, $interactions\ allocated = 100$. The agent does not receive any negative rewards for hitting the *Lava*.

Baselines:

We compare our *LgTS* method against four baseline approaches and an oracle approach.

1. Learning from scratch (LFS) where the goal for the agent is to learn a single policy that learns to satisfy the final high-level goal state using RL without any shaped reward.
2. Teacher-student curriculum learning (TSCL) approach where the Teacher agent samples most promising task (based on average success rate) without the use of any graphical structure to guide the learning progress of the agent. In our experiments,

Approach	# Interactions (Mean $\pm$ SD)	Success Rate (Mean $\pm$ SD)
<i>LgTS</i>	$(3.98 \pm 0.42) \times 10^6$	0.92 ± 0.03
<i>AgTS</i>	$(2.67 \pm 0.36) \times 10^6$	0.94 ± 0.02
<i>LFS</i>	5×10^7	0 ± 0
AgRS	5×10^7	0.05 ± 0.04
LgRS	5×10^7	0 ± 0
TSCL	5×10^7	0 ± 0

Table 4.1: Table comparing #interactions & success rate for the DoorKey domain. Our proposed approach, LgTS (highlighted in grey) outperforms baselines and is comparable to the oracle approach- AgTS. Other approaches are not able to learn a set of successful policies because the task is either too complex or the approach falls into a local optima, preventing the agent from learning policies to reach the goal state.

the set of tasks included every feasible transition in the automaton description of the task.

3. **(Oracle approach)** Automaton-guided Teacher-Student learning (AgTS) where the graphical structure of the sub-goal is generated using the finite-trace Linear Temporal Logic (LTL_f) formula given by an oracle. For this task, the LTL_f formula is: $\varphi_f := \mathbf{G}\neg Lava \wedge \mathbf{F}((Key_1|Key_2) \wedge \mathbf{F}(Door \& \mathbf{F}(Goal)))$ where G and F represent *Always* and *Eventually* respectively. We use the equivalent DFA representation of the above LTL_f formula as the graphical representation, and perform the Teacher-Student learning approach outlined in section 4.
4. Automaton-guided Reward Shaping (AGRS) where the DFA representation of the LTL_f formula is used as a reward shaping mechanism to guide the agent toward the final high-level goal state. The reward given to the agent is proportional to the distance from the goal node.
5. LLM-guided Reward Shaping (LgRS) where the graph generated from the n high-level sub-goal sequences is used as a reward shaping mechanism to guide the agent toward the final high-level goal state. The reward given to the agent is proportional to the distance from the goal node.

The results in Table 4.1 (averaged over 10 trials) show that *LgTS* reaches a successful

policy quicker compared to the learning from scratch, teacher-student curriculum learning, and LLM-guided reward shaping baseline approaches. We observe that the number of environmental interactions taken by our proposed approach are comparable to the automaton-guided teacher student (AgTS) algorithm where the ground truth graph is in the form of an automaton, and the graph is provided by an oracle. Several of the other baseline approaches such as LFS, TSCL, LgRS, AgRS fail to learn a successful policy for reaching the high-level goal state demonstrating that approaches that tend to learn a single policy for the entire objective are unable to satisfy the goal condition. Reward shaping fails as agent greedily favors reaching the high level state q_1 over q_2 and is unable to reach node q_3 from q_1 .

We evaluated the average graph edit distance (GED) between the graphs generated using the *LgTS* and the AgTS approach. The GED is the number of edge/node changes needed to make two graphs isomorphic. We observed an average graph edit distance of 2.1 ± 0.2 . This indicates that the graph generated by the oracle through AgTS, which has five nodes and five edges, can be converted to the graph generated by LgTS by performing ~ 2.1 changes.

4.6.2 LgTS with distractor entities

To answer Q2, we evaluated LgTS on a task environment that contains entities that do not affect the optimal path for reaching the high-level goal state. During each run, the environment contains 1 – 3 instances of distractor objects that are modeled in the LLM prompt and in the environment dynamics. For our experiment, the distractor items are household kitchen items such as apple, plate, fruit etc. Since the optimal path or the task solution has not changed, the paths suggested by the LLM should ignore the distractor objects.

The results in Table 4.2 (averaged over 10 trials) show that *LgTS* reaches a successful policy quicker compared to the LFS, TSCL, and LgRS. The overall number of interactions to learn a set of successful policies for satisfying the high-level goal objective are higher in presence of distractor objects because of low level agent interactions with these objects

Approach	# Interactions (Mean $\pm$ SD)	Success Rate (Mean $\pm$ SD)
<i>LgTS</i>	$(4.64 \pm 1.72) \times 10^6$	0.84 ± 0.08
<i>AgTS</i>	$(3.21 \pm 0.57) \times 10^6$	0.90 ± 0.04
LFS	5×10^7	0 ± 0
AgRS	5×10^7	0 ± 0
LgRS	5×10^7	0 ± 0
TSCL	5×10^7	0 ± 0

Table 4.2: Table comparing #interactions & success rate for the DoorKey domain with distractor objects. Our proposed approach, LgTS (highlighted in grey) outperforms base-lines and is comparable to the oracle approach- AgTS. Other approaches are not able to learn a set of successful policies because the task is either too complex or the approach falls into a local optima, preventing the agent from learning policies to reach the goal state.

and the increased dimensionality of the state space of the RL agent. For the experiment with distractor objects, we observed a graph edit distance of 3.4 ± 0.4 between the LgTS approach and the graph generated using the AgTS approach, which is higher than the graph edit distance that was computed without the presence of distractor objects. This difference indicates that the graphs generated using the LgTS approach did contain paths that involved distractor objects, however, the graph also contained paths that did not involve the distractor object and the RL agent was able to learn successful policies for such paths.

4.6.3 LgTS - modified prompt

Recent approaches that use LLM for task guidance have a curated prompt and a fine-tuned LLM that prevents generalization to newer prompts that have similar meaning but different descriptors. This finetuning prevents generalization to unseen out-of-distribution prompts and descriptors. To demonstrate how the prompt influences the LLM output which in turn affects learning progress, we evaluated LgTS by changing the prompt to the LLM. In this test, a fraction (at random) of entity and predicate descriptors were changed to a synonym chosen from Thesaurus (for e.g., Key was replaced with Code; Door was replaced with Gate).

The results in Table 4.3 (averaged over 10 trials) show that *LgTS* reaches a successful

Approach	# Interactions (Mean $\pm$ SD)	Success Rate (Mean $\pm$ SD)
<i>LgTS</i>	$(4.87 \pm 0.74) \times 10^6$	0.81 ± 0.09
<i>AgTS</i>	$(2.67 \pm 0.36) \times 10^6$	0.94 ± 0.02
<i>LFS</i>	5×10^7	0 ± 0
AgRS	5×10^7	0.05 ± 0.04
LgRS	5×10^7	0 ± 0
TSCL	5×10^7	0 ± 0

Table 4.3: Table comparing #interactions & success rate for the DoorKey domain with modified prompt. Our proposed approach, LgTS (highlighted in grey) outperforms baselines and is comparable to the oracle approach- AgTS.

policy quicker compared to the LFS, TSCL, and LgRS. The overall number of interactions to learn a set of policies that satisfy the high-level goal objective are higher when the prompt was changed as compared to LgTS with a constant and curated prompt. We observed that the LLM was able to accommodate the new prompt and suggest paths that satisfied the high-level objective.

4.6.4 LgTS - Search and Rescue task

To demonstrate how LgTS performs when the plan length becomes deeper, we evaluated LgTS on a more complex urban Search and Rescue domain. In this domain, the agent acts in a grid setting where it needs to perform a series of sequential sub-tasks to accomplish the final goal of the task. The agent needs to open a door using a key, then collect a fire extinguisher to extinguish the fire, and then find and rescue stranded survivors. A fully-connected graph generated using the above mentioned high-level states consists of 24 distinct transitions. This is a multi-goal task as the agent needs to extinguish fire as well as rescue survivors to reach the goal state. We use the LLM to produce seven distinct high-level paths that help prune transitions that the LLM does not recommend while providing little information about the environment as possible.

The results in Table 4.4 (averaged over 10 trials) show that *LgTS* reaches a successful policy quicker compared to the LFS, TSCL, and LgRS. The overall number of interactions to learn a set of successful policies for satisfying the high-level goal objective are higher when the prompt was changed as compared to LgTS with a constant and curated prompt.

Approach	# Interactions (Mean $\pm$ SD)	Success Rate (Mean $\pm$ SD)
<i>LgTS</i>	$(1.13 \pm 0.26) \times 10^7$	0.76 ± 0.11
<i>AgTS</i>	$(8.61 \pm 0.12) \times 10^6$	0.87 ± 0.04
<i>LFS</i>	5×10^6	0 ± 0
AgRS	5×10^7	0.05 ± 0.04
LgRS	5×10^7	0 ± 0
TSCL	5×10^7	0 ± 0

Table 4.4: Table comparing #interactions & success rate for the Search and Rescue domain. Our proposed approach, LgTS (highlighted in grey) outperforms baselines and is comparable to the oracle approach- AgTS. Other approaches are not able to learn a set of successful policies because the task is either too complex or the approach falls into a local optima, preventing the agent from learning policies to reach the goal state.

The LLM was able to accommodate the new prompt and suggest paths that satisfied the high-level goal objective.

4.7 Experiments - LSTS

We aim to answer the following questions: (Q1) Does *LSTS* yield sample efficient learning compared to state-of-the-art baselines? (Q2) After reaching a sub-task goal state, can we sample a new sub-task to continue training and improve sample efficiency? (Q3) Does *LSTS* yield sample efficient learning for complex robotic tasks with partially observable or continuous control settings? (Q4) How does *LSTS* scale to complex search-and-rescue scenarios?

4.7.1 LSTS - Gridworld Domain

To answer (Q1), we evaluated *LSTS* on a Minigrid [16] inspired domain with the SPECTRL objective:

$$\phi_f^{grid} := (\text{achieve}(k_1) \text{ or } \text{achieve}(k_2);$$

$$\text{achieve}(d); \text{achieve}(g)) \text{ensuring}(\neg l) \quad (4.5)$$

Approach	# Interactions (Mean $\pm$ SD)	Success Rate (Mean $\pm$ SD)
<i>LSTS</i>	$(2.72 \pm 0.31) \times 10^6$	0.96 ± 0.02
<i>LSTS^{ct}</i>	$(2.45 \pm 0.25) \times 10^6$	0.95 ± 0.01
DiRL ^c	$(4.06 \pm 0.37) \times 10^6$	0.95 ± 0.03
DiRL	$(5.47 \pm 0.40) \times 10^6$	0.94 ± 0.01
QRM	5×10^7	0.05 ± 0.04
GSRS	5×10^7	0 ± 0
TSCL	5×10^7	0 ± 0
LFS	5×10^7	0 ± 0

Table 4.5: Table comparing #interactions & success rate.
LSTS (highlighted) outperformed all baselines

where k_1, k_2, d, g, l represent the atomic propositions *Key*₁, *Key*₂, *Door*, *Goal*, *Lava* respectively. The environment and its ϕ are given in Fig. 6.1. The agent needs to *pick-up* any of the *Keys* before heading to the *Door*. After *toggling* the *Door* open, the agent needs to reach the *Goal*. At all times, the agent needs to avoid the *Lava*. For our experiments, we assume an episodic setting where an episode ends if the agent touches the *Lava* object, reaches the *Goal* or exhausts the number of allocated interactions.

The problem is complex as the agent needs to perform a series of high-level correct actions to satisfy ϕ_f^{grid} . The action space of the agent consists of three navigational actions: *move forward*, *rotate left* and *rotate right*. The agent can also perform: *pick-up* action that adds a *Key* to the agent’s inventory if the agent is facing the *Key*, *drop* places the *Key* in the next grid if grid is empty and *Key* is present in the inventory, and, a *toggle* action for the *Door* (closed $\leftrightarrow$ open) only if the agent is holding the *Key*. In this experiment, we assume a fully-observable setting where the environmental state is an image-like low-level encoding of the state. For each cell in the grid, the encoding returns an integer that describes the item occupying the grid, along with any additional information (e.g., the *Door* can be open or closed).

We employ PPO [109] for the Student RL agent, which works for discrete and continuous action spaces. We consider a standard actor-critic architecture with 2 convolutional layers followed by 2 fully connected layers. For *LSTS*, the reward function is sparse. The agent gets a reward of $(1 - 0.9 \frac{(\text{interactions taken})}{(\text{interactions allocated})})$ (where *interactions allocated* = 100)

if it achieves the goal in the sub-task, and a reward of 0 otherwise.

Baselines: We compare our *LSTS* method with six baseline approaches: learning from scratch (LFS), Reward Machine-based (RM) baselines: GSRS [14], QRM [49]; and Compositional RL from Logical Specifications (DiRL) [55]. All the baselines are implemented using the RL algorithm (PPO), described above. GSRS assigns reward inversely proportional to the distance from the RM goal node. QRM employs a separate Q-function for each node in the RM, and DiRL uses Dijkstra’s algorithm (edge cost is the average RL policy success rate) to guide the agent in choosing a path from the specification graph. For the fifth baseline, we modify DiRL such that instead of manually specifying a limit on the number of interactions, which needs to be fine-tuned to suit the task, we stop learning a sub-task once it has reached the convergence criteria defined in Section 4.4. We call this modified baseline as DiRL^c . The sixth baseline (TSCL [76]) follows a curriculum learning strategy where the Teacher samples most promising task without the use of any automaton to guide the learning progress of the agent.

The results in Table 4.5 (averaged over 10 trials) show that *LSTS* reaches a successful policy quicker compared to all baselines. $LSTS^{ct}$ is a modified version of *LSTS* and is described below. Our custom baseline, DiRL^c is more sample-efficient than DiRL, and both outperform other baselines, which do not learn a meaningful policy. We performed an unpaired t-test [59] to compare *LSTS* against the best performing baselines at the end of 10^7 training steps and we observed statistically significant results (95% confidence). Thus, *LSTS* not only achieves a better success rate, but also converges faster. Time-to-threshold metric is defined as the difference in number of interactions between two approaches to reach a desired performance [85]. The time-to-threshold between *LSTS* and the best-performing baseline DiRL^c is 1.34×10^6 interactions for 95% success rate.

Statistical Significance

To demonstrate that the average convergence rate of *LSTS* is consistently higher than the baseline approaches, we perform an unpaired t-test [59]. For the experiment, we consider

a confidence interval of 95% and evaluate the p-value between *LSTS* approach and the best-performing baseline approach by comparing the success rates of the two approaches on 100 episodes after training for 10^7 interactions. Thus, through the results, we see that our proposed approach, *LSTS* has a consistent performance in the convergence rate. The results are always statistically significant. In all the experiments, *LSTS* has a much more sample-efficient performance. Thus, *LSTS* not only achieves a better success rate, but also adapts to converges quicker.

4.7.2 (LSTS + Cont. Training) - Gridworld Domain

In *LSTS*, while learning a policy for $\text{Task}(q, p)$, we reinitialize the environment to a random initial environmental state $s \sim \mathcal{S}_0$ once the agent reaches a state where the propositions ($b_{(q,p)}$) hold true. To answer the question Q2, instead of resetting the environment after reaching such a state where $b_{(q,p)}$ hold true, we let the Teacher agent sample a task (let's say $\text{Task}(p, r)$) from the set $\mathcal{X}[p] \setminus \text{DT}$, where $\mathcal{X}$ is the adjacency matrix for the graph, and DT is the set of Discarded Tasks, as defined in Algo. 3. This helps the agent continue its training by attempting to learn a policy $\pi_{(p,r)}$ for $\text{Task}(p, r)$ while simultaneously learning a separate policy $\pi_{(q,p)}$ for $\text{Task}(q, p)$. If the agent fails to satisfy $\text{Task}(p, r)$, we reset the environment to state $s \sim \mathcal{S}_0$. Otherwise, the agent continues its training until its trajectory satisfies the high-level objective ϕ . We call this approach *LSTS^{ct}*. Results in Table 4.5 demonstrate that this approach improves sample efficiency by reducing the number of interactions required to learn a successful policy for the gridworld task, with a time-to-threshold metric of 2.7×10^5 interactions as compared to *LSTS*.

LSTS - Continued Training - Detailed Algorithm

A detailed algorithm for *LSTS^{ct}* is described in Algo. 3. While the overall algorithm is similar to *LSTS* (described in Section 4), *LSTS^{ct}* differs from *LSTS* in lines 9-29. In *LSTS*, while learning a policy for the task $\text{Task}(q, p)$, which corresponds to the transition $q \xrightarrow{\phi_{(q,p)}} p$, we reinitialize the environment to a state $s \in \mathcal{S}_0$ once the agent reaches a state

Algorithm 3 $LSTS^{ct} (\mathcal{G}_\phi, M, \eta, x)$

Output: Set of learned policies : Π^* , Edge-Policy Dictionary $\mathcal{P}$

```
1: Set of Active Tasks:  $\text{AT} \leftarrow \emptyset$ ; Learned Tasks:  $\text{LT} \leftarrow \emptyset$ ; Discarded Tasks:  $\text{DT} \leftarrow \emptyset$ 
2: Edge-Policy Dictionary  $\mathcal{P} : e \rightarrow \pi$ 
3: Teacher Q-Value Dictionary:  $Q : e \rightarrow -\infty$  Initialize all Teacher Q-Values to  $-\infty$ 
4:  $\mathcal{X} \leftarrow \text{Adjacency\_Matrix}(\mathcal{G}_\phi)$  Get the adjacency matrix from the DAG representation
5:  $\text{AT} \leftarrow \text{AT} \cup \{\mathcal{X}[q_0]\}$  Add all outgoing edges from  $q_0$  to AT
6:  $Q[e] = 0 \forall e \in \text{AT}$  Initialize Q-values to 0 for all tasks from the initial node  $q_0$ 
7: while True do
8:    $e \leftarrow \text{Sample}(Q, \text{AT})$ ;  $\bar{g} = []$  Sample a task from AT based on Teacher Q-Values
9:   for  $y \in x$  do
10:     $\text{task\_outcome}, \mathcal{P}[e], g \leftarrow \text{Learn\_Episode}(M, \mathcal{G}_\phi, e, \mathcal{P})$  Learn task  $e$  for 1 episode
11:     $\bar{g}.\text{append}(g)$ ;  $\bar{E}_{AT} \leftarrow \text{Next\_Tasks}(\mathcal{X}, e, \text{DT})$ 
12:    if  $\text{task\_outcome} = \text{Success}$  and  $|\bar{E}_{AT}| \neq 0$  then
13:      while True do
14:         $\tilde{e} \leftarrow \text{Sample}(Q, \bar{E}_{AT})$  Sample a task from  $\bar{E}_{AT}$ 
15:         $\text{task\_outcome}, \mathcal{P}[\tilde{e}], g \leftarrow \text{Learn\_Episode}(M, \mathcal{G}_\phi, \tilde{e}, \mathcal{P})$ 
16:         $\text{Update\_Teacher}(Q, \tilde{e}, g)$  Update the Teacher Q-Values
17:         $\bar{E}_{AT} \leftarrow \text{Next\_Tasks}(\mathcal{X}, \tilde{e}, \text{DT})$ 
18:        if  $\text{task\_outcome} = \text{Success}$  and  $|\bar{E}_{AT}| \neq 0$  then
19:          continue
20:        else
21:          break
22:        end if
23:      end while
24:    end if
25:  end for
26:   $\text{Update\_Teacher}(Q, e, \text{mean}(\bar{g}))$  Update the Teacher Q-Values
27:  if  $\text{Convergence}(Q, e, g, \eta)$  then
28:     $\Pi^* \leftarrow \Pi^* \cup \mathcal{P}[e]$ ;  $\text{LT} \leftarrow \text{LT} \cup \{e\}$ ;  $\text{AT} \leftarrow \text{AT} \setminus \{e\}$  Update the sets.
29:     $\bar{E}_{DT} \leftarrow \text{Discarded\_Tasks}(\mathcal{X}, e)$  Find any tasks that need to be discarded
30:     $\text{DT} \leftarrow \text{DT} \cup \bar{E}_{DT}$  Add the identified tasks that need to be discarded to DT
31:     $\bar{E}_{AT} \leftarrow \text{Next\_Tasks}(\mathcal{X}, e, \text{DT})$  Find new tasks that can be sampled
32:    if  $|\bar{E}_{AT}| = 0$  then
33:      break break once we reach  $q \in F$ 
34:    end if
35:     $\forall \bar{e} \in \bar{E}_{AT} : Q[\bar{e}] = 0$  Set Teacher Q-Value for new tasks to 0 for sampling
36:     $\text{AT} \leftarrow \text{AT} \cup \bar{E}_{AT}$  Add the new tasks to the set of Active Tasks AT
37:  end if
38: end while
39: return  $\Pi^*, \mathcal{P}$ 
```

where the propositions for p hold true.

4.7.3 LSTS and LSTS (CT) - Robotic Domains

To answer (Q3), we test *LSTS* and *LSTS^{ct}* on two simulated robotic environments with high interaction cost. The task in Fig. 4.5a has the following SPECTRL objective:

$$\begin{aligned} \phi_f^{navigation} := & (\text{achieve}(Key_1) \text{ or } \text{achieve}(Key_2); \\ & \text{achieve}(Goal)) \text{ ensuring } (\neg Lava) \end{aligned} \quad (4.6)$$

In this task, the agent (a simulated TurtleBot) needs to collect any of the keys (yellow blocks) present in a $[3m, 3m]$ continuous environment before reaching the goal position (gray block). At all times, the agent needs to avoid the lava object (red wall) present in the center. The move forward (backward) action causes the robot to move forward (backward) by $0.1m$ and the robot rotates by $\pi/8$ radians with each rotate action. The pick-up and drop actions have effects similar to the gridworld domain. The robotic domain is more complex as objects can be placed at continuous locations. The agent receives an ego-centric image view of the environment (top-right corner of Fig. 4.5a), which makes the task partially observable in nature and more complex to get a successful policy. The RL agent is described in Sec. 4.7.1.

The second environment (Fig. 4.5c) consists of a simulated robotic arm pushing two objects to their target locations [35] with the SPECTRL formula:

$$\phi_f^{manipulation} := \text{achieve}(p_1) ; \text{achieve}(p_2) \quad (4.7)$$

where p_1 and p_2 are the atomic propositions for ‘push-object-1’, ‘push-object-2’. The robot has continuous action parameters for moving the arm and a binary gripper action (close/open). An episode begins with the two objects randomly initialized on the table, and the robotic arm has to push these two objects to its final location. The agent receives its current end-effector pose, positions and velocities of the two objects, and the desired goal position for the two objects. For this task, we use the Deep Deterministic Policy Gradient with Hindsight Experience Replay (DDPG-HER) [5] as our RL algorithm.

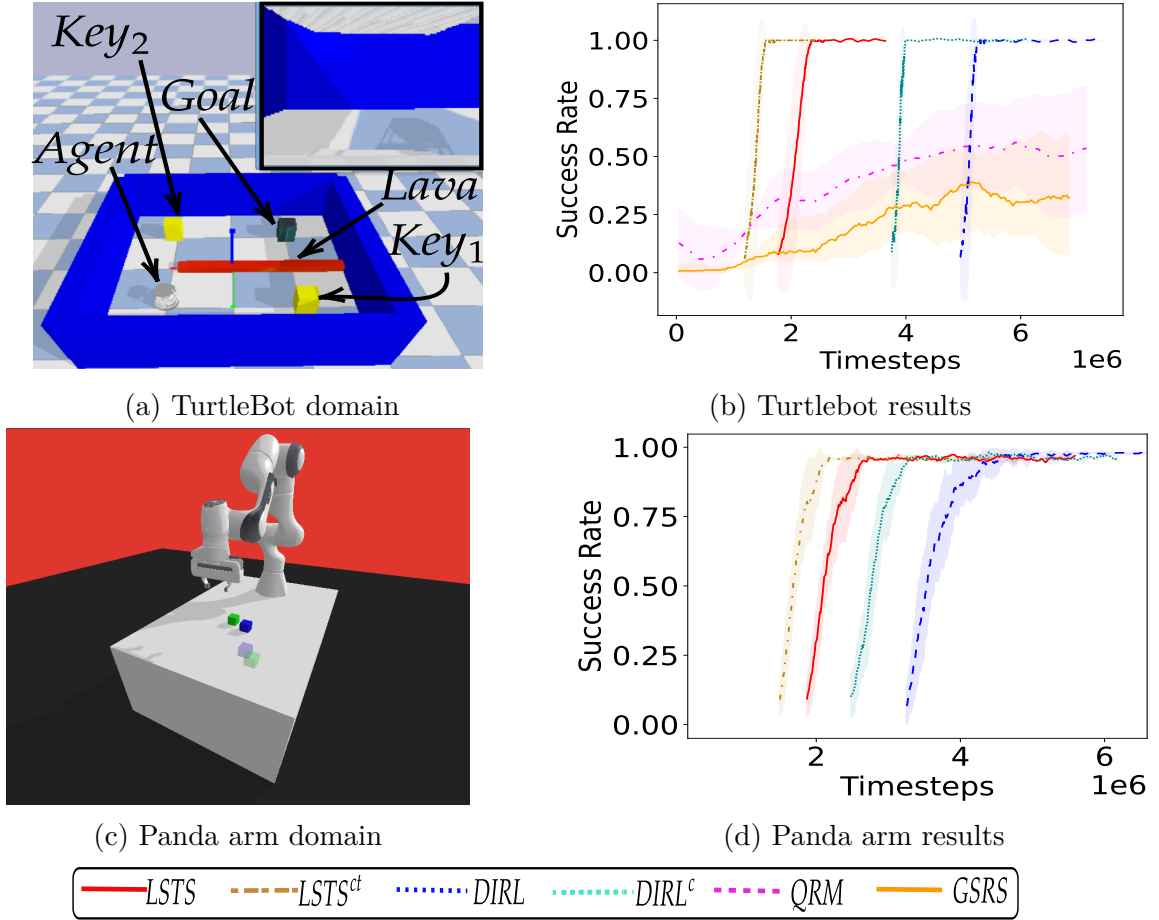


Figure 4.5: Learning curves (Averaged over 10 trials) for the two robotic domains.

DDPG-HER is implemented using the OpenAI baselines [24]. Both the robotic domains were modeled using PyBullet [19], and the reward structure for both the RL agents was sparse, similar to the one described in Sec. 4.7.1. The learning curves in Fig. 4.5 have an offset on the x-axis to account for the interactions in the initial sub-tasks before moving on to the final task in the specification, signifying strong transfer. For both domains, $LSTS$ outperforms all the baselines in terms of learning speed. $LSTS^{ct}$ further speeds-up learning for both the robotic domains. The time-to-threshold between $LSTS$ and the best performing baseline (our custom implementation) $DIRL^c$, is 2×10^6 for the TurtleBot domain and 5×10^5 for the Panda arm domain.

Approach	# Interactions (Mean $\pm$ SD)	Success Rate (Mean $\pm$ SD)
<i>LSTS</i>	$(8.61 \pm 0.12) \times 10^6$	0.87 ± 0.04
<i>LFS</i>	5×10^7	0 ± 0
GSRS	5×10^7	0.05 ± 0.04
QRM	5×10^7	0.05 ± 0.04
TSCL	5×10^7	0 ± 0

Table 4.6: Table comparing #interactions & success rate for the Search and Rescue domain. Our proposed approach, LSTS outperforms the baselines.

4.7.4 LSTS - Search and Rescue task

To demonstrate how LSTS performs when the plan length becomes deeper, we evaluated LSTS on a complex urban Search and Rescue domain with multi-goal objectives. In this domain, the agent acts in a grid setting where it needs to perform a series of sequential sub-tasks to accomplish the final goal of the task. The agent needs to open a door using a key, then collect a fire extinguisher to extinguish the fire, and then find and rescue stranded survivors. The order in which these individual sub-goals such as opening the door, rescuing the survivors, and extinguishing the fire are achieved does not matter. A fully-connected graph $\mathcal{G}_\phi$ generated using the above mentioned high-level states consists of 24 distinct DAG paths. This is a multi-goal task as the agent needs to find the key to open the door, then extinguish fire and rescue survivors to reach the goal state. The results in Table 4.6 (averaged over 10 trials) show that *LSTS* reaches a successful policy quicker compared to the LFS, GSRS, QRM and TSCL. The overall number of interactions to learn a set of successful policies for satisfying the high-level goal objective are higher compared to the door key experiment because of the additional complexity of task. We observe that LSTS is able to accommodate the task and learn RL policies that satisfy the high-level goal objective.

Hyperparameters

Table 4.7 summarizes the hyperparameters for the PPO [109] used for the gridworld and the TurtleBot navigation experiments.

All the experiments were conducted using a 64-bit Linux Machine, having Intel(R)

Core(TM) i9-9940X CPU @ 3.30GHz processor, 126GB RAM memory and an Nvidia 2080 GPU. The maximum duration for running the experiments was set at 24 hours.

Parameter	Value
Discount factor γ	0.99
Learning rate α	1×10^{-3}
Optimizer	Adam
PPO clipping parameter	0.2
GAE λ	0.95
Entropy regularization coefficient	0.001
Action distribution	Categorical with 6 bins
Network architecture	2 Conv layers of [64,64], followed by 2 linear layers with ReLU activation

Table 4.7: Parameters used for training the Proximal Policy Optimization (PPO).

For training the Panda arm environment, we used the default implementation of DDPG-HER from OpenAI baselines [24, 35].

4.7.5 Discussion: Improving LLM Sub-Goal Generation with Type Hierarchies

We designed a method that queries an LLM to produce sub-goal sequences based on entities and predicates known about the task. Each entity and predicate is assumed to have a single word natural language description. An off-the-shelf LLM is prone to associate certain entities with certain predicates based on its training data and procedure. For e.g., when we attempted to make the search and rescue task even more complex by adding a *debris* element that needs to be moved using the *moving* predicate, we observed that the LLM is associating the predicate with other entities already present in the environment, such as fire extinguisher, door etc. Since the LLM does not have access to the type hierarchy that associates predicates with entities, the LLM is conflicted when two similar entities are applicable to the same predicate. As an experiment, we also provided the type hierarchies to the LLM and we observed that the graph generated using LgTS had a graph edit distance of 4.6 compared to the graph given by an oracle,

which was lower than the graph edit distance observed without the presence of type hierarchies, which was found to be 7.3. Thus, incorporating information that informs the LLM about predicate-entity associations helps produce sub-goal sequences that are semantically closer to the sub-goals given by the LTL_f formula suggested by an oracle.

The prompts generated by the LLM also depend on the type of language model used. When we changed our LLM from LLAMA2 to GPT-4 on the complex search and rescue task mentioned above, we observed a graph edit distance of 5.1 compared to the graph given by an oracle, which was lower than the GED for LLAMA2, which was 7.3. This shows that GPT-4 was successful in producing responses and in turn graphs which were closer to the graph generated from an oracle. With further advancements in the LLM capabilities, we might observe even further improvements in the reasoning ability of such models, which in turn will produce better and meaningful entity-predicate associations. Existing tools such as chain-of-thought prompting and access to a dataset that can fine-tune the LLM to produce valid and useful outputs will further improve the prediction accuracy of the LLM. However, even with such advancements, the environmental configuration will be unknown to an agent that does not have access to the transition dynamics model. This work is a step in that direction. LgTS attempts to bridge the gap between the LLM-generated sub-goal outputs and the policies that an agent can learn to satisfy these sub-goals while minimizing the number of times it interacts with the environment.

4.8 Summary

For LgTS, the natural language description of the entities and the predicates might be incorrect or unavailable. In that case, the sub-goal sequences suggested by the LLM will be based on these incorrect descriptions, and the sequences might harm the learning progress of the agent. Our approach recognizes and discards sub-tasks for which policies exist that can satisfy the sub-tasks' goal objective. This technique minimizes the number of interactions with the environment. One way to deal with this would be to bias away from sub-tasks rather than completely discarding them once the target node is reached,

so in the limit, optimal or near-optimal policies can be obtained.

We also proposed *LSTS*, a framework for dynamic task sampling for RL agents using the high-level SPECTRL objective coupled with the Teacher-Student learning strategy. Through experiments, we demonstrated that *LSTS* accelerates learning, converging to a desired success rate quicker as compared to other curriculum learning and automaton-guided RL baselines. *LSTS^{ct}* further improves sample efficiency by continuing exploration on a new sub-task once a goal state for a sub-task is reached. We also evaluate our approach on long-horizon complex robotic tasks where the state space is large and the actions are continuous. *LSTS* reduces training time without relying on human-guided dense reward function, accelerating learning when the high-level objective is available [111].

The above approaches do not require the object oriented MDP representation of the environment to generate a curriculum. As observed in the experimental results, curriculum generation for high-fidelity environments (such as the robotic navigation and manipulation) is still sample inefficient. Additionally, high-fidelity environments (such as physical environments) require additional human input to reset the environment to a state where the reinforcement learning algorithm can be implemented. In the next chapter, we look at a new technique aimed to transfer the structure of the curriculum from a low-fidelity environment to a high-fidelity environment, so to further improve the sample efficiency in expensive high-fidelity environments.

Chapter 5

Curriculum Schema Transfer between Fidelities Using Object-Oriented MDPs

In the previous two chapters, we went over new techniques for curriculum generation for reinforcement learning agents. As observed in the experiments of the previous chapter, curriculum generation for high-fidelity environments (such as the robotic navigation and manipulation) is still sample inefficient. Additionally, high-fidelity environments (such as physical environments) require additional human input to reset the environment to a state where the reinforcement learning algorithm can be implemented. This makes use of reinforcement learning techniques for learning behaviors in high-fidelity environments impractical.

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.¹

5.1 Introduction

Curriculum Learning (CL) attempts to optimize the order in which an agent accumulates experience, increasing performance while reducing training time for complex tasks [85, 33, 84]. 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 [85]. 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, 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 [143, 142], 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 [85]), and to our knowledge, our novel approach is the first to address this problem. We refer to this

¹This chapter is based on the following two papers:
 Shukla, Yash, Christopher Thierauf, Ramtin Hosseini, Gyan Tatiya, and Jivko Sinapov. “ACuTE: Automatic Curriculum Transfer from Simple to Complex Environments.” In Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems, pp. 1192-1200. 2022.. [114]
 Shukla, Yash, and Jivko Sinapov. ”A Framework for Curriculum Schema Transfer From Low-Fidelity to High-Fidelity Environments.” In 3rd Workshop on Closing the Reality Gap in Sim2Real Transfer for Robotics at RSS, vol. 2022. 2022.. [112]

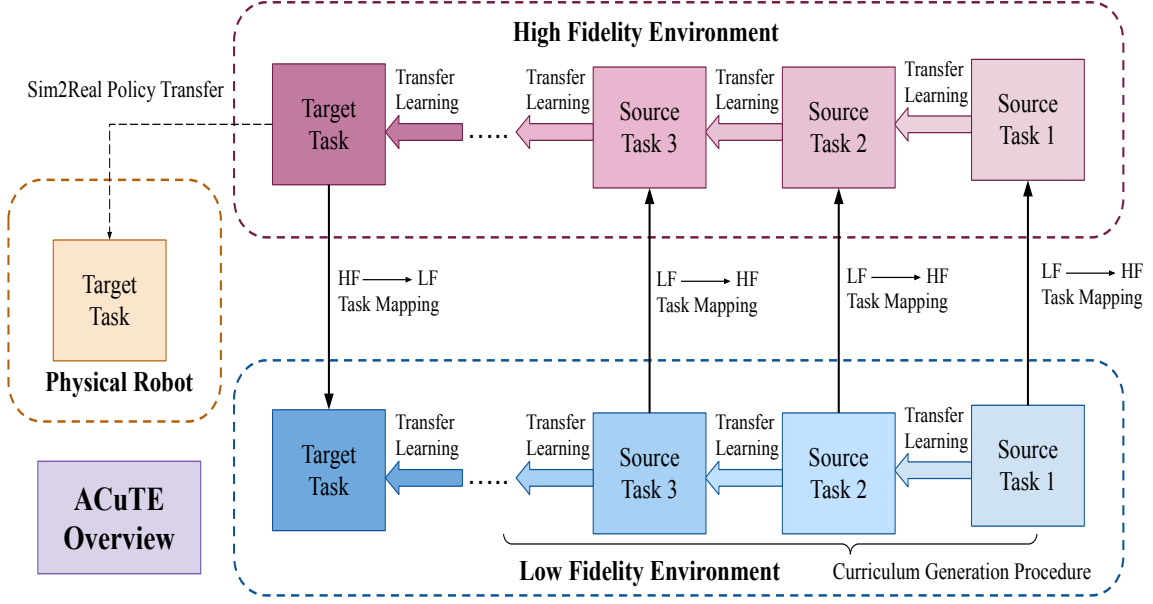


Figure 5.1: Overview of the proposed curriculum transfer approach ACuTE. The curriculum is generated and optimized in the low-fidelity environment, which is then mapped to generate a curriculum of the high-fidelity environment and learned before learning the final target task. The policy is then transferred to a physical robot.

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 [105, 121]. 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 5.1. We consider a complex task and call it the HF target task, and map it to its simplified LF representation using techniques from Object Oriented Markov Decision Processes [27]. 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.

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 [15], Self-Play [125] and Teacher-Student curriculum learning [76]. 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.

5.2 Related Work

Transfer Learning uses knowledge from learned tasks and transfers it to a complex target task [128]. 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 [83, 20, 128, 65]. 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 [71, 1, 72, 65].

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.* [129] 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.* [92]. Continual learning on incremental

simulations can help tackle the sample inefficiency problem of domain randomization [53]. In contrast, Operational Space Control [58] avoids domain randomization while speeding up training with fewer hyperparameters. Carr *et. al.* [15] 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 [29], control problems [107] and in supervised classification tasks [8]. CL has been extensively used for RL, with applications ranging from complex games [37, 140], to robotics [57], and self-driving cars [98]. In [85], 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 [120, 61]. Task sequencing orders the generated tasks to facilitate efficient transfer from the source to the target task [84, 125, 76]. Metaheuristic search methods are a popular tool to evaluate the performance of the task sequencing optimization framework [33]. 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.

5.3 Theoretical Framework

5.3.1 Markov Decision Processes

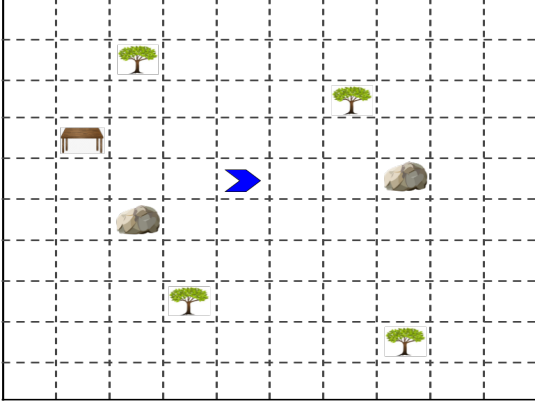
An episodic Markov Decision Process (MDP) M is defined as a tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $p(s'|s, a)$ is the transition function, $r(s', a, s)$ is the reward function and $\gamma \in [0, 1]$ is the discount 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 .

5.3.2 Curriculum Learning (CL)

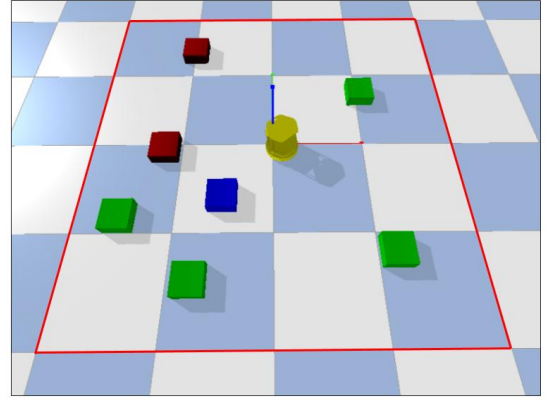
Let $\mathcal{T}$ be a set of tasks, where $M_i = (\mathcal{S}_i, \mathcal{A}_i, p_i, r_i)$ is a task in $\mathcal{T}$. Let $\mathcal{D}^\mathcal{T}$ be the set of all possible transition samples from tasks in $\mathcal{T}$: $\mathcal{D}^\mathcal{T} = \{(s, a, r, s') \mid \exists m_i \in \mathcal{T} \text{ s.t. } s \in \mathcal{S}_i, a \in \mathcal{A}_i, s' \sim p_i(\cdot|s, a), r \leftarrow r_i(s, a, s')\}$. A curriculum $C = [M_1, M_2, \dots, M_n]$ 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 .

5.3.3 Object-Oriented MDP

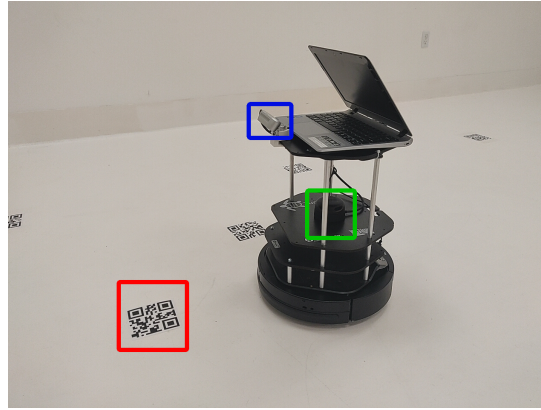
The Object-Oriented MDP (OOMDP) representation [27] abstracts the task description, enabling us to intuitively generalize the HF environment. In OOMDP setting, the task space is abstracted by a set of classes $\mathcal{C} = \{C_1, C_2, \dots\}$, and each class $C_i \in \mathcal{C}$ has a set of parameters $Par(C_i) = \{C_i.p_1, \dots, C_i.p_{|p|}\}$. Each parameter $C_i.p_i$ has a range of values the parameter can attain, given by $Range(C_i.p_i)$. At any given instance, an environment consists of a set of objects $\mathcal{O} = \{o_1, o_2, \dots, o_{|o|}\}$, where each object o_i is an instance of a class C_i and is defined by the values of the parameters for that class $o_i = Par(C_i(o_i))$. For a task, the OOMDP state s_{oo} is given by the union of all object states $s_{oo} = \bigcup_{o_i \in \mathcal{O}} State(o_i)$, where each object state is the value of the parameters of that object $State(o_i) = \{o_i.p_1, \dots, o_i.p_{|p|}\}$.



(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 5.2: Illustration of the final target task in LF, HF, and physical environments. 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.

5.3.4 Problem Formulation

CL aims to generate a curriculum and train an agent on a sequence of tasks $\{M_1, M_2, \dots, M_U\}$, such that the agent’s convergence performance increases, or its time-to-threshold performance on the final target task (M_U) improves relative to learning from scratch. Time-to-threshold (Δ) metric computes how much 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. Here δ is desired performance threshold, in success rate or episodic reward. The domain $\mathcal{T}^{HF}$ of possible tasks is a set of MDPs in the high-fidelity (HF) environment, obtained by varying the objects in the target task $\mathcal{O}_{M_i^{HF}}$, their parameters and the goal condition $g_{M_i^{HF}}$ for task M_i^{HF} .

A suitable candidate for a Low-Fidelity environment is one that has a corresponding OOMDP state s_{oo}^{LF} for any HF OOMDP state s_{oo}^{HF} . To be qualified as a Low-Fidelity environment, the time-to-threshold for the LF task M_i^{LF} must be lower than the time-to-threshold for the equivalent HF task M_i^{HF} , i.e. $\Delta(M_i^{LF} - M_i^{HF}) > 0$.

The goal of this work is four-fold:

1. Assuming existence of an LF environment, find a LF environment that will satisfy the above constraints.
2. Assuming existence of a set of mapping functions $\mathcal{F} := \{f_1, \dots, f_n\}$ that provide a convenient mapping between the OOMDP states of the LF and the HF environments, the goal is to find one such mapping.
3. Generate and optimize the curriculum in low-fidelity.
4. Transfer the schema of the curriculum from LF to HF and learn the target task in HF through this curriculum.

Let $\mathcal{C}_U^{\mathcal{T}^{LF}}$ be the set of all curricula over tasks $\mathcal{T}^{LF}$ of length U in the LF environment. Similarly, let $\mathcal{C}_U^{\mathcal{T}^{HF}}$ be the set of all curricula over tasks $\mathcal{T}^{HF}$ of length U in the HF environment. Each task of the optimized curriculum in LF $c_U^{\mathcal{T}^{LF}}$ is mapped through $\mathcal{F}$ to obtain an optimized curriculum for HF $c_U^{\mathcal{T}^{HF}}$, where the mapping is an affine transformation given by:

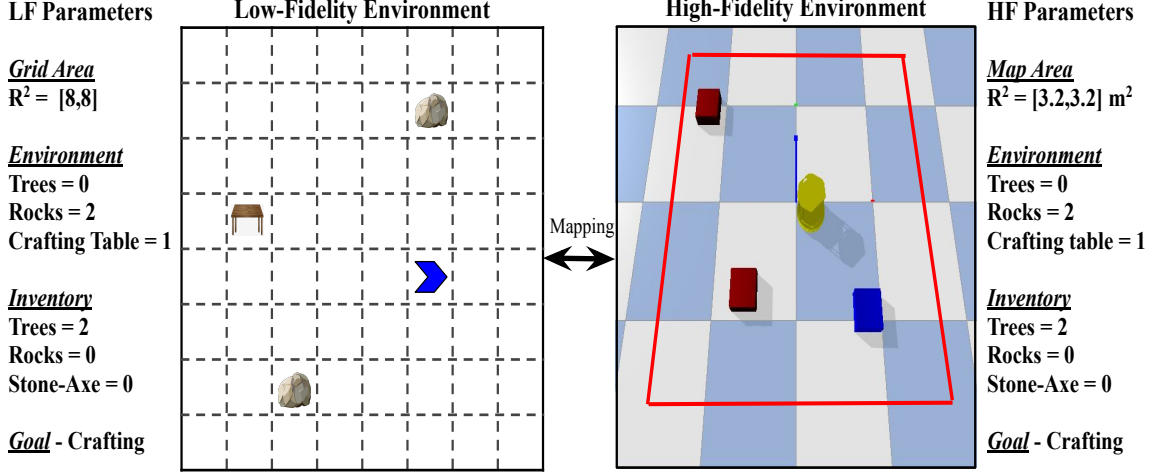


Figure 5.3: Illustrative example of the low-fidelity to high-fidelity mapping in the crafting task. Each of the class parameters in the LF domain are mapped to generate corresponding parameter in the HF domain.

$$s_{oo}^{HF} = K \odot s_{oo}^{LF} + L$$

where $K = [k_1, \dots, k_n]^T \in \mathbb{R}^n$ and $L = [l_1, \dots, l_n]^T \in \mathbb{R}^n$ denote linear mapping and translation vectors and $\odot$ is the Hadamard product. Thus, a parameter mapping ($f_i : o_i \cdot a_i^{LF} \rightarrow o_i \cdot a_i^{HF}$) is given by:

$$o_i \cdot p_i^{HF} = k_i \cdot (o_i \cdot p_i^{LF}) + l_i$$

We assume that the source tasks of the curriculum in the HF environment are learned before learning the final target task, as described in Section 5.3.2.

5.4 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 5 presents our approach. The first step entails generating the LF target task from the HF target task using the approach described above.

The next step entails generating the curriculum in LF. To generate and sequence the source tasks in the LF environment, we compared two approaches. The first approach,

Algorithm 4 LF_Gen($\mathcal{C}^{HF}, \mathcal{P}^{HF}, \mathcal{O}_{M_U^{HF}}, \mathcal{X}, n$)

Output: LF target task descriptors: $\mathcal{C}^{LF}, \mathcal{P}^{LF}, \mathcal{O}_{M_U^{LF}}$

Set of mapping functions: $\mathcal{F}$

Placeholder Initialization: Set of Classes in LF: $\mathcal{C}^{LF} \leftarrow \emptyset$

Set of Parameters for each class in LF: $\mathcal{P}^{LF} \leftarrow \emptyset$

Set of Objects in LF target task: $\mathcal{O}_U^{LF} \leftarrow \emptyset$

Set of Mapping functions $\mathcal{F} \leftarrow \emptyset$

Algorithm:

```
1:  $\mathcal{C}^{LF} = \{c \mid c \in \mathcal{C}^{HF}\}$  ▷ Assign set of classes in LF
2: for  $o \in \mathcal{O}_{M_U^{HF}}$  do
3:    $c \leftarrow \text{getBaseClass}(o)$ 
4:    $\mathcal{O}_{M_U^{LF}} \leftarrow \mathcal{O}_{M_U^{LF}} \cup \{\text{createObject}(c)\}$  ▷ Create objects in LF
5:    $\mathcal{P}_c^{HF} = \mathcal{P}^{HF}[c]$ 
6:   for  $p \in \mathcal{P}_c^{HF}$  do
7:      $\mathcal{P}_c^{LF} \leftarrow \mathcal{P}_c^{LF} \cup \{p\}$  ;  $\mathcal{P}^{LF} \leftarrow \mathcal{P}^{LF} \cup \mathcal{P}_c^{LF}$ 
8:     if  $|\text{Range}(o.p)| \leq n$  then ▷ Compare parameter range with maximum permissible
       cardinality constant given by human expert
9:        $o.p^{LF} = o.p$ 
10:       $f_{o.p} = 1$ 
11:    else
12:       $o.p^{LF} = (o.p/\mathcal{X}[c,p])$  ▷ LF value given by hyperparam
13:       $f_{o.p} = 1/\mathcal{X}[c,p]$ 
14:    end if
15:     $\mathcal{O}_{M_U^{LF}}[o].p = o.p^{LF}$  ▷ Assign value to LF object
16:     $\mathcal{F} \leftarrow \mathcal{F} \cup \{f_{o.p}\}$ 
17:  end for
18: end for
19: return  $\mathcal{C}^{LF}, \mathcal{P}^{LF}, \mathcal{O}_{M_U^{LF}}, \mathcal{F}$ 
```

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).

5.4.1 Low-Fidelity Environment Generation

The curriculum generation for the high-fidelity (HF) task consists of three parts: (1) Generation of a suitable low-fidelity (LF) environment along with its HF mapping functions ($\mathcal{F}$); (2) curriculum optimization in the LF environment, and (3) curriculum schema transfer to the HF environment. We provide a framework for obtaining a suitable LF environment, and generating the mapping functions according to constraints described in Section 5.3.4.

To generate a suitable LF environment, we begin by modeling the HF target task using OOMDP approach [27, 120]. The HF environment is modeled using a set of classes $\mathcal{C}^{HF} = \{C_1^{HF}, \dots, C_{|c|}^{HF}\}$, and each class $C_i^{HF} \in \mathcal{C}^{HF}$ has a set of parameters $Par(C_i^{HF}) = \{C_i^{HF}.p_1, C_i^{HF}.p_2, \dots, C_i^{HF}.p_{|p|}\}$, each parameter $C_i^{HF}.p_i$ contained within a range $Range(C_i^{HF}.p_i)$. The target task in the HF environment, with MDP M_U^{HF} , has a set of objects $\mathcal{O}_{M_U^{HF}} = \{o_1^{HF}, o_2^{HF}, \dots, o_{|o|}^{HF}\}$ with object states given by the value of the parameters of that object $State(o_i^{HF}) = \{o_i^{HF}.p_1, o_i^{HF}.p_2, \dots, o_i^{HF}.p_{|p|}\}$.

The overall approach of generating the LF environment and the mapping function is given in Algorithm 4. The input to **LF_Gen** is the set of HF environment classes $\mathcal{C}^{HF}$, the set of set of parameters $\mathcal{P}^{HF}$, objects in the the HF target task $\mathcal{O}_{M_U^{HF}}$, a set of discretization hyperparameter constants $\mathcal{X}$, and a human-expert specified finite scalar n , denoting the maximum desired cardinality for the parameter range for the task in hand. The output of **LF_Gen** is the set of LF environment classes $\mathcal{C}^{LF}$, the set of set of parameters $\mathcal{P}^{LF}$, objects in the the LF target task $\mathcal{O}_{M_U^{LF}}$ (together making up the LF target task), along with the LF $\leftrightarrow$ HF mapping function set. To begin, all the classes in the HF environment have a corresponding class in the LF environment (line 1). For each object in the HF target task, **LF_Gen** first determines the base class for the object, and creates an LF object for the class (lines 2-4). Then, for each parameter of that class, **LF_Gen** determines if the range for the parameter lies with the desired cardinality for the task. If it does, then the value of the parameter of the HF target task object is equivalent with the value of the parameter of the LF target task object, and the mapping between these two parameters is 1 (lines 8-10). If the range of the parameter exceeds the desired cardinality, then the LF object parameter is determined by the discretization hyperparameter for the HF parameter ($\mathcal{X}[c, p]$), and the mapping is given by the reciprocal of the hyperparameter (lines 11-13) [113].

Thus, by iterating over all parameters for all classes in the HF environment, **LF_Gen** bounds the range of the parameter space, thereby decreasing the number of OOMDP states, while keeping the objective of the task unchanged. This accelerates the curriculum optimization procedure, and provides a convenient mapping for curriculum schema

transfer.

Once a suitable LF environment is generated, the next step entails optimizing the curriculum in LF environment. The curriculum in the LF is obtained using “ACuTE” [114], where the agent generates source tasks by varying the parameters of the objects present in the final target task, and uses beam search to optimize the source task sequence based on the transfer potential to the target task. Then, each source task of the curriculum in LF is mapped to a source task in HF using the set of mapping functions ($\mathcal{F}$).

5.4.2 Handcrafted and Automated Curriculum Transfer Generation

Task Generation and Sequencing to Generate Handcrafted Curriculum

The agent learns the first task M_1^L in the curriculum with some initial policy $\pi_{1,i}^L$. The task terminates when the agent has met the *stopping criterion* (determining if the agent met the goal $\geq \delta_G$ in the past α episodes) (line 12). Failure to meet the *stopping criterion* implies agent has reached maximum episodes, denoting $\delta < \delta_G$. For policy transfer in our handcrafted (and automated) curriculum task sequencing approaches, the agent uses the learned policy from the first task $\pi_{1,f}^L$ as an initial policy for the next task M_2^L in the curriculum. This transfer helps the agent to learn the new source task while performing actions that would yield higher reward for the previous learned tasks. After recursively learning all the source tasks of the curriculum, the agent attempts the final task (M_U). The central idea behind this approach is to incrementally build up the policy from the experiences observed by the agent.

Task Generation and Sequencing to Generate Automated Curriculum

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 6. 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 5 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.

Algorithm 5 ACuTE($N, W, U, \mathcal{C}^{HF}, \mathcal{P}^{HF}, \mathcal{O}_{M_U^{HF}}, \mathcal{X}, n, s, b$)

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

```

1:  $M_U^{LF}, \mathcal{F} \leftarrow \text{LF\_Gen}(\mathcal{C}^{HF}, \mathcal{P}^{HF}, \mathcal{O}_{M_U^{HF}}, \mathcal{X}, n)$ 
2: if curriculum = HC then
3:    $C^{LF} \leftarrow \text{Generate\_HC}(M_U^{LF}, U)$ 
4: else if curriculum = AC then
5:    $C^{LF} \leftarrow \text{Generate\_AC}(M_U^{LF}, U, N, W, \text{seeds})$ 
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:     $\pi_u^{HF} \leftarrow \text{Learn}(M_u^{HF}, \pi_{u-1}^{HF})$ 
12:    if  $\mathbb{E}[\pi_u^{HF}[\text{episode} - s :]] \geq \delta_G$  then
13:      break
14:    end if
15:  end while
16: end for
17: return  $\pi_U^{HF}$ 

```

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 [74]. 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).

Algorithm 6 `Generate_AC( $N, W, U, M_U^{LF}, seeds$ )`

Output: LF Curriculum Parameters: P_W

Placeholder Initialization: Timesteps: $T_1, \dots, T_U \leftarrow \emptyset$

LF task params for all tasks at each beam level $\xi_1, \dots, \xi_U \leftarrow \emptyset$

LF task params at each width and level $\xi_{1,W}, \xi_{2,W}, \dots, \xi_{U,W} \leftarrow \emptyset$

LF curriculum params $P_W \leftarrow \emptyset$

LF task policies for all tasks at each beam level: $\Pi_1, \dots, \Pi_U \leftarrow \emptyset$

LF task policies for each width and level: $\Pi_{1,W}, \dots, \Pi_{U,W} \leftarrow \emptyset$

```

1: for  $u \in U$  do
2:   for  $w \in W$  do
3:     for  $n \in N$  do
4:       if  $u = 1$  then
5:          $P^{LF_u} \leftarrow \text{Init\_Src}(M_U^{LF})$ 
6:          $\pi_{1,w,init} = \text{Init\_Agent}(seeds)$ 
7:       else
8:          $P^{LF_u} \leftarrow \text{Init\_Inter}(\xi_{u-1,W}[w], M_U^{LF})$ 
9:          $\pi_{u,w,init} = \text{Load\_Agent}(\Pi_{u-1,W}[w])$ 
10:      end if
11:       $\xi_u[w, n] \leftarrow P^{LF_u}$ 
12:       $M_{u,w,n}^{LF} = \text{Generate\_Env}(P^{LF_u})$ 
13:       $(t_{u,w,n}, \pi_{u,w,n,fin}) = \text{Learn}(M_{u,w,n}^{LF}, \pi_{u,w,n,init})$ 
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:   $T_{u,W}, \Pi_{u,W}, \xi_{u,W} = \text{Best\_Candidates}(T_u, \Pi_u, W, \xi_u)$ 
18: end for
19:  $P_W \leftarrow \text{Best\_LF\_Params}(\xi_{1,W}, \xi_{2,W}, \dots, \xi_{U,W})$ 
20: return  $P_W$ 

```

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 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 $\geq$ the number of goals attainable. This ensures all the goals available (P_G) are encountered by the agent in the curriculum.

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 5, line 8-15).

5.5 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 the curriculum transfer framework yield successful convergence with imperfect (e.g., noisy) mappings between the HF and LF environments?

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.

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 [18]. An example of the LF $\leftrightarrow$ HF mapping between the tasks in LF and HF environments is shown in Fig 5.3. 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.

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* [65, 33] 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:

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

where $G_{M_f^c}^i$ and $G_{M_f^b}^i$ 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 [128, 85], 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.

5.5.1 Curriculum Generation in High Fidelity

We used the algorithm presented in Algorithm 5 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 $\mathbb{R}_{LF}^2 \rightarrow [10 \times 10]$, as seen in Fig 5.2a, and the high-fidelity (HF) target task area is a continuous $\mathbb{R}_{HF}^2 \rightarrow [4m \times 4m]$ space, as seen in Fig 5.2b. Both these environments contain 4 trees, 2 rocks and 1 crafting table, placed at random locations.

The RL algorithm was a Policy Gradient [138] 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.

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 [125], Teacher-Student Curriculum learning [76] and Domain Adaptation for RL [15]. 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.

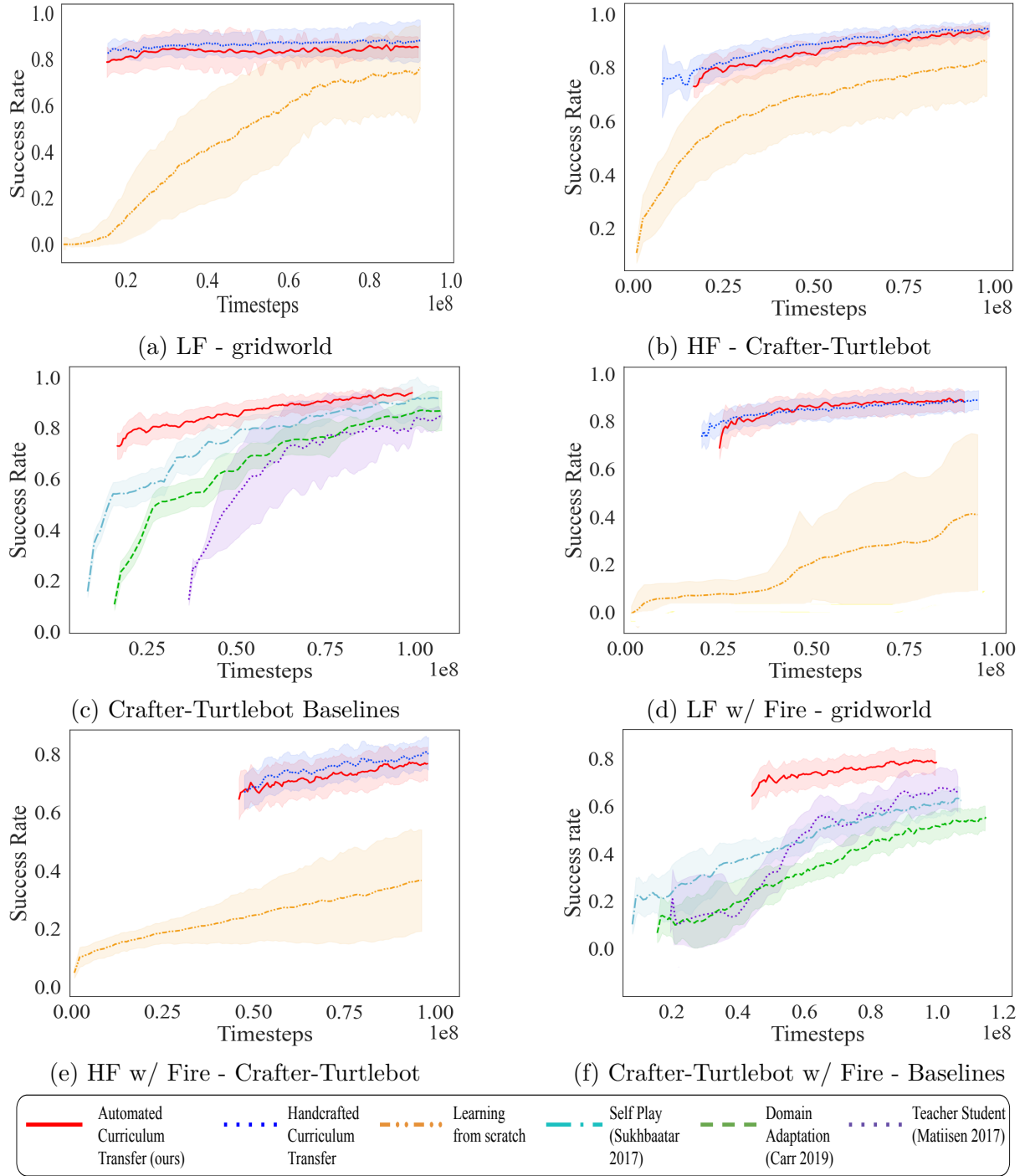


Figure 5.4: Learning Curves for low fidelity (LF) and high fidelity (HF) environments with and without Fire. A common trend observed from the learning curves is the improved performance of ACuTE (red) compared to the baseline approaches.

The results in Figs 5.4a and 5.4b 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 5.4f, the curriculum transfer method outperforms the three baseline approaches 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* [128]. 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.

Table 5.2 compares the *jumpstart* values for AC with the baselines. The higher *jumpstart*, the better performance of the curriculum. 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).

An Example of a Curriculum Candidate

An example of the curriculum generated by the Automated Curriculum procedure is shown in Fig 5.5. The goal of the agent in the first source task (Task-1) is to navigate to the rock present in the environment. Once the agent meets the *stopping criteria* required, the agent uses the learned policy to learn the next source task (Task-2) in the curriculum. The goal of the agent in the second source task is to navigate to the object present in the environment and break it to collect the object in the agent’s inventory. Similarly, when the agent meets the *stopping criteria*, the policy is transferred to learn Task-3, whose goal is to break the trees and rocks present in the environment, collect them in the inventory

Parameter	Minimum value	Maximum value
Width	LF - 6, HF - $2m$	LF - 10, HF - $4m$
Height	LF - 6, HF - $2m$	LF - 10, HF - $4m$
Number of trees	0	5
Number of rocks	0	2
Number of crafting tables	0	1

Table 5.1: Parameters used for training the Policy Gradient on the High-Fidelity task

and then craft a stone-axe at the crafting table.

Once the agent has learned all the source tasks in the curriculum, the agent attempts learning the final target task, as shown in the figure. As shown in the example, the agent is exposed to all the sub-goals available in the environment (P_G), and is thus able to learn the final target task efficiently, as seen from the learning curves.

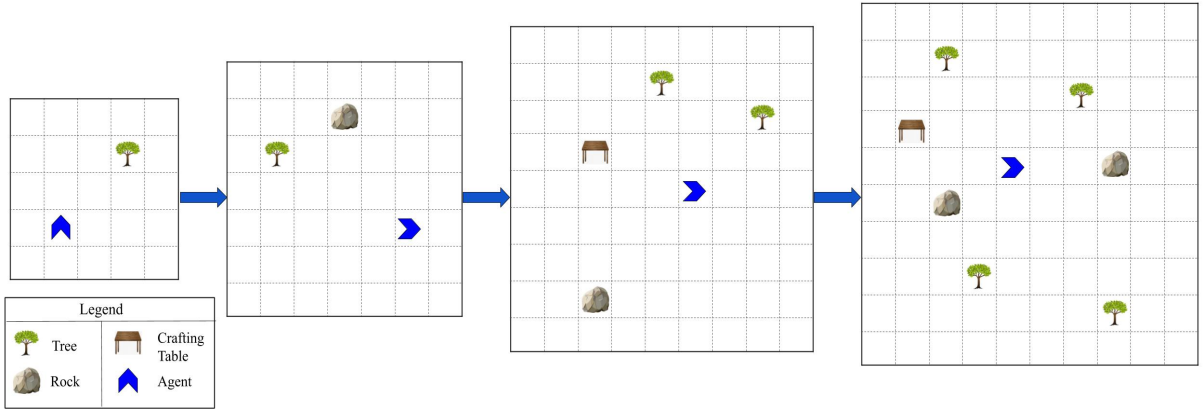


Figure 5.5: An example of the Automated Curriculum procedure in the low-fidelity environments. The agent learns complex tasks incrementally, before learning the final target task.

To generate the source tasks in the high-fidelity environment, each of the LF source tasks is mapped to generate the HF source tasks. The HF source tasks are learned iteratively, culminating in the HF target task. The sub-goal can be either to break the required number of items or to craft a stone-axe.

The range of values attained by the parametric variables is listed in the table 5.1 below.

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

Table 5.2: Table comparing *jumpstart* (mean $\pm$ 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.

5.5.2 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 5.5.1 are used to optimize AC in the LF environment. From learning curves in Figs. 5.4d, 5.4e, 5.4f and Table 5.2, 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 5.2 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.

5.5.3 Baseline Adaptation

As mentioned, we adapted three baseline approaches from the literature: Asymmetric Self-Play [125], Domain Adaptation for RL [15] and Teacher-Student Curriculum learn-

ing [76].

Asymmetric Self-Play

We adapted the asymmetric self-play approach proposed by [125] to our problem of crafting a stone-axe. Alice, the goal-proposing policy, proposes a goal for Bob, whose goal is to reach the goal proposed by Alice. We used the approach in *Repeat* mode, where Bob repeats the task instead of reversing it, and tries to reach the goal proposed by Alice.

The maximum number of steps for Alice and Bob were set to 600, equal to the steps allowed for Automated Curriculum approach. We tuned the hyperparameters through experimentation for our task, using a heuristic grid search. The parameters we used were: Self-Play percentage: 10%, Self-play reward scale γ : 0.01 and the self-play mode at *repeat*.

According to [125], Alice’s reward function is given by $r_A = \max(0, t_B - t_A)$ where t_B is the time taken for Bob to reach the goal state proposed by Alice, whereas t_A is the time taken by Alice to propose a goal (execute ‘stop’ action). In the target task, the goal is to craft a stone axe. We see that Alice’s rewards are sparse, and for Alice to keep receiving positive rewards, it needs to propose a goal that is just difficult for Bob to repeat. In our task, the large goal space makes it difficult for Alice to propose a task that is just difficult for Bob to repeat, making it difficult to optimize the curriculum according to the goals proposed by Alice, taking it longer to obtain a curriculum.

One difficulty is that the time (in interactions) spent by Alice in proposing a task is a sunk cost, and requires tuning the self-play percentage parameter to not increase this added cost. The self-play approach does help Bob explore the space, and is responsible for the jumpstart Bob achieves initially in the target task, evident from the baseline comparison learning curves. From our experiments, we observed that the exploration helped Bob in navigation, but was not effective in performing the sequential actions required for the target task. In the presence of Fire in the Crafter-TurtleBot environment, Bob’s self-play episodes helped the agent realize the negative reward associated with the fire object, and provided quicker learning in the target task.

Domain Adaptation for RL

We adapted the approach given in [15] for domain adaptation from the low-fidelity grid-world environment (source) to the high-fidelity Crafter-TurtleBot environment (target).

The overall approach used for the transfer is highlighted in Algorithm 7

Algorithm 7 DA_LF_to_HF(M^{LF}, M^{HF})

Output: HF target task policy: π_f^{HF}

Algorithm:

```

1:  $\pi_i^{LF} \leftarrow \emptyset$  ▷ Initialize the LF policy
2:  $\pi_f^{LF} \leftarrow \text{Learn}(M^{LF}, \pi_i^{LF})$  ▷ Learn the LF task
3: for  $epoch \in total\_epoch$  do
4:    $s^{HF}[] \leftarrow \emptyset$  ▷ Initialize target (HF) observations
5:   for  $batch \in total\_batch$  do
6:      $s^{HF}[batch] \leftarrow \mathcal{S}_{M^{HF}}$  ▷ Sample HF observations
7:      $\text{Auto\_Encoder}(s_t^{HF}[batch])$  ▷ Train Auto encoder-decoder network
8:     for  $k \in N$  do
9:        $x^{LF} \sim \text{Sample\_LF}(M^{LF}, \pi_f^{LF})$  ▷ Sample LF encoder latent output
10:       $x^{HF} \sim \text{Sample\_HF}(M^{HF}, \pi_i^{HF})$  ▷ Sample HF encoder latent output
11:       $\theta_D = \text{Train\_Discriminator}(\theta_D, \theta_G, \text{Discriminator\_model})$ 
12:    end for
13:     $y^{HF} \sim \text{Sample\_Observations}(M^{HF}, \pi_i^{HF})$  ▷ Sample HF observations
14:     $\theta_G = \text{Train\_Generator}(\theta_D, \theta_G, \text{Generator\_model})$ 
15:  end for
16: end for
17:  $\pi_f^{HF} = \text{Learn}(M^{HF}, \theta_G, \pi_i^{HF})$ 
18: return  $\pi_U^{HF}$ 

```

The approach we used for transfer from the LF to the HF is summarized below:

1. We trained the agent to accomplish the LF target task, to learn the LF target policy π_f^{LF} .
2. Train autoencoder on HF observations.
3. Generate LF and HF encoder output, representing the embedding space.
4. Train the Adversarial AutoEncoder.
5. Using the Generator model, train the HF policy

Given the semantic similarity between two separate environments, domain adaptation helps in providing a performance boost, over the random initialized neural networks. In

our implementation, we observed that the increased complexity of the neural network, required more training time for the domain adaptation approach, and hence did not outperform the learning from scratch baseline approach initially in presence of fire. In both the HF environments (with and without fire), we observed the domain adaptation converging to a higher reward compared to the learning from scratch approach. We tuned the hyperparameters by performing a heuristic grid search. In some cases, the default hyperparameters performed worse than learning from scratch.

Teacher-Student Curriculum Learning

The proposed teacher-student curriculum learning assumes a teacher optimizing the sequence of the tasks in the curriculum for the student. The teacher proposes those tasks for which the student shows the highest potential in learning, i.e. the tasks for which the student learns the quickest. The ability of a student to learn a task is given by its slope of the curve of plotted reward function. The steeper the slope, the quicker the student is able to learn the task. One limitation of this approach is that it assumes the source tasks of the curriculum are defined. This is difficult in settings where the parameters of the task are continuous. In order to adapt this baseline approach for our method, we had to discretize the continuous parameters of our task to satisfy the constraints of the baseline approach. We used the same policy gradient network for teacher-student curriculum learning approach as we did for the autonomous curriculum transfer. We initialized 9 source tasks and one target task, with varying parameters for each source task. The performance of the agent in the final target task is plotted. We followed the same reward shaping for the tasks in the curriculum as for the source tasks of the automated curriculum transfer approach.

In teacher-student curriculum learning, the sunk cost of the algorithm is the interactions in the source task. The sunk cost is very high when the interactions are costly, as in our high fidelity setup. This contributes to the high requirement of computational resources.

PPO and DQN Implementation

Below are the parameters used for implementation of PPO and DQN algorithms.

Parameter	Value
discount factor γ	0.99
learning rate α	1×10^{-3}
optimizer	Adam
PPO clipping parameter	0.2
Generalized Advantage Estimation λ	0.95
Entropy regularization coefficient	0.001
action distribution	categorical with 5 bins

Table 5.3: Parameters used for training the Proximal policy optimization on the High-Fidelity task

Parameter	Value
discount factor γ	0.995
learning rate α	1×10^{-3}
optimizer	Adam
batch size	256
action distribution	categorical with 5 bins

Table 5.4: Parameters used for training the Deep Q-Learning on the High-Fidelity task

Trends in Curriculum Optimization

We observed some interesting trends while computing the automated curriculum transfer. For the complicated HF task with “fire”, the agent’s approach to optimizing the sequence of source tasks involved learning to avoid the “fire” object in a larger grid, followed by learning to learning to break and craft.

This was in contrast to the handcrafted and teacher-student baseline approaches that tried to incorporate the fire object in the later stages of curriculum. If the detrimental “fire” object was introduced in later stages, the agent tried to re-learn the policies it had already learned and thus spent longer in optimizing the curriculum. Thus, we observe that the automated curriculum transfer approach was able to better generalize to the difficult task through its beam search algorithm, as it can identify the bottlenecks in the pipeline early on, and tackle them accordingly.

5.5.4 Curriculum Generation with Different RL Algorithms

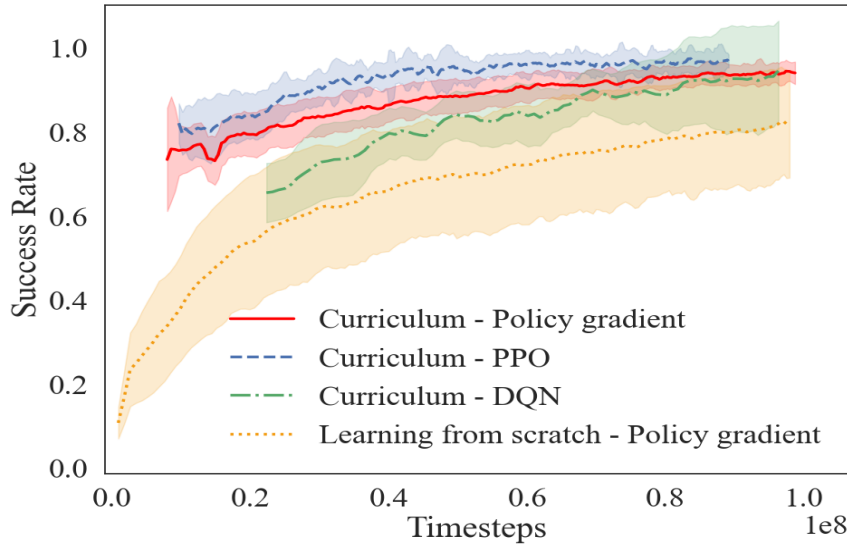


Figure 5.6: Learning HF target task through different RL algorithms.

To answer the third question, we conducted experiments by making the HF learning algorithm different (PPO [109] and DQN [82]) from the RL algorithm used for generating the curriculum in the LF environment (Policy gradient). Fig 5.6 shows the result of this test. 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.

5.5.5 Curriculum Transfer Evaluation in Presence of Noisy Mappings

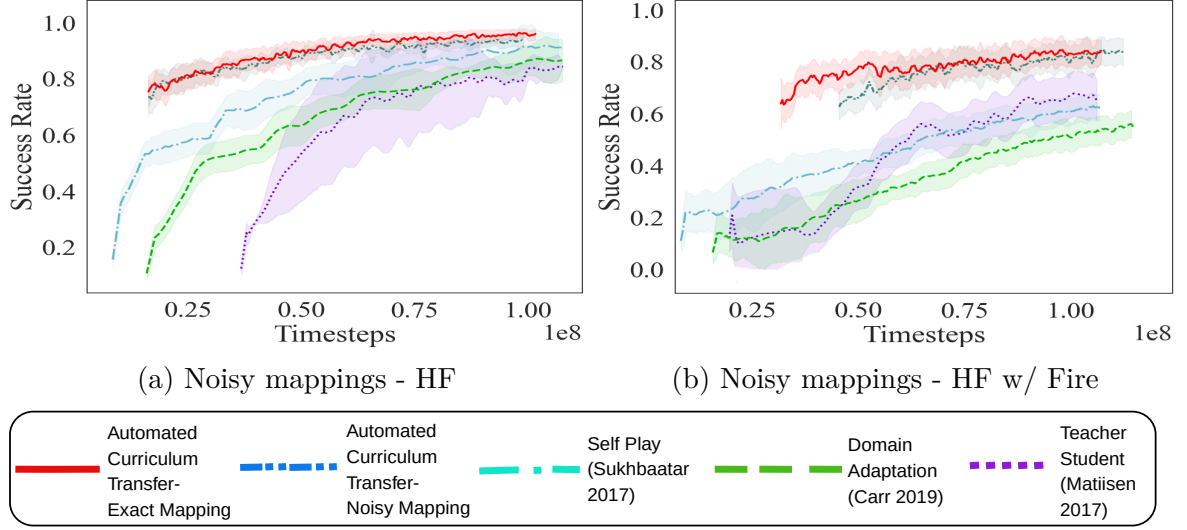


Figure 5.7: Comparison of Learning Curves for high fidelity (HF) environments. Here, the HF environments are generated by assuming an imperfect (noisy) LF to HF mapping.

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 para-

metric 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 5.7. 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.

5.5.6 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 5.8 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 optimize the curriculum in the LF environment and the interactions required to learn the source tasks in the HF environment.

5.6 Summary

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

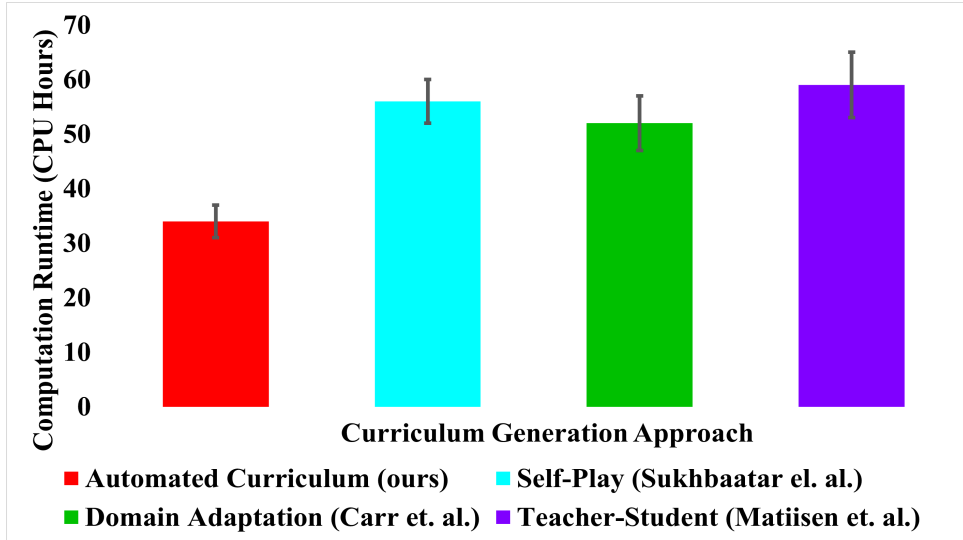


Figure 5.8: Runtime comparison of different approaches. Since our curriculum transfer approach generates the curriculum in Low-Fidelity environment, the curriculum generation is faster as compared to baselines that generate curriculum purely in the High-Fidelity domains.

comparable to a curriculum generated by a human expert in terms of number of environmental interactions taken to learn a successful policy. 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, or other techniques like DAgger [104] can be combined with curriculum transfer such that the agent can learn the tasks in the curriculum even faster.

In this chapter, we looked at a technique for generating the curriculum in the low-fidelity environment, and mapping the schema of the curriculum to a high-fidelity environment. In the high-fidelity environment, the policies are learned from scratch, without

any policy, value function or other types of knowledge transfer from the low-fidelity environment. In the next chapter, we look at cross domain policy transfer. For a successful policy transfer, we look at techniques aimed at mapping the state space of the source domain with the target domain, and performing behavior cloning to map the policy of the agent in the source domain to the target domain.

Chapter 6

Simulation to Real-World Policy

Transfer Using Observation

Mapping and Behavior Cloning

Despite recent progress in Reinforcement Learning for robotics applications, many tasks remain prohibitively difficult to solve because of the expensive interaction cost. In the previous three chapters, we looked at three ways of curriculum generation and one technique for curriculum schema transfer between two domains. In this chapter, we look at transferring the knowledge learned in a source task to a target task through transfer learning techniques. Transfer learning helps reduce the training time in the target domain by transferring knowledge learned in a source domain. Sim2Real transfer helps transfer knowledge from a simulated robotic domain to a physical target domain. Knowledge transfer reduces the time required to train a task in the physical world, where the cost of interactions is high. However, most existing approaches assume exact correspondence in the task structure and the physical properties of the two domains. This work proposes a framework for Few-Shot Policy Transfer between two domains through Observation Mapping and Behavior Cloning. We use Generative Adversarial Networks (GANs) along with a cycle-consistency loss to map the observations between the source and target

domains and later use this learned mapping to clone the successful source task behavior policy to the target domain. We observe successful behavior policy transfer with limited target task interactions and in cases where the source and target task are semantically dissimilar.¹

6.1 Introduction

Recent advances in Reinforcement Learning (RL) have enabled agents to learn the optimal behavior for a wide range of tasks, ranging from Atari games to autonomous driving [44, 94]. Despite these advancements, RL still suffers from sample-complexity issues, as the transition dynamics of the environment are unknown to the agent, and the agent needs to explore the environment to figure out the optimal behavior. Partially observable environments, sparse reward settings, and the dimensionality curse induced by large state and action spaces make many RL tasks prohibitively expensive to learn. This is a major concern in robotics that involves a costly and labor-intensive setup.

Transfer Learning (TL) [128] reduces the number of interactions required in a target domain by transferring relevant knowledge (obtained from a human expert or learned autonomously) from a source domain. Imitation learning [48] is a type of TL that tries to mimic the policy provided by an expert user, thereby learning a policy in scenarios where the reward function of the task is inaccessible. For robotic settings, where training a behavior policy in a simulation environment is inexpensive, Sim2Real Transfer [47] is a special case of TL that allows a model trained in a simulation to be deployed on a physical robot. Training in simulation is easier as it does not have the physical time constraint of real-world settings. However, it suffers from “Reality Gap” [129], where the simulation policy performs poorly on transfer. Domain randomization requires access to a simulator where color, texture and other simulator parameters can be varied. This makes the learned policy fail on out-of-distribution test scenario. Continual learning

¹This chapter is based on the following paper: Shukla, Yash, Bharat Kesari, Shivam Goel, Robert Wright, and Jivko Sinapov. “A framework for few-shot policy transfer through observation mapping and behavior cloning.” *In 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7104-7110. IEEE, 2023. [115]

on incrementally realistic simulations helps mitigate the Sim2Real problem as the agent attempts to transfer relevant information [52]. One key limitation of Sim2Real approaches is the need for a simulation whose Markov Decision Process (MDP) representation exactly matches the complex dynamics of the realistic scenario, and is not always feasible [91].

A major challenge in transferring knowledge from a source to a target task is finding a mapping between the two tasks [136]. Even with the access to a mapping, transferring knowledge remains difficult as the success of the transfer depends on the quality of the mapping. In this work, we propose a framework to achieve few-shot policy transfer from a simulated domain to a physical robotic domain. Our approach does not require task-specific engineering to generate the mapping. It also does not require paired observations (images) between the simulated and the physical domains to generate the mapping. Our approach to obtaining a successful behavior policy in the target task is shown in Figure 6.1. We first obtain a successful behavior policy in the source domain using an off-the-shelf RL algorithm (e.g., PPO, DQN). Next, to obtain the observation mapping, we employ the CycleGAN [144] framework that enforces a pixel-level cycle-consistency loss to obtain the target domain observations given a simulated domain observation. Finally, using the source task policy and the observation mapping, we employ behavior cloning [75] to adapt the successful source task policy to the target task. Recent works have addressed a task-agnostic framework for Sim2Real transfer [102, 46], and we extend this framework to a partially-observable setting that also contain semantic dissimilarities in the two domains.

Further, we compare how our approach scales when limited data is available from the target domain and when the target domain has different transition dynamics. This is necessary when the interactions in the target domain are costly and require an expensive setup. We conclude by scaling our approach to a target task that has different semantics and actions than the source task (e.g., the action 'move forward' moves the agent forward differently by different amounts).

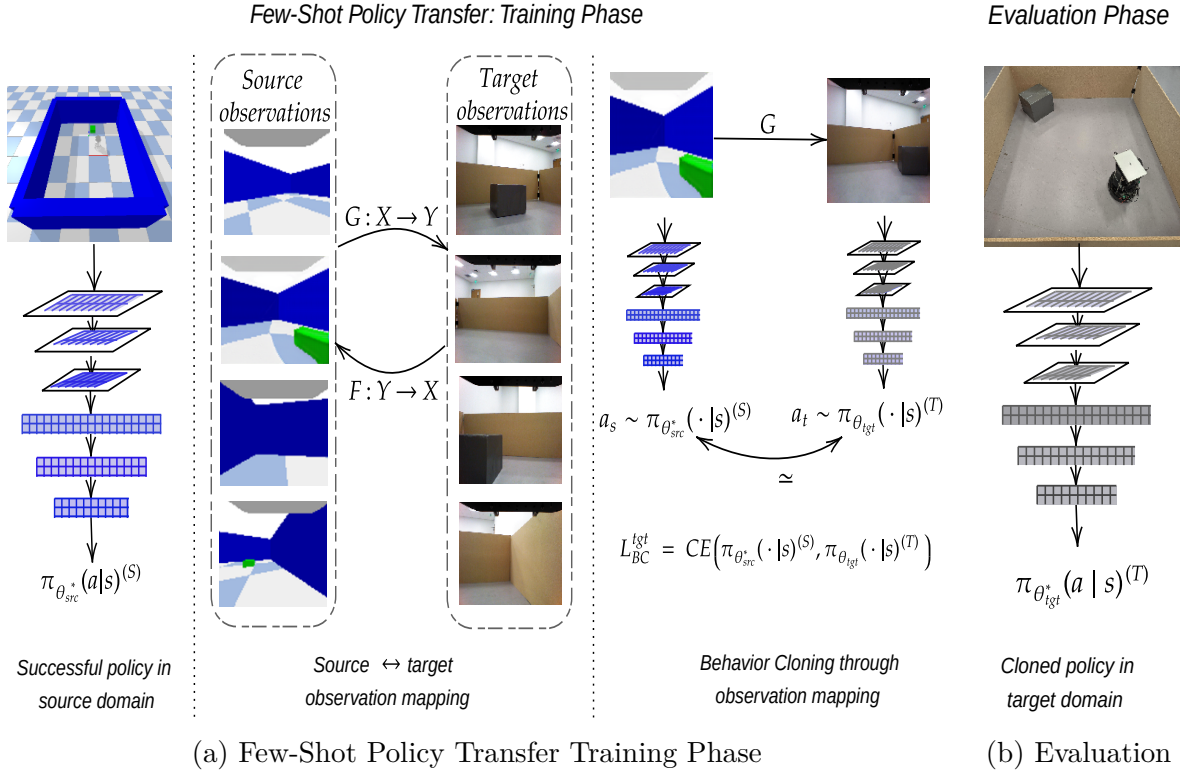


Figure 6.1: Overview of the Few-Shot Policy Transfer approach. During training, the agent learns a mapping between the observations of the source and target domain and clones the successful source task behavior policy. During the evaluation, the agent tests the cloned policy on the target domain.

6.2 Related Work

Transfer Learning (TL) uses knowledge from learned tasks and transfers it to a target task [128, 145]. In RL settings, one popular TL technique is to transfer the source task value function parameters to initialize the value function of the target task [116, 60, 65]. Policy transfer is another popular approach in which the policy learned in the source task is used to initialize the policy for the target task [58, 114, 128].

Behavior Cloning [6, 75] intends to learn a model that imitates expert demonstrations. In Behavior Cloning, the agent is provided with the raw observations and the actions taken by the expert demonstrator, and the algorithm trains a model that can predict the action given the observation during inference time. In BC, the agent does not need any additional training during test time and can generalize to novel in-distribution states during testing. Recent works have used BC for a wide range of applications, including training a quadcopter to fly along a path in a forest [103] and for autonomous

driving [9, 30]. Here, each observation is mapped with the corresponding action to perform behavior cloning. In the aforementioned works, the goal is to learn a policy for the same task as the expert demonstrations belong to. We are interested in cross-domain behavior cloning, where the task MDP of the target domain does not match the task MDP in which expert demonstrations are collected.

6.3 Theoretical Framework

In this section, we discuss the background information and how it fits into our few-shot policy transfer framework.

6.3.1 Markov Decision Processes

An episodic Markov Decision Process (MDP) M is defined as a tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma, \mathcal{S}_0, \mathcal{S}_f)$, 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, and $\mathcal{S}_0$ and $\mathcal{S}_f$ are the sets of starting states and terminal states, respectively. At 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_{θ^*} , maximizing its discounted return $R_0 = \sum_{k=0}^K \gamma^k r(s_{k+1}, a_k, s_k)$ until the end of the episode at timestep K .

6.3.2 CycleGAN

Generative Adversarial Networks (GAN) [40] are a set of two networks, in which a generator and a discriminator contest with each other in a zero-sum game. The goal of the generator is to generate new fake datapoints that fool the discriminator, whereas the goal of the discriminator is to classify real datapoints from the fake ones produced by the generator. CycleGANs [144] are an extension of GANs that learn a mapping between two image domains X and Y , through unpaired image samples $x \in X$ and $y \in Y$. In our experiments, the two observation spaces X and Y correspond to the observation datasets from source domain and the target domain, respectively. The role of the generators in

CycleGAN is to learn two mappings, $G : X \rightarrow Y$, a mapping from domain X to domain Y and $F : Y \rightarrow X$, a mapping from domain Y to the domain X . While the role of the discriminators D and E is to distinguish real source task images x from the adapted target task images $F(y)$; and the real target task images y from the adapted source task images $G(x)$. The loss for the source to target mapping is:

$$\mathcal{L}_{X \rightarrow Y}(G, E, X, Y) = \mathbb{E}_{x \sim X} [\log(1 - E(G(x)))] + \mathbb{E}_{y \sim Y} [\log E(y)]$$

Thus, the objective of the generator G is to convert images from domain X to resemble images from target domain Y by minimizing the above objective, and the discriminator E tries to maximize the above objective, giving us the optimization problem $\min_G \max_E \mathcal{L}_{X \rightarrow Y}(G, E, X, Y)$. The cycle-consistency loss of CycleGAN encourages pixel-wise consistency of images in the source task x and the adapted source task image $F(G(x))$, and of the target task image y and the adapted target task image $G(F(y))$:

$$\mathcal{L}_{cyclic}(G, F) = \mathbb{E}_{x \sim X} d(x, F(G(x))) + \mathbb{E}_{y \sim Y} d(y, G(F(y)))$$

where the distance metric d is the mean-squared error. CycleGAN promotes generating fake datapoints that lie in distribution, without access to a paired mapping function.

6.3.3 Behavior Cloning

Behavior Cloning (BC) aims to learn a policy for a task that matches the given expert demonstrations. Using the expert demonstrations (set of N state-action pairs $\{(s_i, a_i)_{i=1}^N\}$), BC performs maximum likelihood estimation, using a classifier or a regressor to determine the parameters of the model that would fit the expert demonstrations. The maximum likelihood problem of BC can be defined as:

$$\theta^* = \operatorname{argmax}_{\theta} \prod_{i=0}^N \pi_{\theta}(a_i | s_i)$$

where θ are the parameters of the network π_{θ} . The goal of the gradient in BC is to change θ such that the probability of the expert action (a_i) increases in the imitation policy's distribution $\pi_{\theta}(\cdot | s_i)$, based on the expert's policy distribution.

6.4 Problem Formulation

Let us consider a source task $M^{(S)}$ defined using the MDP

$(\mathcal{S}^{(S)}, \mathcal{A}^{(S)}, p^{(S)}, r^{(S)}, \gamma^{(S)}, \mathcal{S}_0^{(S)}, \mathcal{S}_f^{(S)})$. Similarly, the target task $M^{(T)}$ is defined using the MDP $(\mathcal{S}^{(T)}, \mathcal{A}^{(T)}, p^{(T)}, r^{(T)}, \gamma^{(T)}, \mathcal{S}_0^{(T)}, \mathcal{S}_f^{(T)})$.

The source and the target domains S and T have different set of states $\mathcal{S}^{(S)} \neq \mathcal{S}^{(T)}$. The set of actions in the source and the target domains need not be consistent, but are defined using a mapping $\mathcal{X}_{\mathcal{A}}$ that maps actions in the source task $\mathcal{A}^{(S)}$ to actions in the target task $\mathcal{A}^{(T)}$. We assume similar transition dynamics and reward functions in the two domains; $p^{(S)} \approx p^{(T)}$ and $r^{(S)} \approx r^{(T)}$.

The agent also has access to two trajectory datasets, one in the source domain $\mathcal{D}^{(S)} = \{(s_t^i, a_t^i, s_{t+1}^i)_{i=1}^n\}$ and the other in the target domain $\mathcal{D}^{(T)} = \{(s_t^i, a_t^i, s_{t+1}^i)_{i=1}^m\}$, where n and m are the number of datapoints in the two datasets. The datasets are collected using randomly initialized behavior policies and may not encapsulate all possible states in the two domains. Additionally, the learning agent does not have access to the policies that collected the datasets. In this work, each state in the datasets is an egocentric image collected by a camera mounted on the robot. The trajectory dataset in the target domain is static, i.e., no further interactions are possible in the target domain for training a policy in the target domain. We make these assumptions to address real-world situations where the learning agent only has access to an offline dataset, and interacting with the target domain requires an expensive setup. The goal of the agent is to learn a behavior policy in the source task and transfer it to the target task such that the zero-shot performance of the transferred policy achieves expected episodic return $R^{(T)} \geq \delta$ in the target task, where δ is predetermined threshold performance.

6.5 Few-shot Policy Transfer Approach

The Few-Shot Policy Transfer Approach consists of three sections: (1) Obtaining a successful behavior policy for the source task; (2) Generating a mapping between the source and target task observations, and (3) Cloning the source task policy to the target task

through the mapping.

The overall approach of our method is given in Algorithm 8. The goal is to transfer a behavior policy from a source task to a target task that has semantic and observational dissimilarities. The agent has access to the source task simulator and also has access to two static datasets (one in the source domain and one in the target domain) collected from a randomly initialized behavior policy.

First, we use Proximal Policy Optimization [109] to learn a successful behavior policy for the task in the source domain (line 1). In the source domain, the cost of interactions is low (e.g., a simulator) and hence we can reasonably learn a behavior policy for the task. The RL algorithm takes in the egocentric image view of the robot as the input and returns the action that the robot should take according to the policy. The next step entails learning an observation mapping between the states provided in the source and target domain datasets. Given the two static datasets $\mathcal{D}^{(S)}$ and $\mathcal{D}^{(T)}$, where each state in the dataset is an image, we use CycleGAN to learn the mapping between the images (line 2). CycleGAN does not require paired image data between the two domains to learn a mapping, and can be used when datasets are collected using randomly initialized policies. Additionally, the mapping does not assume a temporal relationship between the individual samples; which allows us to use *i.i.d* (independent and identically distributed) datasets. The output of the learned mapping function is a tuple consisting of the two generators and two discriminators. The two generators, G and F , provide a source→target and a target→source mapping respectively, whereas the two discriminators, D and E learn a classification between a real image captured by the camera on the robot and a fake one generated by the generators. Next, we incorporate this mapping and learn a target task policy by cloning the source task policy. Cloning the policy from the source task to the target task involves minimizing the action probability distribution mismatch between the learned source task policy and the target task policy.

At each time step, the source task state from the simulator is passed through the mapping generator G to produce a corresponding state in the target task (line 7). The ground truth action probability distribution for this generated target task state is the

Algorithm 8 Policy_Transfer($M^{(S)}, \mathcal{D}^{(S)}, \mathcal{D}^{(T)}, \delta_{BC}$)

Output: Target task behavior policy: $\pi_{\theta_{tgt}^*}$

```
1:  $\pi_{\theta_{src}^*} \leftarrow \text{Learn}(M^{(S)})$  ▷ Learn a source task behavior policy
2:  $G, F, D, E \leftarrow \text{CycleGAN}(\mathcal{D}^{(S)}, \mathcal{D}^{(T)})$  ▷ Learn source  $\leftrightarrow$  target mapping
3:  $\pi_{\theta_{tgt}} \leftarrow \text{Initialize\_Policy}()$ 
4: while True do
5:    $s^{(S)} \leftarrow \text{Environment.Reset}(M^{(S)})$ 
6:   while True do
7:      $s^{(T)} \leftarrow G(s^{(S)})$  ▷ Adapted state in target domain
8:      $L_{BC}^{tgt} = \text{Cross} - \text{Entropy}(\pi_{\theta_{src}^*}(\cdot | s^{(S)}), \pi_{\theta_{tgt}}(\cdot, s^{(T)}))$  ▷ Minimizing action distribution mismatch
9:      $\pi_{\theta_{tgt}} \leftarrow \text{Update\_Network}(L_{BC}^{tgt}, \pi_{\theta_{tgt}})$ 
10:     $s^{(S)}, rew, done = \text{Environment.Step}(s^{(S)}, a^{(S)})$ 
11:    if done then
12:      break
13:    end if
14:  end while
15:  if  $L_{BC}^{tgt} < \delta_{BC}$  then
16:     $\pi_{\theta_{tgt}^*} \leftarrow \pi_{\theta_{tgt}}$ 
17:    break
18:  end if
19: end while
20: return  $\pi_{\theta_{tgt}^*}$ 
```

action probability distribution provided by the learned source task policy. Since the transition dynamics and the goal of the tasks in the source and the target domain are approximately equivalent, the resulting states of the agent in the source and the target domain after taking equivalent actions should have correspondence (line 10-12). Thus, the behavior cloning loss is simply the cross entropy loss between the source task and target task action probabilities. Through the mentioned few-shot policy-transfer approach, we intend to generate a target task policy whose output action distribution matches the source policy’s action distribution. When the behavior cloning loss converges to a pre-determined threshold value, we attain a successful target task behavior policy and then evaluate this policy on the target domain.

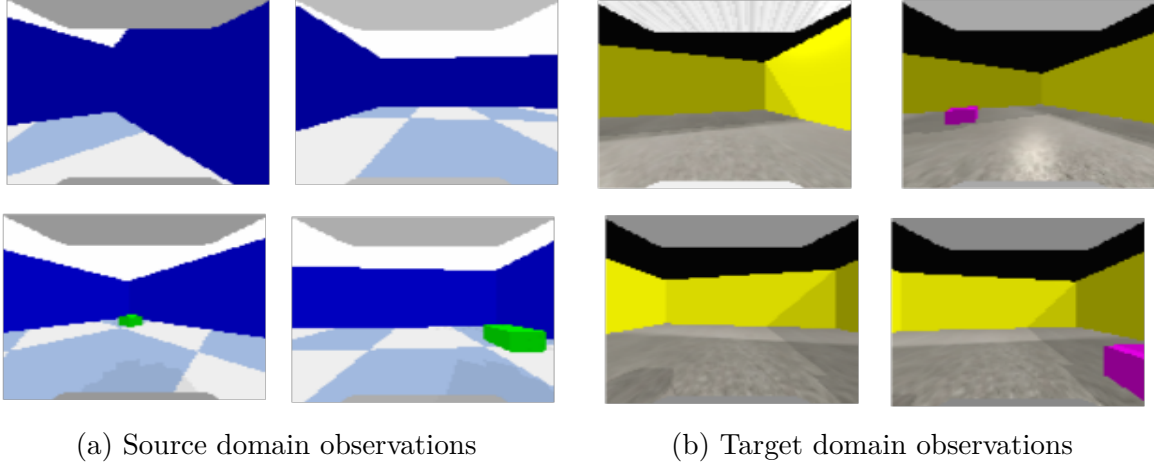


Figure 6.2: Examples of source and target domain observations. The observations are images retrieved from an ego-centric camera attached to the robotic agent.

6.6 Experiments and Results

Through experiments, we aim to answer: (1) Can we perform a successful few-shot policy transfer when there exists a mismatch in the observations between two domains? (2) Given the costly interactions in the target domain, how does this approach scale to different numbers of interactions? (3) Can it scale to target environments that are semantically different in the observations and the action spaces? (4) How does this approach perform for Sim2Real transfer?

6.6.1 Sim2Sim Transfer with Observation Mismatch

To answer the first question, we evaluated our few-shot policy transfer approach on two simulated domains with a significant mismatch in the observations. In both domains, the goal of the agent is to reach the goal location, which is denoted by a box in the environment. The environment is partially observable, i.e., at each time step, the agent observes an egocentric view image using the camera mounted on the robot (Fig 6.2). The agent is a Turtlebot acting in a simulated environment, implemented using PyBullet [18]. The set of discrete navigation actions available for the robot are moving forward and backward by $0.08m$, and turning left and right by $\frac{\pi}{10}$ radians. In each episode, the goal is spawned randomly within the bounds of the environment ($[2m \times 2m]$). The agent acts in a sparse reward setting, and achieves a positive reward of $+1000$ once it reaches the

Source-to-Target Transfer Approach	Success Rate
Direct Policy Transfer	$2\% \pm 2$
GAN+Behavior Cloning	$68\% \pm 10$
CycleGAN+Behavior Cloning (Ours)	$94\% \pm 4$

Table 6.1: Success rate comparison of different Source-to-Target Transfer Approaches. For the GAN+Behavior Cloning, and for CycleGAN+Behavior Cloning, 4000 source and target task images were used for generating the mapping.

goal object, and receives a -1 reward at all other timesteps. In this episodic setting, the episode ends when the robot achieves the goal, or reaches the maximum permissible timesteps (100) in an episode. For the first experiments, the only difference between the source and the target domains is a mismatch in the pixel-level agent observations, as seen in Fig 6.2. RL agents are very fragile to changes in the observations. Even a minor pixel level change can lead to catastrophic results [99]. Thus, a direct policy transfer from the source to the target domain yields poor results.

We begin by learning a policy that achieves a 98% success rate on the source task shown in Fig 6.1a. To learn the policy, we use Proximal Policy Optimization [109]. The structure of the model involves three convolutional layers of size 64 followed by three linear layers. Once we have a successful source task policy, the next step entails learning a mapping between the source and target dataset observations using CycleGAN. The source and target dataset consisted of 4000 state images taken using a random behavior policy on the two domains. CycleGAN outputs two generator functions that map images between the source and target domains. Employing this mapping function, we perform behavior cloning to learn the policy for the target domain, as defined in Section 6.5. The experiments were conducted using a 64-bit Linux Machine, with Intel(R) Core(TM) i9-9940X 3.30GHz processor, 126GB RAM, and a NVidia RTX 2080 GPU.

The results from the experiments are shown in Table 6.1 . We observed that the few-shot performance of the transfer yielded a success rate of 94% when evaluated on 100 episodes in the target domain. We compared this approach with two baseline approaches, a direct policy transfer approach, where the trained source policy was directly transferred to the target task, and a few-shot policy transfer through GAN and behavior cloning.

The results from the table show that a few-shot policy transfer through Cycle-GAN achieves higher accuracy (average higher accuracy by 26%) as compared to the baseline approaches. The cycle-consistency loss acts as a regularization for the generators, and guides the image generation process in the target domain closer toward image translation. The baseline GAN+BC did not employ the cycle-consistency loss while learning a mapping between the source and target images. This led to poor mapping and hence a poor policy in the target domain. Direct policy transfer did not show any reasonable success in the target domain.

6.6.2 Comparison with Different Numbers of Interactions

To answer the second question, we generate a mapping between the observations using different sizes of target datasets. The objective of this experiment is to determine how our proposed few-shot policy transfer approach compares when the number of datapoints available in the target dataset is limited. This is particularly useful when interactions in the target dataset require an expensive setup or are not easily attainable. For this experiment, we trained separate CycleGAN models with varying numbers of images in the target dataset, but a fixed number of images in the source dataset (4000 images). The results of this experiment are shown in Fig 6.3. As the size of the target dataset increases, the success rate of the task in the target domain increases, thus verifying that more images in the target dataset translate to a better mapping which in turn translates to an accurate target task representation. However, we see that even with only 250 interactions, the few-shot policy transfer approach achieves a 74% success rate, which can be improved with online fine-tuning on the target task. Our proposed approach outperformed the other baseline approach, GAN+BC, which did not employ the cycle-consistency. Our approach has a mean success rate of 94% as opposed to 68% from the baseline approach. The number of datapoints required to achieve a successful mapping increases as the number of objects in the task increases. From our experiments, ~ 200 images per object are sufficient for achieving a good mapping.

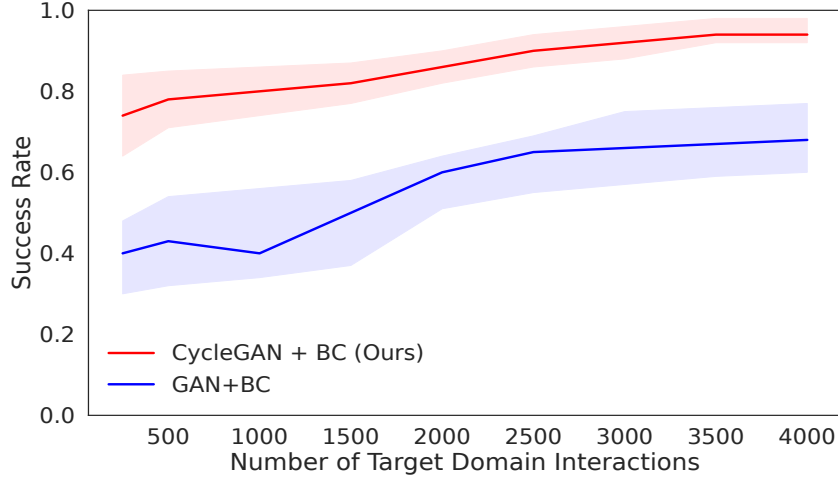


Figure 6.3: Success rate (Mean $\pm$ SD) comparison of the three TL approaches for different target dataset sizes. More target datapoints results in a better mapping between the observations of the two domains, resulting in a better knowledge transfer.

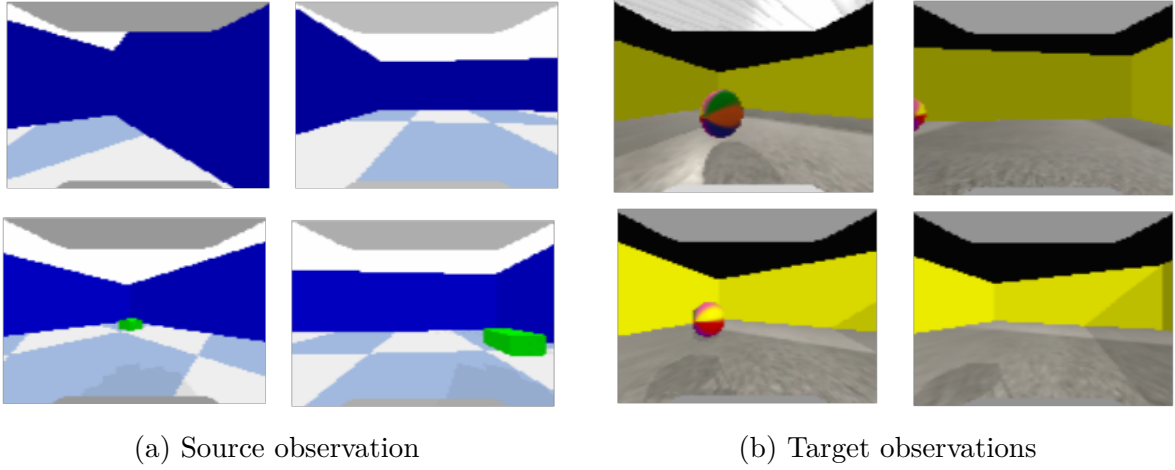


Figure 6.4: Examples of source and target domain observations.

6.6.3 Semantically Different Source and Target Environments

To answer the third question, we evaluated our few-shot policy transfer approach on a target domain that is semantically different than the source domain in the observation and the action spaces. In this target domain, the goal of the agent is still to navigate to a goal object in the environment, but the object is a spherical ball that does not have a fixed base, as shown in Fig 6.4. Whereas, in the source domain, the goal is to navigate to a box, with observations shown in Fig. 6.2a. Unlike the goal object in the source task, the spherical ball produces different images when observed from different angles. This makes it difficult to attain a successful mapping with a limited number of images. Additionally,

in the target task, the actions have finer discretizations, with a forward and backward movement causing the agent to move $0.03m$, and the rotation movements causing the agent to rotate $\frac{\pi}{15}$ radians instead of the actions described in section 6.6.1. The source and target dataset consisted of 4000 images taken using a random behavior policy on the two domains. The results from this experiment are summarized in Table 6.2. We observe that the direct policy transfer is unable to generalize to the target task. The semantic difference between the two domains in terms of observations, actions, and transitions makes it very difficult for direct policy transfer and GAN+BC to accomplish the target task. Our proposed few-shot approach performs much better than the baseline approaches and achieves a 78% success rate on the target task.

Source-to-Target Transfer Approach	Success Rate
Direct Policy Transfer	0%
GAN+Behavior Cloning	20%
CycleGAN+Behavior Cloning (Ours)	78%

Table 6.2: Success rate comparison of different Source-to-Target Transfer Approaches. For the GAN+Behavior Cloning, and for CycleGAN+Behavior Cloning, 4000 source and target task images were used for generating the mapping.

6.6.4 Sim2Real Transfer on a Physical TurtleBot

To answer the fourth question, we evaluated our few-shot policy transfer approach on a Sim2Real task, where the goal of the robot in the real world was to reach the box as shown in Fig. 6.5a. The domain consisted of the robot navigating in an enclosed arena of dimensions $[2.45m, 2.45m]$. The dimensions of the goal object (box) are $[17in \times 19in \times 12in]$. We introduced a mismatch in the dimensions of the arena as well as the goal object to prevent an exact correspondence between the simulation and the real-world domains. For the real-world task, we made use of the TurtleBot 2 with an on-board laptop (running Ubuntu 18.04 and Robot Operating System Melodic [100]) which interfaces with a camera (Orbbec Astra). The laptop has an Intel(R) Core(TM) i7-1165G7 2.8GHz processor and 8GB RAM. The robot is equipped with an RPLIDAR A2M8 360° Laser Scanner, but

our observations consisted of only RGB images. Using the depth information will further enhance the observations, and can be used for tasks where depth information is necessary.

Along with a mismatch in the observations, we introduced a mismatch in the actions of the agents in the real-world and the simulated environment. In the real-world, the agent has access to four navigation actions, namely, move forward and backward ($0.04m$), and rotate clockwise and counter-clockwise ($\pi/100$). The dynamics of the source domain for this Sim2Real task are described in Section 6.6.1. The static dataset for the real-world task consisted of 4000 images taken using a random behavior policy on the robot. The entire dataset was collected in a span of 2 hours on the physical robot. Examples of real-world observations are shown in Fig. 6.5b

A subset of the images do not contain the goal object, and in order to reach the goal object, the robot needs to rotate to find out the location of the goal object.

Even with a significant mismatch in the observation and action spaces of the agent in the simulated domain and the real-world domain, we observed a 90% accuracy of the robot in the real-world. We evaluated our few-shot policy transfer approach on 30 episodes in the real-world, where the robot was able to reach the goal within the maximum number of permissible timesteps in 27 out of the 30 episodes. Our agent’s latency was approximately 1 second (time taken for the agent to capture an image and generate an action after doing a forward pass). In scenarios where the robot was unable to face the goal object, it learned to rotate itself to face the object successfully. Once it was able to locate the goal object in its observation, it was able to navigate and reach the goal object. However, in certain cases, the model was unable to differentiate between shadow caused by the walls and the box itself, because of the similarity in the RGB composition of the shadow and the box. In those scenarios, the robot either took longer to reach the goal or was unable to reach the goal in the predetermined number of timesteps. Here, enriching the observations with depth information will help the robot differentiate between the goal object and the shadow, and help in improving the accuracy.

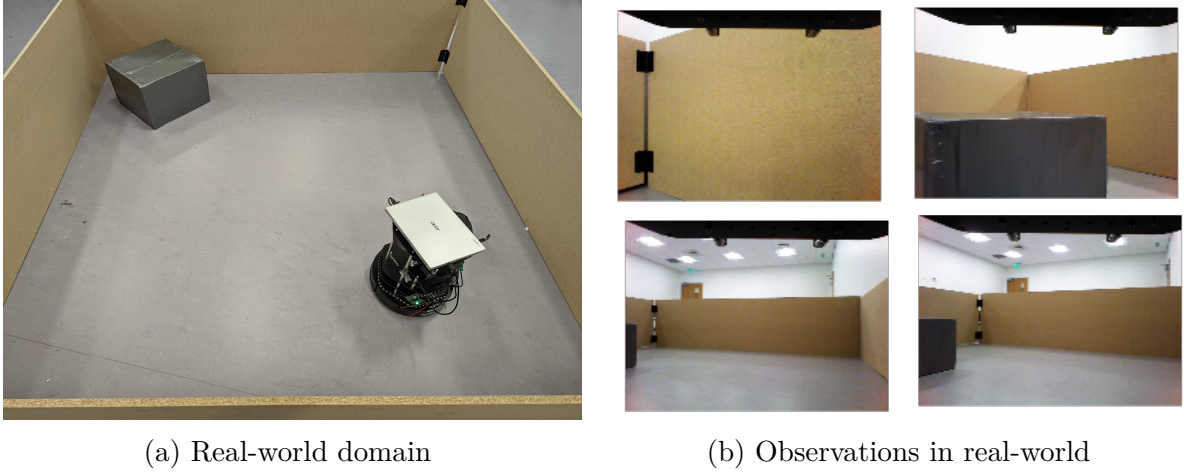


Figure 6.5: Real-world domain (left) and egocentric observations from the robot (right). The robotic agent needs to navigate to the grey box placed randomly in the environment.

6.7 Summary

In this work, we proposed a framework for few-shot policy transfer through observation mapping and behavior cloning. We used CycleGAN to generate the mapping between two dissimilar domains and then used behavior cloning to transfer the learned policy from one domain to another. We observed a successful few-shot transfer even when a limited number of interaction datapoints were available from the target domain. As the number of available target domain datapoints increases, the success rate performance increases. Additionally, when the task goal semantics and the action space parameters were changed in the target domain, we observed a 78% few-shot success rate, which increased online fine-tuning in the target domain. Finally, we performed a Sim2Real transfer, by collecting a static dataset of images in the real-world domain, and then using our few-shot approach for policy transfer. Our method achieved the goal reaching behavior in 27 out of the 30 trials, demonstrating a real-world goal-reaching success rate of 90%.

Our framework for the few-shot policy transfer addressed a simple robot navigation scenario. For complicated tasks, generating a mapping between the observations in the two domains might require changes to the CycleGAN design. This technique of knowledge transfer can be combined with the previous techniques of curriculum generation and curriculum transfer to learn successful policies for complex temporally extended tasks in high-fidelity settings.

Chapter 7

Conclusion and Future Work

Enhancing the sample efficiency of reinforcement learning agents by incorporating symbolic reasoning in the decision making is a challenging aspiration. However, the central challenge is the lack of techniques that can facilitate a successful synergy between the symbolic information and the reinforcement learning agents. The challenge arises due to different sensors, different embodiments, and different capabilities that are available to agents that use symbolic information as compared to reinforcement learning agents. These challenges create a hurdle in the efficient use of symbolic information in reinforcement learning agents. This dissertation tackles the challenges related to incorporation and efficient use of symbolic information for curriculum generation, curriculum transfer and knowledge transfer in reinforcement learning agents. This dissertation introduced three novel techniques for curriculum generation using symbolic knowledge either in the form of object oriented markov decision processes, linear temporal logic, or large language models. Furthermore, this dissertation proposed two techniques - one aimed at transferring the schema of the curriculum between two different fidelities of environments, and one aimed at transferring the policy between simulated and real environments, with minimal real world data. Through experiments in gridworld domains, simulated robotic environments and physical robots, this dissertation underscores the efficacy of the proposed frameworks. The empirical outcomes not only validate the applicability of the approaches, but it also

emphasizes the potential to equip newer reinforcement learning agents, allowing them to be sample efficient when symbolic information is readily available.

7.1 Summary of Contributions

In response to the central question, *How can symbolic information obtained from temporal logic, large language models, and human generated task guidance help in designing automated curricula in a low fidelity setting and how can the curricula and policy information from the low fidelity be transferred to a high fidelity setting to improve the sample efficiency of RL agents?*, this dissertation has made the following significant contributions:

7.1.1 Utilizing Temporal Logical Specifications for Efficient Curriculum Generation in RL

In Chapter 3, we introduced Automaton-guided Curriculum Learning (AGCL), which uses LTL_f specifications and DFAs to automatically generate curricula for RL agents. Our approach improves sample efficiency by efficient knowledge transfer and parallel task learning through the graphical representation of sub-goals. Through extensive experiments in gridworld and robotics environments, we show that AGCL significantly outperforms existing methods, demonstrating improved sample efficiency.

7.1.2 Curriculum as a Dynamic Sub-Task Selection Strategy Using Temporal Logic and LLMs

In Chapter 4, we introduce two approaches: LSTS, which uses formal logical specifications to guide RL agents through sub-goals via a Teacher-Student curriculum learning method, and LgTS, which uses LLMs to automatically generate these sub-goal structures. Our proposed methods do not rely on a dense reward function to guide the learning progress of the agent. Experimental results demonstrate that both approaches achieve faster convergence (by one order of magnitude on a gridworld task) than traditional methods.

7.1.3 Curriculum Schema Transfer between Fidelities Using Object-Oriented MDPs

In Chapter 5, we formulated 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). Our approach, “ACuTE” is a novel framework aimed to solve this problem. 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, and that our approach is also independent of the learning algorithm used for curriculum generation.

7.1.4 Sim2Real Using Observation Mapping and Behavior Cloning

In Chapter 6, we introduced a framework for Few-Shot Policy Transfer between simulated and real domains through Observation Mapping and Behavior Cloning. We use Generative Adversarial Networks (GANs) along with a cycle-consistency loss to map the observations between the source and target domains and later use this learned mapping to clone the successful source task behavior policy to the target domain. We observe successful behavior policy transfer with limited target task interactions and in cases where the source and target task are semantically dissimilar.

7.2 Interconnections among Proposed Frameworks

The three components presented in the dissertation, namely, curriculum generation, curriculum schema transfer and knowledge transfer through policy functions are not standalone components, but are intricately connected through a common thread. While each component and framework tackles distinct challenges and proposes and advances distinct methodologies, their unifying goal is to enhance the sample efficiency of reinforcement learning agents, particularly in environments where data collection is expensive. Sample efficient RL ultimately leads to robust behavior policies, enhanced decision making under

uncertainty, and generalization to a diverse set of tasks.

For the first component, curriculum generation, this dissertation introduces three novel frameworks of generating a curriculum depending on what higher-level symbolic information the agent has access to. The first framework is automaton-guided curriculum generation using object oriented markov decision processes (MDPs) as the symbolic information to generate a sequential and a graph-based curriculum. The second and third framework for curriculum generation formulates curriculum learning as dynamic task selection using high-level specification language and large language models as the source of symbolic information.

For the second component, curriculum schema transfer, this dissertation introduces a new automated curriculum transfer technique. This technique employs object oriented MDPs to learn a mapping between a high-fidelity and a low-fidelity environment. The curriculum generated in the low-fidelity can then be transferred to the high-fidelity setting through this OOMDP mapping. The curriculum transfer approach is agnostic to the curriculum generation procedure and any of the techniques introduced in the curriculum generation framework can be used.

For the third component, simulation to real knowledge transfer, this dissertation introduces a new policy transfer technique between simulated environment and physical environment. The technique uses CycleGAN, an unsupervised technique to learn a mapping between the images in the simulated and the real environment. The mapping does not require paired labeled representations between the two environments, making the knowledge transfer efficient. The policy transfer technique can be combined with curriculum generation and curriculum schema transfer to achieve temporally extended long horizon objectives.

To summarize, the aforementioned components are not individual contributions, but constituents of a unified strategy that emphasizes sample efficient learning for reinforcement learning agents in high fidelity settings. Together they make significant contributions to promote the applications of reinforcement learning, advancing the employment of RL for sample efficient and robust decision making across diverse tasks and domains.

7.3 Applications

Below we highlight certain applications where the proposed methods exhibit efficiency.

7.3.1 Requirement of Robust Policies

Reinforcement Learning is specifically used when the environment is hard to model and the requirement of robust policies is a central concern. However, learning policies using RL is sample inefficient, making it harder to use RL for practical scenarios. The frameworks introduced in this dissertation are aimed for applications where sample efficiency is a key metric. The frameworks utilize symbolic knowledge to reason about sub-goals and sub-tasks to generate a curriculum. We further demonstrate that learning through a curriculum requires fewer samples to attain a successful behavior policy as compared to no curriculum.

7.3.2 Expensive Real-World Data Collection

Collecting real world data is expensive. The reason can be attributed to a number of factors - including time to generate the data, cost to reset environments to repeat the data collection procedure, etc. The frameworks proposed in the work are suitable when the bottleneck is the data collection procedure. Transferring knowledge from a policy trained in simulation to a real world without requiring extensive data collection in the real world mitigates the cost associated with data collection.

7.3.3 Goal Objectives Known *apriori*

The techniques developed in this dissertation require knowledge about the goal objectives that can be depicted through symbolic knowledge. Complementing the learning progress of the RL agents through knowledge derived from symbolic information based on goal objectives helps improve the training, and produces robust policies. Our approaches do not generate a dense reward function, and hence do not suffer from optimal policy design due to suboptimal reward functions.

7.3.4 Expensive Automated Curriculum Generation

In certain scenarios, generating an automated curriculum can be more expensive than training the agent without any curriculum. Our proposed approach tackles this challenge by mapping the environment to its low-fidelity (LF) representation using symbolic knowledge, and generates a curriculum in this low-fidelity environment. Generating a curriculum in LF is inexpensive due to simplified dynamics and essential rendering. The generated curriculum is mapped back to the high-fidelity environment, giving us a curriculum in HF with minimal to no interactions in this expensive HF environment.

7.4 Boundaries

Below we highlight certain boundaries of the proposed methods.

7.4.1 Availability of Symbolic Information

The curriculum generation approaches assume availability of symbolic information. In certain cases, the task may be completely unknown or the symbolic information may be hard to obtain. In such scenarios, the curriculum generation approaches we mentioned might not be able to come up with an effective curricula for sample efficient training. In such scenarios, other techniques in the literature (e.g., meta-learning) would be better suited that do not require symbolic information to learn sample efficient policies.

7.4.2 Dynamic Task Generation

The curriculum generation procedures assume a distribution of tasks that can be generated by varying a set of parameters. The set of parameters is obtained using OOMDP approaches. Varying the parameters is not always feasible, as the tasks can be fixed with no room for sub-task generation. In such scenarios, curriculum generation procedures that rely on OOMDP to come up with a sequence of sub-tasks of increasing difficulty will not perform effectively. In such cases, dynamic task sampling approaches must be used, as they do not require sub-task generation.

7.4.3 Generation of Low-Fidelity Environments

The proposed curriculum schema transfer approach maps the generated curriculum from a low-fidelity environment to a high-fidelity environment. Obtaining a low fidelity environment requires discretizing continuous environment parameters in the high-fidelity environment, which may not be feasible. In such scenarios, the only way to generate an automated curriculum would be in the high-fidelity environment, which can be computationally expensive.

7.4.4 Availability of Real-World Data Samples

Policy transfer approach maps the images obtained in the simulated environment to images from the real world. The mapping does not require paired transitions, however, it requires a few images collected from the physical robot. Absence of images from the physical robot or a substantially different simulator as compared to the real environment would generate a poor observation mapping, making the policy transfer process unsuccessful.

7.4.5 Guarantees on Optimality

Our curriculum generation and schema transfer approaches are aimed at improving the sample efficiency of RL agents. These techniques do not study whether the RL policy attained using these approaches is optimal in terms of success rate or other quantifiable metrics. The approaches may also not be the most sample efficient way to generate the behavior policy.

By exploring the applications and boundaries, this dissertation provides a thorough comprehension of the proposed frameworks, their use-cases in RL and Robotics, and their assumptions and limitations.

7.4.6 Limited Scope of Robotic Policies

One limitation of the approaches discussed in the dissertation is the lack of direct extension to physical robotic scenarios. In this dissertation, the robotic domains discussed are turtlebot navigation, and simulated pick and place for a manipulator robot. The RL policies for such domains cannot directly be extrapolated to real world scenario because of simplified environmental dynamics assumed in the simulation environment. The simulated environments are constrained in terms of the types and number of entities that can appear in the environment. Such an assumption is not true in practical real world scenarios as there are several avenues of uncertainty - sensor uncertainty, action uncertainty etc. A few ways of dealing with such real-world higher fidelity scenarios is listed as a future work 7.5.8

7.5 Future Work

This dissertation introduced comprehensive approaches to enhance the sample efficiency of reinforcement learning agents through symbolic reasoning. In this section, we discuss open questions and promising future avenues that require further investigation.

7.5.1 Automated Generation of Low-Fidelity Environments

This dissertation focused on generating the curriculum in low-fidelity and mapping the schema of the curriculum to high-fidelity. The work assumed the existence of a suitable low-fidelity environment, where curriculum generation would be sample efficient. Obtaining the object oriented MDP parameters for certain high-fidelity environment would not always be feasible. For example, a robotic manipulator arm learning how to fold a deformable clothing object, such as a shirt. Using OOMDP to generate the low-fidelity representation might simplify the complexity involved in manipulating a deformable object. Thus, a curriculum generated with such a simplified LF environment will not be able to adapt to the complex real world scenario. Future work needs to investigate mapping the high-fidelity environment to a higher dimensional manifold representation, and generating samples of low-fidelity by sampling from this manifold. This technique

will eliminate the reliance on object oriented markov decision processes for environment representation.

7.5.2 Curriculum Transfer across Multiple Fidelities

The curriculum schema transfer approach mentioned in this dissertation works by transferring the schema of the curriculum from a low-fidelity environment to a high-fidelity environment. This idea can be extrapolated across multiple fidelities, where the curriculum is generated in an environment with the lowest fidelity, and progressively transferred to a higher-fidelity environment, ultimately culminating at the final target task with the highest fidelity. The spreading out of the environment across different fidelities can also be considered as a curriculum. This method is a direct extension of the curriculum schema transfer approach discussed in the dissertation, and needs to be investigated.

7.5.3 Seamless Integration of Neural and Symbolic Information

In this dissertation, the symbolic information was used to guide the learning progress of the reinforcement learning agent. The policy and the value functions of the RL agent did not directly incorporate any symbolic information. While the frameworks proposed in this dissertation improve the sample efficiency of the RL agents, a promising avenue for future work would be to integrate the symbolic information in the decision making of RL agents at the policy level [39]. This will promote the RL agent to optimize its objectives while also considering the symbolic information as a part of its state space. Integrating the symbolic information into the policy has the potential to further enhance the sample efficiency of RL agents.

7.5.4 Multi-modal Sim2Real Knowledge Transfer

Our framework on Simulation to real world policy transfer involved mapping the image observations between the simulated and the physical environments. While images collected from a camera mounted on the robot are useful in solving the task, they do not convey relevant information about the objects that are present outside the view of the

camera, nor any other auditory or tactile information about the objects present in the environment. This additional information will be particularly useful when images alone are insufficient in solving the task at hand. For example, if the task is finding a can with marbles in it from a group of similar looking cans, audio information should be used to distinguish the can with marbles from others. Future efforts should be directed to such multi-modal simulation to reality transfer techniques that can solve a wider range of tasks effectively.

7.5.5 Developing Simulators that Closely Replicate the Physical World

It is a well studied phenomenon that simulators that closely match the physics of the real world would be better adept at performing a successful simulation to reality transfer. Future avenues of work should look into developing simulators specifically catered for robotic applications that can successfully simulate fluids, deformable objects, objects with high shear friction, along with incorporating common sense reasoning in the simulators. Physics-based simulators help in incorporating the physics in the training objective. However, developing such high-fidelity simulators makes them computationally expensive and training and designing policies in such high-fidelity simulators becomes even more complex. While developing such simulators, further care must be taken to ensure the practicality of these simulators from a machine learning and reinforcement learning perspective.

7.5.6 Human-in-the-Loop Automated Curriculum Generation

Much of the work in this dissertation was focused on automated curriculum generation through symbolic reasoning. None of the proposed automated curriculum generation approaches actively involved human preferences [123] in designing the curriculum. Future work should investigate incorporation of human preferences in the curriculum design so that the generated curricula follow the guidelines set by a human user. This will enable the user to train policies according to their preference and lead to better generalization

of the policies over a wider distribution of preferences.

7.5.7 Different Types of Human Feedback for Improving Sample-Efficiency of RL Algorithms

In this dissertation, I discussed incorporating symbolic reasoning through temporal logic, large language models, and structured object-oriented MDPs to improve the learning progress of the RL agent. Below are a few other ways of incorporating human feedback within the learning objective of RL agents:

Concept-Based Feedback

Concept-based feedback uses natural language corrections to guide the RL agent. A human-user may indicate that a certain grasp by a robotic agent is ‘ungraspable’ or ‘unstable’ and this natural language feedback can be used to guide the agent to learn meaningful behaviors that are explainable and generalizable. Such explainable policies are useful in safety-critical environments.

Feedback through Analogical Reasoning

Feedback through Analogical Reasoning helps agent learn new tasks by drawing parallels from previously learned behaviors. For example, informing the agent that ‘stacking books’ is similar to ‘balancing blocks’ helps the agent learn previously learned behaviors for balancing blocks to learn a new policy for stacking books. A few demonstrations of the new task will help the agent adapt the new task rapidly.

Feedback through Counterfactual Reasoning

Counterfactual Reasoning helps the agent explore a ‘what-if’ scenario. For example, a human may suggest, ‘what-if you placed the bigger block at the bottom of the stack?’. Such a what-if question will nudge the agent in the correct line of reasoning, reduce the complexity of the search the agent was doing, and help it come up with a successful sub-task order sooner.

Feedback through Commonsense Reasoning

In certain scenarios, the agent may lack commonsense reasoning necessary for solving the task. For example, the following feedback from the human user, ‘cups can be stacked’ will help the agent reason over what is expected out of each sub-task and thus improve the overall efficiency of the learning.

Ethical Feedback

Ethical feedback integrates ethics, social norms and fairness into the agent’s learning progress. For example, a human may say, “Avoid making sudden movements near other humans” to a robotic agent. Such a feedback can be used by the agent to learn a behavior that avoids sudden movements near other humans to respect their comfort, safety and manage other expectations. Ethical feedback incorporates non-functional but essential values into the agent’s reasoning.

7.5.8 Employing Curriculum Generation and Transfer for Real-World Robotic Tasks

As mentioned in the Boundaries subsection 7.4.6, the RL policies for the domains and tasks discussed in this dissertation cannot directly be extrapolated to real world scenario because of simplified environmental dynamics assumed in the simulation environment. Here are a few ways of extending this work for complex dexterous robotic tasks:

Parameterizing Action Space

Instead of learning control policies that output actions at a controller level, a better approach would be to encode the actions using parameters and have the RL network only output the parameters. For e.g., instead of predicting the raw values of torques, the network predicts high-level actions such as pick-up or reach using only a few parameters relevant to the high-level actions.

While such parameterization increases the required human engineering effort, it reduces the load on the RL policy as it only needs to learn a few parameters instead of learning the entire complex behavior. Such a change will allow learning complex behaviors much more sample efficiently and will indirectly increase the complexity of task permitted using the techniques mentioned in the dissertation.

Incorporating Foundation Models for Action Learning

One approach is to use latent action models, where the policy operates in a learned action embedding space rather than directly predicting motor commands. For example, a variational autoencoder (VAE) or contrastive learning framework can be used to encode primitive actions into a latent space, and the policy learns to predict latent parameters rather than raw control outputs.

Additionally, motor primitive representations allow for more structured policy learning. Instead of predicting raw torques or velocities, the policy learns to adjust parameters of predefined movement primitives such as dynamic movement primitives (DMPs) or task-space policies. This enables smoother and more interpretable behavior, making real-world deployment more feasible.

By parameterizing the action space in this manner, policies can leverage prior knowledge about structured action representations, leading to improved sample efficiency, better transferability to real-world conditions, and reduced susceptibility to sensor and actuation noise.

Foundation Models will reduce the need to learn controller level policies using RL, and will enable execution of complex behaviors that require precise control, such as manipulating complex objects and completing long-horizon tasks (e.g., making a sandwich) that require reasoning over the action order along with the requirement for complex motion planning.

Employing Behavior Cloning for Warm-Starting RL

A dataset of expert demonstrations collected from human users can be used to train a behavior policy using a supervised learning approach. Once the agent learns the underlying behaviorally cloned policy, RL can further be used to fine-tune this policy, thereby improving generalization and robustness beyond what was available and learned through the demonstrations. One such technique is DAGGER [104], where an expert provides corrections iteratively reducing the scope for compounding errors in long-horizon tasks. Other techniques such as Advantage-Weighted Regression [93] integrate behavior cloning from expert demonstrations with RL, promoting policy improvement beyond demonstrations.

Bibliography

- [1] David Abel, Yuu Jinnai, Sophie Yue Guo, George Konidaris, and Michael Littman. Policy and value transfer in lifelong reinforcement learning. In *International Conference on Machine Learning*, pages 20–29. PMLR, 2018.
- [2] Mohammad Afzal, Sankalp Gambhir, Ashutosh Gupta, Ashutosh Trivedi, and Alvaro Velasquez. Ltl-based non-markovian inverse reinforcement learning. In *Proc. of the 2023 Intl. Conf. on Autonomous Agents and Multiagent Systems*, pages 2857–2859, 2023.
- [3] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.
- [4] Rajeev Alur, Suguman Bansal, Osbert Bastani, and Kishor Jothimurugan. A framework for transforming specifications in reinforcement learning. In *Principles of Systems Design*. Springer, 2022.
- [5] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, and Wojciech Pieter Abbeel, Zaremba. Hindsight experience replay. *NeurIPS*, 2017.
- [6] Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine Intelligence 15, Intelligent Agents [St. Catherine’s College, Oxford, July 1995]*, pages 103–129, 1999.

- [7] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19, 2006.
- [8] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proc. of the 26th annual Intl Conf. on machine learning*, pages 41–48, 2009.
- [9] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Praseen Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiaakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.
- [10] Alper Kamil Bozkurt, Yu Wang, Michael M Zavlanos, and Miroslav Pajic. Control synthesis from linear temporal logic specifications using model-free reinforcement learning. In *2020 IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 10349–10355. IEEE, 2020.
- [11] John D Bransford, Ann L Brown, Rodney R Cocking, et al. *How people learn*, volume 11. Washington, DC: National academy press, 2000.
- [12] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.
- [13] Mingyu Cai, Erfan Aasi, Calin Belta, and Cristian-Ioan Vasile. Overcoming exploration: Deep reinforcement learning for continuous control in cluttered environments from temporal logic specifications. *IEEE RAL*, 8(4):2158–2165, 2023. doi: 10.1109/LRA.2023.3246844.
- [14] Alberto Camacho, Oscar Chen, Scott Sanner, and Sheila A McIlraith. Non-markovian rewards expressed in ltl: Guiding search via reward shaping. In *GoalsRL, a workshop collocated with ICML/IJCAI/AAMAS*, 2018.

- [15] Thomas Carr, Maria Chli, and George Vogiatzis. Domain adaptation for reinforcement learning on the atari. In *Proc. of the 18th Intl. Conf. on Autonomous Agents and MultiAgent Systems*, pages 1859–1861, 2019.
- [16] Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for gymnasium, 2018. URL <https://github.com/Farama-Foundation/Minigrid>.
- [17] Felipe Codevilla, Eder Santana, Antonio M López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9329–9338, 2019.
- [18] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and ml, 2016–2019.
- [19] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. <http://pybullet.org>, 2021. Accessed: 2023-04-02.
- [20] Felipe Leno Da Silva and Anna Helena Reali Costa. A survey on transfer learning for multiagent reinforcement learning systems. *Journal of Artificial Intelligence Research*, 64:645–703, 2019.
- [21] Alessandro Daniele, Tommaso Campari, Sagar Malhotra, and Luciano Serafini. Simple and effective transfer learning for neuro-symbolic integration. In *International Conference on Neural-Symbolic Learning and Reasoning*, pages 166–179. Springer, 2024.
- [22] Giuseppe De Giacomo and Moshe Y Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI’13*, pages 854–860. Association for Computing Machinery, 2013.
- [23] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

- [24] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines, 2017.
- [25] Konstantinos Dimitropoulos, Ioannis Hatzilygeroudis, and Konstantinos Chatzilygeroudis. A brief survey of sim2real methods for robot learning. In *International Conference on Robotics in Alpe-Adria Danube Region*, pages 133–140. Springer, 2022.
- [26] Yan Ding, Xiaohan Zhang, Chris Paxton, and Shiqi Zhang. Leveraging common-sense knowledge from large language models for task and motion planning. In *RSS 2023 Workshop on Learning for Task and Motion Planning*, 2023.
- [27] Carlos Diuk, Andre Cohen, and Michael L Littman. An object-oriented representation for efficient reinforcement learning. In *25th ICML*, pages 240–247, 2008.
- [28] Yuqing Du, Olivia Watkins, Trevor Darrell, Pieter Abbeel, and Deepak Pathak. Auto-tuned sim-to-real transfer. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1290–1296. IEEE, 2021.
- [29] Jeffrey L Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99, 1993.
- [30] Wael Farag and Zakaria Saleh. Behavior cloning for autonomous driving using convolutional neural networks. In *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, pages 1–7. IEEE, 2018.
- [31] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.
- [32] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, pages 482–495. PMLR, 2017.

- [33] Francesco Foglino, Christiano Coletto Christakou, and Matteo Leonetti. An optimization framework for task sequencing in curriculum learning. In *Intl Conf. on Development and Learning*, 2019.
- [34] Jason R Frank, Linda S Snell, Olle Ten Cate, Eric S Holmboe, Carol Carraccio, Susan R Swing, Peter Harris, Nicholas J Glasgow, Craig Campbell, Deepak Dath, et al. Competency-based medical education: theory to practice. *Medical teacher*, 32(8):638–645, 2010.
- [35] Quentin Gallouédec, Nicolas Cazin, Emmanuel Dellandréa, and Liming Chen. panda-gym: Open-Source Goal-Conditioned Environments for Robotic Learning. *4th Robot Learning Workshop: Self-Supervised and Lifelong Learning at NeurIPS*, 2021.
- [36] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of machine learning research*, 17(59):1–35, 2016.
- [37] Yifan Gao and Lezhou Wu. Efficiently mastering the game of nogo with deep reinforcement learning supported by domain knowledge. *Electronics*, 10(13):1533, 2021.
- [38] Jennifer Gervais. The operational definition of competency-based education. *The Journal of Competency-Based Education*, 1(2):98–106, 2016.
- [39] Shivam Goel, Yash Shukla, Vasanth Sarathy, Matthias Scheutz, and Jivko Sinapov. Rapid-learn: A framework for learning to recover for handling novelties in open-world environments. In *2022 IEEE International Conference on Development and Learning (ICDL)*, pages 15–22. IEEE, 2022.
- [40] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

- [41] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.
- [42] Charles R Graham. Emerging practice and research in blended learning. In *Handbook of distance education*, pages 351–368. Routledge, 2013.
- [43] Marek Grzes. Reward shaping in episodic reinforcement learning. *AAMAS*, 2017.
- [44] Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Thomas Paine, Sergio Gómez, Konrad Zolna, Rishabh Agarwal, Josh S Merel, Daniel J Mankowitz, Cosmin Paduraru, et al. Rl unplugged: A suite of benchmarks for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:7248–7259, 2020.
- [45] Lewis Hammond, Alessandro Abate, Julian Gutierrez, and Michael Wooldridge. Multi-agent reinforcement learning with temporal logic specifications. *arXiv preprint arXiv:2102.00582*, 2021.
- [46] Daniel Ho, Kanishka Rao, Zhuo Xu, Eric Jang, Mohi Khansari, and Yunfei Bai. Retinagan: An object-aware approach to sim-to-real transfer. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10920–10926. IEEE, 2021.
- [47] Sebastian Höfer, Kostas Bekris, Ankur Handa, Juan Camilo Gamboa, Melissa Mozifian, Florian Golemo, Chris Atkeson, Dieter Fox, Ken Goldberg, John Leonard, et al. Sim2real in robotics and automation: Applications and challenges. *IEEE transactions on automation science and engineering*, 18(2):398–400, 2021.
- [48] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.
- [49] Rodrigo Toro Icarte, Toryn Klassen, Richard Valenzano, and Sheila McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *ICML*, 2018.

- [50] Rodrigo Toro Icarte, Toryn Q Klassen, Richard Valenzano, and Sheila A McIlraith. Reward machines: Exploiting reward function structure in reinforcement learning. *Journal of Artificial Intelligence Research*, 73:173–208, 2022.
- [51] Yuqian Jiang, Suda Bharadwaj, Bo Wu, Rishi Shah, Ufuk Topcu, and Peter Stone. Temporal-logic-based reward shaping for continuing rl tasks. In *AAAI Conf. on AI*, 2021.
- [52] Josip Josifovski, Mohammadhossein Malmir, Noah Klarmann, and Alois Knoll. 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*, pages Nicht-veröffentlichter, 2020.
- [53] Josip Josifovski, Mohammadhossein Malmir, Noah Klarmann, and Alois and Knoll. 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*, page Nicht veröffentlichter Vortrag, 2020. URL <https://sim2real.github.io/assets/papers/2020/josifovski.pdf>.
- [54] Kishor Jothimurugan, Rajeev Alur, and Osbert Bastani. A composable specification language for reinforcement learning tasks. In *Advances in NeurIPS*, volume 32, pages 13041–13051, 2019.
- [55] Kishor Jothimurugan, Suguman Bansal, Osbert Bastani, and Rajeev Alur. Compositional reinforcement learning from logical specifications. *NeurIPS*, 2021.
- [56] Sham Machandranath Kakade. *On the sample complexity of reinforcement learning*. University of London, University College London (United Kingdom), 2003.
- [57] Andrej Karpathy and Michiel Van De Panne. Curriculum learning for motor skills. In *Canadian Conf. on AI*, 2012.

- [58] Manuel Kaspar, Juan David Munoz Osorio, and Jürgen Bock. Sim2real transfer for reinforcement learning without dynamics randomization. *arXiv preprint arXiv:2002.11635*, 2020.
- [59] Tae Kyun Kim. T test as a parametric statistic. *Korean journal of anesthesiology*, 68(6):540–546, 2015.
- [60] George Konidaris, Ilya Scheidwasser, and Andrew G Barto. Transfer in reinforcement learning via shared features. *Journal of Machine Learning Research*, 2012.
- [61] Anil Kurkcu, Domenico Campolo, and Keng Peng Tee. Autonomous curriculum generation for self-learning agents. In *16th Intl Conf. on Control, Automation, Robotics and Vision*, 2020.
- [62] Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. Reward design with language models. *arXiv preprint arXiv:2303.00001*, 2023.
- [63] Adam Daniel Laud. *Theory and application of reward shaping in reinforcement learning*. University of Illinois at Urbana-Champaign, 2004.
- [64] Diana Laurillard. *Teaching as a design science: Building pedagogical patterns for learning and technology*. Routledge, 2013.
- [65] Alessandro Lazaric. Transfer in reinforcement learning: a framework and a survey. In *Reinforcement Learning*, pages 143–173. Springer, 2012.
- [66] Cecilia Le, Rebecca E Wolfe, and Adria Steinberg. The past and the promise: Today’s competency education movement. students at the center: Competency education research series. *Jobs for the Future*, 2014.
- [67] Xiao Li, Cristian-Ioan Vasile, and Calin Belta. Reinforcement learning with temporal logic rewards. In *IROS 2017*, 2017.
- [68] William Liang, Sam Wang, Hung-Ju Wang, Osbert Bastani, Dinesh Jayaraman, and Yecheng Jason Ma. Environment curriculum generation via large language models. In *8th Annual Conference on Robot Learning*, 2024.

- [69] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *ICLR (Poster)*, 2016.
- [70] Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023.
- [71] Yaxin Liu and Peter Stone. Value-function-based transfer for reinforcement learning using structure mapping. In *AAAI*, pages 415–420, 2006.
- [72] Yong Liu, Yujing Hu, Yang Gao, Yingfeng Chen, and Changjie Fan. Value function transfer for deep multi-agent reinforcement learning based on n-step returns. In *IJCAI*, pages 457–463, 2019.
- [73] Pierrick Lorang, Shivam Goel, Yash Shukla, Patrik Zips, and Matthias Scheutz. A framework for neurosymbolic goal-conditioned continual learning in open world environments. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12070–12077. IEEE, 2024.
- [74] Bruce T Lowerre. The harpy speech recognition system. Technical report, Carnegie Mellon University, PA, Department of Computer Science, 1976.
- [75] Abdoulaye O Ly and Moulay Akhloufi. Learning to drive by imitation: An overview of deep behavior cloning methods. *IEEE Transactions on Intelligent Vehicles*, 6(2):195–209, 2020.
- [76] Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher-student curriculum learning. *IEEE Trans. Neural Networks Learn. Syst.*, 31(9):3732–3740, 2020. doi: 10.1109/TNNLS.2019.2934906. URL <https://doi.org/10.1109/TNNLS.2019.2934906>.
- [77] Christoforos Mavrogiannis, Francesca Baldini, Allan Wang, Dapeng Zhao, Pete Trautman, Aaron Steinfeld, and Jean Oh. Core challenges of social robot navi-

- gation: A survey. *ACM Transactions on Human-Robot Interaction*, 12(3):1–39, 2023.
- [78] Rory McGreal, Wanjira Kinuthia, Stewart Marshall, and Tim McNamara. *Open educational resources: Innovation, research and practice*. Commonwealth of Learning, 2013.
- [79] Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, 2021.
- [80] Reuth Mirsky, Shahaf S Shperberg, Yulin Zhang, Zifan Xu, Yuqian Jiang, Jiaxun Cui, and Peter Stone. Task factorization in curriculum learning. In *Decision Awareness in Reinforcement Learning Workshop at ICML 2022*, 2022.
- [81] Mayank Mittal, Calvin Yu, Qinxi Yu, Jingzhou Liu, Nikita Rudin, David Hoeller, Jia Lin Yuan, Ritvik Singh, Yunrong Guo, Hammad Mazhar, Ajay Mandlekar, Buck Babich, Gavriel State, Marco Hutter, and Animesh Garg. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023. doi: 10.1109/LRA.2023.3270034.
- [82] 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, Dhharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 0028-0836, 1476-4687. doi: 10.1038/nature14236. URL <http://www.nature.com/articles/nature14236>.
- [83] Akshay Narayan, Zhuoru Li, and Tze-Yun Leong. Seapot-rl: selective exploration algorithm for policy transfer in rl. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

- [84] Sanmit Narvekar, Jivko Sinapov, and Peter Stone. Autonomous task sequencing for customized curriculum design in reinforcement learning. In *IJCAI*, pages 2536–2542, 2017.
- [85] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *J. of Machine Learning Research*, 21:1–50, 2020.
- [86] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pages 278–287. Citeseer, 1999.
- [87] Allan C. Ornstein and Francis P. Hunkins. *Curriculum: Foundations, principles, and issues*. Pearson Higher Ed, 2017.
- [88] Kei Ota, Devesh K Jha, Siddarth Jain, Bill Yerazunis, Radu Corcodel, Yash Shukla, Antonia Bronars, and Diego Romeres. Autonomous robotic assembly: From part singulation to precise assembly. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13525–13532. IEEE, 2024.
- [89] Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation? a typology of computational approaches. *Front. in neurorobotics*, page 6, 2009.
- [90] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.
- [91] Liam Paull and Anthony Courchesne. On assessing the value of simulation for robotics. In *2nd Workshop on Closing the Reality Gap in Sim2Real Transfer for Robotics. RSS*, 2020.
- [92] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE Intl. Conf. on robotics and automation (ICRA)*, pages 1–8. IEEE, 2018.

- [93] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.
- [94] Óscar Pérez-Gil, Rafael Barea, Elena López-Guillén, Luis M Bergasa, Carlos Gomez-Huelamo, Rodrigo Gutiérrez, and Alejandro Diaz-Diaz. Deep reinforcement learning based control for autonomous vehicles in carla. *Multimedia Tools and Applications*, 81(3):3553–3576, 2022.
- [95] Jean Piaget. Extracts from piaget’s theory. *Manual of child psychology*, pages 703–732, 1970.
- [96] Rey Pocius, David Isele, Mark Roberts, and David Aha. Comparing reward shaping, visual hints, and curriculum learning. In *AAAI Conf. on Artificial Intelligence*, volume 32, 2018.
- [97] Rémy Portelas, Cédric Colas, Katja Hofmann, and Pierre-Yves Oudeyer. Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments. In *Conference on Robot Learning*, pages 835–853. PMLR, 2020.
- [98] Zhiqian Qiao, Katharina Muelling, John M Dolan, Praveen Palanisamy, and Priyantha Mudalige. Automatically generated curriculum based reinforcement learning for autonomous vehicles in urban environment. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1233–1238. IEEE, 2018.
- [99] X Qu, Z Sun, Y S Ong, A Gupta, and P Wei. Minimalistic attacks: How little it takes to fool deep reinforcement learning policies. *IEEE Transactions on Cognitive and Developmental Systems*, 2020.
- [100] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, number 3.2 in 1, page 5. Kobe, Japan, 2009.

- [101] Shreyas Sundara Raman, Vanya Cohen, Eric Rosen, Ifrah Idrees, David Paulius, and Stefanie Tellex. Planning with large language models via corrective re-prompting. *arXiv preprint arXiv:2211.09935*, 2022.
- [102] Kanishka Rao, Chris Harris, Alex Irpan, Sergey Levine, Julian Ibarz, and Mohi Khansari. RL-cyclegan: Reinforcement learning aware simulation-to-real. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11157–11166, 2020.
- [103] Erick Rodríguez-Hernandez, Juan Irving Vasquez-Gomez, and Juan Carlos Herrera-Lozada. Flying through gates using a behavioral cloning approach. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1353–1358. IEEE, 2019.
- [104] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.
- [105] Mohammad Rostami, David Isele, and Eric Eaton. Using task descriptions in lifelong machine learning for improved performance and zero-shot transfer. *Journal of Artificial Intelligence Research*, 67:673–704, 2020.
- [106] Andrei A Rusu, Matej Večerík, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. Sim-to-real robot learning from pixels with progressive nets. In *Conference on robot learning*, pages 262–270. PMLR, 2017.
- [107] Terence D Sanger. Neural network learning control of robot manipulators using gradually increasing task difficulty. *IEEE transactions on Robotics and Automation*, 10(3):323–333, 1994.
- [108] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

- [109] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL <http://arxiv.org/abs/1707.06347>.
- [110] Jinghuan Shang, Karl Schmeckpeper, Brandon B May, Maria Vittoria Minniti, Tarik Kelestemur, David Watkins, and Laura Herlant. Theia: Distilling diverse vision foundation models for robot learning. *arXiv preprint arXiv:2407.20179*, 2024.
- [111] Yash Shukla. Formal and natural language assisted curriculum generation for reinforcement learning agents. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, pages 2776–2778, 2024.
- [112] Yash Shukla and Jivko Sinpov. A framework for curriculum schema transfer from low-fidelity to high-fidelity environments. In *3rd Workshop on Closing the Reality Gap in Sim2Real Transfer for Robotics at RSS*, volume 2022, 2022.
- [113] Yash Shukla, Kaleb Loar, Robert Wright, and Jivko Sinapov. An object-oriented approach for generating low-fidelity environments for curriculum schema transfer. In *Scaling Robot Learning Workshop at ICRA*, volume 2022, page 202, 2022.
- [114] Yash Shukla, Christopher Thierauf, Ramtin Hosseini, Gyan Tatiya, and Jivko Sinapov. Acute: Automatic curriculum transfer from simple to complex environments. In *21st Intl. Conf. on Autonomous Agents and Multiagent Systems*, pages 1192–1200, 2022.
- [115] Yash Shukla, Bharat Kesari, Shivam Goel, Robert Wright, and Jivko Sinapov. A framework for few-shot policy transfer through observation mapping and behavior cloning. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7104–7110. IEEE, 2023.
- [116] Yash Shukla, Abhishek Kulkarni, Robert Wright, Alvaro Velasquez, and Jivko Sinapov. Automaton-guided curriculum generation for reinforcement learning agents. In *Proc. of the 33rd Intl. Conf. on Automated Planning and Scheduling*, 2023.

- [117] Yash Shukla, Tanushree Burman, Abhishek N Kulkarni, Robert Wright, Alvaro Velasquez, and Jivko Sinapov. Logical specifications-guided dynamic task sampling for reinforcement learning agents. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 34, pages 532–540, 2024.
- [118] Yash Shukla, Wenchang Gao, Vasanth Sarathy, Alvaro Velasquez, Robert Wright, and Jivko Sinapov. Lgts: Dynamic task sampling using llm-generated sub-goals for reinforcement learning agents. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, pages 1736–1744, 2024.
- [119] Simon Buckingham Shum and Rebecca Ferguson. Social learning analytics. *Journal of educational technology & society*, 15(3):3–26, 2012.
- [120] Felipe Leno Da Silva and Anna Helena Reali Costa. Object-oriented curriculum generation for reinforcement learning. In *Proc.of the 17th Intl Conf. on Autonomous Agents and MultiAgent Systems*, pages 1026–1034, 2018.
- [121] Jivko Sinapov, Sanmit Narvekar, Matteo Leonetti, and Peter Stone. Learning inter-task transferability in the absence of target task samples. In *Proc.of the 2015 Intl Conf. on Autonomous Agents and Multiagent Systems*, pages 725–733, 2015.
- [122] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2998–3009, 2023.
- [123] James Staley, Elaine Short, Shivam Goel, and Yash Shukla. Agent-centric human demonstrations train world models. In *Reinforcement Learning Conference*, 2024.
- [124] Austin Stone, Ted Xiao, Yao Lu, Keerthana Gopalakrishnan, Kuang-Huei Lee, Quan Vuong, Paul Wohlhart, Brianna Zitkovich, Fei Xia, Chelsea Finn, et al. Open-world object manipulation using pre-trained vision-language models. *arXiv preprint arXiv:2303.00905*, 2023.

- [125] Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. In *ICLR*, 2018.
- [126] Maxwell Svetlik, Matteo Leonetti, Jivko Sinapov, Rishi Shah, Nick Walker, and Peter Stone. Automatic curriculum graph generation for reinforcement learning agents. In *AAAI*, pages 2590–2596, 2017.
- [127] Gyan Tatiya, Yash Shukla, Michael Edegware, and Jivko Sinapov. Haptic knowledge transfer between heterogeneous robots using kernel manifold alignment. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5358–5363. IEEE, 2020.
- [128] Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7), 2009.
- [129] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. 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)*, pages 23–30. IEEE, 2017.
- [130] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ Intl Conf. on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.
- [131] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4950–4957, 2018.
- [132] Rodrigo Toro Icarte, Toryn Q Klassen, Richard Valenzano, and Sheila A McIlraith. Teaching multiple tasks to an rl agent using ltl. In *Autonomous Agents and MultiAgent Systems*, 2018.
- [133] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield.

- Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 969–977, 2018.
- [134] Ralph W Tyler. Basic principles of curriculum and instruction. In *Curriculum studies reader E2*, pages 60–68. Routledge, 2013.
- [135] Alvaro Velasquez, Brett Bissey, Lior Barak, Andre Beckus, Ismail Alkhouri, Daniel Melcer, and George Atia. Dynamic automaton-guided reward shaping for monte carlo tree search. In *Proc. of the AAAI Conf. on Artificial Intelligence*, 2021.
- [136] Álvaro Visús, Javier García, and Fernando Fernández. A taxonomy of similarity metrics for markov decision processes. *arXiv preprint arXiv:2103.04706*, 2021.
- [137] Daphna Weinshall, Gad Cohen, and Dan Amir. Curriculum learning by transfer learning: Theory and experiments with deep networks. In *ICML*, 2018.
- [138] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.
- [139] David Wood, Jerome S Bruner, and Gail Ross. The role of tutoring in problem solving. *Journal of child psychology and psychiatry*, 17(2):89–100, 1976.
- [140] Yuechen Wu, Wei Zhang, and Ke Song. Master-slave curriculum design for reinforcement learning. In *IJCAI*, pages 1523–1529, 2018.
- [141] Zhe Xu and Ufuk Topcu. Transfer of temporal logic formulas in reinforcement learning. In *IJCAI: Proc. of the Conf.*, volume 28, page 4010. NIH Public Access, 2019.
- [142] Yunzhi Zhang, Pieter Abbeel, and Lerrel Pinto. Asymmetric self-play for automatic goal discovery in robotic manipulation. *Advances in Neural Information Processing Systems Deep Reinforcement Learning Workshop*, 2020.

- [143] Yunzhi Zhang, Pieter Abbeel, and Lerrel Pinto. Automatic curriculum learning through value disagreement. *Advances in Neural Information Processing Systems*, 33, 2020.
- [144] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.
- [145] Zhuangdi Zhu, Kaixiang Lin, and Jiayu Zhou. Transfer learning in deep reinforcement learning: A survey. *arXiv preprint arXiv:2009.07888*, 2020.